Nonlinear Filtering based on Log-homotopy Particle Flow

Methodological Clarification and Numerical Evaluation

Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Muhammad Altamash Ahmed Khan

aus

Islamabad, Pakistan

Bonn, 2018

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Wolfgang Koch
2. Gutachter:	Prof. Dr. Reinhard Klein

Tag der Promotion:	02.08.2018
Erscheinungsjahr:	2018

Abstract

The state estimation of dynamical systems based on measurements is an ubiquitous problem. This is relevant in applications like robotics, industrial manufacturing, computer vision, target tracking etc. Recursive Bayesian methodology can then be used to estimate the hidden states of a dynamical system. The procedure consists of two steps: a process update based on solving the equations modeling the state evolution, and a measurement update in which the prior knowledge about the system is improved based on the measurements. For most real world systems, both the evolution and the measurement models are nonlinear functions of the system states. Additionally, both models can also be perturbed by random noise sources, which could be non-Gaussian in their nature. Unlike linear Gaussian models, there does not exist any optimal estimation scheme for nonlinear/non-Gaussian scenarios.

This thesis investigates a particular method for nonlinear and non-Gaussian data assimilation, termed as the *log-homotopy based particle flow*. Practical filters based on such flows have been known in the literature as Daum Huang filters (DHF), named after the developers. The key concept behind such filters is the gradual inclusion of measurements to counter a major drawback of single step update schemes like the particle filters i.e. namely the *degeneracy*. This could refer to a situation where the likelihood function has its probability mass well separated from the prior density, and/or is peaked in comparison. Conventional sampling or grid based techniques do not perform well under such circumstances and in order to achieve a reasonable accuracy, could incur a high processing cost. DHF is a sampling based scheme, which provides a unique way to tackle this challenge thereby lowering the processing cost. This is achieved by dividing the single measurement update step into multiple sub step, such that particles originating from their prior locations are graduated incrementally until they reach their final location. The motion is controlled by a differential equation, which is numerically solved to yield the updated states.

DH filters, even though not new in the literature, have not been fully explored in the detail yet. They lack the in-depth analysis that the other contemporary filters have gone through. Especially, the implementation details for the DHF are very application specific. In this work, we have pursued four main objectives. The first objective is the exploration of theoretical concepts behind DHF. Secondly, we build an understanding of the existing implementation framework and highlight its potential shortcomings. As a sub task to this, we carry out a detailed study of important factors that affect the performance of a DHF, and suggest possible improvements for each of those factors. The third objective is to use the improved implementation to derive new filtering algorithms. Finally, we have extended the DHF theory and derived new flow equations and filters to cater for more general scenarios.

Improvements in the implementation architecture of a standard DHF is one of the key contributions of this thesis. The scope of applicability of DHF is expanded by combining it with other schemes like the Sequential Markov chain Monte Carlo and the tensor decomposition based solution of the Fokker Planck equation, resulting in the development of new nonlinear filtering algorithms. The standard DHF, using improved implementation and the newly derived algorithms are tested in challenging simulated test scenarios. Detailed analysis have been carried out, together with the comparison against more established filtering schemes. Estimation error and the processing time are used as important performance parameters. We show that our new filtering algorithms exhibit marked performance improvements over the traditional schemes.

***Index terms*— Log homotopy based particle flow filter**, **Fokker Planck equation**, **Massive sensor data**, **Sequential Markov chain Monte Carlo**, **Confidence sampling**, **Subsampling**, **Tensor decomposition**, **Alternating least squares**, **Gaussian mixture models**.

Acknowledgements

All praise belongs to the God Almighty, the fashioner and sustainer of the heavens and the earth, without Whose will and blessings I couldn't have managed to finish this enormous undertaking.

I would like to thank my advisor Prof. Wolfgang Koch. It was an honor and absolute pleasure to work under him at Fraunhofer FKIE and to be his Ph.D. student. He has been extremely supportive and helpful during the whole span of my doctoral studies. He always showed faith in me, which together with his openness to discuss ideas and continual encouragement, helped me to keep up the motivation level even during toughest periods and to finish the thesis on time.

I would like to express sincere gratitude to my project supervisor Dr. Martin Ulmke, who has been a source of consistent guidance and support. I appreciate all his contributions of time and ideas to make my Ph.D. experience productive and stimulating. I found him to be a very kind and well-natured person and an equally distinguished researcher. It was through his invaluable advice and mentoring that I could clearly formulate the research problem and subsequently delve deeper into the topic.

Members of the SDF group at FKIE have been a source of friendships, as well as good advice and collaboration. In particular, I would like to mention Dr. Felix Govaers and Dr. Bruno Demissie, whose guidance and professional advice greatly helped me in my academic endeavors. Their work on novel nonlinear filtering methods became the motivation to explore tensor decomposition based filtering methodology which constitutes an important part of my research work. I would also thank them for proofreading my thesis and correcting mistakes.

I thank Dr. Lyudmila Mihaylova for extending full support during my stay at the University of Sheffield. Our joint work became the basis for the development of log homotopy based nonlinear filtering for massive sensor data.

My special thanks and gratitude would go to my family for all their love and encouragement. For my parents who raised me with a sense of curiosity and wonder, and fully supported me in all my pursuits. For my brother, whom I am ever proud of for being such an inspiration, and for my sister whose love and affection I truly cherish. I would not have been what I am and where I am without my family.

Finally, I would like to acknowledge the financial support from the European Union under the Marie Curie Actions in the Seventh Framework Program [FP7 2013-2017] TRAcking in compleX sensor systems (TRAX) with grant agreement no. 607400.

Contents

Chapter 1

Introduction

The natural world is full of individual bodies and objects that interact with each other, and through those interactions they influence each other's behavior. In scientific terms, such a group of interacting entities which constitute one complex whole is referred to as a *system*. The first step in analyzing a system is to chart out its specific input/output behavior. It starts with specifying the condition of a system as a function of time, and is referred to as its *state*. By describing a system in terms of its state(s), allows it to be studied quantitatively. All systems can be categorized into two groups: static or dynamical systems. A static system is one that does not change its state(s) over the time, or does so very slowly. Examples of this include mechanical systems like a bridge, a high rise building, etc. A dynamical system, on the other hand, exhibits a perceivable change in its state over time. Examples of such systems include the solar system, interacting molecules in a chemical reaction, human body responding to the administration of a drug, network of computers, ecological system, socio-cultural systems etc.
Another categorization can be made on the basis of the change systems undergo w.r.t. the time. They can be either be *continuous time* systems, for which the state evolution is perpetual over the time, or *discrete time* systems that exhibit change only at certain time instances. Almost all real world systems belong to the former category. In most of the cases, the study of the continuous time dynamical systems is rather complex to be carried out effectively. Therefore, as a possible solution, they are converted into an equivalent discrete time representation through a mathematical procedure called the *discretization*.
State estimation of a discretized dynamical system is usually carried out in a *state space* framework, where a set of defining states, comprising a state vector is used for its description. The state evolution is defined by a system function, which could depend on the state values from the previous time steps, and external control inputs. Furthermore, there could be inherent uncertainties in the evolutionary process, which could lead to a non-deterministic state progression. In most cases, it is not possible to directly observe the state vector. Instead, a related set of variables are usually observed. This set is referred to as the *measurements*. Measurements and the states could relate to each other in a linear or nonlinear manner. Furthermore, measurements could also be corrupted by random noises. In Figure 1.1, we depict a state space model for a generic discrete time system. Here, k is the time index and $\mathbf{x}_k$, $\mathbf{u}_k$ and $\mathbf{z}_k$ are the state, control and the measurement vectors, respectively. Also, $\mathbf{w}_k$ and $\mathbf{v}_k$ are the system and measurement noises. The system block refers to the state evolutionary function, while the measurement block represents the functional relationship between states and measurements.

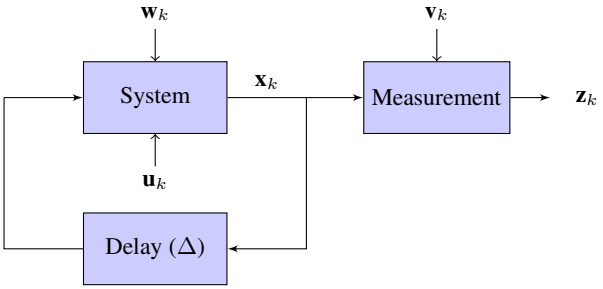


Figure 1.1: State space model for a generic discrete time dynamical system

The delay block points to the fact that the state evolution is a recursive process. Since both the system and the measurement models have random noise contributions, the state estimation based on a simple inverse transformation (measurements to state) could be imprecise. Therefore, the state estimation has to be done in a framework, which includes the statistical formulation of the uncertainties.

Luckily, we have just that type of framework in the form of *Recursive Bayesian Estimation* (RBE). Instead of calculating a single point estimate, RBE allows for the formulation of information about the state in terms of a probability density function (pdf). Measurements are represented through a likelihood function. The process starts with an initial state pdf, which represents the measure of uncertainty at the beginning. Upon the arrival of measurements, the initial density is updated with the help of the likelihood function, yielding the posterior pdf. This is also called the *Data assimilation* step. This is followed by the time update of the posterior pdf through the System, leading to the formation of a prior belief for the second data assimilation step. The procedure continues thereafter in a similar recursive manner. We show a generic RBE procedure in Figure 1.2,

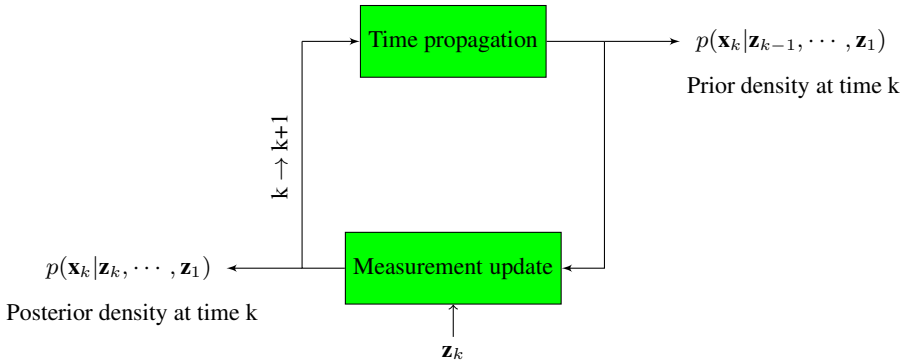


Figure 1.2: Bayesian recursive estimation

The posterior density in this form is also called the *filtering density*. The main focus of this thesis is to develop new methods to solve the RBE for the filtering density, in an efficient manner.

1.1 Motivating examples

Satellite based positioning

Accurate navigation has always been a main concern of travelers throughout the history. In the ancient days, stars were used by the caravans criss-crossing deserts and sailors on their voyages to the known and unknown lands. In particular, navigation on the high seas was of special interest, as it was through there that most of the world trade and exploration was carried out. Star based positioning was later on augmented by the use of instruments like astrolabes, clocks, compasses and gyroscopes. This led to the use of *Dead reckoning* to plot the current location of a ship based on its last estimated position, current speed, time elapsed since the last way-point and its bearing. Given the rather unsophisticated nature of the measurement devices, inherent faults and natural limitations (like requirement for the clear weather) were always present, leading to faulty navigation over the large stretches of oceans. The paradigm was significantly changed with the advent of radio communications making the maritime navigation much more accurate. This evolved through the use of the radio navigation systems like OMEGA, TACAN, DECCA, LOREN-C etc. to estimate the direction to a known radio source. These systems were later aided by the more accurate, though localized, radar based navigation techniques used both by the maritime and airborne traffic. However, a key requirement for using the radio navigation methods was the existence of the supporting terrestrial infrastructure in the area of operation e.g. radio stations. This limited their usage.

