Contents

List of Figures

List of Tables

Acknowledgements

We would like to extend our thanks to

- Guy Power who made our stay at NASA possible by fighting the system and convincing people of our noble intentions.
- Dr. Kevin Wheeler who has been a true inspiration to us during the entire process as a very mature and extraordinary supervisor. Thank you for many good and long discussions and for being a good friend and helping us in all matters.
- Dr. Chuck Jorgensen for inviting us to your home and making our stay unforgettable.
- Everybody at NEL for their hospitality and for taking good care of us
- And a special thanks to our sponsors who made this project financially possible:
  - Civilingeniør Frants Allings fond
  - Vera og Carl Johan Michaelsens fond
  - Julie Damms Studiefond
  - Alexandre Haynman og hustru Nina Haynmans fond
  - Incentive Fonden
  - Ingeniørforeningen i Danmark (IDA)
  - Familien Hede Nielsens Fond
  - Danmarks Tekniske Universitet (IMM, Internationalt kontor & Rektors rejsedispositionsbeløb)
  - RS Components A/S
  - Oticon Fonden
  - Institut for Informatik & Matematisk Modelering (IMM), DTU
  - Knud Højgaards Fond

Dedications

Morten Andersen:

Dedicated to my family and my girlfriend Hanne.
I missed y’all and your visits were unforgettable!

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science at The technical University of Denmark (DTU), Lyngby, Denmark.

Authors are students Morten Nonboe Andersen (c971828) and Rasmus Ørum Andersen (c971596).

Thesis supervisor is Prof. Lars Kai Hansen, Dept. of Informatics and Mathematical Modeling, DTU and assistant supervisor is Dr. Kevin Wheeler, NASA Ames Research Center, California, USA.

Project work was done at NASA Ames Research Center, California, USA from Oct. 2002 - April 2003.

Resume

I dette eksamensprojekt prœsenteres the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), generic Particle Filter (PF, a.k.a. condensation, survival of the fittest, bootstrap filter, SIR, sequential Monte Carlo, etc.), Particle Filter med MCMC steps (PFMC), Particle Filter med EKF proposal (PFEKF) og MCMC steps (PFEKFMC), Particle Filter med UKF proposal (PFUKF) og MCMC steps (PFUKFMC).

En teoretisk gennemgang af filtrene afsluttes med opskrivning af pseudo-koden (fra [26]) for en implementering af det enkelte filter.

Desuden implementeres de forskellige filtre i et Dynamisk Bayesiansk netvœrks (DBN) framework vha. Matlab og vi demonstrerer og sammenligner teknikkerne vha. et simpelt en-dimensionalt state estimations problem og en større og mere kompleks simulation af et vandtankssystem. Endelig anvendes udvalgte filtre på et problem fra den virkelige verden, hvor vi anvender en cyberglove til at inferere vinkel, vinkelhastighed og vinkelacceleration for et enkelt fingerled i bevœgelse og anvender disse variable som skjulte knuder i et 2T-DBN med EMG målinger fra underarmen som observationer.

I rapporten vises, hvorledes de enkelte filtre adskiller sig fra hinanden i såvel teori som i praksis, og hvornår og hvordan deres styrker og svagheder kommer til udtryk. Desuden konkluderes hvilke filtre, der er mere eller mindre anvendelige i forskellige praktiske scenarier og under forskellige forudsœtninger.

Teori og implementering er baseret p teorien og pseudo-koden i [26].

Keywords: Hybrid Bayesian Networks, Dynamic Bayesian Networks, Particle Filtering, Extended Kalman Filter, Unscented Kalman Filter, Markov Chain Monte Carlo

Abstract

In this thesis we describe the use of the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), generic Particle Filter (PF, a.k.a. condensation, survival of the fittest, bootstrap filter, SIR, sequential Monte Carlo, etc.), Particle Filter with MCMC steps (PFMC), Particle Filter with EKF proposal (PFEKF) and MCMC steps (PFEKFMC), Particle Filter with UKF proposal (PFUKF) and MCMC steps (PFUKFMC) in theory as well as in a practical framework.

We present pseudo-code (from [26]) for all algorithms and implement the filters in a Dynamic Bayesian Network (DBN) framework using Matlab. Furthermore, we demonstrate and compare the implementations on a simple one-dimensional state estimation problem, a more complex simulation of a watertank system and finally on a real-life problem in which we use a cyberglove to infer the angle, angular velocity and angular acceleration of a single fingerjoint during movement and use these variables as hidden nodes in a 2T-DBN with EMG measurements from the lower arm as observations.

Furthermore, we show how the filters differ theoretically as well as practically and when and how their strengths and weaknesses become visual. Finally, we conclude which filters are superior under different conditions and in different practical scenarios.

Theory and implementation is based on the theory and pseudo-code presented in [26].

Keywords: Hybrid Bayesian Networks, Dynamic Bayesian Networks, Particle Filtering, Extended Kalman Filter, Unscented Kalman Filter, Markov Chain Monte Carlo

Chapter 1

Introduction

Dealing with uncertainty is a common phenomenon in everyday life, e.g. trying to predict the weather forecast etc. Probability theory provides us with clear semantics for maintaining our beliefs giving only partial information and how to update our beliefs if new evidence arrives. However, probabilistic models has not had the impact in the AI community as one might expect, perhaps as a consequence of the lack of ways to represent probability distributions and ways to reason with them.

In a combined approach to dealing with uncertainty, probability and graph theory joins forces in *Bayesian networks* (BN) [Pea88] exploring properties of both worlds. In a Bayesian network, probability distributions are represented in a graphical model as a Directed Acyclic Graph (DAG). Nodes in the graph corresponds to random variables connected via arcs representing Conditional Probability Distributions (CPD). A Bayesian network is an intuitively plausible and compact way of representing probability distributions and have been used in a variety of problem domains, expanding from fault diagnosis systems, medical expert systems and Microsoft Windows troubleshooting assistance © Microsoft, see http://freelock.com/technical/KE.pdf for details. However, a lot of published literature is very sparse and often the described problems are very simple, e.g. much of the work has been focused on discrete BN's, where all nodes can take on only discrete values.

One of the main issues in BN's is the problem of inference. As an example, one might like to know the probability of having a cold giving that one has observed a running nose, fever and headache, i.e. calculating the probability distribution of a random variable X representing 'cold' given some observations or evidence

$$P(X|E = e) \qquad \text{e = (running nose, fever, headache)} \tag{1.1}$$

Inference may be divided into two broad categories: exact and approximate inference. The former giving an exact answer to a probabilistic query whereas the latter algorithms only provide an approximate answer. In many applications, approximate inference is adequate and as exact inference is often intractable, approximate inference is a very attractive research area allowing for much more complex models to be investigated.

In this work we focus on the use of approximate inference.

In most real life applications, systems involve combinations of discrete and continuous attributes. In the medical domain, blood pressure, skin conductance and temperature are just a few examples of continuous observations.

In this project we keep all observations in their 'true' domain leading to models with both discrete and continuous valued nodes, so called *Hybrid Bayesian Networks* (HBN). Hybrid models may also be divided into submodels. First, models are labelled according to their CPD's, depending on whether the models are restricted to linear relations between continuous relations or if they allow non-linear relations. An example of the former is a temperature sensor that is a linear combination of the actual temperature and some colored noise

$$\textit{sensor} = \textit{temperature} + \textit{W} \qquad W \in N(0, \sigma^2) \tag{1.2}$$

Secondly, models may or may not allow continuous parents to have discrete children. All continuous observations can be mapped into a finite set of discrete values e.g. using a softmax distribution, but this approach is quite difficult and artificial and the error introduced often leads to poor performance.

In this work, we investigate models having linear as well as non-linear relations, but do not allow continuous parents to have discrete children.

Finally, models can exhibit either static or temporal behavior depending on whether the relations between network variables vary over time. Such temporal models, denoted *Dynamic Bayesian Networks* (DBN, [6]) allow for much more complex scenarios reflecting real-life behavior. As an example, the effects of a drug injected into a patient may change over time or change if the patient eats, falls asleep etc.

In this work we assume a Markovian, stationary model and setup a 2 Time-slice Dynamic Bayesian Network (2T-DBN) in which nodes given at time t is dependent only on other variables at time t and $t-1$.

As an outline, our objective is to give the reader an easy-to-read overview of the involved theory.

From an experimental point of view, the first major contribution of this paper is an introduction to a practical realization of the different filtering techniques through a very simply simulation of a one-dimensional state estimation problem in which the issues treated in the theory are visualized.

Next, the second major contribution follows in which we model a complicated physical system by a hybrid 2T-DBN. In the system, two watertanks, in which water is flowing into one of the tanks from an external source, are interconnected via pipes and also has pipes leading the water away from each tank. In this simulation we apply different filtering techniques in an attempt to setup a simulated fault detection system to report the state of the system - especially to report if any of the pipes have started drifting or has bursted or if the measurement equipment is failing. We thoroughly compare the different implementations to expose advantages and weaknesses of each implementation in an attempt to understand their validity in a real-life simulation.

The final major contribution is a real-life application in which we attempt to track the angle, angular velocity and angular acceleration of finger movements using EMG signals from surface electrodes placed on the lower arm and a cyberglove that measures the ground truth. This is a very interesting application as one would be able to interact with a virtual environment which could have an enormous impact on the computer science society. Today a great number of people are suffering from injuries caused by repeated use of a standard keyboard, mouse and joystick etc. which puts a lot of stress on joints and muscles.

The thesis is structured as follows:

First, we introduce the basic theory of Bayesian networks and Hybrid Bayesian networks. Next, we introduce the filtering concept and describe the 8 different filtering techniques.

In the experimental part we start off with the simple one-dimensional simulation in which we demonstrate characteristics of the different implementations. Then we move on to a detailed treatment of the more complex watertank problem and end our work with a presentation of our real-life application.

Chapter 2

Bayesian Networks

2.1 Notation

$P(X)$ denotes the probability distribution of the random variable (RV) X and $P(X = x)$ or $P(x)$ is the probability that X takes on the value x. Furthermore, $P(X, Y)$ is the joint distribution of X and Y, $P(X|Y)$ is the conditional distribution of X given Y and $P(X|y)$ the conditional distribution of X given $Y = y$.

2.2 Representation

As mentioned, Bayesian networks form a joint probability distribution over a set of random variables **X** using

- A directed acyclic graph G with nodes corresponding to RV's $X_i \in \mathbf{X}$
- A set of Conditional Probability Distributions, one for each node in G, represented graphically by directed arcs between nodes

Thus, each node X_i depends directly upon its parents $Pa(X_i)$ as parameterized in the CPD for X_i.

As such, every node is independent of its non-descendants given its parents. These conditional independencies are captured by the graphical structure and are easily identified.

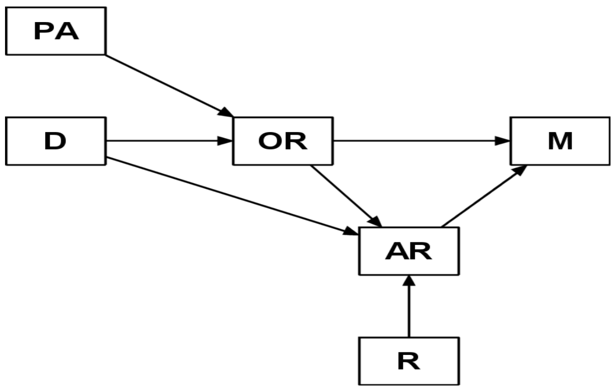


Figure 2.1: Example of simple Bayesian network illustrating a model in which a bachelor is dating single women. The variables or nodes are Date (D): Excellent, Average, Disaster; Physical Attraction (PA): True, False; Offer Rose (OR): True, False; Accept Rose (AR): True, False; Rich (R): True, False; Married (M): True, False

The joint distribution is computed via the Chain Rule. For a BN with n nodes $X_1, ..., X_n$ we have

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i)) \tag{2.1}$$

A simple example is depicted in figure 2.1.

In this network, a bachelor is dating 25 women. After each date he has to offer a rose to the woman if he wants to keep seeing her. If she accepts the rose, they go on another date, but if she rejects the rose or is not offered one, she leaves. In the end, the bachelor may choose to propose to the woman who makes it to the final date. Two factors influence the bachelors decision on whether or not to offer a girl a rose: The *Date (D)* with the girl, which may have been Excellent, Average or a Disaster and his *Physical Attraction (PA)* to her, which is either present or not. Our Bachelor is assumed to increase his desire to *Offer Rose (OR)* if the date went well and also if he is physically attracted to her. Similarly, the women are more likely to *Accept Rose (AR)* if the date went well, but their decision is positively affected if the Bachelor is *Rich (R)* rather than by physical attraction. Finally, the chances of *Marriage (M)* are non-zero if and only if the woman is offered a rose and accepts it. Note that the probability of *Marriage (M)* is not independent of *Rich (R)* and *Physical attraction (PA)*, but it *is* independent of *Rich (R)* and *Physical Attraction (PA)* given its parents, *Accept Rose (AR)* and *Offer Rose (OR)*. Furthermore, *Rich (R)* and *Physical Attraction (PA)* are independent, which is a fair assumption.

The CPD's parameterize the probability distribution, e.g. $P(R = True) = 0.8$ and $P(AR = True|Rich = False) = 0.1$.

Inference in the network corresponds to answering different queries, say, what is the probability of marriage and a disastrous date, that the Bachelor is rich, is attracted to the woman and offers her a rose, but she rejects it?

This query may be answered using the Chain Rule:

$$
\begin{aligned}
&P(D = Disaster, R = True, PA = True, OR = True, AR = False, M = True)\\
&= P(D = Disaster) \cdot P(R = True) \cdot P(PA = True)\cdot\\
&P(OR = True|D = Disaster, PA = True)\cdot\\
&P(AR = False|D = Disaster, R = True, OR = True)\cdot\\
&P(M = True|OR = True, AR = False)\\
&= \ldots
\end{aligned}
$$

A less specific query could be the probability of marriage and a disaster date:

$$
\begin{aligned}
&P(D = Disaster, M = True)\\
&= \textstyle\sum_{r,pa,or,ar} P(D = Disaster, R = r, PA = pa, OR = or, AR = ar, M = True)\\
&= \textstyle\sum_{r,pa,or,ar} P(D = Disaster) \cdot P(R = r) \cdot P(PA = pa)\cdot\\
&P(OR = or|D = Disaster, PA = pa)\cdot\\
&P(AR = ar|D = Disaster, R = r, OR = or)\cdot\\
&P(M = True|OR = or, AR = ar)
\end{aligned}
$$

with r, pa, or and ar either $True$ or $False$. Note that although this is a very simple and discrete valued network, this computation requires summing over 16 possible events. Hence, inference becomes very computationally challenging in large and complex networks.

Having computed the joint distribution we are able to infer the conditional distribution using Bayes theorem:

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \tag{2.2}$$

In our example this would yield

$$P(M = True|D = Disaster) = \frac{P(D = Disaster, M = True)}{P(D = Disaster)} \tag{2.3}$$

2.3 Inference

Inference in a BN over the variables $\boldsymbol{X}$ typically involves computing the distribution over some *query variables* $\boldsymbol{Q}$ given some *evidence* or *observations* $\boldsymbol{E} = \boldsymbol{e}$, i.e. $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e}), \boldsymbol{Q}, \boldsymbol{E} \subseteq \boldsymbol{X}$. As the conditional distribution can be expressed as the ratio of the two marginals (the joint distribution and $P(\boldsymbol{E} = \boldsymbol{e})$), we take as the simplest case $\boldsymbol{E} = \emptyset$. However, even then inference is NP-hard.

Theorem 2.1 *Given a Bayesian network over* **X** *and some variables* $\mathbf{Q} \subseteq \mathbf{X}$ *then in general computing* $P(\mathbf{Q})$ *is NP-hard, even if* $|\mathbf{Q}| = 1$

[4]

Fortunately, we can take advantage of the structure or the parameters of the Bayesian network making exact or approximate inference possible.

In the exact case, techniques s.a. *variable elimination* or the *Junction Tree* algorithm which is a more advanced technique *based on* variable elimination exist, e.g. [23], sec. 2.3 and 2.4. However, when the factors involved in the variable elimination of the Junction Tree algorithm become too large to handle efficiently due to the many parameters involved in large and/or complex networks, e.g. using both discrete and continuous valued nodes, approximate inference techniques become an attractive alternative. Even though in general approximate inference in Bayesian networks is NP-hard ([7]), there are many important cases in which there exist efficient approximate inference algorithms that lead to provable good approximations.

In this work we focus on *sampling* based techniques. The basic idea of these algorithms is to randomly assign values to the random variables (samples) and estimate properties of the joint distribution using these samples. We will discuss these techniques in details in section A.4.

Chapter 3

Hybrid Bayesian Networks

As mentioned, most real-life applications involve both discrete and continuous variables. Bayesian networks allowing both kind of node domains are known as *Hybrid Bayesian Networks (HBN)*. Initially, we will review the properties of the normal distribution which forms the basis of our (and many other) models.

3.1 The normal distribution

The family of normal or Gaussian distributions is very common and popular for several reasons:

- If $X_i, ..., X_n$ are i.i.d. random variables and $Y = \sum_i X_i$, the distribution of Y converges to a normal distribution for $n \to \infty$ (under some weak technical conditions)
- Normal distributions arise naturally in many real-life applications
- The mathematical theory is simple and tractable, e.g. the normal distribution is closed under operations s.a. summation, multiplication and conditioning

In the univariate case the normal distribution is characterized by the parameters μ and σ^2 yielding the density function of a random variable X:

$$P(X) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})} \tag{3.1}$$

i.e. X follows a normal distribution with mean μ and variance σ^2. The mean parameter is responsible for the location of the Gaussian (the value where the Gaussian obtains its maximum) and the variance parameter expresses how peaked the Gaussian is: the larger the variance, the less peaked it is. Formally, these parameters corresponds to the first two moments, i.e. $\mu = E[X]$ and $\sigma^2 = E[X^2] - E[X]^2$.

In the multivariate case, the normal distribution has a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$ as parameters:

$$P(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)} \tag{3.2}$$

For $\boldsymbol{X} = X_1, ..., X_n$, $\boldsymbol{\mu}$ is a vector of length n and $\boldsymbol{\Sigma}$ is a symmetric positive-definite matrix of size $n \times n$ with $\Sigma_{i,i}$ being the variance of X_i and $\Sigma_{i,j} = \Sigma_{j,i}, (i \neq j)$ being the covariance between X_i and X_j.

Next, we outline some of the important properties of the normal distribution, for more properties and proof of theorems, see for example [31], esp. sec. 5.7 and 6.7.

The joint normal distribution over $\boldsymbol{X}, \boldsymbol{Y}$ where $\boldsymbol{X}, \boldsymbol{Y} \in R^n$ we have

$$P(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu_X} \\ \boldsymbol{\mu_Y} \end{pmatrix}, \begin{bmatrix} \boldsymbol{\Sigma_{XX}} & \boldsymbol{\Sigma_{XY}} \\ \boldsymbol{\Sigma_{YX}} & \boldsymbol{\Sigma_{YY}} \end{bmatrix}\right) \tag{3.3}$$

where $\boldsymbol{\mu_X} \in \Re^n, \boldsymbol{\mu_Y} \in \Re^m, \boldsymbol{\Sigma_{XX}}$ is a matrix of size $n \times n$, $\boldsymbol{\Sigma_{XY}}$ is size $n \times m$, $\boldsymbol{\Sigma_{YX}} = \boldsymbol{\Sigma^T_{XY}}$ is size $m \times n$ and $\boldsymbol{\Sigma_{YY}}$ is size $m \times m$.

Theorem 3.1 *Let $\boldsymbol{X}, \boldsymbol{Y}$ have a joint normal distribution as defined in eq. 3.3. The marginal distribution over $\boldsymbol{Y}$ is a normal distribution $\mathcal{N}(\boldsymbol{Y}; \mu_Y, \Sigma_{YY})$*

Theorem 3.2 *Let $\boldsymbol{X} = X_1, ..., X_n$ have a joint normal distribution $\mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$.*
Let $\beta_0 \in \Re, \boldsymbol{\beta} = (\beta_1, ..., \beta_n) \in \Re^n$ where $\boldsymbol{\beta} \neq \boldsymbol{0}$ and let $\sigma_W^2 > 0$.
Define $Y = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{X} + W$ where $P(W) = \mathcal{N}(0, \sigma_W^2)$. Then:

- *The conditional distribution $P(Y|\boldsymbol{x}) \sim \mathcal{N}$*

  $P(Y|\boldsymbol{x}) = \mathcal{N}(Y; \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}, \sigma_W^2)$

- *The distribution of Y is a normal distribution $P(Y) = \mathcal{N}(Y; \mu_Y, \sigma_Y^2)$ where*

  $\mu_Y = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{\mu}$

  $\sigma_Y^2 = \sigma_W^2 + \boldsymbol{\beta}^T \boldsymbol{\Sigma} \boldsymbol{\beta}$

Theorem 3.3 *Let $\boldsymbol{X}, \boldsymbol{Y}$ have a joint normal distribution as defined in eq. 3.3. The conditional distribution $P(\boldsymbol{Y}|\boldsymbol{X})$ is a normal distribution $\mathcal{N}(\boldsymbol{Y}; \boldsymbol{\mu}'_{\boldsymbol{Y}}, \boldsymbol{\Sigma}'_{\boldsymbol{YY}})$ where*

$$\boldsymbol{\mu}'_{\boldsymbol{Y}} = \boldsymbol{\mu}_{\boldsymbol{Y}} + \boldsymbol{\Sigma}_{\boldsymbol{YX}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{XX}}(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{X}})$$

$$\boldsymbol{\Sigma}'_{\boldsymbol{YY}} = \boldsymbol{\Sigma}_{\boldsymbol{YY}} - \boldsymbol{\Sigma}_{\boldsymbol{YX}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{XX}}\boldsymbol{\Sigma}_{\boldsymbol{XY}}$$

Corollary 3.1 *Let $\boldsymbol{X}, Y$ have a joint normal distribution as defined in eq. 3.3. The conditional distribution $P(Y|\boldsymbol{X})$ is a normal distribution $\mathcal{N}(Y; \beta_0 + \boldsymbol{\beta}^T\boldsymbol{X}, \sigma^2)$ where*

$$\beta_0 = \mu_Y - \boldsymbol{\Sigma}_{Y\boldsymbol{X}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{XX}}\boldsymbol{\mu}_{\boldsymbol{X}}$$

$$\beta = \boldsymbol{\Sigma}_{Y\boldsymbol{X}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{XX}}$$

$$\sigma^2 = \Sigma_{YY} - \boldsymbol{\Sigma}_{Y\boldsymbol{X}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{XX}}\boldsymbol{\Sigma}_{\boldsymbol{X}Y}$$

3.2 Linear Gaussians

Corollary 3.1 is a way to convert a multivariate Gaussian distribution into a Bayesian network by ordering the variables $X_1, ..., X_n$ topologically (parents before children). The distribution of a child conditioned on its parents is computed as

$$P(X_i|X_1, ..., X_{i-1}) = \mathcal{N}\left(X_i; \beta_{i,0} + \sum_{j=1}^{i-1} \beta_{i,j}X_j, \sigma_i^2\right) \tag{3.4}$$

An edge from X_j to $X_i (1 \leq j < i)$ corresponds to $\beta_{i,j} \neq 0$.

The CPD of X_i is called a *linear CPD* . If X_i is a root node, the CPD is simply a univariate Gaussian. A BN with linear CPD's is a *linear Gaussian (LG)* .

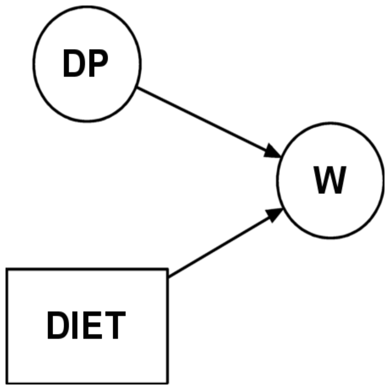


Figure 3.1: Example of simple Bayesian network illustrating how the bodyweight of Homer Simpson is influenced. The variables or nodes are Weight (W), Donut Price (DP) $\sim \mathcal{N}(2, 0.5)$ and Diet (DIET): True, False

3.3 Conditional Linear Gaussians

Conditional Linear Gaussian's (CLG) are Bayesian networks combining discrete variables (denoted Δ) and continuous variables (denoted Γ) and thus encapsulates a broader class of distributions than discrete BN's and LG's.

However, the following restrictions apply:

- A continuous node can not have discrete children (this is theoretically possible, but would involve a discretization, say using a soft-max function, but this approach is artificial and involves discretization error)
- The CPD of a continuous variable is a linear CPD given any combination of its discrete parents. Formally, a continuous node Y with parents $\boldsymbol{X} = X_1, ..., X_k \subseteq \Gamma$ and $\boldsymbol{D} = D_1, ..., D_l \subseteq \Delta$ has a CPD parameterized as:

$$\forall \boldsymbol{d} \in Dom(\boldsymbol{D}), P(Y|\boldsymbol{x}, \boldsymbol{d}) = \mathcal{N}\left(Y; \beta_{d,0} + \sum_{i=1}^{k} \beta_{d,i} x_i, \sigma_d^2\right) \tag{3.5}$$

Hence, if all discrete variables are known, the CPD's of the continuous variables are all linear CPD's and the CLG is reduced to a LG. In other words, a CLG is a mixture of Gaussian's where every mixture component corresponds to an instantiation of the discrete variables.

As a simple example, consider the network given in figure 3.1 which follows the usual notation that discrete variables are depicted as squares or rectangles and continuous variables are depicted as circles or ellipsoids.

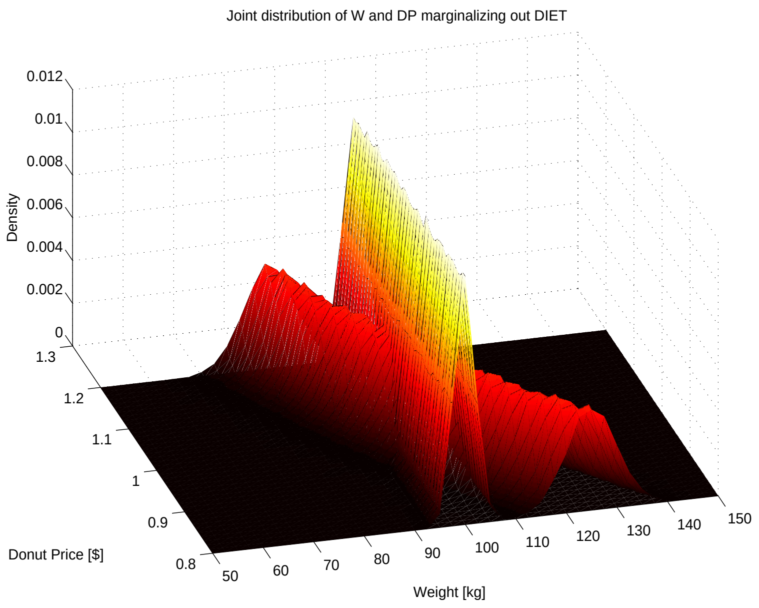


Figure 3.2: Joint distribution of W, DP after marginalizing out $DIET$

In this model, the bodyweight of Homer Simpson (character in the cartoon series *The Simpson*) in kg (continuous variable W) is dependent on the continuous variable DP corresponding to the unit price (in US dollars) on donuts at Apu's (supermarket in *The Simpsons*) (prior distribution $\mathcal{N}(2, 0.5)$) and the discrete variable $DIET$ indicating whether or not Homer is presently on a diet (True with probability 0.01 and False with prob. 0.99).

Homer's bodyweight is modelled using the following CPD:

$$P(W|DIET, DP) = \begin{cases} \mathcal{N}(200 + \frac{1}{DP}, 100) & DIET = FALSE \\ \mathcal{N}(200, 25) & DIET = TRUE \end{cases} \tag{3.6}$$

Hence, Homer's bodyweight is a normal distribution with a mean value inverse proportional to the unit price on donuts and an increased variance if he is not on a diet. Otherwise, his bodyweight is more stable and does not depend on the donut price (which is most unlikely). The joint distribution of W, DP after marginalizing out $DIET$ is a mixture of two Gaussian's and is shown in Figure 3.2. The peaked Gaussian corresponds to $DIET = TRUE$.

Chapter 4

Filtering

This chapter is a summary of [26]. A more detailed version is found in Appendix A.

The original technical report can be downloaded at http://www.cs.ubc.ca/ nando.

In general, filtering is the problem of estimating the state of a system using a set of observations that becomes available on-line. This problem is solved by modelling the evolution of the system and the noise on the measurements. There exist many modelling strategies and filtering algorithms, but the resulting models most often show complex non-linearities and non-Gaussian distributions which rules out analytical solutions.

4.1 Dynamic State Space Model

The general state space model (without control input) consists of a *state transition or state process* model and *a state measurement* model

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{4.1}$$

$$p(\mathbf{y}_t|\mathbf{x}_t) \tag{4.2}$$

where $\mathbf{x}_t \in \Re^{n_x}$ are the states (hidden variables or parameters) of the system at time t and $\mathbf{y}_t \in \Re^{n_y}$ are the observations. The state transitions are a first order Markov process and the observations are assumed to be independent given the states.

For example, a non-linear, non-Gaussian model can be expressed as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{4.3}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{n}_t) \tag{4.4}$$

with $\mathbf{y}_t \in \Re^{n_y}$ being the output observations, $\mathbf{x}_t \in \Re^{n_x}$ the states of the system, $\mathbf{v}_t \in \Re^{n_v}$ the process noise and $\mathbf{n}_t \in \Re^{n_n}$ the measurement noise.

The mappings $f : \Re^{n_x} \times \Re^{n_v}$ and $h : \Re^{n_y} \times \Re^{n_n}$ represent the deterministic process and measurement models and $p(\mathbf{x}_0)$ is the prior distribution at time $t = 0$.

Our goal is to compute the filtering density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ recursively to avoid computing the complete posterior density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Thus we avoid keeping track of the complete history of the states and are still able to compute estimates of the mean, confidence intervals etc. of the systems states.

4.2 Extended Kalman Filter

In the Extended Kalman Filter (EKF) the standard Kalman filter (for linear systems) is applied to non-linear systems with additive white noise by continually updating a linearization around the previous state estimate, starting with an initial guess, i.e. it is a minimum mean-square-error (MMSE) estimator based on the Taylor series expansion of the non-linear functions **f** and **h** around the estimates $\bar{\mathbf{x}}_{t|t-1}$ of the states $\mathbf{x}_t$, e.g.

$$\mathbf{f}(\mathbf{x}_t) = \mathbf{f}(\bar{\mathbf{x}}_{t|t-1}) + \frac{\partial \mathbf{f}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\Big|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}(\mathbf{x}_t - \bar{\mathbf{x}}_{t|t-1}) + \ldots \tag{4.5}$$

Using only the linear expansion terms, the update equations for the mean $\bar{\mathbf{x}}$ and covariance **P** of the Gaussian approximation to the posterior distribution of the states become

$$\begin{aligned}
\bar{\mathbf{x}}_{t|t-1} &= \mathbf{f}(\bar{\mathbf{x}}_{t-1}, 0) &(4.6)\\
\mathbf{P}_{t|t-1} &= \mathbf{F}_t\mathbf{P}_{t-1}\mathbf{F}_t^T + \mathbf{G}_t\mathbf{Q}_t\mathbf{G}_t^T &(4.7)\\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}\mathbf{H}_t^T[\mathbf{U}_t\mathbf{R}_t\mathbf{U}_t^T + \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T]^{-1} &(4.8)\\
\bar{\mathbf{x}}_t &= \bar{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}, 0)) &(4.9)\\
\mathbf{P}_t &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} &(4.10)
\end{aligned}$$

where $\mathbf{K}_t$ is the Kalman gain, **Q** is the variance of the process noise (assumed to be zero-mean Gaussian), **R** is the variance of the measurement noise (assumed to be zero-mean Gaussian), $\mathbf{F}_t \triangleq \frac{\partial \mathbf{f}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\Big|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}$ and $\mathbf{G}_t \triangleq \frac{\partial \mathbf{f}(\mathbf{v}_t)}{\partial \mathbf{v}_t}\Big|_{(\mathbf{v}_t=\bar{\mathbf{v}})}$ are the Jacobians of the process model and $\mathbf{H}_t \triangleq \frac{\partial \mathbf{h}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\Big|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}$ and $\mathbf{U}_t \triangleq \frac{\partial \mathbf{h}(\mathbf{n}_t)}{\partial \mathbf{n}_t}\Big|_{(\mathbf{n}_t=\bar{\mathbf{n}})}$ are the Jacobians of the measurement model.

4.3 Unscented Kalman Filter

As the EKF only uses the first order terms of the Taylor series expansion of the non-linear functions, it may introduce significant errors in the estimations of the posterior distribution of the states. Especially if the models are highly non-linear where the local linearity assumptions do not hold.

The Unscented Kalman Filter (UKF, [20]) is a recursive MMSE estimator that does not approximate the non-linear process and measurement models, but uses the true models and approximates the distribution of the state random variable. The state distribution is still represented by a Gaussian random variable, but by a minimal set of deterministically chosen sample points that completely capture the true mean and covariance of the Gaussian random variable. When this variable is propagated through the true non-linear system, it captures the true mean and covariance to the second order for any non-linearity.

4.3.1 The scaled unscented transformation

To calculate the statistics of a random variable undergoing a non-linear transformation, as required by the UKF, the scaled unscented transformation (SUT) is used. SUT is based on the principle that it is easier to approximate a probability distribution than an arbitrary non-linear function ([19]).

Let **x** be a n_x dimensional random variable propagated through an arbitrary non-linear function **g** to generate **y**

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \tag{4.11}$$

Assume **x** to have mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_x$. The first two moments of **y** are calculated by first deterministically choosing $2n_x + 1$ weighted samples or *sigma points* $S_i = \{W_i, \boldsymbol{\chi}_i\}$ so as to completely capture the true mean and covariance of the prior random variable **x** as follows ([26]):

Let

$$\lambda = \alpha^2(n_x + \kappa) - n_x \tag{4.12}$$

and select the sigma points using

$$\begin{aligned}
\boldsymbol{\chi}_0 &= \bar{\mathbf{x}} \\
\boldsymbol{\chi}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i, \quad i = 1, ..., n_x \\
\boldsymbol{\chi}_i &= \bar{\mathbf{x}} - \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i, \quad i = n_x + 1, ..., 2n_x \\
W_0^{(m)} &= \lambda/(n_x + \lambda) \\
W_0^{(c)} &= \lambda/(n_x + \lambda) + (1 - \alpha^2 + \beta) \\
W_i^{(m)} &= W_i^{(c)} = 1/\{2(n_x + \lambda)\}, \quad i = 1, \ldots, 2n_x
\end{aligned} \tag{4.13}$$

where κ, α and β are positive scaling parameters and $\left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i$ is the i'th row or column of the matrix square root of $(n_x+\lambda)\mathbf{P}_x$ and W_i is the weight associated with the i'th point s.t. $\sum_{i=0}^{2n_x} W_i = 1$. The sigma points are propagated through the non-linear function

$$\boldsymbol{Y}_i = g(\boldsymbol{\chi}_i), \quad i = 0, ..., 2n_x \tag{4.14}$$

The estimated mean and covariance of **y** is

$$\overline{\boldsymbol{y}} = \sum_{i=0}^{2n_x} W_i \boldsymbol{Y}_i \tag{4.15}$$

$$\boldsymbol{P}_y = \sum_{i=0}^{2n_x} W_i \left(\boldsymbol{Y}_i - \overline{\boldsymbol{y}}\right)\left(\boldsymbol{Y}_i - \overline{\boldsymbol{y}}\right)^T \tag{4.16}$$

which are accurate to the second order (third order for Gaussian priors) of the Taylor series expansion of $\mathbf{g}(\mathbf{x})$ for any non-linear function ([26]).

4.3.2 Implementation

In the Unscented Kalman Filter (UKF), the SUT is applied to the state random variable defined as the concatenation of the original state and noise variables, i.e. $\mathbf{x}_t^a = \left[\mathbf{x}_t^T \mathbf{v}_t^T \mathbf{n}_t^T\right]^T$ yielding a sigma point matrix $\boldsymbol{\chi}_t^a$. The complete pseudo algorithm is given below

1. Initialization

$$\bar{\mathbf{x}}_0 = E[\mathbf{x}_0] \tag{4.17}$$

$$\mathbf{P}_0 = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T] \tag{4.18}$$

$$\bar{\mathbf{x}}_0^a = E[\mathbf{x}^a] = [\bar{\mathbf{x}}_0^T \mathbf{0}\ \mathbf{0}]^T \tag{4.19}$$

$$\mathbf{P}_0^a = E[(\mathbf{x}_0^a - \bar{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \bar{\mathbf{x}}_0^a)^T] = \begin{pmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{pmatrix} \tag{4.20}$$

2. For $t \in \{1, \ldots, \infty\}$

   (a) Calculate sigma points

$$\boldsymbol{\chi}^a_{t-1} = \left[\begin{array}{cc} \bar{\mathbf{x}}^a_{t-1} & \bar{\mathbf{x}}^a_{t-1} \pm \sqrt{(n_a + \lambda)\mathbf{P}^a_{t-1}} \end{array} \right] \tag{4.21}$$

   (b) Time update

$$\boldsymbol{\chi}^x_{t|t-1} = \mathbf{f}(\boldsymbol{\chi}^x_{t-1}, \boldsymbol{\chi}^v_{t-1}) \tag{4.22}$$

$$\bar{\mathbf{x}}_{t|t-1} = \sum_{i=0}^{2n_a} W_i^{(m)} \chi^x_{i,t|t-1} \tag{4.23}$$

$$\mathbf{P}_{t|t-1} = \sum_{i=0}^{2n_a} W_i^{(c)} [\chi^x_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}][\chi^x_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}]^T \tag{4.24}$$

$$\boldsymbol{\gamma}_{t|t-1} = \mathbf{h}(\boldsymbol{\chi}^x_{t|t-1}, \boldsymbol{\chi}^n_{t-1}) \tag{4.25}$$

$$\bar{\mathbf{y}}_{t|t-1} = \sum_{i=0}^{2n_a} W_i^{(m)} \gamma^v_{i,t|t-1} \tag{4.26}$$

   (c) Measurement update

$$\mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} = \sum_{i=0}^{2n_a} W_i^{(c)} [\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}][\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}]^T \tag{4.27}$$

$$\mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} = \sum_{i=0}^{2n_a} W_i^{(c)} [\chi_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}][\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}]^T \tag{4.28}$$

$$\mathbf{K}_t = \mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} \mathbf{P}^{-1}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} \tag{4.29}$$

$$\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1}) \tag{4.30}$$

$$\mathbf{P}_t = \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} \mathbf{K}_t^T \tag{4.31}$$

where $\mathbf{x}^a = \left[\begin{array}{ccc} \mathbf{x}^T & \mathbf{v}^T & \mathbf{n}^T \end{array} \right]^T$, $\boldsymbol{\chi}^a = \left[\begin{array}{ccc} (\boldsymbol{\chi}^x)^T & (\boldsymbol{\chi}^v)^T & (\boldsymbol{\chi}^n)^T \end{array} \right]^T$, λ is the composite scaling parameter, $n_a = n_x + n_v + n_n$, **Q** is the process noise covariance, **R** is the measurement noise covariance, **K** is the Kalman gain and W_i are the weights.

From a computational perspective, the UKF is superior to EKF, as it does not require explicit calculation of Jacobians (or Hessians), but computes a covariance matrix square root which can be done using a Cholesky factorization in order $n_x^3/6$. However, by expressing the covariance matrices recursively, this can be done in order n_x^2 using a recursive update to the Cholesky factorization ([26]).

4.4 Particle Filtering

EKF and UKF both rely on a Gaussian approximation. In this section we present a method that does not require this assumption, but presents other problematic issues. To overcome some of these problems, the particle filtering strategy is combined with EKF and UKF in section 4.5 and 4.6.1 resp.

4.4.1 Monte Carlo simulation

In Monte Carlo simulation, the posterior distribution is approximated by an empirical estimate computed using a set of N weighted particles (samples) $\{\mathbf{x}_{0:t}^{(i)}; i = 1, \dots, N\}$ drawn from the posterior distribution

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{1=1}^{N} \delta_{\boldsymbol{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{4.32}$$

where $\delta(\cdot)$ denotes the Dirac delta function. Hence, the expectation

$$\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) = \int \mathbf{g}_t(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \tag{4.33}$$

is approximated by

$$\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}_t(\mathbf{x}_{0:t}^{(i)}) \tag{4.34}$$

where the particles $\mathbf{x}_{0:t}^{(i)}$ are assumed to be i.i.d.

4.4.2 Bayesian importance sampling

However, as it is often impossible to sample from the posterior density, we sample from a known, easy-to-sample, proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ and use the following substitution

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\
&= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\
&= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{\omega_t(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \qquad (4.35)
\end{aligned}$$

where $\omega_t(\mathbf{x}_{0:t})$ are the un-normalized weights

$$\omega_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \tag{4.36}$$

The unknown normalizing density $p(\mathbf{y}_{1:t})$ is removed as follows

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \frac{1}{p(\mathbf{y}_{1:t})}\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\
&= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})\frac{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}d\mathbf{x}_{0:t}} \\
&= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int \omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}} \\
&= \frac{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t})\mathbf{g}_t(\mathbf{x}_{0:t}))}{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t}))} \qquad (4.37)
\end{aligned}$$

where $\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}$ means expectation over the proposal distribution $q(\cdot|\mathbf{y}_{1:t})$.

Now, the expectation is approximated by

$$\begin{aligned}
\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} &= \frac{1/N\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)})\omega_t^{(i)}(\mathbf{x}_{0:t}^{(i)})}{1/N\sum_{i=1}^{N}\omega_t^{(i)}(\mathbf{x}_{0:t}^{(i)})} \\
&= \sum_{i=1}^{N}\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)})\tilde{\omega}_t(\mathbf{x}_{0:t}^{(i)}) \qquad (4.38)
\end{aligned}$$

where the normalized importance weights $\tilde{w}_t^{(i)}$ are given by

$$\tilde{\omega}_t = \frac{\omega_t^{(i)}}{\sum_{j=1}^{N}\omega_t^{(i)}} \qquad (4.39)$$

4.4.3 Sequential importance sampling

In this paper our goal is to perform filtering on the given models, i.e. to compute a sequential estimate of the posterior distribution at time t without modifying the previously simulated states $\mathbf{x}_{0:t-1}$ allowing proposal distributions of the form

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) \qquad (4.40)$$

Assuming the states follow a Markov process and that the observations are conditionally independent given the states yields

$$p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0)\prod_{j=1}^{t}p(\mathbf{x}_j|\mathbf{x}_{j-1}) \quad \text{and} \quad p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) = \prod_{j=1}^{t}p(\mathbf{y}_j|\mathbf{x}_j) \qquad (4.41)$$

By substituting (4.41) into (4.36) we get a recursive estimate for the importance weights

$$\begin{aligned}\omega_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \\ &= \omega_{t-1}\frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{x}_{0:t-1})p(\mathbf{x}_{0:t-1})}\frac{1}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \\ &= \omega_{t-1}\frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})} \end{aligned} \tag{4.42}$$

Now, given a proposal distribution and a set of prior samples, we are able to sequentially sample and evaluate likelihood, transition probabilities and importance weights leading to estimates such as (4.35).

The transition prior

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) \mathrel{\hat{=}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{4.43}$$

is the most popular choice of proposal distribution (even though it gives higher variance as it does not include the most recent observations) simply because it is easier to implement. For an additive Gaussian process noise model the transition prior simplifies to

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(f(\mathbf{x}_{t-1},0),\mathbf{Q}_{t-1}\right) \tag{4.44}$$

4.4.4 Degeneracy

Unfortunately, in Sequential Importance Sampling (SIS) the variance of the importance weights increases stochastically over time. This is seen by expanding equation (4.42):

$$\begin{aligned}\omega_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= \frac{p(\mathbf{y}_{1:t},\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &\propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \end{aligned} \tag{4.45}$$

Thus, the importance weights are proportional to the *importance ratio*, which variance increases over time ([10]).

In practice, what happens is that after a few time steps one of the normalized importance weights is close to 1 while the rest is close to 0. In other words, a lot of the samples become useless and are neglected. To avoid this phenomenon, *degeneracy*, the particles need to be resampled (selection step) to eliminate particles with low importance weights and multiply particles with high importance weights.

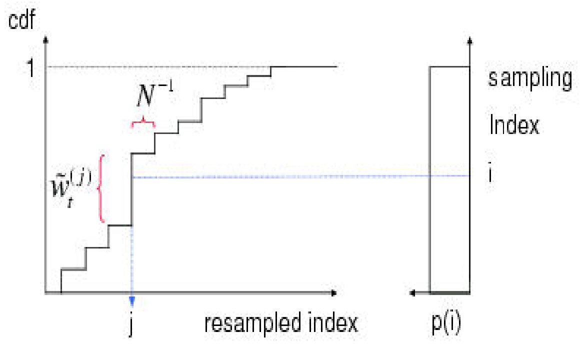


Figure 4.1: Illustration of resampling principle where a random measure $\left\{\mathbf{x}_{1:t}^{(j)}, N^{-1}\right\}$ is mapped into an equally weighted random measure $\left\{\mathbf{x}_{1:t}^{(j)}, N^{-1}\right\}$ by drawing the index i from a uniform distribution (from [26])

4.4.5 Resampling

In resampling, each particle $\mathbf{x}_{0:t}^{(i)}$ is assigned a number of ”children”, say $N_i \in \mathbb{N}$, s.t. $\sum_{i=1}^{N} N_i = N$. In this paper, we have chosen to implement a number of different sampling schemes described in the following sections.

Sampling Importance Resampling (SIR) and multinomial sampling

Resampling involves mapping the Dirac random measure $\left\{\mathbf{x}_{0:t}^{(i)}, \tilde{w}_t^{(i)}\right\}$ into an equally weighted random measure $\left\{\mathbf{x}_{0:t}^{(j)}, N^{-1}\right\}$ by sampling uniformly from the discrete set $\left\{\mathbf{x}_{0:t}^{(i)}, i = 1, \dots, N\right\}$ with probabilities $\left\{\tilde{w}_t^{(i)}; i = 1, \dots, N\right\}$, see [15]. Figure 4.1 illustrates this principle. Having set up the cumulative distribution of the discrete set, a uniformly drawn sampling index i is projected onto the distribution range and then onto the distribution domain. The new sample index j is the intersection with the domain and hence the new sample is the vector $\mathbf{x}_{0:t}^{(i)}$. At the end of the day, the larger importance weighted samples will have more replicates.

Sampling N times from the cumulative discrete distribution $\sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{x_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ corresponds to samples $(N_i; i = 1, \dots, N)$ from a multinomial distribution with parameters N and $\tilde{w}_t^{(i)}$ with a computational complexity of $O(N)$, ([9]). The variance becomes $var(N_i) = N\tilde{w}_t^{(i)}\left(1 - \tilde{w}_t^{(i)}\right)$ as we are sampling from a multinomial distribution.

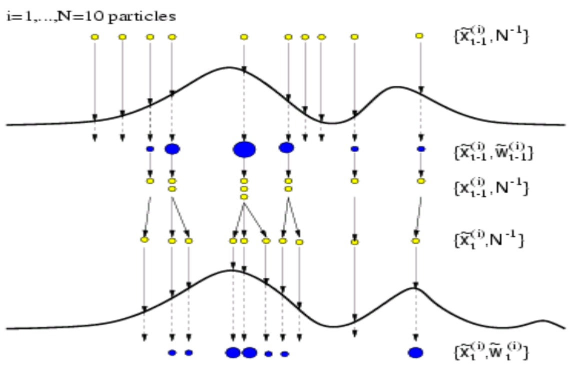

i=1,...,N=10 particles
$\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
$\{x_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t}^{(i)}, \tilde{w}_{t}^{(i)}\}$

Figure 4.2: Illustration of standard particle filter: The filter starts at time $t-1$ with an unweighted measure $\left\{\tilde{\boldsymbol{x}}_{t-1}^{(i)}, N^{-1}\right\}$ providing an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-2})$. For each particle the importance weights are calculated using the information at time $t-1$ giving a weighted measure $\left\{\tilde{\boldsymbol{x}}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$. Next, the resampling step selects the fittest samples to obtain the unweighted measure $\left\{\tilde{\boldsymbol{x}}_{t-1}^{(i)}, N^{-1}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$. Finally, the sampling step introduces variety giving the measure $\left\{\tilde{\boldsymbol{x}}_{t}^{(i)}, N^{-1}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t}|\boldsymbol{y}_{1:t-1})$ (from [26])

Residual resampling

([24] for details). Set $\tilde{N}_i = \lfloor N\tilde{w}_t^{(i)} \rfloor$ and apply the SIR scheme to compute the remaining $\bar{N}_t = N - \sum_{i=1}^{N} \tilde{N}_i$ samples with corresponding weights $w_t^{'(i)} = \bar{N}_t^{-1}\left(\tilde{w}_t^{(i)}N - \tilde{N}_i\right)$.

