

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra kybernetiky

## Diplomová práce

PLZEŇ, 2023

Jan Trejbal

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd  
Akademický rok: 2022/2023

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Jan TREJBAL**  
Osobní číslo: **A21N0124P**  
Studijní program: **N3918 Aplikované vědy a informatika**  
Studijní obor: **Kybernetika a řídicí technika**  
Téma práce: **Numerická řešení bayesovských vztahů v úloze terénní navigace**  
Zadávací katedra: **Katedra kybernetiky**

## Zásady pro vypracování

1. Seznamte se s numerickými metodami pro řešení úlohy odhadu stavu (bayesovské rámce).
2. Analyzujte a porovnejte deterministické a stochastické přístupy ( tj. částicový filtr, filtr bodových mas).
3. Navrhňte řešení pro optimální umístění bodů numerických pravidel.
4. Simulačně ověřte navržené přístupy v úloze terénní navigace a zhodnoťte výsledky.

Rozsah diplomové práce: **40-50 stránek A4**  
Rozsah grafických prací:  
Forma zpracování diplomové práce: **tištěná**  
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

- [1] Matousek, Jakub. (2020). Point-mass method in state estimation and navigation. 10.13140/RG.2.2.30208.87044.  
[2] Särkkä, S. (2013). Bayesian Filtering and Smoothing (Institute of Mathematical Statistics Textbooks). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139344203  
[3] Örgüner, Umut & Skoglar, Per & Tornqvist, David & Gustafsson, Fredrik. (2010). Combined Point-Mass and Particle Filter for target tracking. IEEE Aerospace Conference Proceedings. 1 – 10. 10.1109/AERO.2010.5446678.  
[4] D. Frisch and U. D. Hanebeck, „Rejection Sampling from Arbitrary Multivariate Distributions Using Generalized Fibonacci Lattices,“ 2022 25th International Conference on Information Fusion (FUSION), 2022, pp. 1-7, doi: 10.23919/FUSION49751.2022.9841322.

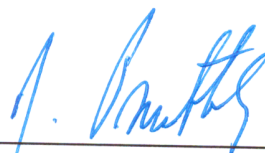
Vedoucí diplomové práce: **Doc. Ing. Jindřich Duník, Ph.D.**  
Katedra kybernetiky

Datum zadání diplomové práce: **1. října 2022**  
Termín odevzdání diplomové práce: **22. května 2023**



---

**Doc. Ing. Miloš Železný, Ph.D.**  
děkan



---


**Prof. Ing. Josef Psutka, CSc.**  
vedoucí katedry

## PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 19.5.2023



.....

## Abstrakt

Diplomová práce se zabývá úlohou odhadu stavu stochastických nelineárních dynamických systémů. Cílem této diplomové práce je určit optimální umístění bodů numerických metod řešení bayesovských rekurzivních vztahů. Numerické metody řešení bayesovských vztahů jsou využívány v úloze odhadu stavu nelineárních systému, například v úloze terénní navigace. Teoretická část této práce se nejdříve zabývá analýzou filtrů založených na dvou hlavních přístupech – filtru bodových mas založeném na deterministickém přístupu a částicovém filtru založeném na stochastickém přístupu. Následně byly tyto dva filtry porovnány a analyzovány jejich výhody a nevýhody a uvažována možnost tvorby filtru spojujícího oba přístupy za účelem vytvoření filtru se silnými stránkami obou filtrů. Nakonec byly provedeny simulace za účelem porovnání kvality odhadu a výpočetní náročnosti globálních filtrů. V těchto simulacích byly globální filtry diskutované v teoretické části porovnávány nejen proti sobě, ale i proti Kálmánovo filtru a filtrům z něj odvozených. Na základě výsledků simulací bylo navrženo optimální umístění bodů.

**Klíčová slova:** filtr bodových mas, částicový filtr, particle-point mass fusion filter, odhad stavu, nelineární filtrace, stavový model

## Abstract

The thesis deals with the problem of state estimation of stochastic nonlinear dynamical systems. The aim of this thesis is to determine the optimal location of the points of numerical methods for solving Bayesian recursive relations. Numerical methods for solving Bayesian relations are used in the problem of state estimation of nonlinear systems, for example, in the problem of terrain aided navigation. The theoretical part of this thesis first deals with the analysis of filters based on two main approaches solving Bayesian relations - the point mass filter based on the deterministic approach and the particle filter based on the stochastic approach. Subsequently, these two filters were compared and their advantages and disadvantages were analyzed and the possibility to create a filter that fuses both approaches in order to create a filter with the strengths of both filters was considered. Finally, simulations were performed to compare the estimation quality and computational complexity of the global filters. In these simulations, the global filters discussed in the theoretical part were compared not only against each other, but also against the Kalman filter and filters derived from it. Based on the results of the simulations, the optimal placement of the points was proposed.

**Keywords:** point mass filter, particle filter, particle-point mass fusion filter, state estimation, nonlinear filtering, state-space model

University of West Bohemia  
Faculty of Applied Sciences  
Department of Cybernetics

## MASTER'S THESIS

Numerical solutions to Bayesian recursive relations for  
terrain-aided navigation

PILSEN, 2023

Jan Trejbal

# Contents

<b>List of Figures</b>	<b>3</b>
<b>Used Symbols and Abbreviations</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Terrain Aided Navigation . . . . .	5
1.2 System Description . . . . .	6
1.3 Bayesian Methods . . . . .	6
1.3.1 Derivation of Equations . . . . .	6
1.3.2 Meaning of the Word Filter . . . . .	8
1.3.3 Motivation and Aim of the Thesis . . . . .	9
<b>2 Point-Mass Filter</b>	<b>10</b>
2.1 Computational Complexity of the Point Mass Filter . . . . .	12
2.2 General Point Mass Filter Algorithm . . . . .	12
2.2.1 Basic PMF Algorithm Steps . . . . .	13
2.3 Grid Adaptation/Redesign . . . . .	14
<b>3 Particle Filter</b>	<b>17</b>
3.1 Basic PF algorithm . . . . .	18
3.2 Implementation Challenges Associated with the Design and Implementation of PF . . . . .	19
3.2.1 Degeneracy Problem . . . . .	20
3.2.2 Sample Impoverishment . . . . .	20
3.2.3 Particle Filter Divergence . . . . .	20
3.2.4 Real Time Execution and Accuracy . . . . .	21
3.3 Resampling . . . . .	22
3.3.1 Resampling Algorithms . . . . .	23
3.3.2 Resampling Schemes . . . . .	23
3.3.3 Resampling Implementation Pseudocode . . . . .	24
3.4 General Particle Filter Algorithm with Resampling . . . . .	24
<b>4 Comparison and Fusion of PF and PMF</b>	<b>27</b>
4.1 Alternative PMF Algorithm . . . . .	28
4.2 Particle-Point Mass Fusion Filter . . . . .	30
4.3 Particle-Point Mass Fusion Filter Algorithm . . . . .	30
<b>5 Kalman Filters</b>	<b>33</b>
5.1 Linear System . . . . .	33
5.2 Kalman Filter Algorithm . . . . .	33
5.3 Kalman Filter Based Variants for Nonlinear Systems . . . . .	34
5.4 Extended Kalman Filter . . . . .	35
5.5 Unscented Kalman Filter . . . . .	36
<b>6 Simulation Setup</b>	<b>38</b>

6.1	Systems Used for Simulations . . . . .	39
6.1.1	First System Used for Simulations . . . . .	39
6.1.2	Second System Used for Simulations . . . . .	40
6.1.3	Third System Used for Simulations . . . . .	40
<b>7</b>	<b>Simulation Results</b>	<b>42</b>
7.1	Results of Simulation with the First System . . . . .	42
7.2	Results of Simulations with the Second System . . . . .	45
7.3	Results of Simulations with the Third System . . . . .	47
7.3.1	First Simulation Using the Third System . . . . .	48
7.3.2	Second Simulation Using the Third System . . . . .	49
7.3.3	Final filter comparison . . . . .	50
<b>8</b>	<b>Conclusion</b>	<b>51</b>



# List of Figures

2.1	Illustration of the coverage of the one-dimensional state space by PMF grid. . . . .	10
2.2	Comparison of approaches to grid redesign . . . . .	16
3.1	Illustration of the coverage of the one-dimensional state space by particles . . . . .	17
4.1	Comparison of state space coverage by supports (grid points and particles) for different filters. . . . .	28
6.1	Terrain map used in the third system . . . . .	41
7.1	Simulation result - RMSE during the trajectory of the first system . .	43
7.2	Simulation result - SD during the trajectory of the first system . . . .	43
7.3	Simulation result - Covariance during the trajectory of the first system	44
7.4	Simulation result - Inaccuracy during the trajectory of the first system	44
7.5	Example of the trajectory used in the simulation with the second system	45
7.6	Simulation result - RMSE during the trajectory of the second system	46
7.7	Simulation result - SD during the trajectory of the second system . .	46
7.8	Example of the trajectory used in the simulation with the third system	47
7.9	Simulation result - RMSE during the trajectory of the third system . .	48
7.10	Simulation result - SD during the trajectory of the third system . . .	48
7.11	Simulation result - RMSE during the trajectory of the third system .	49
7.12	Simulation result - SD during the trajectory of the third system . . .	50

# Used Symbols and Abbreviations

$N(\bar{x}, P)$  normal distribution with mean  $\bar{x}$  and covariance  $P$ .

**EKF** extended Kalman filter.

**KF** Kalman filter.

**PDF** probability density function.

**PF** particle filter.

**PMF** point mass filter.

**PPFF** particle-point mass fusion filter.

**UKF** unscented Kalman filter.

# 1. Introduction

Accurate, reliable and robust navigation is key to the safe operation of both civil and military aviation. Knowing the exact state of the aircraft is crucial especially at low altitudes, i.e. during take-off and landing, to ensure sufficient clearance from obstacles and terrain. In addition to a reliable estimate of the aircraft state, the navigation system is required to provide information on the accuracy of this estimate.

The simplest method of navigation is the use of a global navigation satellite system (GNSS), such as the Global Positioning System (GPS). GPS provides a continuous estimate of position and speed anywhere on the Earth with high accuracy and low cost. Due to weak signal strength, GPS is vulnerable to both accidental and deliberate jamming (blocking or interfering with GPS signals) and spoofing (fooling the receiver with a false signal, which can lead to the receiver providing a false location estimate) [19].

Other important method of navigation used in aviation is radio navigation. Radio navigation uses one or more radio beacons whose position is known. These beacons transmit electromagnetic signals which can be used to determine the position of the aircraft relative to these beacons. The disadvantage is that these signals can also be jammed, or spoofed [2].

Another possible approach to navigation is to use a known initial state( position, orientation and velocity) and track the current position and orientation using continuous series of measurements from accelerometers and gyroscopes. This approach is used in Inertial navigation. An inertial navigation system abbreviated as INS is used to continuously calculate the current position along the trajectory by integrating signals from accelerometers and gyroscopes. Inertial navigation is used in a wide range of applications including the navigation of aircraft, tactical and strategic missiles, spacecraft, submarines and ships [20].

Inertial navigation is theoretically independent of external inputs. However, unfortunately, dead reckoning error accumulates over time. INS therefore require continuous alignment. The usual approach is therefore to use a combination of dead-reckoning navigation using INS and fix position updates. A simple solution is to use GPS for this alignment, but this results in a loss of independence from external inputs. If we want to avoid the use of external inputs for INS alignment, it is possible to use TAN(Terrain Aided Navigation) [20, 19].

## 1.1 Terrain Aided Navigation

The main idea of TAN is to measure the height of the terrain beneath the aircraft and determine the position of the aircraft by comparing it to a known reference map. The altitude of the terrain can be determined by comparing the altitude of the aircraft over mean sea-level measured with a barometric altimeter with the distance of the aircraft from the terrain below, which is measured with a radar altimeter. In order to obtain an estimate of the aircraft position using TAN, it is necessary to solve a nonlinear recursive estimation problem. Kalman filter-based

methods(local filters) for solving the nonlinear recursive estimation problem have proven to be an unsuitable solution in many cases due to the strong nonlinearity of the measurements. Global filters based on Bayesian methods/approach have been successfully tested for various TAN cases [1, 2].

## 1.2 System Description

In the theoretical part of this thesis the following discrete time state-space model of a stochastic dynamic system with additive noises was considered

$$x_{k+1} = f_k(x_k, u_k) + w_k, \quad k = 1, 2, \dots \quad (1.1)$$

$$z_k = h_k(x_k) + v_k, \quad k = 1, 2, \dots \quad (1.2)$$

The measurement and input vectors  $z_k \in \mathbb{R}^{n_z}$ ,  $u_k \in \mathbb{R}^{n_u}$  are assumed to be known and the state vector  $x_k \in \mathbb{R}^{n_x}$  is assumed to be unknown. The state and measurement functions  $f_k : \mathbb{R}^{n_x \times n_u} \rightarrow \mathbb{R}^{n_x}$ ,  $h_k : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_z}$  are considered as known. The actual realizations of the noises  $v_k$  and  $w_k$  are unknown, but their probability density functions  $p(v_k)$  and  $p(w_k)$  are assumed to be known and independent of the probability density function (hereafter abbreviated as PDF) of the initial state  $p(x_0|Z_{-1})$  [10].

$p_{v_k}(x)$  describes the probability density value corresponding to the measurement noise evaluated at point  $x$ . If the additive measurement noise is determined by a normal distribution with zero mean and covariance  $R_k$ , then  $p_{v_k}(x) = N(x, R_k)$ . In a similar way,  $p_{w_k}(x)$  describes the probability density value corresponding to the system noise evaluated at point  $x$ . If the additive system noise is determined by a normal distribution with zero mean and covariance  $Q_k$ , then  $p_{w_k}(x) = N(x, Q_k)$ .

## 1.3 Bayesian Methods

The goal of the state estimation algorithm is to estimate the state  $x_k$  at time  $k$  based on all available measurements  $Z_k = (z)_0^k$  up to time  $k$ . For this, the well-known Bayes' theorem will be used:

$$p(A|B) = \frac{p(A, B)}{p(B)} = \frac{p(B|A)p(A)}{p(B)} \quad (1.3)$$

The probabilities should also be conditioned on the known input  $u_k$ , but for the clarity of the text this conditionality will not be stated. In TAN, the state can be the position and velocity of the moving aircraft.

### 1.3.1 Derivation of Equations

The PDF  $p(x_k|Z_k)$  can be determined using Bayes' theorem as:

$$p(x_k|Z_k) = \frac{p(z_k|x_k)p(x_k|Z_{k-1})}{p(z_k|Z_{k-1})} \quad (1.4)$$

The individual terms of Bayes' theorem can be referred to as:

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{marginal likelihood}} \quad (1.5)$$

- **posterior** =  $p(x_k|Z_k)$  is a conditional probability distribution expressing what states are likely after incorporating the current measurement based on prior knowledge of the state.
- **likelihood** =  $p(z_k|x_k)$  describes the probability that the actual measurement (realization) was measured under the condition that the system is in the state  $x_k$ . Likelihood is based on the measurement equation (1.2)
- **prior** =  $p(x_k|Z_{k-1})$  describes previous knowledge of the state  $x_k$ . based on past measurements - prediction.
- **marginal likelihood** =  $p(z_k|Z_{k-1})$  corresponds to the probability of generating a received measurement from the prior.

By combining the system model with the equation (1.4), the following equations can be derived

$$p(x_k|Z_k) = \alpha_k^{-1} p_{v_k}(z_k - h(x_k)) p(x_k|Z_{k-1}) \quad (1.6)$$

$$\alpha_k = \int p_{v_k}(z_k - h(x_k)) p(x_k|Z_{k-1}) dx_k \quad (1.7)$$

Where  $\alpha_k$  is the normalisation constant

The PDF  $p(x_k|Z_{k-1})$  is a one-step prediction that can be determined using the Chapman-Kolmogorov equation:

$$p(x_k|Z_{k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|Z_{k-1}) dx_{k-1} \quad (1.8)$$

Where  $p(x_k|x_{k-1})$  can be determined using known state update equation (1.1)

The density update between two moments can be derived by combining the system model with the equation (1.8). The resulting density update equations are

$$p(x_{k+1}|Z_k) = \int p_{w_k}(x_{k+1} - f_k(x_k, u_k)) p(x_k|Z_k) dx_k \quad (1.9)$$

These equations determine the recursive solution for calculating the state estimate.

After initialization  $p(x_0|Z_{-1}) = p(x_0)$  the conditional probability density can be calculated by recursion:

- 1. step: **Filtering/Measurement update**

$$p(x_k|Z_k) = \alpha_k^{-1} p_{v_k}(z_k - h(x_k)) p(x_k|Z_{k-1}) \quad (1.10)$$

Normalisation constant  $\alpha_k$  is calculated as:

$$\alpha_k = \int p_{v_k}(z_k - h(x_k)) p(x_k|Z_{k-1}) dx_k \quad (1.11)$$

- 2. step: **Prediction/Time update**

$$p(x_{k+1}|Z_k) = \int p_{w_k}(x_{k+1} - f_k(x_k, u_k)) p(x_k|Z_k) dx_k \quad (1.12)$$

Bayesian recursive relations provide estimates in the form of probability densities. If we want an estimate in the form of a point estimate and covariance it can be easily calculated. Point estimate at time  $k$  is determined as:

$$\hat{x}_k = \int x_k p(x_k | Z_k) dx_k \quad (1.13)$$

If we assume unbiased estimates, the covariance at time  $k$  can be determined:

$$C_k = \int (x_k - \hat{x}_k)(x_k - \hat{x}_k)^T p(x_k | Z_k) dx_k \quad (1.14)$$

This algorithm requires solving several integrals for each time  $k$ . For a general non-linear system there is no analytical solution of integrals in equations (1.11), (1.12), (1.13) and (1.14). There are various ways to approximate the probabilities so that these integrals can be solved [2].