With the arrival of the space age, new possibilities arose to significantly improve the navigation across the whole globe. The concept was to use space vehicles orbiting the earth in lower orbits, to broadcast the necessary information to accurately position a ground object anywhere on the globe. The archetype is called the Global Navigation Satellite System (GNSS), which is usually based on a constellation of satellites circling the earth in low to medium earth orbits. Examples of GNSS include the American NAVSTAR Global Positioning system or *GPS*, the Russian *GLONASS*, the European *Galileo* and the Chinese *Beidou* navigation system.

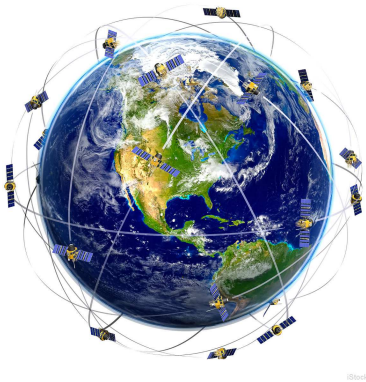


Figure 1.3: Global Navigation Satellite System (GNSS) constellation

Amongst all, the GPS is the most matured and well established system, which is also the most

widely used one. GPS consists of three major components: space, control and users. Space component consist of twenty four satellites placed in six orbital planes equispaced around the equator, transmitting navigational signals. The arrangement ensures that any user on the earth has always a direct line of sight with at least four satellites. The control component includes the ground tracking stations used for monitoring the satellites and adjustments in their parameters. The last segment consists of users equipped with GPS receivers. These receivers are responsible for *Radio Frequency* and the *Baseband* processing of the received signal in order to calculate the estimated satellite positions and the signal transit time. This information is further used by an embedded *Navigation* unit to estimate the user states i.e position, velocity, heading and the time. The transit time, calculated in the form of pseudo-ranges, are rendered inaccurate due to the presence of atmospheric delays, selective availability, multipaths, receiver noise, carrier frequency tracking ambiguities and the satellite and receiver clock biases [Far08]. Furthermore, the pseudo-range measurements are nonlinear functions of the receiver location. This makes the user state estimation a nonlinear RBE problem, requiring the use of appropriate models and filtering algorithms.

Maritime Surveillance

Sea borne trade is the backbone of global trade, and one of the major drivers of the world's economy. According to the annual report of United Nations Conference on Trade and Development (UNCTAD), 80 percent of the global trade by volume and 70 percent of the global trade by value is done through the maritime routes [UNC15]. Presented below, is a snapshot of the maritime traffic across the world oceans and inland waterways.

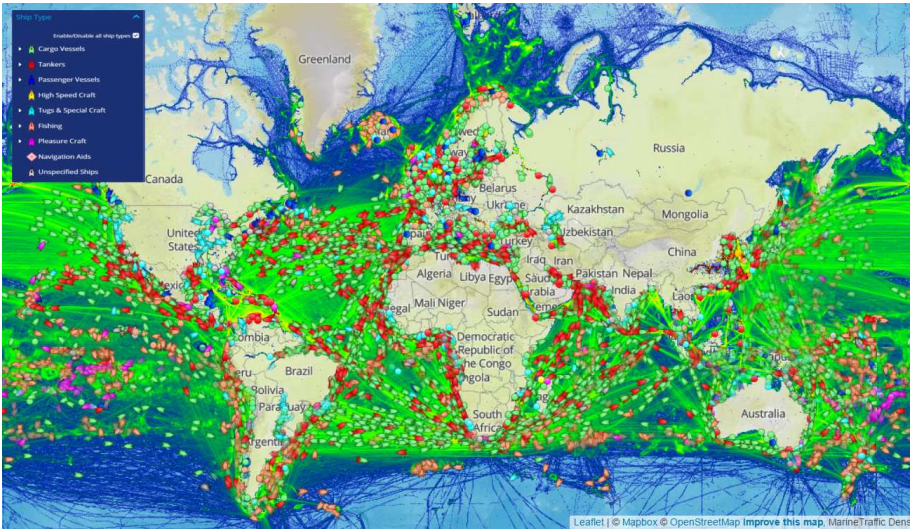


Figure 1.4: Maritime traffic map (Source: https://www.marinetraffic.com/en/ais/home)

Therefore, the monitoring and supervision of traffic on the high seas and inland waterways

is of prime interest. There are several types of operations that fall in this category [EU117]. Firstly, navies all around the world maintain vigil to monitor their respective national maritime zones against enemy surface or sub-surface incursions. This also includes keeping a close eye on the coastal areas for thwarting any potential attempts of infiltration by enemy military personnel for sabotage and other subversive activities. Next, the law enforcement agencies of countries with extensive coastline have to be on a look out for smuggling activities and illegal trafficking. in this task, they can be aided by military maritime patrol assets. Specialized operations like visit, board, search, and seizure (VBSS) can be conducted, designed to capture enemy vessels, to combat terrorism, piracy and smuggling, and to conduct customs, safety and other inspections. Another important task is the protection and regulation of fishing and exploitation of the natural resources in a country's exclusive economic zones. Lastly, maritime agencies might have to respond to the distress calls in the case of vessel damage or sinking. This might require launching search and rescue missions (SAR), involving helicopters and boats to the last reported position of the vessel.

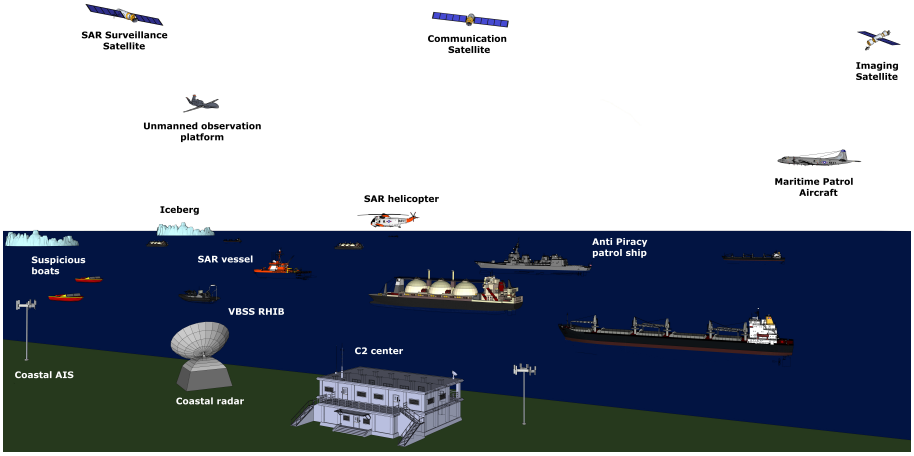


Figure 1.5: A typical maritime surveillance scenario

A main feature of any maritime surveillance operation is the acquisition of vessels positioning data. This could be done through three methods: *Self reporting*, *Observation based data* and *Information database*. Sea vessels typically self report their position and other relevant information through the *Automatic Identification System* (AIS) in their vicinity to avoid collisions. They can also transmit the data via satellite communication links to the competent authorities for identification and tracking purposes. As the second mode of vessel positioning, passive or active sensors could be employed to gather data. This might include space borne platforms like Synthetic Aperture Radar (SAR) or Electro-Optical (EO) sensors equipped satellites, high flying unmanned observation platforms (UAV) or maritime patrol aircraft (MPA). Thirdly, information about the vessels like their appearance, construction, history, type, etc. can also be used to identify them. All of these methods have their inherent limitations, which restrict their stand alone application for the general maritime surveillance. For example, sensors could have different sampling rates, latencies, errors etc. Also not all information might be available to

maritime authorities at any given time. In addition, the self reported information could be falsified. Therefore, the fusion of data is necessary to improve the positioning accuracy of sea vessels. This is done at a command and control (C^2) center, where the data gathered by all sources is merged. The information generated thereof is used in the situation prediction (to forecast traffic maps), anomaly detection (identifying abnormal traffic pattern which could be due to an illegal activity), mapping activities at the sea and improving the overall maritime safety and security [EUJ17]. Sensor fusion lies at the heart of all such activities, and is carried out in a recursive Bayesian estimation context.

Advanced Driver Assistance Systems

Passive automotive safety systems are designed to lessen the effects of accidents or collisions. They cannot influence the occurrence of such incidents in the first place. Examples of such systems include seat belts, air bags and belt pre-tensioner etc. However, with the proliferation of cheap sensors and computational technology, the automotive industry has been provided with the necessary tools to significantly improve drivers safety and comfort. This calls for the development of specialized hardware and the software, that could present the driver with the most needed information, thereby augmenting his/her ability to make timely decisions through better judgment. This could not only result in a smoother and safer driving, but it could also lead to the automation of major driving tasks, hence reducing driver's work load. In the later case, an on board computer can facilitate the driver by assisting him/her or by completely taking over the driving tasks. The whole concept relies on the fusion of sensory data, gathered by the on-board sensors to create a general picture of vehicle's surroundings, which is subsequently

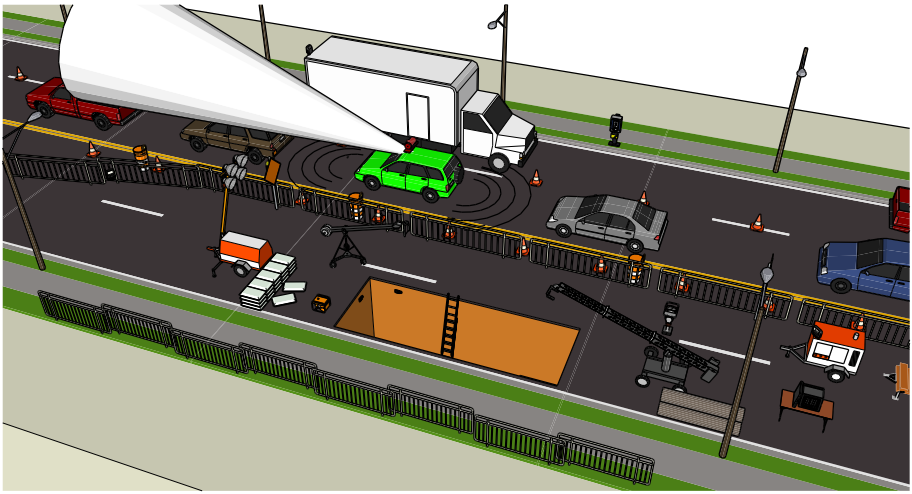


Figure 1.6: Sensor fusion based vehicular ADAS system. The figure shows a typical highway scenario, with ongoing works on the one half. Ego vehicle (green) employs ADAS to help the driver in safely navigating through the jam. The two sensor shown are the top mounted camera and radars located in the front and back of the vehicle. The white cone represents the field of vision of the camera.

used by control units to perform the required tasks. This constitutes the realm of Advanced Driver Assistance Systems (ADAS). ADAS are designed to prevent accidents by changing the dynamical parameters of the car. Examples of such systems include collision avoidance, Anti-lock Braking System (ABS) and Adaptive Cruise Control (ACC). A detailed description of such systems can be found in [Eid04], [Jan05] and [Gus05]. All active safety systems employ state estimation of the ego vehicle i.e. the vehicle to be controlled, and most of the times, of the other vehicles and the surrounding road objects. Typical vehicular states include the position, velocity, acceleration, three angles (pitch, roll, yaw) and their rates. Usually a host of sensors are employed in the estimating those states. This could include, GPS, steering angle, wheel speed, Inertial Measurement unit (IMU), Radar, Camera, Map database etc. Some early works in this field include [Nwa93], [SFD02], [SSCT04], [GdlEA13] and [SY16].
A typical ADAS scenario is shown in Figure 1.6. The car is shown to be carrying two sensors aboard, a radar and a camera. The camera is used to extract feature information from the captured images, which can be further used to track objects. In addition to that, the radar provides information regarding the range, relative velocity, acceleration and the relative heading. A camera sensor is very good at locating object in the azimuth but suffers from poor radial range accuracy. Radar, on other hand is very good at estimating the direct distance between vehicles but its cross range uncertainty is relatively large. It should be noted that measurements from both sensors are nonlinearly related to the true states, and are also corrupted by noise. One of the methods used to combine the two data is the so called *Covariance intersection* (CI) method, as described in the context of the Track-to-Track fusion (T2TF) in [SR08]. This happens in two stages. In the first stage, each sensor runs its own Bayesian filter on the data and generates a mean and a covariance estimate. In the second step, the processed estimates from the two filters are combined using the CI method. CI results are optimal if the noise densities are Gaussian and there exist no cross variance between two filtered estimates. The *Covariance intersection* is shown below,

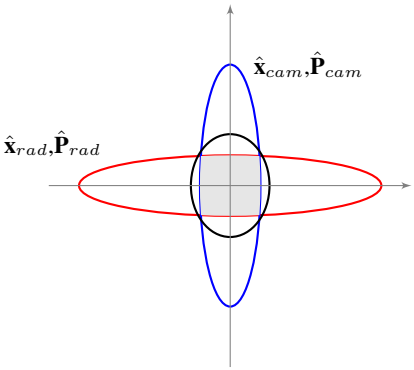


Figure 1.7: Fusion of camera & radar based estimates using *Covariance intersection*

However, neither of the two assumptions hold valid in the reality. The noise densities are seldom Gaussian, and most of the times, the two estimates (from radar and camera) are correlated. Hence, the need for a better nonlinear estimation methodology, to be used in the sensor fusion context is ever present. At this point it is worth rephrasing a particular comment from [Gus05]: *It is more important to have accurate state information than advanced control algorithms*.