The variance $\left(var(N_i) = \bar{N}_t\tilde{w}_t^{'(i)}\left(1 - \tilde{w}_t^{'(i)}\right)\right)$ is smaller than for the SIR and also computationally cheaper.

Minimum variance sampling

([5] for details). Sample a set of N points $U \in [0;1]$ with a distance of N^{-1} apart. The number of children N_i is the number of points between $\sum_{j=1}^{i-1} \tilde{w}_t^{(j)}$ and $\sum_{j=1}^{i} \tilde{w}_t^{(j)}$.

The variance of this strategy is $\left(var(N_i) = \bar{N}_t\tilde{w}_t^{'(i)}\left(1 - \bar{N}_t\tilde{w}_t^{'(i)}\right)\right)$ and the computational complexity $O(N)$.

4.4.6 Implementation

Generic Particle Filter

1. Initialization

   - For $i = 1, \ldots, N$, draw the particles $x_0^{(i)}$ from the prior $p(x_0)$

2. For $t = 1, 2, \ldots$

   (a) Importance sampling step

   - For $i = 1, \ldots, N$, sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$ and set $\hat{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \hat{\mathbf{x}}_t^{(i)})$
   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)})p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{4.46}$$

   - For $i = 1, \ldots, N$, normalize the importance weights

$$\tilde{w_t}^{(i)} = w_t^{(i)} \left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \tag{4.47}$$

   (b) Selection step (resampling)

   - Multiply/suppress samples $\hat{\mathbf{x}}_{0:t}^{(i)}$ with high/low importance weights $\tilde{w_t}^{(i)}$, respectively, to obtain N random samples $\mathbf{x}_{0:t}^{(i)}$ approximately distributed according to $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$
   - For $i = 1, \ldots, N$, set $w_t^{(i)} = \tilde{w_t}^{(i)} = \frac{1}{N}$

   (c) Output: A set of samples to approximate the posterior distribution as

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{(x_{0:t}^{(i)})}(d\mathbf{x}_{0:t}) \tag{4.48}$$

4.5 MCMC move step

([12] for details) As mentioned, we are looking for a way of introducing sample variety after the selection step without affecting the validity of the approximation. One strategy is to introduce a MCMC step of invariant distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ on each particle ([2]). If the particles are distributed according to the posterior $p(\tilde{\mathbf{x}}_{0:t}|\mathbf{y}_{1:t})$, and we apply a Markov chain transition kernel $\mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ with invariant distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ s.t. $\int \mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ $p(\tilde{\mathbf{x}}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, we still have a set of particles distributed according to the posterior. However, the particles might have been moved to areas of higher likelihood and the total variance of the current distribution with respect to the invariant distribution can only decrease ([26]).

The MCMC move step corresponds to sampling from the finite mixture distribution $N^{-1}\sum_{i=1}^{N}\mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ (see [14] for convergence issues). This principle can be generalized by applying MCMC steps on the product space with invariant distribution $\prod_{i=1}^{N} p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, i.e. to the entire population of particles. However, in this work we only consider the former and simpler case. For the standard particle filter we sample from the transition prior and accept according to a Metropolis-Hastings (MH) step

Smoothing MH step

- Sample $u \sim U_{[0,1]}$
- Sample the proposal candidate $\mathbf{x}_t^{*(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
- If $u \leq min\left\{1, \frac{p(\mathbf{y}_t|\mathbf{x}_t^{*(i)})}{p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(i)})}\right\}$

  - then accept move

$$\mathbf{x}_{0:t}^{(i)} = (\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t^{*(i)}) \tag{4.49}$$

  - else reject move

$$\mathbf{x}_{0:t}^{(i)} = \tilde{\mathbf{x}}_{0:t}^{(i)} \tag{4.50}$$

  End If

More complex proposals exist, s.a. mixtures of MH steps to ensure an efficient exploration of the sample space ([18]) and reversible jump MCMC steps (see [16]) to allow particles to move from one subspace to other subspaces of, possible, different dimension ([2]).

4.6 Extended Kalman Particle Filter

One way of generating proposal distributions that are more accurate in their approximation of the optimal importance distribution is *local linearization*. This method incorporates the most current observation with the optimal Gaussian approximation of the state ([9]), and is based on the first order Taylor series expansion of the likelihood and transition prior as described in section 4.1 and a Gaussian assumption on all RV's. In this work, the EKF approximates the optimal MMSE estimator of the system state by computing the conditional mean of the state given all observations. This is done recursively through time by propagating the Gaussian approximation of the posterior distribution and combining it with the new observation available at each time step. That is, the EKF computes the recursive approximation of the true posterior filtering density given by

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx p_{\mathcal{N}}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}\left(\bar{\mathbf{x}}_t, \hat{\mathbf{P}}_t\right) \tag{4.51}$$

Using the EKF in particle filtering, a separate EKF is used to generate and propagate a Gaussian proposal distribution for each particle

$$q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) \doteq \mathcal{N}(\mathbf{x}_t|\mathbf{y}_{1:t}) \qquad i = 1, \ldots, N \tag{4.52}$$

i.e. at time $t-1$ the mean and covariance of the importance distribution for each particle are computed using the EKF equations and the new observation. Thus, we need to propagate the covariance $\hat{P}^{(i)}$ and specify the EKF process and measurement noise covariances. Secondly, the i-th particle is sampled from this distribution. This filter is called the *Extended Kalman Particle Filter* and the pseudo-code is given in the following subsection.

As the EKF is an MMSE estimator, this filter leads to an improved annealed sampling algorithm in which the variance of each proposal distribution changes over time. Optimally, the search is based on a large region of the error surface and moved towards regions of lower error as time progresses. However, even though the EKF moves the prior towards the likelihood, we are still faced with the Gaussian assumption on the form of the posterior and linearization approximations. Comparing equation (4.51) with the Gaussian transition prior in equation (4.44), it is noted that the proposal distribution generated by the EKF includes the most current observation at time t. In general though, the true form of this density will *not* be Gaussian - even with Gaussian process and measurement noise - which can be shown using a Bayes rule expansion of the proposal distribution. This implies that we are left with an experimental judgement of the gain versus the loss of filter performance.

4.6.1 Implementation

Extended Kalman Particle Filter

1. Initialization
   - For $i = 1, \ldots, N$, draw the particles $x_0^{(i)}$ from the prior $p(x_0)$
2. For $t = 1, 2, \ldots$

   (a) Importance sampling step
   - For $i = 1, \ldots, N$
     - Compute the Jacobians $\mathbf{F}_t^{(i)}, \mathbf{G}_t^{(i)}$ of the process model and $\mathbf{H}_t^{(i)}, \mathbf{U}_t^{(i)}$ of the measurement model
     - Update the particles with EKF

$$\begin{aligned}
\bar{\mathbf{x}}_{t|t-1}^{(i)} &= f(\mathbf{x}_{t-1}^{(i)}) && (4.53)\\
\mathbf{P}_{t|t-1}^{(i)} &= \mathbf{F}_t^{(i)}\mathbf{P}_{t-1}^{(i)}\mathbf{F}_t^{T(i)} + \mathbf{G}_t^{(i)}\mathbf{Q}_t\mathbf{G}_t^{T(i)} && (4.54)\\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}^{(i)}\mathbf{H}_t^{T(i)}[\mathbf{U}_t^{(i)}\mathbf{R}_t\mathbf{U}_t^{T(i)} + \mathbf{H}_t^{(i)}\mathbf{P}_{t|t-1}^{(i)}\mathbf{U}_t^{T(i)}]^{-1} && (4.55)\\
\bar{\mathbf{x}}_t^{(i)} &= \bar{\mathbf{x}}_{t|t-1}^{(i)} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}^{(i)})) && (4.56)\\
\hat{\mathbf{P}}_t^{(i)} &= \mathbf{P}_{t|t-1}^{(i)} - \mathbf{K}_t\mathbf{H}_t^{(i)}\mathbf{P}_{t|t-1}^{(i)} && (4.57)
\end{aligned}$$

     - Sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{(i)}, \hat{\mathbf{P}}_t^{(i)})$
   - For $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} \propto \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)})p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \qquad (4.58)$$

   - For $i = 1, \ldots, N$, normalize the importance weights

$$\tilde{w_t}^{(i)} = w_t^{(i)} \left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \qquad (4.59)$$

   (b) Selection step
   - Multiply/suppress samples $(\hat{\mathbf{x}}_{0:t}^{(i)}, \hat{\mathbf{P}}_{0:t}^{(i)})$ with high/low importance weights $\tilde{w_t}^{(i)}$, respectively, to obtain N random samples $(\tilde{\mathbf{x}}_{0:t}^{(i)}, \tilde{\mathbf{P}}_{0:t}^{(i)})$

   (c) MCMC step (optional)
   - Apply a Markov transition kernel with invariant distribution given by $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$ to obtain $(\mathbf{x}_{0:t}^{(i)}, \mathbf{P}_{0:t}^{(i)})$

   (d) Output See 'Generic Particle Filter'

EKF MH step

- Sample $u \sim U_{[0,1]}$
- Compute the Jacobians $\mathbf{F}_t^{*(i)}, \mathbf{G}_t^{*(i)}$ of the process model and $\mathbf{H}_t^{*(i)}, \mathbf{U}_t^{*(i)}$ of the measurement model
- Update the particles with EKF

$$\bar{\mathbf{x}}_{t|t-1}^{*(i)} = f(\tilde{\mathbf{x}}_{t-1}^{(i)}) \tag{4.60}$$

$$\mathbf{P}_{t|t-1}^{*(i)} = \mathbf{F}_t^{*(i)}\tilde{\mathbf{P}}_{t-1}^{(i)}\mathbf{F}_t^{*T(i)} + \mathbf{G}_t^{*(i)}\mathbf{Q}_t\mathbf{G}_t^{*T(i)} \tag{4.61}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}^{*(i)}\mathbf{H}_t^{*T(i)}[\mathbf{U}_t^{*(i)}\mathbf{R}_t\mathbf{U}_t^{*T(i)} + \mathbf{H}_t^{*(i)}\mathbf{P}_{t|t-1}^{*(i)}\mathbf{U}_t^{*T(i)}]^{-1} \tag{4.62}$$

$$\bar{\mathbf{x}}_t^{*(i)} = \bar{\mathbf{x}}_{t|t-1}^{*(i)} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}^{*(i)})) \tag{4.63}$$

$$\mathbf{P}_t^{*(i)} = \mathbf{P}_{t|t-1}^{*(i)} - \mathbf{K}_t\mathbf{H}_t^{*(i)}\mathbf{P}_{t-1}^{*(i)} \tag{4.64}$$

- Sample $\mathbf{x}_t^{*(i)} \sim q(\mathbf{x}_t|\tilde{\mathbf{x}}_{0:t-1}^{(i)}\mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{*(i)}, \hat{\mathbf{P}}_t^{*(i)})$
- If $u \leq min\left\{1, \frac{p(\mathbf{y}_t|\mathbf{x}_t^{*(i)})p(\mathbf{x}_t^{*(i)}|\tilde{\mathbf{x}}_{t-1}^{(i)})q(\tilde{\mathbf{x}}_t^{(i)}|\tilde{\mathbf{x}}_{0:t-1}^{(i)},\mathbf{y}_{1:t})}{p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(i)})p(\tilde{\mathbf{x}}_t^{(i)}|\tilde{\mathbf{x}}_{t-1}^{(i)})q(\mathbf{x}_t^{*(i)}|\tilde{\mathbf{x}}_{0:t-1}^{(i)},\mathbf{y}_{1:t})}\right\}$

  - then accept move

$$\mathbf{x}_{0:t}^{(i)} = (\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t^{*(i)}) \tag{4.65}$$

$$\mathbf{P}_{0:t}^{(i)} = (\tilde{\mathbf{P}}_{0:t-1}^{(i)}, \mathbf{P}_t^{*(i)}) \tag{4.66}$$

  - else reject move

$$\mathbf{x}_{0:t}^{(i)} = \tilde{\mathbf{x}}_{0:t}^{(i)} \tag{4.67}$$

$$\mathbf{P}_{0:t}^{(i)} = \tilde{\mathbf{P}}_{0:t}^{(i)} \tag{4.68}$$

End If

4.7 Unscented Filter

As described in section 4.2, the UKF propagates the mean and covariance of the Gaussian approximation to the state distribution more accurately than the EKF and tends to generate better estimates of the true covariance of the state ([26]). Furthermore, the distributions generated by the UKF generally have a broader overlap with the true posterior distribution compared to the EKF estimates which is partly due to the fact that the UKF computes the posterior covariance accurately to the third order (Gaussian prior) whereas the EKF uses a first order biased approximation. The UKF also includes the latest observations, but in a more accurate way. Furthermore, the UKF is able to scale the approximation errors in higher order moments of the posterior distribution allowing heavier tailed distributons. As the sigma points in the UKF are deterministically designed to capture certain characteristics of the prior distribution, it is possible to explicitly tune the algorithm to work on distributions that have heavier tails than Gaussian distributions, e.g. Student-t distributions. All in all, the UKF is more likely to generate more accurate proposal distributions within the particle filtering framework. Using the UKF as proposal distribution generator leads to the *Unscented Filter*, [26], (in this work abbreviated PFUKF). The pseudo-code for this filter is shown below.

4.7.1 Implementation

Unscented Filter

1. Initialization

   - For $i = 1, \dots, N$, draw particles $\mathbf{x}_0^{(i)}$ from the prior $p(\mathbf{x}_0)$ and set

$$\begin{aligned}
\bar{\mathbf{x}}_0^{(i)} &= E[\mathbf{x}_0^{(i)}] && (4.69)\\
\mathbf{P}_0^{(i)} &= E[(\mathbf{x}_0^{(i)} - \bar{\mathbf{x}}_0^{(i)})(\mathbf{x}_0^{(i)} - \bar{\mathbf{x}}_0^{(i)})^T] && (4.70)\\
\bar{\mathbf{x}}_0^{(i)a} &= E[\mathbf{x}^{(i)a}] = [(\bar{\mathbf{x}}_0^{(i)})^T \mathbf{0}\ \mathbf{0}]^T && (4.71)\\
\mathbf{P}_0^{(i)a} &= E[(\mathbf{x}_0^{(i)a} - \bar{\mathbf{x}}_0^{(i)a})(\mathbf{x}_0^{(i)a} - \bar{\mathbf{x}}_0^{(i)a})^T] = \begin{pmatrix} \mathbf{P}_0^{(i)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{pmatrix} && (4.72)
\end{aligned}$$

2. For $t = 1, 2, \ldots$

   (a) Importance sampling step

   - For $i = 1, \ldots, N$
     - Update the particles with UKF
       * Calculate sigma points

$$\boldsymbol{\chi}_{t-1}^{(i)a} = \left[\begin{array}{cc} \bar{\mathbf{x}}_{t-1}^{(i)a} & \bar{\mathbf{x}}_{t-1}^{(i)a} \pm \sqrt{(n_a + \lambda)\mathbf{P}_{t-1}^{(i)a}} \end{array} \right] \tag{4.73}$$

       * Propagate particle (time update)

$$\begin{aligned} \boldsymbol{\chi}_{t|t-1}^{(i)x} &= \mathbf{f}(\boldsymbol{\chi}_{t-1}^{(i)x}, \boldsymbol{\chi}_{t-1}^{(i)v}) && (4.74) \\ \bar{\mathbf{x}}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(m)} \chi_{j,t|t-1}^{(i)x} && (4.75) \\ \mathbf{P}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\chi_{j,t|t-1}^{(i)x} - \bar{\mathbf{x}}_{t|t-1}^{(i)x}][\chi_{j,t|t-1}^{(i)x} - \bar{\mathbf{x}}_{t|t-1}^{(i)}]^T && (4.76) \\ \gamma_{t|t-1}^{(i)} &= \mathbf{h}(\boldsymbol{\chi}_{t|t-1}^{(i)x}, \boldsymbol{\chi}_{t-1}^{(i)n}) && (4.77) \\ \bar{\mathbf{y}}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(m)} \gamma_{j,t|t-1}^{(i)} && (4.78) \end{aligned}$$

       * Incorporate new observation (measurement update)

$$\begin{aligned} \mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}][\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}]^T && (4.79) \\ \mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\chi_{j,t|t-1}^{(i)} - \bar{\mathbf{x}}_{t|t-1}^{(i)}][\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}]^T && (4.80) \\ \mathbf{K}_t &= \mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} \mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t}^{-1} && (4.81) \\ \bar{\mathbf{x}}_t^{(i)} &= \bar{\mathbf{x}}_{t|t-1}^{(i)} + \mathbf{K}_t(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1}^{(i)}) && (4.82) \\ \hat{\mathbf{P}}_t^{(i)} &= \mathbf{P}_{t|t-1}^{(i)} - \mathbf{K}_t \mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} \mathbf{K}_t^T && (4.83) \end{aligned}$$

     - Sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{(i)}, \hat{\mathbf{P}}_t^{(i)})$
     - Set $\hat{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \mathbf{x}_t^{(i)})$ and $\hat{\mathbf{P}}_{0:t}^{(i)} \triangleq (\mathbf{P}_{0:t-1}^{(i)}, \mathbf{P}_t^{(i)})$

- For $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} \propto \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)})p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{4.84}$$

- For $i = 1, \ldots, N$, normalize the importance weights

$$\tilde{w_t}^{(i)} = w_t^{(i)} \left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \tag{4.85}$$

(b) <u>Selection step</u>

- Multiply/suppress samples $(\hat{\mathbf{x}}_{0:t}^{(i)}, \hat{\mathbf{P}}_{0:t}^{(i)})$ with high/low importance weights $\tilde{w_t}^{(i)}$, respectively, to obtain N random samples $(\tilde{\mathbf{x}}_{0:t}^{(i)}, \tilde{\mathbf{P}}_{0:t}^{(i)})$

(c) <u>MCMC step (optional)</u>

- Apply a Markov transition kernel with invariant distribution given by $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$ to obtain $(\mathbf{x}_{0:t}^{(i)}, \mathbf{P}_{0:t}^{(i)})$

(d) <u>Output</u> See 'Generic Particle Filter'

Chapter 5

Simple simulation

Initially, we apply our algorithms to a simple one-dimensional simulated problem in which we can control (and hence know) all parameters and values allowing us to validate our implementation. Furthermore, we can easily compare the algorithms and visualize their differences and characteristics without having too many unknown parameters and factors that may influence the outcome of our estimates in various ways.

Our DBN has three nodes and is illustrated in Figure 5.1.

The true process model is a simple noisy AR(1) model given by

$$x_t = 2 + cos(\alpha \pi t) + \beta x_{t-1} \tag{5.1}$$

The measurement model consists of three parts in which the first part is a first order relation, the second part a second order and the final part a third order relation given by

$$y_t = \begin{cases} \theta_1 x_t & t <= 20 \\ \theta_2 {x_t}^2 & 20 < t <= 40 \\ \theta_3 {x_t}^3 & t > 40 \end{cases} \tag{5.2}$$

The measurement model was chosen to illustrate the strengths and weaknesses of the algorithms for distributions with increasing non-linearity, especially with respect to the

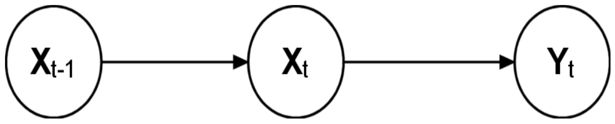


Figure 5.1: DBN for simple one-dimensional problem

comparison of EKF, PFEKF, PFEKFMC vs. UKF, PFUKF, PFUKFMC algorithms as the estimates of the posterior mean and variance are accurate to the second order of the Taylor series expansion of $\boldsymbol{g}(\boldsymbol{x})$ in the unscented transformation as opposed to the first order accuracy of the EKF, see section 4.1 and 4.2 resp.

In all experiments the process and measurement equation parameters were set to $\alpha = 0.05, \beta = 0.5, \theta_1 = -1, \theta_2 = 0.05$ and $\theta_3 = 0.01$ in the data generation.

Furthermore, all experimental results were based on an average over 10 runs to eliminate the statistical influence and all particle filters (i.e. all algorithms except EKF and UKF) used a fairly low number of particles allowing us to show how the different particle filters perform without using too many particles. Using a lot more particles we would obtain better results all over, but the primary intension with these experiments were to compare the algorithms and validate them on a simple problem.

In all experiments the values $\alpha = 1, \beta = 0$ and $\kappa = 2$ were used, as these are the optimal for the scalar case ([26]) and the choice of resampling algorithm did not seem to influence the results significantly and hence residual resampling was chosen arbitrarily (they are all $O(N)$ algorithms, [26]).

In our study of the different algorithms and different scenarios, we have chosen to divide our experiments into three scenarios presented in the following subsections.

First, we experiment with 'optimal' conditions, i.e. by adding Gaussian noise to the data which is the basic assumption (on the additive noise) for EKF and UKF and use the true process and measurement models as proposal models. This allows us to see and compare the outcome of the different algorithms when applied to a problem in which they do not use approximate process and measurement models which could affect their performance.

Next, we violate the Gaussian noise assumption and draw process noise samples from a Gamma distribution allowing us to compare the robustness of the EKF and UKF implementations when the basic assumptions are not true and compare with PF/PFMC which do not assume Gaussian distributed noise.

Finally, in most real life applications it is not possible to model the data generation exactly using a DBN which is a probabilistic model expressing our beliefs of how data is generated and how variables are related and even if it was, the true process and measurement models would often be unknown. Thus our model becomes an approximation to the ground truth. In the previous experiments we were only faced with the challenge of tracking in a noisy environment as we generated the data using the exact same DBN with the same process and measurement models. Hence, to really expose the capability of the different algorithms we propose a process model different from the true one to simulate an approximate model.

In [26] experiments using a similar network are presented, but the process and measurement models are different. Most importantly, the measurement model in [26] does not include a third order relation and experiments are only performed using the true process and measurement models and using Gamma process noise and Gaussian measurement noise.

True process and measurement proposal models, Gaussian process/measurement noise

In this experiment we expect all algorithms to perform well as the basic noise assumption of EKF and UKF are true and we propose the same process and measurement models as used in the data generation. Hence, to see any difference we use a low number of particles and apply a fairly large process noise compared to the actual values of the data thereby lowering the signal-to-noise ratio making the state estimation more difficult.

Using 100 particles and drawing the process and measurement noise samples from the Gaussian distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(0,1e-4)$ resp. and using the same noise levels as proposal noise levels, we get the summarizing values in Table 5.1 of the average Root Mean Square Error (RMSE) of the state mean estimates (which can only be calculated as were are doing a simulation and so the hidden state variables are - practically - not hidden). The second column is the variance of the RMSE over the 10 runs.

Algorithm	RMSE	
	mean	var
Extended Kalman Filter (EKF)	0.2711	0
Unscented Kalman Filter (UKF)	0.2593	0
Particle Filter - generic (PF)	0.7621	0.0099
Particle Filter - Metropolis-Hastings move (PFMC)	0.6797	0.0023
Particle Filter - EKF proposal (PFEKF)	0.2543	4.97e-06
Particle Filter - EKF proposal and MH move (PFEKFMC)	0.2478	3.79e-05
Particle Filter - UKF proposal (PFUKF)	0.1034	3.12e-05
Particle Filter - UKF proposal and MH move (PFUKFMC)	0.1003	4.46e-05

Table 5.1: Mean and variance of RMSE values of state mean estimates using the true process and measurement models and Gaussian process/measurement noise

Even in these 'optimal' conditions we still see a significant difference in performance, but to get further insight we need to see the variations over the three stages in the measurement model.

The corresponding plots of the state mean estimates vs. the true values are shown in Figure 5.2, 5.3 and 5.4.

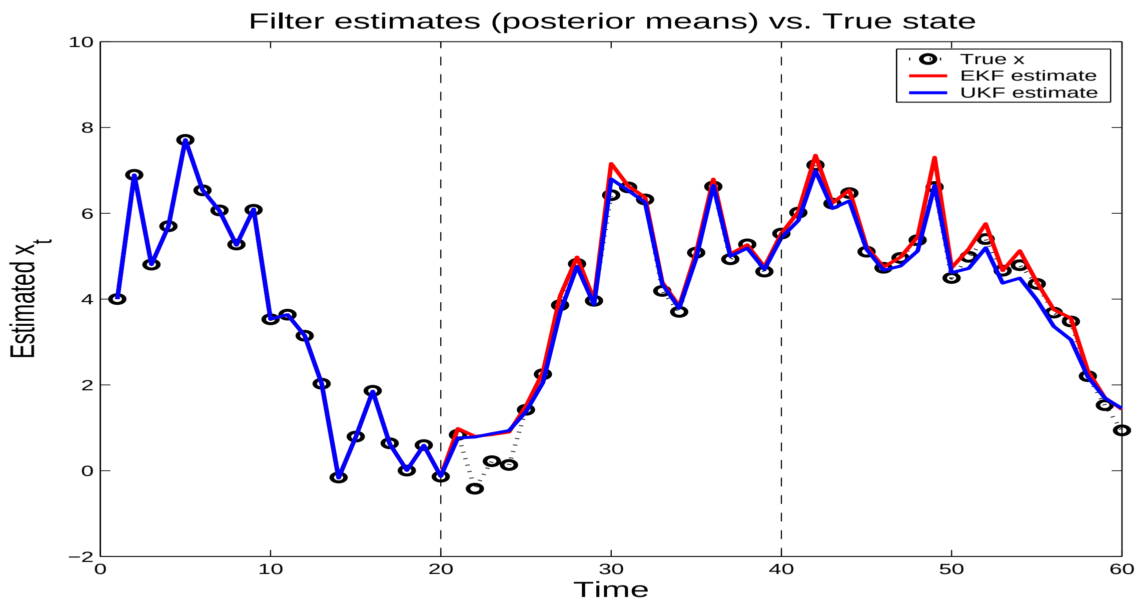


Figure 5.2: True state values, EKF and UKF estimates using the true process and measurement models and Gaussian process/measurement noise

In Figure 5.2 we notice how EKF and UKF are giving the same state mean estimates in the first order stage of the measurement model in accordance with our expectations, as both filters capture the true posterior mean and variance (of the Gaussian approximation to the true posterior), see sec. 4.1 and 4.2. In the second and third order stages we start to notice a difference in their estimates. Sometimes EKF is overestimating while UKF is accurate (e.g. $t = 30$) and at other times EKF is more accurate while UKF is underestimating (e.g. $t = 53$). This does not surprise us as we expect EKF to have a lower state variance estimate as it uses a first order Taylor series approximation of the non-linear models and thus it is not able to move its predictions as far as UKF which uses the true non-linear models and approximates the distribution of the state random variable. However, UKF's ability to move its predictions further tends to cause situations in which UKF moves *too* far.

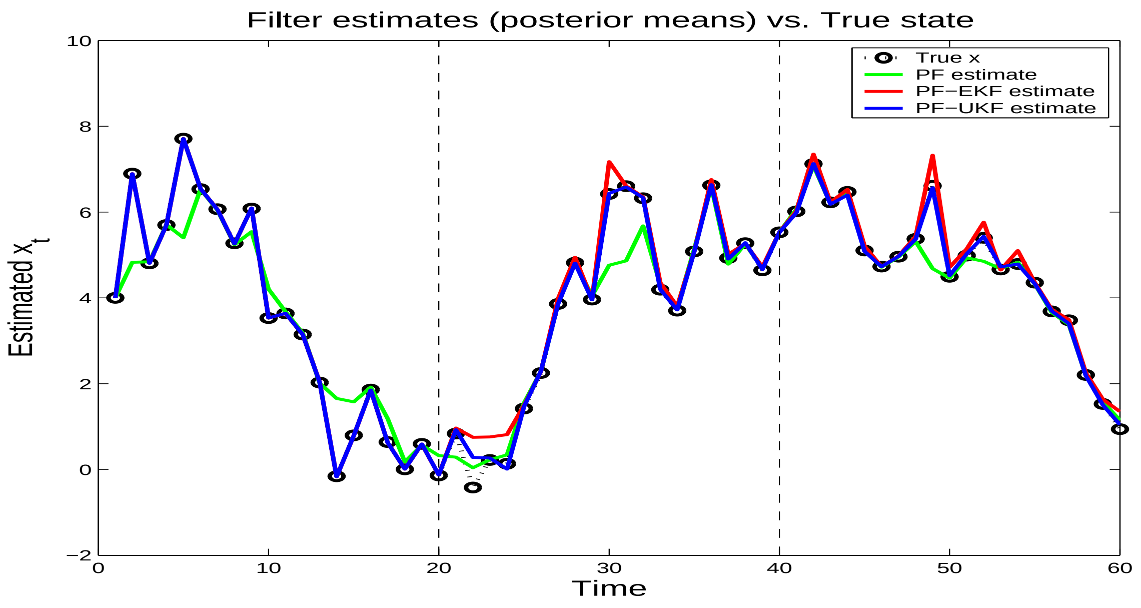


Figure 5.3: True state values, PF, PFEKF and PFUKF estimates using the true process and measurement models and Gaussian process/measurement noise

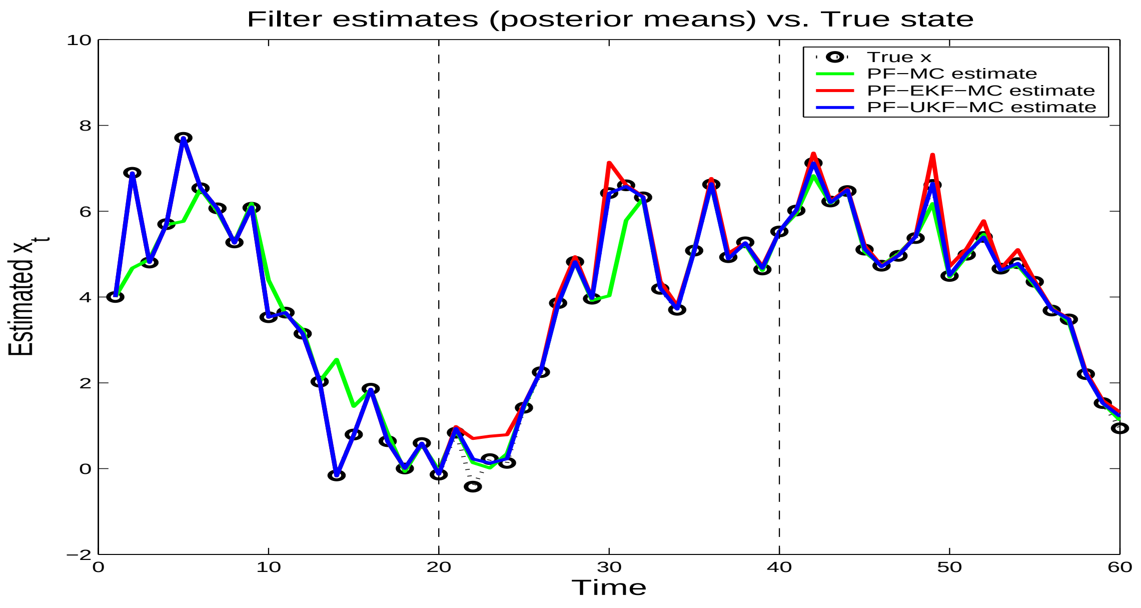


Figure 5.4: True state values, PFMC, PFEKFMC and PFUKFMC estimates using the true process and measurement models and Gaussian process/measurement noise

Now, in Figure 5.5, 5.6 and 5.7 the noise level and the RMS error for the EKF/UKF and PFEKF/PFUKF state mean estimates are shown. The figures illustrate how EKF consequently overestimates (negative error value) in the higher order stages of the measurement model due to its 1. order Taylor series approximation of the process and measurement models and the error amplitude follows the noise amplitude in a consistent manner. In comparison, UKF tends to underestimate when the noise sample is low (e.g. $t = 26, 41$), but is able to make accurate estimates for larger noise samples taking advantage of its ability to capture the true posterior mean and variance (e.g. $t = 49$).

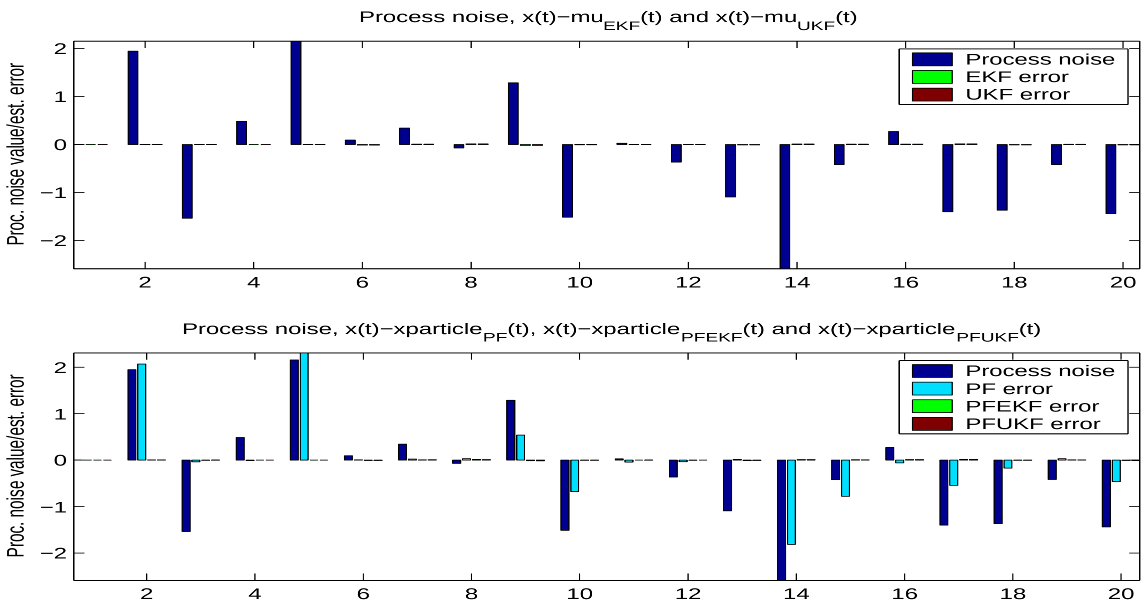


Figure 5.5: Noise level and RMS error for EKF, UKF, PF, PFEKF and PFUKF state mean estimates for $t \leq 20$ using the true process and measurement models and Gaussian process/measurement noise

We notice further how PFEKF is not able to correct the mistakes made by EKF in the higher order stages as sampling from a normal distribution with mean and variance values proposed by EKF and corresponding low variance does not improve the situation. In comparison PFUKF is able to correct the mistakes made by UKF by sampling from a distribution with a more accurate mean and much larger and more accurate variance, i.e. when the UKF overestimated, PFUKF assigns low weights to samples corresponding to overestimation and high weights to samples closer to the true state mean.

Finally, PF is the least accurate filter in the first order stage as it does not sample from a distribution that is guaranteed to encapsulate the true posterior mean and covariance. Furthermore, the likelihood which is the basis for the weighting of the particles in PF is very peaked (low measurement error) causing a lot of problems for this simple filter using the prior as proposal.

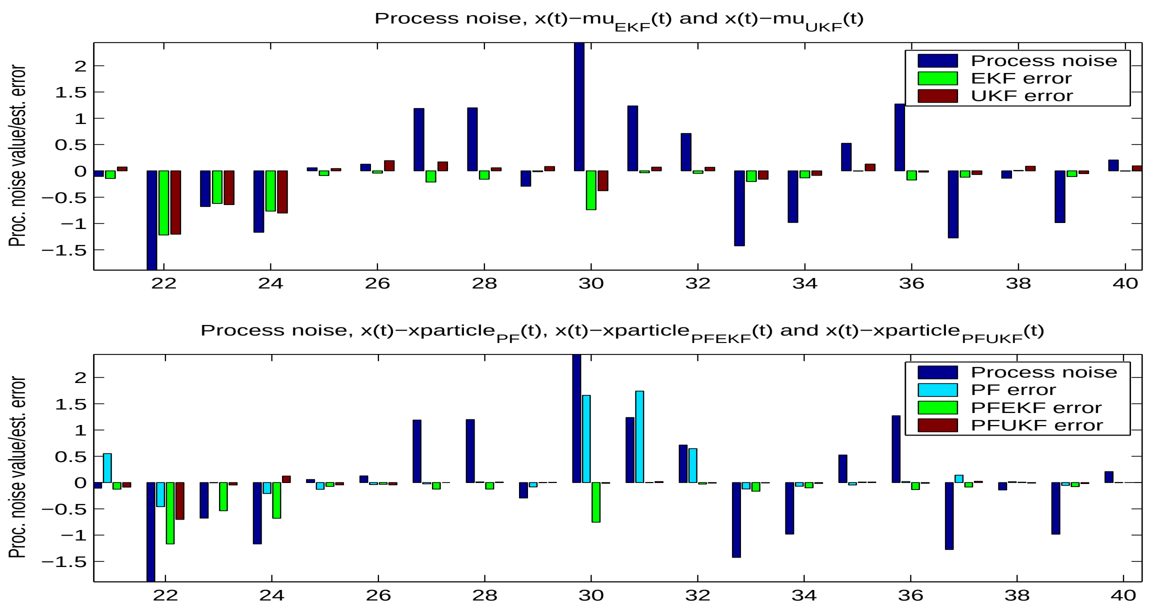


Figure 5.6: Noise level and RMS error for EKF, UKF, PF, PFEKF and PFUKF state mean estimates for $20 < t \leq 40$ using the true process and measurement models and Gaussian process/measurement noise

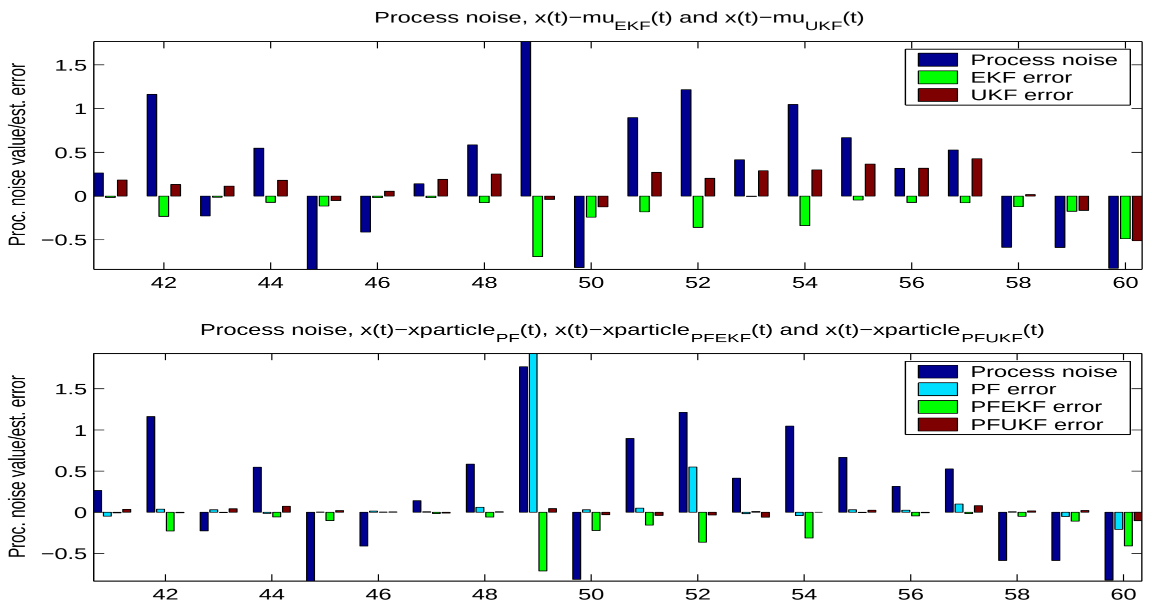


Figure 5.7: Noise level and RMS error for EKF, UKF, PF, PFEKF and PFUKF state mean estimates for $t > 40$ using the true process and measurement models and Gaussian process/measurement noise

It is worth noting how the MC step has a significant positive effect on the estimates in PFMC by moving the particles towards regions of high likelihood which is crucial for the generic particle filter. By comparing Figures 5.3 and 5.4 we see how PFMC saves the poor estimates of PF for $t = 8, 21, 49$ and 52. In comparison, the MC move has very little effect on PFEKF and PFUKF, but for different reasons. In PFEKFMC we are still troubled by the very small state variance estimate giving the MC step no chance of moving the particles far enough to be within reach of the very peaked likelihood. In PFUKFMC the state mean estimates from PFUKF are already very close to the true value and the MC step has no significant influence.

In Figure 5.8 the state variance estimates using EKF/UKF, PFEKF/PFUKF and PFEKFMC/ PFUKFMC resp. are shown. The plot illustrates how the UKF based estimates are much larger than the EKF based estimates in the two final stages of the measurement model corresponding to the second and third order relations. Recalling that UKF is able to capture the true mean and covariance to second order (of the Gaussian approximation to the true posterior) as opposed to only first order in EKF, the fundamental difference between the two approaches which makes UKF a much better proposal generator is visualized.

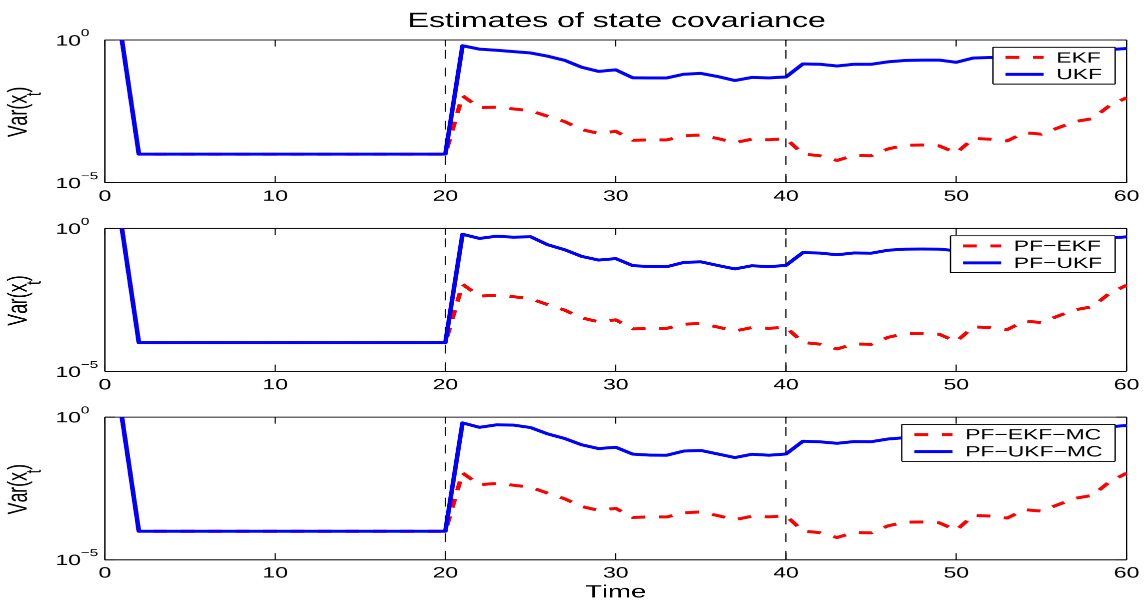


Figure 5.8: State variance estimates of EKF/UKF, PFEKF/PFUKF and PFEKFMC/PFUKFMC using the true process and measurement models and Gaussian process/measurement noise

To illustrate some of the up- and downsides of the different filters and how this influences the performance, we include a few examples from the experiment showing the sampling steps in PF, PFEKF and PFUKF in different situations. Each plot has three subplots, the first showing the true state $x(t)$ (black dot), the mean estimate $\mu(t)$ (blue dot), the 99% confidence interval of $p(x_t|y_t)$ (red line) and the predicted state $x_{pred}(t)$ (green dot).

Next subplot shows how the first 10 particles are weighted and the final subplot shows how one of the particles of particular interest is evaluated.

In Figure 5.9 corresponding to $t = 17$ it is illustrated how PFUKF can make matters worse when the UKF estimate is very close to the true state. In this case the proposal overlaps completely with the likelihood (the prior has no influence), but the measurement noise assigns particle 7 a low likelihood and the particle ends up having the lowest weight even though it is very close to the true state. We compare this with Figure 5.2 for $t = 17$ and notice how the estimate of PFUKF is slightly worse than UKF's, but not significantly as all the particles are close to the true state.

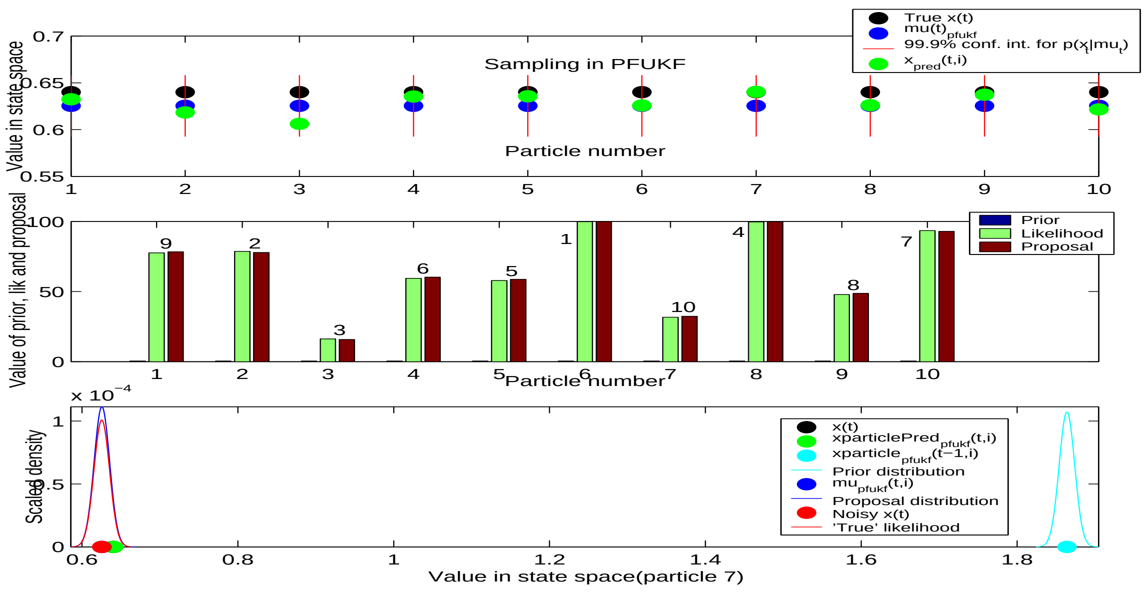


Figure 5.9: Illustration of sampling and weighting of particles in PFUKF for $t = 17$. Note that $x(t)$ and $\mu_{pfukf}(t,i)$ are not visible as they hidden below Noisy $x(t)$ and $xparticlePred_{pfukf}(t,i)$.

Figure 5.10 showing the sampling step in PFUKF at $t = 30$ illustrates how the proposal distribution captures the likelihood and produce a lot of good particles close to the true state. However, the peaked likelihood means that particle 7 has a very high likelihood and corresponding weight as the prior is without influence and the proposal is small compared to this single likelihood value. A MC step would have been helpful to move the remaining particles closer to the likelihood distribution to obtain a broader representation of particles. However, as particle 7 was also the one closest to the true value we obtain a more accurate estimate compared to UKF which is seen in Figure 5.3 as a decrease in error at $t = 30$.