If the system is linear with white Gaussian noises, then the Kalman filter is the optimal estimator.

The Kalman filter can only be used for a linear system. For a non-linear system it is necessary to use one of the local filters based on the Kalman filter. The performance of local filters based on the Kalman filter - the extended Kalman filter and the unscented Kalman filter, etc. is limited because they work only with the first two moments of the conditional PDF - the conditional mean and covariance, which makes them computationally inexpensive. The extended Kalman filter uses a local approximation of nonlinear functions, which can easily lead to divergence. This makes it questionable to how wide range of systems it can be applied to. The unscented Kalman filter can be applied to a wide range of systems if the probability density is reasonable - it can be reasonably represented by the first two moments [18, 17, 14].

Global filters, in contrast, make no assumptions about the form of the probability density and approximate the conditional probability density, which significantly reduces the probability of divergence at the cost of higher computational demands. In the case of strongly nonlinear systems, the estimation capability of local filters is severely limited, while global filters are able to provide a reasonable estimate [15, 2, 10].

### 1.3.2 Meaning of the Word Filter

The goal of a filter is usually to let something pass and do not let something else pass. For example, a water filter is designed to allow water to pass through but not microbiological contamination, unwanted metals (iron manganese), etc. A low-pass filter will allow a low-frequency signal that contains a required information to pass through and will not allow high-frequency noise to pass through. The filters that will be discussed later - point mass filter, particle filter, Kalman filter, etc. work with information input that contains noise, uncertainty and may contain erroneous measurements. These filters try to separate (let through) useful information from noise/errors (which they do not let through) from this information input and provide the correct output/estimate containing the least amount of uncertainty [14].

### 1.3.3 Motivation and Aim of the Thesis

The theory and application of the point mass filter and the particle filter are often discussed in the literature. Both of these filters have advantages and disadvantages, so neither can be considered generally better than the other. Unfortunately, the research and development of each of these filters is often done by different groups, so comparisons between them are rarely made. There are even papers in the literature discussing filters that combine the point mass filter and the particle filter. The goal of combining these filters is to create a filter that has the strengths of both of these filters.

The main objective of this thesis is to investigate and compare different approaches to approximate probabilities in the problem of global state estimation using a finite number of grid points/particles. In this thesis, the particle filter (stochastic approach), the point mass filter (deterministic approach) and their combination will be presented and analyzed. Different grid choices will be explored for the point mass filter. Finally, Monte Carlo simulations will be performed to compare the different filter variants and the results will be commented.

## 2. Point-Mass Filter

The main idea of the point-mass filter (hereafter abbreviated as PMF) is to use a probability density approximation in the form of a piece-wise constant probability density further called point-mass density - PMD.

The main idea of the point mass-filter (hereafter abbreviated as PMF) is to approximate conditional probability densities in Bayesian recursive relations by a piece-wise constant probability density, which is called the point mass density abbreviated as PMD. The probability density is evaluated in  $N$  grid points  $(\xi)_{i=1}^N$ , and is assumed to be constant around a grid point. The neighborhood of all points have usually the same size and these neighborhoods do not overlap but fully covers the whole considered part of the state space.

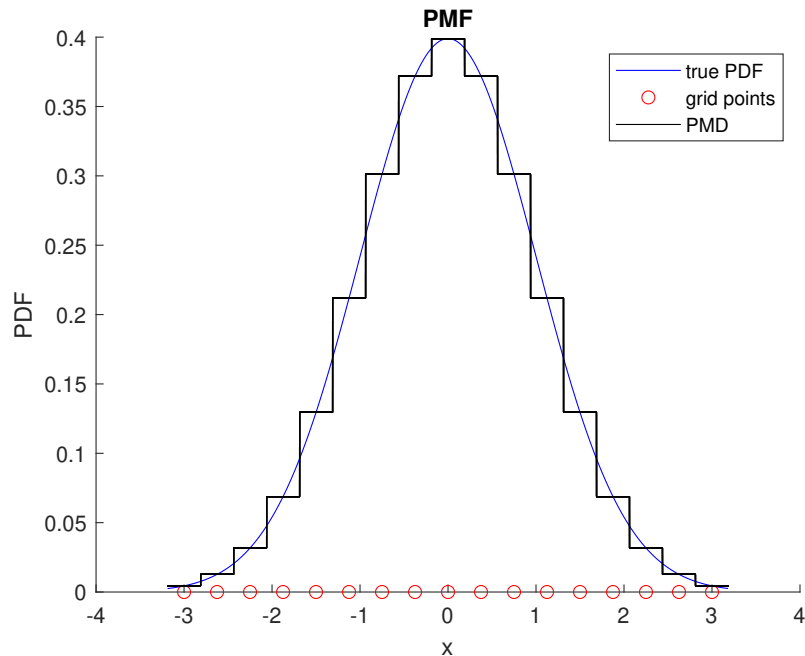


Figure 2.1: Illustration of the coverage of the one-dimensional state space by PMF grid.

In dissertation of Bergman [2] several approaches that can be used to obtain the point-mass algorithm are mentioned, for example:

- Discretize the state space and restrict the considered values of  $x_k \in \mathbb{R}^{n_x}$  to a finite number of levels
- Perform numerical approximation of integrals by Riemann sums over finite intervals
- Divide the state space into regions and express the probability that the current state is in that region.



The algorithms obtained by the different approaches are almost identical, therefore only one general PMF algorithm will be considered.

The resulting Point-Mass Approximation of probability densities can be described as:[10]

$$p(x_k|Z_m) \approx \hat{p}(x_k|Z_m; \xi_k) = \sum_{i=1}^N P_{k|m}(\xi_k^{(i)}) S\{x_k; \xi_k^{(i)}; \Delta_k\}. \quad (2.1)$$

This expression is the predictive PDF approximation for  $m = k - 1$  and the filtering PDF approximation for  $m = k$ . The individual terms in (2.1) mean:

- $\xi_k^{(i)}$  is the  $i^{th}$  grid point at time  $k$ ,
- $P_{k|m}(\xi_k^{(i)}) = c_k \tilde{P}_{k|m}(\xi_k^{(i)})$ , where  $\tilde{P}_{k|m}(\xi_k^{(i)})$  denotes  $p(\xi_k^{(i)}|Z_m)$  - value of the conditional PDF evaluated at  $i^{th}$  grid point at time  $k$ ,  $c_k = \delta_k \sum_{i=1}^N \tilde{P}_{k|m}(\xi_k^{(i)})$  denotes normalization constant, where  $\delta_k$  is a volume of neighbourhood of a each grid points, if the neighbourhood volumes of all points are equal. If the neighborhood volumes of the grid points are not the same for all points, the formula must be adjusted. The volume of the neighborhood of a point becomes dependent on the individual grid point in the case of unequal neighborhoods of individual grid points and would be a function of the point  $\delta_k(\xi_k^{(i)})$ ,
- $\Delta_k = [\Delta_k(1) \ \Delta_k(2) \ \dots \ \Delta_k(nx)]$  denotes the hyperrectangular neighborhood of the grid point  $\xi_k^{(i)}$  in which the PDF is considered constant.
- $S\{x_k; \xi_k^{(i)}; \Delta_k\}$  denotes the selection function that is defined:

$$S\{x_k; \xi_k^{(i)}; \Delta_k\} = \begin{cases} 1 & \text{if } x_k(d) \in [\xi_k(d) - 0.5\Delta_k(d), \xi_k(d) + 0.5\Delta_k(d)] \ \forall d = 1 \dots nx \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

where  $x_k(d)$  denotes  $d^{th}$  element of the vectors  $x_k(d)$ . That is, the selection function is equal to one if  $x_k$  is in the neighborhood of the grid point  $\xi_k^{(i)}$ .

- As a result:

$$\int S\{x_k; \xi_k^{(i)}\} dx_k = \Delta_k(1) \times \dots \times \Delta_k(nx) = \delta_k \quad (2.3)$$

By combining the Bayesian relations(1.10) ,(1.11) ,(1.12) ,(1.14) and (1.13) ) with the point-mass approximation (2.1) of the probability density, the following equations can be obtained [2, 10]:

$$\begin{aligned} \alpha_k &= \sum_{i=1}^N p(z_k|x_k = \xi_k^{(i)}) P_{k|k-1}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \\ &= \sum_{i=1}^N p_{v_k}(z_k - h(\xi_k^{(i)})) P_{k|k-1}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \end{aligned} \quad (2.4)$$

$$\begin{aligned} P_{k|k}(\xi_k^{(i)}) &= \alpha_k^{-1} p(z_k|x_k = \xi_k^{(i)}) P_{k|k-1}(\xi_k^{(i)}) \\ &= \alpha_k^{-1} p_{v_k}(z_k - h(\xi_k^{(i)})) P_{k|k-1}(\xi_k^{(i)}) \end{aligned} \quad (2.5)$$

Point estimate at time  $k$  is determined as:

$$\hat{x}_k = \sum_{i=1}^N \xi_k^{(i)} P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \quad (2.6)$$

And if we assume unbiased estimates, the covariance at time  $k$  can be determined:

$$C_k = \sum_{i=1}^N (\xi_k^{(i)} - \hat{x}_k)(\xi_k^{(i)} - \hat{x}_k)^T P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \quad (2.7)$$

If we consider that the grid points at time  $k+1$  may be different from the grid points at time  $k$ , we need to calculate the PDF at each point of the new grid separately. This step is very time-consuming, because the effect of the PDF of each point of the old grid on the PDF of each point of the new grid must be calculated. For each  $j$  grid point at time  $k+1$  it is necessary to perform the calculation:

$$\begin{aligned} P_{k+1|k}(\xi_{k+1}^{(j)}) &= \sum_{i=1}^N p(\xi_{k+1}^{(j)} | x_k = \xi_k^{(i)}) P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \\ &= \sum_{i=1}^N p_{w_k}(\xi_{k+1}^{(j)} - f_k(\xi_k^{(i)}, u_k)) P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \end{aligned} \quad (2.8)$$

## 2.1 Computational Complexity of the Point Mass Filter

The most computationally demanding part of the PMF is usually recalculating the PDF values between the old and new grid points after a new grid has been constructed. The computational complexity of this process increases as a quadratic function of the number of grid points. Possible ways of solving this problem are to use Rao-Blackwellization or parallel computing, see subsection 3.2.4, or to use a computationally efficient implementation of the predictive step of the PMF algorithm-convolution. Due to the quadratic growth in computational complexity, one step of the PMF algorithm with  $N$  grid points, for a reasonably large  $N$ , takes significantly longer to compute than one step of the particle filter algorithm with  $N$  particles, because the standard version of particle filter does not calculate all particles against all particles.

## 2.2 General Point Mass Filter Algorithm

The general PMF algorithm consists of several steps. The first/null step is to initialize the grid based on the initial state estimate and its covariance. This is followed by iterative steps: a measurement update - filtering step, see 1. step subsection 1.3.1, and a predictive step consisting of grid adaptation and calculation of predictive PDFs at the new grid points, see 2. step subsection 1.3.1. The presented algorithm will be based on the assumption that a new measurement is available for each time instant  $k$  - no multi-step prediction is considered.

## 2.2.1 Basic PMF Algorithm Steps

1. **Initialization** of the grid based on the known initial PDF  $p(x_0|Z_{-1})$ .

$$\hat{p}(x_0|Z_{-1}; \xi_k) = \sum_{i=1}^N P_{1|0}(\xi_1^{(i)}) S\{x_1; \xi_1^{(i)}; \Delta_1\} \quad (2.9)$$

A grid is created covering the part of the state space where the initial PDF has significant values and conditional PDF is calculated at all grid points based on known initial PDF.

Set  $k = 1$

2. **Recalculation of the conditional PDF** at all grid points based on newly available measurements

$$\hat{p}(x_k|Z_k; \xi_k) = \sum_{i=1}^N P_{k|k}(\xi_k^{(i)}) S\{x_k; \xi_k^{(i)}; \Delta_k\} \quad (2.10)$$

$$\begin{aligned} \alpha_k &= \sum_{i=1}^N p(z_k|x_k = \xi_k^{(i)}) P_{k|k-1}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \\ &= \sum_{i=1}^N p_{v_k}(z_k - h(\xi_k^{(i)})) P_{k|k-1}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \end{aligned} \quad (2.11)$$

$$\begin{aligned} P_{k|k}(\xi_k^{(i)}) &= \alpha_k^{-1} p(z_k|x_k = \xi_k^{(i)}) P_{k|k-1}(\xi_k^{(i)}) \\ &= \alpha_k^{-1} p_{v_k}(z_k - h(\xi_k^{(i)})) P_{k|k-1}(\xi_k^{(i)}) \end{aligned} \quad (2.12)$$

Based on the recalculated values of the conditional PDF, an estimate of the current state and its covariance is calculated

$$\hat{x}_k = \sum_{i=1}^N P_{k|k}(\xi_k^{(i)}) S\{x_k; \xi_k^{(i)}; \Delta_k\} = \sum_{i=1}^N \xi_k^{(i)} P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \quad (2.13)$$

$$C_k = \sum_{i=1}^N (\xi_k^{(i)} - \hat{x}_k)(\xi_k^{(i)} - \hat{x}_k)^T P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \quad (2.14)$$

3. **Grid adaptation** is based on the state equation and filtering PMD  $\hat{p}(x_k|Z_k; \xi_k)$ . A new grid is calculated whose points  $\xi_{k+1}^{(j)}$  cover the expected state at time  $k+1$ . Different approaches to grid adaptation are described in the section 2.3.
4. **Calculation of the predictive PDF** at the new grid points based on the old grid.

$$\hat{p}(x_{k+1}|Z_{k+1}; \xi_{k+1}) = \sum_{i=1}^N P_{k+1|m}(\xi_{k+1}^{(i)}) S\{x_{k+1}; \xi_{k+1}^{(i)}; \Delta_{k+1}\} \quad (2.15)$$

$$\begin{aligned}
\tilde{P}_{k+1|k}(\xi_{k+1}^{(j)}) &= \sum_{i=1}^N p(\xi_{k+1}^{(j)} | x_k = \xi_k^{(i)}) P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)}) \\
&= \sum_{i=1}^N p_{w_k}(\xi_{k+1}^{(j)} - f_k(\xi_k^{(i)}, u_k)) P_{k|k}(\xi_k^{(i)}) \delta_k(\xi_k^{(i)})
\end{aligned} \tag{2.16}$$

$\tilde{P}_{k+1|k}(\xi_{k+1}^{(j)})$  are normalized similarly to step 2:

$$P_{k+1|k}(\xi_{k+1}^{(j)}) = \frac{\tilde{P}_{k+1|k}(\xi_{k+1}^{(j)})}{\sum_{i=1}^N \tilde{P}_{k+1|k}(\xi_{k+1}^{(i)}) \delta_{k+1}(\xi_{k+1}^{(i)})} \tag{2.17}$$

5. Awaiting the arrival of a new measurement and then return to step 2.

This algorithm is based on [2, 10].

## 2.3 Grid Adaptation/Redesign

The choice of grid points is a critical part of the PMF algorithm. The grid must cover a sufficiently large region of the state space to contain a significant portion of the conditional PDF, and it is also necessary that this space is covered densely enough. In contrast, the grid is usually required to have no more than a certain number of points, due to the computational complexity and the requirement that the incoming measurement must be processed before a new one arrives. One approach [3] that seems reasonable is to calculate/attempt to estimate the first two predictive moments, the mean and covariance, and use them to determine the area to be covered by the grid. The grid should cover the area around the predicted mean and its size should depend on the predicted covariance. It seems reasonable to cover the area  $\pm l\sigma$ , ( $\sigma$  is the Standard Deviation, which corresponds to the predicted (co)variance) around the predicted mean, and the value of  $l$  can be chosen depending on how confident we want to be that the grid will cover the actual position. If the computed values of the mean and covariance were accurate and the predictive PDF had a close to normal distribution, then the  $\pm 5\sigma$  region should cover the area that should cover the actual position to more than 99.9999% [10]. The predictive covariance and mean can be estimated by Kalman filter (if the state equation is linear) or by unscented Kalman filter (if the state equation is nonlinear). The basic variant is a hyperrectangular grid with uniformly distributed points. The disadvantage of this approach is that the approximation of the real PDF by a piecewise continuous function is not accurate and the error of this approximation depends on the steepness of the approximated function. One would expect the real PDF to have significantly more steepness around the mean value-that is, the approximation error would be higher around the mean value. While the tail regions of the PDF will be much more flat- we can expect smaller approximation errors. One possible solution proposed in [10] is to use unevenly spaced points - it would make sense to make the grid denser around the middle value and less dense at the tail of the PDF.