1.2 Traditional state estimation methodologies: Strengths & Weaknesses

Based on the form of the state propagation and measurement functions, a system can be classified as either being linear or nonlinear. Another categorization can be made based on the nature of uncertainties. In the case of a linear Gaussian system (LGS), i.e. one in which both state propagation and measurement functions are linear and the two noises together with the initial uncertainty are Gaussian, a closed form estimator can be derived. This turns out to be the much celebrated Kalman Filter (KF), as derived by Rudolf E. Kalman [Kal60]. Kalman filter is optimal for LG conditions in the sense that, there exists no other estimator which has a lower mean square error (MSE). The filter has an elegant form, and it is also very simple to implement. Since the filter is based on the Gaussian assumption, it remains bounded in its dimensionality throughout the time. This is because the sufficient statistics to represent the Gaussian densities are just the mean and the covariance matrix. However, real life systems are seldom linear and/or Gaussian. Hence, Kalman filter can not be directly applied in those cases.
Generally, it is quite hard to derive an exact closed form estimator for nonlinear / non-Gaussian cases, and therefore, approximate methods are sought after. One can broadly categorize most of these methods into four classes: Kalman filter based methods, Grid based schemes, Sequential Monte Carlo (SMC) and Sequential Markov Chain Monte Carlo (SMCMC) methods. In a first approach, Kalman filter theory is extended to incorporate the effect of nonlinear functions, while still assuming the Gaussian assumptions for the initial uncertainty and noises. In its first manifestation, the linearization of the two models is performed around the current state estimate, allowing for the Kalman filter equation to be applied. Alternatively, a set of deterministically sampled values are propagated through nonlinear functions. The transformed points are then used in the estimation of the prior and posterior point estimates e.g. mean and covariance. Examples of these method include the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Cubature Kalman Filter (CKF). However, these methods are generally suboptimal and their performance degrades with the increase in the nonlinearity, and also when the transition and measurement densities are non-Gaussian (e.g. multimodal, exponential). For an illustration, we solve the Bayesian recursion for the following dynamical system,

$$x_{k+1} = 0.5x_k + 25\frac{x_k}{1+x_k^2} + 2\cos(1.2k) + w_k \qquad , \qquad z_{k+1} = \frac{x_{k+1}^2}{20} + v_{k+1}$$

and plot the evolution of the true posterior density alongside the EKF based density estimates in Figure 1.8. Here, the initial density is given by $\mathcal{N}(3,5)$, whereas the process and measurement noises w_k and v_k are $\mathcal{N}(0,10)$ and $\mathcal{N}(0,1)$, respectively. We can observe that the true posterior density is clearly non-Gaussian (bi-modal) for most of the time, which makes the EKF based density estimates diverge from the true posterior density.
The second class of estimators numerically approximates the prior and posterior densities over a discretized region of the state space, as discussed in [BL97], [SKS02], [SKS06], [SK15a] and [KUD$^+$16]. Process update is usually carried out by numerically solving a Partial Differential Equation (PDE) or an Integral Equation (IE), yielding a prior density estimate over the discretized region. In the measurement update step, the prior density is point-wise multiplied with the likelihood and normalized to form the posterior density estimate The main disadvantage with these type of methods is unfavorable scaling of the number of grid points with increasing dimensionality.

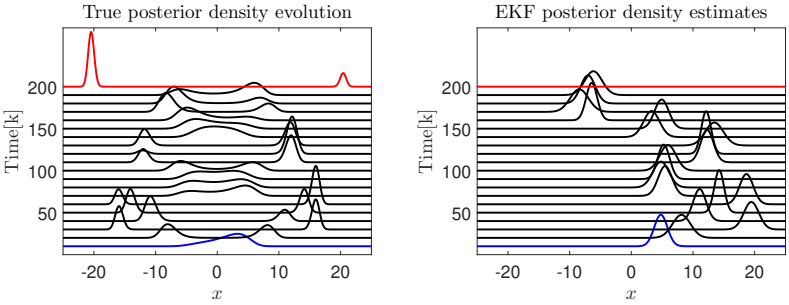


Figure 1.8: EKF based estimation of the true posterior density

The third class of nonlinear/non-Gaussian estimators fall in the category of Sequential Monte Carlo (SMC) methods, more famously known as particle filters [AMGC02], [GSS93]. In its most basic form, a particle filter is initialized with a set of particles, which are drawn from some initial distribution. State update is performed by sampling from an importance density. On the arrival of measurements, particles are weighted according to their likelihood. Particles are then used to estimate the state mean and covariance. Finally, the most likely particles are replicated and assigned uniform weights, while the rest are discarded. This procedure is performed recursively. Several versions of particle filters have been proposed in the literature, e.g. the sampling importance resampling (SIR) filter [GSS93] also known as the bootstrap particle filter, the auxiliary sampling importance resampling (ASIR) filter [PS99], the regularized particle filter (RPF) [MOLG01] etc.

Sequential Markov Chain Monte Carlo (SMCMC) based methods constitute yet another class of state estimation procedures. Traditionally, MCMC has been employed in sampling from complex distributions occurring in statistical physics [MRR$^+$53]. A Markov chain is created whose stationary distribution is the distribution of interest. New samples are generated using a proposal density, which can either be accepted or rejected. If accepted, the chain is said to have moved. When applied to sampling from a posterior distribution in a sequential setting, such as in the target tracking, the procedure is known as SMCMC. SMCMC methods provide a powerful tool for the state estimation of nonlinear / non-Gaussian system [KBD05]. Depending on the length of the constructed chain and the space exploration methodology, they can provide very accurate estimates of the posterior distribution.

While grid based methods, SMC and SMCMC based approaches can effectively deal with nonlinearities and non-Gaussian noises, they suffer from the so called *curse of dimensionality*. It refers to the exponential increase in the size of the discretized problem (in the first case) or the required number of statistical samples (for the rest of the two) to adequately represent the involved densities, with a linear increase in the system dimensionality. As a consequence, it could become computationally very hard to efficiently solve the RBE, even for moderately sized problems. Another problem, faced specifically by SMC methods is the *weight degeneracy*. Weight degeneracy refers to the fact that after few updates all but one or very few particles have negligible weights. Weight degeneracy occurs when the posterior distribution does not significantly overlap with the prior distribution. SMCMC also has its specific limitations. E.g. the proposal distribution has to be very well adapted to the target distribution (posterior), if its processing time has to be kept under a certain threshold. In the other case, the required number of iterations

could be very large even for simple high dimensional models, as illustrated in [FSMG15]. We will elaborate on these concepts in more detailed manner in the Chapter 2.

1.3 Particle flow : A paradigm for nonlinear & non-Gaussian data assimilation

It was earlier pointed out that the exponentially increasing required number of particles and their degeneracy are some of the major issues faced by nonlinear / non-Gaussian estimation algorithms. A careful analysis can reveal that these two issues are coupled to each other. A degenerated set of particles, upon re-sampling, leads to a particle set with too small variance. Particle replication could be helpful, but at the cost of reduced statistical diversity. This means that the target density cannot be accurately approximated if degeneration of particles happens. This problem gets exacerbated in higher dimensions because the inter-sample distance in higher dimensions could be very large. Thus, by reducing the set of *good* particles, the density approximation gets even worse. This concept is explained in more detail in paragraph 2 of the section 2.3.3.

It was realized that one possible remedy to this problem could be a gradual inclusion of measurements, in the measurement update step of a RBE algorithm. This could be done is several ways. For example, one could form a set of intermediate densities, filling in between the prior and the posterior densities, and sample from that set in a sequential manner [GC01]. Another concept could be to parameterize the densities and then introduce the likelihood in a sequential manner, to the updated parameters of the posterior density in steps [HF03], [HJSM11]. A more thorough approach is provided by the authors in [Rei13], [HDP15], [BG14], and [EMFD15], where particles from the prior density are *moved* through a *fictitious time* by solving an ordinary differential equation (ODE), such that its solution yields the updated posterior samples. These methods circumvent the need for the solution of a particular partial differential equation (PDE), governing the change in the density, and instead approximately update the particle states by solving a relatively simple differential equation. The PDE being referred to here is the Fokker-Planck equation (FPE), which describes the evolution of the probability w.r.t. some measure of time, for particles flowing in a force field undergoing simultaneous drift and diffusion.

A common issue with all of the above mentioned methods is that, while they are better than SMC in performance, they are usually computationally very demanding. Therefore, we look for another type of method, which though similar in concept, strikes a better trade-off between the complexity and estimation accuracy. We find that such a method for nonlinear/non-Gaussian has been suggested by F. Daum and J. Huang in a series of papers [DH07]- [DH09a], which is also based on the gradual inclusion of the measurements. The key idea there is to model the transition of particles from the prior to the posterior density as a physical flow under the influence of an external force (measurements). Particles are sampled from the state transition density and a notion of synthetic time also called the pseudo-time is introduced in which particles are flown, until they reach their posterior locations. The FPE describes the density evolution. Again, instead of solving it, an ODE termed as the *flow equation* is solved, which is derived analytically by solving the FPE under different assumptions. The flow vector is integrated numerically, yielding the updated particles states. This filter is termed as log-homotopy based particle flow filter or simply Daum Huang filter (DHF), named after the developers. Different flow solutions have been derived, including the incompressible flow [DH07], zero diffusion exact flow [DH09b], Coulomb's law flow [DHN11a] and zero-curvature flow [DH13b] non zero

diffusion flow (NZD) [DH13a]. It has been shown that the DHF are generally more effective in dealing with the particle degeneracy and they outperform particle filters for solving difficult high dimensional nonlinear / non-Gaussian problems ([DC12], [KU15], [KBS15], [MC15], [LC16]).