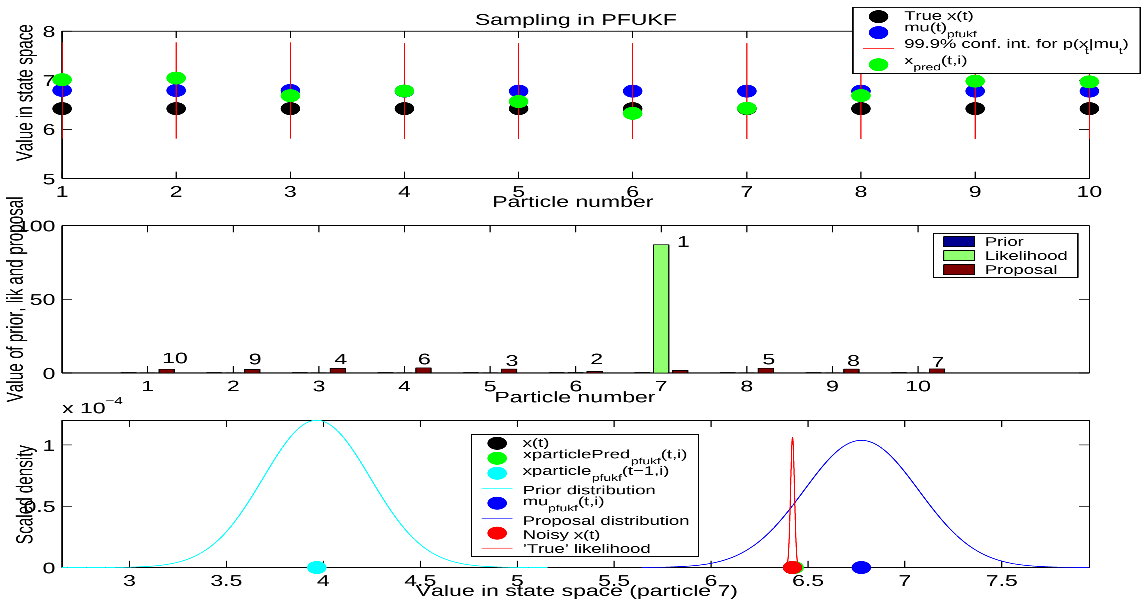


Figure 5.10: Illustration of sampling and weighting of particles in PFUKF for $t = 30$. Note that $x(t)$ and $xparticlePred_{pfukf}(t, i)$ are not visible as they hidden below Noisy $x(t)$.

Figure 5.11 illustrates how PF suffers severely from a peaked likelihood positioned in the tail of the prior, i.e. the probability of generating samples from the prior within the likelihood distribution is very, very low. As illustrated for $t = 52$, all 10 particles have 0 likelihood and the particles are weighted equally even though it is clear that some particles are very close to the true value, esp. particle 4 as illustrated. This is where the Metropolis-Hastings move is very useful as it is capable of moving the particles to regions of higher likelihood. In Figure 5.3 this gives rise to a poor estimate for PF at $t = 52$ which is turned into a much more accurate estimate in PFMC in Figure 5.4.

For comparison, the sampling step for PFEKF at $t = 52$ is shown in figure 5.12. We notice how the variance of the proposal distribution is very low making sampling within reach of the likelihood distribution impossible. As the prior is very far away, PFEKF primarily bases the weighting on the proposal and it has no change of correcting its poor predictions. In Figure 5.4 we notice a significant error for PFEKF at $t = 52$ about the same size as the errorbar for EKF.

PFUKF on the other hand has a much larger state variance estimate and its proposal captures the likelihood distribution as illustrated in Figure 5.13 for $t = 24$. All the proposed particles are close to the true state and most are within reach of the likelihood distribution and as shown in Figure 5.4 the error introduced in UKF is removed by PFUKF as it manages to move the estimates even closer to the true state and evaluate them in a sensible way using a combination of the prior, likelihood and proposal. It is important that several particles are within reach of the likelihood to give a broad representation of particles as the particle giving the highest likelihood (particle 1 as illustrated) is *not* the one closest to the true state due to the measurement noise.

Figure 5.14 summarizes the results of this experiment illustrating the distribution of the state estimation error over the three stages of the measurement model and shows how the error in the EKF based filters are introduced in the higher order stages, how PFUKF in general improves the basic UKF approach (especially in the higher order stages) and how the MC step manages to move the PF estimates towards regions of higher likelihood.

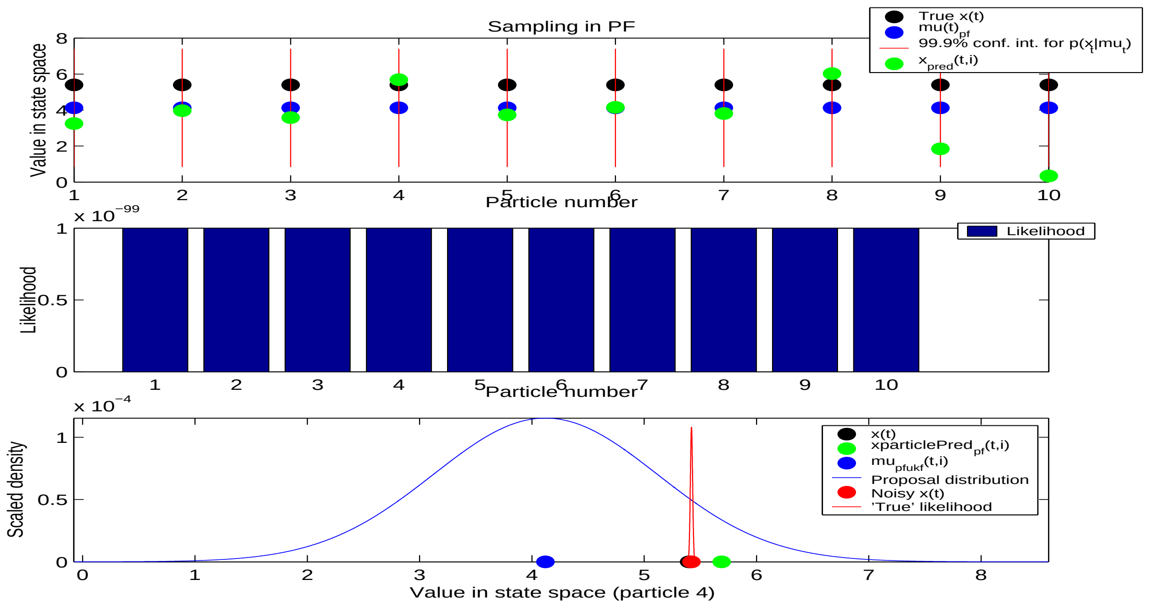


Figure 5.11: Illustration of sampling and weighting of particles in PF for $t = 52$. Note that $x(t)$ is almost not visible as it is hidden below Noisy $x(t)$.

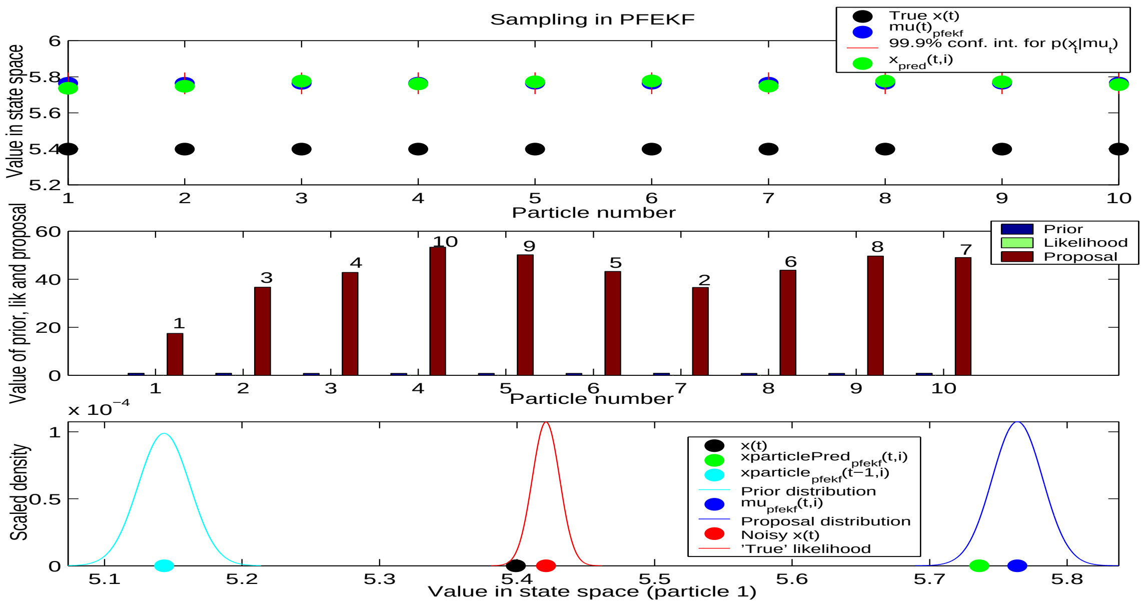


Figure 5.12: Illustration of sampling and weighting of particles in PFEKF for $t = 52$

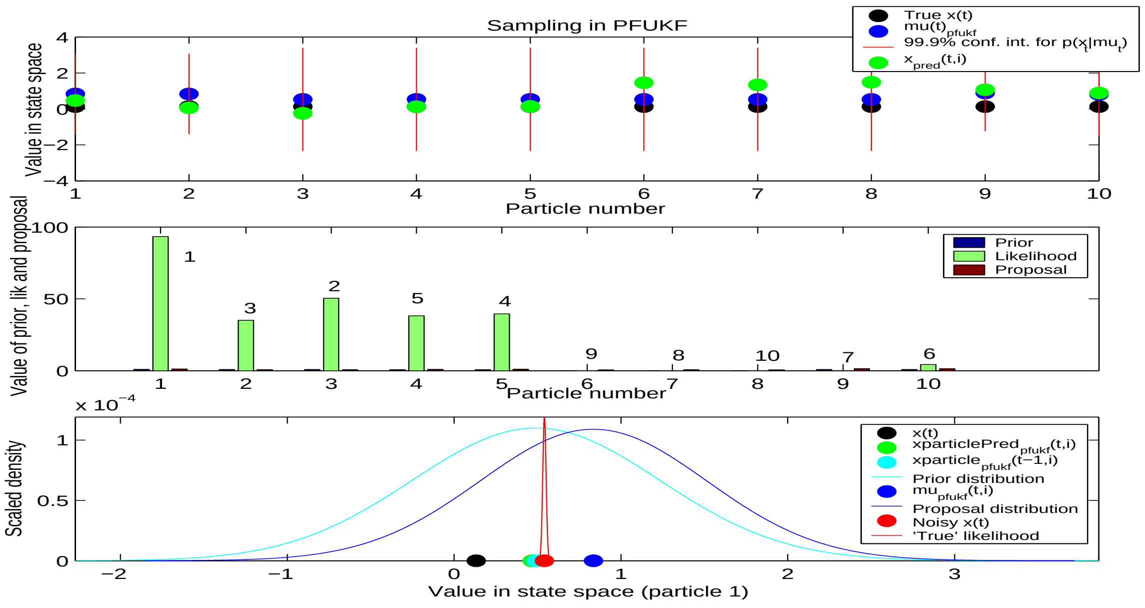


Figure 5.13: Illustration of sampling and weighting of particles in PFUKF for $t = 24$. Note that $xparticlePred_{pfukf}(t,i)$ is almost not visible as it is hidden below $xparticle_{pfukf}(t-1,i)$

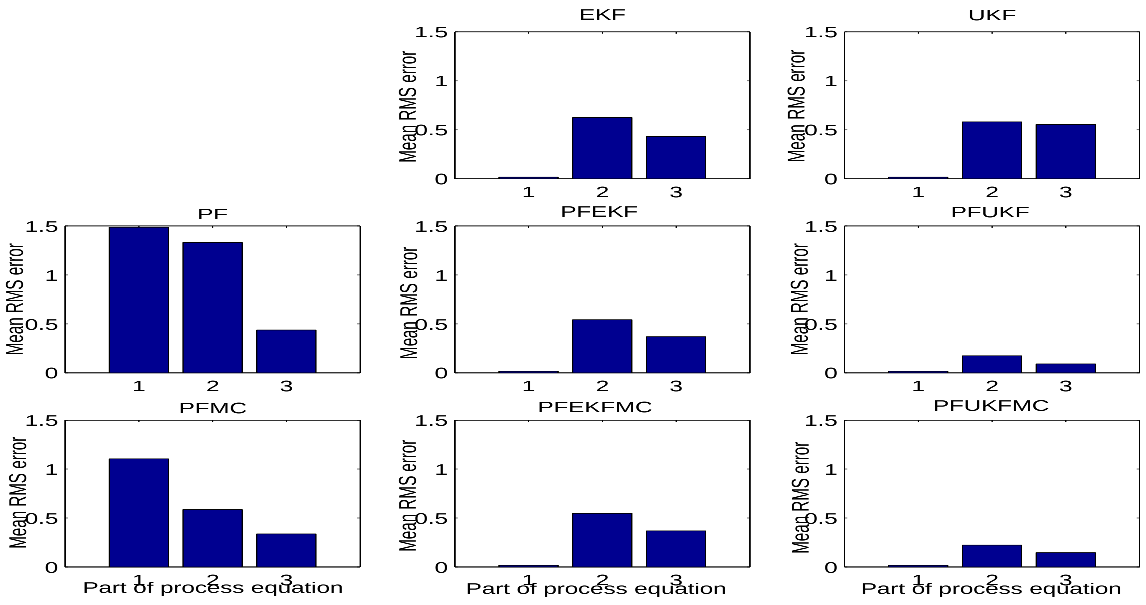


Figure 5.14: Distribution of error over the three stages of the measurement equation using the true process and measurement models and Gaussian process/measurement noise

Finally, Figure 5.15 shows the influence of the proposed noise variance on the state variance estimates in EKF and UKF for the three stages in the measurement model. The plots illustrate how EKF can benefit from larger proposals of process as well as measurement noise thus increasing its state variance estimates which in the experiment proved to be a major drawback of this filter. We have chosen not to propose larger noise variances in these simple experiments to be able to compare the filters using the same noise proposals, i.e. not 'helping anyone along'. But we include the plot to show that it is possible to tune the performance of the filters in different ways. UKF is comparable to EKF in the first order stage, but in the second order stage its estimates of the state variance seem to be independent of the proposed measurement noise and in the third order stage the effect is dramatically reduced. We keep these results in mind when we apply any of the filters to problems where our main objective is not to compare, but to maximize performance.

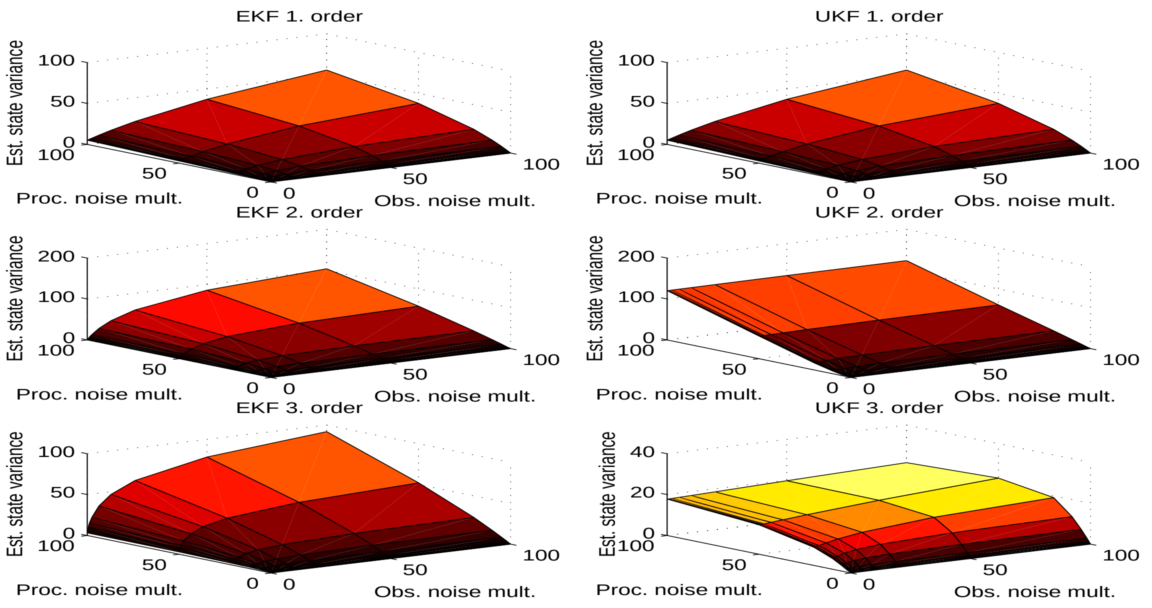


Figure 5.15: State variance estimate of EKF and UKF for the three stages of the measurement model as a function of process and measurement noise variance proposal multipliers K_X and K_Y, i.e. $\sigma^2_{Xprop} = K * \sigma^2_X$ and $\sigma^2_{Yprop} = K * \sigma^2_Y$ resp.

True process and measurement models, Gamma process noise/Gaussian measurement noise

In this experiment we violate the Gaussian assumption (on the additive noise) and draw process noise samples from a Gamma distribution $G(3,1)$, i.e. $\mu = 3, \sigma^2 = 3$, but keep drawing measurement noise samples from the Gaussian distribution $\mathcal{N}(0, 1e-4)$. We propose Gaussian noise levels $\mathcal{N}(0,3)$ and $\mathcal{N}(0, 1e-4)$ resp. The results using 500 particles are shown in Table 5.2.

Algorithm	RMSE	
	mean	var
Extended Kalman Filter (EKF)	1.9503	4.93e-32
Unscented Kalman Filter (UKF)	0.6080	0
Particle Filter - generic (PF)	0.2044	0.0379
Particle Filter - Metropolis-Hastings move (PFMC)	0.1185	0.0133
Particle Filter - EKF proposal (PFEKF)	1.9503	2.25e-06
Particle Filter - EKF proposal and MH move (PFEKFMC)	1.9518	2.42e-06
Particle Filter - UKF proposal (PFUKF)	0.0291	5.44e-04
Particle Filter - UKF proposal and MH move (PFUKFMC)	0.0290	0.0030

Table 5.2: Mean and variance of RMSE values of state mean estimates using the true process and measurement models and Gamma process noise/Gaussian measurement noise

Initially, we notice how the error levels have all changed compared to the experiment in section 5. The error level of EKF and UKF have increased due to the violation of the Gaussian assumption (on the additive noise), but this has a much more significant influence on the performance of EKF. The lower estimate of the state variance becomes much more visual in this non-Gaussian noise environment. This is well illustrated in Figure 5.16 which shows how EKF and UKF have the same state mean estimates for the first order stage of the measurement model, but in the second order and especially the third order stage, EKF keeps overestimating the state mean (due to the non-negative Gamma noise, i.e. the true state value giving a certain measurement is always lower) and the low state variance estimates make it impossible for the algorithm to correct its wrong predictions. In comparison, the UKF state mean estimates are much better taking advance of the higher state variance estimates. The effect is most evident for $t = 59$, where the true state is very different from the noisy state (corresponding to a high process noise sample as illustrated in Figure 5.19, subplot 1 and $t = 59$) and thus EKF needs to correct its prediction, but fails to do this.

UKF also fails to estimate the state mean correctly primarily because the Gaussian approximation to the true posterior is not adequate in the higher order parts of the measurement model and the violation of the Gaussian noise assumption, but makes a much better approximation than EKF.

The generic particle filter has a lower error level in this experiment compared to the previous experiment which seems reasonable as it does not make any assumptions of Gaussian noise and thus the effect of using more particles is seen. The estimates from PF are even more accurate than the EKF and UKF estimates, especially when used in conjunction with a Metropolis-Hastings move in PFMC. The significant influence of the MH move, primarily due to the very peaked likelihood density, becomes even more evident than in the previous experiment as we are using more particles.

Once again, the effect of using EKF as proposal generator in a particle filter setup does not seem to improve the state mean estimates as the state variance estimates are simply too small (with the proposed noise level that is).

Finally, PFUKF and PFUKFMC still proves to be the preferred solution taking advantage of the principles of the particle filter setup which does not assume Gaussian distributed noise and using a proposal generator with more accurate state variance estimates and broader overlap with the posterior density.

The filter estimates vs. the true states for PF, PFEKF and PFUKF are shown in Figure 5.17. Notice how PFUKF have 'saved' all the poor estimates of UKF while PFEKF is still failing. At $t = 59$ an interesting event occurs. Having drawn a large process noise sample, PFEKF is over-estimating dramatically, while PFUKF samples from an improved proposal distribution with much larger variance that captures the true state. In comparison, the PF estimate is less accurate as it weights its samples based on the likelihood only which in this case is not only peaked ($\sigma^2 = 1e - 4$), but has no overlap with the prior due to the large noise.

In Figure 5.18 and 5.19 the level of the noise samples and the corresponding estimation errors are shown for the higher order stages of the process model. Notice how the size of the EKF/PFEKF estimation error follows the level of noise and how they keep overestimating the state mean corresponding to a negative value of the estimation error. In comparison, UKF only overestimates when the noise level is high and to a much lower extent, but when the noise level is low, it underestimates. These results are in accordance with the higher state variance estimates of UKF. In conclusion, UKF clearly outperforms EKF when the process noise sample is large, whereas EKF makes more accurate estimates for small noise samples, but the relative difference between gain and loss of accuracy using UKF makes it by far the preferred filter. And as illustrated, using the UKF as proposal generator in a particle filter setup improves the accuracy for both small an higher levels of noise making PFUKF/PFUKFMC filters much more reliable.

In Figure 5.20 the summed estimation error for each stage of the measurement model is shown and visualizes the effect of the Gaussian violation leading EKF and UKF to perform poorly and the increasing influence on the performance of the EKF due to the poor state variance estimates for higher order non-linearities.

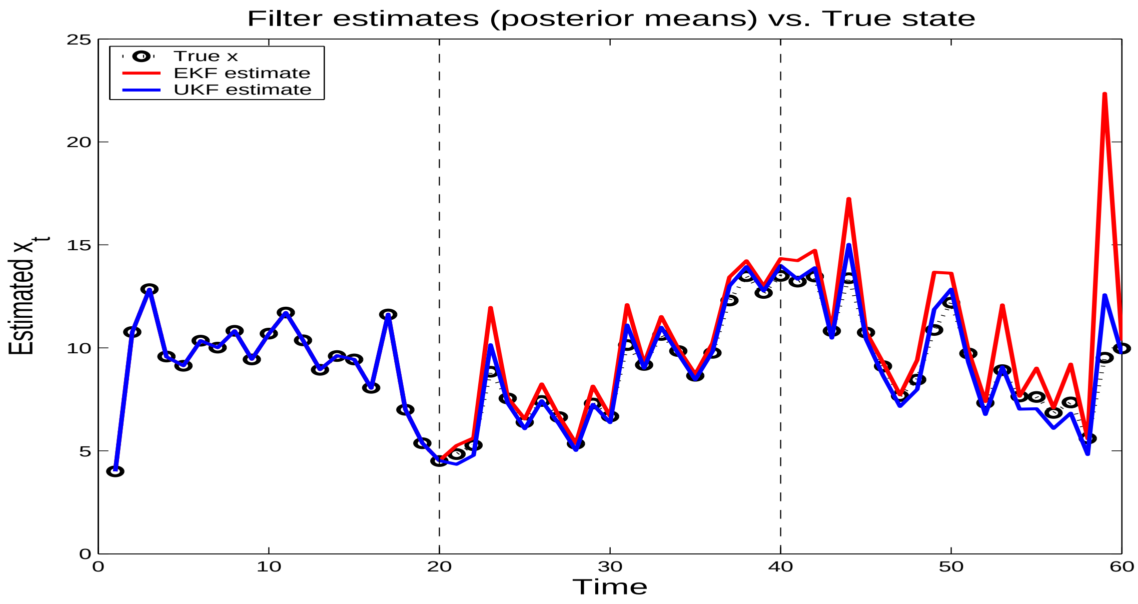


Figure 5.16: True state values, EKF and UKF estimates using the true process and measurement models and Gamma process noise/Gaussian measurement noise

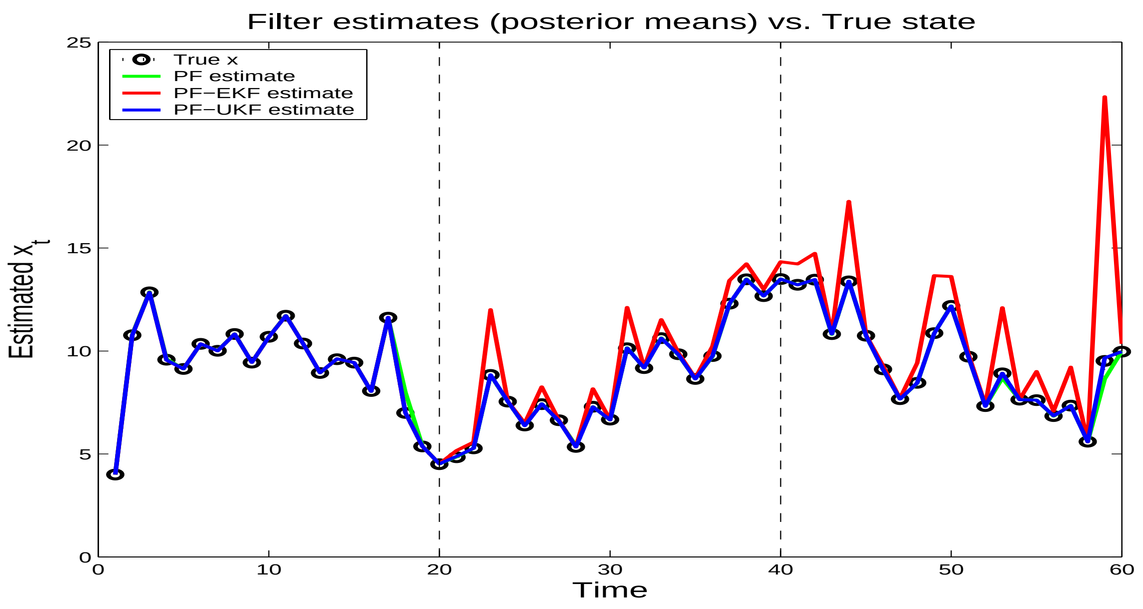


Figure 5.17: True state values, PF, PFEKF and PFUKF estimates using the true process and measurement models and Gamma process noise/Gaussian measurement noise

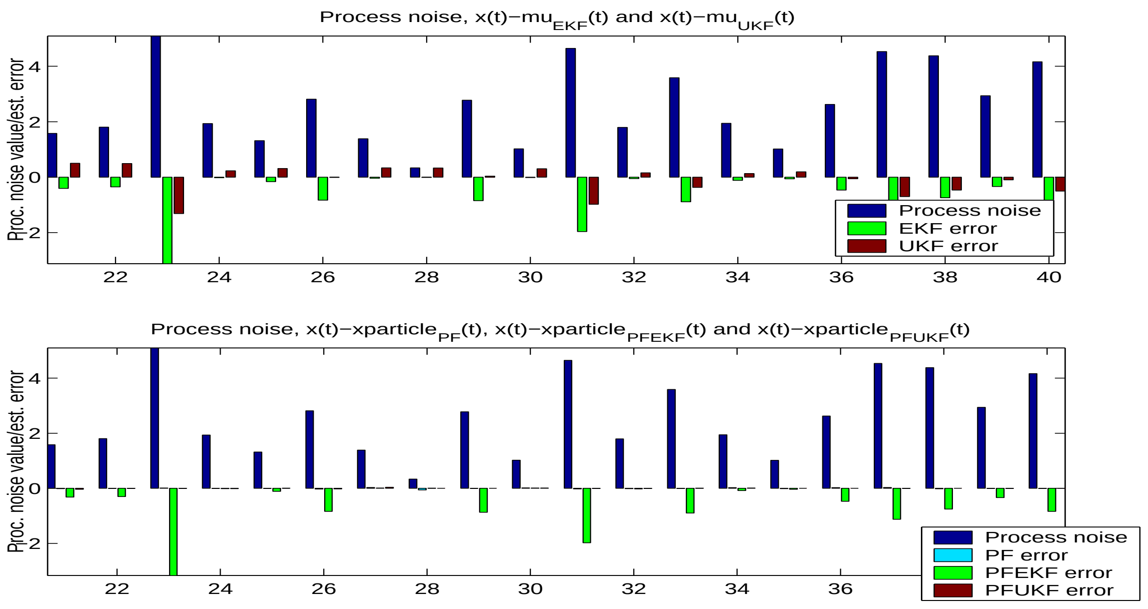


Figure 5.18: Noise level and RMS error for PF, PFEKF and PFUKF state mean estimates for $20 < t \leq 40$ using the true process and measurement models and Gamma process noise/Gaussian measurement noise

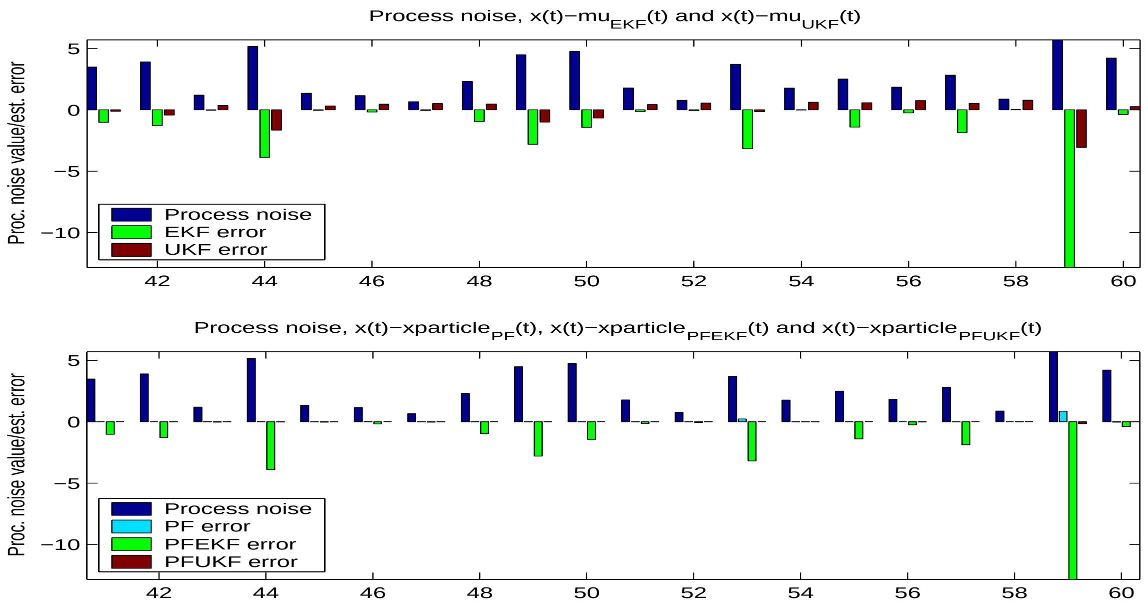


Figure 5.19: Noise level and RMS error for PF, PFEKF and PFUKF state mean estimates for $t > 40$ using the true process and measurement models and Gamma process noise/Gaussian measurement noise

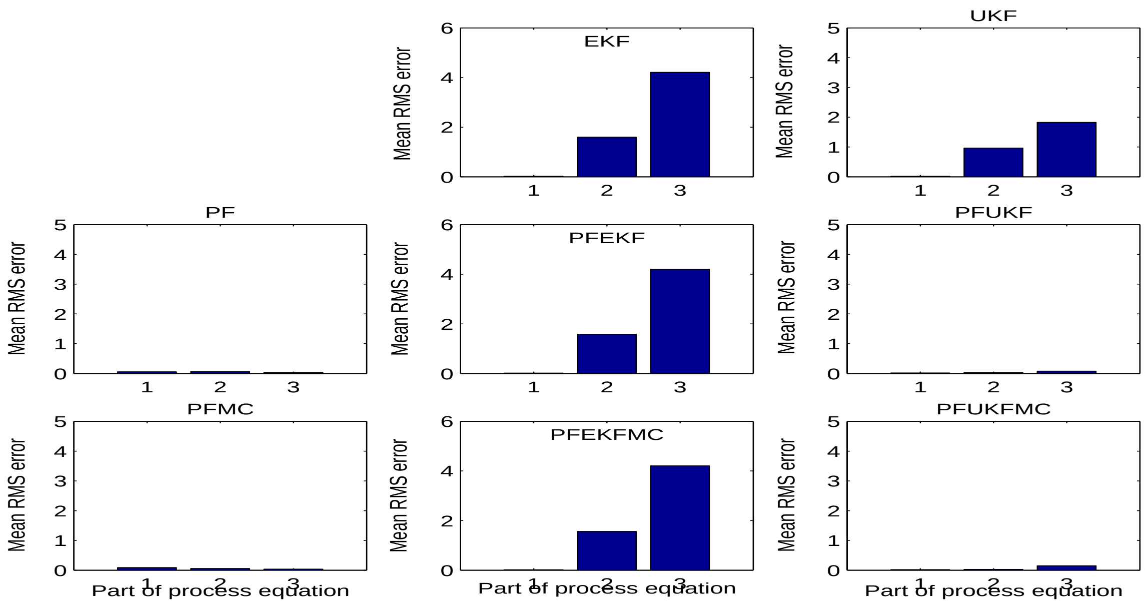


Figure 5.20: Distribution of error over the three stages of the measurement model using the true process and measurement models and Gamma process noise/Gaussian measurement noise

False process model and true measurement model, Gaussian process/measurement noise

In this final experiment we propose a false process model to test the robustness of each filter when the proposed model does not fit the actual data generation.

The proposed process model was

$$x_t = 2 + cos(\alpha \pi t) + \tilde{\beta} x_{t-1} \tag{5.3}$$

using $\alpha = 0.05$ and $\tilde{\beta} = 0.2$.

In the data generation we draw process and measurement noise samples from the Gaussian distributions $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 1e - 4)$ resp. and propose similar noise levels.

The results of this experiment using 500 particles are shown in Table 5.3 and the tracking plots in Figure 5.21 and 5.22. In Figure 5.23 and 5.24 the noise level and the RMS error for the EKF/UKF and PFEKF/PFUKF state mean estimates are shown. And finally in Figure 5.25 the summed estimation error for each stage of the measurement model is shown.

Algorithm	RMSE	
	mean	var
Extended Kalman Filter (EKF)	0.8120	4.11e-33
Unscented Kalman Filter (UKF)	0.3682	0
Particle Filter - generic (PF)	0.9666	0.0084
Particle Filter - Metropolis-Hastings move (PFMC)	0.9640	0.0016
Particle Filter - EKF proposal (PFEKF)	0.7977	3.85e-06
Particle Filter - EKF proposal and MH move (PFEKFMC)	0.7961	2.95e-05
Particle Filter - UKF proposal (PFUKF)	0.0823	0.0014
Particle Filter - UKF proposal and MH move (PFUKFMC)	0.0821	0.0011

Table 5.3: Mean and variance of RMSE values of state mean estimates using the false process and true measurement models and Gaussian process/measurement noise

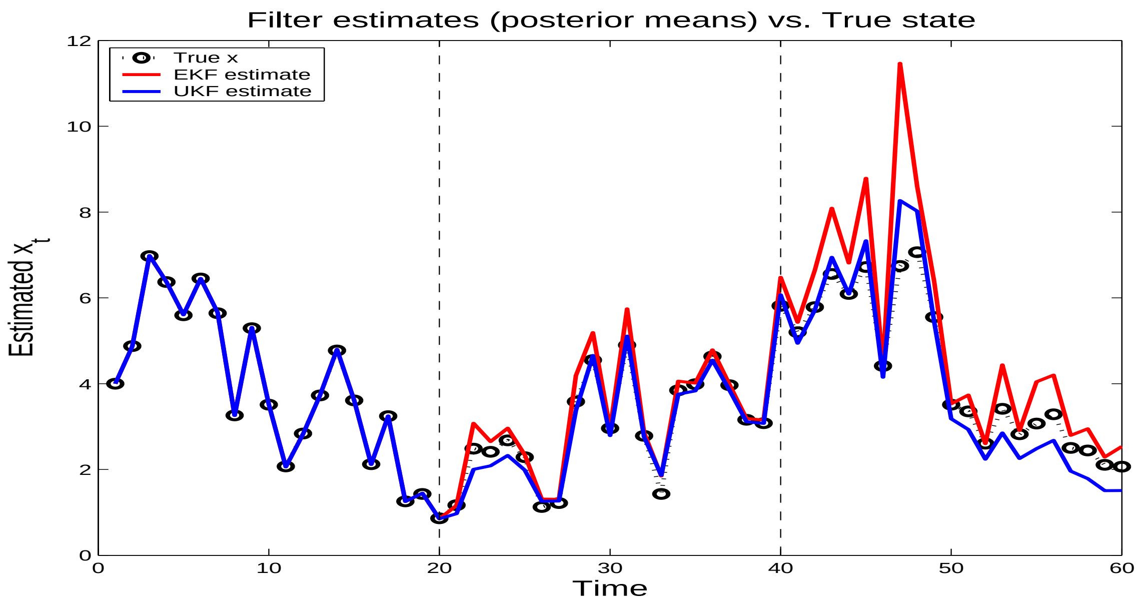


Figure 5.21: True state values, EKF and UKF estimates using the false process and true measurement models and Gamma process noise/Gaussian measurement noise

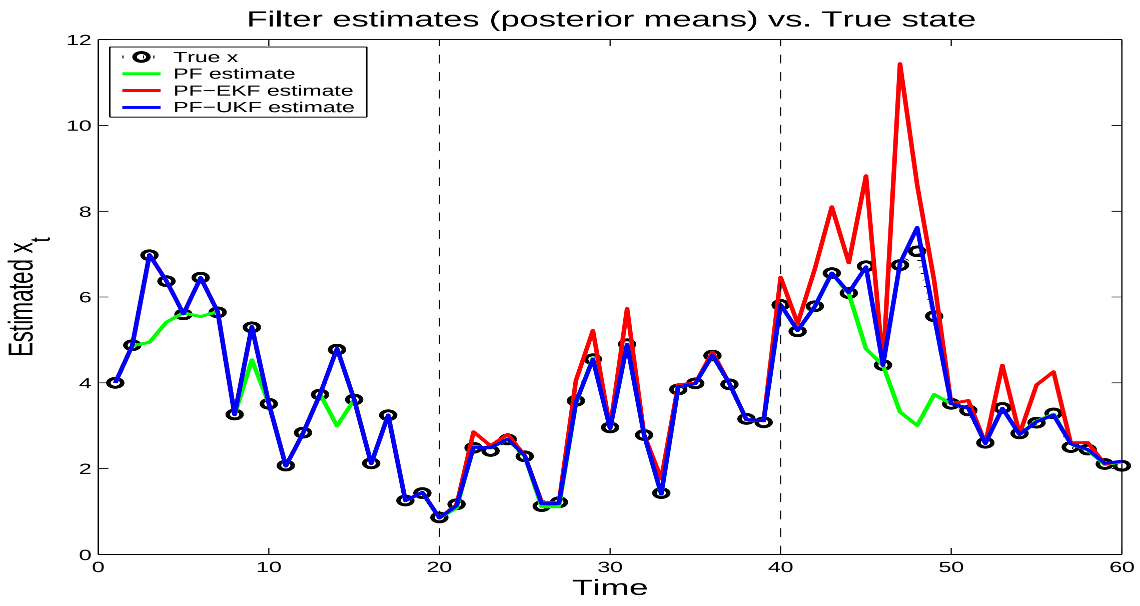


Figure 5.22: True state values, PF, PFEKF and PFUKF estimates using the false process and true measurement models and Gamma process noise/Gaussian measurement noise

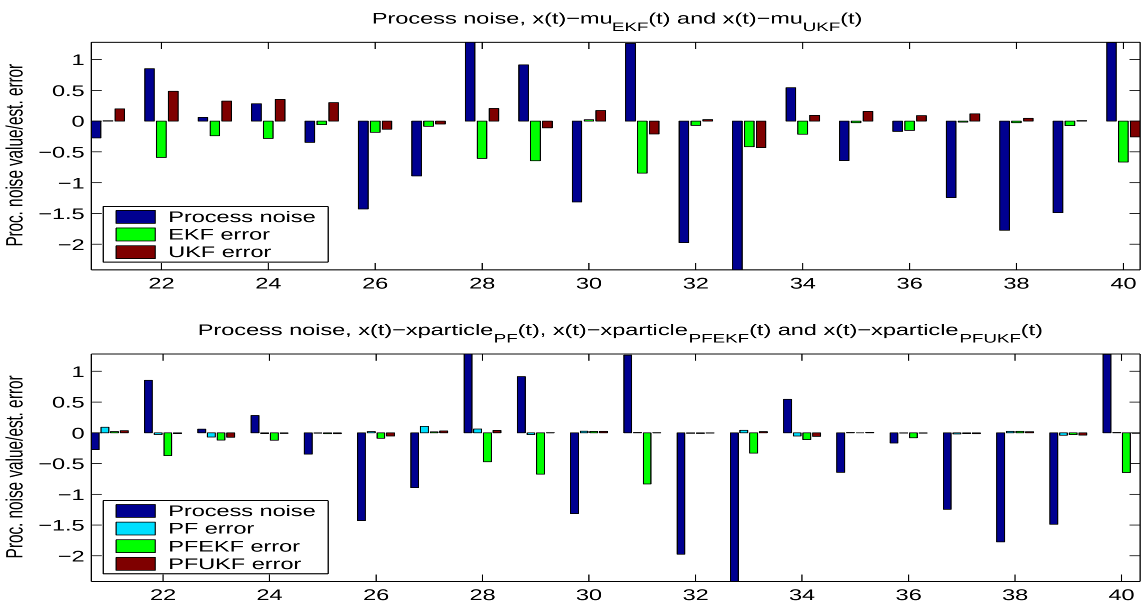


Figure 5.23: Noise level and RMS error for PF, PFEKF and PFUKF state mean estimates for $20 < t \leq 40$ using the false process and true measurement models and Gamma process noise/Gaussian measurement noise

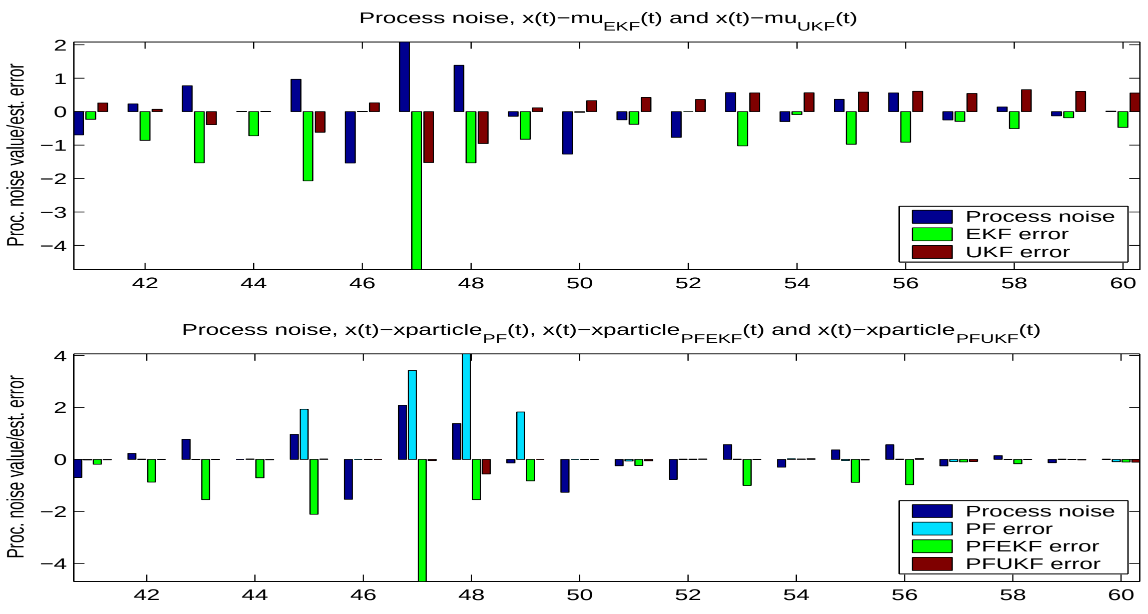


Figure 5.24: Noise level and RMS error for PF, PFEKF and PFUKF state mean estimates for $t > 40$ using the false process and true measurement models and Gamma process noise/Gaussian measurement noise

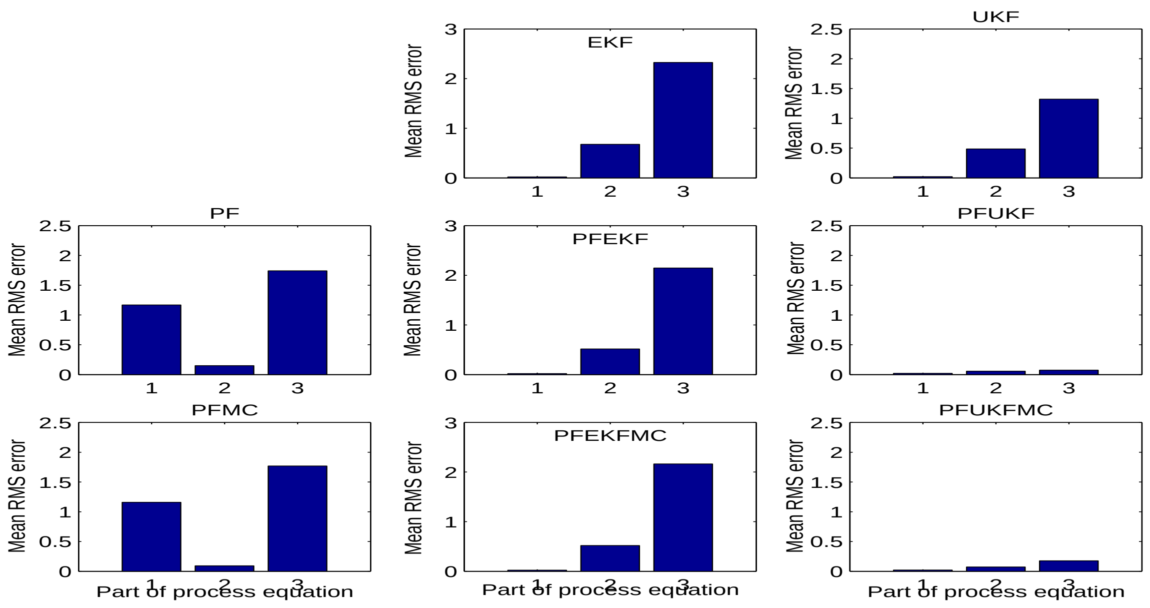


Figure 5.25: Distribution of error over the three stages of the measurement model using the false process and true measurement models and Gaussian process/measurement noise

Initially, we notice how all algorithms have introduced a higher estimation error using the false process model compared to the experiment in section 5 even though we are using a lot more particles, except for PFUKF/PFUKFMC. The use of a false process model naturally makes it a lot harder for the algorithms to estimate the state mean values as our beliefs of how the data was generated are no longer true. As mentioned, this is the most likely situation in real-life applications and thus we need a filter which does not fail dramatically when our beliefs are wrong.

By further inspection we notice how the generic particle filter is no longer capable of making better state mean estimates than EKF and UKF (with this number of particles) as sampling from the prior using a false process model leads to rather poor results for obvious reasons. And the MC move is not enough to improve the situation significantly as the prior is too far from the likelihood using the proposed noise level.

However, it is interesting to see how the main loss of accuracy compared to EKF is in the first order stage of the measurement model which is seen by comparing the estimates of EKF in Figure 5.21 and the estimates of PF in Figure 5.22. This is the stage where EKF is able to capture the true posterior mean and covariance. In the higher order stages PF is actually giving better state mean estimates than EKF.

PFEKF/PFEKFMC perform slightly better than EKF taking advantage of the particle filtering principles as the state variance estimates of EKF are now even worse partly based on the Jacobian of a false process model. It is still capable of tracking in the first order stage of the measurement model though as illustrated in Figure 5.21, but its limitations are shown in the higher order stages.

And finally, PFUKF and PFUKFMC prove their superiority once again by keeping the estimation error on a very low level. UKF itself is suffering from the use of a false process model in the higher-order stages of the measurement model, but is still capable of working as proposal generator in a particle filter setup.

In conclusion, we note that if our assumptions and beliefs are true there is not much difference between EKF and UKF using this simple network and first order models, but when the model order is increased, UKF makes more accurate estimates using its ability to capture the true posterior mean an variance up to second order whereas EKF tends to underestimates the noise. However, UKF can sometimes overestimate the noise when the noise samples are small compared to the proposed noise levels. Furthermore, PFUKF can save poor estimates from UKF, but can also make matters worse if the UKf estimates are close to the true states. We also noticed that PF was the least accurate filter, but significant improvements were made using an MC step. In general, the use of an MC step is very important in the standard particle filter setup, but has less influence when we use EKF or UKF as proposal generator (in this setup).

Furthermore, we showed how the use of UKF as proposal generator in a particle filter seems to be the all over recommended solution as it is capable of making much more accurate estimates and is a much more robust solution in scenarios where assumptions (e.g. using Gamma process noise) and beliefs (e.g. using a false process model) are wrong causing other algorithms to fail.