If we compare the two approaches to grid adaptation (assuming that the number of grid points depends only on the available computing power and the required computation time - it is not user-defined) :

- **Normal approach to grid formation** - The region to be covered by the grid is specified - the region  $\pm l_1 \sigma$  around the predicted mean - the value of  $\sigma$  is tied to the predicted covariance, which together with the predicted mean is determined by a fast local filter from the filter estimates at the previous time step. This region is then covered by equidistantly spaced grid points. The only user-defined parameter is the constant  $l_1$ , which determines how large an area should be covered.
- **Density specific grid design** The region to be covered by the grid is specified - the region  $\pm m_1 \sigma$  around the predicted mean. Subsequently, a central subregion is identified in this region-this region should cover the vicinity of the mean value where a larger slope/change in PDF can be expected. This central subregion is identified in the vicinity of the mean value -  $\pm m_2 \sigma$ . The rest of the region covered by the grid that is not covered by the central subregion is the tail subregion. Next, the number of points that will cover each of the subregions must be determined. This number is determined from the number of available points using the constant  $m_3$  - the central region is covered by  $m_3 N$  points, where  $N$  is the number of available points and the tail subregion is covered by  $(1 - m_3)N$  points. Each of these subregions is covered by equally spaced points(within the region). The aggregated grid is covered by unevenly spaced points. This has to be taken into consideration in all steps of the PMF algorithm - the neighborhood of point  $\delta_k$  is no longer constant, but is a function of the individual points, which makes the neighborhood size of point  $\delta_k$  a function of the individual points. The number of user-defined parameters has increased to three - the constants  $m_1$ ,  $m_2$  and  $m_3$ .

These different approaches to grid design are described in [10, 11].

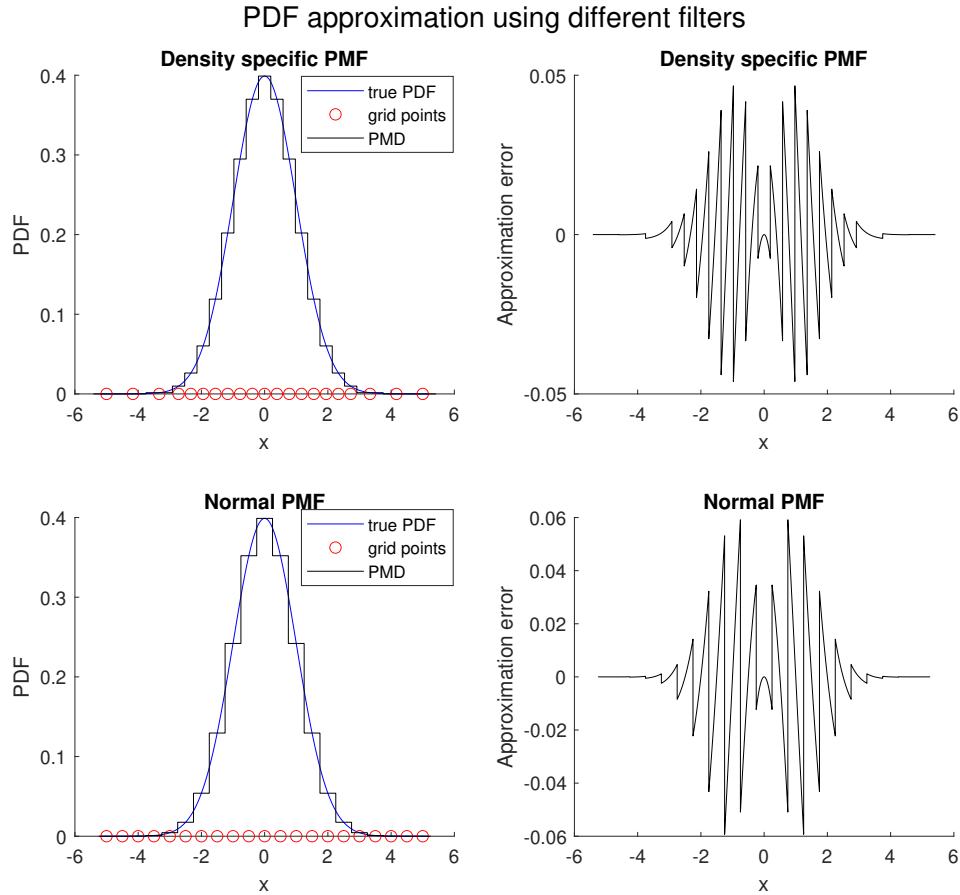


Figure 2.2: Comparison of approaches to grid redesign

In this case, if we compare the integral of the absolute approximation error, we find that the density specific grid design has a smaller overall approximation error. The approximation error of the density specific grid design is smaller in the neighborhood of the mean and larger in the tails, due to the fact that the neighborhood of the mean is covered more densely by grid points at the cost of worse coverage of the tails compared to the normal grid redesign approach.

### 3. Particle Filter

The main idea of the particle filter (hereafter abbreviated as PF) is to approximate the PDF by the sum of the weighted samples/particles. The approximation of the filtering PDF is determined by a set of  $N_s$  samples of state  $x_k$ , where each sample represents a possible realization of the state sequence and by the weights assigned to each sample [5, 16]. The empirical density used by the PF is:

$$p(x_k|Z_k) \approx \sum_{i=1}^{N_s} w_k^{(i)}(x_k^{(i)})\delta_{dir}(x_k - x_k^{(i)}). \quad (3.1)$$

where

- The individual samples at time  $k$  are denoted as:  $x_k^{(i)}$  and its corresponding weight as:  $w_k^{(i)}(x_k^{(i)})$
- Sample weights represent the relative importance(  $\sum_{i=1}^{N_s} w_k^{(i)}(x_k^{(i)}) = 1$  ) of individual samples - samples with high weights should be closer to the real state than samples with low weights.
- The Dirac delta function  $\delta_{dir}(\mu)$  is equal to zero everywhere except the point  $\mu$  and its integral is equal to one.
- Unlike in the PMF, where the probability density is approximated by a piece-wise continuous probability density that covers a significant part of the state space, in the PF the probability density is approximated by the sum of the particle weights multiplied by the Dirac pulse - the approximation of the probability density is not piece-wise continuous.

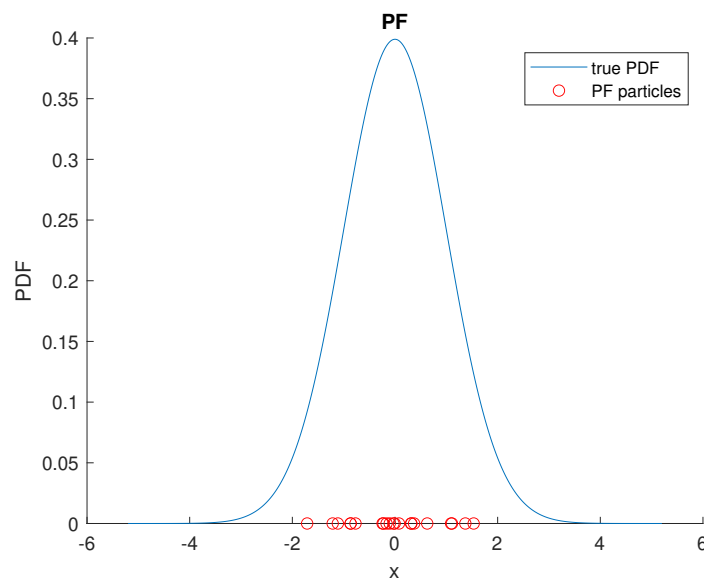


Figure 3.1: Illustration of the coverage of the one-dimensional state space by particles

The ideal option would be to take a sufficient number of samples from the posterior PDF, then assign relative weights to them and obtain an approximation of the PDF based on these weighted particles. The obvious problem is the fact that the posterior PDF is unknown, thus it is impossible to obtain samples from it. Therefore, it is necessary to sample from some other distribution. This other/alternate distribution is usually called the importance density or proposal density and will be denoted as  $q$ . The only requirement for a function to be used as an importance density is that it must be positive where the posterior is positive. The weights make it possible to use samples from one distribution to reasonably approximate another distribution, as long as the distributions cover the same space and the weights are chosen accordingly [5].

Now the question is how to compute the weights accordingly without knowledge of the posterior but with knowledge of the importance density. It has been proven that the correct way to calculate the weights is [5, 16] :

$$w_k^{(i)}(x_k^{(i)}) \propto \frac{p(x_{0:k}^{(i)}|Z_k)}{q(x_{0:k}^{(i)}|Z_k)} \quad (3.2)$$

where  $\propto$  means proportional to. This equation can be modified into a form for recursive filtering:

$$w_k^{(i)}(x_k^{(i)}) \propto w_{k-1}^{(i)}(x_{k-1}^{(i)}) \frac{p(z_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q((x_k^{(i)}|x_{k-1}^{(i)}, z_k)} \quad (3.3)$$

where  $w_{k-1}^{(i)}(x_{k-1}^{(i)})$  denotes the weight of the  $i^{th}$  sample at time  $k-1$ .

The most common choice of importance density is  $p(x_k^{(i)}|x_{k-1}^{(i)})$ . For this choice, the relation (3.3) simplifies to:

$$w_k^{(i)}(x_k^{(i)}) \propto w_{k-1}^{(i)}(x_{k-1}^{(i)})p(z_k|x_k^{(i)}) \quad (3.4)$$

### 3.1 Basic PF algorithm

The basic PF algorithm [5] (without resampling) can be summarized as follows:

1. **Initialization** of particles and their weights based on the known initial PDF  $p(x_0|Z_{-1})$ .  $N_s$  samples are selected from the initial PDF. The weights are initialized to the identical value. The weights are then normalized:

$$w_0^{(i)}(x_1^{(i)}) = 1/N_s \quad \forall i = 1 \dots N_s \quad (3.5)$$

Set  $k = 1$

2. **Weight assignment - filtering step** is based on the equation (3.3). For the choice of the importance density as  $p(x_k^{(i)}|x_{k-1}^{(i)})$  and when the additive noise of the  $v_k$  measurement is considered as  $v_k \sim N(0, R)$ , the temporary particle weights are determined as:

$$\tilde{w}_k^{(i)}(x_k^{(i)}) = w_{k-1}^{(i)}(x_{k-1}^{(i)})p(z_k|x_k^{(i)}) \quad (3.6)$$



where  $p(z_k|x_k^{(i)})$  is calculated as:

$$p(z_k|x_k^{(i)}) = N(z_k - h_k(x_k^{(i)}), R) \quad (3.7)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_k^{(i)}(x_k^{(i)}) = \frac{\tilde{w}_k^{(i)}(x_k^{(i)})}{\sum_{i=1}^{N_s} \tilde{w}_k^{(i)}(x_k^{(i)})} \quad (3.8)$$

And based on the recalculated values of the conditional PDF(weights), an estimate of the current state and its covariance is calculated:

$$\hat{x}_k = \sum_{i=1}^{N_s} x_k^{(i)} w_k^{(i)}(x_k^{(i)}) \quad (3.9)$$

$$C_k = \frac{\sum_{i=1}^{N_s} w_k^{(i)}(x_k^{(i)}) (x_k^{(i)} - \hat{x}_k)(x_k^{(i)} - \hat{x}_k)^T}{1 - \sum_{i=1}^{N_s} w_k^{(i)}(x_k^{(i)})^2} \quad (3.10)$$

3. **Particle Propagation - prediction step**  $N_s$  samples are drawn from the importance density  $q(x_{k+1}^{(i)}|x_k^{(i)}, z_k)$ . For the choice of the importance density as  $p(x_{k+1}^{(i)}|x_k^{(i)})$  and when the additive state noise  $w_k$  is considered as  $w_k \sim N(0, Q)$ , the individual particles are drawn from a normal distribution with mean corresponding to the expected particle position at time  $k + 1$  based on the position at time  $k$  and input  $u_k$  and with covariance  $Q$ :

$$x_{k+1}^{(i)} \sim N(f_k(x_k^{(i)}, u_k), Q) \quad (3.11)$$

The weights of these new/propagated particles are the same as those of the old particles:

$$w_k^{(i)}(x_{k+1}^{(i)}) = w_k^{(i)}(x_k^{(i)}) \quad \forall i = 1 \dots N_s \quad (3.12)$$

4.  $k = k + 1$  and return to step 2

## 3.2 Implementation Challenges Associated with the Design and Implementation of PF

The paper [5] proved to be a good single entry point into particle filters. In this article, five challenges associated with the design and implementation of a particle filter were presented and their solutions were reviewed and discussed.

1. Degeneracy Problem
2. Sample Impoverishment
3. Particle Filter Divergence
4. Selecting the Importance Density
5. Real Time Execution

### 3.2.1 Degeneracy Problem

When using the basic PF algorithm (without resampling), after several iterations the weight of one particle is close to one, while the weights of the other particles are negligible. This problem cannot be avoided when using the basic PF algorithm. As a result, the contribution of all but one part is very small and the computational effort invested in them is almost irrelevant for state estimation. The estimation is based on a single particle that represents only one point of the state space. Thus, we lose the ability to represent/approximate the PDF. The state estimation will be poor and the PF will diverge. Since the main reason for introducing PF was to represent probabilities using a finite number of particles, a change to the basic PF algorithm is necessary.

The solution to this problem is the use of resampling. After the third step - Weight assignment, the additional step will be added. In this step, a new set of particles is selected based on the current set of particles and their weights. The number of particles usually remains constant. New particles are usually selected stochastically from the current particles, with the probability of selecting a particle corresponding to its weight - particles with larger weights are more likely to be selected for the new particle set. The set of new particles is likely to contain several copies of particles with large weights, while particles with small weights will cease to exist. There are many different ways of how and when to perform resampling. Some of them will be presented in the section 3.3.

### 3.2.2 Sample Impoverishment

During the resampling step, which is necessary to avoid the degeneracy problem, multiple copies of particles with large weights are created. If we consider importance density as  $p(x_k^{(i)}|x_{k-1}^{(i)})$  and additive additive state noise  $w_k$  as  $w_k \sim N(0, Q)$ , then particle propagation consists of a deterministic part - converting the state of the particle at time  $k - 1$  using the state function  $f_k$  and the known input  $u_k$  to the mean value of the expected state of the particle at time  $k$ , and a stochastic part - drawing a sample.  $x_k^{(i)} \sim N(f_k(x_{k-1}^{(i)}, u_k), Q)$  Identical particles (copies created during resampling) are converted into different particles during the particle propagation step, due to state noise  $w_k$ . If the state noise  $w_k$  is too small, then these particles will be converted to almost identical particles - the converted particles will be close to each other in the state space. In a few steps, the particles will accumulate at one point in the state space. These particles will again represent only one point of the PDF, which is not desirable. If the state noise  $w_k$  is large enough, then nothing needs to be done. Otherwise, the algorithm must be modified. One possible solution is roughening, where the state noise is artificially increased. One possibility is, for example, to add artificial noise after resampling, or to modify the process model used in particle propagation.

### 3.2.3 Particle Filter Divergence

When designing a particle filter for a real-world problem, it is always necessary to consider the risk of divergence. If the PF divergence occurs, the filter is unable to provide correct estimates. Monitoring the PF divergence should be part of the

particle filter deployed on real problems. If the PF divergence occurs, the easiest solution is to reinitialize the PF. The PF divergence can occur due to various reasons:

- If the process model does not cover all possible state transitions, the actual state of the system may be in a region into which no particles are propagated. In this case, the particles are not able to accurately represent the real state.
- If the modeled measurement noise is significantly smaller than the real measurement noise, then very small weights may be assigned to particles that may represent the real state. If too little measurement noise is considered, then based on the measurements we believe that the real state is very likely to be in a significantly smaller neighborhood of the expected state (the state that would correspond to the measurement with zero additive measurement noise) than it may actually be. As a result, a particle very close to the real state can be assigned a small weight. The assignment of small weights results in the loss of most of these particles during resampling, and consequent under-coverage of the part of the state space where the real state may be located.
- The use of measurements affected by unmodeled noise error e.g. by unwanted radar reflections, jamming, etc.

### 3.2.4 Real Time Execution and Accuracy

The number of particles required to cover the state space increases exponentially with the dimension of the state space [5]. One way to deal with this curse of dimensionality is to use Rao-Blackwellization. In Rao-Blackwellization, the state is split into two sub-states, with one of the sub-states depending only on the measurement linearly and the other on the measurement non-linearly and potentially the other sub-state. The idea is that the Kalman filter can be used to solve one of these sub-states, which makes it necessary to solve a lower dimensional estimation problem using PF, resulting in lower computational cost.

Another important option is the use of parallel computing to reduce computation time. For example, it is possible to divide the state space into disjoint subspaces and perform calculations in parallel in each subspace separately [5, 8].

Interesting option is to use adaptive particle filters. Their main idea is to maintain a certain minimum quality of estimation, using the minimum number of particles required [16].

One interesting approach to adapt the PF particle number is to use the Kullback-Leibler distance. The Kullback-Leibler (KL) distance determines the distance/discrepancy between two PDFs  $p_1(x)$  and  $p_2(x)$ . Therefore, it is possible to measure the difference between the approximated filtering PDF and the sample based approximation in the form of PF. The Kullback-Leibler (KL) distance is given by:

$$D(p_1, p_2) = \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \quad (3.13)$$

This equation can be rewritten as the difference of two components:

$$D(p_1, p_2) = \underbrace{\int p_1(x) \log \frac{1}{p_2(x)} dx}_{K(p_1, p_2)} - \underbrace{\int p_1(x) \log \frac{1}{p_1(x)} dx}_{H(p_1)} \quad (3.14)$$

where  $K(p_1, p_2)$  is inaccuracy and  $H(p_1)$  is a Shannon differential entropy (SDE). Inaccuracy is given by the difference between the PDFs  $p_1(x)$  and  $p_2(x)$ . The SDE is determined by the entropy measure  $p_1(x)$ . This relation for calculating the KL distance using two components has a convenient form for comparing the empirical PDF, which is denoted as  $r_N(x)$ , (the PDF given by the weights and PF particles) with the real filtering PDF. If the empirical PDF  $r_N(x)$  is substituted for  $p_1(x)$  and the filtering density  $p(x)$  is substituted for  $p_2(x)$ , the component  $H(p_1)$  drops out of the relation. The resulting relationship for calculating the inaccuracy for PF will be:

$$\frac{\frac{1}{Ns} \sum_{i=1}^{Ns} w_k^{(i)}(x_k^{(i)}) \log \frac{1}{p(x_k^{(i)})}}{\frac{1}{Ns} \sum_{i=1}^{Ns} w_k^{(i)}(x_k^{(i)})} \quad (3.15)$$

The inaccuracy for PMF can be calculated in a similar way:

$$P_{k|k}(\xi_k^{(i)}) \delta_k \log \frac{1}{p(\xi_k^{(i)})} \quad (3.16)$$

Since both filters approximate the real probability density with the number of particles/grid points  $N$  limitingly close to infinity  $N \rightarrow \infty$  exactly, the following applies:

$$\lim_{N \rightarrow \infty} K(r_N, p) = K(p, p) = H(p) \quad (3.17)$$

It is therefore possible to measure the difference between the SDE  $H(p)$  and the calculated inaccuracy. Inaccuracy is a random variable, so it is possible to let the user choose the probability that the difference between SDE and inaccuracy is within certain limits.