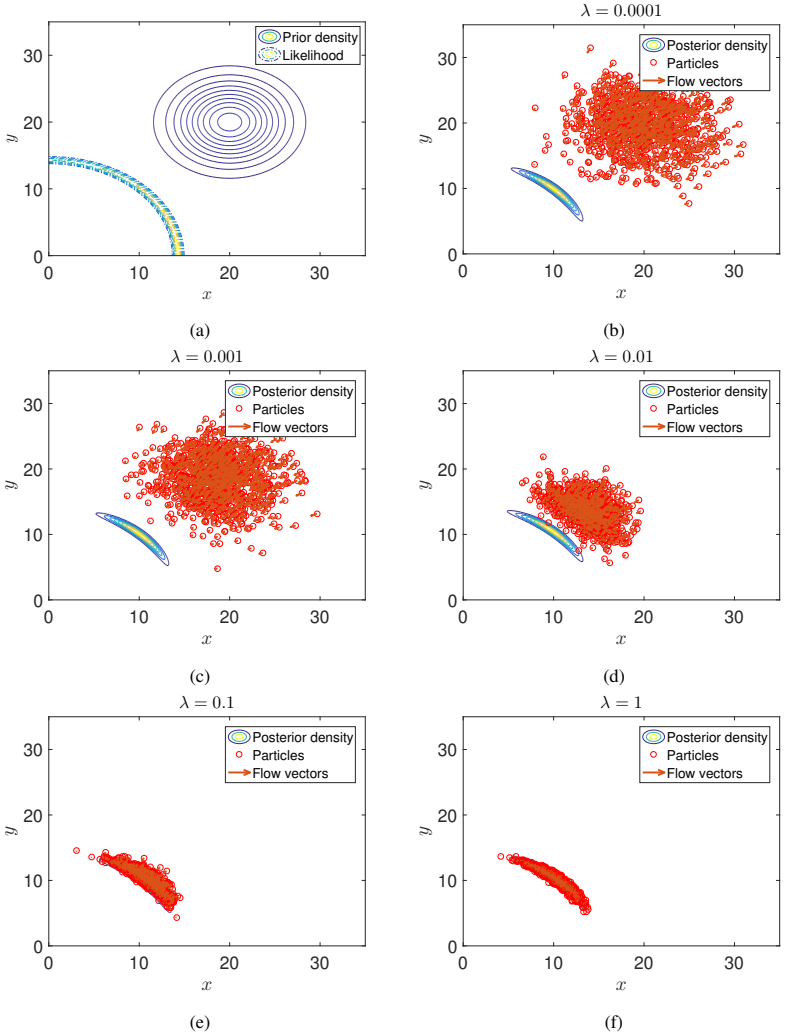


Figure 1.9: Prior density & likelihood (a) and the particle flow based posterior density approximations at different values of λ.

To illustrate the workings of the log homotopy based particle flow method, we show NZD flow based single measurement update step for a two dimensional state space system in Figure 1.9.

We use the following measurement model,

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ \tan^{-1}(\frac{y}{x}) \end{bmatrix} + \begin{bmatrix} v_r \\ v_\theta \end{bmatrix} \tag{1.1}$$

where $v_r \sim \mathcal{N}(0, 0.1)$ and $v_\theta \sim \mathcal{N}(0, 5)$ represent the additive measurement noises. The prior density is given by a Gaussian, $\mathcal{N}\left(\begin{pmatrix} 20 \\ 20 \end{pmatrix}, \begin{pmatrix} 15 & 0 \\ 0 & 15 \end{pmatrix}\right)$. We use 40 geometrically space pseudo-time λ points between 0 and 1. The main thing to note is that the high probability mass regions of the prior density and the likelihood are well separated, which makes the scenario very challenging from the particle degeneration aspect. Details on the exact procedure behind the log homotopy based particle flow will be given in the Chapter 3 of this thesis.

1.4 Thesis Scope and Contribution

This thesis achieves four objectives: it creates a better understanding of the main theoretical concepts behind the log-homotopy based particle flow filters in relation to similar approaches mentioned in the literature; a detailed analysis is performed to understand the shortcomings of the current implementation architecture of DHF and the suggestions of possible improvements; the derivation is provided of new algorithms based on these flows for the estimation of complex nonlinear/non-Gaussian problems and the extension of existing theoretical framework to cater for more general cases. Below, we enumerate and explain the individual contributions in detail.

1. The first objective is develop a thorough understanding of the main principles behind the log homotopy based particle flow method, as described by F. Daum and J. Huang in their series of papers. This includes the elaboration of the log-homotopic relationship between the probability densities in the context of Bayesian data assimilation, and the use of Fokker Planck equation to derive a generic particle flow equation. Furthermore, different flow solutions of the FPE are discussed in detail.

2. DHF implementations have been reported in several publications. While conceptually quite intuitive, DHF performance suffers in practice due to several assumptions made in the implementation. The second contribution of this thesis is in gaining better understanding of the implementation aspects of DHF. The objective is to carefully study the process in order to discern key steps, which if not properly implemented, could result in bad results. In the continuation of this, we identify most important factors affecting the performance of the DHF. As the continuation, a detailed study of those key factors has carried out. We individually discuss the possible options for each of them. Finally, we suggest possible improvements in the DHF implementation architecture to increase the efficiency.

3. Thirdly, we develop new filtering algorithms by combining the particle flow based measurement update with other methods.

   (a) In the first instance, we embed the particle flow based data assimilation step inside an SMCMC framework for the Bayesian estimation of massive sensor data . It refers to a scenario where a large number of measurements are available to be

processed at any given time instance. A straightforward application of SMCMC is often computationally quite expensive. Also, SMCMC needs a *well-suited* proposal distribution for achieving good performance in a reasonable amount of time. We realized that the DHF can be used to form a good proposal for sampling the posterior distribution in the case of massive sensor data. We build upon the existing work of De Freitas et.al. [FSM15], and combine a clustering driven DHF with the *Confidence sampling* based SMCMC to considerably improve upon the estimation accuracy of the later, while keeping the processing cost under a bearable limit. The new scheme performs considerably well against the increasing state dimensionality.

(b) The use of the tensor decomposition based representation for the probability densities and linear operators has been noted as a key factor in defeating the *Curse of dimensionality* in discretized problems. These concepts can be used to develop a framework for the propagation of a pdf through time, by numerically solving the Fokker-Planck equation (FPE). This has been discussed in [SK14], [SK15b], [SK15c] and [SK15d], where it has been highlighted that the discretized problem based on a tensor decomposition approach exhibits a much lower degree of freedom. This is essential to check the exponential growth of the number of free variables. Building on these works, we develop a *Continuous-Discrete* filtering algorithm, whereby we combine the log homotopy based flow with the tensor decomposition based solution of the FPE. We solve FPE twice, first in the continuous time propagation step of the RBE to form a prior density estimate, and the secondly together with the log homotopy based flow in the measurement update step in order to get the posterior density.

4. Lastly, we expand on the existing mathematical framework and derive new flow equations for sum of Gaussian based prior densities, and the filtering algorithms based on them.

In the following, we define the outline of the thesis.

Chapter 2

This is a survey chapter, in which both the traditional and progressive measurement update based solution for the Bayesian data assimilation are mentioned in detail. We start with the generic formulation of RBE for state space systems. Next, we describe the RBE solution for linear & Gaussian systems i.e. the Kalman filter. This is followed by the discussion on various traditional solution for nonlinear and/or non-Gaussian systems, namely, the Kalman filter based methods, Grid based methods, Sequential Monte Carlo or particle filters and SMCMC. Particularly, in the reference to the last two, we discuss the curse of dimensionality and the particle degeneracy. Later on, we describe a number of non-traditional methods for data assimilation based on the gradual inclusion of measurements. This becomes the major part of this chapter. Finally, we conclude the chapter by highlighting the strengths and weaknesses of these methods.

Chapter 3

This is the main chapter of this thesis. We start by expanding on the concepts of SDE and FPE. This is followed by the description of the log homotopy based particle flow method. We mainly follow the derivational guidelines mentioned by F.Daum and J.Huang in their papers.

We derive a generic particle flow equation and solve it under different assumption, leading to specific flow solutions. We describe each of the derived flow solutions in detail. Next, the typical implementation methodology for the DHF is described, and the key steps are highlighted. We also refer to our previous results where the sub standard performance of the DHF was noted. This becomes a strong motivation for looking into the ways to improve the implementation architecture of the DHF. In the proceeding section, the important factors in the DHF are studied in detail, with several possible options suggested for each of them. This culminates into our modified implementation termed as the *Improved DHF*. To test the performance of the new implementation, we consider state estimation of a complex nonlinear system, with both Gaussian and non-Gaussian measurement noises. The effect of different methods on the performance of DHF is studied individually for both noise cases. The main take away lesson from this chapter is by carefully designing a DHF, its performance can be substantially improved over more traditional implementations.

In the following, we list our publications related to this chapter.

[KU14] M. Altamash Khan and M. Ulmke. Non-linear and non-Gaussian state estimation using log-homotopy based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2014*, pages 1–6, October 2014

[KU15] M. Altamash Khan and M. Ulmke. Improvements in the Implementation of Log-Homotopy based Particle Flow Filters. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 74–81, July 2015

[KUK17] M. Altamash Khan, M. Ulmke, and W. Koch. Analysis of Log-Homotopy based Particle Flow Filters. *Journal of Advances in Information Fusion (JAIF)*, 12(1), June 2017

Chapter 4

In this chapter, we propose a novel approach for the Bayesian processing of the massive sensor data, by combining the log homotopy based particle flow with SMCMC. We propose an initial measurement clustering, after which the log-homotopy flow is applied. The samples after the flow are assumed to be approximately in the vicinity of their actual posterior locations, though not exactly there. Hence, they can form an excellent proposal to be used in the subsequent *Confidence sampling* driven MCMC procedure. The main purpose of the last step is to have the convergence guarantee, that comes associated with later procedure. In this way, we essentially bring the strength of both methods under one banner.
In this chapter, we start by revisiting the basics of SMCMC. Next, some important solutions for massive sensor data processing via SMCMC are highlighted. Probabilistic sub sampling i.e. confidence sampling methodology is identified as one such promising method, which we discuss in detail. We then move on to describing potential issues with the choice of proposal distribution in the context of SMCMC. The use of DHF together with the data clustering to form a better proposal is also advocated in the same section. Our DHF based new algorithm for the Bayesian processing of massive sensor data is mentioned in the next section. At the end, we test our algorithm in a challenging multi target tracking scenario using range & bearing measurements in the presence of strong clutter. We study the effect of different algorithm and

system parameters in detail. It is shown that our newly devised scheme outperforms the other nonlinear estimation schemes.

The related publications include an accepted conference paper and a planned journal paper, which will most likely be submitted to the *Journal of Advanced in Information Fusion* (JAIF).

[KdFL+17] M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Bayesian Processing of Big Data using Log Homotopy Based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2017*, pages 1–6, October 2017

[KdFL+18] M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Improving Adaptive Markov chain Monte Carlo using Log-Homotopy Particle Flow based proposal. To be submitted to Journal of Advances in Information Fusion (JAIF), 2018

Chapter 5

In this chapter, we explore another interesting approach to the solution of RBE i.e. the grid based methodology. In particular, we build on the tensor decomposition based framework, as developed by Y. Sun and M. Kumar in their series of papers, and combine it with the log homotopy based particle flow to devise a novel nonlinear filtering method. We employ the FPE not only to predict the probability density into future times, but also to perform the measurement update. We include measurements using the log-homotopy flow by solving the tensorized FPE w.r.t. the pseudo-time to get the estimated posterior density, hence completing one cycle of the Bayesian recursion.
Since the solution of FPE in higher dimensions entails solving a tensor equation, we start the main presentation by discussing the basics of tensors. Next, we present the basics of the separated representation of multi-dimensional functions and the tensor decomposition. Based on the notations and methodology developed, we study problems admitting stationary solutions and derive equations describing the FPE in the tensorized format. This is done to gain basic insights in the tensor decomposition based solution for FPE. Next, we provide details of the unified framework for the solution of the FPE, as described by Y. Sun and M. Kumar in their series of paper [SK14], [SK15b], [SK15c] and [SK15d]. We not only reproduce the details of the methodology as mentioned in these papers, but we also give a detailed derivation of the related equations. Based on this, we derive the tensor decomposition based nonlinear filtering algorithm, which we term as the *tensorized filter*. We study the effect of different algorithm parameters on its performance. We use optimally tuned *tensorized filter* to estimate a nonlinear scenario.