Finally, we note that (PF)EKF is much more dependent of a good noise proposal than (PF)UKF.

Chapter 6

Watertank simulation

6.1 Problem outline

In this chapter we move on to a more complex problem in which we apply our filtering techniques to a model commonly used as a benchmark in the fault diagnostics community ([21]). The physical system is shown in Figure 6.1 and the 2T-DBN structure is shown in Figure 6.2.

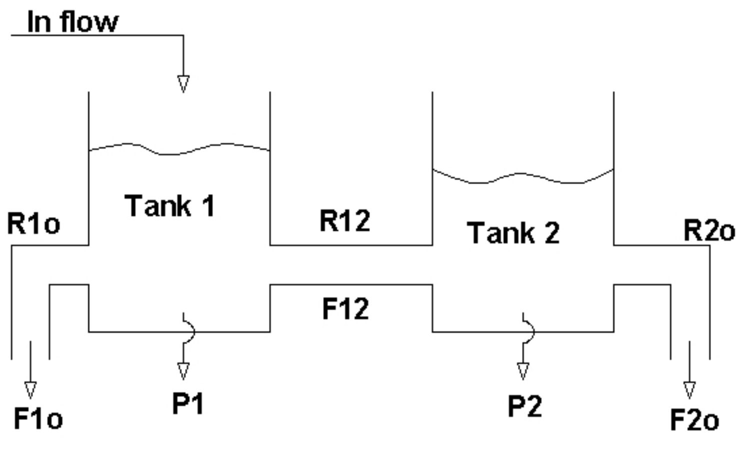


Figure 6.1: Physical illustration of the watertank system

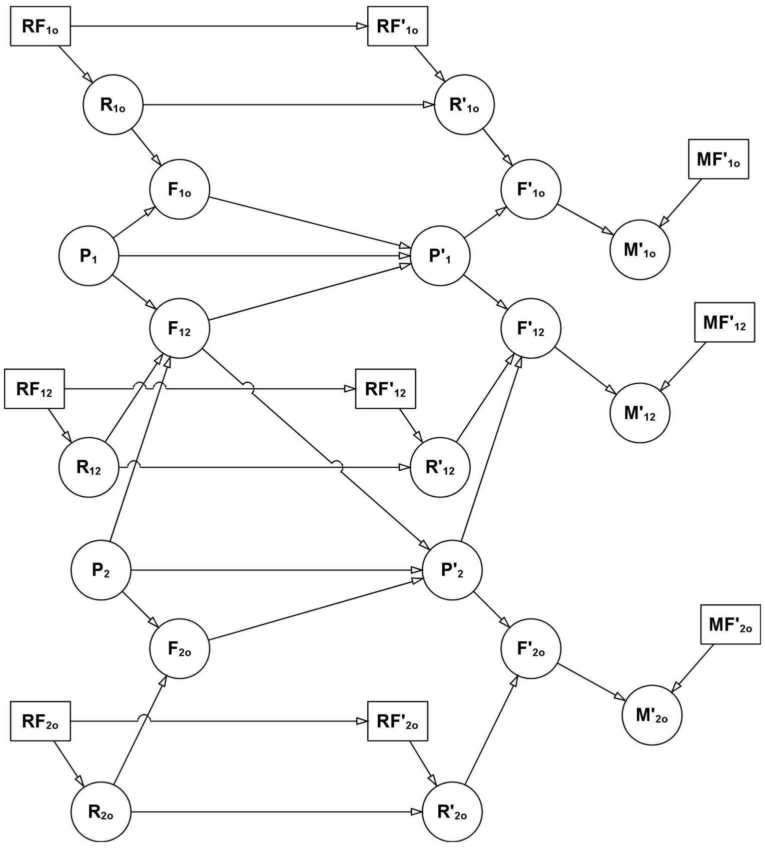


Figure 6.2: DBN for the watertank system

The system consists of two water tanks $Tank1$ and $Tank2$ in which the pressure is P_1 and P_2 resp. The tanks are connected via a pipe with resistance R_{12} and the flow in the pipe is F_{12}. Each tank has a pipe from which water flows out of the tank. The pipes have resistance R_{10} and R_{20} and the flow is F_{10} and F_{20} resp. The first tank also has a constant flow of water F_{in} going into it. We get measurements $M10$, $M12$ and $M20$ of the flows in each of the three pipes.

Flow, pressure and pipe resistance are related to each other as described in equation 6.1.

$$\begin{aligned} F_{10} &= \frac{P_1}{R_{10} * \rho * g} \\ F_{12} &= \frac{P_1 - P_2}{R_{12} * \rho * g} \\ F_{20} &= \frac{P_2}{R_{20} * \rho * g} \end{aligned} \tag{6.1}$$

where ρ is the density of the water and g is the gravity.

The dynamics of the physical system follows a set of differential equations corresponding to the mass balance equations:

$$\begin{aligned} dh_1 &= \frac{F_{in} - F_{10} - F_{12}}{A_1 \rho} \\ dh_2 &= \frac{F_{12} - F_{20}}{A_2 \rho} \end{aligned} \tag{6.2}$$

where dh_1 and dh_2 corresponds to small changes in the water level in tank 1 and 2 resp. A_1 and A_2 are the surface areas of tank 1 and 2.

dh_1 and dh_2 are proportional to dp_1 and dp_2:

$$\begin{aligned} dp_1 &= dh_1 \rho g \\ dp_2 &= dh_2 \rho g \end{aligned} \tag{6.3}$$

Combining equations 6.1, 6.2 and 6.3 provides us with a set of differential equations based on the pressure in tank 1 and 2:

$$\begin{aligned} dP_1 &= \frac{F_{in} g}{A_1} - \frac{P_1}{A_1 R_{10} \rho} - \frac{P_1 - P_2}{R_{12} \rho A_1} \\ dP_2 &= \frac{P_1 - P_2}{A_2 R_{12} \rho} - \frac{P_2}{R_{20} \rho A_2} \end{aligned} \tag{6.4}$$

Instead, an approximation of the physical system was used in which both pressures are updated based on the previous pressures allowing the dynamics of the system to be incorporated in a 2T-DBN.

The differential equations describe how flow, pressure and resistance are related in theory. Practically, the process and measurement models and the measurements themselves are noisy. Furthermore, there are three possible type of failures in the system:

Measurement failure Usually measurements are quite reliable, but in the case of a measurement failure, the measurement becomes extremely noisy. In this case the value of the measurement has almost nothing to do with the actual flow; thus, to track the system correctly, it is necessary to identify these measurement failures and ignore them

Pipe bursts A pipe can suddenly burst and change its resistance to some unknown value

Drifts The resistance of the pipe can drift, which gradually increases or decreases the pipe's resistance

The 2T-DBN modelling the two-tank system is shown in Figure 6.2 and follows the usual notation that discrete variables are depicted as squares or rectangles and continuous variables are depicted as circles or ellipsoids. The nodes labelled RF indicate faults in the resistance of the pipes (drifts or bursts) and the nodes labelled MF indicate measurement failures. The P, F and R nodes are continuous valued and indicate pressure, flow and pipe resistance resp. Finally, nodes labelled M indicate flow measurements and are observation nodes. All other variables are hidden, i.e. their values are not available at any time (in theory that is).

6.2 Implementation

Initially, we experiment using the true process and measurement models, i.e. the same relations as used in the data generation. The network structure which is intuitively plausible allows us to approximate the true differential equations expressed in terms of DBN relations.

6.2.1 Modeling issues

By explicitly modelling the pipe resistance, we accurately model the physical system. However, since the flow is the ratio between the pressure and the resistance, the system is not a linear system. Dealing with ratios is difficult, especially when the values are close to zero. Therefore instead of modelling the resistances we choose to model the conductances (the conductance is defined as the reciprocal of the resistance). This transformation results in products rather than ratios. The system is still non-linear, but this does not pose a problem for our algorithms, which are general enough to deal with non-linear CPD's.

The implemented process and measurement models are given below:

PROCESS MODEL

Flow in pipe 10 (from tank 1 and out)

$$F_{10}(new) = \frac{P_1(new) * C_{10}(new)}{\rho * g} \tag{6.5}$$

Flow in pipe 12 (from tank 1 towards tanks 2)

$$F_{12}(new) = \frac{(P_1(new) - P_2(new)) * C_{12}(new)}{\rho * g} \tag{6.6}$$

Flow in pipe 20 (from tank 2 and out)

$$F_{20}(new) = \frac{P_2(new) * C_{20}(new)}{\rho * g} \tag{6.7}$$

Pressure in tank 1

$$P_1(new) = P_1(old) + \frac{(F_{in} - F_{10}(old) - F_{12}(old)) * g}{A_{tank1}} \tag{6.8}$$

Pressure in tank 2

$$P_2(new) = P_2(old) - \frac{(F_{20}(old) + F_{12}(old)) * g}{A_{tank2}} \tag{6.9}$$

Conductance in pipe 10 (from tank 1 and out)

$$C_{10}(new) = \begin{cases} C_{10}(old) & \text{normal} \\ C_{10}(old) + drift & \text{positive drift} \\ C_{10}(old) - drift & \text{negative drift} \\ C_{10}(max) & \text{burst} \end{cases} \tag{6.10}$$

Conductance in pipe 12 (from tank 1 towards tanks 2)

$$C_{12}(new) = \begin{cases} C_{12}(old) & \text{normal} \\ C_{12}(old) + drift & \text{positive drift} \\ C_{12}(old) - drift & \text{negative drift} \\ C_{12}(max) & \text{burst} \end{cases} \tag{6.11}$$

Conductance pipe 20 (from tank 2 and out)

$$C_{20}(new) = \begin{cases} C_{20}(old) & \text{normal} \\ C_{20}(old) + drift & \text{positive drift} \\ C_{20}(old) - drift & \text{negative drift} \\ C_{20}(max) & \text{burst} \end{cases} \tag{6.12}$$

MEASUREMENT MODEL

Measurement of flow in pipe 10 (from tank 1 and out)

$$M_{10}(new) = \begin{cases} F_{10}(new) & \text{no measurement failure} \\ F_{10}(new) + error & \text{measurement failure} \end{cases} \tag{6.13}$$

Measurement of flow in pipe 12 (from tank 1 towards tanks 2)

$$M_{12}(new) = \begin{cases} F_{12}(new) & \text{no measurement failure} \\ F_{12}(new) + error & \text{measurement failure} \end{cases} \tag{6.14}$$

Measurement of flow in pipe 20 (from tank 2 and out)

$$M_{20}(new) = \begin{cases} F_{20}(new) & \text{no measurement failure} \\ F_{20}(new) + error & \text{measurement failure} \end{cases} \tag{6.15}$$

Note the difference between the measurement failure nodes and the pipe faults. The pipe faults are persistent and therefore appear both at time t and time $t+1$. In fact, the pipe faults are a part of the belief state. Measurement failures, on the other hand, are transient and therefore appear only at time $t+1$ and need not be included in the belief state. As a result, the network has six pipe fault variables and three measurement failure variables, leading to 32,768 different discrete states.

The physical system is modelled using the following restrictions:

- The flow into Tank 1 is constant
- Measurement failures occur with a constant probability
- Drifting (pos. or neg.) begins with a constant probability and persists with a higher, constant probability. Drifting is modelled by adding (or subtracting) a constant value from the previous resistance (conductance).
- Pipe bursts occur with a constant probability and is modelled by setting the given pipe resistance (conductance) to a specific value for the remaining time. To simplify things, the pipe connecting the two tanks can not burst. This would imply writing a new set of mass balance equations to model the physical system. As a consequence, the discrete variable state space is reduced to 18,432 states
- The model can not handle negative values (except for the flow between the two tanks which implies a flow from tank 2 towards tank 1) as this is not valid from a real-world perspective.

Unfortunately, this network is still far too complicated to be able to use exact inference. In fact, the belief state at time t is a mixture of Gaussian's with a number of mixture components that grows exponentially with t, and with the number of discrete variables in each time slice.

Suboptimally, we would like to sample all discrete indicator variables, i.e. conductance and measurement failures, which can be grouped into two vector-valued nodes $\boldsymbol{CF}_t$ and $\boldsymbol{MF}_t$ with dimensionality 3, and apply exact inference on the remaining continuous valued nodes, which we group into a single vector-valued node, $\boldsymbol{X}_t$ with dimensionality 8. The observation nodes are likewise grouped into a single vector-valued node $\boldsymbol{Y}_t$ with dimensionality 3 allowing a transformation of the fairly complicated network into a simple network as illustrated in Figure 6.3. The network corresponds exactly to the original network as all connections are expressed in form of the transition matrices $\boldsymbol{A}$ (dimensionality 8x8), $\boldsymbol{C}$ (dimensionality 3x3) and $\boldsymbol{B}$ (dimensionality 8x3) as described in equation 6.16.

$$\boldsymbol{X}_t = \boldsymbol{A} \cdot \boldsymbol{X}_{t-1} \tag{6.16}$$

$$\boldsymbol{CF}_t = \boldsymbol{C} \cdot \boldsymbol{CF}_{t-1} \tag{6.17}$$

$$\boldsymbol{Y}_t = \boldsymbol{B} \cdot \boldsymbol{X}_t \tag{6.18}$$

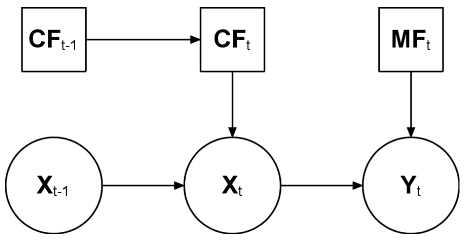


Figure 6.3: Simplified DBN for the watertank problem

This transformation allows us to sample part of the network and apply exact inference algorithms to the remaining part. A technique known as *Rao-Blackwellisation*.

Unfortunately, in this model, although the noise is Gaussian, the dynamics are non-linear, making it hard to integrate out $\boldsymbol{X}_t$. Hence, we apply our approximate inference techniques, Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), but we are no longer doing strict Rao-Blackwellisation. Furthermore, we apply particle filtering to both the discrete and the continuous valued nodes in form of the generic Particle Filter (PF), Particle Filter with Metropolis-Hastings step (PFMC), Particle Filtering with EKF proposal (PF-EKF) and Particle Filtering with UKF proposal (PF-UKF).

Particle Filtering with EKF proposal and Metropolis-Hastings step (PF-EKFMC) and Particle Filtering with UKF proposal and Metropolis-Hastings step (PF-UKFMC) were not applied to the watertank problem. Based on the observations in chap. 5 and the fact that we are working on a first order system with Gaussian noise we found the increased computational complexity of these two filters to be too large compared to our expectations of increased estimation accuracy.

6 programs were created in Matlab implementing the 6 algorithms mentioned above and named accordingly. All programs except PF and PFMC are designed as a two-step serial process. The first process samples the discrete nodes using a PF algorithm, but without updating the continuous state variables. The continuous states are then estimated (for each particle) in the second process using EKF, UKF, PFEKF or PFUKF. This two-step process was used as all the EKF and UKF based algorithms were able to give good estimates of the continuous nodes based on poor estimates of the discrete nodes due to the correction step, see sec. 4.1 and 4.6.1. Merging the two processes into one process updating both discrete and continuous variables simultaneously would make tracking of the conductance and measurement failures very hard as e.g. a particle with a wrong combination of conductance and measurement failures could still give a better state mean estimate than a particle with the true conductance and measurement failure combination. PF and PFMC do not use an EKF or UKF proposal and hence we update both discrete and continuous variables simultaneously.

The pseudo-code for the modified particle filter used to sample the discrete failure nodes is given below:

Modified Particle Filter

```
1. Initialization
    • For $i = 1, 2, ..., N$
        - initialize $\mathbf{x}_{t=0}^{(i)}$, $\mathbf{CF}_{t=0}^{(i)}$ and $\mathbf{MF}_{t=0}^{(i)}$.
2. For t = 1, 2,...
    (a) Importance sampling step
        • For $i = 1, ..., N$
            - sample $\hat{\mathbf{CF}}_t^{(i)} \sim q(\mathbf{CF}_t|\mathbf{CF}_{t-1})$ and set $\hat{\mathbf{CF}}_{0:t}^{(i)} \triangleq (\mathbf{CF}_{0:t-1}^{(i)}, \hat{\mathbf{CF}}_t^{(i)})$
            - sample $\hat{\mathbf{MF}}_t^{(i)} \sim q(\mathbf{MF}_t)$ and set $\hat{\mathbf{MF}}_{0:t}^{(i)} \triangleq (\mathbf{MF}_{0:t-1}^{(i)}, \hat{\mathbf{MF}}_t^{(i)})$
            - sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \hat{\mathbf{CF}}_t^{(i)}, \hat{\mathbf{MF}}_t^{(i)})$ and set $\hat{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \hat{\mathbf{x}}_t^{(i)})$
            - evaluate the importance weights up to a normalizing constant
                  $w_t^{(i)} = p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{y}_t)$
            - normalize the importance weights
                  $\tilde{w_t}^{(i)} = w_t^{(i)} \left[\sum_{j=1}^{N} w_t^{(j)}\right]^{-1}$
    (b) Selection step (resampling)
        • Multiply/suppress samples $\hat{\mathbf{x}}_{t-1}^{(i)}$,$\hat{\mathbf{CF}}_t^{(i)}$ and $\hat{\mathbf{MF}}_t^{(i)}$ with high/low importance
          weights $\tilde{w_t}^{(i)}$ resp.
    (c) Estimate continuous state variables using an EKF or UKF based algorithm
        • For $i = 1, ..., N$
            - estimate $\hat{\mathbf{x}}_t^{(i)}$ based on $\hat{\mathbf{x}}_{t-1}^{(i)}$,$\hat{\mathbf{CF}}_t^{(i)}$ and $\hat{\mathbf{MF}}_t^{(i)}$ using an EKF or UKF
              based algorithm
```

6.3 Results

Using the true process and measurement models and the true initial values the algorithms are expected to perform well. This is a nice way of testing the validity of the algorithms: If they do not perform well using the true process and measurement models they are hardly expected to perform well using other model proposals.

Obviously, the choice of process and measurement model is most critical. In a real-life simulation as opposed to simulations, the true models are not known (and may not exist if the data generation can not be expressed as a 2T-DBN) and need to be approximated using a variety of techniques, e.g. using statistical tools. At the end of the day, one needs to do a thorough a priori study of the problem and the data in order to come up with a valid network structure and process/measurement models.

6.3.1 Parameters

There is a number of parameters which influence the filtering algorithms ability to estimate the continuous state variables and track the discrete conductance and measurement failures. These parameters can be divided in two categories. One set of parameters which change the physical problem making prediction easier or harder and one set of parameters which influence the algorithms ability to deal with the given problem.

System parameters

In PFEKF and PFUKF the particles are weighted according to a ratio between the likelihood, the prior and the proposal of a given particle, see equation 4.58 and 4.84.

For a multi-dimensional Gaussian distributed state and observation variable with covariance matrices $\mathbf{\Sigma}_x$ and $\mathbf{\Sigma}_y$ resp. the likelihood, prior and proposal are calculated using the following equations in which $\boldsymbol{P}_t^{xx}$ is the estimated covariance matrix of the state variables at time t:

$$\begin{aligned}
p(\boldsymbol{y}_t|\boldsymbol{x}_t) &= \frac{1}{|\mathbf{\Sigma}_{\boldsymbol{y}}|^{1/2}} e^{\left((\boldsymbol{y}_t-\bar{\boldsymbol{y}}_t)\cdot\mathbf{\Sigma}_{\boldsymbol{y}}\cdot(\boldsymbol{y}_t-\bar{\boldsymbol{y}}_t)^T\right)} \\
p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) &= \frac{1}{|\mathbf{\Sigma}_{\boldsymbol{x}}|^{1/2}} e^{\left((\boldsymbol{x}_t-\boldsymbol{x}_{t-1})\cdot\mathbf{\Sigma}_{\boldsymbol{y}}\cdot(\boldsymbol{x}_t-\boldsymbol{x}_{t-1})^T\right)} \\
p(\boldsymbol{x}_t|\bar{\boldsymbol{x}}_t) &= \frac{1}{|\boldsymbol{P}_t^{xx}|^{1/2}} e^{\left((\boldsymbol{x}_t|\bar{\boldsymbol{x}}_t)\cdot\boldsymbol{P}_t^{xx}\cdot(\boldsymbol{x}_t|\bar{\boldsymbol{x}}_t)^T\right)}
\end{aligned}$$

In the watertank simulation, the prior was often below the numerical precision of Matlab. To avoid underflow 1e-99 was added. The prior is calculated based on the difference between the state variables at time $t-1$ and time t. Large changes in the state variables from time step to time step implies a small prior, especially if the covariance matrix is small. In the watertank simulation, pressure and conductance undergo large changes in the beginning of the simulation until an equilibrium is reached yielding underflow in the computation of the prior. Typically, all prior calculations at a given time step were 'saved' by adding 1e-99 and the weights were thus based on the likelihood and the proposal only. If all the priors have the same value they will not have any effect on the weights as these are numerated. One way to deal with the prior problem is to evaluate the weights based on the log values of the likelihood, prior and proposal.

$$w_t^{(i)} \propto log\left(p(\boldsymbol{y}_t|\hat{\boldsymbol{x}}_t^{(i)})\right) + log\left(p(\hat{\boldsymbol{x}}_t^{(i)}|\boldsymbol{x}_{t-1}^{(i)})\right) - log\left(q(\hat{\boldsymbol{x}}_t^{(i)}|\boldsymbol{x}_{0:t-1}^{(i)}, \boldsymbol{y}_{1:t})\right) \tag{6.19}$$

However, this non-linear transformation of the weights left no chance of returning to the true weights as these values also turned out to be below the numerical precision of Matlab due to the very low prior values.

One way of avoiding the underflow problem was to change the system so that the changes of the state variables for each time step were reduced. This could easily be done by increasing the resistance in the pipes reducing the changes in flow and hence pressure. Increasing the resistance corresponds to decreasing the conductance which would effect the conductance drifts as they would have to be decreased to avoid negative conductance. Restrictions to keep the conductance positive would complicate matters unnecessarily leaving the filtering algorithms with a small change of estimating the true state space values and tracking conductance failures.

A small drift also implies the use of a smaller conductance process noise level to keep a fair signal-to-noise ratio leaving the filtering algorithms with a fair chance of distinguishing conductance drifts from noise. A smaller conductance process noise level unfortunately also resulted in underflow problems as the proposal and prior are both calculated using the inverse of the state covariance matrix (see eq. 6.19). To avoid the low conductance process noise level, a scaling parameter was introduced in the process and measurement models scaling the actual value of the conductance used in the calculation of the pressure and flow resp. This allowed the conductance to have values in the same domain as the pressure and flow and thus a small conductance process noise level was avoided. Increasing the conductance scaling parameter corresponds to decreasing the conductance without actually changing the conductance node values and without changing the conductance process noise level.

Initially, the algorithms are given optimal conditions in the sense that they are using the exact same models as was used to generate the data. Sections 6.3.6 and 6.3.7 describe what happens when the algorithms are not given the optimal conditions. But for now, all filtering algorithms use the true process and measurement models and the same parameters as used in the data generation. The problem is still fairly complex as changes in flow can be explained by a number of events: Conductance failure causing a drift or a burst, measurement failure, process and measurement noise.

The probability of a pipe starting to drift and the probability of a drift persisting effects the complexity of the problem. A high probability of drifting and a low probability of persisting obviously make the problem more difficult with drifting frequently beginning and ending. It is not hard to imagine a number of wrong conductance failure combinations being able to explain a change in the flow almost as well as the true conductance failure combination.

In the case of a conductance drift, a constant value is added or subtracted from the previous conductance value at time $t-1$ to give the conductance value at time t. This constant value should be compared with respect to the process noise levels as it would be nearly impossible to track a drift if the change in conductance is smaller than the noise level. In that case, the filtering algorithms would just explain the conductance drift with noise. In order to recognize conductance failures, the drifting value has to be (much) larger than the process noise levels. Of course, a large measurement noise also makes it hard to recognize a small change in conductance caused by drifting.

The change in flow caused by a conductance drift will also be more obvious for large flow values as the flow is the product of the pressure and the conductance. The watertank is initialized to contain a much larger amount of water than the amount of water corresponding to the equilibrium where the flow output is equal to the flow input in $Tank1$. This means that the flow changes are large in the beginning of the simulation and then decreases until an equilibrium is reached were the flow only varies due to conductance failures, measurement failures and noise. The length of the simulation thus becomes important as prediction and tracking becomes harder due to the decreased flow change. A short simulation time might give good results whereas the results are poor for a long time period. The simulation time was chosen to allow the system to reach its equilibrium phase.

When a pipe bursts, the pipe conductance is set to a maximum value. This makes it very easy to track a burst as this event is not easily explained by other events and visa versa, unless the conductance actually drifts close to the maximum conductance during the simulation or the flows are close to zero. A high probability of a pipe burst makes the problem easier as it is easy to track a bursts and once a pipe bursts, the pipe remains in that state reducing the number of possible failure combinations. However, if a pipe bursts, the flow increases dramatically and may cause the system to reach another equilibrium very fast making tracking of the discrete failures very hard.

Measurement failures occur with a constant probability and when a failure occurs, a constant value is added to the true measurement node. This event can be explained by almost any other event which implies that the constant value added to the true flow measurement has to be a fairly large number to give the algorithms a fair chance of recognizing measurement failures. As mentioned earlier, noise is added to both the hidden nodes (process noise) and the observation nodes (measurement noise). If the noise levels are too high, tracking of measurement and conductance failures would be impossible. All in all, there are a number of parameters to play around with making the problem easier or harder.

In all experiments the following event probabilities were used:

- The probability of a measurement failure: 0.03
- The probability of a pipe burst: 0.01
- The probability of a drift beginning: 0.1
- The probability of a drift persisting: 0.9

This choice of probabilities ensured that a lot of events (drift, burst, measurement failure) happened even for relative small time periods.

6.3.2 Initialization

The watertank problem consists of two watertanks which are initialized with a certain amount of water at time $t = 1$ (see Figure 6.1). The amount of water determines the pressure in the bottom of each watertank labelled $P1$ and $P2$ in Figure 6.1. The conductance had to be initialized with restrictions to ensure that the conductance would not drift below zero or drift over a preset maximum conductance value corresponding to a pipe burst. It does not make any sense to have negative conductance which would imply that the pipes were able to suck in water. If a pipe bursts, the conductance is set to a maximum value which should not be possible to reach by simple drifting during the simulation leaving the filtering algorithms with a less than fair chance of distinguishing between a burst and a drift. At initialization point there is no flow in any of the three pipes and there is no water flowing into $Tank1$. At time $t = 2$ all the pipes are 'opened' and water flows out of the two tanks, between the tanks and into $Tank1$.

No flow activity makes it impossible for the filtering algorithms to track conductance failures. If a certain event does not have any influence on the flow (which is the case if the flow is zero), it is impossible to detect the event as the algorithms attempt to estimate the states based on the flow measurements. In general, low flow values makes it hard to estimate the states and high flow values makes tracking easier. If the amount of water in the two tanks is initialized to the same value there will be no pressure difference between the two watertanks and no water would flow between the tanks at initialization point. This could of course also happen doing the simulation whenever the flow between the tanks chances direction.

It would also not make any sense to have negative conductance or negative pressure. After generating the data a data check was performed to ensure that conductance and pressure values were all positive.

6.3.3 Network design

In the proposed network structure, the flow nodes are the only nodes directly connected to the observation nodes (see Figure 6.1). This design favors a good estimation of the flow over the pressure and conductance using EKF and PF.

The Kalman gain using EKF is partly based on the Jacobian of the measurement model (see equation 4.6). As the Jacobian is calculated by taking the derivative of the measurement model with respect to the 8 continuous state variables, only the flow variables will be represented in the Jacobian. Even though pressure and conductance are highly correlated with the flow, the Kalman gain only influences the flow estimates. As pressure at time t is calculated using pressure and flow at time $t-1$, a good flow estimate corresponds to a good pressure estimation. However, this is only guaranteed if the pressure nodes are initialized correctly and if there is no noise added to the pressure in the process model. A poor pressure initialization leaves EKF with little chance of correcting this later on as well as it has little chance of correcting for the noise. Same story goes for the estimation of the conductance although it is not calculated based on the flow. The conductance at time t is only based on the conductance at time $t-1$ and the conductance failure at time t. Assuming the right conductance failures are chosen in the PF step, EKF will produce fairly good estimates of the conductance if it is initialized well and if the level of noise is low. Otherwise, EKF will not be able to correct for a poor initialization or noise. Figure 6.4 shows the relative RMSE of the flow, conductance and pressure estimates for a typical run using EKF with correct initialization of all state variables. Both process and measurement noise samples were drawn from a Gaussian distribution $\mathcal{N}(0, 0.1)$ which is a fairly high noise level compared to the drifting factor of 2. Figure 6.5 shows the results of the same experiment, but with an initialization 10% off the true initial values. 150 time steps and 50 particles were used in both experiments.

Figure 6.4 illustrates that EKF is making poor conductance and pressure estimates whereas the flow estimates are very accurate for all time steps. The relative RMSE for conductance and pressure seems to increase with an almost constant slope corresponding to the noise added in each time step in the data generation, i.e. the conductance and pressure estimates are slowly drifting away from the true values as EKF can not correct for the noise. Figure 6.5 shows that EKF only accurately estimates the flow nodes after a bad initialization of all state variables.

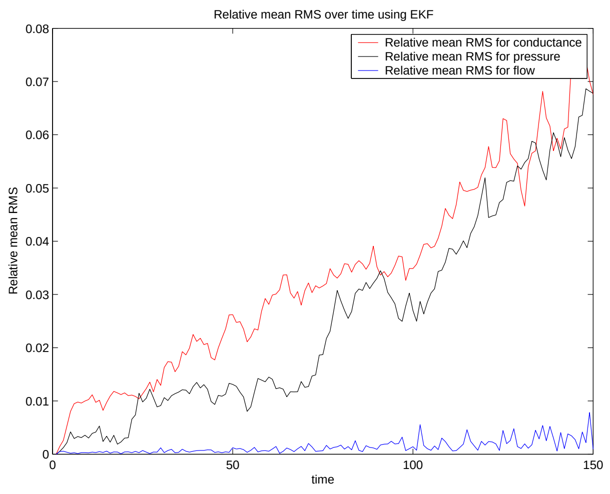


Figure 6.4: Relative RMSE for conductance (red graph), pressure (black graph) and flow (blue graph) estimates for a typical run using EKF with correct initialization. EKF only makes good estimation of the flows.

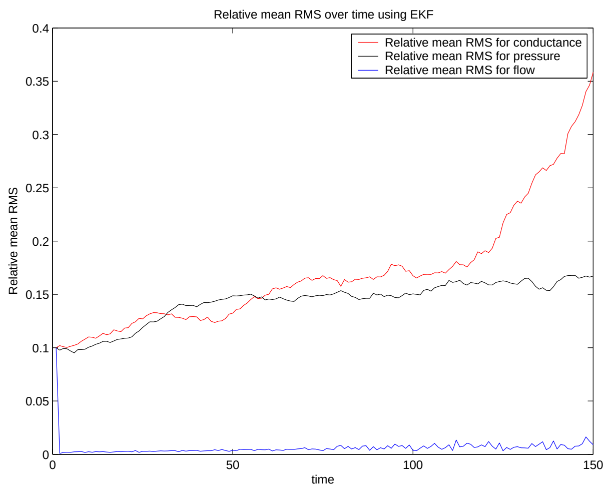


Figure 6.5: Relative RMSE for conductance (red graph), pressure (black graph) and flow (blue graph) estimates for a typical run using EKF with an initialization 10% off the true initial values. Notice that EKF only correct the flow after a bad initialization of all variables

In PF, a number of particles are produced by sampling from the true process model (true prior). In each time step, all particles are weighted according to their likelihood using equation 4.46 and then normalized. The likelihood is based on the difference between the true and predicted values of the observation nodes. With only the flow nodes connected to the observation nodes, a particle with correct flow values will thus give a high likelihood and a corresponding high weight regardless of whether the particle has poor conductance or pressure estimates. This problem is illustrated in figure 6.6 which shows how 10 particles are weighted for a given time step using PF. The plot shows three different weights for the particles: The weights used in PF (blue bars), the optimal weights based on the distance from the true continuous state values (green bars) and the weights based on the distance to the true flow values (red bars).

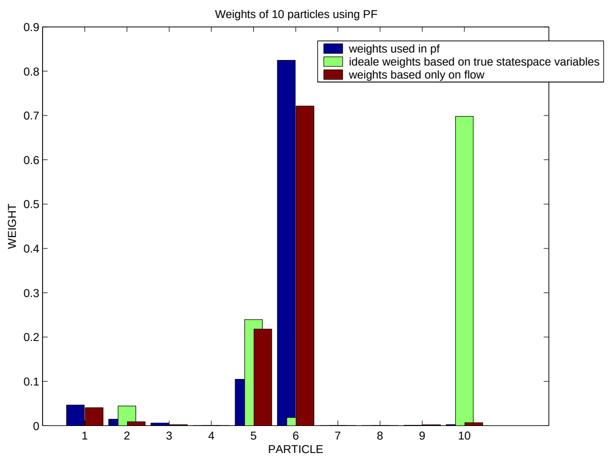


Figure 6.6: The weights used in PF (blue bars), the optimal weights based on the distance from the true continuous state values (green bars) and the weights based on the distance to the true flow values (red bars) for 10 particles using PF

The weights used in PF follow the weights based on the flow values and not the optimal weights. A large process noise would make this problem even worse.

As mentioned, both PF and EKF make poor estimates of the continuous state variables that are not directly connected to the observation nodes. This is not the case in UKF which makes good estimates of all continuous state variables regardless of the network structure. Instead of using the Jacobian of the process and measurement models, the Kalman gain in UKF is based on a number of sigma points that are propagated trough the network using the true process and measurement models capturing the true posterior mean and covariance up to second order (see 4.2), which are then used to calculate the Kalman gain (see 4.3.1). UKF uses the true non-linear models and approximates the distribution of the state random variable whereas EKF approximate the non-linear process and observation models. Hence, UKF is able to capture the true covariance up to second order between all continuous state variables and observation nodes and incorporate this in the Kalman gain. Both pressure and conductance are highly correlated with the observation nodes, even though they are not directly connected, which makes UKF able to updates all continuous state variables.

With the current network, PF and EKF make accurate estimates of the flow values, but poor pressure and conductance estimates. One of the objectives in the watertank problem is to track conductance failures making accurate estimation of the conductance a crucial point. With this in mind, a new network was proposed by eliminating the flow nodes from the continuous state variables allowing conductance and pressure nodes to be directly connected to the observation nodes (see Figure 6.7).

The flow measurements are still used as observations nodes, but instead of adding noise to the flow nodes, the observations are now predicted using the calculated flow based on pressure and conductance plus noise. The equations used to calculate the flow in the old network were then changed into the measurement model in the new network. In summary, the new network introduces three major changes:

- The dimension of the continuous state space is reduced from 8 to 5 nodes making the Matlab programs more robust with regards to overflow and underflow issues and also a lot faster

- Connecting all continuous state variables with the observation nodes allows for better estimation of all continuous state variables using EKF and PF

- The level of noise added to the data is reduced

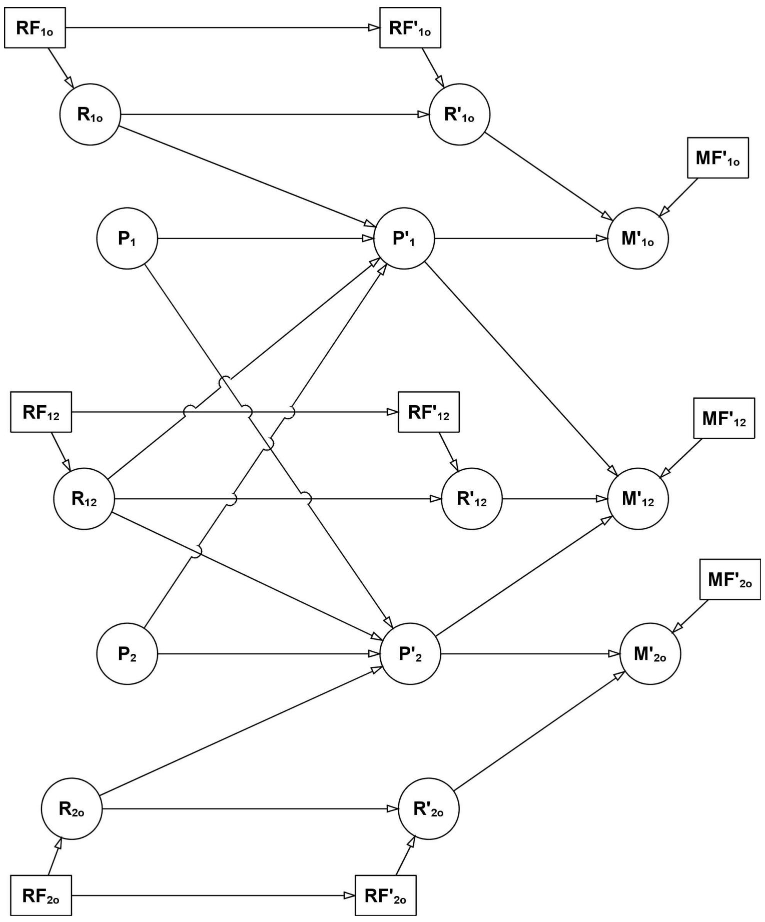


Figure 6.7: Proposed 2T-DBN for the watertank problem eliminating the flow nodes from the continuous state variables allowing conductance and pressure nodes to be directly connected to the observation nodes

The first two points improve the modelling whereas the third point actually makes the problem simpler. When data is generated using the old network, noise is added to the flows making the data more noisy than data generated in the new network. In order to compare the two networks, the old network was used to generate data used in both networks. The flow values were simply removed from the data when used in the new network. One would expect the old network to make more accurate estimates than the new network using a data set generated by itself. Hence, if the new network performs better on a data set generated by the old network, it is definitely the obvious choice of network.

Figure 6.8 shows the average RMSE for the conductance and pressure estimates from the two networks using PF, EKF and UKF. The results are based on 10 different data sets using 10 runs for each data set. The process and measurement noises samples were drawn from a Gaussian distribution $\mathcal{N}(0, 1e-1)$ which is fairly high compared to a drifting factor of 2.

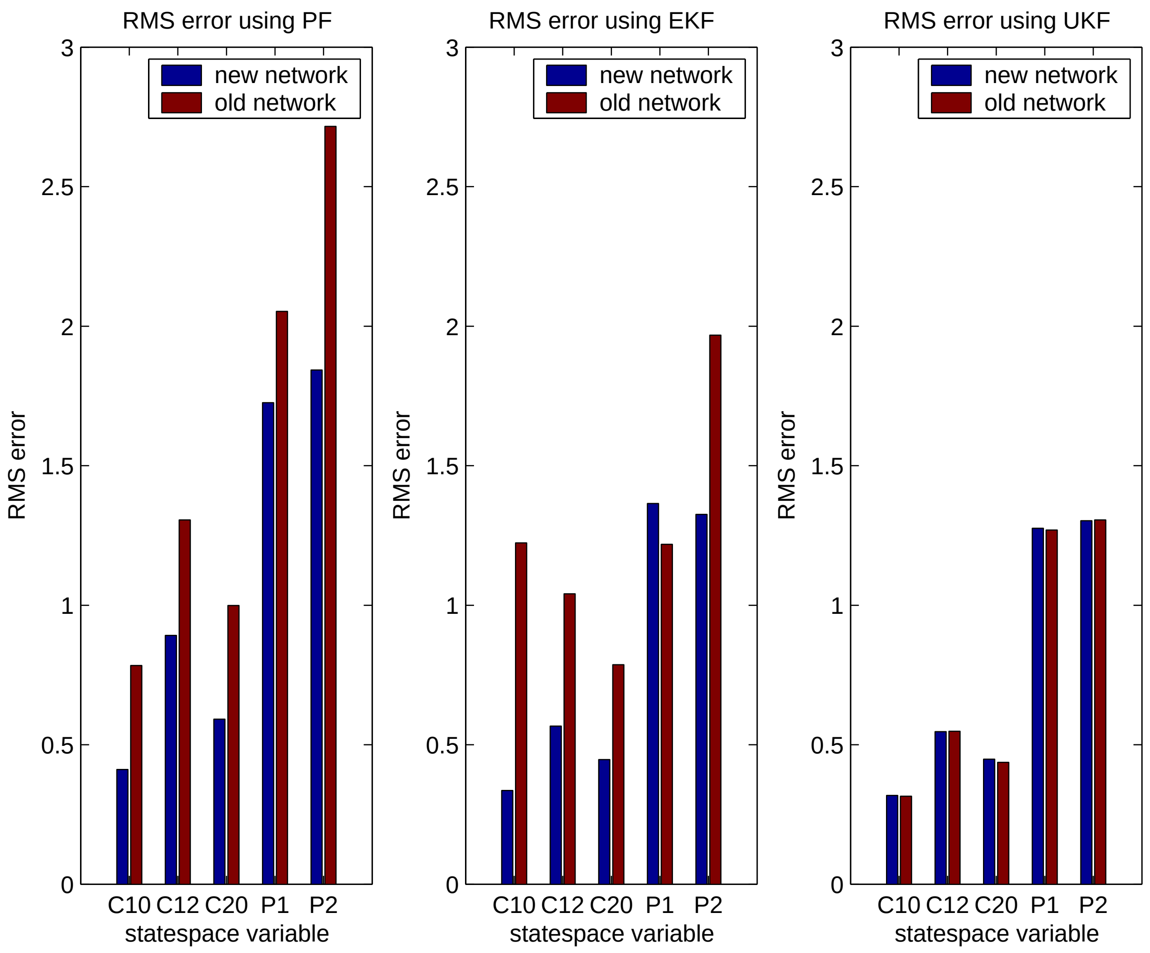


Figure 6.8: Average RMSE for conductance and pressure estimates using the old network (red bars) and the new network (blue bars) with PF, EKF and UKF resp.

As expected, the new network outperforms the old network for all continuous state mean estimates using PF and EKF (see Figure 6.8). The difference in performance using EKF becomes even more obvious using a poor initialization of the pressure and conductance values. The performance of UKF does not depend on the network structure and the two networks perform equally well.

Figure 6.9 and 6.10 show the estimates of the pressure and conductance variance over time for both networks using UKF. Both variance estimates were based on the same data set generated by the old network. The two variance estimates have similar shape, but are shifted horizontally due to the extra noise added to the flow in the old network.

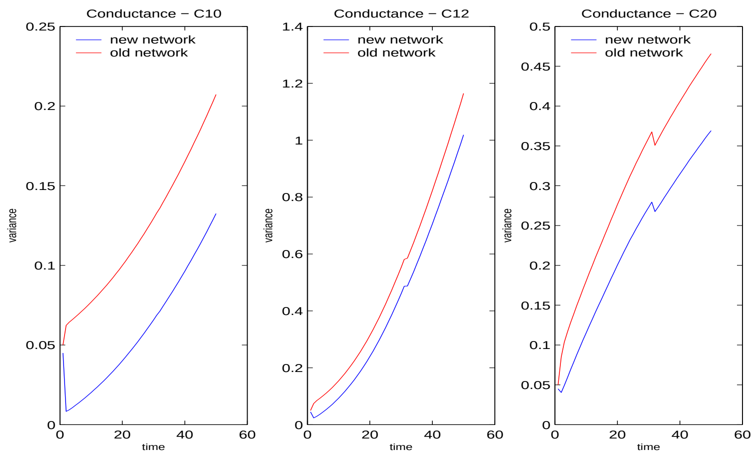


Figure 6.9: Estimates of conductance variance using UKF in the old and the new network

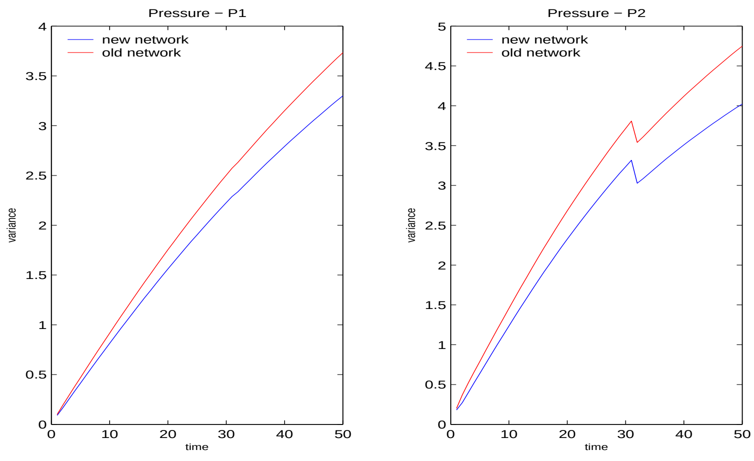
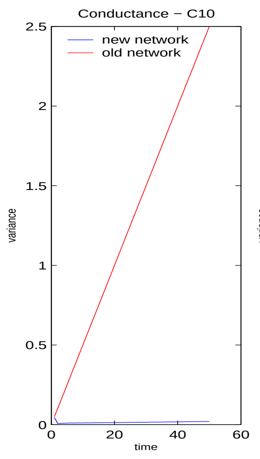
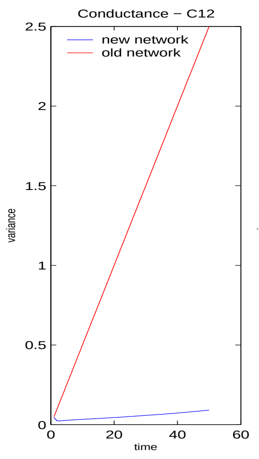


Figure 6.10: Estimates of pressure variance using UKF in the old and the new network

Figure 6.11 and 6.12 show the estimates of conductance and pressure variance resp. of both networks using the EKF algorithm. The pressure and conductance variance estimates increase with a constant slope using the old network whereas the new network design makes EKF capable of tracking the conductance and the pressure.





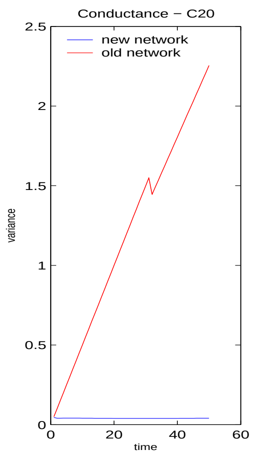
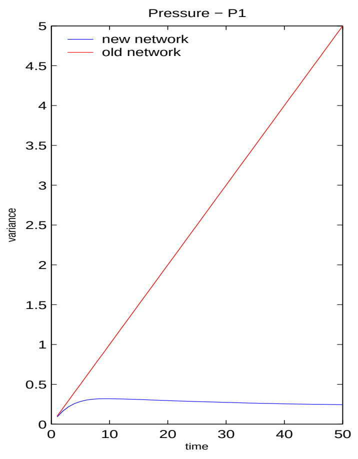


Figure 6.11: Estimates of conductance variance using EKF in the old and the new network



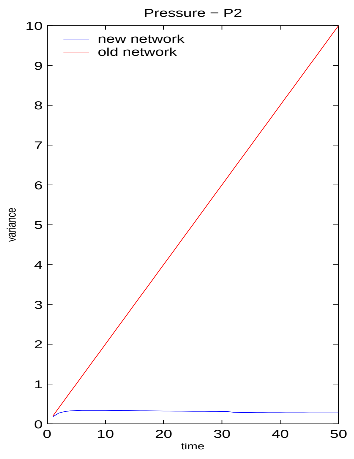


Figure 6.12: Estimates of pressure variance using EKF in the old and the new network

All in all, the change of network has no real effect on the performance of UKF. The performances of EKF and PF, however, are increased for all continuous state mean estimates even though the old network was used to generate the data. Based on the experiments, the new network structure was preferred and used in all forthcoming experiments.

Furthermore, we noticed how UKF was once again superior compared to EKF and in all forthcoming experiments the three EKF based implementations were discarded. Based on the simple simulation in 5 and the presented experiment showing how the network structure is critical for EKF we see no further reason to include the EKF based implementations also having in mind that later on we will introduce more noise and a false process model. In the simple simulation it was shown how the UKF based algorithms were much less affected by these changes than the EKF based algorithms. Having settled for a network structure and the process/measurement models, our implementations still leaves us with a variety of parameters to adjust.

These are outlined below and are treated in the forthcoming sections.