If we take the value of inaccuracy as the quality of the approximation, then it is possible to measure the effect of the number of particles on the quality of the approximation PDF. One can assume that from a certain number of particles that sufficiently covers the state space, the value of inaccuracy will decrease very slowly and it is clear that it will not fall below the threshold  $H(p)$ . In testing/simulations, one must keep in mind that inaccuracy depends on the particles, so the PF can have different values of inaccuracy for the same trajectory due to stochastic nature of the particle propagation step.[5, 16]

### 3.3 Resampling

The degeneracy problem makes resampling a necessary part of any PF. The main requirement for resampling is that the new set of particles should contain particles that are a good representation of the part of the state space where the real state is likely to be located. Typically, new particles are selected from old particles, and we require that particles with larger weights have a better chance of being selected for the new set of particles, potentially in multiple copies. The number of the new particles may or may not be the same as the number of the old particles. We have two main questions / topics:

1. How to create a new set of particles - different resampling algorithms.
2. When resampling should take place - different resampling schemes.

After the resampling is complete, the weights of the new set of particles are in most cases reset and set to:  $w_k^{(i)}(x_k^{(i)}) = 1/Ns \forall i = 1 \dots Ns$ .

### 3.3.1 Resampling Algorithms

There is a large number of resampling algorithms. A thorough analysis and comparison of all possible approaches and algorithms would be a topic for a whole thesis. In [8], the following classification of resampling methods is proposed:

- Sequential implementation
  - Single distribution sampling
  - Compound-sampling
  - Special strategies
- Parallel implementation

Despite the existence of a large number of resampling algorithms, most particle filters use one of four basic algorithms: multinomial, systematic, stratified, residual. All these four algorithms fall into the group of single distribution sampling. Opinions vary as to which of these four basic algorithms is best. In general, it cannot be claimed that any of these algorithms is the best. It depends on which property of the algorithm we take as the parameter on which to base the comparison, whether computational complexity, standard deviation of the number of times each particle is replicated, number of random numbers used and so on. The main idea of multinomial, systematic, stratified resampling algorithms is to create a vector containing a cumulative sum of particle weights - at  $n^{th}$  place in the vector is the sum of particle weights from one to  $n$ . Then, a number from zero to one is repeatedly generated, and the particle whose cumulative sum of weights is greater than the generated number and whose cumulative sum of weights is the smallest among the set of particles that satisfy the condition that their cumulative sum of weights is greater than the generated number is selected to be part of the set of new particles [8, 5].

From the resampling algorithms, the Stratified resampling algorithm was selected for comparison of different filters.

### 3.3.2 Resampling Schemes

The simplest choice is to perform resampling at each time step of the algorithm. The disadvantage is that particle diversity is reduced during resampling and resampling has a notable computational cost. In [5], 3 resampling schemes are proposed:

1. Resampling at every time step
2. Every time step  $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s} (w_k^{(i)}(x_k^{(i)}))^2}$  is calculated and if  $\hat{N}_{eff} < N_s/p_1$  resampling is performed. The proposed value of threshold  $p_1$  is 4.
3. If  $1/\max(w_k^{(i)}(x_k^{(i)})) < (1/p_2)$ , then resampling is performed. The proposed value of threshold  $p_2$  is 0.005.

From these resampling schemes, the scheme 2 was selected for further testing and algorithm comparison [5].

### 3.3.3 Resampling Implementation Pseudocode

In this subsection, the pseudocodes necessary to implement the stratified resampling algorithm will be presented. For the stratified resampling algorithms, it is necessary to compute the cumulative sum of the normalized weights. The basic algorithm for computing the cumulative sum is [8]

---

#### Algorithm 1 Cumulative sum function

---

```

function CUMSUM( $\{w_k^{(i)}(x_k^{(i)})\}_{i=1}^{Ns}$ )
   $Q_k^{(1)} = Q_k^{(1)}$ 
  for  $m = 2 : Ns$  do
     $Q_k^{(m)} = Q_k^{(m-1)} + w_k^{(m)}(x_k^{(m)})$ 
  end for
  return  $\{Q_k^{(i)}\}_{i=1}^{Ns}$  ▷ Cumulative sum
end function

```

---

An algorithm to compute stratified resampling, where  $N$  is the number of particles in the new particle set - the goal is to sample  $N$  particles and  $\sim U(0, l)$  means randomly selected from a uniform distribution between 0 and 1, without zero [8]

---

#### Algorithm 2 Stratified resampling function

---

```

function STRATIFIED_RESAMPLING( $\{w_k^{(i)}(x_k^{(i)})\}_{i=1}^{Ns}$ ,  $\{x_k^{(i)}\}_{i=1}^{Ns}, N$ )
   $\{Q_k^{(i)}\}_{i=1}^{Ns} = \text{cumsum}(\{w_k^{(i)}(x_k^{(i)})\}_{i=1}^{Ns})$ 
   $m = 1$ 
  while  $n \leq N$  do
     $u_0 \sim U(0, 1/N)$ 
     $u = u_0 + n/N$ 
    while  $Q_k^{(m)} < u$  do
       $m = m + 1$ 
    end while
     $n = n + 1$ 
     $\chi_k^n = x_k^m$  ▷  $\chi_k^n$  is  $n^{th}$  particle selected to new set of particles
  end while
  return  $\{\chi_k^j\}_{j=1}^N$  ▷ New particles
end function

```

---

## 3.4 General Particle Filter Algorithm with Resampling

This presented algorithm is based on [5, 16].

1. **Initialization** of particles and their weights based on the known initial PDF  $p(x_0|Z_{-1})$ .  $N_s$  samples are selected from the initial PDF. The weights are initialized to the identical value. The weights are then normalized:

$$w_0^{(i)}(x_1^{(i)}) = 1/Ns \forall i = 1 \dots Ns \quad (3.18)$$

Set  $k = 1$

2. **Weight assignment - filtering step** is based on the equation (3.3). For the choice of the importance density as  $p(x_k^{(i)}|x_{k-1}^{(i)})$  and when the additive noise of the  $v_k$  measurement is considered as  $v_k \sim N(0, R)$ , the temporary particle weights are determined as:

$$\tilde{w}_k^{(i)}(x_k^{(i)}) = w_{k-1}^{(i)}(x_k^{(i)})p(z_k|x_k^{(i)}) \quad (3.19)$$

Where  $p(z_k|x_k^{(i)})$  is calculated as:

$$p(z_k|x_k^{(i)}) = N(z_k - h_k(x_k^{(i)}), R) \quad (3.20)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_k^{(i)}(x_k^{(i)}) = \frac{\tilde{w}_k^{(i)}(x_k^{(i)})}{\sum_{i=1}^{Ns} \tilde{w}_k^{(i)}(x_k^{(i)})} \quad (3.21)$$

And based on the recalculated values of the conditional PDF(weights), an estimate of the current state and its covariance is calculated:

$$\hat{x}_k = \sum_{i=1}^{Ns} x_k^{(i)} w_k^{(i)}(x_k^{(i)}) \quad (3.22)$$

$$C_k = \frac{\sum_{i=1}^{Ns} w_k^{(i)}(x_k^{(i)})(x_k^{(i)} - \hat{x}_k)(x_k^{(i)} - \hat{x}_k)^T}{1 - \sum_{i=1}^{Ns} w_k^{(i)}(x_k^{(i)})^2} \quad (3.23)$$

3. **Test of Particle Degeneracy** The value of  $\hat{N}_{eff}$  which indicates particle degeneracy is calculated.

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{Ns} (w_k^{(i)}(x_k^{(i)}))^2} \quad (3.24)$$

A threshold crossing test is performed. If the condition:

$$\hat{N}_{eff} < Ns/p_1 \quad (3.25)$$

is true, then resampling is performed. Else it is proceeded to the step 6.

4. **Resampling** Resampling is discussed in section 3.3. The Stratified resampling algorithm that was used in the simulations is shown in subsection 3.3.3.

5. **Particles weights reset**

$$w_k^{(i)}(x_k^{(i)}) = 1/Ns \forall i = 1 \dots Ns \quad (3.26)$$

6. **Prediction step**  $N_s$  samples are drawn from the importance density  $q(x_{k+1}^{(i)} | x_k^{(i)}, z_k)$ .

For the choice of the importance density as  $p(x_{k+1}^{(i)} | x_k^{(i)})$  and when the additive state noise  $w_k$  is considered as  $w_k \sim N(0, Q)$ , the individual particles are drawn from a normal distribution with mean corresponding to the expected particle position at time  $k + 1$  based on the position at time  $k$  and input  $u_k$  and with covariance  $Q$ :

$$x_{k+1}^{(i)} \sim N(f_k(x_k^{(i)}, u_k), Q) \quad (3.27)$$

The weights of these new/propagated particles are the same as those of the old particles:

$$w_k^{(i)}(x_{k+1}^{(i)}) = w_k^{(i)}(x_k^{(i)}) \quad \forall i = 1 \dots N_s \quad (3.28)$$

7.  $k = k + 1$  and return to step 2



## 4. Comparison and Fusion of PF and PMF

Both previously discussed global filters used for state estimation of nonlinear systems have their advantages and drawbacks. If the estimated probability density was spiky and did not have significant tails the system model was accurate and there were no sudden disturbances or faulty measurements, then PF should provide better estimates than PMF and with significantly less computational effort [9]. After convergence, the PF particles cover well those regions of the state space where the value of the PDF is significant, but the tail regions of the PDF are usually not well covered. When estimating the state in a TAN, sensor errors or sudden state changes can easily occur due to nonlinearity in altimeter measurements, etc. [9]. The PF struggles to react quickly enough to these sudden changes after convergence and the quality of its estimation can be significantly degraded. The PMF is significantly more robust against these abrupt changes compared to the PF at the cost of higher computational complexity [12, 9]. PMF typically covers a sufficiently large area of the state space with its grid, but at the cost of lower resolution. Another important difference between PF and PMF is that while the PMF is deterministic - for repetitive processing the same trajectory, its estimates remain the same. The PF is stochastic due to particle propagation - by sampling particles from the importance density.

The idea of combining PF and PMF to achieve a more robust and accurate filter was introduced in [12] and developed in [9], where the Rao-Blackwellised particle-point mass fusion filter (hereafter abbreviated as (RB)PPFF) was introduced. The original work was based on the idea that the state transition  $p(x_k|x_{k-1})$  can be decomposed into the product of two components - high tailed and low tailed

$$p(x_k|x_{k-1}) = \beta p^{ht}(x_k|x_{k-1}) + (1 - \beta)p^{lt}(x_k|x_{k-1}) \quad (4.1)$$

This separation is also transcribed into the resulting algorithm. In addition, the proposed algorithm contains the user-defined parameter, which value determines the trade-off between the variance of the weights and the randomness of the resampled density [12]. This parameter has to be tuned manually. In contrast, the (RB)PPFF does not rely on splitting the state transition  $p(x_k|x_{k-1})$  into two parts and does not contain the user-defined parameter [9]. Due to the fact that it is not easy to decompose the state transition of simulated systems into a sum of high and low tailed state transitions and manual tuning of the user-defined parameter can be tricky and its wrong value could invalidate the simulation results, the PPFF algorithm was selected for comparison in the simulation part.

Before presenting the PPFF algorithm, it is necessary to present an alternative version of the PMF algorithm. In this alternative version, the weights correspond to the grid points of the PMF, similar to the PF algorithm, and the weights are also normalized as in the PF algorithm. As a result, these grid points can then be combined with the PF particles to form total support and their corresponding total weights [12, 9].

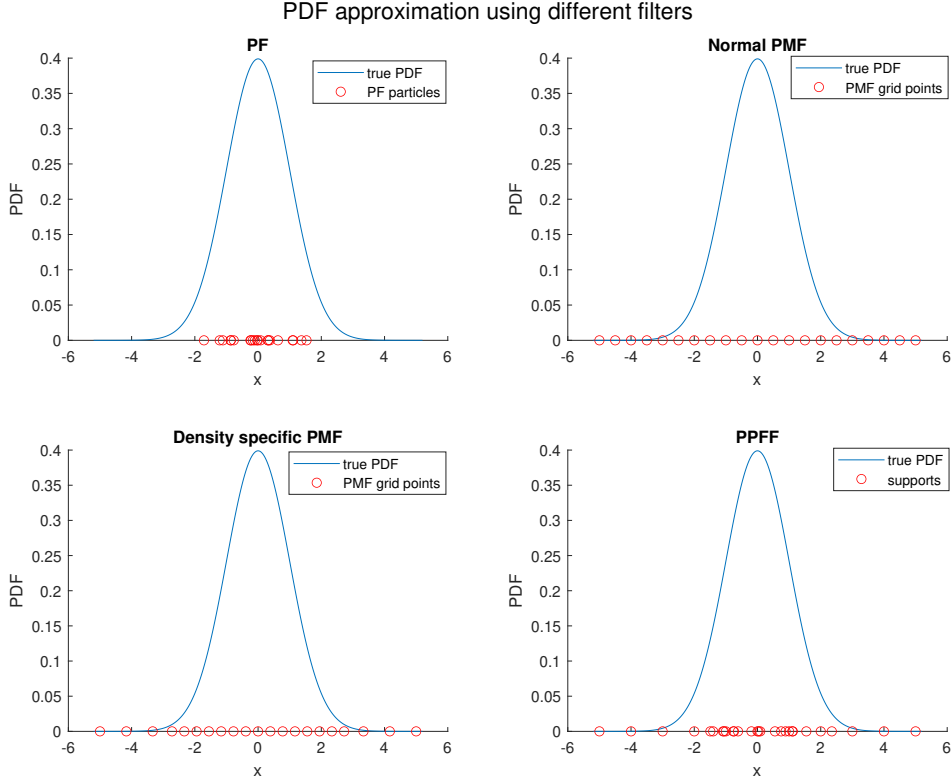


Figure 4.1: Comparison of state space coverage by supports (grid points and particles) for different filters.

We can see that the normal PMF covers the entire state space equally densely. The density specific PMF covers the center of the PDF more densely at the cost of slightly worse tail coverage, see section. The PF covers the state space around the center of the PDF well, but more or less does not cover the tails. The PPFF covers the center well due to stochastic PF particles, while also covering the tails to some extent due to the deterministic PMF grid points.

## 4.1 Alternative PMF Algorithm

For the equidistant grid, it is possible to adapt the PMF algorithm into a form that is suitable for fusing the PMF and PF algorithms.

1. **Initialization** of the grid based on the known initial PDF  $p(x_0|Z_{-1})$ .

$$\hat{p}(x_0|Z_{-1}; \xi_k) = \sum_{i=1}^N \omega_{1|0}(\xi_1^{(i)}) \xi_1^{(i)} \quad (4.2)$$

Set  $k = 1$

2. **Recalculation of the grid points weights** based on newly available measurements:

$$\tilde{w}_{k|k}^{(i)}(\xi_k^{(i)}) = w_{k|k-1}^{(i)}(\xi_k^{(i)})p(z_k|\xi_k^{(i)}) \quad (4.3)$$

Where  $p(z_k|\xi_k^{(i)})$  is calculated as:

$$p(z_k|\xi_k^{(i)}) = N(z_k - h_k(\xi_k^{(i)}), R) \quad (4.4)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_{k|k}^{(i)}(\xi_k^{(i)}) = \frac{\tilde{w}_{k|k}^{(i)}(\xi_k^{(i)})}{\sum_{i=1}^{Ns} \tilde{w}_{k|k}^{(i)}(\xi_k^{(i)})} \quad (4.5)$$

And based on the recalculated values of the conditional PDF, an estimate of the current state and its covariance is calculated

$$\hat{x}_k = \sum_{i=1}^N \xi_k^{(i)} w_{k|k}^{(i)}(\xi_k^{(i)}) \quad (4.6)$$

$$C_k = \frac{\sum_{i=1}^N w_{k|k}^{(i)}(\xi_k^{(i)}) (\xi_k^{(i)} - \hat{x}_k)(\xi_k^{(i)} - \hat{x}_k)^T}{1 - \sum_{i=1}^N w_{k|k}^{(i)}(\xi_k^{(i)})^2} \quad (4.7)$$

3. **Grid adaptation** is based on the state equation and filtering estimation PDF  $\hat{p}(x_k|Z_k; \xi_k)$ . A new grid is calculated whose points  $\xi_k + 1^{(j)}$  cover the expected state at time k+1.
4. **Calculation of predictive weights** at the new grid points based on the old grid.

$$\begin{aligned} \tilde{\omega}_{k+1|k}(\xi_{k+1}^{(j)}) &= \sum_{i=1}^N p(\xi_{k+1}^{(j)}|x_k = \xi_k^{(i)})\omega_{k|k}(\xi_k^{(i)}) \\ &= \sum_{i=1}^N p_{w_k}(\xi_{k+1}^{(j)} - f_k(\xi_k^{(i)}, u_k))\omega_{k|k}(\xi_k^{(i)}) \end{aligned} \quad (4.8)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_{k+1|k}^{(j)}(\xi_{k+1}^{(j)}) = \frac{\tilde{\omega}_{k+1|k}^{(j)}(\xi_{k+1}^{(j)})}{\sum_{j=1}^N \tilde{\omega}_{k+1|k}^{(j)}(\xi_{k+1}^{(j)})} \quad (4.9)$$

5.  $k = k + 1$  **and return to step 2**

This algorithm differs from the normal PMF algorithm in that instead of the values of the conditional PDF  $P_{k|m}(\xi_k^{(i)})$ , which are normalized so that the PMD integral is equal to one, the weights are used, which are normalized in the same way that the weights in PF - so that their sum is equal to one. This alternative form can only be used for a PMF with an equidistant grid.