Publications related to this chapter are given below.

[KUD+16] M. Altamash Khan, M. Ulmke, B. Demissie, F. Govaers, and W. Koch. Combining Log-Homotopy Flow with Tensor decomposition based solution for Fokker-Planck equation. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 2229–2236, July 2016

[DKG16] B. Demissie, M. Altamash Khan, and F. Govaers. Nonlinear filter design using Fokker-Planck propagator in Kronecker tensor format. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 1–8, July 2016

Chapter 6

In the work leading to this chapter, we have been using the existing flow solutions to develop new filtering algorithms. In this chapter, we expand on the existing mathematical framework and start the work with a more general assumption in order to derive new flow equations. In particular, we use a sum of Gaussians based form for the prior density. We solve the FPE for the unknown flow, by assuming it to be of the same form as the *exact flow*. This leads to the derivation of two flow equations. This is a new theoretical development in the context of log homotopy based particle flow filters, and is one of our recent works. We use the newly derived flows to develop a new filtering algorithm. We test it for the estimation of scalar nonlinear and non-Gaussian example.

We have the following publication related to this chapter.

[KUK16] M. Altamash Khan, M. Ulmke, and W. Koch. A log homotopy based particle flow solution for mixture of Gaussian prior densities. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 546–551, September 2016

Chapter 7

This is the final chapter, in which we conclude the whole thesis. We also hint out directions for the possible extension of this work.

Chapter 2

Progressive measurement update methods

The Bayesian estimation framework offers an intuitive way for the estimation of hidden states of a dynamical system, based on the observational data. As discussed in the previous chapter, the Bayesian estimation is carried out recursively, typically consisting of a prediction and a correction step. A transition density describes the time evolution of the state conditioned on the previous values, while a measurement density describes the likelihood of measurements given the current state. These densities can then be used in the RBE framework for the evaluation of prior and posterior state distributions at any given moment of time.
This is a survey chapter, in which we look into the existing literature on nonlinear state estimation with the main focus on the progressive measurement update methods. We start by defining the general recursive Bayesian estimation in section 2.1. The optimal estimator for a linear Gaussian case is described in section 2.2. Next, in section 2.3, we start by describing a general nonlinear estimation problem and highlight the fact that the an optimal closed form estimator does not exist. Therefore, for the derivation of a practical nonlinear / non-Gaussian estimator, some level of approximation has to be made. Based on this fact, we describe the traditional solutions for nonlinear RBE in detail. This is followed by section 2.4, where we list some of the novel techniques for data assimilation in nonlinear systems based on the gradual inclusion of the measurements. Finally, the conclusion is given in section 2.5.

2.1 Bayesian recursive estimation

We start with the general formulation of Bayesian recursive estimation for a Markovian state space system. Let $\mathbf{x}_k \in \mathbb{R}^d$ denote the state vector and $\mathbf{z}_k \in \mathbb{R}^m$ denote the measurement vector at time k. Similarly, let $\mathbf{u}_k \in \mathbb{R}^n$ denote a control vector. The three vectors are related through the following set of general nonlinear and/or non-Gaussian recursive equations.

$$\begin{aligned} \mathbf{x}_{k+1} &= \boldsymbol{\phi}_{k+1}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{z}_{k+1} &= \boldsymbol{\psi}_{k+1}(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) \end{aligned} \tag{2.1}$$

where $\boldsymbol{\phi}_k$ is termed as the process / dynamical model and $\boldsymbol{\psi}_k$ as the measurement model. $\mathbf{w}_k$ and $\mathbf{v}_k$ are referred to as the process and the measurement noises, respectively. The expression 2.1 is general enough in the sense that the two noises can be statistically correlated, non-Gaussian or

multiplicative in nature, or all. The following diagram depicts a first order Markovian nonlinear state space system.

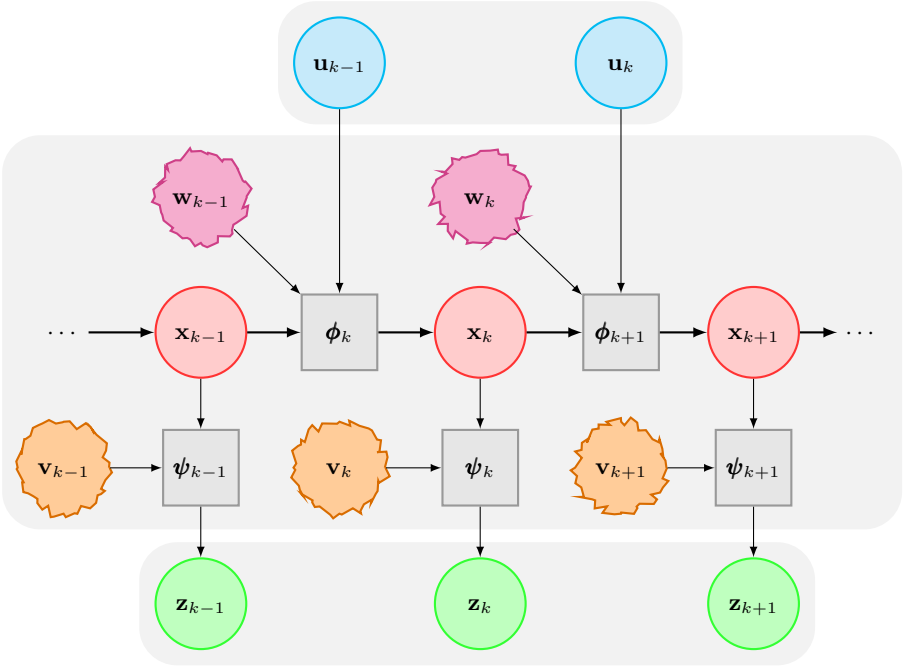


Figure 2.1: A non-linear Markovian state space system

The vector $\mathbf{u}_k$ represent the effect of the an external control input and is included here for the sake of completeness. Its presence is more important when studying the control behavior of dynamical systems. Since, the main focus of the discussion is limited to the state estimation $\mathbf{u}_k$ can be dropped, without the loss of any generality.

In an alternative formulation, the state space model can also be expressed in the terms of conditional probabilities,

$$\begin{aligned} \mathbf{x}_{k+1} &\sim p(\mathbf{x}_{k+1}|\mathbf{x}_k) \\ \mathbf{z}_{k+1} &\sim p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) \end{aligned}$$

$p(\mathbf{x}_{k+1}|\mathbf{x}_k)$ and $p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})$ are referred to as the transition and the measurement/likelihood densities. Assuming additive process and measurement noises w_k and v_k we can write

$$p(\mathbf{x}_{k+1}|\mathbf{x}_k) = p_{w_k}(\mathbf{x}_{k+1} - \phi_{k+1}(\mathbf{x}_k)) \tag{2.2}$$

$$p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) = p_{v_{k+1}}(\mathbf{z}_{k+1} - \psi_{k+1}(\mathbf{x}_{k+1})) \tag{2.3}$$

Furthermore, let $\mathbf{Z}_k$ denote the set of measurements up to time k including $\mathbf{z}_k$, such that, $\mathbf{Z}_k = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$. Then according to the Chapman-Kolmogorov equation and the Bayes

theorem, the prior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$ and the posterior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ are recursively defined as,

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_k) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_k)d\mathbf{x}_k \tag{2.4}$$

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) = \frac{p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{Z}_k)}{p(\mathbf{z}_{k+1}|\mathbf{Z}_k)} \tag{2.5}$$

where $p(\mathbf{x}_k|\mathbf{Z}_k)$ is posterior density at time *k*, also referred to as the filtering density. These are also referred to as the process and measurement update equations respectively. The conditional density $p(\mathbf{z}_{k+1}|\mathbf{Z}_k)$ appears as a normalization constant in the measurement update formula, and it describes the distribution of measurement at time *k+1*, conditioned on the set of all previous measurements. The Bayesian recursion starts with an initial density $p(\mathbf{x}_0)$, which is then propagated through time using (2.4) to yield $p(\mathbf{x}_1|\mathbf{Z}_0)$. At the arrival of the first measurement, the density is updated using the equation (2.5) which results in the first posterior density $p(\mathbf{x}_1|\mathbf{Z}_1)$. For the next time step, the posterior density becomes the input to the equation (2.4) resulting in the prior estimate for the time step 2, which is again followed by the measurement update step. This recursion can be continued as long as it is needed. The objective of any estimation algorithm is to solve the Bayesian recursion at each time step, in order to derive an exact or approximate form of the prior and posterior densities, which could be used in the calculation of the lower order central moments.

2.2 Linear Recursive Estimation

The Linear State Space (LSS) model can be derived from (2.1). The main assumption is that the state propagation and the measurements are linear functions of the state variable. If additive noise is assumed, the LSS model can be expressed as,

$$\mathbf{x}_{k+1} = \mathbf{F}_{k+1}\mathbf{x}_k + \mathbf{G}_{k+1}\mathbf{w}_{k+1} \tag{2.6}$$

$$\mathbf{z}_{k+1} = \mathbf{H}_{k+1}\mathbf{x}_{k+1} + \mathbf{v}_{k+1} \tag{2.7}$$

where $\mathbf{F}_{k+1}$, $\mathbf{H}_{k+1}$, $\mathbf{G}_{k+1}$ are the propagation, measurement and the noise gain matrices. It is assumed that the two noises are Gaussian in nature such that, $\mathbf{w}_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k+1})$ and $\mathbf{v}_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{k+1})$. To simplify matters, it is further assumed that the two noises are statistically uncorrelated i.e. $\mathbb{E}[\mathbf{w}_{k+1}\mathbf{v}_{k+1}^T] = \mathbf{0}$. If the initial state density is also Gaussian, then for such a linear Gaussian system, a finite dimensional analytical solution can be derived to solve the RBE i.e. (2.4) and (2.5). It is also known as the *Kalman Filter* or KF [Kal60], which has both the prior and the posterior densities given by Gaussians i.e. $p(\mathbf{x}_{k+1}|\mathbf{Z}_k) \sim \mathcal{N}(\hat{\mathbf{x}}_{k+1|k}, \mathbf{P}_{k+1|k})$ and $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) \sim \mathcal{N}(\hat{\mathbf{x}}_{k+1|k+1}, \mathbf{P}_{k+1|k+1})$. KF is an optimal estimator for linear Gaussian systems in the minimum mean square error (MMSE) sense, i.e there exists no estimator with a lower mean square error (MSE) [Gus13]. Filter dimensionality remains bounded through out the time, as Gaussian densities are completely described by their means and covariance matrices. The KF is described in the Algorithm 1.