Number of particles Number of particles used in the particle filtering algorithms

Noise Process and measurement noise

Process and measurement models Proposed process and measurement models

Furthermore, in all experiments the values $\alpha = 1, \beta = 0$ and $\kappa = 0$ were used, as the choice of these parameters did not seem to influence the results significantly (as we are working on a first order system). Same reasoning goes for the choice of resampling algorithm and hence residual resampling was chosen arbitrarily (they are all $O(N)$ algorithms, [26]).

6.3.4 Deeper insight

Before we continue our experiments, we would like to give the reader a little more insight into how the particle filters using UKF as proposal generators work in practice in our simulation.

Before we go into details with the algorithm some implementation issues are discussed. First, a straightforward implementation of the PFUKF algorithm was done by simply drawing one sample from the UKF proposal for each particle from the modified PF step and then performing an extra importance sampling step based on the likelihood, prior and the proposal for each particle (see equation 4.84). This turned out to be crucial for the estimation of the discrete failure nodes as PFUKF was able to make good state estimates for particles with a wrong discrete failure combination.

Now, we could have chosen to update only the continuous state variables in the PFUKF part, but this implied completely breaking the link between the continuous state variables and the discrete failure nodes. A wrong combination of discrete failure nodes can of course still lead to poor state estimates using PFUKF and visa versa. The link between the continuous and discrete nodes are lost if we perform a separate updating step for the continuous variables and one of the main objectives is to be able to track failures.

Instead, a different approach was used in which the discrete and continuous state variables for each particle were updated independently: A set of so called subparticles were drawn for each particle and the PFUKF algorithm was applied to each particle independently. The UKF algorithm estimates the mean and covariance matrix for each subparticle and these values form the proposal distribution from which a new subparticle is drawn. The subparticles are then resampled according to the importance ratio. Thus, a particle is defined by a set of subparticles. In the modified PF step, the mean of the resampled subparticles is used to generate a new particle. Figure 6.13 illustrate the different steps in our approach. Notice that even though the mean of the subparticles is used in the modified PF step, each particle saves the subparticles.

Figure 6.14, 6.15 and 6.14 illustrate the steps in the PFUKF algorithm for a given time t. The plots show the pressure P1 and conductance C10 for a particle i using PFUKF. The figure contains 6 plots which evolves from plot to plot illustrating the different steps in the PFUKF algorithm.

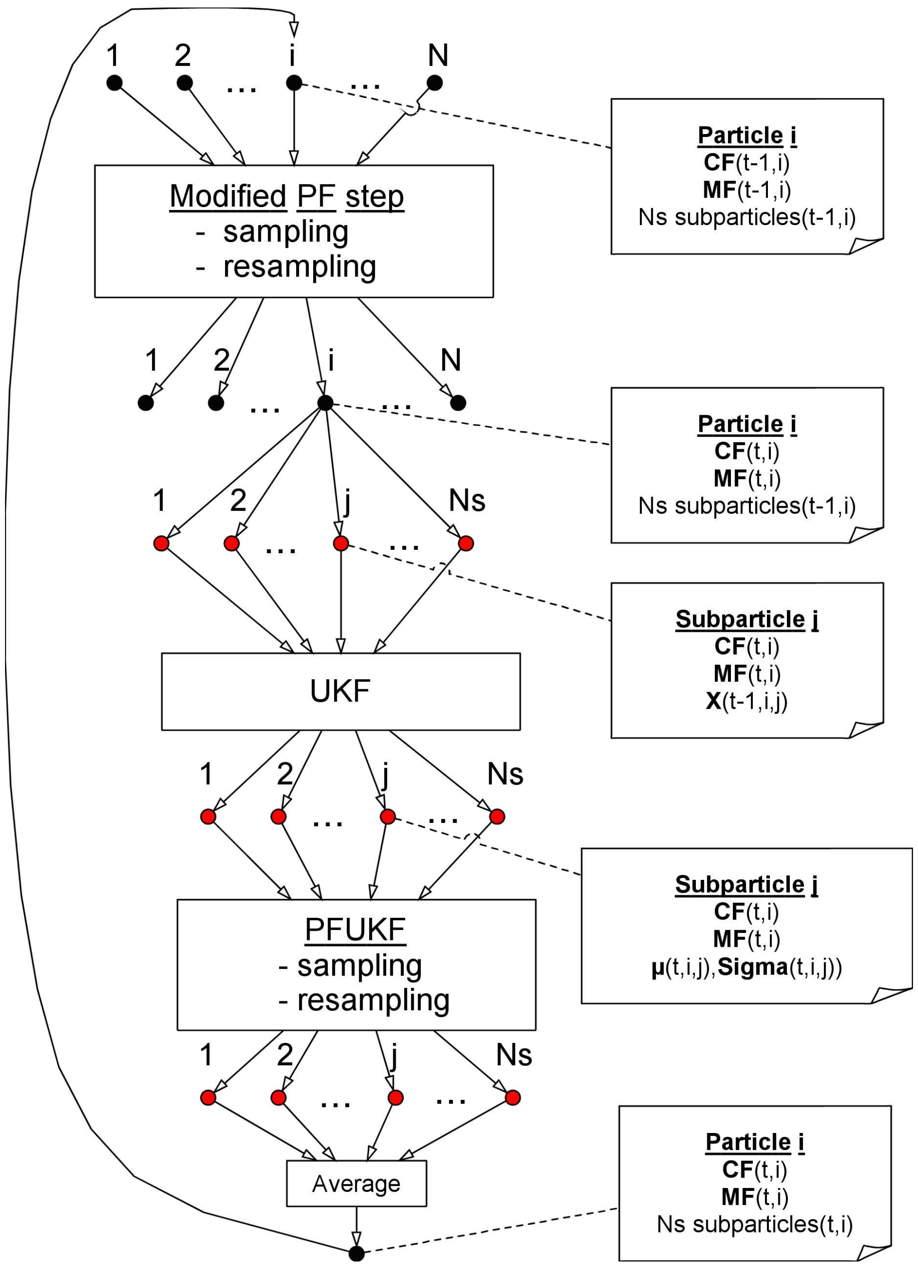


Figure 6.13: Illustration of the different steps in the PFUKF algorithm and serve to give a better understanding of how the subparticles are used

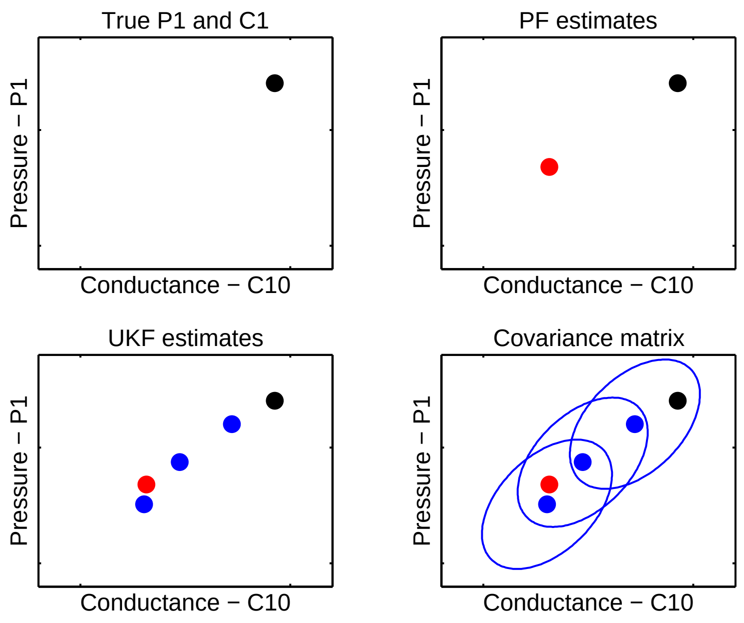


Figure 6.14: Steps in the PFUKF algorithm (black dot = true state value, red dot = PF state estimate, blue dots = UKF state estimates, blue ellipse = 95% confidence interval contour of P1 and C10)

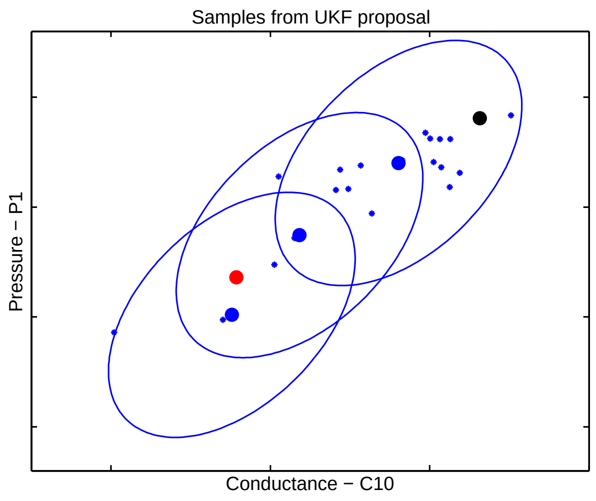


Figure 6.15: Subparticles (blue stars) drawn from a Gaussian distribution for a particle using PFUKF with mean values (blue dots) and 95% confidence interval contours (blue ellipsoids) estimated by a separate UKF. The black dot is the true state value and the red dot is the PF state estimate.

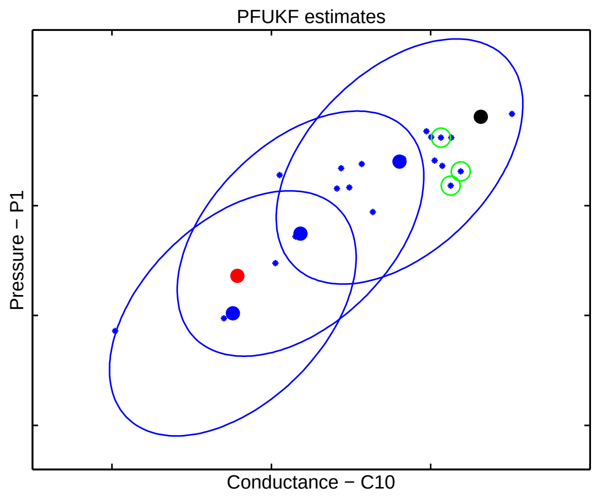


Figure 6.16: The resampling scheme selects three subparticles (indicated with a green circle) to represent the next set of subparticles for particle *i*. Blue stars = subparticles drawn from UKF proposal distributions with mean values (blue dots) and 95% confidence interval contours (blue ellipsoids), red dot = PF state estimate, black dot = true state value

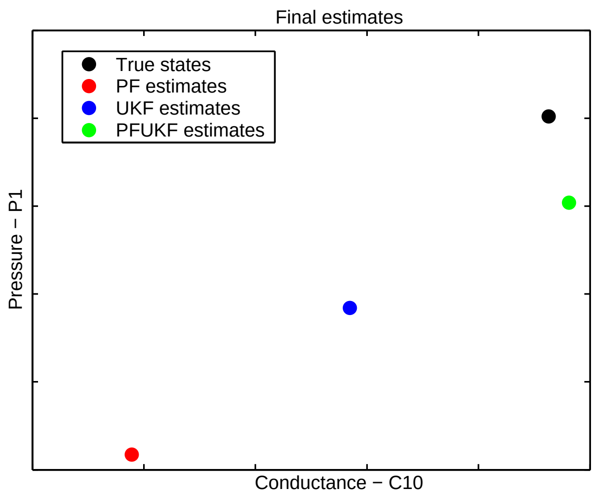


Figure 6.17: True state value (black dot), PF state estimate (red dot), UKF state estimate (blue dot) and PFUKF state estimate (green dot) for one time step

Step 1 The first plot in Figure 6.14 shows the pressure P1 and conductance C10 for particle i at time step t represented by a black dot. The objective is to come as close as possible to the true state (black dot).

Step 2 The first step in the algorithm samples the particle with respect to the discrete failure nodes. This is done using a modified PF algorithm (see section 6.1). The discrete failure nodes are sampled for each particle and used to calculate new continuous state estimates. The particles are then resampled according to their likelihood, but instead of using the estimated continuous states for the resampled particles, a PFUKF algorithm is used to make better continuous state estimates for the resampled particles. So it is actually the continuous state estimates at time $t-1$ which are resampled in the modified PF step with the new samples of the discrete failures at time t. The red dot in plot 2, Figure 6.14 represents the estimate of P1 and C10 for particle i at time step t if we had used the resampled state estimate for time t.

Step 3 Each particle corresponds to a set of subparticles which are used in the following steps. First, a separate UKF step is used to generate a Gaussian proposal distribution for each subparticle. The UKF algorithm estimates the mean and covariance matrix of the importance distribution for each subparticle based on the subparticle at time $t-1$. The mean of the subparticles corresponding to one particle i is shown in plot 3, Figure 6.14 (blue dots). In this example, only three different subparticles are used.

Step 4 PFUKF approximates a Gaussian proposal distribution for each subparticle based on the mean values (blue dots) and a covariance matrix proposed by the UKF step. The 95% confidence interval contours are illustrated with blues ellipsoids for each subparticle in plot 4, Figure 6.14.

Step 5 Figure 6.15 shows the next step in the PFUKF algorithm for particle i. Each subparticle is represented by an approximated Gaussian distribution from which a single subparticle is drawn. The subparticles are marked as blue stars.

Step 6 The final step for particle i is shown in Figure 6.16. The new subparticles are weighted according to the importance ratio and resampled, see eq. 4.58. In this example the resampling scheme selects the three subparticles marked with green circles to represent the next set of subparticles for particle *i*. Again, there are only tree different subparticles representing particle i.

The process illustrated in Figure 6.14, 6.15 and 6.16 is applied to all particles independently. Figure 6.17 shows the mean state estimate of P1 and C10 of all particles for the different steps in the PFUKF algorithm. The PF step (red dot), UKF step (blue dot) and PFUKF (green). For this particular time t, PFUKF manages to move the particle towards the true values of P1 and C10 (black dot), i.e. PFUKF outperforms both the PF-step and the UKF-step in the estimate of P1 and C10 at time t.

6.3.5 Number of particles

As mentioned, all implementations of the UKF based algorithms use a separate modified PF step to sample the discrete failure nodes. The particles are resampled as in the normal PF (based on the likelihood), but instead of using the continuous state estimates proposed by the PF step, UKF or PFUKF is used to give a better estimate of the state spaces for the resampled particles. The estimates of the continuous states do not depend directly on the number of particles in the sense that an increased number of particles automatically implies an improved state estimate. Rather, the state estimates depend on accurate estimates of the discrete failure nodes which *do* depend directly on the number of particles. And hence the estimates of the state variables depend indirectly on the number of particles. Correct tracking of the discrete failure nodes should - to some extent of course - make the estimates of the continuous states better and visa versa.

The crucial point is thus to have just enough particles to be able to track the correct values of the failure nodes. Using more particles does not have any significant influence on the RMSE as shown in Figure 6.18 illustrating the RMSE and the total number of wrong failure nodes as a function of the number of particles using UKF and PFUKF. Data was generated by drawing process and measurement noise samples from the Gaussian distribution $\mathcal{N}(0, 0.1)$ which is a fairly high noise level compared to the drifting level set to 1 (i.e. drifting one unit). 15 subparticles were used and the simulation period included 30 time steps. The relatively short time period was chosen to reduce the total experimental time. Each point is based on 10 different data sets using 10 runs for each data set. Outliers were removed.

The left plot in Figure 6.18 shows the RMSE as a function of the number of particles using UKF and PFUKF. The right plots show the number of failure tracking errors as a function of the number of particles. UKF and PFUKF is able to track the discrete failure nodes correctly using approximately 16 particles. As soon as the algorithms get the conductance failures right there is hardly any RMSE reduction on the state estimates to see by increasing the number of particles further using UKF as well as PFUKF.

Notice that PFUKF only improves the UKF state estimates when UKF is making poor estimates which happens using a small numbers of particles (see Figure 6.18). PFUKF actually makes the UKF estimates a little worse for a large number of particles. In order to investigate this behavior a number of experiments were performed which focus on the importance sampling of the UKF subparticles with respect to the RMSE.

Each subparticle i is weighted in PFUKF using the importance ratio described in section 4.6.1.

$$ratio_i = \frac{likelihood_i \cdot prior_i}{proposal_i} \tag{6.20}$$

The prior only influences the importance ratios when the changes in the continuous states over time are not too large (or the noise level is very high), otherwise they become smaller than the numerical precision in Matlab. Hence, the prior will not influence the ratio significantly in a very dynamic system. The watertank simulation is a highly dynamic system

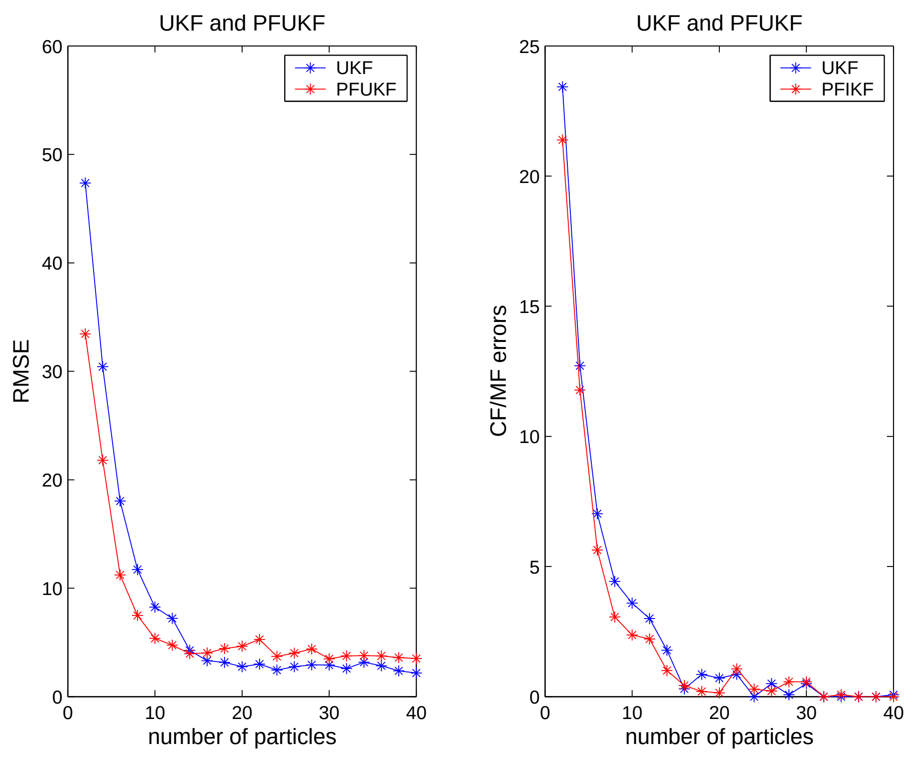


Figure 6.18: RMSE and number of failure-tracking errors as a function of the number of particles using UKF (blue) and PFUKF (red)

until it reaches the equilibrium phase, i.e. the importance ratio for each subparticle is often based only on the likelihood and the proposal.

$$\tilde{ratio}_i = \frac{likelihood_i}{proposal_i} \tag{6.21}$$

As the proposal appears in the denominator of the ratio, it increases the weights of the subparticles when they appear in the tails of the proposal. In a way it ensures a more broad distribution of samples. If the subparticles were only based on a narrow likelihood appearing in the tail of the proposal, only a few subparticles would get a significant importance weight. It is therefore important to move the particles towards the likelihood distribution.

Figure 6.19 illustrates how PFUKF moves particles with respect to the likelihood and the RMSE compared to UKF. The bars in Figure 6.19 represent the absolute difference in RMSE between the UKF and the PFUKF. For each plot, the four bars represent the 4 combinations between increased/decreased likelihood and increased/decreased RMSE (*increased likelihood* and *increased RMSE* (green bars), *increased likelihood* and *decreased RMSE* (blue bars), *decreased likelihood* and *increased RMSE* (red bars), *decreased likelihood* and *decreased RMSE* (yellow bars)). I.e. the green and red bars represent the total improvement in RMSE using PFUKF compared to UKF and the blue and yellow bars represent how much PFUKF decreases the RMSE compared to UKF.

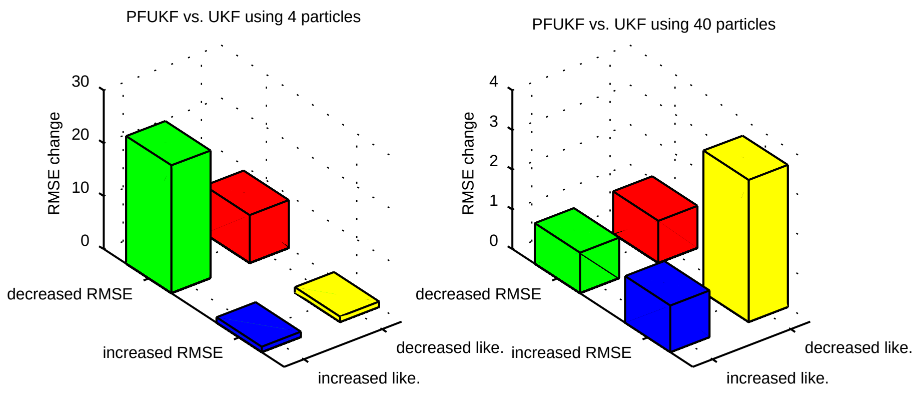


Figure 6.19: Relative particle movement using PFUKF compared to UKF with respect to increased/decreased likelihood and increased/decreased RMSE. Left plot corresponds to 4 particles whereas right plot corresponds to 40 particles. 15 subparticles were used in both cases

Increased or decreased RMSE *and* likelihood (green and yellow bars) makes sense, but how is it possible to improve the likelihood and disprove the RMSE (blue bars) and visa versa (red bars)?

First of all, the measurement noise which moves the likelihood away from the true continuous states: As the likelihood calculation is based on the noisy measurements, the true continuous states do not correspond to the highest likelihood. This implies that we can select particles with higher likelihood that are further away from the true state (blue bars). Furthermore, it is possible to have higher likelihood values for state estimates far away from the true continuous states as the prediction of the observation node (flow) given as the product of the conductance and the pressure (plus measurement noise) can be close to the true value using wrong conductance and pressure values, e.g. a too high pressure and a too low conductance or vice versa would still give the correct flow.

It is also possible to move away from the noisy state and come close to the true state (red bars) as the importance ratio also includes the proposal and hence we do not weight only on behalf of the likelihood. It could also happen if we use a small number of subparticles and hence we may have to select between a few number of subparticles where all, by coincidence, are further away from the noisy state and close to the true state.

The left plot in Figure 6.19 was made using 4 particles and the right plot corresponds to 40 particles. In both cases the same data set and 15 subparticles were used. A small number of particles makes it difficult to track the true failure combination. As the UKF is not able to make up for the wrong failure estimates a lot of improvement is left for PFUKF. With the UKF estimate far from the true state a solid connection between increased likelihood and decreased RMSE exists. Using only 4 particles, PFUKF is able to move a lot of the particles towards regions of higher likelihood and more accurate state estimates (green bar).

It is also able to decrease the RMSE without decreasing the likelihood (red bar). This is due to the measurement noise which moves the likelihood away from the true state. Almost no RMSE increase is seen using PFUKF compared to UKF (low blue and yellow bar).

Using enough particles to make the modified PF step capable of tracking the true failures, the UKF makes accurate estimates of all state variables leaving little improvement to PFUKF. This is illustrated in the right plot in Figure 6.19. Notice the different z scale on the two plots. Almost no change in RMSE is seen using 40 particles compared to using only 4 particles. In fact, PFUKF makes the estimates worse! The yellow and blue bars become larger than the red and green bars, i.e. PFUKF is making more bad moves than good moves with respect to the RMSE. The modified PF step is tracking all the failures correctly and UKF is able to make very accurate state estimates. PFUKF is making just as many moves decreasing the likelihood (red bars) as moves which are fitting the noise (blue bars). The green bar corresponding to a decreased RMSE and increased likelihood is no longer the primary bar. Instead, a lot of moves are made increasing both the RMSE and the likelihood (yellow bar).

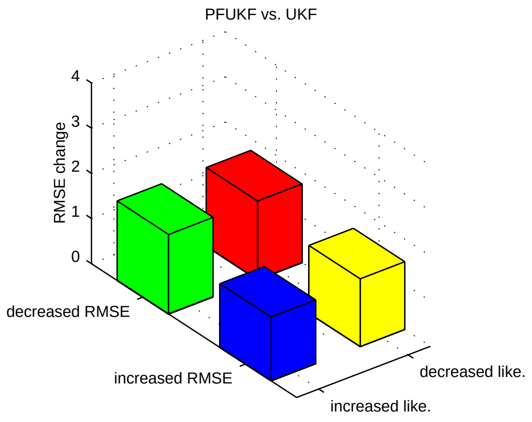


Figure 6.20: Relative particle movement using PFUKF compared to UKF with respect to increased/decreased likelihood and increased/decreased RMSE. 40 particles and 40 subparticles were used.

One explanation might be that PFUKF is sampling from a too large proposal distribution. Sampling from a Gaussian distribution with a too large covariance matrix decreases the probability of making small moves towards the true state. If this explanation is true, using a large number of subparticles should turn some of the bad moves represented by the yellow bar into good moves represented by the green bar. Figure 6.20 shows the results of the same experiment using 40 subparticles and 40 particles. By increasing the number of subparticles, PFUKF was able to reduce the number of bad moves and increase the number of good moves.

PF and PFMC update the estimates of the discrete and continuous variables simultaneously making the RMSE directly dependent on the number of particles as shown in Figure 6.21. Data was generated by drawing process and noise samples from a Gaussian distribution $\mathcal{N}(0, 0.1)$ and a drift factor of 1. The simulation period included 30 time steps. The relatively short time period was chosen to reduce the total experimental time.

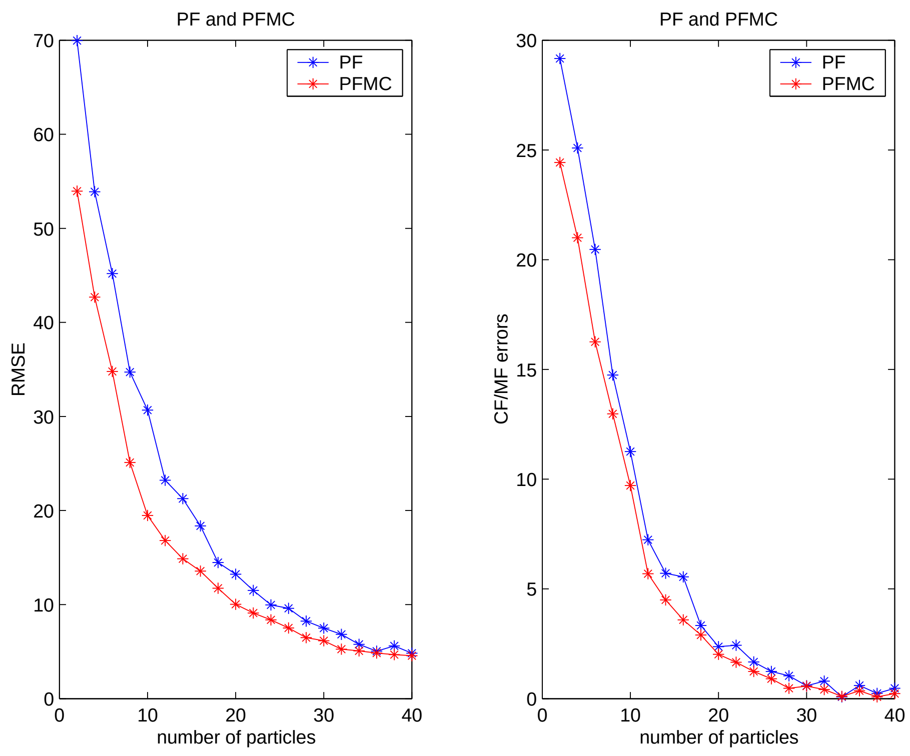


Figure 6.21: RMSE and number of failure estimation errors as a function of the number of particles using PF (blue) and PFMC (red)

Even though PF and PFMC are able to track the conductance and measurement failures, the RMSE is decreased by increasing the number of particles further. By comparing Figure 6.3.5 and Figure 6.21 it is seen that UKF and PFUKF are making better estimates than PF and PFMC for any number of particles below 40. These experiments clearly indicate that the UKF based algorithms are performing much better than the simple PF algorithm. Using a MC step improves the PF algorithm, but UKF and PFUKF are still superior.

6.3.6 Noise

Simulations allow us to have access to most information about the given problem. In real life we have to propose a network structure, process and measurement models describing our beliefs about the generation of the ground truth before we apply our filters. So what happens if one proposes a wrong model? How robust are the algorithms if they are not given optimal conditions? This section concerns the algorithms capability to deal with a 'wrong' proposal of the process and measurement noise compared to the noise levels used in the data generation. Section 6.3.7 investigates the filters robustness when we propose a wrong measurement model.

UKF assumes Gaussian process and measurement noise. Initially, one has to propose a certain process and measurement noise covariance matrix which is incorporated into the Kalman updating step. Hence, it would be interesting to see the effect of proposing wrong process and measurement noise covariance matrices.

The UKF algorithm initializes the covariance matrix based on N samples from the prior $p(x_0)$ which follows the Gaussian process noise distribution (see section 4.6.1). Furthermore, the dimension of the process noise covariance matrix is 5x5 and we therefore expect the process noise to influence the UKF variance estimation more than the measurement noise covariance matrix of dimension 3x3. Figure 6.22 shows the UFK variance estimates for all 5 state space variables using 9 different process noise proposals (left plots) and 9 different measurement proposals (right plots) for 100 time steps. While changing the process noise proposals the true measurement noise was used as measurement noise proposal and visa versa. 10, 25, 50, 75, 100, 125, 150, 175 and 200 % of the true noise covariance was used in both experiments. A diagonal covariance matrix with elements $1e-1$ was used for both the process and the measurement noise covariance matrix in all experiments to generate data. The same data set was used in both tests for comparison.

As expected, the process noise proposal has a greater effect on the UKF variance estimates (left plots in Figure 6.22) compared to the UKF variance estimates for the same relative changes in measurement noise proposal (right plots in Figure 6.22). Changing the measurement noise hardly has any effect on the UKF variance estimates. Especially the variance estimates of C10, P1 and P2 change so little for the different measurement noise proposals that it is hard to see more than one line.

As shown, the UKF posterior covariance estimates seem to depend on our process noise proposal. But how does this influence the estimates of the continuous states in UKF as well as PFUKF? As mentioned in section 4.6.1, PFUKF approximates Gaussian filtering densities $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ using the mean and covariance matrix proposed by UKF. Furthermore, the importance weights are partly based on the prior and the likelihood which is based on the process and measurement noise proposals resp.

Figure 6.23 shows the result of two experiments using UKF and PFUKF. The two left plots correspond to a change in the process noise proposal while using the true measurement noise level. The opposite situations are shown in the two plots to the right. The top

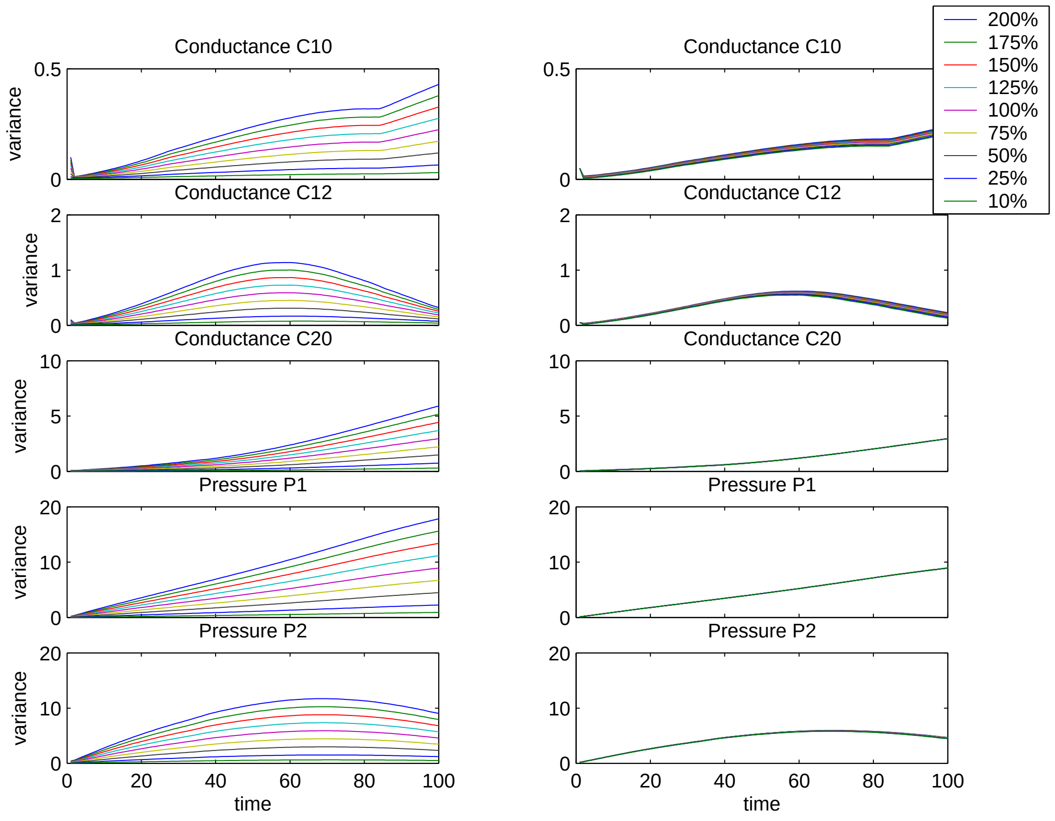


Figure 6.22: UFK variance estimates for all state variables using 10 different process noise proposals (left plots) and 10 different measurement proposals (right plots). The noise proposals were 10%, 25%, 50%, 75%, 100%, 125%, 150%, 175% and 200% of the true noise level

plots show the RMSE for UKF and PFUKF and the lower plots show the relative particle movement in PFUKF compared to UKF. The lower plots illustrate the relative movement in PFUKF compared to UKF with respect to increasing/decreasing the RMSE and increasing/decreasing the likelihood.

The results presented in Figure 6.23 were based on 10 different data sets and 10 runs for each data set. 60 particles and 30 subparticles were used in every simulation which lasted 30 time steps. Outliers were removed. The UKF state estimates seem independent of the choice of measurement and process noise proposal and has almost the same RMSE value for all different noise proposal combinations (see blue line in the two top plots in Figure 6.23). Whereas the RMSE using PFUKF is slowly increasing with the increased process noise proposal (see lower left plot). The RMSE is constant for changes in the measurement noise proposal below the true noise level and seems to increase with a constant slope using measurement noise proposals above the true noise level.

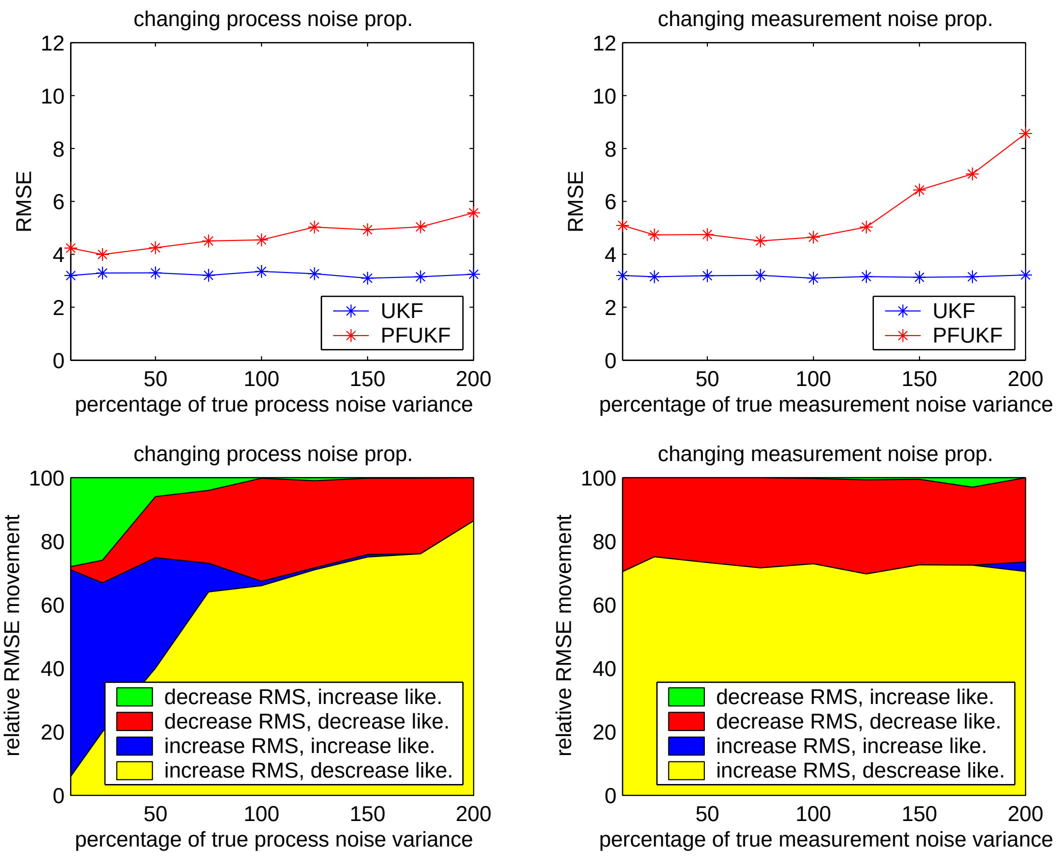


Figure 6.23: The plots correspond to experiments using UKF and PFUKF with ten process noise proposals (left plots) and ten different measurement noise proposals (right plots). The top plots show the RMSE for UKF (blue line) and PFUKF (red line) and the lower plots show the relative particle movement in PFUKF compared to UKF)

The two lower plots in Figure 6.23 showing the relative particle movement in PFUKF illustrates how PFUKF is affected by the different process and measurement noise proposals. PFUKF samples from the approximated Gaussian posterior distribution using the UKF covariance and mean estimates for each time step. The UKF mean estimate is not affected by changes in the noise proposals so apparently it is only the covariance estimate which is affected by changes in the noise proposals in the range of 10% to 200% of the true noise level. With a UKF mean estimate fairly close to the true state we would prefer a smaller noise covariance matrix rather than a larger noise covariance matrix (compared to the true noise level). A lot of samples are needed to improve a good mean estimate using a too large covariance matrix.

Now, let us discuss the relative movement changing the process noise proposal (lower left plot Figure 6.23). Using a smaller process noise proposal (left plots) making the posterior covariance estimates small allows us to take small steps towards the true state. Some good moves are made with decreasing RMSE and increased likelihood (green area) using a small process noise proposal. However, a lot of the subparticles seem to be fitting the true measurement noise represented by the blue area (increased RMSE and likelihood). The likelihood moves the subparticles away from the true state when the UKF state estimate is too close to the true state. This trend disappears with a larger process noise proposal making the UKF variance estimates larger (see lower left plot in Figure 6.23). PFUKF now samples from a Gaussian distribution that is too large and draw subparticles that are further away from the true state. This is represented by the yellow colored area corresponding to increased RMSE and decreased likelihood. All subparticles are worse estimates than the ones proposed by UKF with regard to RMSE as well as likelihood.

Using a too small process noise proposal the subparticles tend to fit the measurement noise (blue area) whereas a too large process noise proposal allows sampling from areas far away from the true state (yellow area). No improvements were seen using PFUKF. It is better to sample using a too small process noise proposal rather than a too large process noise proposal if UKF makes good state estimates.

The plots to the right in Figure 6.23 show the results of changing the measurement noise proposal. The relative movements stay the same for all nine measurement noise proposals even though the RMSE value changes. The UKF estimates of the state covariance are hardly effected (see Figure 6.22) changing the measurement noise proposal. The same holds for the UKF mean estimates and the PFUKF therefore samples from the same Gaussian distribution. Hence, it is no surprise that the relative movements stay the same for the different measurement noise proposals. But what makes the RMSE increase? PFUKF resamples the subparticles based on the importance ratio and by increasing the measurement noise proposal we increase the likelihood distribution making it more and more difficult to distinguish between good and poor subparticles. The prior is not affecting the importance ratio due to numerical problems, but the proposal appears as the inverse in the importance ratio moving the subparticles towards the tails of the proposal distribution.

All in all, the experiment indicates that it is difficult to improve an accurate UKF estimate in a 5 dimensional space by sampling further from an approximated Gaussian posterior. Of course, increasing the number of subparticles and particles might improve the outcome of PFUKF. Furthermore, the experiments show that it is better to propose too small a noise rather than too large if UKF is making accurate state estimates. In a real life simulation, however, where the true process and measurement models are not known, the UKF is most likely to make poor estimates leaving a lot of improvement for the PFUKF.

One of the main objectives in the watertank problem is to be able to track conductance and measurement failures. To evaluate the performance of our estimates of the true failure values we need to compare the process and measurement noise levels with the extent to which a certain failure influences the flow. To give an indication of how robust the algorithms are with regards to the noise level, experiments were performed changing the noise level for both the process and the measurement noise. Only UKF and PFUKF were used as these algorithms showed superior performance in the previous experiments. In all experiments the true noise levels were used as process and measurement noise proposals.

A positive/negative conductance drift factor of 2 was used, the conductance was set to 600 in case of a burst and 100 was added to the measurement in case of a measurement failure. Four different process and measurement noise levels were used (0.01, 0.1, 0.2 and 0.4) giving 16 different combinations. The RMSE and the number of wrong conductance and measurement failure estimates as a function of the process and measurement noise levels are shown in Figure 6.24. A time period of 100 time steps were used in all tests. The results were based on 20 different data sets using 10 runs for each data set. Outliers were removed.

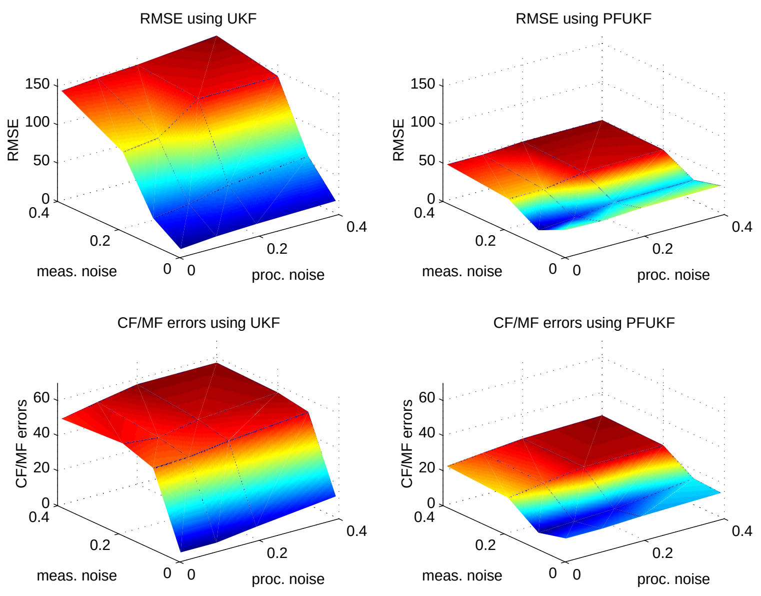


Figure 6.24: Surface plots showing the RMSE (top plots) and CF/MF estimation errors (lower plots) using UKF (left plots) and PFUKF (right plots) based on 16 different measurement and process noise combinations

First of all, notice the nice correlation between the RMSE and the number of wrong failure estimates for UKF (two left plots in Figure 6.24) and PFUKF (two right plots in Figure 6.24). An accurate state estimate corresponds to a small RMSE making it easier to track the discrete failures and vice versa. Both the RMSE and the number of wrong failure estimates using UKF and PFUKF are more influenced by the level of the measurement noise than the process noise. This does not come as a surprise as the state estimates are updated based on their connection to the observation nodes. A noisy connection between state variables and observation variables is more crucial than a noisy connection between the previous and present set of continuous states. The filtering algorithms are estimating the noisy continuous states and the algorithms have to make up for the noise added to the observation nodes. It is clearly more difficult to remove noise from the observation nodes than to estimate noisy continuous states based on a noise-less connection to the observation nodes. Furthermore, Figure 6.24 once again illustrates the relationship between UKF and PFUKF. PFUKF is doing much better than UKF for large measurement noise levels, but notice that the RMSE increases using the smallest process noise level (0.01) and is actually doing worse than UKF. This behavior corresponds to the observations made in the previous experiments: When UKF makes accurate estimates PFUKF can make matters worse by either fitting the measurement noise or sampling from a Gaussian distribution that is too large.

Similar experiments were performed using PF and PFMC with different process and measurement noise combinations. As in the experiments with UKF and PFUKF, the measurement noise proved to dominate the RMSE and the number of wrong failure estimates in both PF and PFMC. Figure 6.25 shows the RMSE and the number of wrong failure estimates as a function of the measurement noise level. The results are shown only for the process noise level 0.1. A time period of 100 time steps was used in all four tests. The results were based on 20 different data sets using 10 runs for each data set and removing outliers.

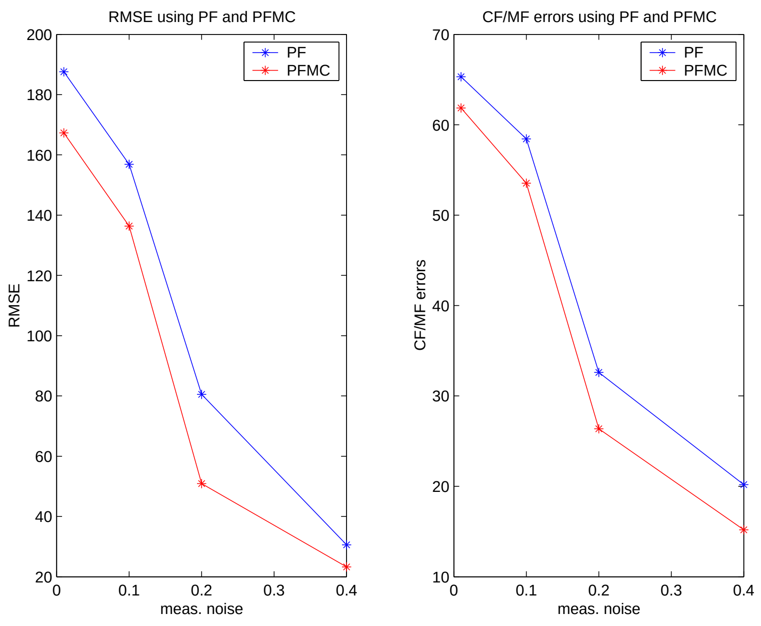


Figure 6.25: RMSE (left plot) and wrong CF/MF failure estimates (right plot) using PF (blue) and PFMC (red) based on four different measurement noise levels. The process noise level was constantly set to 0.1.

PFMC is able to make improvements of the poor PF estimates (see Figure 6.25) with respect to a reduction of the RMSE and the number of wrong failure estimates. PFMC is, however, still performing worse than UKF and PFUKF.

Let us recapture some of the main observations in this section. First of all, if UKF is making accurate estimates it is difficult to make improvements by further sampling using PFUKF. Often, PFUKF makes the state estimates worse. The performance of UKF and PFUKF with respect to the state estimates seemed to depend more on the measurement noise level than the level of process noise. Large measurement noise levels made the

UKF estimates poor and PFUKF was able to move the particles towards the true state, i.e. reducing the RMSE. Both PF and PFMC were performing worse than UKF and PFUKF with respect to RMSE and the number of wrong failure estimates for all tests with different measurement noise levels.

6.3.7 Choice of process and measurement model

In a real life simulation one of the major challenges is to come up with reasonable process and measurement models. Simulating the actual data gives us all the knowledge about the problem we need and makes us capable of providing our algorithms with optimal conditions. That is why UKF is able to make very accurate state estimates leaving no real space for improvement by sampling from the approximated Gaussian distributions proposed by UKF as it is done in PFUKF. So far we have seen that PFUKF is able to improve the UKF estimates if the measurement noise level is high or if a small number of particles is used making it difficult to track the discrete failure nodes. Hence, PFUKF really proves its superiority in scenarios where UKF is not given optimal conditions. To further investigate this relationship, we experiment by changing the proposed measurement model. This should affect UKF negatively leaving a lot of space for improvement to PFUKF. PF and PFMC were also included in the experiments for the sake of comparison.

There is a number of different ways to ”sabotage” the measurement model. In this case it was done by simply adding 5% to all the flow estimates.

100 particles and 30 subparticles were used over a time period of 60 time steps. The process and measurement noise samples were drawn from the Gaussian distribution $\mathcal{N}(0, 0.1)$. Table 6.1 shows the mean RMSE and variance using ten different data sets and 10 runs for each data set. The second column shows the average number of incorrect failure estimates.