## 4.2 Particle-Point Mass Fusion Filter

As already mentioned, the goal of the PPF is to create a filter that is robust to sudden measurement errors and sudden state changes, while at the same time having the best possible quality of estimation. This is achieved by combining the PF and the PMF. The idea is that these two filters will interact with each other - the PF will improve the accuracy of the PMF by using the PF particles during grid redesign, and the PMF will improve the robustness of the PF by having the PMF grid points sampled into the PF particles during resampling.

Due to the fact that the PPF includes two subfilters, it is first necessary to choose the number of supports and how they will be labeled. The total number of supports will be labeled  $N$  and it is known that the total set of supports  $N$  consists of  $N_D$  deterministic grid points of the PMF and  $N_S$  stochastic particles of the PF. Hence, that  $N = N_D + N_S$ .

The resulting algorithm can be again divided into initialization, Prediction step/-time propagation(Grid adaptation + Calculation of the predictive PDF + Particle propagation) and Filtering step/Measurement update(weights assignment and normalization for PF and PMF + resampling).

## 4.3 Particle-Point Mass Fusion Filter Algorithm

1. **Initialization** of the grid points and particles and their weights based on the known initial PDF  $p(x_0|Z_{-1})$ .

- **PMF:** A grid is created covering the part of the state space where the initial PDF has significant values and the weights of the grid points are determined based on the PDF. Temporary grid points weights are calculated as:

$$\tilde{w}_0^{(i)}(\xi_1^{(i)}) = p(x_0|Z_{-1}). \quad (4.10)$$

. The weights are then normalized:

$$w_0^{(i)}(\xi_1^{(i)}) = \frac{\tilde{w}_0^{(i)}(\xi_1^{(i)})}{\sum_{i=1}^{N_S} \tilde{w}_0^{(i)}(\xi_1^{(i)})} \quad (4.11)$$

- **PF:**  $N_S$  samples are selected from the initial PDF. The weights are initialized to the identical value

$$w_0^{(i)}(x_1^{(i)}) = 1/N_S \quad \forall i = 1 \dots N_S \quad (4.12)$$

- Set  $k = 1$

2. **Measurement update - filtering step+resampling**

- **PMF:** Recalculation of the grid points weights based on newly available measurements:

$$\tilde{w}_{k|k}^{(i)}(\xi_k^{(i)}) = w_{k|k-1}^{(i)}(\xi_k^{(i)})p(z_k|\xi_k^{(i)}) \quad (4.13)$$

where  $p(z_k|\xi_k^{(i)})$  is calculated as:

$$p(z_k|\xi_k^{(i)}) = N(z_k - h_k(\xi_k^{(i)}), R) \quad (4.14)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_{k|k}^{(i)}(\xi_k^{(i)}) = \frac{\tilde{w}_{k|k}^{(i)}(\xi_k^{(i)})}{\sum_{i=1}^{N_S} \tilde{w}_{k|k}^{(i)}(\xi_k^{(i)})} \quad (4.15)$$

- **PF** For the choice of the importance density as  $p(x_k^{(i)}|x_{k-1}^{(i)})$  and when the additive noise of the  $v_k$  measurement is considered as  $v_k \sim N(0, R)$ , the temporary particle weights are determined as:

$$\tilde{w}_k^{(i)}(x_k^{(i)}) = w_{k-1}^{(i)}(x_k^{(i)})p(z_k|x_k^{(i)}) \quad (4.16)$$

Where  $p(z_k|x_k^{(i)})$  is calculated as:

$$p(z_k|x_k^{(i)}) = N(z_k - h_k(x_k^{(i)}), R) \quad (4.17)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_k^{(i)}(x_k^{(i)}) = \frac{\tilde{w}_k^{(i)}(x_k^{(i)})}{\sum_1^{N_S} \tilde{w}_k^{(i)}(x_k^{(i)})} \quad (4.18)$$

- **Total resampling** Resampling is a necessary part of any particle filter. Total resampling is designed to sample the PMF grid points into a new set of the PF particles at a predetermined ratio.

For resampling, the PF particles and the PMF grid points are used to form a so-called total support - a joint matrix containing both PF particles and PMF grid points. Similarly, a vector containing both weights of the PF particles and the PMF grid points is created. This vector is called the total weight. To make the sum of the weights in the total weight equal to one, the weights corresponding to PMF are multiplied by  $\gamma$  and the weights corresponding to PF are multiplied by  $1 - \gamma$ . Then the normal resampling algorithm (Stratified resampling) is performed - whereby the total support is sampled based on the corresponding weights in the total weight. The value of the parameter  $\gamma$  is determined by the proportion of deterministic supports to the total number of supports - the number of PMF grid points compared to the total number of supports:

$$\gamma = \frac{N_D}{N} = \frac{N_D}{N_D + N_S} \quad (4.19)$$

Total support:

$$X_k = \begin{bmatrix} \xi_k^{(i)} & x_k^{(i)} \end{bmatrix} = \begin{bmatrix} \xi_k^{(1)} \dots \xi_k^{(N_D)} & x_k^{(1)} \dots x_k^{(N_S)} \end{bmatrix} \quad (4.20)$$

Total weight:

$$\begin{aligned} \Omega_k &= \begin{bmatrix} \gamma w_k^{(i)}(\xi_k^{(i)}) & (1 - \gamma)w_k^{(i)}(x_k^{(i)}) \end{bmatrix} \\ &= \begin{bmatrix} \gamma w_k^{(1)}(\xi_k^{(1)}) \dots \gamma w_k^{(N_D)}(\xi_k^{(N_D)}) & (1 - \gamma)w_k^{(1)}(x_k^{(1)}) \dots (1 - \gamma)w_k^{(N_S)}(x_k^{(N_S)}) \end{bmatrix} \end{aligned} \quad (4.21)$$

After the resampling is complete, the PF weights are reset:

$$w_k^{(i)}(x_k^{(i)}) = 1/N_s \forall i = 1 \dots N_s \quad (4.22)$$

### 3. Time update - prediction step

- **PMF:** As already mentioned, during grid redesign the PMF is affected by PF particles. The estimate of the system state at time  $k$  on which the new grid placement is based is determined both by the old PMF grid points and by the PF particles. The point estimate at time  $k$ , which corresponds to the mean value of total support and forms the centre of the new grid, is determined as

$$\hat{x}_k = \sum_{i=1}^{N_D} \left( \gamma \xi_k^{(i)} w_k^{(i)}(\xi_k^{(i)}) \right) + \sum_{i=1}^{N_S} \left( (1 - \gamma) x_k^{(i)} w_k^{(i)}(x_k^{(i)}) \right) \quad (4.23)$$

To calculate the covariance of the estimate, total support and total weights were again created, see equations 4.20 4.21. The following equation was used to calculate the PPF covariance

$$C_k = \sum_{i=1}^N (X_k^{(i)} - \hat{x}_k)(X_k^{(i)} - \hat{x}_k)^T \Omega_k^{(i)}(X_k^{(i)}) \quad (4.24)$$

Based on the calculated mean  $\hat{x}_k$  and covariance  $C_k$  and knowledge of the system dynamics, the mean and covariance are estimated at time  $k+1$ . Based on these, a new grid is constructed. For each point of the new grid, its predictive weight is calculated:

$$\tilde{\omega}_{k+1|k}(\xi_{k+1}^{(j)}) = \sum_{i=1}^N p(\xi_{k+1}^{(j)} | x_k = \xi_k^{(i)}) \omega_{k|k}(\xi_k^{(i)}) = \sum_{i=1}^N p_{w_k}(\xi_{k+1}^{(j)} - f_k(\xi_k^{(i)}, u_k)) \omega_{k|k}(\xi_k^{(i)}) \quad (4.25)$$

Once all the temporary weights have been calculated, they must be normalised:

$$w_{k+1|k}^{(i)}(\xi_{k+1}^{(j)}) = \frac{\tilde{\omega}_{k+1|k}^{(j)}(\xi_{k+1}^{(j)})}{\sum_{j=1}^N \tilde{\omega}_{k+1|k}^{(j)}(\xi_{k+1}^{(j)})} \quad (4.26)$$

- **PF:**  $N_s$  samples are drawn from the importance density  $q(x_{k+1}^{(i)} | x_k^{(i)}, z_k)$ . As already mentioned, for the choice of the importance density as  $p(x_{k+1}^{(i)} | x_k^{(i)})$  and when the additive state noise  $w_k$  is considered as  $w_k \sim N(0, Q)$  this corresponds to

$$x_{k+1}^{(i)} \sim N(f_k(x_k^{(i)}, u_k), Q) \quad (4.27)$$

The weights of these new/propagated particles are the same as those of the old particles:

$$w_k^{(i)}(x_{k+1}^{(i)}) = w_k^{(i)}(x_k^{(i)}) \forall i = 1 \dots N_s \quad (4.28)$$

4.  $k = k + 1$  and return to step 2

This algorithm was introduced in [9].

## 5. Kalman Filters

The Kalman filter, hereafter abbreviated as KF, is named after Rudolf E. Kálmán, who was one of its co-authors and creators. Other co-authors included his collaborator R. S. Bucy and Stanley F. Schmidt, who worked for NASA and whose group implemented the KF on a digital computer, verified by Monte Carlo simulations that it is possible to estimate uncertainty using the Riccati equation, and designed the extended Kalman filter. The Kalman filter builds on the ideas of Wiener and Kolmogorov - the Wiener filter. These earlier works had been derived in the frequency domain, while KF is in the time domain. The KF was first proposed in the late sixties and early seventies. In the 1970s it was significantly developed and even used successfully in the Apollo programme. The use of the Kalman filter back in the seventies was made possible by the fact that the KF has very little computational requirements and requires very little memory to store variables [6].

The Kalman filter can be viewed as a solution to Bayesian relations for linear system under Gaussianity assumptions, or as an incremental fusing of estimates. In other words, we actually want to fuse all available estimates/measurements in order to get the best estimate of the current state [13].

The Kalman filter was used as a benchmark in simulations using a linear system (a system with a linear state function and a linear measurement function). For a linear system with white additive noise, the Kalman filter is the optimal linear unbiased estimator (BLUE) in the sense of minimizing the MSE. For non-Gaussian additive noises, the Kalman filter is the optimal linear estimator/filter - there may be a nonlinear unbiased estimator for non-Gaussian additive noises whose MSE will be lower.

### 5.1 Linear System

In order to present the KF algorithm, it is first necessary to give the equations of the linear system to make it clear what each matrix/symbol means. This system is a specific variant of the system presented in the introduction of the thesis.

$$x_{k+1} = F_k x_k + G_k u_k + w_k, \quad k = 1, 2, \dots \quad (5.1)$$

$$z_k = H_k x_k + v_k, \quad k = 1, 2, \dots \quad (5.2)$$

Additive noise of the  $v_k$  measurement is  $v_k \sim N(0, R_k)$  and additive state is noise  $w_k \sim N(0, Q_k)$ . The mean and covariance of the initial state  $p(x_0|Z_{-1}) = N(x_{1|0}, P_{1|0})$  is known, but its exact realization is unknown.

### 5.2 Kalman Filter Algorithm

As any Bayesian estimator the Kalman filter algorithm repeats two steps - prediction and correction. During prediction, an estimate of the expected future state is made and during correction, a correction of this estimate is made based on the available

measurements. The correction rate depends on the Kalman gain multiplied by the difference between the output prediction - the expected measurement if the predictive estimate were accurate - and the actual measurement. The covariance estimate can be expected to increase during prediction-we predict the future state, which is affected by noise, about whose realization we have no information, and to decrease during correction-we use the newly obtained information. The state and covariance estimates after correction will be considered as the output of the filter.

The prior estimate of the state and covariance - the output of the prediction will be denoted  $\hat{x}_{k+1|k}$  or  $\hat{x}_{k|k-1}$  in next step and  $\hat{P}_{k+1|k}$  or  $\hat{P}_{k|k-1}$  in next step and the posterior estimate of the state and covariance - the output of the correction will be denoted:  $\hat{x}_{k|k}$  and  $\hat{P}_{k|k}$ .  $I$  is the Identity matrix  $nx \times nx$  where  $nx$  is the dimension of the state space [13, 14].

1. **Initialization** The estimation of the mean and covariance is initialized based on the initial state:  $\hat{x}_{1|0} = x_{1|0}$  and  $\hat{P}_{1|0} = P_{1|0}$  Set  $k = 1$
2. **Correction - filtering step** First, the Kalman gain is calculated

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \quad (5.3)$$

Then a correction is made to the state estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \quad (5.4)$$

Finally, the covariance estimate is recalculated: The basic equation for recalculating the covariance is

$$\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1} \quad (5.5)$$

An alternative numerically stable Joseph form has been implemented

$$\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T \quad (5.6)$$

3. **Prediction - prediction step** State and covariance prediction is performed

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k \quad (5.7)$$

$$\hat{P}_{k+1|k} = F_k \hat{P}_{k|k} F_k^T + Q_k \quad (5.8)$$

4.  $k = k + 1$  and return to step 2

## 5.3 Kalman Filter Based Variants for Nonlinear Systems

The Kalman filter algorithm works only for linear systems. If we would like to use the KF-based algorithm for nonlinear systems, we need to modify the basic KF algorithm. There is a huge number of different modifications of KF. The two main ways in which KF algorithms can be modified for use with nonlinear systems are the extended Kalman filter (hereafter abbreviated as EKF) and the unscented Kalman filter (hereafter abbreviated as UKF). The EKF uses a linear approximation of the nonlinear functions  $f$  and  $h$  in (1.1),(1.2) using a Taylor Series with neglect



of higher order terms. The UKF approximates probability distributions using a set of weighted sigma points. The sigma points, together with the weights, are chosen to preserve the mean and covariance of the probability distribution. These points are then propagated through the non-linear state equations and after propagation the weighted mean and covariance are computed [13].

## 5.4 Extended Kalman Filter

If we consider a system with a nonlinear state and measurement equation, then KF cannot be applied to that system because we do not have the state matrix  $F_k$  and the measurement matrix  $H_k$  that we need to calculate the covariance matrices. If the nonlinear state and measurement equations and their first derivatives are continuous, then it is possible to linearize them in the neighborhood of the current state estimate. If an approximation of the nonlinear functions  $f_k$  and  $h_k$  in the neighborhood of the current state estimate  $\hat{x}_k$  is used in the form first two terms of Taylor series

$$f_k(x_k, u_k) = f_k(\hat{x}_k, u_k) + J_{f,k}(\hat{x}_k)(x_k - \hat{x}_k) \quad (5.9)$$

where  $J_{f,k} = \frac{\partial f_k}{\partial x} |_{\hat{x}_k, u_k}$  is the Jacobian of  $f_k$ .

$$h_k(x_k) = h_k(\hat{x}_k, u_k) + J_{h,k}(\hat{x}_k)(x_k - \hat{x}_k) \quad (5.10)$$

where  $J_{h,k} = \frac{\partial h_k}{\partial x} |_{\hat{x}_k, u_k}$  is the Jacobian of  $h_k$ .

Higher order terms of the Taylor expansion are neglected. Using this approximation, the resulting EKF algorithm can be derived. In this algorithm, the Jacobians of the nonlinear functions  $f$  and  $h$  evaluated in the current state estimation are used instead of the state and measurement matrices [18, 7].