Algorithm 1 Kalman Filter

1: Initialize the filter as $\hat{\mathbf{x}}_{0|0} = \mathbb{E}(\mathbf{x}_0)$ and $\mathbf{P}_{0|0} = \mathbb{E}[(\hat{\mathbf{x}}_{0|0} - \mathbf{x}_0)(\hat{\mathbf{x}}_{0|0} - \mathbf{x}_0)^T]$.
2: **for** k = 0 : $k_{max} - 1$ **do**

1. **Time Update step**

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{F}_{k+1}\hat{\mathbf{x}}_{k|k} \tag{2.8}$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_{k+1}\mathbf{P}_{k|k}\mathbf{F}_{k+1}^T + \mathbf{G}_{k+1}\mathbf{Q}_{k+1}\mathbf{G}_{k+1}^T \tag{2.9}$$

2. **Measurement Update step**

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T[\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}]^{-1} \\ \times (\mathbf{z}_{k+1} - \mathbf{H}_{k+1}\hat{\mathbf{x}}_{k+1|k})\end{aligned} \tag{2.10}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k}[\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}]^{-1}\mathbf{H}_{k+1}\mathbf{P}_{k+1|k} \tag{2.11}$$

3: **end for**

2.3 Traditional solutions to Non-Linear Recursive Estimation

Most of the actual real-world problems are nonlinear and/or non-Gaussian in nature. With the assumption of additive noise, such systems can be expressed through the following set of equations,

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}_{k+1}(\mathbf{x}_k) + \mathbf{w}_k \tag{2.12}$$

$$\mathbf{z}_{k+1} = \boldsymbol{\psi}_{k+1}(\mathbf{x}_{k+1}) + \mathbf{v}_{k+1} \tag{2.13}$$

The noise processes are assumed to have arbitrary distributions. Even if the initial state is assumed to be Gaussian, the state density at times greater than 0 will not remain so. That is because the state undergoes a nonlinear transformation every time the process update is carried out. Finite dimensional solutions exist in only few cases e.g. [Dau86], [KE97]. However, no exact solution (in the MMSE sense) are available for the state estimation of a general nonlinear/non-Gaussian system. Instead, approximate methods are used for the state estimation of such systems, resulting in sub-optimal algorithms. Most of such methods can be categorized into four main types.

2.3.1 Kalman filter based methods

These type of methods are based on the Kalman filter (KF) theory applied to nonlinear systems, with Extended Kalman Filter (EKF) being the simplest. Like KF, it also makes the Gaussian assumption for the transition density $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$ and the likelihood $p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})$. Model nonlinearities are dealt in a simple manner; approximating them by a linear or quadratic function based on the Taylor series expansion. Therefore, the time/process and measurement updates step look very similar to those of a KF, but with the presence of the corresponding Jacobian/Hessian terms. The approximation gives satisfactory results for problems where the nonlinearities in the process and measurement models are not too strong. While the EKF presents itself as a simple and intuitive solution, it fails to adequately approximate the posterior density when the degree

of nonlinearity is high, potentially resulting in the filter divergence. To compensate the drawbacks of EKF, other solutions have been proposed. This usually involves the selection of a set of deterministically sampled points to capture a number of low-order moments of the prior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$ in the best possible manner. These points are propagated through nonlinear functions (both transition and measurement). Point estimates are formed by using the transformed points together with the set of corresponding weights. Example of such methods include the Unscented Kalman Filter (UKF) [JUDW95] and the Cubature Kalman Filter (CKF) [AH09]. In the former, the sampled points are termed as the *Sigma-points* while in the later these are called the *Cubature points*. These methods are derivative free, since they do not involve the Taylor series approximations. In general, however these methods are also sub-optimal and their performance degrades with the increase in the nonlinearity, and also when the transition and measurement densities are non-Gaussian (e.g. multi modal, exponential).

2.3.2 Grid-based methods

The second class of estimators numerically approximate the prior and posterior densities over a discretized region of the state space. An example of this type is the so called *Point Mass Filter* or PMF [BL97], [SKS02], [SKS06]. Typically, in all implementations of PMF, the integrals in (2.4) and (2.5) are replaced with Riemann sums over a finite interval. The state space is divided into regions, each characterized by an indicator function (e.g. hypercubes) and the probability for each such function is calculated as being its weight. The posterior density is approximated as the sum of weighted and shifted indicators functions. Another approach to implement the PMF is using the frequency domain. It is assumed that the densities involved are band limited, and therefore can be appropriately sampled. The process update (2.4) can then be seen as a convolution integral; it results in a simple multiplication of the two spectra, while the measurement update can be carried out using multiplication in the time domain after taking the inverse Fast Fourier transform (iFFT). Another type of grid based method involves tensorization of the involved densities. The process update is then carried out by solving the Fokker Planck equation yielding in a prior density estimate [SK15b], [KCJ07]. In the measurement update step, the estimated density tensor is combined with the likelihood to form the posterior density estimate [KUD$^+$16], [SK15a].

The main disadvantage with grid based methods is unfavorable scaling of the number of grid points with increasing dimensionality i.e. more and more points are required to discretize the space as the number of state variables increase. Also, a uniform domain discretization can lead to an inefficient sampling. However, the tensorization approach has been shown to be able to thwart this ill effect to a certain extent in problems having special structure. For more details, please refer to Chapter 5.

2.3.3 Sequential Monte Carlo (SMC)

The third class of nonlinear/non-Gaussian estimators fills the category of Sequential Monte Carlo (SMC) methods, more famously known as the *particle filters*. [AMGC02], [GSS93].

Algorithm 2 Sequential Monte Carlo

1: Draw a set of particles $\{\hat{\mathbf{x}}_0^i\}_{i=1}^{N_p}$ from some distribution $p(\mathbf{x}_0)$ ▷ Initialize particles
2: **for** k = 0 : $k_{max} - 1$ **do**
3: $\hat{\mathbf{x}}_{k+1}^{(i)} \sim q(\mathbf{x}_{k+1}|\mathbf{x}_k^{(i)}, \mathbf{Z}_{k+1}) \qquad \forall i = 1, 2, \cdots, N_s$ ▷ Importance sampling
4: $\hat{\mathbf{x}}_{0:k+1}^{(i)} \cong \{\hat{\mathbf{x}}_{0:k}^{(i)}, \hat{\mathbf{x}}_{k+1}^{(i)}\}$
5: $w_{k+1}^{(i)} \propto w_k^{(i)} \frac{p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}^{(i)}) p(\hat{\mathbf{x}}_{k+1}^{(i)}|\mathbf{x}_k^{(i)})}{q(\hat{\mathbf{x}}_{k+1}^{(i)}|\hat{\mathbf{x}}_k^{(i)}, \mathbf{Z}_{k+1})}$ ▷ Weight calculation
6: $\hat{w}_{k+1}^{(i)} = w_{k+1}^{(i)} \left[\sum_{i=1}^{N_s} w_{k+1}^{(i)}\right]^{-1}$ ▷ Weight normalization
7: $N_{eff} = \left[\sum_{i=1}^{N_s} (\hat{w}_{k+1}^{(i)})^2\right]^{-1}$ ▷ Effective sample size
8: **if** $N_{eff} \leq$ Threshold **then** ▷ Re-sampling
- Resample $\{\mathbf{x}_{k+1}^{(i)}\}_{i=1}^{N_s}$ with replacement from $\{\hat{\mathbf{x}}_{k+1}^{(i)}\}_{i=1}^{N_s}$ according to $\hat{w}_{k+1}^{(i)}$
- Set $\{\hat{w}_{k+1}\}_{i=1}^{N_s} = \frac{1}{N_s}$

9: **end if**
10: **end for**

The main idea is to represent the posterior density by a weighted set of random samples (particles), which are then used to form point estimates, e.g., mean and variance [AMGC02]. No prior assumption is made for the type of the densities involved. In particle filters, the prior and posterior densities are recursively estimated by solving (2.4) and (2.5). In its most basic form, a particle filter is initialized with a set of particles, which are drawn from some initial distribution. State update is performed by sampling from an importance density $q(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{Z}_{k+1})$. On the arrival of measurements, particles are re-weighted according to their likelihood, which are then used to estimate the central moments like state mean and covariance. Finally, the most likely particles are replicated and assigned uniform weights, while the rest are discarded. This procedure is performed recursively, as shown in Algorithm 2. The posterior density under these settings approximately represents the path distribution, i.e., distribution of the state through the time, conditioned on the measurements.

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) \approx \sum_{i=1}^{N_s} \hat{w}_{k+1}^{(i)} \delta(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^{(i)}) \tag{2.14}$$

Several version of particle filters have been proposed in the literature, e.g., Sampling Importance Re-sampling filter also known as Bootstrap particle Filter (SIR-PF) [GSS93], Auxiliary Sampling Importance Re-sampling particle filter (ASIR-PF) [PS99], Regularized particle filter (RPF) [MOLG01] etc. While SMC based approaches can effectively deal with the system nonlinearities and non-Gaussian noises, they suffer from the two major problems: *Weight degeneracy* and the *Curse of dimensionality*.

2.3.3.1 Weight degeneracy

Weight degeneracyrefers to the fact that after few updates all but one particle have negligible weights. Weight degeneracy occurs when the posterior distribution does not significantly overlap with the prior distribution e.g. when the likelihood function is quite peaked and/or far from the region of significant probability mass of the prior density. This is shown in Figure 2.2. 1

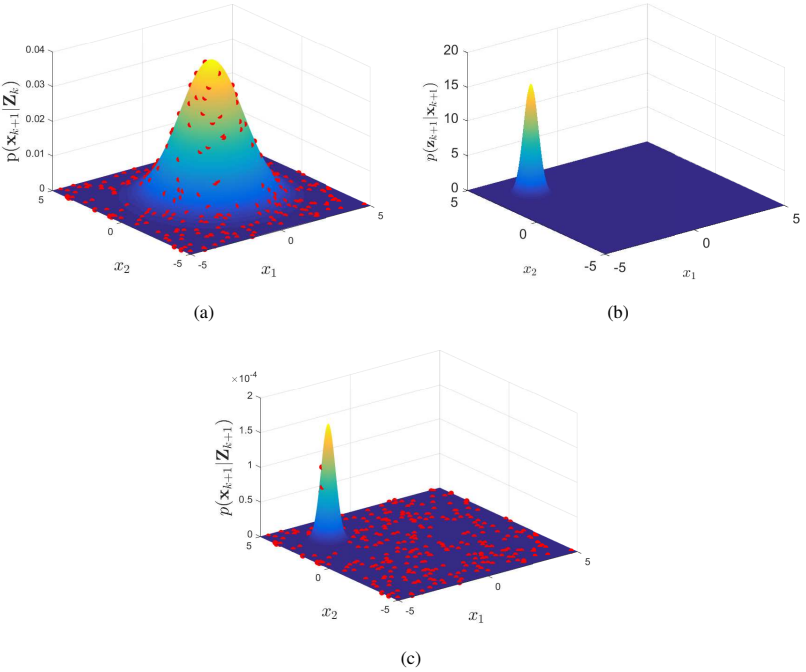


Figure 2.2: (a) Prior density, (b) likelihood and (c) the posterior density depicting a scenario with particle degeneration. Red dots represent sampled particles.

It can be seen that the likelihood is very sharp and only few samples have non-zero weights. For an SMC method, these have to be replicated to make up for the rest of the lot. As a result, the statistical diversity in the samples is lost and the particle set looses it ability to properly represent the posterior distribution.

2.3.3.2 Curse of dimensionality

The curse of dimensionality is the other issue faced by SMC methods.It refers to the geometric increase in the size of the discretized problem with the increase in the size of state dimensionality. In fact, this is one of the most severe issues faced by all numerical methods employed to solve higher dimensional problems. In the context of particle filtering, it means that to maintain a certain performance level, the required number of particles increases exponentially with the increase in the state dimension, as reported in [DH03] e.g. if 100 particles are required to

adequately represent a density in one dimensional space, the requirement will be 100×100 for representing a similar density in 2D space, $100 \times 100 \times 100$ for a 3D space and so on and so forth. This is depicted in the following figure.