PFUKF is not surprisingly the filtering algorithm that is most capable of dealing with a wrong measurement model as this was already indicated in the simple simulation in chap. 5. PFUKF is able to move the particles towards regions of higher likelihood which reduces the RMSE and makes tracking of the discrete failure nodes easier (see Figure 6.26). PFUKF is hardly making any bad moves with respect to the RMSE. We might be able to further improve PFUKF estimates using more particles or subparticles. As shown in sec. 6.3.6, it is possible to change the size of the approximated Gaussian distribution (proposed by UKF) by proposing higher noise levels.

The experiment once again showed that PFUKF is capable of making more accurate state estimates than UKF when the algorithms are not performing under optimal conditions. Furthermore, UKF and PFUKF made more accurate state estimates than PF and PFMC.

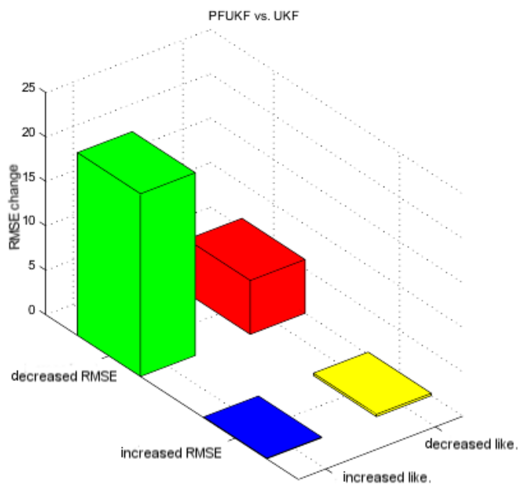


Figure 6.26: Relative particle movement using PFUKF compared to UKF with respect to increased/decreased likelihood and increased/decreased RMSE. The relative movement is based on a number of experiments using a wrong measurement model

6.3.8 Tracking

The watertank simulation operates with two kinds of failures, conductance failures and measurement failures. Both type of failures are sampled in a modified PF step. Given optimal conditions, the PF algorithm should be able to weight the particles according to the true failure combination. Now, what does optimal conditions imply for the PF step? First of all, to track the true failure combination at least one sample of the discrete failure node combinations corresponding to the true combination has to be drawn. The probability of

Algorithm	RMS		CF/MF errors
	mean	var	
Particle Filter - generic (PF)	256	204	56.3
Particle Filter - Metropolis-Hastings move (PFMC)	227	187	33.8
Unscented Kalman Filter (UKF)	208	43	30.1
Particle Filter - UKF proposal (PFUKF)	178	76	23.4

Table 6.1: Mean and variance of RMSE values of state estimates and average conductance and measurement failure errors (CF/MF errors) using a wrong measurement model. The results were based on 10 different data sets using 10 runs for each data set

doing this increases with the number of particles. Hence, to provide the modified PF with optimal conditions a large number of particles should be chosen. Secondly, we want the modified PF step to weight and resample the particles based on a good estimate of the continuous states from the previous time step. A good estimate of the continuous states for all particles should make it easier for the modified PF step to distinguish between good or bad particles with respect to the discrete failure combination. This is realized by implementing a high-performance filtering algorithm serial linked with the modified PF step, which is able to make good estimates of all continuous states.

In [21], Daphne Koller and Uri Lerner apply a generic particle filter to the watertank problem using the old network structure, see sec. 6.3.3, but propose as future work an implementation of a combination of PF (sampling the discrete failure nodes) and a more sophisticated particle filter to sample the continuous variables as we have done in this work. It is impossible to make direct comparisons between this work and the work of Koller and Lerner as they use the old network structure, use different probabilities for the events and probably model various aspects differently (s.a. pipe bursts or measurement failures). However, as we are also using the PF algorithm in our network a comparison between the other filtering techniques and PF can be made. As mentioned, in this work the network structure was changed, but as shown in section 6.3.3 the transformation only improves the performance of PF. Still, UKF and PFUKF have outperformed both PF and PFMC in every single experiment and have proved to be the best choice of filtering technique.

So provided with a good state estimate using UKF or PFUKF and using a large number of particles, the modified PF step should have a very good chance of tracking the true failure combination by sampling from a proposal distribution close to the true one (which is known in a simulation).

Figure 6.27, 6.28 and 6.29 show the tracking of C10, C20 and P2 for a simulation of 100 time steps using the UKF algorithm. A drift factor of 1 was used and both process and measurement noise samples were drawn from the Gaussian distribution $\mathcal{N}(0, 0.5)$. The results are based on ten runs with one data set using 300 particles.

Table 6.2 lists the different events occurring doing the simulation.

Notice the large error bar at time step 31 in figure 6.27 and 6.29 corresponding to the burst of pipe 3 at time step 30 (only every second error bar was plotted for visual reasons). Apparently, UKF had no problem recognizing the measurement failures as these events are not seen in the error bars for any of the three variables, as this would have caused very large errorbars at the given points in time. Furthermore, notice the larger error bars in Figure 6.27 when the drifting ends at $t = 52$ which might indicate that the system in some runs estimated positive/negative drift of C_{10} as this is not crucial for the flow measurements, when the conductance is very high (low resistance). At $t = 80$ it locates the burst of pipe 1 and stays at the bursting level.

Event	Event time
Positive conductance drift begins for C10	t=27
Pipe 3 bursts	t=30
End of positive conductance drift for C10	t=52
Measurement failure for flow F12	t=56
Measurement failure for flow F10	t=66
Pipe 1 bursts	t=80

Table 6.2: Event sequence for a typical run using 300 particles and drawing noise samples from the Gaussian distribution $\mathcal{N}(0, 0.5)$.

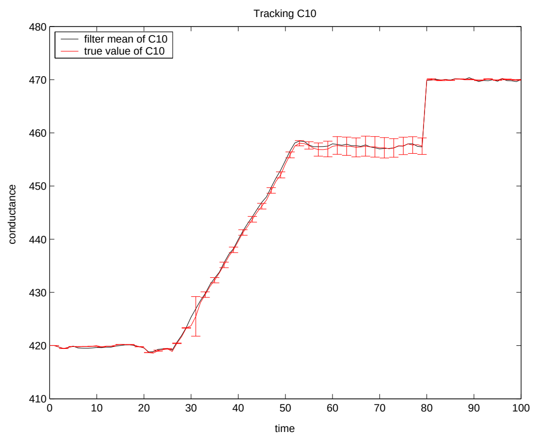


Figure 6.27: Tracking of conductance variable C10 using a process and measurement noise level of 0.5 (variance in Gaussian distribution with mean 0). The estimated conductance C10 (red line) using UKF is plotted with confidence intervals (plus, minus two standard deviations from the mean estimate) and the true conductance C10 (black line).

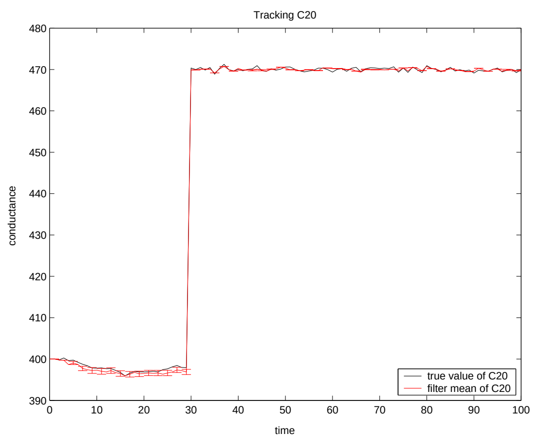


Figure 6.28: Tracking of conductance variable C20 using a process and measurement noise level of 0.5 (variance in Gaussian distribution with mean 0). The estimated conductance C10 (red line) using UKF is plotted with confidence intervals (plus, minus two standard deviations from the mean estimate) and the true conductance C20 (black line).

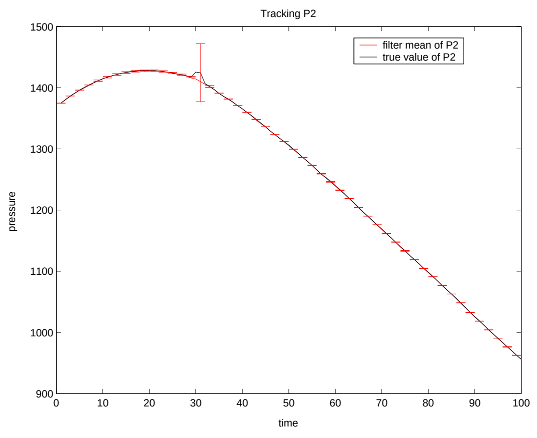


Figure 6.29: Tracking of pressure variable P2 using a process and measurement noise level of 0.5 (variance in Gaussian distribution with mean 0). The estimated conductance C10 (red line) using UKF is plotted with confidence intervals (plus, minus two standard deviations from the mean estimate) and the true conductance P2 (black line).

Chapter 7

Cyberglove

Having demonstrated and compared our implemented algorithms in a simulated physical setup, we would like to investigate their validity on a real-life problem, where we can only approximate the data generation by expressing our beliefs in a DBN, but where we are still able to collect the ground truth and measure the performance of the algorithms.

In Neuro Engineering Lab (NEL) at NASA Ames Research Center, California, they use a *cyberglove*, which is a glove equipped with strain gage sensors that are able to register the angle of the three joints on every finger. Angular velocity and angular acceleration can be derived.

Our ultimate objective is to be able to predict the values of the glove sensors by setting up a DBN in which the hidden nodes are the joint angles, angular velocities and angular accelerations for each finger and the observed nodes are measured EMG signals from surface electrodes on the arm, i.e. being able to compute the exact movement of the fingers using electrodes placed on the surface of the arm. If this is possible, one would be able to interact with a virtual environment which e.g. would have an enormous impact in the computer science society. Today, a great number of people are suffering from injuries caused by repeated use of an ordinary keyboard, mouse and joystick which puts a lot of stress on joints and muscles.

However, with a limited amount of time available we must constrain our problem as the modelling itself, the data acquisition and the required a priori studies of the system and the data asks for much more time. With 3 measurements for each joint it adds up to 9 variables for each finger, i.e. 45 variables for the entire hand. Creating a model with a hidden state variable for each of the joint and using EMG signals as measurement variables, say 4 signals corresponding to 4 pair of electrodes, one would need a DBN with 49 variables, which leads to computations in a very high dimensional space.

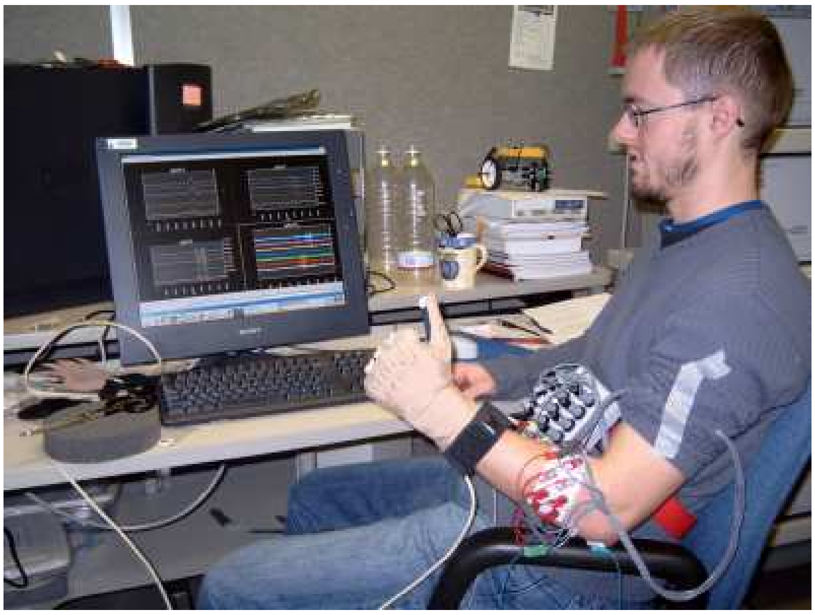


Figure 7.1: Physical setup for acquisition of cyberglove and EMG data

Hence, in this work we 'remove' the thumb by keeping it steady during the movements and keeping the four remaining fingers close together to form a 'single finger' to make a single movement of all four fingers at the same time.

7.1 Data acquisition

In Figure 7.1 the physical setup of the data acquisition is illustrated.

The subject wore the cyberglove on his left hand and 8 pairs of dry surface electrodes were attached to the lower part of the left arm. For each pair, the two electrodes were placed next to each other. 4 pairs were placed on the upper side of the lower arm and the remaining 4 on the underside. First pair on both sides were placed approximately 2 centimeters from the lower arm bone and 7 centimeters from the elbow tip. The other pairs were then placed accordingly as illustrated in Figure 7.2 and numbered 1-8 so electrode pair 1 is on the upper side, closest to the bone and electrode pair 8 is on the underside, closes to the bone. The reference electrode was placed on the elbow tip.

In the recordings we used a sampling frequency of 1024 Hz. for the EMG signals as well as for the cyberglove. Furthermore, the glove measurements were filtered during the recordings using a Hamming windowed low-pass filter of order 1024 with a 5 Hz. cut-off frequency to obtain smooth signals. These values were then shifted accordingly to match the EMG measurements.

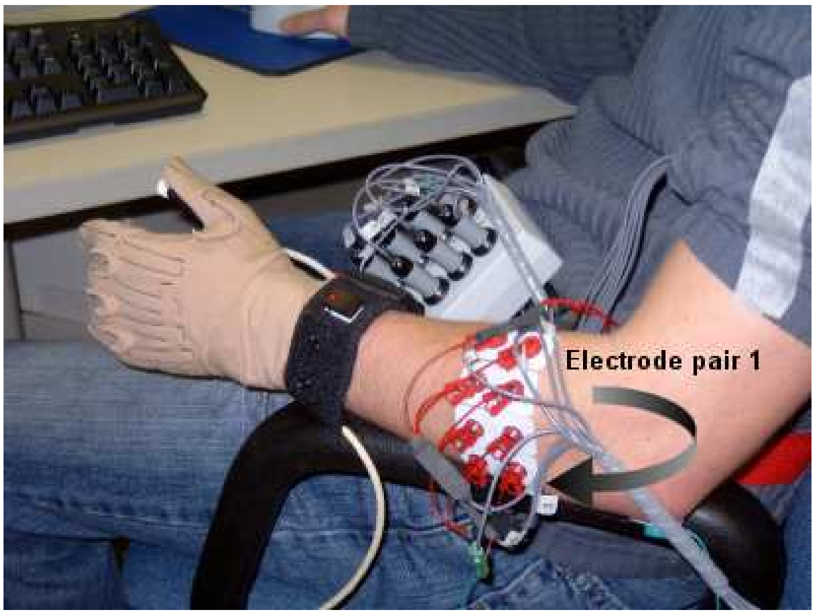


Figure 7.2: Electrode configuration for acquisition of cyberglove and EMG data

Next, the absolute EMG measurements were calculated and filtered using a Hamming windowed low-pass filter of order 1024 with a 7 Hz. cut-off frequency. This was done to remove noise and to work on signals with a non-zero mean value.

Finally, the glove angle measurements were smoothed further using a Hamming windowed low-pass filter of order 1024 with a 4 Hz. cut-off frequency to remove remaining noise and artifacts.

Instead of filtering the velocity and acceleration we used the filtered angle measurements and calculated the velocity and acceleration using the forward-backward relations

$$v(t) = \frac{x(t+\Delta t) - x(t-\Delta t)}{2\Delta t} \tag{7.1}$$

and

$$a(t) = \frac{v(t+\Delta t) - v(t-\Delta t)}{2\Delta t} \tag{7.2}$$

The subject was asked to move his fingers from an extended position and bending them to make a fist (without using excessive force to tighten the fist, i.e. just touching the palm) and then move the fingers back to the extended position. These movements were repeated forty times without interruptions in each of three data recording sessions and divided into two subcategories from here on known as *Close hand* corresponding to moving the fingers from the extended position and down and *Open hand* corresponding to the opposite movement. Furthermore, the subject was asked to do approximately the same movements over and over again in a consistent manner, i.e. to attempt to use roughly the same amount of time for each movement and not changing the extreme positions too much. This was done to keep the domain of gestures within reasonably boundaries for this simple first-time approach to the problem.

7.2 Model selection

The model was developed in an attempt to setup a very simple mechanical model of a finger movement. As we keep the fingers together we can imagine the system as the movement of a single finger as illustrated in Figure 7.3.

In the model, the movement of each joint is viewed as a mechanical system consisting of a wheel which turns when the finger is bended or stretched. The finger moves when the muscles in the arm are extended corresponding to pulling strings attached to the wheel. In the Figure (resembling a finger on the left hand), the finger is stretched using the muscles on the upper side of the lower arm and it is bended using the muscles on the underside of the lower arm.

Using the basic physical principle of

$$Force = Mass * Acceleration \tag{7.3}$$

the registered EMG signals were assumed to be proportional to the acceleration of the wheel. However, we acknowledge the fact that the acceleration is zero if one is keeping the fingers in an extended position or by keeping a solid fist, but the force one is using is definitely not zero! This is due to the fact that the resting position of the hand is in neither of these positions, but somewhere in between. We model this fact by attaching a couple of springs to the wheel, both trying to keep the wheel in a resting position α radians from a horizontal line. However, during a movement one does not expect to see a dip in the EMG recordings when the fingers are passing the resting position as the fingers are still accelerating and hence a force is needed.

So far, we imagine the EMG measurements as a force being the sum of the acceleration of the wheel times a proportionality constant (representing the mass of the wheel) and a spring force with opposite sign.

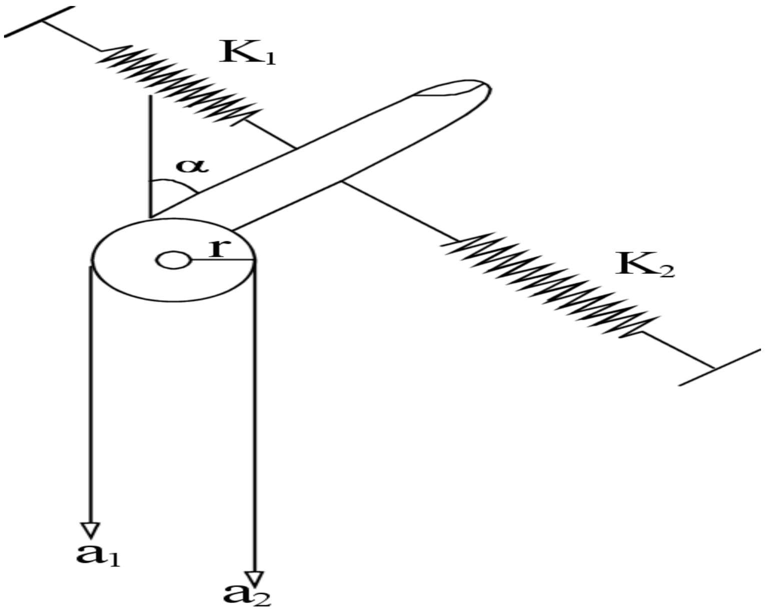


Figure 7.3: Mechanical model of finger movement

However, during our data acquisition we noticed that the EMG signals overlapped as illustrated in Figure 7.10. One would expect the signals to be well separated as it seems intuitively plausible that the upper side muscles are resting when the fingers are bended and vice versa leaving it to the springs to move to the resting position, but human muscles do not work in nice, isolated units responsible for specific movements. Just imagine bending your hand very slowly. This takes a force from muscles on both sides of the arm which is easily felt. And one can not expect to obtain EMG signals measuring the extension of specific muscles without measuring the activity of surrounding muscles. This makes EMG measurements quite noise and difficult to model.

Hence, we need to add a contribution from the opposite muscles for both movements. As we do not have the necessary physiological knowledge to model this contribution in a more detailed way, we simply add the oppositely measured force times a constant.

We model this physical setup in a DBN using only a single joint (for simplicity reasons - the system can easily be extended to include all three joints of the finger). As the proposed model involves a large amount of uncertainty due to the simplified physical model and especially the contribution from the opposite muscles, we choose to predict only one observable node (its value corresponding to the EMG measurement from an electrode placed on the side of the arm measuring the primary muscles used in the given movement) and using the values from an electrode placed on the opposite side of the arm as observable input to the prediction.

By assigning the extended position the angular value 0 and assigning angles between this position an the fist position negative values (in accordance with the data acquisition) we end up with the following measurement model:

Close hand gesture

$$a_2 = \frac{\alpha - \theta}{2\pi} r k_2 - \ddot{\theta}\frac{r}{2\pi} + k_3 a_1 \qquad \theta \in [\alpha; 0] \tag{7.4}$$

$$a_2 = \frac{\alpha - \theta}{2\pi} r k_1 - \ddot{\theta}\frac{r}{2\pi} + k_3 a_1 \qquad \theta \in [-\pi; \alpha] \tag{7.5}$$

Open hand gesture

$$a_1 = \frac{-\alpha + \theta}{2\pi} r k_1 - \ddot{\theta}\frac{r}{2\pi} + k_4 a_2 \qquad \theta \in [\alpha; 0] \tag{7.6}$$

$$a_1 = \frac{-\alpha + \theta}{2\pi} r k_2 - \ddot{\theta}\frac{r}{2\pi} + k_4 a_2 \qquad \theta \in [-\pi; \alpha] \tag{7.7}$$

where r is the radius of the wheel, α is the (negative) angle from the extended position to the resting position, k_1 and k_2 are the spring constants corresponding to spring 1 and 2 resp., θ is the angle, $\ddot{\theta}$ is the angular acceleration and k_3 and k_4 are proportionality constants. We would of course have to write another set of equations for a_2 during an open hand gesture and for a_1 during a close hand gesture, but as mentioned we predict only one electrode to keep things simple.

As the measurement model is dependent on the performed gesture, we choose to include the gesture (*Close hand* or *Open hand*) as a discrete valued node which we sample by drawing from a uniform distribution on a binary interval

$$\begin{aligned} P(G = OpenHand) &= 0.5 \\ P(G = CloseHand) &= 0.5 \end{aligned} \tag{7.8}$$

Having settled for a measurement model expressing the connections between the hidden nodes at time t and the observed nodes at time t, we need to propose a process model, i.e. a model expressing how the the hidden variables at time $t-1$ and time t are related. The angular acceleration is modelled using two different proposal distributions based on the study of the measured glove data (prior knowledge):

Sine curve:

$$\ddot{\theta}_t = A_t * \sin(P_t * \pi * (\tilde{t} + 1) + \frac{\pi}{2}) \tag{7.9}$$

$$\tilde{t} = sin^{-1}\left(\frac{\ddot{\theta}_{t-1}}{A_t}\right)\frac{1}{P_t \pi} - \frac{\pi}{2} \tag{7.10}$$

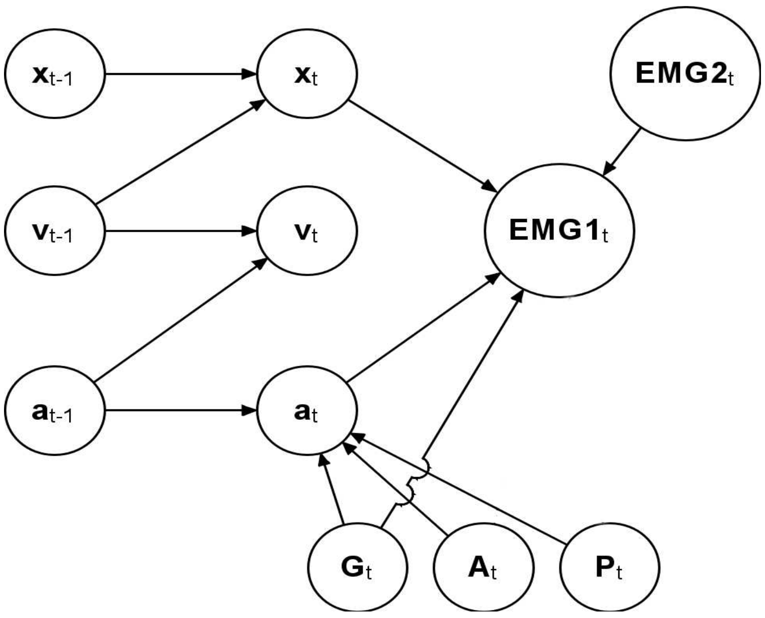


Figure 7.4: DBN structure for cyberglove model

Linear approximation of sine curve:

$$\ddot{\theta}_t = \ddot{\theta}_{t-1} \pm \frac{A_t}{P_t} \tag{7.11}$$

where A_t is the amplitude and slope resp. and P_t the period at time t, i.e. the former is based on a strong assumption of a sine formed angular acceleration curve. It is calculated using the angular acceleration from time $t-1$ as input to compute $\tilde{t}$ representing the position on the proposed sine curve for time t. The acceleration at time t is found as the evaluation of the proposed sine curve at time $\tilde{t}+1$.

The latter approach is a simple linear approximation to a sine curve and computes the angular acceleration at time t by adding a small amount $\pm\frac{A_t}{P_t}$ (depending on the gesture) to the acceleration at time $t-1$. This proposal is less explicit as it does not base its estimate at time t on a computed index.

The angle and the angular velocity are calculated as follows:

$$\dot{\theta}_t = \dot{\theta}_{t-1} + \ddot{\theta}_{t-1} \tag{7.12}$$
$$\theta_t = \theta_{t-1} + \dot{\theta}_{t-1} \tag{7.13}$$

The resulting graphical structure of the DBN is illustrated in Figure 7.4.

The combination of external nodes attached to a Kalman structure implies that we use a modified PF step to sample the external nodes and apply the different filters to the remaining nodes, just as in the watertank simulation (see chap. 6).

Note that, based on the experiments described in sec. 6.3.3, we are aware of the fact that the angular velocity node is *not* connected to the measurement nodes. This has a significant effect on PF and PFEKF. In PF the particles are weighted according to their likelihood. As the angular velocity is only indirectly linked to the measurement nodes through the angle node, PF is considering the angle and angular acceleration nodes to be the 'most important' as they directly influence the predicted measurement and thus the likelihood. This might imply incorrect particle weighting although an accurate angular velocity estimate is more likely to give a more accurate angle estimate.

In PFEKF the situation is much worse as the filter will not update the angular velocity estimates as the Kalman gain is 0 for this node (as the Jacobian of the measurement model is 0 for non-linked nodes, see sec. 4.5). Furthermore, it will influence the state covariance estimates for all nodes as the update of this value is also based on the Jacobian of the measurement model (see sec. 4.5). This implies that even when we are proposing a linear acceleration curve and thus have first order process and measurement models, we will observe different state covariance estimates in PFEKF and PFUKF.

7.3 Parameter selection

The reason for using two proposal distributions is that during our study of the angular acceleration curves we found that signals of high quality, i.e. where the subject managed to move his fingers in a nice, smooth way and thus keeping the maximal amount of signal info after the filter has been applied, the angular acceleration curves could be well fitted using sine curves with different amplitudes and periods. However, to not limit ourselves to this domain of signals, we also choose a very simple first order process model proposal which seemed to make a fair approximation to most of the curves of lower quality. Thus, we have a more explicit approach that relies on sine curves and asks for high quality signals and a more general approach that does not ask for the computation of a curve index and thus should have a better 'all-round' chance of fitting signals of various quality.

In both process models we need to know the period and the amplitude of the sine curve resp. the slope of the linear approximation. Once again, we choose to sample these values by drawing from a normal distribution with a predetermined mean and variance calculated as the mean and variance of the periods and amplitudes/slopes of 6 curves for the sine case (three for each gesture) and 12 curves for the linear case. These curves were manually fitted curves to 6 signals of high quality resp. 12 signals of lower quality.

Sine/high quality

$$A \sim \mathcal{N}(0.8, 0.1) \tag{7.14}$$

$$P \sim \mathcal{N}(250, 25) \tag{7.15}$$

Linear/lower quality

$$A \sim \mathcal{N}(0.7, 0.2) \tag{7.16}$$

$$P \sim \mathcal{N}(250, 25) \tag{7.17}$$

Furthermore, we could have sampled all other unknown constants in the physical measurement model, but this would make things too complicated for this simple approach. Instead we find a Minimum Mean Square estimate of the parameters for each of the two gestures using the measurement model to fit a curve taken as the mean of the same signals used in the sampling of the gesture and amplitude nodes.

Hence, our domain of gestures is based on an average of 18 gestures (a *Training Set* as it forms our prior knowledge) and it will therefore influence our results significantly. Optimally, we would like to do our parameter estimation based on a much larger number of signals without having to constrain ourselves to 'nice' manually selected signals, but the data recording showed that it was quite difficult to perform these movements in a consistent manner, especially keeping the signal length fairly constant and to make a smooth movement as one tends to make somewhat 'shaky' movements, i.e. the filtering had a significant influence as they were too strong for some signals removing part of the signal and too soft for others leaving noise and artifacts.

7.4 Results

As EMG measurements we used values from electrode pair 2 and 6 (see Figure 7.2) as they proved to be the most useful electrodes (most signal value and least noise).

An example of a data recording is shown in Figure 7.5, 7.6 and 7.7 illustrating the glove measurements (hidden values in the DBN) and the EMG measurements for the two electrodes used in the DBN in their raw and filtered form in Figure 7.8 and 7.9. In Figure 7.10 the EMG measurements are plotted in the same figure, notice how the signals overlap.

With the previous experiments in mind we limit ourselves to the use of the PF, PFEKF and PFUKF filters. Moving on to a real-life problem as opposed to the simulations we acknowledge the fact that our model structure, process/measurement equations and assumptions are very simple and approximate we find it useless to apply EKF and UKF as they have proven to be most valuable in scenarios where the model resembles the ground truth. Furthermore, we do not apply a Metropolis-Hastings move in any of the particle filters as this is a very computationally expensive procedure and is most valuable when we expect the filters to favor single particles and need to jitter the particles or move them to regions of higher likelihood. In this initial approach there is no guarantee that a MH move will improve our estimates and the computational trade-off is too large using signals of length ~ 250.

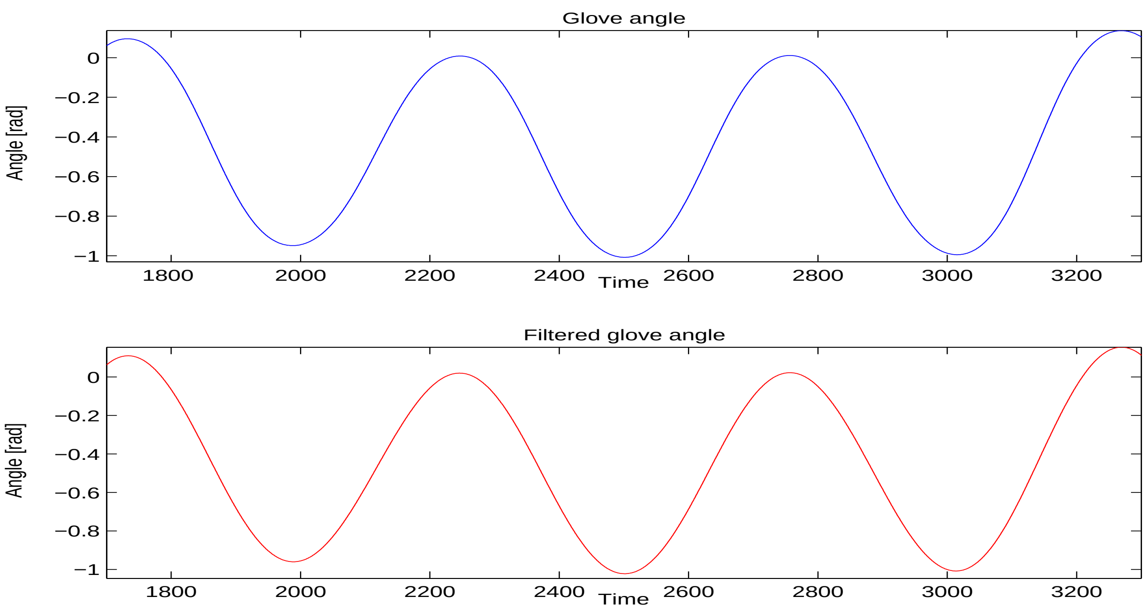


Figure 7.5: Example of glove angle and filtered glove angle measurements

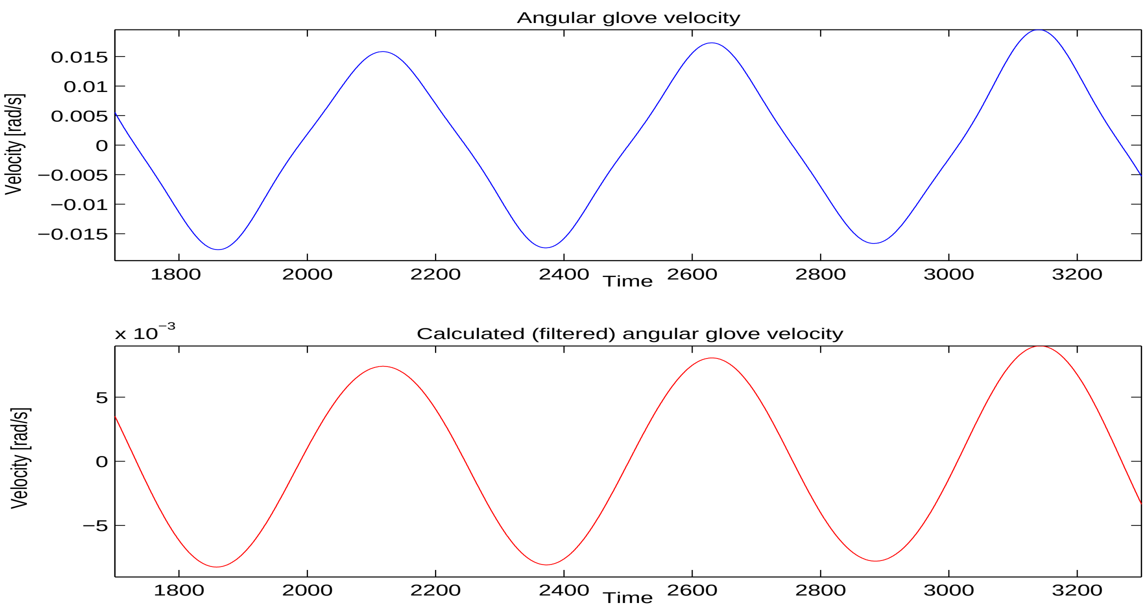


Figure 7.6: Example of angular glove velocity and filtered angular glove velocity measurements

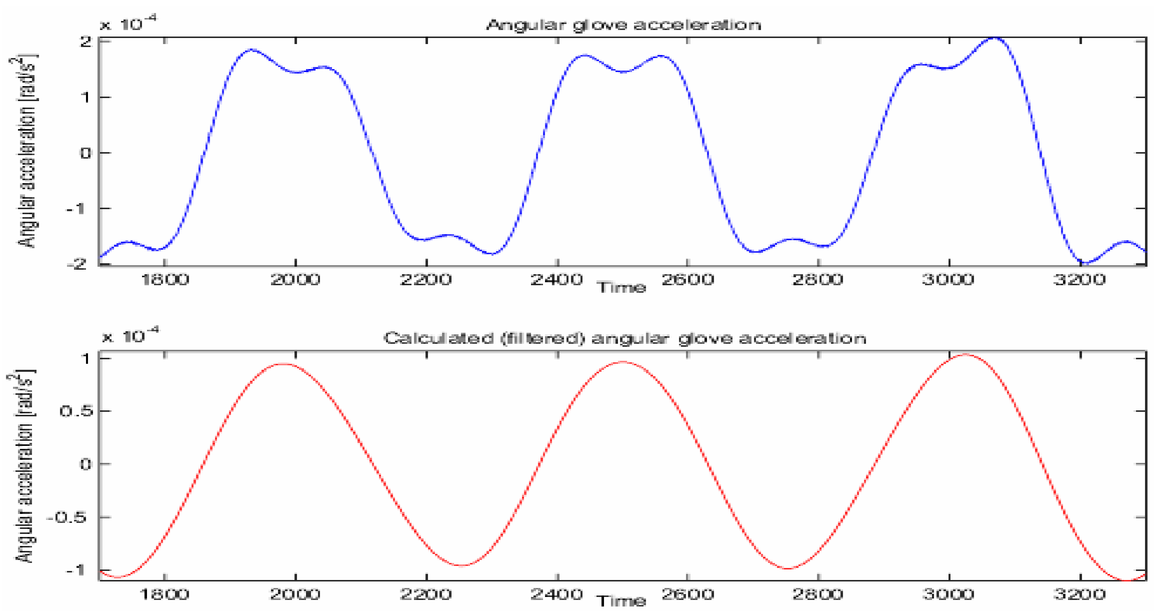


Figure 7.7: Example of angular glove acceleration and filtered angular glove acceleration measurements

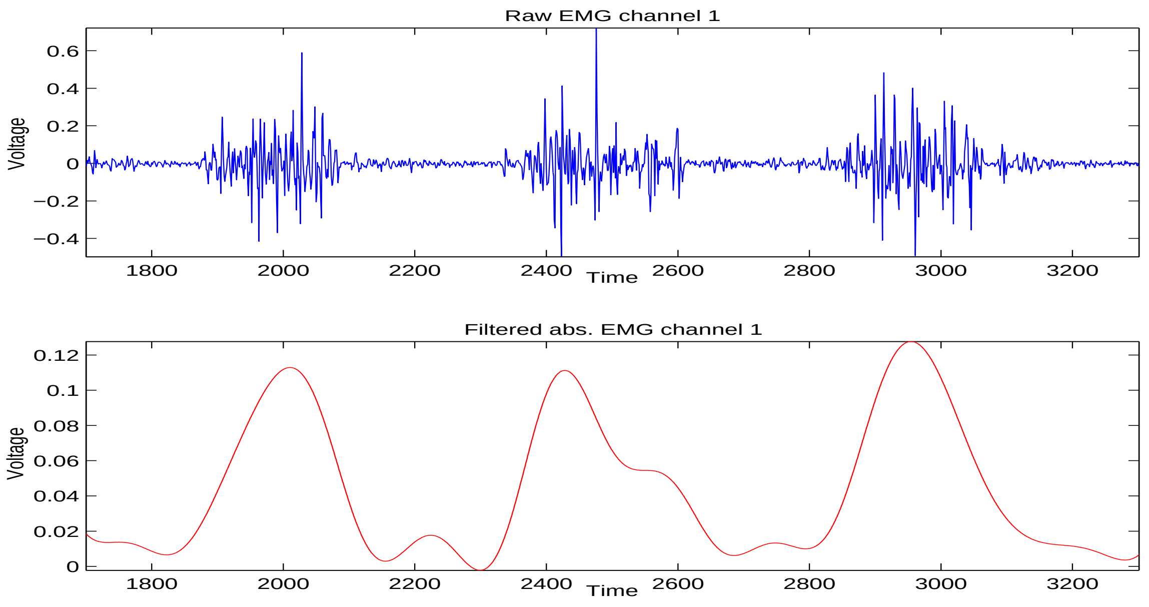


Figure 7.8: Example of raw and filtered value of abs. EMG measurements from electrode on the upper side of the lower arm, electrode 2

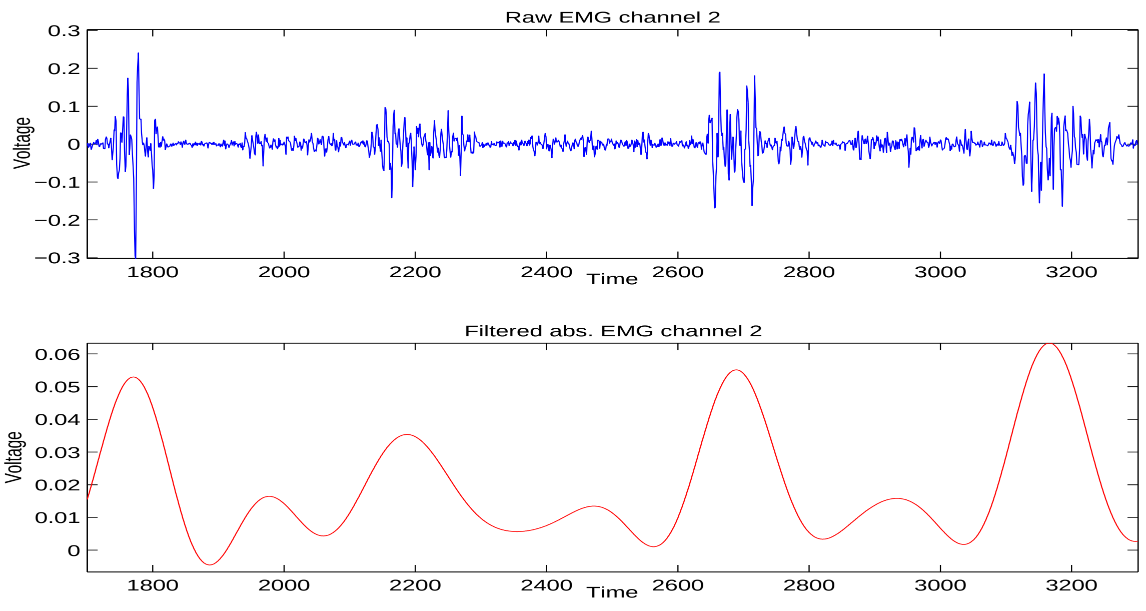


Figure 7.9: Example of raw and abs. filtered EMG measurements from electrode on the underside of the lower arm, electrode 6

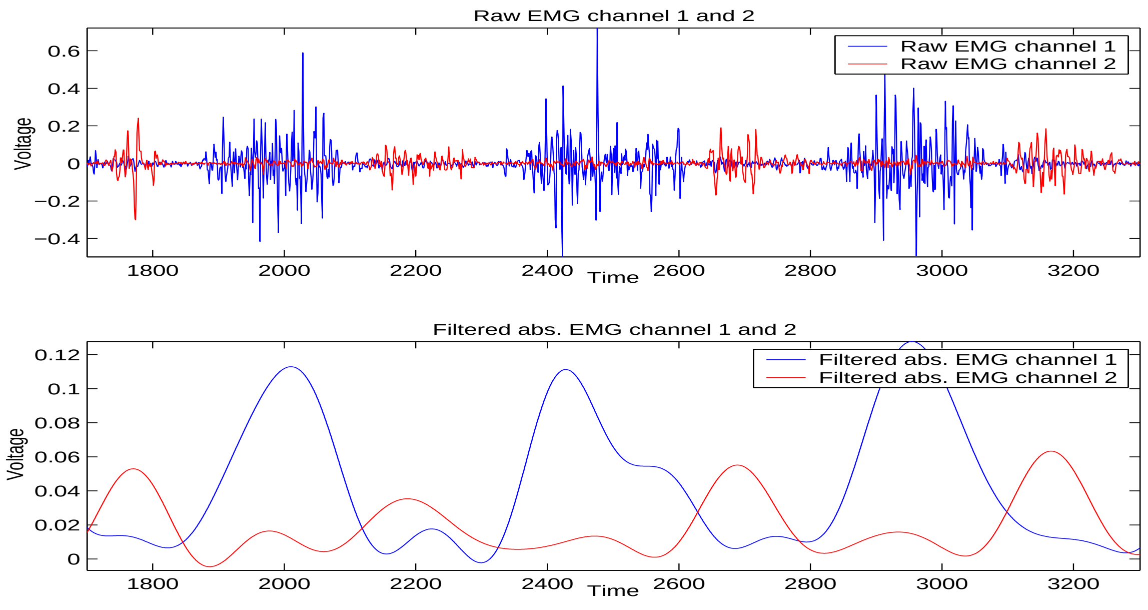


Figure 7.10: Example of raw and filtered value of abs. EMG measurements from electrode on the upper- (blue) and underside (red) of the lower arm

We do not discard PF and PFEKF based on our previous experiences. First of all, PFEKF and PFUKF are computationally much more expensive than PF, so if PF is able to estimate within reasonably boundaries we might not need to use the advanced algorithms. Secondly, evaluating on the likelihood only, PF will be a good indicator of the validity of our model as it is not able to move its predictions using a Kalman gain as PFEKF and PFUKF. PFEKF was included to make a non-simulated comparison of its performance with the PFUKF without the knowledge of the 'true' models. Furthermore, even though the measurement model is a first order relation, the process model is only a first order model when the angular acceleration proposal is linear. When the proposal is a sine curve, PFEKF relies on a first order Taylor approximation of the posterior mean and variance whereas PFUKF uses the true non-linear models (see sec. 4.1 and 4.2 resp.) and approximates the state random variable.

In Figure 7.11 a first (as used in EKF) and third order (for comparison) Taylor series approximation to a sine curve is shown. It is clear that a first order approximation is very poor and introduce estimation error in PFEKF.

Finally, the network structure may play a significant role as mentioned previously.

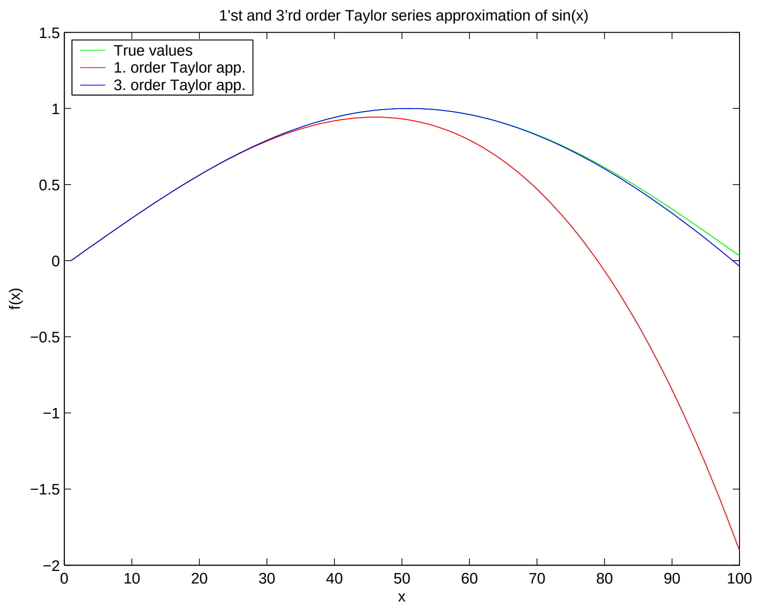


Figure 7.11: First and third order (for comparison) Taylor series approximation of a sine curve

The objective of these experiments is three-some:

- The main objective is to test whether any of our filters is able to recognize the performed gesture and estimate the glove measurements without too much error using our simple model

- Secondly, we would like to compare the PF, PFEKF and PFUKF filters on a real-life problem as they are very different approaches
- Finally, we would like to investigate the influence of the two angular acceleration proposal curves using signals of different quality

All tests were performed on signals not included in the Training Set and were conducted using residual resampling and 500 particles as the signals are computationally demanding with signals of length ~ 250. The final results were based on an average over 5 runs and in all experiments the values $\alpha = 1, \beta = 0$ and $\kappa = 0$ used in the UKF (see sec. 4.2) were used.

Our experiments are divided into subcategories to cross-validate our signals vs. angular acceleration curve proposal and gesture, i.e. a total of 8 tests as listed below:

- Open/close hand gesture, high-quality signal, sine curved angular acceleration process model
- Open/close hand gesture, high-quality signal, linear angular acceleration process model
- Open/close hand gesture, semi-quality signal, sine curved angular acceleration process model
- Open/close hand gesture, semi-quality signal, linear angular acceleration process model

Each case is treated in the order listed.

7.4.1 Open/close hand gesture, high-quality signal, sine curved angular acceleration process model

Using an open hand signal of high quality and proposing a sine curve as angular acceleration process model, we get the results shown in Table 7.1 showing the Mean Square Error of the glove measurement estimates and the percentage of correct gesture classifications. The tracking plot of the glove measurement estimates vs. their true values is shown in Figure 7.12.

The results using a close hand signal are shown in Table 7.2 and Figure 7.13.

Initially, by inspection of Table 7.1, Table 7.2 and Figure 7.12, Figure 7.13, we see that PFEKF and PFUKF are able to recognize the performed gesture in more than 93% of the time, which is more than acceptable. We also notice that both filters are able to track the angle with low estimation error though PFUKF and is the most accurate. In comparison, PF makes consistently less accurate estimates and in the close hand signal the estimates are rather poor.