The resulting EKF algorithm is [18]:

1. **Initialization** The estimation of the mean and covariance is initialized based on the initial state:  $\hat{x}_{1|0} = x_{1|0}$  and  $\hat{P}_{1|0} = P_{1|0}$  Set  $k = 1$
2. **Correction - filtering step** Kalman gain is calculated:

$$K_k = P_{k|k-1} J_{h,k}(\hat{x}_{k|k-1})^T (J_{h,k}(\hat{x}_{k|k-1}) P_{k|k-1} J_{h,k}(\hat{x}_{k|k-1})^T + R_k)^{-1} \quad (5.11)$$

Then a correction is made to the state estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - h_k(\hat{x}_{k|k-1})) \quad (5.12)$$

The covariance matrix is then recalculated

$$\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1} \quad (5.13)$$

An alternative numerically stable Joseph form of calculation of the covariance matrix has been implemented

$$\hat{P}_{k|k} = (I - K_k J_{h,k}(\hat{x}_{k|k-1})) \hat{P}_{k|k-1} (I - K_k J_{h,k}(\hat{x}_{k|k-1}))^T + K_k R_k K_k^T \quad (5.14)$$

3. **Prediction - prediction step** State and covariance prediction is performed

$$\hat{x}_{k+1|k} = f_k(\hat{x}_{k|k}, u_k) \quad (5.15)$$

$$\hat{P}_{k+1|k} = J_{f,k}(\hat{x}_{k|k}, u_k) \hat{P}_{k|k} J_{f,k}(\hat{x}_{k|k}, u_k)^T + Q_k \quad (5.16)$$

4.  $k = k + 1$  **and return to step 2**

## 5.5 Unscented Kalman Filter

The main idea of the UKF is to use a set of deterministically chosen weighted points, called sigma points, to represent prior distribution. The sigma points are chosen so that their weighted mean and covariance are the same as the mean and covariance of the prior distribution. These points are then propagated through the nonlinear measurement/state functions. After propagation, the weighted sigma points are used to compute a new estimate of the mean and covariance. This process is called unscented transformation. A major advantage of the UKF over the EKF is that in the EKF, higher order terms are neglected during linearization, which can lead to poor estimation quality if the higher order terms are significant. There is also no need to deal with the analytical calculation of the Jacobian, unlike in the case of EKF, which can be challenging or even impossible. In general, the UKF should be more robust and accurate, but may require more computational overhead than the EKF. The comparison of the computational demands of EKF and UKF is not entirely clear, because the computation of the Jacobian matrix in EKF can be very demanding, which makes EKF slower than UKF under certain circumstances [7, 17, 13].

The resulting UKF algorithm can again be divided into two recursive steps - prediction and filtering: Good value for the scaling parameter  $\kappa$  is  $\kappa = 3 - nx$  for  $nx \leq 3$  and  $\kappa = 0$  for  $nx > 3$ .

1. **Initialization** The estimation of the mean and covariance is initialized based on the initial state:  $\hat{x}_{1|0} = x_{1|0}$  and  $\hat{P}_{1|0} = P_{1|0}$  Set  $k = 1$
2. **Correction - filtering step** First,  $2nx + 1$  sigma points with their weights  $X_{k|k-1} = \{\chi_{k|k-1}^i, W_i\}_{i=0}^{2nx}$  are selected/determined according to the formula

$$\chi_{k|k-1}^0 = \hat{x}_{k|k-1} \quad (5.17)$$

$$\chi_{k|k-1}^i = \hat{x}_{k|k-1} + \sqrt{nx + \kappa} s_{k|k-1}^i, i = 1, \dots, nx \quad (5.18)$$

$$\chi_{k|k-1}^j = \hat{x}_{k|k-1} + \sqrt{nx + \kappa} s_{k|k-1}^{j-nx}, j = nx + 1, \dots, 2nx \quad (5.19)$$

$$W_0 = \frac{\kappa}{nx + \kappa} \quad (5.20)$$

$$W_i = \frac{1}{2(nx + \kappa)}, i = 1, \dots, 2nx \quad (5.21)$$

where  $\kappa$  is the scaling parameter and  $s_{k|k-1}^i$  is the  $i^{th}$  column of the matrix  $S_{k|k-1}$  that satisfies the equation  $S_{k|k-1} S_{k|k-1}^T = P_{k|k-1}$ . The weighted mean and covariance of the points selected using this formula equals  $x_{k|k-1}$  and  $P_{k|k-1}$  as required. These sigma points are then propagated through the nonlinear measurement function  $h$

$$\mathcal{Z}_{k|k-1}^i = h_k(\chi_{k|k-1}^i), i = 0, \dots, 2nx \quad (5.22)$$

And based on the propagated points  $\mathcal{Z}_k^i$ , the predictive characteristics are calculated

$$\hat{z}_{k|k-1} = \sum_{i=0}^{2nx} W_i \mathcal{Z}_{k|k-1}^i \quad (5.23)$$

$$P_{k|k-1}^z = \sum_{i=0}^{2nx} W_i (\mathcal{Z}_{k|k-1}^i - \hat{z}_{k|k-1}) (\mathcal{Z}_{k|k-1}^i - \hat{z}_{k|k-1})^T + R_k \quad (5.24)$$

$$P_{k|k-1}^{xz} = \sum_{i=0}^{2nx} W_i (\chi_{k|k-1}^i - x_{k|k-1}) (\mathcal{Z}_{k|k-1}^i - \hat{z}_{k|k-1})^T \quad (5.25)$$

These predictive moments are fed into an alternative form of equations to calculate the filtering mean and covariance in the KF

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_{k|k-1}^{xz} (P_{k|k-1}^z)^{-1} (z_k - \hat{z}_{k|k-1}) \quad (5.26)$$

$$\hat{P}_{k|k} = \hat{P}_{k|k-1} P_{k|k-1}^{xz} (P_{k|k-1}^z)^{-1} (P_{k|k-1}^{xz})^T \quad (5.27)$$

**3. Prediction - prediction step** At the beginning of the predictive step, it is necessary to calculate a new set of sigma points and their weights based on the filtering mean and covariance

$$\chi_{k|k}^0 = \hat{x}_{k|k} \quad (5.28)$$

$$\chi_{k|k}^i = \hat{x}_{k|k} + \sqrt{nx + \kappa} s_{k|k}^i, i = 1, \dots, nx \quad (5.29)$$

$$\chi_{k|k}^j = \hat{x}_{k|k} + \sqrt{nx + \kappa} s_{k|k}^{j-nx}, j = nx + 1, \dots, 2nx \quad (5.30)$$

where  $\kappa$  is the scaling parameter and  $s_{k|k}^i$  is the  $i^{th}$  column of the matrix  $S_{k|k}$  that satisfies the equation  $S_{k|k} S_{k|k}^T = P_{k|k}$ . The weights are the same as in the filtering step because they are not a function of the mean and covariance, but only of the state dimension  $nx$  and  $\kappa$  and The sigma points are propagated through the nonlinear state function  $f$  and a prediction of the mean and covariance is calculated based on the propagated points

$$\chi_{k+1|k}^i = f_k(\chi_{k|k}^i, u_k), i = 0, \dots, 2nx \quad (5.31)$$

$$\hat{x}_{k+1|k} = \sum_{i=0}^{2nx} W_i \chi_{k+1|k}^i \quad (5.32)$$

$$\hat{P}_{k+1|k} = \sum_{i=0}^{2nx} W_i (\chi_{k+1|k}^i - \hat{x}_{k+1|k}) (\chi_{k+1|k}^i - \hat{x}_{k+1|k})^T + Q_k \quad (5.33)$$

**4.  $k = k + 1$  and return to step 2** [4]

## 6. Simulation Setup

A series of Monte Carlo simulations were performed in Matlab to verify the performance of the different global filters and their support selection method. Matlab was chosen for the simulations due to the fact that I have good experience with it, it is frequently used in the Department of Cybernetics, it allows easy implementation, testing and debugging of algorithms, a large amount of and open-source code is available, and it allows a good and easy graphical representation of the results. The goal of these simulations is to examine different methods of selecting supports. Different proposed and/or used support selection methods in global filters have been discussed in the theoretical part of this thesis and for further comparison the following global filters have been selected:

- Particle Filter - In tables and graphs with simulation results abbreviated as *PF*
- Point Mass Filter with adapting grid and normal approach to grid redesign - In tables and graphs with simulation results abbreviated as *PMF normal*
- Point Mass Filter with partially adapting grid and normal approach to grid redesign - In tables and graphs with simulation results abbreviated as *PMF alter*
- Point Mass Filter with adapting grid and density specific approach to grid redesign In tables and graphs with simulation results abbreviated as - In tables and graphs with simulation results abbreviated as *PMF double*
- Particle-Point Mass Fusion Filter - In tables and graphs with simulation results abbreviated as *PPFF*

PMF that is part of PPFF is PMF with adapting grid and normal approach to grid formation - *PMF normal*. Partially adapting grid means that the grid size is predefined - the number of points in each dimension is predefined and when redesigning the grid, only the area of the state space that should be covered is decided, whereas in the case of a fully adaptive grid, the number of points in each dimension is adjusted according to the proportions of predicted variance in each of those dimensions. The filters were not implemented in the Rao Blackwellized version, due to the fact that dealing with the Rao Blackwellized version of all filters would be quite challenging and beyond the scope of this manuscript. Filters based on the Kalman filter were also implemented as a benchmark:

- Kalman Filter - In tables and graphs with simulation results abbreviated as *KF*
- Extended Kalman Filter (EKF) - In tables and graphs with simulation results abbreviated as *EKF*
- Unscented Kalman Filter(UKF) - In tables and graphs with simulation results abbreviated as *UKF*

## 6.1 Systems Used for Simulations

In the simulation part of this work, 3 systems were used. All of these systems had a two-dimensional state, because it is easy to display the position of the supports (PMF grid and PF particles) in a two-dimensional state space, and it is still possible to display the position of the supports along with their weights - the combination of supports with their corresponding weights is still three-dimensional, so reasonably displayable. Another reason I chose two-dimensional systems is that PMF suffers from the curse of dimensionality, so simulations for a high-dimensional system would take an unpleasantly long time. Each of these systems allows certain properties of the filters to be verified/explored. All systems satisfy the general equations (1.1) (1.2) and the noises of all systems are white, Gaussian, have zero mean and are described by their covariance matrix -  $w_k \sim N(0, Q)$  and  $v_k \sim N(0, R)$ . The initial state of the systems was generated for each trajectory from a normal distribution with known mean and covariance -  $p(x_0|Z_{-1}) \sim N(\bar{x}_0, \bar{P}_0)$ .

### 6.1.1 First System Used for Simulations

The first system is a purely linear system. The advantage of the linear system is the possibility of using the Kalman filter as a benchmark. This allows me to verify that with a sufficiently large number of supports, the global filters provide as good an estimate as the Kalman filter. It is also easy to verify that the filters provide a consistent estimate of the covariance/standard deviation. This system is described by the following state equation and measurement equation

$$x_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_k + u_k + w_k, \quad k = 1, 2, \dots \quad (6.1)$$

$$z_k = [1 \quad 0] x_k + v_k, \quad k = 1, 2, \dots \quad (6.2)$$

Noise covariance matrices have the following values

- $Q = \begin{bmatrix} 3 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}$
- $R = [10]$

The mean and covariance of the initial state was given by

- $\bar{x}_0 = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$
- $\bar{P}_0 = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$

This system represents one dimensional motion - the first state represents position and the second state represents velocity. The velocity is only changed by external input using  $u_k$  and by the state noise  $w_k$ . If the input  $u_k$  is not used, the motion is a motion with randomly changing velocity. The position is measured - i.e. the first state. The velocity must be estimated from the measured position. In the simulation with the first system, the input was not used, so the system used in simulation was a system with random velocity.

### 6.1.2 Second System Used for Simulations

The second system has a linear state equation and a non-linear measurement equation. Second system represents the measurement of the position of an object using bearing only radar. This radar is placed at the beginning of the coordinate system - its coordinates are  $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ . The state vector contains the x and y coordinates of the object's position in the two-dimensional state space. The bearing is measured using the Matlab function `atan2`. This function provides a four-quadrant inverse tangent based on the x and y position of the object. The state is only changed by the input  $u_k$  and the state noise  $w_k$  - the object should move along a fixed trajectory determined by the sequence of inputs  $u_k$ , but due to the state noise it will only move in its proximity. The planned trajectory is known, but of course its realization is not. I chose this system to test the global filters in a semi-nonlinear environment and to make a basic comparison of their performance against the EKF and UKF. Furthermore, this system is intuitive and allows intuitive display of trajectory and state estimates using different filters. The last important thing is that there is an analytically calculable Jacobian of the nonlinear function  $h$  and this Jacobian is not too complex. The state and measurement equations of the second system are

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + u_k + w_k, \quad k = 1, 2, \dots \quad (6.3)$$

$$z_k = h(x_k)v_k, \quad k = 1, 2, \dots \quad (6.4)$$

Where the function  $h$  is `atan2`. The noise covariance matrices were chosen as follows

- $Q = \begin{bmatrix} 0.5 & 0.25 \\ 0.25 & 0.5 \end{bmatrix}$
- $R = [0.05]$

The mean and covariance of the initial state was given by

- $\bar{x}_0 = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$
- $\bar{P}_0 = \begin{bmatrix} 10 & 5 \\ 5 & 10 \end{bmatrix}$

The Jacobian of `atan2` is

$$J_h = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{-x_2}{x_1^2+x_2^2} & \frac{x_1}{x_1^2+x_2^2} \end{bmatrix} \quad (6.5)$$

### 6.1.3 Third System Used for Simulations

The third system represents the TAN in the simulation environment. The state of the system is again defined by the x and y coordinates of the object - it can be thought of as an aircraft. A key aspect of this system is the terrain map.

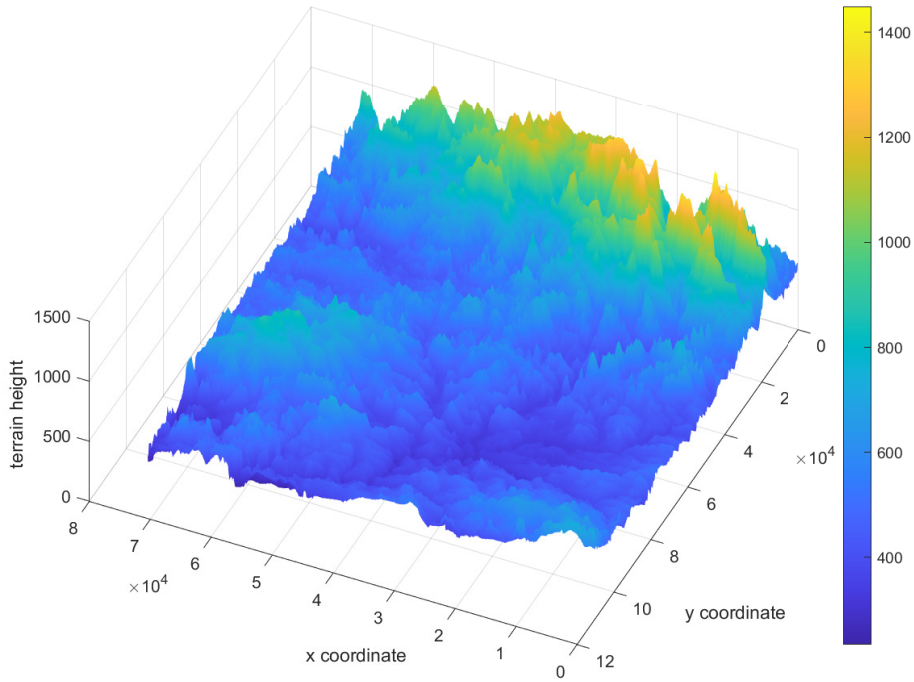


Figure 6.1: Terrain map used in the third system

The nonlinear measurement function of the third system  $h$  works by interpolating a known grid of terrain points to obtain the terrain height at a given point. This interpolation is done using the Matlab *griddedInterpolant* function. The analytical Jacobian of the measurement function cannot be calculated, so the Jacobian of the  $h$  function was calculated numerically. The error in the terrain height measurement is represented by the additive measurement noise. The state equation was chosen identically as in the case of the second system - the states are changed only by the input  $u_k$ , which defines relative movement of the object between two subsequent time instant. The planned trajectory was chosen to pass through different environments (terrain height). The third system is described by the same measurement and state equations as the second system

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + u_k + w_k, \quad k = 1, 2, \dots \quad (6.6)$$

$$z_k = h_k(x_k) + v_k, \quad k = 1, 2, \dots \quad (6.7)$$

The noise covariance matrices are

- $Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$
- $R = [64]$

I chose the center of the grid as the mean of the initial state and the covariance of the initial state was

$$\bar{P}_0 = \begin{bmatrix} 100 & 50 \\ 50 & 100 \end{bmatrix}$$

# 7. Simulation Results

In this section the results of Monte Carlo simulations will be presented. The results of the simulations with each system will be presented in individual sections. The filters were compared using the Root Mean Squared Error (abbreviated as RMSE) of their estimate, their estimated Standard Deviation (abbreviated as SD), and in some cases the covariance of their estimate and inaccuracy. RMSE /SD  $x_k(m)$  means RMSE/ SD of the  $m^{th}$  state. Furthermore, the computation time of the filter for one trajectory - runtime is compared. The number of used PMF grid points and/or PF particles will significantly affect the performance of the global filters - an insufficient number of supports reduces the quality of the estimation and may lead to divergence, but on the other hand, a higher number of supports significantly increases the time required for computation, especially in the case of PMF. The number of supports for all global filters was chosen in such a way that their runtime is similar.

The tables always show the average RMSE, SD and inaccuracy of all experiments/trajectories and all time steps. The graphs always show the average over the different trajectories, but without averaging over the individual time steps. Runtime is averaged over the different trajectories and is the sum of the times spent calculating over all trajectory steps.

All Monte Carlo simulations consisted of 100 trajectories. I consider this number to be large enough to ensure that the results are not significantly affected by randomness, and at the same time reasonably large to ensure that the simulations do not take an unreasonably long time.

## 7.1 Results of Simulation with the First System

The goal of the simulation with the first system was to verify the expectation that with a sufficiently large number of supports, the performance of the global filters should be close to the performance of the KF.