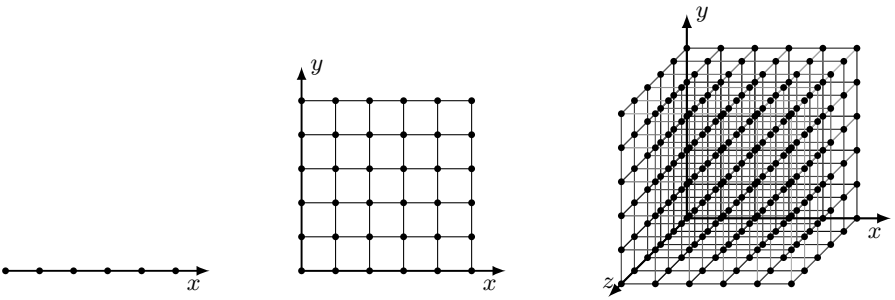


Figure 2.3: Increasing requirement for the number of discretized points with increasing dimension

It is more of an issue when the individual dimensions are independent, making the probability mass along each of them (marginal distribution) approximately the same. This point can be very well illustrated by comparing the volumes of a d-dimensional hyper-sphere to a hypercube of similar dimensions. If we take r as the radius of the hypersphere and $2r$ to be the length of each side of the hypercube, the ratio between their volumes R_d can be expressed as,

$$R_d = \frac{S_d(r)}{C_d(r)} = \frac{\frac{2^{-d}\pi^{\frac{d}{2}}r^d}{\Gamma(\frac{d}{2}+1)}}{r^d} = \frac{2^{-d}\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} \tag{2.15}$$

The ratio is plotted as a function of the state dimension d in Figure 2.4a. It can be noted that the ratio falls rather quickly as the number of dimensions is increased, in fact becoming quite insignificant for $d \approx 10$. This indicates that as the dimensionality increases, more and more volume gets concentrated around the surface of the hypercube. To understand the consequences of this result for the sampling of higher dimensional probability density, we consider the volume of a standard Gaussian multivariate density inside the radius of 1.65,

$$\text{Prob}(|X|_{\mu=\mathbf{0},\sigma=\mathbf{1}} \leq \alpha) = \text{erf}_d\left(\frac{\alpha}{\sqrt{2}}\right) \tag{2.16}$$

The exact solution for the above equation has been derived in detail in [M.B63]. Here, we just quote the results. For an even number of dimensions, erf_d is given by,

$$\text{erf}_{2m}(x) = 1 - e^{-x^2}\left(1 + \frac{x^2}{1!} + \frac{x^4}{2!} + \cdots + \frac{x^{2(m-1)}}{(m-1)!}\right) \tag{2.17}$$

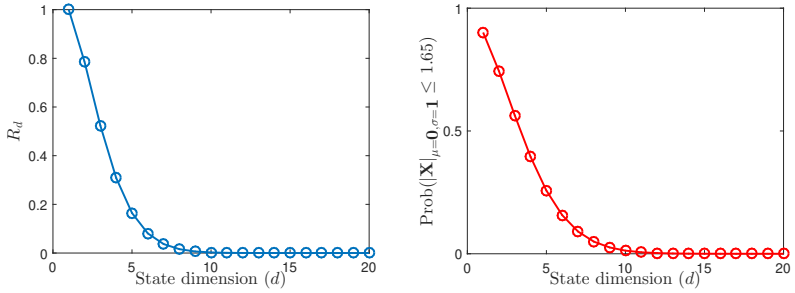
On the other hand, when the d is odd the formula is,

$$\text{erf}_{2m+1}(x) = \text{erf}_0 - e^{-x^2}\left(\frac{(2x)0!}{1!} + \frac{(2x)^3 1!}{3!} + \cdots + \frac{(2x)^{2m-1}(m-1)!}{(2m-1)!}\right) \tag{2.18}$$

with erf_0 being the standard error function for 1 dimension given by,

$$\text{erf}_0(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} dy \tag{2.19}$$

For a univariate case, $\text{Prob}(|X|_{\mu=\mathbf{0},\sigma=\mathbf{1}} \leq \alpha)$ is just around 90%. We plot the $\text{erf}_d(x)$ against d in Figure 2.4b.



(a) (b)

Figure 2.4: (a)Ratio of the volumes of a hyper-sphere to a hyper-cube in d-dimensions and (b) Volume of a d-dimensional standard Gaussian density within radius 1.65 vs. the state dimension d.

We see the trend is quite similar to the one exhibited by R_d. What this essentially means is that for higher dimensional Gaussian densities, most of the volume is concentrated in the tails and not near the center. The implication of this result is that the tails for densities in higher dimensions cannot be ignored when generating samples, thus increasing the required number of samples for the adequately representation. This also has consequences for the choice of the importance densities for state spaces with large number of variables. Another subtle point is that the curse of dimensionality and weight degeneracy become intertwined issues in higher dimensions, one complementing the other.

2.3.4 Sequential Markov chain Monte Carlo

Sequential Markov Chain Monte Carlo (SMCMC) based methods constitute yet another class of state estimation algorithms. Traditionally, MCMC has been employed in sampling from complex distributions occurring in statistical physics [MRR+53]. A Markov chain is created whose stationary distribution is the distribution of interest. New samples are generated using a proposal density, which can accepted or rejected. If accepted, the chain is said to have moved. This is referred to as the *Metropolis-Hastings* (MH) step. When applied in sampling from a posterior distribution in a sequential setting, such as in target tracking, the procedure is known as SMCMC. SMCMC methods provide a powerful tool for the state estimation of nonlinear / non-Gaussian system. Depending on the length of the constructed chain and the space exploration methodology, they can provide very accurate estimate of the posterior distribution. A typical SMCMC implementation is shown in the Algorithm 3,

Algorithm 3 Sequential Markov Chain Monte Carlo

Initialize the particle $\{\bar{\mathbf{x}}_0^i\}_{i=1}^{N_p}$
for k = 0 : $k_{max} - 1$ **do**
  Initialize the Markov chain: $\mathbf{x}_{k+1}^0$
  for m = 0 : $N_c + N_b$ **do**
    $\mathbf{x}_{k+1}^* \sim q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)$ ▷ Draw from the proposal distribution
    $u \sim \mathcal{U}[0,1]$ ▷ Uniform random draw
    $\beta(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) = \log\left[u \frac{p(\mathbf{x}_{k+1}^*|\mathbf{Z}_{k+1}) q(\mathbf{x}_{k+1}^m|\mathbf{x}_{k+1}^*)}{p(\mathbf{x}_{k+1}^m|\mathbf{Z}_{k+1}) q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)}\right]$ ▷ MH step
    if $\beta(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) > 0$ **then**
      $\mathbf{x}_{k+1}^m = \mathbf{x}_{k+1}^*$
    else
      $\mathbf{x}_{k+1}^m = \mathbf{x}_{k+1}^{m-1}$
    end if
  end for
  $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) \approx \frac{1}{N_c} \sum_{m=N_b+1}^{N_b+N_c} \delta(\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^m)$
end for

where $q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)$ is the proposal distribution, while the posterior distribution $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ is considered the target to be sampled. SMCMC can be used together with SMC, e.g. posterior distribution in the SMC can be used as the proposal in the SMCMC to further improve upon the accuracy. Alternatively, SMCMC step can be used within the SMC loop as a *resample-move* step to increase the statistical diversity [BG01]. However, these methods tend to be quite processing intensive, and therefore generally are not the first choice while choosing an estimator. Primarily this is due to three reasons: the difficulty in the choice of a right proposal density and evaluation of the target density itself. A bad proposal could result in the chain stuck in the regions of lower probability for most of the times. Hence, in order to increase the estimation accuracy, chains of longer lengths could become necessary. On the other hand, the evaluation of a complex target distribution at each iteration could also result in the greatly increased processing time e.g. in the case of extended object tracking or when multiple measurements are present. The main disadvantage is the evaluation of the likelihood term, which could be very expensive [FSM15]. Moreover, SMCMC also suffers from the *curse of dimensionality*.

2.4 Progressive measurement update based nonlinear data assimilation

It was noted in the previous section that SMC based methods are plagued by the problem of the particle degeneracy. As a possible remedy, it has been noted that the gradual inclusion of measurements could lead to a more equitable distribution of the particle weights. There are a number of ways in which this can be accomplished. It can either be done by sampling the particles from a set of *intermediate densities* [GC01], introducing the likelihood in a sequence and modeling the posterior by a Gaussian mixture model [HF03], [HJSM11] or moving the set of particles w.r.t. a fictitious time by solving an ordinary differential equation (ODE) [Rei13], [HDP15], [BG14], and [EMFD15]. A progression parameter controls the measurements inclusion rate in all such cases. As a result of gradual inclusion of measurements, the likelihood starting as a flat function, is continuously deformed until it gets into its actual form. This leads to a step by step formation of the posterior distribution estimate. It is hoped that by adopting this approach, the problem of degeneracy can be solved to a great extent.

In the rest of the chapter, we present an overview of the most commonly cited methods, paving the way for the introduction of the log homotopy based particle flow methods [DH07]-[DHN16], which is the main focus of this thesis as described in the subsequent chapters. Please note that only the details of the measurement update step are provided here for the described algorithms, while assuming the process update to be of generic nature.

2.4.1 Bridging densities

Degeneracy among the particles can happen either due to the separation of the areas of significant probability mass or limited number of samples used to represent the prior density, or both. To improve upon this, the use of MCMC step after the weight normalization (step 8, Algorithm 2) has been suggested in [GC01], with the posterior density as the stationary distribution of the chain. As noted by the authors, if the particles are already drawn correctly in the importance sampling step, a MCMC step will replace a good set of particles with an equally good set. If particles are far from their posterior locations, the application of MCMC step will move the particles closer to the target, thus improving on the accuracy. Instead of working on the filtering density, authors in [GC01] have described their method using the framework of smoothing density defined as following,

$$p(\mathbf{x}_{1:k+1}|\mathbf{Z}_{1:k+1}) \backsimeq \sum_{i=1}^{N_s} \hat{w}_{k+1}^{(i)} \delta(\mathbf{x}_{1:k+1} - \hat{\mathbf{x}}_{1:k+1}^{(i)}) \tag{2.20}$$

Its has however been noted that the required number of iterations in the MCMC stage can become quite large, leading to an increased processing overhead. Therefore, an alternative approach is presented where a number of intermediate distributions (L-1) are placed between the importance distribution $\pi_0 = q(\mathbf{x}_{1:k+1})$ and the posterior $\pi_L = p(\mathbf{x}_{1:k+1}|\mathbf{Z}_{1:k+1})$. The intermediate distributions are called *bridging densities* and defined as,

$$\pi_m(\mathbf{x}_{k+1}) \propto q(\mathbf{x}_{1:k+1})^{\lambda_l} p(\mathbf{x}_{1:k+1}|\mathbf{Z}_{1:k+1})^{1-\lambda_l} \tag{2.21}$$

where λ_l is the progression parameter with values such that $1 > \lambda_1 > \lambda_2 \cdots > \lambda_L > 0$. As an example, if the importance density is the same as in the SIR particle filter,

$$q(\mathbf{x}_{1:k+1}) = p(\mathbf{x}_{k+1}|\mathbf{x}_k) \sum_{i=1}^{N_s} \hat{w}_k^{(i)} \delta(\mathbf{x}_k - \hat{\mathbf{x}}_k^{(i)}) \tag{2.22}$$

then the intermediate densities can be written as,

$$\pi_l(\mathbf{x}_{k+1}) \propto p(\mathbf{x}_{1:k+1}|\mathbf{Z}_{1:k})p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})^{1-\lambda_l} \tag{2.23}$$

In the exact word used by authors, *the task now is to move the particle cloud through this sequence of densities by any available means*. In practical terms, particles are moved towards their posterior locations in steps. Starting with $\pi_0(\mathbf{x}_{k+1})$, $\pi_1(\mathbf{x}_{k+1})$ becomes the target distribution for the MCMC step at the first stage. The sampling process continues, such that for the *lth* intermediate step the target is $\pi_l(\mathbf{x}_{k+1})$. The corresponding weight update is given by,

$$\hat{w}_{k+1}^{(i)(l)} \propto \hat{w}_{k+1}^{(i)(l-1)} \frac{\pi_l(\mathbf{x}_{1:k+1}^{(i)(l-1)})}{\pi_{l-1}(\mathbf{x}_{1:k+1}^{(i)(l-1)})} \tag{2.24}$$

It can be hoped that by graduating in steps, the problem of a sudden transition will be resolved, with a small increase in the processing cost.