Algorithm	RMS			True gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.0650	0.6363	0.0137	92.8%
PF - EKF proposal (PFEKF)	0.0521	1.0432	0.0095	93.6%
PF - UKF proposal (PFUKF)	0.0494	0.2557	0.0056	94.1%

Table 7.1: RMS values of state estimates and true gesture recognition percentage using a high quality open hand signal and proposing a sine curve as angular acceleration process model

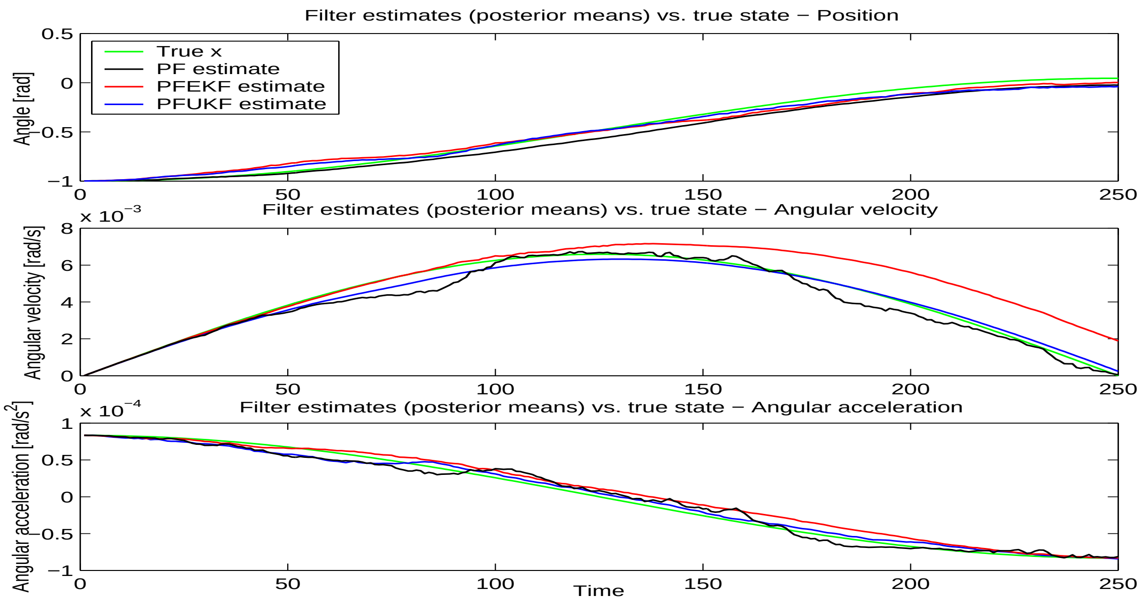


Figure 7.12: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a high quality open hand signal and proposing a sine curve as angular acceleration process model

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.3283	2.0025	0.0165	72.5%
PF - EKF proposal (PFEKF)	0.0731	1.0536	0.0224	93.7%
PF - UKF proposal (PFUKF)	0.0455	0.3245	0.0066	95.2%

Table 7.2: RMS values of state estimates and true gesture recognition percentage using a high quality close hand signal and proposing a sine curve as angular acceleration process model

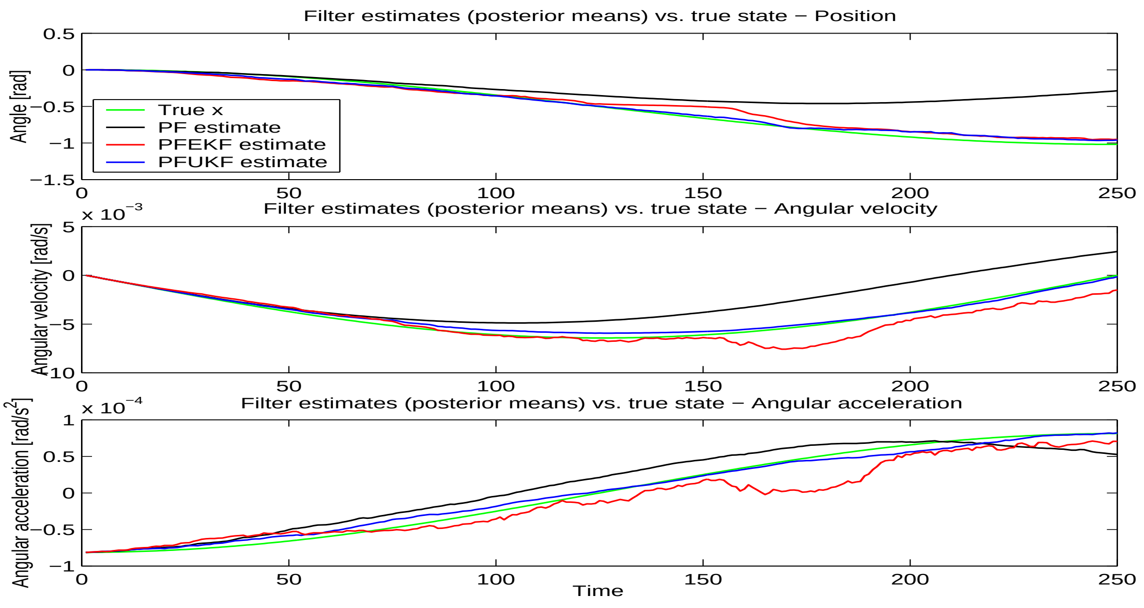


Figure 7.13: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a high quality close hand signal and proposing a sine curve as angular acceleration process model

The estimation of the angular velocity shows a somewhat different picture as only PFUKF is able to track the angular velocity within reasonable error boundaries. PFEKF's estimates of the state covariance are lower than PFUFK's due to the 1. order Taylor series approximation of the sine curve angular acceleration proposal and the missing link from the angular velocity node to the measurements causing the Jacobian of the measurement model w.r.t. this variable to be 0, see sec. 4.5. This implies that PFEKF is not able to correct its estimates of the angular acceleration as much as necessary. The error introduced in the angular acceleration estimates thus affects the estimates of the angular velocities as these are *based* on the angular acceleration estimates, see Figure 7.4.

Furthermore, the missing link from the angular velocity node to the measurements and the corresponding Jacobian of the measurement model implies that PFEKF is not updating the estimates of the angular velocity, see sec. 4.5. Hence, the angular velocity is based entirely on the angular acceleration value and follows the angular acceleration estimates nicely as seen in Figure 7.12, but this also implies that when ever it makes a poor estimate of the angular acceleration the error is directly transferred to the angular velocity. From $t \sim 60$ in Figure 7.12 and for $t \sim 150-200$ in Figure 7.13, the PFEKF angular acceleration estimates are consistently too high resp. too low and the angular velocity estimates become very inaccurate.

In PF, the poor angular velocity estimates are partly due to inaccurate angular acceleration estimates, but also due to the selected network structure. As explained previously, the missing link from the angular velocity node to the measurements implies that PF is weighting its particles based primarily on the angle and the angular acceleration estimates in the calculation of the likelihood and only weights the angular velocity indirectly through its connection to the angle. However, the missing link has less influence in PF than in PFKEF and the PF estimates of the angular velocity are thus more accurate than PFKEF's in the *open hand* case. In the *close hand* case, PFEKF's estimates are saved by the more accurate estimates of the angular acceleration compared to PF except for $t \sim 150 - 200$.

Once again, PFUKF makes the most accurate estimates of all hidden nodes taking advantage of its ability to capture the true posterior mean and covariance to second order and thus making a much better estimate of the sine curve. Furthermore, as demonstrated in the simple one-dimensional simulation in sec. 5 and the watertank problem in sec. 6.3.3 it is not suffering from the chosen network structure with its 'missing' link between the angular velocity node and the observation node. Finally, as also demonstrated in sec. 5 and 6.3.7 it is a much more robust filter when the proposed process and measurement models are far from the 'true' models.

7.4.2 Open/close hand gesture, high-quality signal, linear curved angular acceleration process model

Using an open hand signal of high quality and proposing a linear curve as angular acceleration process model, we get the results shown in Table 7.3 showing the Mean Square Error of the glove measurement estimates and the percentage of correct gesture classifications. The tracking plot of the glove measurement estimates vs. their true values is shown in Figure 7.14.

The results using a close hand signal are shown in Table 7.4 and Figure 7.15.

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.2625	1.0761	0.0294	80.3%
PF - EKF proposal (PFEKF)	0.0492	0.7465	0.0226	93.4%
PF - UKF proposal (PFUKF)	0.0460	0.6453	0.0184	93.9%

Table 7.3: RMS values of state estimates and true gesture recognition percentage using a high quality open hand signal and proposing a linear curve as angular acceleration process model

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.7852	5.072	0.0384	74.4%
PF - EKF proposal (PFEKF)	0.0485	1.2955	0.0260	93.4%
PF - UKF proposal (PFUKF)	0.0476	0.9552	0.0227	94.7%

Table 7.4: RMS values of state estimates and true gesture recognition percentage using a high quality close hand signal and proposing a linear curve as angular acceleration process model

When we compare the results of the sine curve proposal with the linear proposal, we notice how the much simpler linear proposal has a dramatic influence on the performance of the generic particle filter as the RMS values increase significantly all over and the recognition percentage drops to around 80%. This implies that our proposal is too simple to be of much use, when we are working on high quality signals and can not update our estimates.

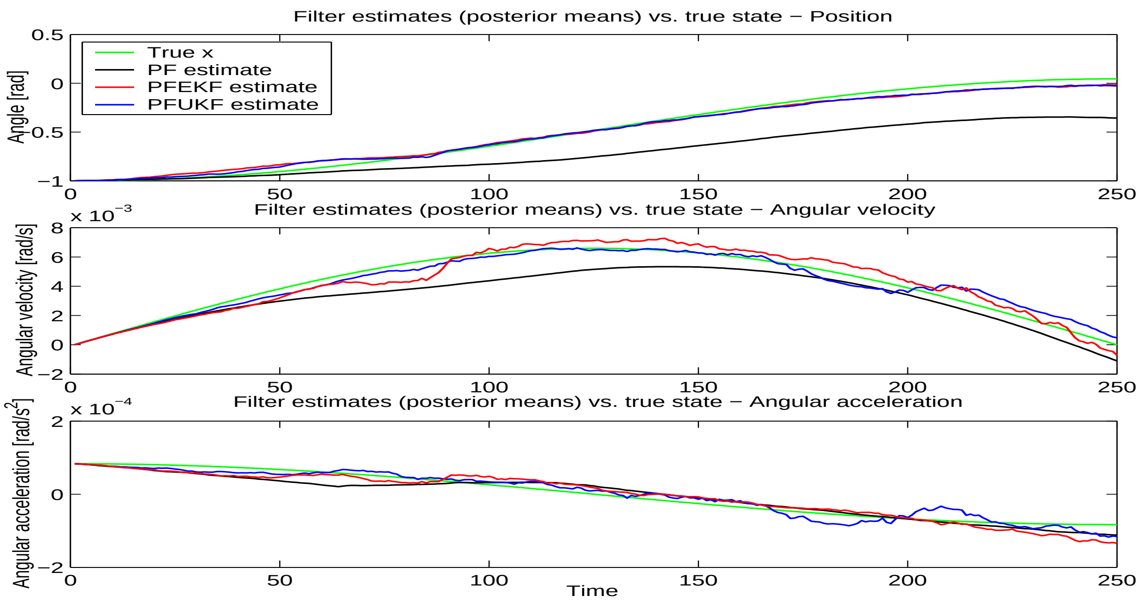


Figure 7.14: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a high quality open hand signal and proposing a linear curve as angular acceleration process model

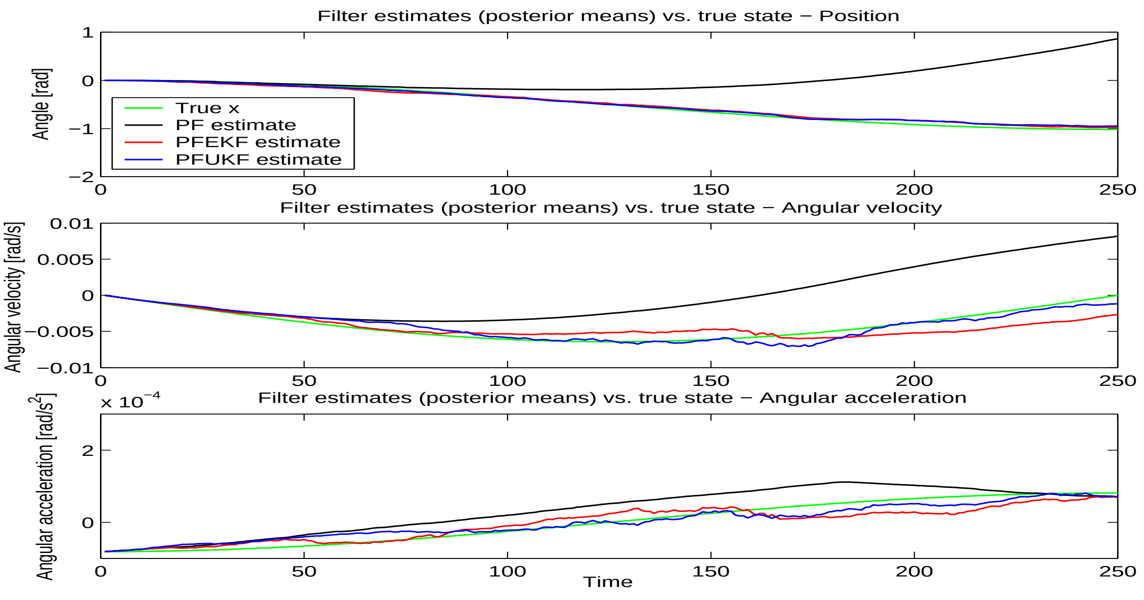


Figure 7.15: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a high quality close hand signal and proposing a linear curve as angular acceleration process model

The linear curve proposal also affect the results of the PFEKF and PFUKF algorithms, but in a less dramatic way. The angular acceleration estimates are less accurate in both cases and the angular velocity estimates are also less accurate in the PFUKF case, but actually more accurate in the PFEKF *open hand* case compared to the sine curve proposal. This is because the angular velocity estimates are based on the angular acceleration estimates that are too low compared to the ground truth and even though they are less accurate, they make the angular velocity estimates smaller and thus closer to the ground truth.

It is also interesting to notice how the estimates of the angle have not changed much for PFKEF and PFUKF with the linear curve proposal. Both algorithms seem to be able to correct the angle estimates fairly independent of the choice of angular acceleration curve proposal as the angle itself is directly connected to the measurements. In comparison, PF does not update its estimates and is not able to recognize the true gesture using a poor angular acceleration curve proposal.

7.4.3 Open/close hand gesture, semi-quality signal, sine curved angular acceleration process model

Using an open hand signal of semi-high quality and proposing a sine curved angular acceleration process model, we get the results shown in Table 7.5 showing the Mean Square Error of the glove measurement estimates and the percentage of correct gesture classifications. The tracking plot of the glove measurement estimates vs. their true values is shown in Figure 7.16.

The results using a close hand signal are shown in Table 7.6 and Figure 7.17.

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.2067	1.0752	0.0219	83,0%
PF - EKF proposal (PFEKF)	0.0941	3.0848	0.0194	94,0%
PF - UKF proposal (PFUKF)	0.0584	1.0070	0.0288	94,1%

Table 7.5: RMS values of state estimates and true gesture recognition percentage using a semi-high quality open hand signal and proposing a sine curve as angular acceleration process model

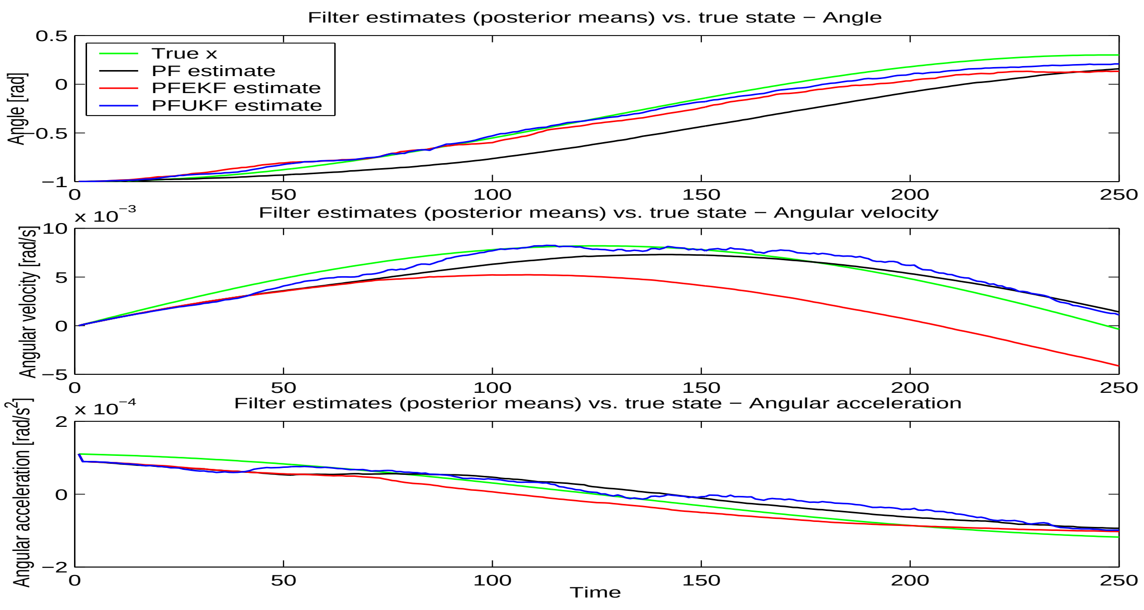


Figure 7.16: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a semi-high quality open hand signal and proposing a sine curve as angular acceleration process model

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.5827	2.6583	0.0384	58,0%
PF - EKF proposal (PFEKF)	0.4045	2.8022	0.0312	59,6%
PF - UKF proposal (PFUKF)	0.0555	0.9170	0.0244	96,4%

Table 7.6: RMS values of state estimates and true gesture recognition percentage using a semi-high quality close hand signal and proposing a sine curve as angular acceleration process model

First, we notice the effect of the poor angular acceleration process model as the PF estimates are very inaccurate all over. PFEKF and PFUKF are also suffering from the poor process model, especially PFEKF in the estimation of the angular velocity in the open hand signal and the angle and angular velocity in the close hand signal. PFUKF's ability to update the angular velocity estimates is once again demonstrated and affects the estimation of the angle significantly. The results might imply that in general (not distinguishing between signals of different quality) we are better off using a simpler angular acceleration curve process model as the more complex sine curve process model introduces significant error when the signal is of lower quality.

However, we also notice that the angular estimation in PFUKF is still very accurate and the gesture recognition percentage in PFEKF and PFUKF remains above 90% showing that the choice of angular acceleration process model does not influence the angle estimates significantly. The angular velocity estimates in PFUKF are also rather accurate whereas the estimates in PFEKF suffer from the poor angular acceleration estimates and the missing link.

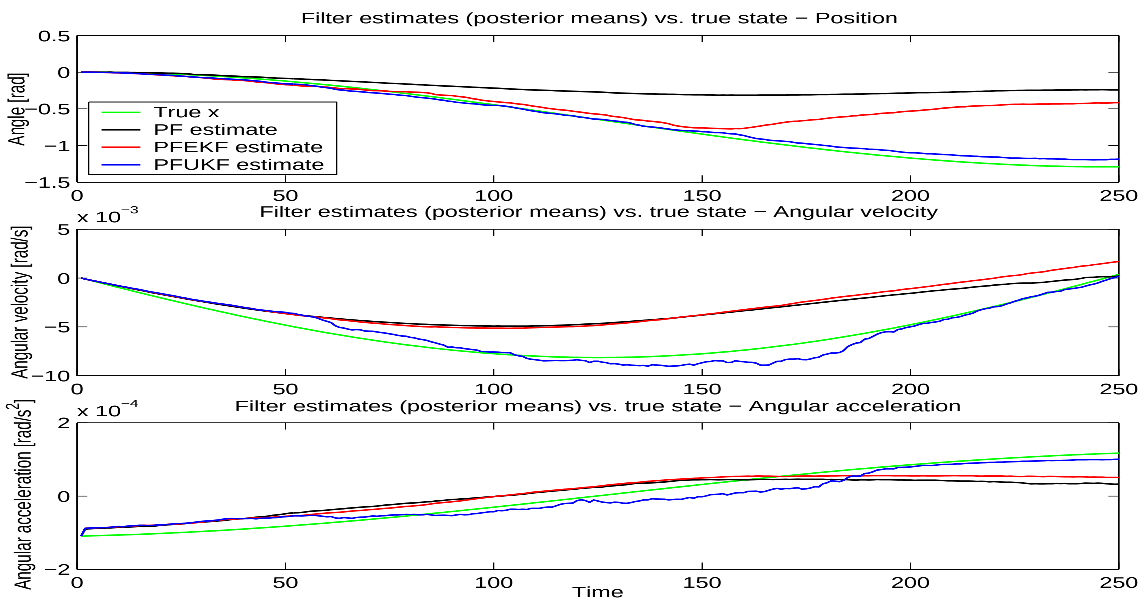


Figure 7.17: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a semi-high quality close hand signal and proposing a sine curve as angular acceleration process model

7.4.4 Open/close hand gesture, semi-quality signal, linear curved angular acceleration process model

Using an open hand signal of semi-high quality and proposing a linear curved angular acceleration process model, we get the results shown in Table 7.5 showing the Mean Square Error of the glove measurement estimates and the percentage of correct gesture classifications. The tracking plot of the glove measurement estimates vs. their true values is shown in Figure 7.16.

The results using a close hand signal are shown in Table 7.6 and Figure 7.17.

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.1441	0.9416	0.0190	89,4%
PF - EKF proposal (PFEKF)	0.0635	0.6256	0.0207	94,1%
PF - UKF proposal (PFUKF)	0.0612	0.5946	0.0127	94,2%

Table 7.7: RMS values of state estimates and true gesture recognition percentage using a semi-high quality open hand signal and proposing a linear curve as angular acceleration process model

Algorithm	RMS			Correct gesture recog.
	Angle	Vel. [1e-3]	Acc. [1e-3]	
Particle Filter - generic (PF)	0.5261	2.6449	0.0391	58,1%
PF - EKF proposal (PFEKF)	0.0572	1.4562	0.0285	96,1%
PF - UKF proposal (PFUKF)	0.0569	0.8239	0.0198	96,4%

Table 7.8: RMS values of state estimates and true gesture recognition percentage using a semi-high quality close hand signal and proposing a linear curve as angular acceleration process model

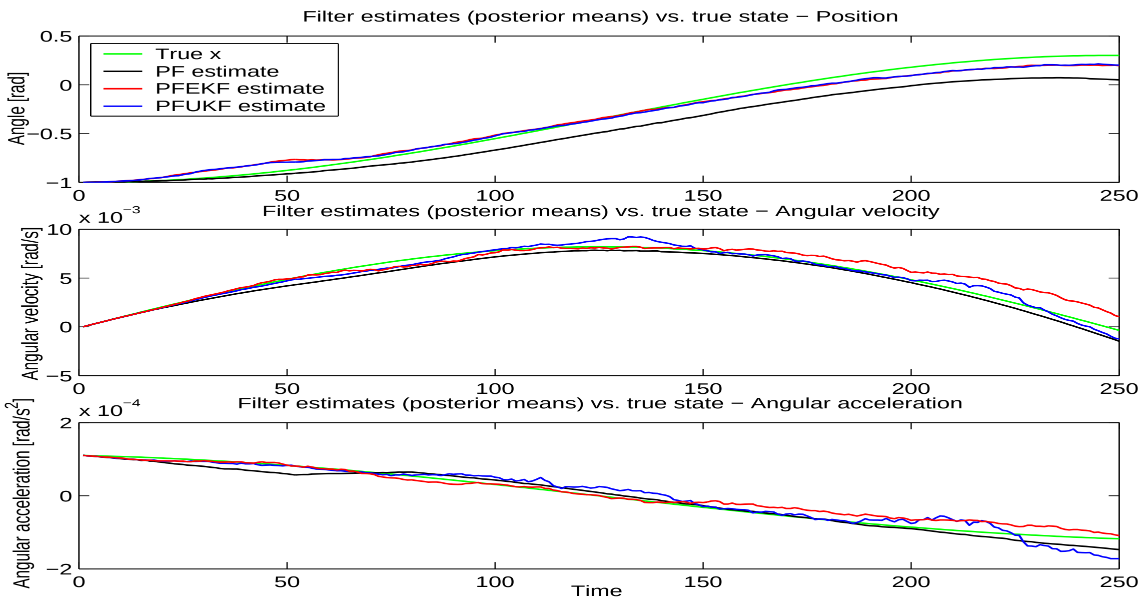


Figure 7.18: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a semi-high quality open hand signal and proposing a linear curve as angular acceleration process model

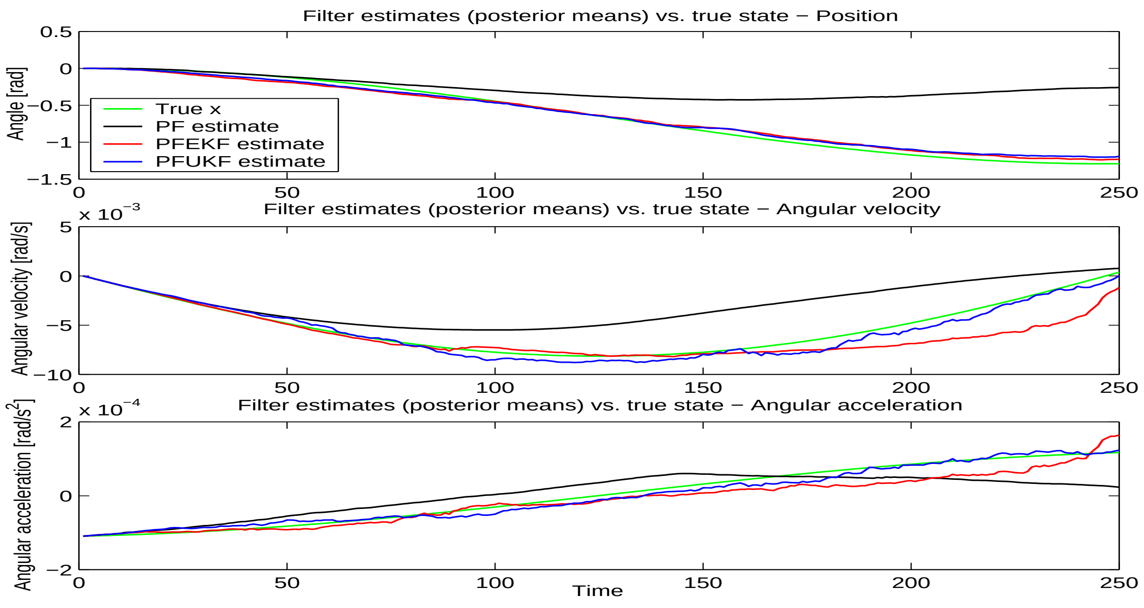


Figure 7.19: True angular, angular velocity and angular acceleration values vs. PF, PFEKF and PFUKF estimates resp. using a semi-high quality close hand signal and proposing a linear curve as angular acceleration process model

Substituting the sine curve process model with a linear curve has a significant effect on the accuracy of the state estimates. Especially PF in the open hand signal takes advantage of the much simpler process model that does not ask for a high quality signal with assumptions of a sine formed angular acceleration curve. However, it still fails to make accurate estimates in the close hand signal which implies that using a simple angular acceleration process model, improved estimation accuracy is *not* guaranteed, but depends heavily on the signal and if PF is able to sample in areas of high likelihood. A higher number of particles or a MC step might improve the situation.

PFEKF and PFUKF also show comparable or as in most cases increased estimate accuracy with reference to the previous experiment. Most dramatic improvements are seen in the angle estimates in PF and the angle and angular velocity estimates on PFEKF where the estimates were very poor using the sine curve process model.

The outcome is a highly improved gesture recognition percentage for PF and slightly improved results for PFEKF and PFUKF as the recognition percentages for these algorithms were already very high.

7.5 Discussion

In conclusion, we notice that once again PFUKF seems to outperform PF and PFEKF in almost every instance. Its superiority is especially evident in the estimation of the angular velocity where PF and PFEKF fail as the angular velocity is not directly linked to the observation node. It is also a much more accurate estimator when the proposed process model is vague and not able to account for the generation of the ground truth.

As a consequence, we note that the network and process/measurement model design is very critical for the effectiveness of PF and PFEKF, but less significant for PFUKF.

Furthermore, we notice that our proposed network structure and process/measurement models need improvement as we are not able to make accurate estimates using PF and PFEKF. Especially PF is a good indicator of the validity of the chosen network/models as the algorithm does not update its estimates and thus relies heavily on a well chosen network/model. However, it seems as if our proposed network/models is adequate enough to make reliable, accurate estimates using PFUKF as we are able to track all glove measurements and especially recognize the performed gesture with acceptable error margin.

To improve the model, we suggest that the angular velocity node is linked to the measurements for several reasons: First of all, the structure would be a great advantage for PF and PFEKF as explained earlier. Secondly, we notice in Figure 7.7 that our filtering of the angular acceleration values might be the reason for observing app. sine curved angular acceleration curves. If we take a look at the original angular acceleration curve (subplot 1) we notice an indication of a curve that could be divided into three: First the angular acceleration increases, then it is app. constant (besides from the dip that could be artifacts) and then it decreases. Hence, we have indications that we need to improve our data acquisition and our filtering, that we might need a more complex angular acceleration process model and that we definitively need to link the angular velocity node to the observation node. And it does seem intuitively plausible that there is a relation between the angular velocity and the measured EMG if we extend our mechanical view of muscular activity. When the muscles fibres are extended, it is not a frictionless movement which suggests that there is proportionality between the force necessary to move the wheel and the velocity of the wheel: Higher velocity implies increased friction and thus more force is needed.

Chapter 8

Conclusion

In this thesis we have made a theoretical treatment of the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), generic Particle Filter (PF, a.k.a. condensation, survival of the fittest, bootstrap filter, SIR, sequential Monte Carlo, etc.), Particle Filter with MCMC steps (PFMC), Particle Filter with EKF proposal (PFEKF) and MCMC steps (PFEKFMC), Particle Filter with UKF proposal (PFUKF) and MCMC steps (PFUKFMC).

We have also provided pseudo-code for implementations of all filtering techniques.

In our practical applications we have applied the different filters to a 2T-DBN in a simple simulation of a one-dimensional state estimation problem with a measurement model consisting of a first, second and third order stage. First, we experimented using the true process and measurement models and added Gaussian process and measurement noise which is the basic noise assumption of EKF and UKF. We showed how the EKF and UKF performed equally well in the first order stage, where EKF's first order Taylor series approximation was adequate to capture the true posterior mean and covariance (of the Gaussian approximation to the true posterior). In the second and third order stages, UKF was EKF superior due to its ability to completely capture the true posterior mean and variance to second order (of the Gaussian approximation to the true posterior).

Furthermore, it was shown that PFUKF was capable of saving the poor estimates from the core UKF by sampling from a Gaussian distribution with mean and variance proposed by UKF. In this scenario PF did not perform as well using a fairly low number of particles, but it was shown how the use of a Markov transition kernel could move the particles toward regions of higher likelihood and improve the estimates of PF. Finally, PFUKF proved to be by far the best filter of all and did not seem to gain much using the computationally expensive MCMC step. PFEKF and PFEKF was not capable of improving the EKF estimates as it was sampling from a proposal distribution with low, inaccurate variance.

Next, we violated the Gaussian noise assumption of EKF and UKF and draw process noise samples from a Gamma distribution. This scenario clearly demonstrated the low flexibility of EKF and UKF and showed that PF and especially PFMC was able to outperform these filters when the Gaussian assumption was violated. Furthermore, it was now even more obvious that the PFUKF (PFUKFMC) was the most reliable, robust and flexible filter.

Finally, we used a process model different from the true one and demonstrated how PFUKF (PFUKFMC) was the only filter to make accurate estimates. This was the most important indication of the superiority of this filter as most real-life applications involves approximations to the ’true’ (unknown) process and measurement models.

Next, we applied 6 of the 8 filters to a 2T-DBN simulation of a watertank problem.

We commented on implementational issues as well as parameter selections and initialization issues.

Initially, we compared two network designs and showed that PF and EKF only made good estimates of those continuous state variables that were connected directly to the observation nodes. UKF was able to update all state space variables regardless of their connection to the observation nodes and UKF thus performed equally well in both networks.

On behalf of this experiment, one network structure was discarded and the EKF based algorithms not used in the remaining experiments.

Next, we experimented using an increasing number of particles. Even though we experimented with a relatively small simulation time, we observed how PF and PFMC improved their tracking and estimation using an increased number of particles as the discrete failure nodes and the continuous state variables are updated simultaneously. We also noticed how UKF and PFUKF were able to track the true failure nodes and make more accurate estimates of the continuous state variables using a smaller number of particles than PF and PFMC. Finally, we observed that if we used enough particles to track the discrete failure nodes, the PFUKF could be less accurate than those obtained using core UKF as the UKF estimates were so close to the true state values that further sampling would move the sample mean away from the true state mean. This could be avoided using a large number of subparticles.

In the noise experiments, changing both process and measurement noise proposals did not influence the continuous state mean estimates using UKF. However, the UKF continuous state variance estimates increased with the size of the noise proposals. And the variance estimates depended more on the process noise proposal than the measurement noise proposal. Experiments also indicated that PFUKF should sample using a small posterior variance estimate, i.e. a small noise proposal, when UKF itself made accurate estimates to avoid moving the sample mean away from the true state mean.

The continuous state mean estimates using PF, PFMC, UKF and PFUKF were depending much more on the measurement noise level than the level of process noise. Large measurement noise levels made the UKF estimates poor and PFUKF was able to move the particles towards the true state, i.e. reducing the RMSE. Both PF and PFMC were performing worse than UKF and PFUKF with respect to RMSE and the number of wrong failure estimates for all tests with different measurement noise levels.

Finally, we experimented using a measurement model different from the true one by adding 5% to all the flow estimates. Again, PFUKF was capable of making more accurate state estimates than UKF when the algorithms were not working under optimal conditions. Furthermore, UKF and PFUKF were making more accurate state estimates than PF and PFMC.

Section 6.3.8 showed that we were able to track the discrete failure nodes with a fairly low number of particles using UKF. These results are to some extent is comparable with the work of Koller and Lerner in [21] in which a core PF algorithm is applied to the watertank problem using the old network structure, see Figure 6.2. We have shown that a different network structure and the use of UKF and PFUKF significantly improves the ability to track the true failure nodes and estimate the continuous state variables with a low number of samples.

In our real life application we applied PF, PFEKF and PFUKF to a 2T-DBN with angle, angular velocity and angular acceleration of a moving finger as hidden state variables and EMG measurements as observations. We showed that we were able to estimate the state variables and track the discrete valued gesture, amplitude and period nodes using a very simple mechanical model as process and measurement model. Especially, we were able to identify the performed gesture in more than 90$\%$ of the time using PFUKF. It was also shown that once again PFUKF was the most accurate filter in all instances and especially it did not suffer from the choice of network structure in which the angular velocity node is not directly connected to the observation node, see Figure 7.4. Furthermore, it was the most robust algorithm with regards to the choice of angular acceleration curve process model.

It was also realized that our model needs to be improved and a suggestion was made in which the angular velocity node is directly connected to the observation node proposing a relation between the angular velocity and the measured EMG.

Bibliography

[1] [Andrieu et al., 1998] A. Doucet (corresponding author), S. Godsill, C. Andrieu. *On Sequential Monte Carlo Sampling Methods for Bayesian Filtering* Technical report CUED/F-INFENG/TR 310, Dept. of Eng., Cambridge University, 1998.

[2] [Andrieu et al., 1999] C. Andrieu, J.F.G de Freitas, A. Doucet. *Sequential MCMC for Bayesian model selection* Technical report CUED/F-INFENG/TR 341, Dept. of Eng., Cambridge University, 1999.

[3] [Crisan, Doucet, 2000] D. Crisan, A. Doucet. *Convergence of generalized particle filters* Technical report CUED/F-INFENG/TR 381, Dept. of Eng., Cambridge University, 2000.

[4] [Cooper, 1987] G. Cooper. *Probabilistic inference using belief networks is NP-hard* Knowledge Systems Laboratory, 1987.

[5] [Crisan, 2000] D. Crisan. *Particle filters - a theoretical perspective* In Sequential Monte Carlo Methods in Practice, pages 17-41, A. Doucet, N. de Freitas, N.J. Gordon. Springer-Verlag. 2001.

[6] [Dean, Kanazawa, 1989] T. Dean, K. Kanazawa. *A model for reasoning about persistence and causation* Computational Intelligence vol. **5**(3): 142-150, 1989.

[7] [Dagum, Luby, 1993] P. Dagum, M. Luby. *Approximating probalistic inference in Bayesian belief networks is NP-hard* Artificial Intelligence vol. **60**(1): 141-153, 1993.

[8] [Doucet, 1999] A. Doucet. *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models* Application to Radiation Signals (Chapters 4 and 5 in English), Ph.D. thesis, University Paris-Sud, Orsay, France, 1999.

[9] [Doucet, 1998] A. Doucet. *On sequential simulation-based methods for Bayesian filtering* Technical report CUED/F-INFENG/TR 310, Dept. of Eng., Cambridge University, 1998.

[10] [Doucet et al., 1999] A. Doucet, N. J. Gordon., V. Krishnamurthy. *Particle filters for state estimation of jump Markov linear systems* Technical Report CUED/FINFENG/TR 359, Dept. of Eng., Cambridge University, 1999.

[11] [Doucet et al., 2000a] A. Doucet, N. de Freitas, K.P. Murphy, S. Russell *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks* In Proceedings of the 16'th Conference on Uncertainty in Artificial Intelligence, pages 176-183, Stanford, 2000.

[12] [Doucet et al., 2000b] A. Doucet, N. de Freitas, K. Murphy, S. Russell. *A Simple Tutorial on RBPF for DBNs* Complement to [Doucet00a], 2000.

[13] [Doucet, 2001] A. Doucet, N. de Freitas, N.J. Gordon. *An Introduction to Sequential Monte Carlo Methods* In Sequential Monte Carlo Methods in Practice: Chapter 1, A. Doucet, N. de Freitas, N.J. Gordon. Springer-Verlag. 2001.

[14] [Gilks, Berzuini, 1998] W.R. Gilks, C. Berzuini. *Monte Carlo inference for dynamic Bayesian models* Unpublished, Medical Research Council, Cambridge, UK, 1998.

[15] [Gordon et al., 1993] N.J. Gordon, D.J. Salmond, A.F.M. Smith. *Novel approach to nonlinear/non-Gaussian Bayesian state estimation* IEEE Proceedings-F vol. **140**(2): 107-113, 1993.

[16] [Green, 1995] P.J. Green. *Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination*, Biometrika vol. **82**: 711-732, 1995.

[17] [Jensen, 1996] F.V. Jensen. *Bayesian Networks Basics* Society for the Study of Artificial Intelligence and Simulation of Behaviour Quarterly Newsletter vol. **94**: 9-23, 1996.

[18] [J.F.G de Freitas, 1999] J.F.G. de Freitas. *Bayesian Methods for Neural Networks* Ph.D. thesis, Dept. of Eng., Cambridge University, 1999.

[19] [Julier, Uhlmann, 1996] S.J. Julier, J.K. Uhlmann. *A General Method for Approximating Nonlinear Transformations of Probability Distributions* Technical report, RRG, Dept. of Eng. Sc., University of Oxford, 1996.

[20] [Julier, Uhlmann, 2002] S.J. Julier, J.K. Uhlmann. *The Scaled Unscented Transformation*, Proceedings of the IEEE American Control Conference, 8-10 May, pages 4555-4559, 2002

[21] [Koller, Lerner, 2000] D. Koller and U. Lerner. *Sampling in Factored Dynamic Systems*, In Sequential Monte Carlo Methods in Practice: 445-464, A. Doucet, N. de Freitas, N.J. Gordon. Springer-Verlag. 2001.

[22] [Lerner, 2002] U. Lerner. *Hybrid Bayesian Networks for reasoning about complex systems* Ph.D. thesis, chapter 1,3,8, Dep. of Comp. Sc., Stanford University, 2002.

[23] [Lerner et al., 2002] U. Lerner, B. Moses, M. Scott, S. McIlraith and D. Koller. *Monitoring a Complex Physical System using a Hybrid Dynamic Bayes Net* In Proceedings of the 18'th Annual Conference on Uncertainty in AI (UAI), pages 301-310, 2002.

[24] [Liu et al., 2000] J. Liu, R. Chen and T. Logvinenko. *A theoretical framework for sequential importance sampling with resampling* Technical report, Dept. of Stat., Stanford University, 2000.

[25] [MacKay, 2002] D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms* Draft 3.1.2. October 20, 2002, Chapter 32-33.

[26] [v.d. Merwe et al., 2000] R. v.d. Merwe, A. Doucet, N. de Freitas and E. Wan. *The Unscented Particle Filter* Technical report CUED/F-INFENG/TR-380, Dept. of Eng., Cambridge University, 2000.

[27] [Murphy, 1998] K.P. Murphy. *Inference and Learning in Hybrid Bayesian Networks* Report No. UCB/CSD-98-990, EECS, UC Berkeley, 1998.

[28] [Murphy, 2002] K.P. Murphy. *Dynamic Hybrid Bayesian Networks* To appear in Probabilistic Graphical Models, M. Jordan.

[29] [N. de Freitas, 2002] N. de Freitas. *Rao-blackwellised Particle Filtering for Fault Diagnosis*, IEEE Aerospace 176-183, 2002

[30] [Settimi et al., 1999] R. Settimi, J.Q. Smith and A.S. Gargoum. *Approximate Learning in Complex Dynamic Bayesian Networks* Proceedings of the 15'th Annual Conference on Uncertainty in Artificial Intelligence (UAI-1999), pages 585-593, 1999.

[31] [Stone, 1996] C. Stone. *A course in probability and statistics* Duxbury Press, 1996.

Index

Appendix A

Filtering

In general, filtering is the problem of estimating the state of a system using a set of observations that becomes available on-line. This problem is solved by modelling the evolution of the system and the noise on the measurements. There exist many modelling strategies and filtering algorithms, but the resulting models most often show complex non-linearities and non-Gaussian distributions which rules out analytical solutions.

A.1 Dynamic State Space Model

The general state space model (without control input) consists of a *state transition or state process* model and *a state measurement* model

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{A.1}$$

$$p(\mathbf{y}_t|\mathbf{x}_t) \tag{A.2}$$

where $\mathbf{x}_t \in \Re^{n_x}$ are the states (hidden variables or parameters) of the system at time t and $\mathbf{y}_t \in \Re^{n_y}$ are the observations. The state transitions are a first order Markov process and the observations are assumed to be independent given the states.

For example, a non-linear, non-Gaussian model can be expressed as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{A.3}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{n}_t) \tag{A.4}$$

with $\mathbf{y}_t \in \Re^{n_y}$ being the output observations, $\mathbf{x}_t \in \Re^{n_x}$ the states of the system, $\mathbf{v}_t \in \Re^{n_v}$ the process noise and $\mathbf{n}_t \in \Re^{n_n}$ the measurement noise.

The mappings $f : \Re^{n_x} \times \Re^{n_v}$ and $h : \Re^{n_y} \times \Re^{n_n}$ represent the deterministic process and measurement models and $p(\mathbf{x}_0)$ is the prior distribution at time $t = 0$.

Our goal is to compute the filtering density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ recursively to avoid computing the complete posterior density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Thus we avoid keeping track of the complete history of the states and are still able to compute estimates of the mean, confidence intervals etc. of the systems states.

A.2 Extended Kalman Filter

In the Extended Kalman Filter (EKF) the standard Kalman filter (for linear systems) is applied to non-linear systems with additive white noise by continually updating a linearization around the previous state estimate, starting with an initial guess, i.e. it is a minimum mean-square-error (MMSE) estimator based on the Taylor series expansion of the non-linear functions **f** and **h** around the estimates $\bar{\mathbf{x}}_{t|t-1}$ of the states $\mathbf{x}_t$, e.g.

$$\mathbf{f}(\mathbf{x}_t) = \mathbf{f}(\bar{\mathbf{x}}_{t|t-1}) + \frac{\partial \mathbf{f}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\bigg|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}(\mathbf{x}_t - \bar{\mathbf{x}}_{t|t-1}) + \ldots \tag{A.5}$$

Using only the linear expansion terms, the update equations for the mean $\bar{\mathbf{x}}$ and covariance **P** of the Gaussian approximation to the posterior distribution of the states become

$$\begin{aligned}
\bar{\mathbf{x}}_{t|t-1} &= \mathbf{f}(\bar{\mathbf{x}}_{t-1}, 0) && \text{(A.6)}\\
\mathbf{P}_{t|t-1} &= \mathbf{F}_t\mathbf{P}_{t-1}\mathbf{F}_t^T + \mathbf{G}_t\mathbf{Q}_t\mathbf{G}_t^T && \text{(A.7)}\\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}\mathbf{H}_t^T[\mathbf{U}_t\mathbf{R}_t\mathbf{U}_t^T + \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T]^{-1} && \text{(A.8)}\\
\bar{\mathbf{x}}_t &= \bar{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}, 0)) && \text{(A.9)}\\
\mathbf{P}_t &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} && \text{(A.10)}
\end{aligned}$$

where $\mathbf{K}_t$ is the Kalman gain, **Q** is the variance of the process noise (assumed to be zero-mean Gaussian), **R** is the variance of the measurement noise (assumed to be zero-mean Gaussian), $\mathbf{F}_t \triangleq \frac{\partial \mathbf{f}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\Big|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}$ and $\mathbf{G}_t \triangleq \frac{\partial \mathbf{f}(\mathbf{v}_t)}{\partial \mathbf{v}_t}\Big|_{(\mathbf{v}_t=\bar{\mathbf{v}})}$ are the Jacobians of the process model and $\mathbf{H}_t \triangleq \frac{\partial \mathbf{h}(\mathbf{x}_t)}{\partial \mathbf{x}_t}\Big|_{(\mathbf{x}_t=\bar{\mathbf{x}}_{t|t-1})}$ and $\mathbf{U}_t \triangleq \frac{\partial \mathbf{h}(\mathbf{n}_t)}{\partial \mathbf{n}_t}\Big|_{(\mathbf{n}_t=\bar{\mathbf{n}})}$ are the Jacobians of the measurement model.

A.3 Unscented Kalman Filter

As the EKF only uses the first order terms of the Taylor series expansion of the non-linear functions, it may introduce significant errors in the estimations of the posterior distribution of the states. Especially if the models are highly non-linear where the local linearity assumptions do not hold.

The Unscented Kalman Filter (UKF, [20]) is a recursive MMSE estimator that does not approximate the non-linear process and measurement models, but uses the true models and approximates the distribution of the state random variable. The state distribution is still represented by a Gaussian random variable, but by a minimal set of deterministically chosen sample points that completely capture the true mean and covariance of the Gaussian random variable. When this variable is propagated through the true non-linear system, it captures the true mean and covariance to the second order for any non-linearity.

A.3.1 Unscented Transformation

To calculate the statistics of a random variable undergoing a non-linear transformation, as required by the UKF, the unscented transformation (UT) is used. UT is based on the principle that it is easier to approximate a probability distribution than an arbitrary non-linear function ([19]).

Let **x** be a n_x dimensional random variable propagated through an arbitrary non-linear function **g** to generate **y**

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \tag{A.11}$$

Assume **x** to have mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_x$. The first two moments of **y** are calculated by first deterministically choosing $2n_x + 1$ weighted samples or *sigma points* $S_i = \{W_i, \boldsymbol{\chi}_i\}$ so as to completely capture the true mean and covariance of the prior random variable **x** as follows ([26])

$$\begin{array}{lll} \boldsymbol{\chi}_0 = \bar{\mathbf{x}} & W_0 = \kappa/(n_x+\kappa) & i = 0 \\ \boldsymbol{\chi}_i = \bar{\mathbf{x}} + \left(\sqrt{(n_x+\kappa)\mathbf{P}_x}\right)_i & W_i = 1/\{2(n_x+\kappa)\} & i = 1,\dots,n_x \\ \boldsymbol{\chi}_i = \bar{\mathbf{x}} - \left(\sqrt{(n_x+\kappa)\mathbf{P}_x}\right)_i & W_i = 1/\{2(n_x+\kappa)\} & i = n_x+1,\dots,2n_x \end{array} \tag{A.12}$$

where κ is a scaling parameter and $\left(\sqrt{(n_x+\kappa)\mathbf{P}_x}\right)_i$ is the i'th row or column of the matrix square root of $(n_x+\kappa)\mathbf{P}_x$ and W_i is the weight associated with the i'th point s.t. $\sum_{i=0}^{2n_x} W_i = 1$. The sigma points are propagated through the non-linear function

$$\boldsymbol{Y}_i = g(\boldsymbol{\chi}_i), \quad i = 0, ..., 2n_x \tag{A.13}$$

The estimated mean and covariance of **y** is

$$\overline{\boldsymbol{y}} = \sum_{i=0}^{2n_x} W_i \boldsymbol{Y}_i \tag{A.14}$$

$$\boldsymbol{P}_y = \sum_{i=0}^{2n_x} W_i \left(\boldsymbol{Y}_i - \overline{\boldsymbol{y}}\right)\left(\boldsymbol{Y}_i - \overline{\boldsymbol{y}}\right)^T \tag{A.15}$$

which are accurate to the second order (third order for Gaussian priors) of the Taylor series expansion of $\mathbf{g}(\mathbf{x})$ for any non-linear function ([26]). The errors introduced in higher order terms are scaled by the parameter κ.