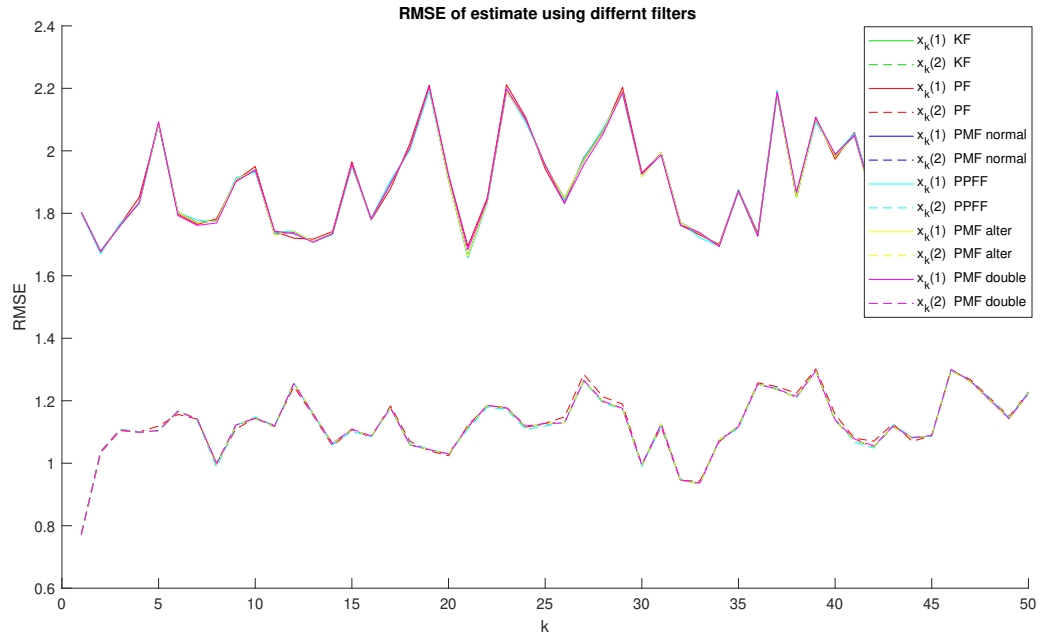


Figure 7.1: Simulation result - RMSE during the trajectory of the first system

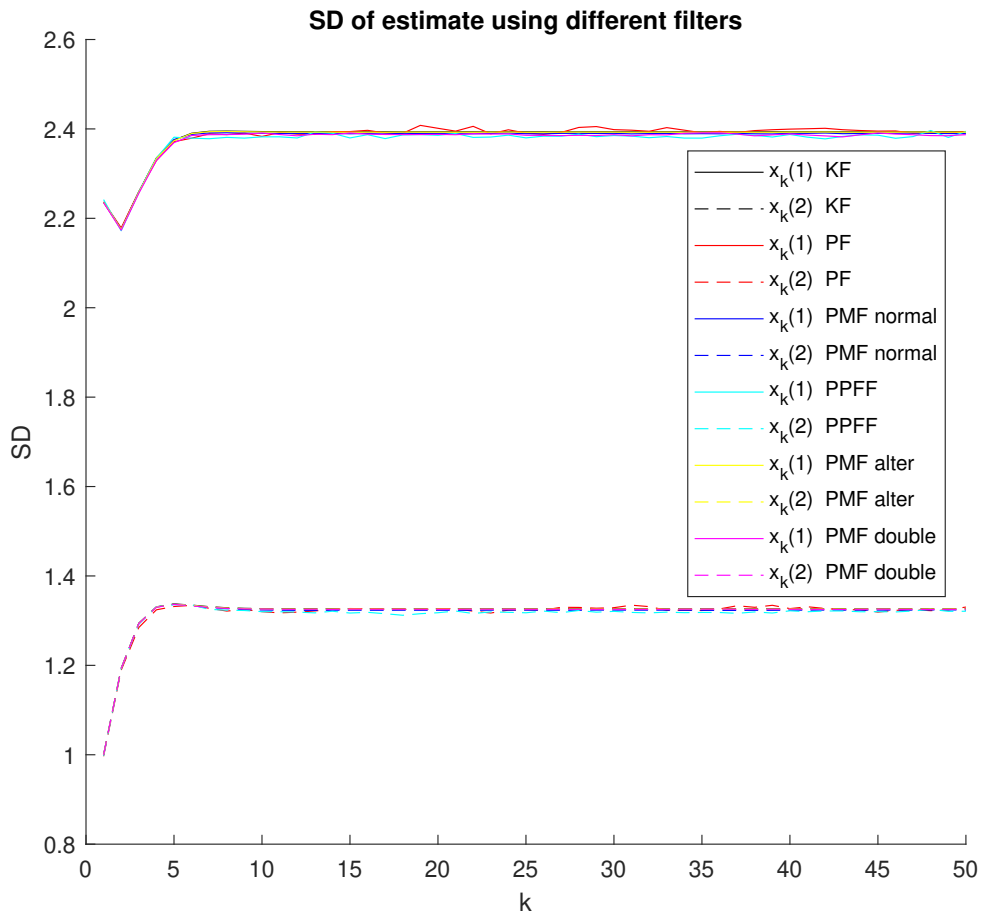


Figure 7.2: Simulation result - SD during the trajectory of the first system

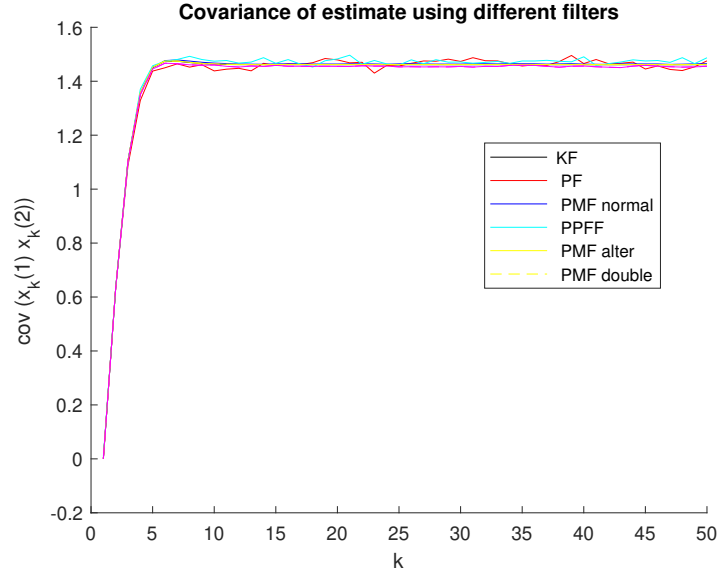


Figure 7.3: Simulation result - Covariance during the trajectory of the first system

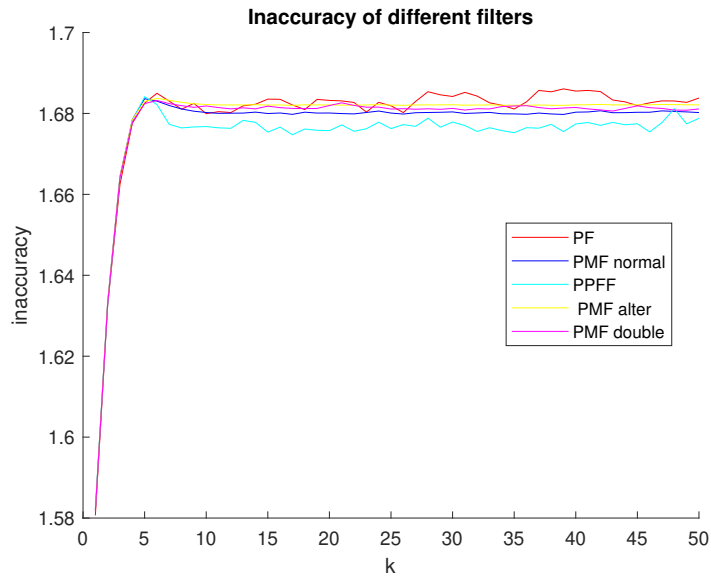


Figure 7.4: Simulation result - Inaccuracy during the trajectory of the first system

filter	RMSE $x_k(1)$	RMSE $x_k(2)$	SD $x_k(1)$	SD $x_k(2)$	runtime [s]	inaccuracy
KF	1.902	1.1227	2.3821	1.3183	$6.8515 \cdot 10^{-4}$	undefined
PF	1.9071	1.1256	2.383	1.3166	0.46193	1.6794
PMF normal	1.9021	1.1227	2.3787	1.3155	0.44751	1.6771
MF alter	1.902	1.1228	2.3814	1.3178	0.45501	1.6788
PMF double	1.903	1.1235	2.3761	1.3171	0.42604	1.6781
PPFF	1.9027	1.1218	2.3736	1.3117	0.36434	1.6741

Table 7.1: Results of simulations with the first system

Simulation results show that with a sufficiently large number of supports, the performance of all global filters matches the performance of Kalman filter quite well

for a linear system. The differences of the global filter results(RMSE and SD) from the KF results(RMSE and SD) are well below one percent. Based on these results, I deduce that with a sufficient number of supports, the quality of the estimation of all considered global filters for the linear system is the same as the quality of the estimation by KF-that is, the best possible.

The inaccuracy of all filters was very similar. PF had slightly higher inaccuracy, which corresponds to its slightly worse results. The inaccuracy of the PPF may have been biased by the fact that the prescription for calculating the inaccuracy of the PF was used to calculate it, due to the fact that there is not enough literature available about the PPF. It is questionable whether this is a completely correct approach.

## 7.2 Results of Simulations with the Second System

The aim of the simulation with the second system was to compare performance of local and global filters in a simple nonlinear estimation problem. I expect that with enough support, the quality of estimation using global filters should be at least as good as local filters. I also expect the UKF to perform better, at most as well as the EKF, due to the absence of linearization errors.

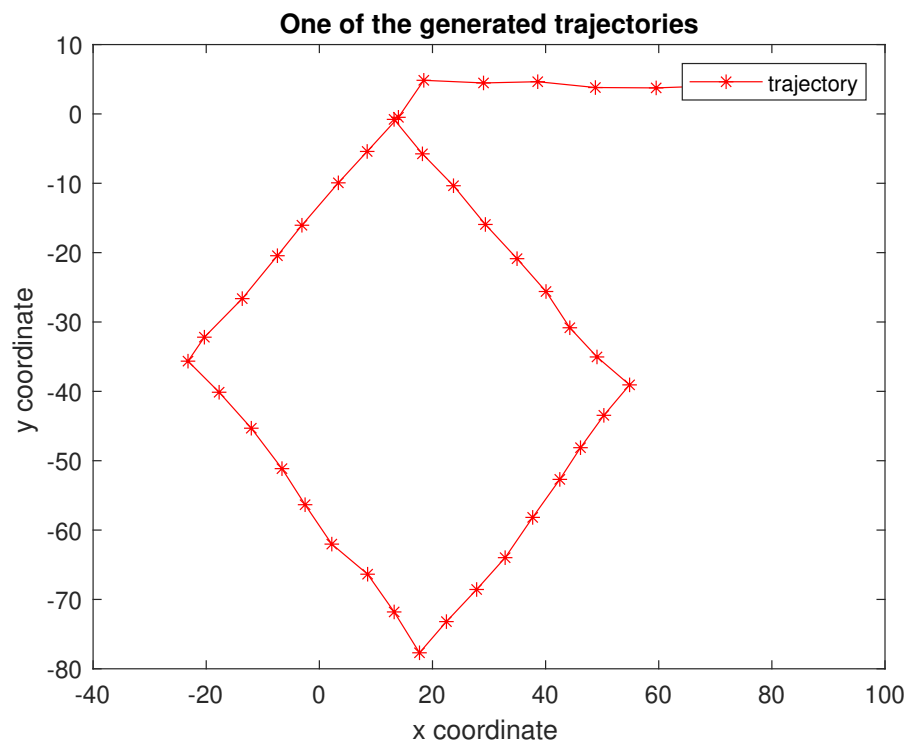


Figure 7.5: Example of the trajectory used in the simulation with the second system

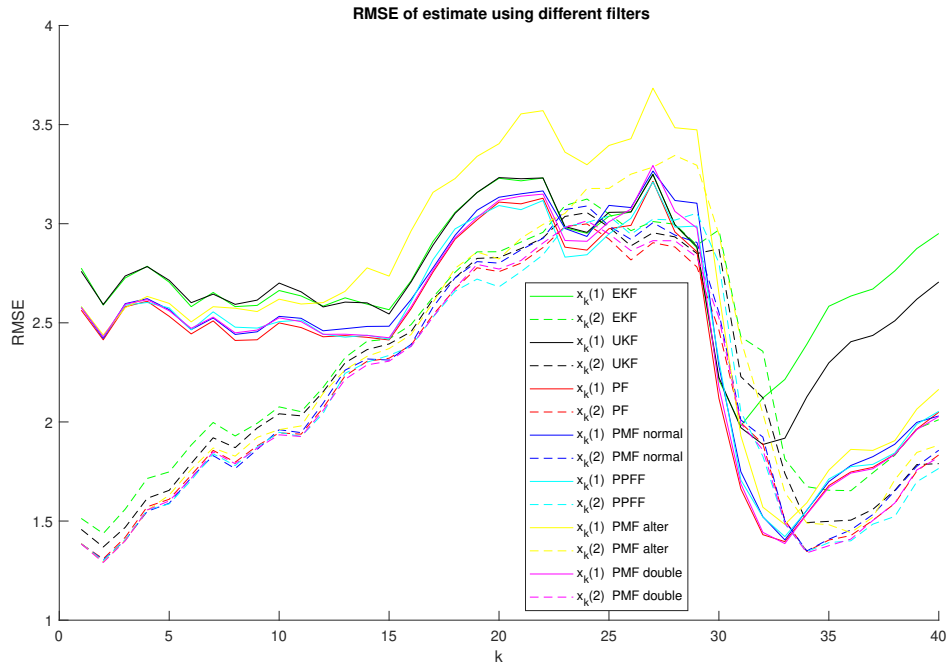


Figure 7.6: Simulation result - RMSE during the trajectory of the second system

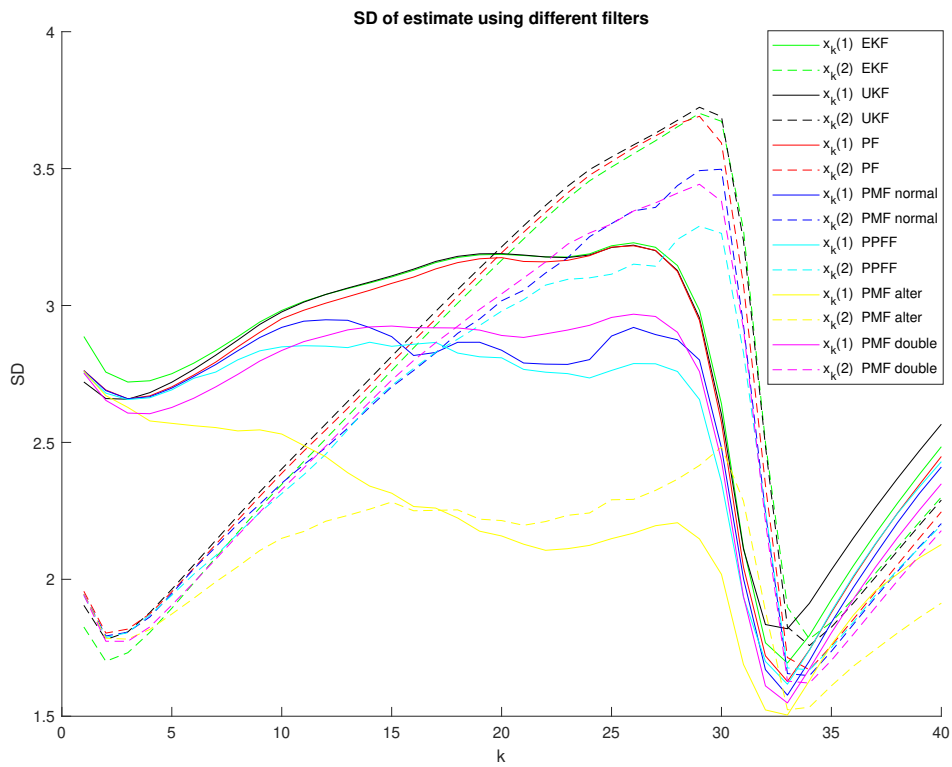


Figure 7.7: Simulation result - SD during the trajectory of the second system

filter	RMSE $x_k(1)$	RMSE $x_k(2)$	SD $x_k(1)$	SD $x_k(2)$	runtime [s]
EKF	2.7429	2.2979	2.8182	2.7007	$1.0768 \cdot 10^{-3}$
UKF	2.6878	2.2151	2.8205	2.7327	$1.7554 \cdot 10^{-3}$
PF	2.4426	2.1297	2.784	2.7003	0.36303
PMF normal	2.4945	2.1585	2.6376	2.5897	0.3224
PMF alter	2.6801	2.2528	2.2345	2.0818	0.31989
PMF double	2.4663	2.1296	2.6333	2.5821	0.30844
PPFF	2.4676	2.1413	2.6011	2.5322	0.2557

The results of the simulations with the second system are partially consistent with expectations. The only surprise is the poor performance of the alternative version of PMF. It turns out that a grid with the same number of points in each dimension is not as efficient as grids in which the number of points in a dimension is a function of the SD if the number of grid points is constrained to the same maximum value. A grid where the number of points in a dimension is a function of the current SD estimate is likely to cover the state space better and should be a better choice for real applications. As expected, the performance of the EKF is worse than that of the UKF. All global filters except the PMF alter achieved better results than EKF and UKF. The PMF with density specific grid design had better performance than the PMF with normal grid redesign method. PF and PMF double had the best results among the global filters. The PPFF performance was neither significantly good nor significantly poor.

### 7.3 Results of Simulations with the Third System

Two simulations were performed with the third system. Both simulations had the same planned trajectory, but the first simulation was not affected in any way, while *in the second simulation the tenth measurement is deliberately biased*. This distortion simulates a sensor error. Thus, the goal of the first simulation is to evaluate and compare the performance of different filters in the TAN estimation problem and the goal of the second simulation is to test the response to sensor error.

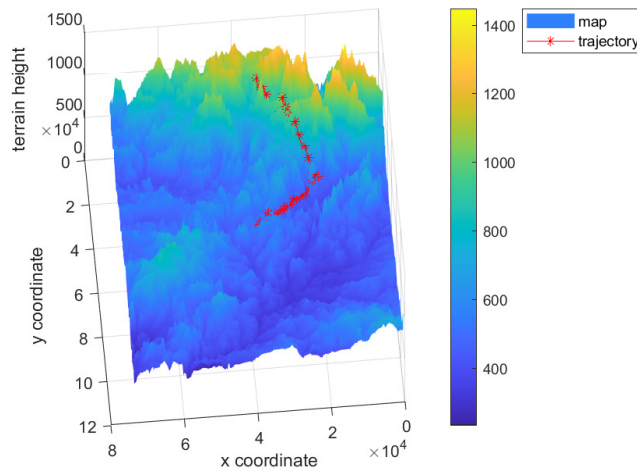


Figure 7.8: Example of the trajectory used in the simulation with the third system

### 7.3.1 First Simulation Using the Third System

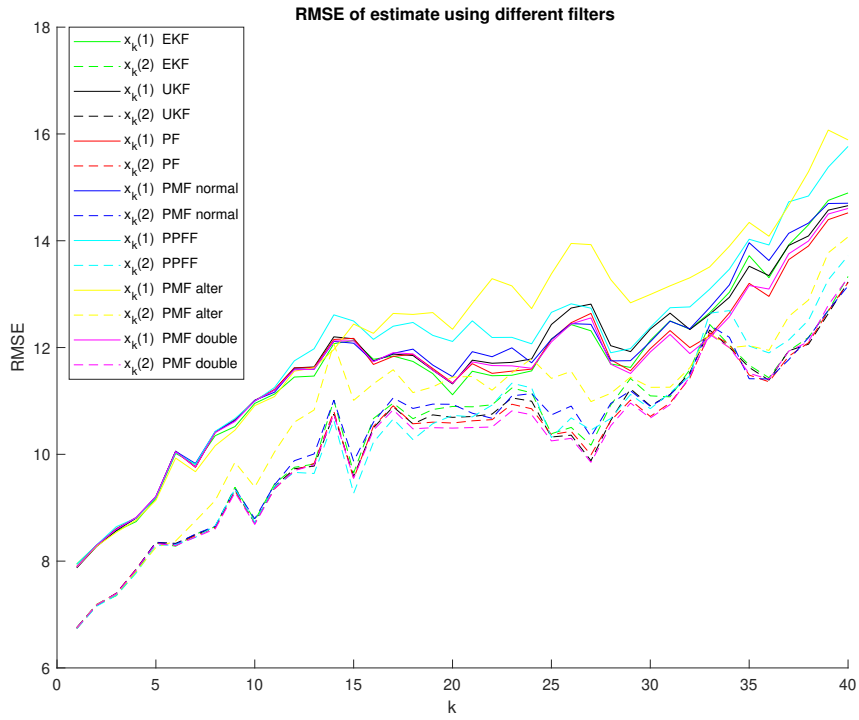


Figure 7.9: Simulation result - RMSE during the trajectory of the third system

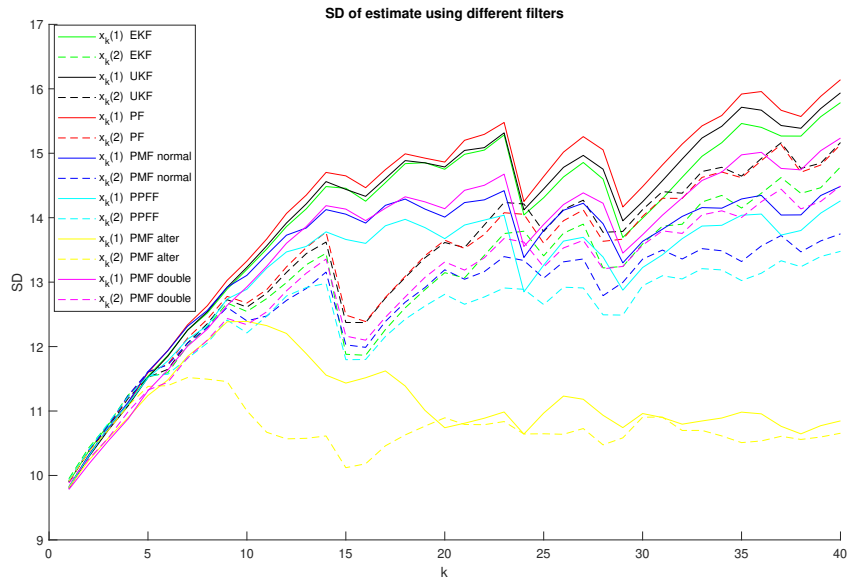


Figure 7.10: Simulation result - SD during the trajectory of the third system

filter	RMSE $x_k(1)$	RMSE $x_k(2)$	SD $x_k(1)$	SD $x_k(2)$	runtime [s]
EKF	11.6117	10.3452	14.0149	13.0641	$1.2333 \cdot 10^{-2}$
UKF	11.6931	10.2598	14.114	13.3977	$2.964 \cdot 10^{-3}$
PF	11.564	10.2239	14.2744	13.3736	0.26562
PMF normal	11.709	10.3569	13.4812	12.7473	0.32685
PMF alter	12.2877	10.8112	11.1746	10.7359	0.33295
PMF double	11.5649	10.1887	13.6338	12.9644	0.31546
PPFF	12.0107	10.3424	13.1856	12.4894	0.26875

Table 7.2: Results of first simulation with the third system

The results of the first simulation with the third system are somewhat similar to the results of the simulation with the second system. The difference is that the EKF achieved slightly better results for the  $x_k(1)$  estimate than the UKF and the UKF achieved better results for the  $x_k(2)$  estimate than the EKF. It is also no longer true that all global filters except the PMF alter had better results than EKF and UKF. As in the case of the simulation with the second system, the PMF double and the PF achieved the best results. The performance of the PPFF was not clearly better than the performance of any other filter except PMF alter.

### 7.3.2 Second Simulation Using the Third System

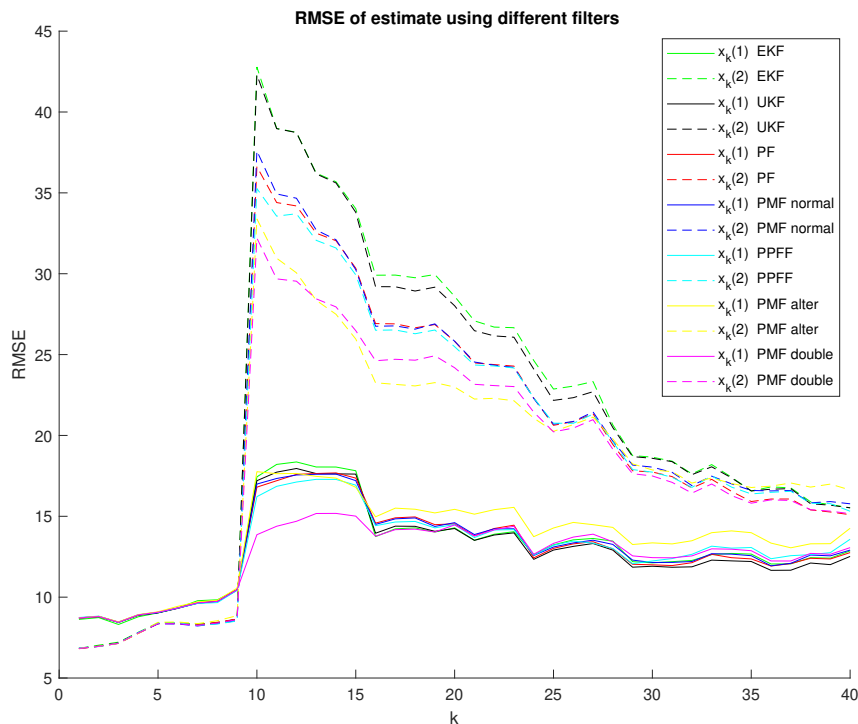


Figure 7.11: Simulation result - RMSE during the trajectory of the third system

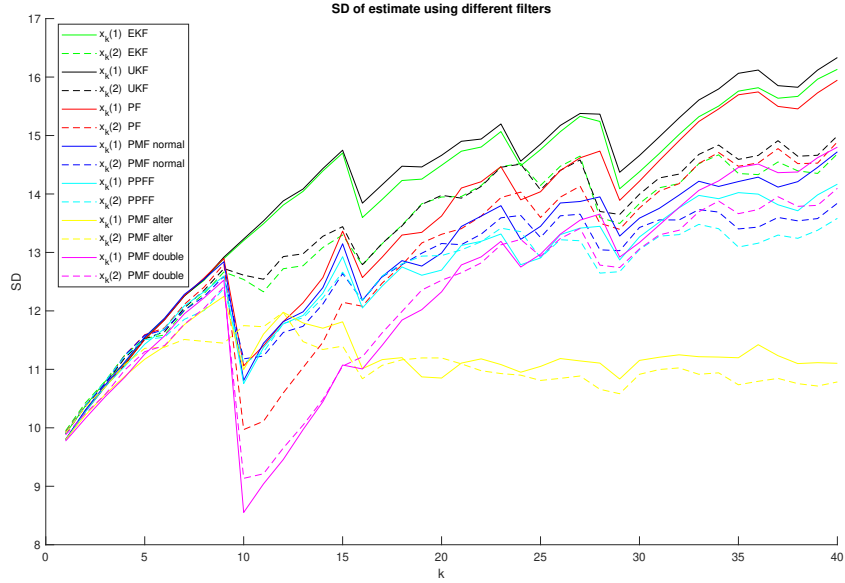


Figure 7.12: Simulation result - SD during the trajectory of the third system

filter	RMSE $x_k(1)$	RMSE $x_k(2)$	SD $x_k(1)$	SD $x_k(2)$	runtime [s]
EKF	12.9488	21.1758	14.1112	13.3556	$1.273 \cdot 10^{-2}$
UKF	12.7477	20.9272	14.2464	13.4517	$3.0152 \cdot 10^{-3}$
PF	12.8921	19.4796	13.5481	12.9514	0.38972
PMF normal	12.9377	19.6567	12.9633	12.6985	0.3546
PMF alter	13.6251	18.4848	11.2064	11.0431	0.37616
PMF double	12.5776	18.3811	12.4101	12.2632	0.3399
PPFF	12.9342	19.4038	12.6995	12.5327	0.29879

Table 7.3: Results of second simulation with the third system

In the ability to deal with wrong measurements, the PMF double clearly won. The UKF achieved the second smallest error in the  $x_k(1)$  estimate, but at the cost of a large error in the  $x_k(2)$  estimate. The performance of the PPF is better than that of the PMF normal and similar to that of the PF. The performance of the UKF was better than that of the EKF. The performance of the global filters except the PMF alter was better than that of the local filters.

### 7.3.3 Final filter comparison

In my opinion, the winner of the simulations is PMF double - PMF with density specific grid design. PPF failed to outperform the other global filters. It is possible that the simulations performed did not contain scenarios that would clearly show the weaknesses of the other filters, so I do not dare to claim that PPF is a worse filter. Considering that one of the ideas behind both PMF double and PPF is to cover the area around the center of the grid, where the expected value of the PDF is high, more efficiently with supports than in the case of PMF with equidistant grid, it seems that the approach used in the case of PMF with density specific grid design is better. UKF has shown that by not using linearization, it can outperform EKF.



## 8. Conclusion

In this thesis, I studied numerical methods for solving the problem of state estimation in Bayesian frames - global filters. These numerical methods are used for nonlinear estimation, for example in the problem of terrain aided navigation(TAN). Two main approaches were first presented - Point Mass Filter(PMF), which is based on deterministic coverage of the state space by grid points, and Particle Filter(PF), which is based on a stochastic approach. In the chapter dedicated to PMF, besides introducing PMF, the ideas behind it and giving its algorithm, two approaches to grid redesign - Normal approach and Density specific approach to grid redesign were discussed. The idea behind the Density specific approach to grid redesign is to cover the state space more efficiently by making better use of the available computational power(number of grid points) and thus improving the quality of the estimation. The chapter dedicated to PF included, besides the introduction of the filter, its idea, algorithm, also a section dedicated to the challenges associated with the implementation and design of PF and a section dedicated to resampling, which is a key component of PF. The last chapter dedicated to global filters dealt with the comparison of PF and PMF, stating its advantages and disadvantages. This chapter also introduced the idea of a filter that combines PF and PMF, which aims to take advantage of the strengths of both filters. The result of this idea is the Particle-Point Mass Fusion Filter (PPFF) which algorithm was presented. The last chapter of the theoretical part of this thesis dealt with local filters - the Kalman filter(KF) and its derivatives Extended Kalman filter (EKF) and Unscented Kalman filter (UKF). The Kalman filter was used as a benchmark to verify the performance of the global filters in a simulation with a linear system. EKF and UKF were used as reference in simulations with a nonlinear system.s The last chapter focuses on simulations, the aim of which is to verify and compare the performance of different filters.

Based on the simulations, I selected PMF with density specific grid redesign as the best way to cover the state space with supports. The question is how to evaluate the performance of PPFF in simulations. The performance of PPFF was reasonably good, but it could not outperform that of PMF with density specific grid redesign, nor could it consistently outperform that of PF and PMF with normal approach to grid redesign. The idea of combining PMF and PF sounds interesting and could be the subject of future work. In this future work, it might be interesting to try to design a filter based on the original idea presented [12] in and use neural networks to select user defined parameters.

# References

- [1] Kjetil Bergh Ånonsen and Ove Kent Hagen. An analysis of real-time terrain aided navigation results from a HUGIN AUV. In *OCEANS 2010 MTS/IEEE SEATTLE*, pages 1–9. IEEE, 2010.
- [2] Niclas Bergman. *Recursive Bayesian estimation: Navigation and tracking applications*. PhD thesis, Linköping University, 1999.
- [3] Jindřich Duník, Miloš Soták, Miloš Veselý, Ondřej Straka, and Wesley Hawkinson. Design of Rao–Blackwellized point-mass filter with application in terrain aided navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 55(1):251–272, 2018.
- [4] Jindřich Duník. *Identifikace systémů a filtrace. 2. přeprac. a rozšíř. vyd.* Západočeská univerzita v Plzni, 2018.
- [5] Jos Elfring, Elena Torta, and René van de Molengraft. Particle filters: A hands-on tutorial. *Sensors*, 21(2):438, 2021.
- [6] Mohinder S Grewal and Angus P Andrews. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.
- [7] Hesam Khazraj, F Faria Da Silva, and Claus Leth Bak. A performance comparison between extended Kalman Filter and unscented Kalman Filter in power system dynamic state estimation. In *2016 51st International Universities Power Engineering Conference (UPEC)*, pages 1–6. IEEE, 2016.
- [8] Tiancheng Li, Miodrag Bolic, and Petar M Djuric. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.
- [9] Jong Nam Lim and Chan Gook Park. RBPPFF for robust TAN. *IET Radar, Sonar & Navigation*, 13(12):2230–2243, 2019.
- [10] Jakub Matoušek, Jindřich Duník, and Ondřej Straka. Point-mass filter: Density specific grid design and implementation. In *15th European Workshop on Advanced Control and Diagnosis (ACD 2019) Proceedings of the Workshop Held in Bologna, Italy, on November 21–22, 2019*, pages 1093–1115. Springer, 2022.
- [11] Jakub Matoušek, Jindřich Duník, and Ondřej Straka. Density Difference Grid Design in a Point-Mass Filter. *Energies*, 13:4080, 08 2020.

- [12] Umut Orguner, Per Skoglar, David Törnqvist, and Fredrik Gustafsson. Combined point-mass and particle filter for target tracking. In *2010 IEEE Aerospace Conference*, pages 1–10. IEEE, 2010.
- [13] Yan Pei, Swarnendu Biswas, Donald S Fussell, and Keshav Pingali. An elementary introduction to Kalman filtering. *Communications of the ACM*, 62(11):122–133, 2019.
- [14] Matthew B Rhudy, Roger A Salguero, and Keaton Holappa. A Kalman filtering tutorial for undergraduate students. *International Journal of Computer Science & Engineering Survey*, 8(1):1–9, 2017.
- [15] Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge university press, 2013.
- [16] Ondřej Straka and Miroslav Šimandl. Adaptive particle filter with fixed empirical density quality. *IFAC Proceedings Volumes*, 41(2):6484–6489, 2008.
- [17] Gabriel A Terejanu. Unscented Kalman filter tutorial. Technical report, University at Buffalo, 2011.
- [18] Gabriel A Terejanu et al. Extended Kalman Filter Tutorial. Technical report, University at Buffalo, 2008.
- [19] Daniela Vaman. *A GPS inspired terrain referenced navigation algorithm*. PhD thesis, Technische Universiteit Delft, 2014.
- [20] Oliver J Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory, 2007.