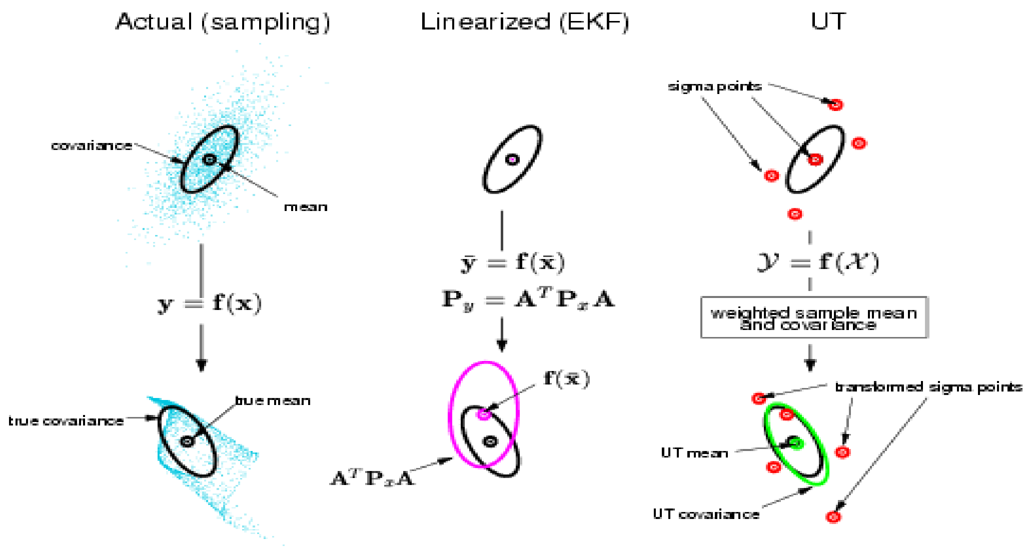

Actual (sampling)
Linearized (EKF)
UT
covariance
mean
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
true covariance
true mean
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
sigma points
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
transformed sigma points
UT mean
UT covariance

Figure A.1: Unscented Transformation vs. the linearization approach in EKF. 5000 Gaussian distributed samples are propagated through a non-linear function and the true posterior mean and covariance is calculated in the left plot. In the middle plot, the posterior mean and covariance is approximated by the linearization approach in EKF with significant error. Finally, the right plot shows the much more accurate UT estimates with almost no bias error (from [26]).

In the above selection scheme, the radius of the sphere bounding the sigma points increases as the dimension of the state space increases. Thus, in order to preserve the true mean and covariance, we are sampling non-local effects, which may cause problems if we are working with severe non-linearities. To address this matter, the sigma points can be scaled towards or away from the mean of the prior distribution depending of the value of κ:

The distance of the i'th sigma point from $\bar{\mathbf{x}}$, $|\boldsymbol{\chi}_i - \bar{\mathbf{x}}|$, is proportional to $\sqrt{(n_x+\kappa)}$. With $\kappa = 0$, the distance is proportional to $\sqrt{n_x}$, with $\kappa > 0$ the points are scaled away from $\bar{\mathbf{x}}$ and with $\kappa < 0$ the points are scaled towards $\bar{\mathbf{x}}$. In the case of $\kappa = 3 - n_x$, the dimensional scaling invariance is obtained by cancelling the effect of n_x, but if $\kappa = 3 - n_x < 0$, the weight $W_0 < 0$ may cause the covariance to be non-positive semidefinite. To solve this problem, the scaled unscented transformation (SUT) was developed ([20]).

A.3.2 The scaled unscented transformation

In SUT, the original sigma points are replaced by a transformed set given by

$$\boldsymbol{\chi}_i^{'} = \boldsymbol{\chi}_0 + \alpha(\boldsymbol{\chi}_i - \boldsymbol{\chi}_0), \quad i = 0, \ldots, 2n_x \tag{A.16}$$

where α is a positive scaling parameter. Choosing a low value for α decreases the effect of higher order terms. By applying the UT to an auxiliary random variable propagation problem related to the original model of equation (A.11) as given below

$$\mathbf{z} = \mathbf{g'(x)} = \frac{\mathbf{g}[\bar{\mathbf{x}} + \alpha(\mathbf{x} - \bar{\mathbf{x}})] - \mathbf{g}(\bar{\mathbf{x}})}{\alpha^2} + \mathbf{g}(\bar{\mathbf{x}}) \tag{A.17}$$

the sigma point scaling can be controlled without the risk of a non-positive semidefinite covariance matrix. The Taylor series expansion of $\bar{\mathbf{z}}$ and $\mathbf{P}_z$ corresponds to that of $\bar{\mathbf{y}}$ and $\mathbf{P}_y$ to the second order and the higher order term scaling determined by α. Thus, the same accuracy is obtained with the scaled transformation with the advantage that higher order terms can be appropriately scaled by the choice of α.

Combining the sigma point selection and scaling into a single step as follows can reduce the number of calculations.

Let

$$\lambda = \alpha^2(n_x + \kappa) - n_x \tag{A.18}$$

and select the sigma points using

$$\begin{aligned}
\boldsymbol{\chi}_0 &= \bar{\mathbf{x}} \\
\boldsymbol{\chi}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i, \quad i = 1, ..., n_x \\
\boldsymbol{\chi}_i &= \bar{\mathbf{x}} - \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i, \quad i = n_x + 1, ..., 2n_x \\
W_0^{(m)} &= \lambda/(n_x+\lambda) \\
W_0^{(c)} &= \lambda/(n_x+\lambda) + (1 - \alpha^2 + \beta) \\
W_i^{(m)} &= W_i^{(c)} = 1/\{2(n_x+\lambda)\}, \quad i = 1, \ldots, 2n_x
\end{aligned} \tag{A.19}$$

The weighting on the zero'th sigma point directly affects the errors in the fourth and higher order terms for symmetric prior distributions ([20]). As the parameter β affects the weighting of the zero'th sigma point for calculation of the covariance, this parameter allows for control of higher order errors if prior knowledge of the distribution of **x** is available.

In summary, the SUT is as follows

1. Choose $\kappa \geq 0$ to ensure positive semi-definiteness of the covariance matrix. Choose $0 \leq \alpha \leq 1$ to control the size of the sigma point distribution which should not be too large to avoid sampling non-local effects. Choose the weighting term $\beta \geq 0$ to incorporate prior knowledge (for Gaussian prior the optimal choice is $\beta = 2$, [26]).
2. Compute the $2n_x + 1$ scaled sigma points and weights $\boldsymbol{S} = \{\boldsymbol{W}, \boldsymbol{\chi}\}$ using (A.18) and (A.19).
3. Propagate the sigma points using (A.13).
4. Compute the mean $\bar{\mathbf{y}}$ and covariance $\bar{\mathbf{P}}_y$ using (A.14).

A.3.3 Implementation

In the Unscented Kalman Filter (UKF), the SUT is applied to the state random variable defined as the concatenation of the original state and noise variables, i.e. $\mathbf{x}_t^a = \left[\mathbf{x}_t^T \mathbf{v}_t^T \mathbf{n}_t^T\right]^T$ yielding a sigma point matrix $\boldsymbol{\chi}_t^a$. The complete pseudo algorithm is given below

1. Initialization

$$\bar{\mathbf{x}}_0 = E[\mathbf{x}_0] \tag{A.20}$$

$$\mathbf{P}_0 = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T] \tag{A.21}$$

$$\bar{\mathbf{x}}_0^a = E[\mathbf{x}^a] = [\bar{\mathbf{x}}_0^T \mathbf{0}\ \mathbf{0}]^T \tag{A.22}$$

$$\mathbf{P}_0^a = E[(\mathbf{x}_0^a - \bar{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \bar{\mathbf{x}}_0^a)^T] = \begin{pmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{pmatrix} \tag{A.23}$$

2. For $t \in \{1, \ldots, \infty\}$

   (a) <u>Calculate sigma points</u>

$$\boldsymbol{\chi}_{t-1}^a = \left[\ \bar{\mathbf{x}}_{t-1}^a \quad \bar{\mathbf{x}}_{t-1}^a \pm \sqrt{(n_a + \lambda)\mathbf{P}_{t-1}^a}\ \right] \tag{A.24}$$

(b) Time update

$$\begin{aligned}
\boldsymbol{\chi}^x_{t|t-1} &= \mathbf{f}(\boldsymbol{\chi}^x_{t-1}, \boldsymbol{\chi}^v_{t-1}) && \text{(A.25)}\\
\bar{\mathbf{x}}_{t|t-1} &= \sum_{i=0}^{2n_a} W_i^{(m)} \chi^x_{i,t|t-1} && \text{(A.26)}\\
\mathbf{P}_{t|t-1} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\chi^x_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}][\chi^x_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}]^T && \text{(A.27)}\\
\gamma_{t|t-1} &= \mathbf{h}(\boldsymbol{\chi}^x_{t|t-1}, \boldsymbol{\chi}^n_{t-1}) && \text{(A.28)}\\
\bar{\mathbf{y}}_{t|t-1} &= \sum_{i=0}^{2n_a} W_i^{(m)} \gamma^v_{i,t|t-1} && \text{(A.29)}
\end{aligned}$$

(c) Measurement update

$$\begin{aligned}
\mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}][\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}]^T && \text{(A.30)}\\
\mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} &= \sum_{i=0}^{2n_a} W_i^{(c)} [\chi_{i,t|t-1} - \bar{\mathbf{x}}_{t|t-1}][\gamma_{i,t|t-1} - \bar{\mathbf{y}}_{t|t-1}]^T && \text{(A.31)}\\
\mathbf{K}_t &= \mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} \mathbf{P}^{-1}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} && \text{(A.32)}\\
\bar{\mathbf{x}}_t &= \bar{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1}) && \text{(A.33)}\\
\mathbf{P}_t &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} \mathbf{K}_t^T && \text{(A.34)}
\end{aligned}$$

where $\mathbf{x}^a = \begin{bmatrix} \mathbf{x}^T & \mathbf{v}^T & \mathbf{n}^T \end{bmatrix}^T$, $\boldsymbol{\chi}^a = \begin{bmatrix} (\boldsymbol{\chi}^x)^T & (\boldsymbol{\chi}^v)^T & (\boldsymbol{\chi}^n)^T \end{bmatrix}^T$, λ is the composite scaling parameter, $n_a = n_x + n_v + n_n$, **Q** is the process noise covariance, **R** is the measurement noise covariance, **K** is the Kalman gain and W_i are the weights.

From a computational perspective, the UKF is superior to EKF, as it does not require explicit calculation of Jacobians (or Hessians), but computes a covariance matrix square root which can be done using a Cholesky factorization in order $n_x^3/6$. However, by expressing the covariance matrices recursively, this can be done in order n_x^2 using a recursive update to the Cholesky factorization ([26]).

A.4 Particle Filtering

EKF and UKF both rely on a Gaussian approximation. In this section we present a method that does not require this assumption, but presents other problematic issues. To overcome some of these problems, the particle filtering strategy is combined with EKF and UKF in section A.7 and A.8 resp.

A.4.1 Monte Carlo simulation

In Monte Carlo simulation, the posterior distribution is approximated by an empirical estimate computed using a set of N weighted particles (samples) $\{\mathbf{x}_{0:t}^{(i)}; i = 1, \dots, N\}$ drawn from the posterior distribution

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N} \sum_{1=1}^{N} \delta_{\boldsymbol{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{A.35}$$

where $\delta(\cdot)$ denotes the Dirac delta function. Hence, the expectation

$$\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) = \int \mathbf{g}_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \tag{A.36}$$

is approximated by

$$\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}_t(\mathbf{x}_{0:t}^{(i)}) \tag{A.37}$$

where the particles $\mathbf{x}_{0:t}^{(i)}$ are assumed to be i.i.d. Using the law of large numbers we get

$$\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} \quad \xrightarrow[N\to\infty]{a.s.} \quad \mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) \tag{A.38}$$

where $\xrightarrow[N\to\infty]{a.s.}$ denotes almost surely converge.

If the posterior variance of $\mathbf{g}_t(\mathbf{x}_{0:t})$ is bounded, i.e. $var_{p(\cdot|\mathbf{y}_{1:t})}\left(\mathbf{g}_t(\mathbf{x}_{0:t})\right) < \infty$ then

$$\sqrt{N}\left(\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} - \mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))\right) \quad \underset{N\to\infty}{\Longrightarrow} \quad \mathcal{N}\left(0, var_{p(\cdot|\mathbf{y}_{1:t})}\left(\mathbf{g}_t(\mathbf{x}_{0:t})\right)\right) \tag{A.39}$$

where $\underset{N\to\infty}{\Longrightarrow}$ denotes convergence in distribution.

A.4.2 Bayesian importance sampling

However, as it is often impossible to sample from the posterior density, we sample from a known, easy-to-sample, proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ and use the following substitution

$$\begin{aligned}\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\ &= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\ &= \int \mathbf{g}_t(\mathbf{x}_{0:t})\frac{\omega_t(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})}q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \qquad \text{(A.40)}\end{aligned}$$

where $\omega_t(\mathbf{x}_{0:t})$ are the un-normalized weights

$$\omega_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \qquad \text{(A.41)}$$

The unknown normalizing density $p(\mathbf{y}_{1:t})$ is removed as follows

$$\begin{aligned}\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \frac{1}{p(\mathbf{y}_{1:t})}\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\ &= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})\frac{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}d\mathbf{x}_{0:t}} \\ &= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t})\omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}}{\int \omega_t(\mathbf{x}_{0:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}} \\ &= \frac{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t})\mathbf{g}_t(\mathbf{x}_{0:t}))}{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}(\omega_t(\mathbf{x}_{0:t}))} \qquad \text{(A.42)}\end{aligned}$$

where $\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}$ means expectation over the proposal distribution $q(\cdot|\mathbf{y}_{1:t})$.

Now, the expectation is approximated by

$$\begin{aligned}\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} &= \frac{1/N\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)})\omega_t^{(i)}(\mathbf{x}_{0:t}^{(i)})}{1/N\sum_{i=1}^{N}\omega_t^{(i)}(\mathbf{x}_{0:t}^{(i)})} \\ &= \sum_{i=1}^{N}\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)})\tilde{\omega}_t(\mathbf{x}_{0:t}^{(i)}) \qquad \text{(A.43)}\end{aligned}$$

where the normalized importance weights $\tilde{w}_t^{(i)}$ are given by

$$\tilde{\omega}_t = \frac{\omega_t^{(i)}}{\sum_{j=1}^{N}\omega_t^{(i)}} \qquad \text{(A.44)}$$

Estimate (A.43) is biased as it contains a ratio of estimates. Asymptotic convergence and a central limit theorem for $\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))}$ is obtained using the following assumptions:

1. $\mathbf{x}_{0:t}^{(i)}$ is a set of i.i.d. samples drawn from the proposal distribution
2. The support of the proposal distribution includes the support of the posterior distribution
3. $\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))$ exists and is finite
4. ω_t and $\omega_t \mathbf{g}_t^2(\mathbf{x}_{0:t})$ over the posterior distribution exist and are finite

The fourth condition holds if the variance of $\mathbf{g}_t(\mathbf{x}_{0:t})$ and the importance weights are bounded ([3]), i.e. for $N \to \infty$ the posterior density can be approximated arbitrarily well by

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \tilde{\omega}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{A.45}$$

A.4.3 Sequential importance sampling

In this paper our goal is to perform filtering on the given models, i.e. to compute a sequential estimate of the posterior distribution at time t without modifying the previously simulated states $\mathbf{x}_{0:t-1}$ allowing proposal distributions of the form

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) \tag{A.46}$$

Assuming the states follow a Markov process and that the observations are conditionally independent given the states yields

$$p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0) \prod_{j=1}^{t} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \quad \text{and} \quad p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) = \prod_{j=1}^{t} p(\mathbf{y}_j|\mathbf{x}_j) \tag{A.47}$$

By substituting (A.47) into (A.41) we get a recursive estimate for the importance weights

$$\begin{aligned} \omega_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \\ &= \omega_{t-1} \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{x}_{0:t-1})p(\mathbf{x}_{0:t-1})} \frac{1}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \\ &= \omega_{t-1} \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \end{aligned} \tag{A.48}$$

Now, given a proposal distribution and a set of prior samples, we are able to sequentially sample and evaluate likelihood, transition probabilities and importance weights leading to estimates such as (A.40).

A.4.4 Optimal proposal distribution

The tricky part of particle filtering is of course to come up with an appropriate proposal distribution. It has been shown that the optimal proposal distribution minimizing the variance of the importance weights is ([10]):

Proposition 1 *The proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t})$ minimizes the variance of the importance weights conditional on $\mathbf{x}_{0:t-1}$ and $\mathbf{y}_{1:t}$.*

However, the transition prior

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{1:t}) \mathrel{\hat{=}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{A.49}$$

is the most popular choice of proposal distribution (even though it gives higher variance as it does not include the most recent observations) simply because it is easier to implement. For an additive Gaussian process noise model the transition prior simplifies to

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(f(\mathbf{x}_{t-1},0),\mathbf{Q}_{t-1}\right) \tag{A.50}$$

As illustrated in figure A.2, only a few particles will dominate and have high importance weight values by evaluation of the likelihood, if we do not use the latest available information. Thus, it is very important that we move the particles towards regions of high likelihood. And as a consequence, if the prior distribution is too wide compared to the likelihood function, the particles are more likely to be outside the region of high likelihood. On the other hand, the prior distribution should be able to include the likelihood distribution. The bottom line is once again, that the choice of proposal distribution is crucial.

A.4.5 Degeneracy

Unfortunately, in Sequential Importance Sampling (SIS) the variance of the importance weights increases stochastically over time. This is seen by expanding equation (A.48):

$$\begin{aligned}
\omega_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\
&= \frac{p(\mathbf{y}_{1:t},\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\
&= \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\
&\propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}
\end{aligned} \tag{A.51}$$

Thus, the importance weights are proportional to the *importance ratio*, which variance increases over time ([10]).

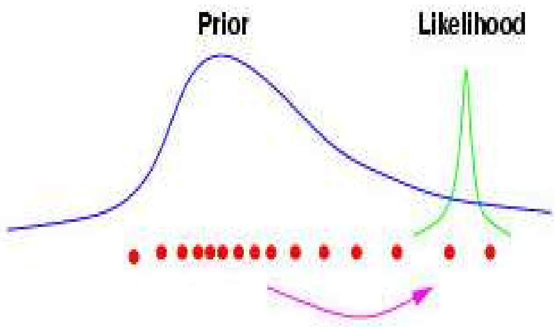


Figure A.2: Using the optimal importance distribution the samples in the prior are moved towards regions of high likelihood which is important when the likelihood is peaked (i.e. low measurement error) (from [26])

Proposition 2 *The unconditional variance (that is, when the observations are regarded as random) of the importance ratio increases over time*

Optimally, we would like to sample from the posterior, i.e. we would like the proposal density to be very close to the posterior. In this case, the mean and variance are ([8])

$$\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right) = 1$$

and

$$var_{q(\cdot|\mathbf{y}_{1:t})}\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right) = \mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}\left(\left(\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} - 1\right)^2\right) = 0$$

Hence, a reasonable estimate implies a variance close to 0, i.e. if the variance increases over time the accuracy decreases.

In practice, what happens is that after a few time steps one of the normalized importance weights is close to 1 while the rest is close to 0. In other words, a lot of the samples become useless and are neglected. To avoid this phenomenon, *degeneracy*, the particles need to be resampled (selection step) to eliminate particles with low importance weights and multiply particles with high importance weights.

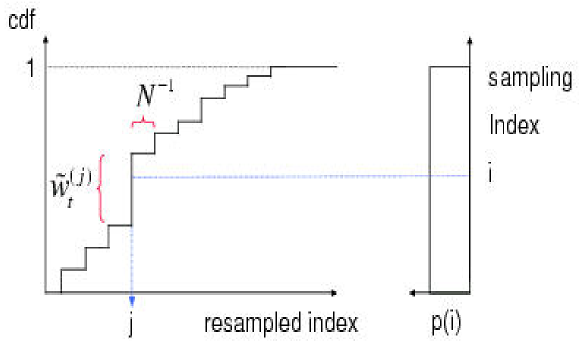


Figure A.3: Illustration of resampling principle where a random measure $\left\{\mathbf{x}_{1:t}^{(j)}, N^{-1}\right\}$ is mapped into an equally weighted random measure $\left\{\mathbf{x}_{1:t}^{(j)}, N^{-1}\right\}$ by drawing the index i from a uniform distribution (from [26])

A.4.6 Resampling

In resampling, each particle $\mathbf{x}_{0:t}^{(i)}$ is assigned a number of ”children”, say $N_i \in \mathbb{N}$, s.t. $\sum_{i=1}^{N} N_i = N$. In this paper, we have chosen to implement a number of different sampling schemes described in the following sections.

Sampling Importance Resampling (SIR) and multinomial sampling

Resampling involves mapping the Dirac random measure $\left\{\mathbf{x}_{0:t}^{(i)}, \tilde{w}_t^{(i)}\right\}$ into an equally weighted random measure $\left\{\mathbf{x}_{0:t}^{(j)}, N^{-1}\right\}$ by sampling uniformly from the discrete set $\left\{\mathbf{x}_{0:t}^{(i)}, i = 1, \ldots, N\right\}$ with probabilities $\left\{\tilde{w}_t^{(i)}; i = 1, \ldots, N\right\}$, see [15]. Figure A.3 illustrates this principle. Having set up the cumulative distribution of the discrete set, a uniformly drawn sampling index i is projected onto the distribution range and then onto the distribution domain. The new sample index j is the intersection with the domain and hence the new sample is the vector $\mathbf{x}_{0:t}^{(i)}$. At the end of the day, the larger importance weighted samples will have more replicates.

Sampling N times from the cumulative discrete distribution $\sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{x_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ corresponds to samples $(N_i; i = 1, \ldots, N)$ from a multinomial distribution with parameters N and $\tilde{w}_t^{(i)}$ with a computational complexity of $O(N)$, ([9]). The variance becomes $var(N_i) = N\tilde{w}_t^{(i)}\left(1 - \tilde{w}_t^{(i)}\right)$ as we are sampling from a multinomial distribution.

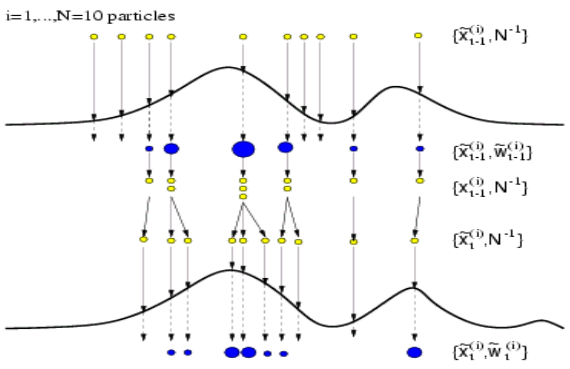

i=1,...,N=10 particles
$\{\tilde{x}^{(i)}_{t-1}, N^{-1}\}$
$\{\tilde{x}^{(i)}_{t-1}, \tilde{w}^{(i)}_{t-1}\}$
$\{x^{(i)}_{t-1}, N^{-1}\}$
$\{\tilde{x}^{(i)}_{t}, N^{-1}\}$
$\{\tilde{x}^{(i)}_{t}, \tilde{w}^{(i)}_{t}\}$

Figure A.4: Illustration of standard particle filter: The filter starts at time $t-1$ with an unweighted measure $\left\{\tilde{\boldsymbol{x}}^{(i)}_{t-1}, N^{-1}\right\}$ providing an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-2})$. For each particle the importance weights are calculated using the information at time $t-1$ giving a weighted measure $\left\{\tilde{\boldsymbol{x}}^{(i)}_{t-1}, \tilde{w}^{(i)}_{t-1}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$. Next, the resampling step selects the fittest samples to obtain the unweighted measure $\left\{\tilde{\boldsymbol{x}}^{(i)}_{t-1}, N^{-1}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$. Finally, the sampling step introduces variety giving the measure $\left\{\tilde{\boldsymbol{x}}^{(i)}_{t}, N^{-1}\right\}$ which is an approximation of $p(\boldsymbol{x}_{t}|\boldsymbol{y}_{1:t-1})$ (from [26])

Residual resampling

([24] for details). Set $\tilde{N}_i = \lfloor N\tilde{w}^{(i)}_t \rfloor$ and apply the SIR scheme to compute the remaining $\bar{N}_t = N - \sum_{i=1}^{N} \tilde{N}_i$ samples with corresponding weights $w'^{(i)}_t = \bar{N}_t^{-1}\left(\tilde{w}^{(i)}_t N - \tilde{N}_i\right)$.

The variance $\left(var(N_i) = \bar{N}_t \tilde{w}'^{(i)}_t \left(1 - \tilde{w}'^{(i)}_t\right)\right)$ is smaller than for the SIR and also computationally cheaper.

Minimum variance sampling

([5] for details). Sample a set of N points $U \in [0;1]$ with a distance of N^{-1} apart. The number of children N_i is the number of points between $\sum_{j=1}^{i-1} \tilde{w}^{(j)}_t$ and $\sum_{j=1}^{i} \tilde{w}^{(j)}_t$.

The variance of this strategy is $\left(var(N_i) = \bar{N}_t \tilde{w}'^{(i)}_t \left(1 - \bar{N}_t \tilde{w}'^{(i)}_t\right)\right)$ and the computational complexity $O(N)$.

A.4.7 Implementation

Generic Particle Filter

1. Initialization
   - For $i = 1, \dots, N$, draw the particles $x_0^{(i)}$ from the prior $p(x_0)$
2. For $t = 1, 2, \dots$

   (a) Importance sampling step
   - For $i = 1, \dots, N$, sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$ and set $\hat{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \hat{\mathbf{x}}_t^{(i)})$
   - For $i = 1, \dots, N$, evaluate the importance weights up to a normalizing constant
   $$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)}) p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{A.52}$$
   - For $i = 1, \dots, N$, normalize the importance weights
   $$\tilde{w}_t^{(i)} = w_t^{(i)} \left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \tag{A.53}$$

   (b) Selection step (resampling)
   - Multiply/suppress samples $\hat{\mathbf{x}}_{0:t}^{(i)}$ with high/low importance weights $\tilde{w}_t^{(i)}$, respectively, to obtain N random samples $\mathbf{x}_{0:t}^{(i)}$ approximately distributed according to $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$
   - For $i = 1, \dots, N$, set $w_t^{(i)} = \tilde{w}_t^{(i)} = \frac{1}{N}$

   (c) Output: A set of samples to approximate the posterior distribution as
   $$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{(x_{0:t}^{(i)})}(d\mathbf{x}_{0:t}) \tag{A.54}$$

A.4.8 Improved proposal distributions

As mentioned, the success of particle filtering is depending on how close the proposal distribution is to the posterior distribution. In this work we implement the Extended Kalman Particle Filter (PFEKF) and the Unscented Filter (PFUKF), but for completeness, we mention here a number of other methods. However, all these suffer from numerous inefficiencies ([26]) and are not included in our practical work.

Prior editing ([15]). Ad-hoc acceptance test for proposing particles in areas of high likelihood.

- After the prediction step, compute the residual error $\mathbf{e}_t = \mathbf{y}_t - \mathbf{h}_t\left(\hat{\mathbf{x}}_t^{(i)}\right)$
- If $|\mathbf{e}_t]| > K_l\sqrt{r}$, where r is the scale of the measurement error model and K_l is a constant chosen to indicate the region of non-negligible likelihood, the sample $\hat{\mathbf{x}}_t^{(i)}$ is rejected.
- Repeat until a selected number of particles is accepted

This method is too heuristic though and computationally expensive unless the rejection rate is small. Furthermore, it introduces a bias on the distribution of the particles.

Rejection methods Accept/reject method based on the principle that it is possible to sample from the optimal importance distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$ if the likelihood is bounded $p(\mathbf{y}_t|\mathbf{x}_t) < M_t$.

- Get a sample from the prior $\hat{\boldsymbol{x}} \sim p\left(\boldsymbol{x}_t \mid \mathbf{x}_{t-1}\right)$ and a uniform variable $u \sim \mathcal{U}_{[0,1]}$
- Accept the sample if $u \leq p(\mathbf{y}_t|\hat{\mathbf{x}}_t)/M_t$
- Otherwise reject and repeat until N samples are obtained

This method requires a random number of iterations at each time step which is computationally expensive in high-dimensional spaces ([9]).

Auxiliary particle filter Approximate sampling from the optimal importance distribution using an auxiliary variable k by sampling from the distribution

$$q(\mathbf{x}_t, k|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mu_t^{(k)})p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})p(\mathbf{x}_{1:t-1}^{(k)}|\mathbf{y}_{1:t-1}) \tag{A.55}$$

where $\mu_t^{(k)}, k = 1, \ldots, N$ is the mean, mode draw or another value associated with the transition prior.

- Evaluate the marginal auxiliary variable weights $g(k|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t}) \propto p(\boldsymbol{y}_t|\mu_t^{(k)})$ $p(\boldsymbol{x}_{1:t-1}^{(k)}|\boldsymbol{y}_{1:t-1})$ and use them to select $M > N$ particles from the transition prior.

- Evaluate the correction weights

$$\omega_t = \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(j)})}{p(\boldsymbol{y}_t|\mu_t^{(k_j)})} \tag{A.56}$$

  where k_j denotes the k'th ”parent” of particle j.

- Use the correction weights to perform a second selection step to acquire N particles approximately distributed according to the posterior distribution.

In comparison with the SIR filter, the auxiliary filter introduces extra variance by the additional selection step. As a consequence, the auxiliary filter generates better estimates of the posterior distribution when the likelihood is at one of the prior tails, but visa versa if the likelihood and prior coincide.

A.5 Improved Particle Filtering

In summary, generic particle filtering is depending on the following assumptions in order to perform reasonable

Monto Carlo (MC) assumption The Dirac point-mass approximation is an adequate representation of the posterior distribution

Importance Sampling (IS) assumption By sampling from a proposal distribution and applying importance sampling corrections, it is possible to sample from the posterior distribution

If this is not the case, one needs to improve the algorithm. E.g. the number of samples used in the resampling stage may eventually end up being just a few samples with high importance weights or even a single sample with a normalized importance weight close to 1. Hence, the estimate of the posterior density function is based on one sample which is clearly inadequate. A simple solution would be to increase the number of particles leading to increased computational effort. A more ’intelligent’ solution is described in the following section.

A.6 MCMC move step

([12] for details) As mentioned, we are looking for a way of introducing sample variety after the selection step without affecting the validity of the approximation. One strategy is to introduce a MCMC step of invariant distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ on each particle ([2]). If the particles are distributed according to the posterior $p(\tilde{\mathbf{x}}_{0:t}|\mathbf{y}_{1:t})$, and we apply a Markov chain transition kernel $\mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ with invariant distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ s.t. $\int \mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ $p(\tilde{\mathbf{x}}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, we still have a set of particles distributed according to the posterior. However, the particles might have been moved to areas of higher likelihood and the total variance of the current distribution with respect to the invariant distribution can only decrease ([26]).

The MCMC move step corresponds to sampling from the finite mixture distribution $N^{-1}\sum_{i=1}^{N}\mathcal{K}(\mathbf{x}_{0:t}|\tilde{\mathbf{x}}_{0:t})$ (see [14] for convergence issues). This principle can be generalized by applying MCMC steps on the product space with invariant distribution $\prod_{i=1}^{N} p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, i.e. to the entire population of particles. However, in this work we only consider the former and simpler case. For the standard particle filter we sample from the transition prior and accept according to a Metropolis-Hastings (MH) step

Smoothing MH step

- Sample $u \sim U_{[0,1]}$
- Sample the proposal candidate $\mathbf{x}_t^{*(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
- If $u \leq min\left\{1, \frac{p(\mathbf{y}_t|\mathbf{x}_t^{*(i)})}{p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(i)})}\right\}$
  - then accept move
$$\mathbf{x}_{0:t}^{(i)} = (\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t^{*(i)}) \tag{A.57}$$
  - else reject move
$$\mathbf{x}_{0:t}^{(i)} = \tilde{\mathbf{x}}_{0:t}^{(i)} \tag{A.58}$$

  End If

More complex proposals exist, s.a. mixtures of MH steps to ensure an efficient exploration of the sample space ([18]) and reversible jump MCMC steps (see [16]) to allow particles to move from one subspace to other subspaces of, possible, different dimension ([2]).

A.7 Extended Kalman Particle Filter

One way of generating proposal distributions that are more accurate in their approximation of the optimal importance distribution is *local linearization*. This method incorporates the most current observation with the optimal Gaussian approximation of the state ([9]), and is based on the first order Taylor series expansion of the likelihood and transition prior as described in section A.2 and a Gaussian assumption on all RV's. In this work, the EKF approximates the optimal MMSE estimator of the system state by computing the conditional mean of the state given all observations. This is done recursively through time by propagating the Gaussian approximation of the posterior distribution and combining it with the new observation available at each time step. That is, the EKF computes the recursive approximation of the true posterior filtering density given by

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx p_{\mathcal{N}}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}\left(\bar{\mathbf{x}}_t, \hat{\mathbf{P}}_t\right) \tag{A.59}$$

Using the EKF in particle filtering, a separate EKF is used to generate and propagate a Gaussian proposal distribution for each particle

$$q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) \doteq \mathcal{N}(\mathbf{x}_t|\mathbf{y}_{1:t}) \qquad i = 1, \ldots, N \tag{A.60}$$

i.e. at time $t-1$ the mean and covariance of the importance distribution for each particle are computed using the EKF equations and the new observation. Thus, we need to propagate the covariance $\hat{P}^{(i)}$ and specify the EKF process and measurement noise covariances. Secondly, the i-th particle is sampled from this distribution. This filter is called the *Extended Kalman Particle Filter* and the pseudo-code is given in the following subsection.

As the EKF is an MMSE estimator, this filter leads to an improved annealed sampling algorithm in which the variance of each proposal distribution changes over time. Optimally, the search is based on a large region of the error surface and moved towards regions of lower error as time progresses. However, even though the EKF moves the prior towards the likelihood, we are still faced with the Gaussian assumption on the form of the posterior and linearization approximations. Comparing equation (A.59) with the Gaussian transition prior in equation (A.50), it is noted that the proposal distribution generated by the EKF includes the most current observation at time t. In general though, the true form of this density will *not* be Gaussian - even with Gaussian process and measurement noise - which can be shown using a Bayes rule expansion of the proposal distribution. This implies that we are left with an experimental judgement of the gain versus the loss of filter performance.

A.7.1 Implementation

Extended Kalman Particle Filter

1. Initialization
   - For $i = 1, \dots, N$, draw the particles $x_0^{(i)}$ from the prior $p(x_0)$
2. For $t = 1, 2, \dots$

   (a) Importance sampling step
   - For $i = 1, \dots, N$
     - Compute the Jacobians $\mathbf{F}_t^{(i)}, \mathbf{G}_t^{(i)}$ of the process model and $\mathbf{H}_t^{(i)}, \mathbf{U}_t^{(i)}$ of the measurement model
     - Update the particles with EKF

$$\begin{aligned}
\bar{\mathbf{x}}_{t|t-1}^{(i)} &= f(\mathbf{x}_{t-1}^{(i)}) && \text{(A.61)}\\
\mathbf{P}_{t|t-1}^{(i)} &= \mathbf{F}_t^{(i)}\mathbf{P}_{t-1}^{(i)}\mathbf{F}_t^{T(i)} + \mathbf{G}_t^{(i)}\mathbf{Q}_t\mathbf{G}_t^{T(i)} && \text{(A.62)}\\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}^{(i)}\mathbf{H}_t^{T(i)}[\mathbf{U}_t^{(i)}\mathbf{R}_t\mathbf{U}_t^{T(i)} + \mathbf{H}_t^{(i)}\mathbf{P}_{t|t-1}^{(i)}\mathbf{U}_t^{T(i)}]^{-1} && \text{(A.63)}\\
\bar{\mathbf{x}}_t^{(i)} &= \bar{\mathbf{x}}_{t|t-1}^{(i)} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}^{(i)})) && \text{(A.64)}\\
\hat{\mathbf{P}}_t^{(i)} &= \mathbf{P}_{t|t-1}^{(i)} - \mathbf{K}_t\mathbf{H}_t^{(i)}\mathbf{P}_{t|t-1}^{(i)} && \text{(A.65)}
\end{aligned}$$

     - Sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{(i)}, \hat{\mathbf{P}}_t^{(i)})$
   - For $i = 1, \dots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} \propto \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)})p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{A.66}$$

   - For $i = 1, \dots, N$, normalize the importance weights

$$\tilde{w_t}^{(i)} = w_t^{(i)}\left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \tag{A.67}$$

   (b) Selection step
   - Multiply/suppress samples $(\hat{\mathbf{x}}_{0:t}^{(i)}, \hat{\mathbf{P}}_{0:t}^{(i)})$ with high/low importance weights $\tilde{w_t}^{(i)}$, respectively, to obtain N random samples $(\tilde{\mathbf{x}}_{0:t}^{(i)}, \tilde{\mathbf{P}}_{0:t}^{(i)})$

   (c) MCMC step (optional)
   - Apply a Markov transition kernel with invariant distribution given by $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$ to obtain $(\mathbf{x}_{0:t}^{(i)}, \mathbf{P}_{0:t}^{(i)})$

   (d) Output See 'Generic Particle Filter'

EKF MH step

- Sample $u \sim U_{[0,1]}$
- Compute the Jacobians $\mathbf{F}_t^{*(i)}, \mathbf{G}_t^{*(i)}$ of the process model and $\mathbf{H}_t^{*(i)}, \mathbf{U}_t^{*(i)}$ of the measurement model
- Update the particles with EKF

$$\begin{aligned}
\bar{\mathbf{x}}_{t|t-1}^{*(i)} &= f(\tilde{\mathbf{x}}_{t-1}^{(i)}) && \text{(A.68)}\\
\mathbf{P}_{t|t-1}^{*(i)} &= \mathbf{F}_t^{*(i)}\tilde{\mathbf{P}}_{t-1}^{(i)}\mathbf{F}_t^{*T(i)} + \mathbf{G}_t^{*(i)}\mathbf{Q}_t\mathbf{G}_t^{*T(i)} && \text{(A.69)}\\
\mathbf{K}_t &= \mathbf{P}_{t|t-1}^{*(i)}\mathbf{H}_t^{*T(i)}[\mathbf{U}_t^{*(i)}\mathbf{R}_t\mathbf{U}_t^{*T(i)} + \mathbf{H}_t^{*(i)}\mathbf{P}_{t|t-1}^{*(i)}\mathbf{U}_t^{*T(i)}]^{-1} && \text{(A.70)}\\
\bar{\mathbf{x}}_t^{*(i)} &= \bar{\mathbf{x}}_{t|t-1}^{*(i)} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{h}(\bar{\mathbf{x}}_{t|t-1}^{*(i)})) && \text{(A.71)}\\
\mathbf{P}_t^{*(i)} &= \mathbf{P}_{t|t-1}^{*(i)} - \mathbf{K}_t\mathbf{H}_t^{*(i)}\mathbf{P}_{t-1}^{*(i)} && \text{(A.72)}
\end{aligned}$$

- Sample $\mathbf{x}_t^{*(i)} \sim q(\mathbf{x}_t|\tilde{\mathbf{x}}_{0:t-1}^{(i)}\mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{*(i)}, \hat{\mathbf{P}}_t^{*(i)})$
- If $u \leq min\left\{1, \frac{p(\mathbf{y}_t|\mathbf{x}_t^{*(i)})p(\mathbf{x}_t^{*(i)}|\tilde{\mathbf{x}}_{t-1}^{(i)})q(\tilde{\mathbf{x}}_t^{(i)}|\tilde{\mathbf{x}}_{0:t-1}^{(i)},\mathbf{y}_{1:t})}{p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(i)})p(\tilde{\mathbf{x}}_t^{(i)}|\tilde{\mathbf{x}}_{t-1}^{(i)})q(\mathbf{x}_t^{*(i)}|\tilde{\mathbf{x}}_{0:t-1}^{(i)},\mathbf{y}_{1:t})}\right\}$
    - then accept move

$$\mathbf{x}_{0:t}^{(i)} = (\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t^{*(i)}) \qquad \text{(A.73)}$$

$$\mathbf{P}_{0:t}^{(i)} = (\tilde{\mathbf{P}}_{0:t-1}^{(i)}, \mathbf{P}_t^{*(i)}) \qquad \text{(A.74)}$$

    - else reject move

$$\mathbf{x}_{0:t}^{(i)} = \tilde{\mathbf{x}}_{0:t}^{(i)} \qquad \text{(A.75)}$$

$$\mathbf{P}_{0:t}^{(i)} = \tilde{\mathbf{P}}_{0:t}^{(i)} \qquad \text{(A.76)}$$

End If

A.8 Unscented Filter

As described in section A.3, the UKF propagates the mean and covariance of the Gaussian approximation to the state distribution more accurately than the EKF and tends to generate better estimates of the true covariance of the state ([26]). Furthermore, the distributions generated by the UKF generally have a broader overlap with the true posterior distribution compared to the EKF estimates which is partly due to the fact that the UKF computes the posterior covariance accurately to the third order (Gaussian prior) whereas the EKF uses a first order biased approximation. The UKF also includes the latest observations, but in a more accurate way. Furthermore, the UKF is able to scale the approximation errors in higher order moments of the posterior distribution allowing heavier tailed distributons. As the sigma points in the UKF are deterministically designed to capture certain characteristics of the prior distribution, it is possible to explicitly tune the algorithm to work on distributions that have heavier tails than Gaussian distributions, e.g. Student-t distributions. All in all, the UKF is more likely to generate more accurate proposal distributions within the particle filtering framework. Using the UKF as proposal distribution generator leads to the *Unscented Filter*, [26], (in this work abbreviated PFUKF). The pseudo-code for this filter is shown below.

A.8.1 Implementation

Unscented Filter

1. Initialization

   - For $i = 1, \ldots, N$, draw particles $\mathbf{x}_0^{(i)}$ from the prior $p(\mathbf{x}_0)$ and set

$$\bar{\mathbf{x}}_0^{(i)} = E[\mathbf{x}_0^{(i)}] \tag{A.77}$$

$$\mathbf{P}_0^{(i)} = E[(\mathbf{x}_0^{(i)} - \bar{\mathbf{x}}_0^{(i)})(\mathbf{x}_0^{(i)} - \bar{\mathbf{x}}_0^{(i)})^T] \tag{A.78}$$

$$\bar{\mathbf{x}}_0^{(i)a} = E[\mathbf{x}^{(i)a}] = [(\bar{\mathbf{x}}_0^{(i)})^T \mathbf{0}\ \mathbf{0}]^T \tag{A.79}$$

$$\mathbf{P}_0^{(i)a} = E[(\mathbf{x}_0^{(i)a} - \bar{\mathbf{x}}_0^{(i)a})(\mathbf{x}_0^{(i)a} - \bar{\mathbf{x}}_0^{(i)a})^T] = \begin{pmatrix} \mathbf{P}_0^{(i)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{pmatrix} \tag{A.80}$$

2. For $t = 1, 2, \ldots$

   (a) <u>Importance sampling step</u>

   - For $i = 1, \ldots, N$
     - Update the particles with UKF
       * Calculate sigma points

$$\boldsymbol{\chi}_{t-1}^{(i)a} = \left[\begin{array}{cc} \bar{\mathbf{x}}_{t-1}^{(i)a} & \bar{\mathbf{x}}_{t-1}^{(i)a} \pm \sqrt{(n_a+\lambda)\mathbf{P}_{t-1}^{(i)a}} \end{array} \right] \quad \text{(A.81)}$$

       * Propagate particle (time update)

$$\begin{aligned}
\boldsymbol{\chi}_{t|t-1}^{(i)x} &= \mathbf{f}(\boldsymbol{\chi}_{t-1}^{(i)x}, \boldsymbol{\chi}_{t-1}^{(i)v}) && \text{(A.82)}\\
\bar{\mathbf{x}}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(m)} \chi_{j,t|t-1}^{(i)x} && \text{(A.83)}\\
\mathbf{P}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\chi_{j,t|t-1}^{(i)x} - \bar{\mathbf{x}}_{t|t-1}^{(i)x}][\chi_{j,t|t-1}^{(i)x} - \bar{\mathbf{x}}_{t|t-1}^{(i)}]^T && \text{(A.84)}\\
\gamma_{t|t-1}^{(i)} &= \mathbf{h}(\boldsymbol{\chi}_{t|t-1}^{(i)x}, \boldsymbol{\chi}_{t-1}^{(i)n}) && \text{(A.85)}\\
\bar{\mathbf{y}}_{t|t-1}^{(i)} &= \sum_{j=0}^{2n_a} W_j^{(m)} \gamma_{j,t|t-1}^{(i)} && \text{(A.86)}
\end{aligned}$$

       * Incorporate new observation (measurement update)

$$\begin{aligned}
\mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}][\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}]^T && \text{(A.87)}\\
\mathbf{P}_{\mathbf{x}_t\mathbf{y}_t} &= \sum_{j=0}^{2n_a} W_j^{(c)} [\chi_{j,t|t-1}^{(i)} - \bar{\mathbf{x}}_{t|t-1}^{(i)}][\gamma_{j,t|t-1}^{(i)} - \bar{\mathbf{y}}_{t|t-1}^{(i)}]^T && \text{(A.88)}\\
\mathbf{K}_t &= \mathbf{P}_{\mathbf{x}_t\mathbf{y}_t}\mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t}^{-1} && \text{(A.89)}\\
\bar{\mathbf{x}}_t^{(i)} &= \bar{\mathbf{x}}_{t|t-1}^{(i)} + \mathbf{K}_t(\mathbf{y}_t - \bar{\mathbf{y}}_{t|t-1}^{(i)}) && \text{(A.90)}\\
\hat{\mathbf{P}}_t^{(i)} &= \mathbf{P}_{t|t-1}^{(i)} - \mathbf{K}_t\mathbf{P}_{\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}_t}\mathbf{K}_t^T && \text{(A.91)}
\end{aligned}$$

     - Sample $\hat{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) = \mathcal{N}(\bar{\mathbf{x}}_t^{(i)}, \hat{\mathbf{P}}_t^{(i)})$
     - Set $\hat{\mathbf{x}}_{0:t}^{(i)} \triangleq (\mathbf{x}_{0:t-1}^{(i)}, \mathbf{x}_t^{(i)})$ and $\hat{\mathbf{P}}_{0:t}^{(i)} \triangleq (\mathbf{P}_{0:t-1}^{(i)}, \mathbf{P}_t^{(i)})$

- For $i = 1, \dots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} \propto \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(i)})p(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\hat{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \tag{A.92}$$

- For $i = 1, \dots, N$, normalize the importance weights

$$\tilde{w_t}^{(i)} = w_t^{(i)} \left[\sum\nolimits_{j=1}^{N} w_t^{(j)}\right]^{-1} \tag{A.93}$$

(b) Selection step

- Multiply/suppress samples $(\hat{\mathbf{x}}_{0:t}^{(i)}, \hat{\mathbf{P}}_{0:t}^{(i)})$ with high/low importance weights $\tilde{w_t}^{(i)}$, respectively, to obtain N random samples $(\tilde{\mathbf{x}}_{0:t}^{(i)}, \tilde{\mathbf{P}}_{0:t}^{(i)})$

(c) MCMC step (optional)

- Apply a Markov transition kernel with invariant distribution given by $p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})$ to obtain $(\mathbf{x}_{0:t}^{(i)}, \mathbf{P}_{0:t}^{(i)})$

(d) Output See 'Generic Particle Filter'