2.4.2 Progressive Bayesian update

A full description of a continuous state posterior probability density estimate for a nonlinear / non-Gaussian system could possibly require an infinite number of parameters. This of course is not practical and hence approximations are sought after. Gaussian mixture models (GMM) have been more commonly used in the literature to approximate an arbitrary probability density [AS72], [KD03]. The modeling accuracy can be controlled by varying the number of GMM components used in the approximation. Updating components individually could lead to a suboptimal result, while a joint update based on minimizing the distance between the true (posterior) density and its approximation is a tough optimization problem. The most favored solution is the Expectation Maximization (EM) method, but it could result in a local minimum. Also, the EM algorithm could exhibit slower convergence if the initialization is not done properly.
In [HF03], a new approach for Bayesian measurement update has been suggested. It relies on approximating the posterior density through a GMM, together with a gradual introduction of the measurements. The new approach is termed as the *Progressive Bayes*. The main idea is to update parameters of the mixture model in a way that the integrated squared error between the true density and its approximation is minimized. Since the true density can be multi-modal, an algorithm based on a simple minimization might require a large computational effort. In order to speed up the convergence while still keeping the processing cost manageable, the progressive approximations are made to the posterior density i.e. the true density is parameterized using a progression parameter λ, which can take values between 0 and 1. When λ is 0, the true density is just the prior density while the transformation is completed to the posterior for λ equals 1. A distance measure is defined between the parameterized true density $p(\mathbf{x}_{k+1}, \lambda|\mathbf{Z}_{k+1})$ and its approximation $\tilde{p}(\mathbf{x}_{k+1}, \boldsymbol{\theta}|\mathbf{Z}_{k+1})$,

$$D(\boldsymbol{\theta}, \lambda) = \frac{1}{2}\int_{\mathbb{R}} \Big(p(\mathbf{x}_{k+1}, \lambda|\mathbf{Z}_{k+1}) - \tilde{p}(\mathbf{x}_{k+1}, \boldsymbol{\theta}|\mathbf{Z}_{k+1})\Big)^2 dx \tag{2.25}$$

As a result, a set of intermediate approximations is made based on the discretization of λ. Since the approximated density is modeled through a GMM, it can be represented as,

$$\tilde{p}(\mathbf{x}_{k+1}, \boldsymbol{\theta}|\mathbf{Z}_{k+1}) = \sum_{i=1}^{M} \alpha^{(i)} \mathcal{N}(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_{(i)}, \boldsymbol{\Sigma}^{(i)}) \tag{2.26}$$

with $\boldsymbol{\theta}^{(i)} = [\alpha^{(i)}, \hat{\mathbf{x}}^{(i)}, \text{vec}(\boldsymbol{\Sigma}^{(i)})]^T$ and $\boldsymbol{\theta} = \left[(\boldsymbol{\theta}^{(1)})^T (\boldsymbol{\theta}^{(2)})^T \cdots (\boldsymbol{\theta}^{(L)})^T\right]^T$. Next, the problem is exactly converted into a system of explicit first order ODEs,

$$\mathbf{v}(\boldsymbol{\theta}, \lambda) = \mathbf{V}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.27}$$

where the coefficients are given by,

$$\mathbf{v}(\boldsymbol{\theta}, \lambda) = \int_{\mathbb{R}} \frac{\partial p(\mathbf{x}_{k+1}, \lambda | \mathbf{Z}_{k+1})}{\partial \lambda} \mathbf{P}(\mathbf{x}_{k+1}, \bar{\boldsymbol{\theta}}) d\mathbf{x}_{k+1} \tag{2.28}$$

and,

$$\mathbf{V}(\boldsymbol{\theta}) = \int_{\mathbb{R}} \mathbf{P}(\mathbf{x}_{k+1}, \bar{\boldsymbol{\theta}}) \mathbf{P}(\mathbf{x}_{k+1}, \bar{\boldsymbol{\theta}})^T d\mathbf{x}_{k+1} \tag{2.29}$$

where, $\bar{\boldsymbol{\theta}}$ is the nominal parameter vector used in the linearization of $\tilde{p}(\mathbf{x}_{k+1}, \boldsymbol{\theta} | \mathbf{Z}_{k+1})$ with the associated gradient,

$$\mathbf{P}(\mathbf{x}_{k+1}, \bar{\boldsymbol{\theta}}) = \frac{\partial}{\partial \bar{\boldsymbol{\theta}}} \tilde{p}(\mathbf{x}_{k+1}, \bar{\boldsymbol{\theta}} | \mathbf{Z}_{k+1}) \tag{2.30}$$

A numerical solution to (2.27) over a discretized domain is sought, which yields the updated parameters for any given value of λ. To make the algorithm more robust, structural adaptations are made by adding more Gaussian components to the GMM representation or merging two closely spaced ones.

While conceptually very interesting with providing a great deal of theoretical insight, an application of progressive Bayes for recursive estimation of a higher dimensional system could be computationally quite demanding. This is due to the discretization of a larger space, and also the higher number of components needed to accurately represent the posterior density. Furthermore, a peaked or tailed likelihood could make the situation worse as it would require a much finer resolution of the progression parameter λ.

2.4.3 Homotopy based optimal parameterization of the posterior density

In [HJSM11], an approach similar to the *Progressive Bayes* has been suggested. The authors have used the progressive measurement inclusion together with a homotopic relationship defined between the prior density and the likelihood, in a way that leads to the following form of the intermediate density,

$$p(\mathbf{x}_{k+1}, \lambda | \mathbf{Z}_{k+1}) = p(\mathbf{x}_{k+1} | \mathbf{Z}^k) p(\mathbf{z}_{k+1} | \mathbf{x}^{k+1})^{\lambda} \tag{2.31}$$

When λ is 0 $p(\mathbf{x}_{k+1}, \lambda | \mathbf{Z}_{k+1})$ is just the prior density, while for λ equal to 1 it corresponds to an unnormalized posterior density.

Also, in a way similar to Hanebeck et.al. [HF03], the intermediate density is approximated by a parameterized density $p(\mathbf{x}_{k+1}, \boldsymbol{\theta} | \mathbf{Z}_{k+1})$, which happens to be a GMM as well, at every λ step. A cost function is defined in terms of a distance measure, whose minimization yields the set of the optimal parameters. One difference to the former scheme is the use of measures other than the integrated squared error. Specifically, the use of Kullback-Leibler (KL) divergence and the squared Hellinger distance have been suggested. The KL distance between the intermediate and the parameterized densities is defined as:

$$D_H(\boldsymbol{\theta} || \lambda)^{\text{KL}} = \int p(\mathbf{x}_{k+1}, \boldsymbol{\theta} | \mathbf{Z}_{k+1}) \log \frac{p(\mathbf{x}_{k+1}, \boldsymbol{\theta} | \mathbf{Z}_{k+1})}{p(\mathbf{x}_{k+1}, \lambda | \mathbf{Z}_{k+1})} d\mathbf{x}_{k+1} \tag{2.32}$$

While this gives a measure of (dis)similarity between the two densities, it is not symmetric, and hence, not a true distance metric. The second distance metric is given as,

$$D_H(\boldsymbol{\theta},\lambda)^{\mathrm{H}^2} = \frac{1}{2}\int \left(\sqrt{p(\mathbf{x}_{k+1},\boldsymbol{\theta}|\mathbf{Z}_{k+1})} - \sqrt{p(\mathbf{x}_{k+1},\lambda|\mathbf{Z}_{k+1})}\right)^2 \mathbf{x}_{k+1} \tag{2.33}$$

Given any of the two measures $D_H(\boldsymbol{\theta},\lambda)^*$, the optimal set of parameters can be found by its minimization,

$$\boldsymbol{\theta}(\lambda) = \arg\min_{\theta}\ D_H(\boldsymbol{\theta},\lambda)^* \tag{2.34}$$

The density $p(\mathbf{x}_{k+1},\boldsymbol{\theta}|\mathbf{Z}_{k+1})$ modelled through a GMM, is supposed to consist of L components. Therefore, the parameter set of the approximated density consists of the weights, means and the covariance matrices i.e. $\{w_i,\hat{\mathbf{x}}_i,\Sigma_i\}_{i=1}^{L}$. A vectorization of this set yields the parameter vector $\boldsymbol{\theta}$, defined in terms of an ODE,

$$\boldsymbol{\theta}'(\lambda) = -\Big[\frac{\partial^2 D_H(\boldsymbol{\theta},\lambda)}{\partial\boldsymbol{\theta}^2}\Big]^{-1}\frac{\partial^2 D_H(\boldsymbol{\theta},\lambda)}{\partial\boldsymbol{\theta}\partial\lambda} \tag{2.35}$$

Equation (2.35) has been termed as the *homotopy differential equation*. Unless the prior and the likelihood are supposed to be Gaussians, (2.35) does not admit exact analytical solutions. Therefore, an approximated solution is sought whereby the integrals are solved approximately over a discretized domain Ω. This is done by replacing integrals with the Riemann sum and partial derivatives by first/second order differences. More over, a first order Euler's method has been suggested to update parameters over the λ, which is also discretized into N points. Applying the numerical approximation, (2.35) can be expressed as,

$$\bar{\boldsymbol{\theta}}_{\lambda_i} = \bar{\boldsymbol{\theta}}_{\lambda_{i-1}} - \mathbf{M}(\bar{\boldsymbol{\theta}}_{\lambda_i},\lambda_i)^{-1}\mathbf{n}(\bar{\boldsymbol{\theta}}_{\lambda_i},\lambda_i)\Delta\lambda_i \tag{2.36}$$

Starting from the $\bar{\boldsymbol{\theta}}_{\lambda_0} = \boldsymbol{\theta}_0$, the parameter vector representing the prior density, the corresponding posterior vector $\bar{\boldsymbol{\theta}}_{\lambda_1}$ is found by recursively solving (2.36) through $\lambda \in [0,1]$.
This method bears a great similarity to the idea of Hanebeck et.al., with the main notable differences being the explicit definition of homotopic relationship between the prior and the un-normalized posterior and the use of alternative distance measures. As a consequence, the parameter vector is updated by solving a first order ODE over a range of λ. The approach has been applied by the authors for the estimation of two dimensional problems involving linear and nonlinear measurement models. While the results are promising in terms of an increased accuracy, comparatively speaking, the homotopy based optimal parameterization of density is a rather expensive solution. As for the progressive Bayes, the requirement of an increased number of GMM components and larger/more finely grated solution domain for degenerated problems could increase the computational expenses. Whereas in the former case analytical expressions could be found for some of the integrals, the use of alternative distance metrics could render an exact solution impossible in the current case. This could add to the processing overhead. All of these issues could become more severe for the estimation in higher dimensions.

From now onwards, we simplify notations by dropping the time index k, and represent the prior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$ by $g(\mathbf{x})$ and the likelihood $p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})$ with the term $h(\mathbf{x})$. Any intermediate density in the progression will be represented with $p(\mathbf{x},\lambda)$ with λ being the progression parameter such that the posterior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ is given by $p(\mathbf{x},1)$. We further assume that the parameter λ is discretized into L values such that $\lambda_0 = 0 \leq \lambda_1 \leq \cdots \leq \lambda_L = 1$.

2.4.4 Guided sequential Monte Carlo

The sequential Monte Carlo family is very effective for the recursive estimation of nonlinear/ non-Gaussian problems, but they suffer from the curse of dimensionality as elaborated in the section 2.3.3.2. The main issue is that the importance sampling fails to adequately sample the regions of higher probability when the number of states gets sufficiently large. On the other hand, techniques like Ensemble Kalman Filter (EnKF) [Eve94], [HM98] have been successfully used in the geosciences community for higher dimensional Bayesian estimation, e.g. weather forecasting. However, this is achieved by assuming the Gaussian nature of the transition and the likelihood densities. In [Rei13], an attempt has been made to merge the two concepts into a single framework, termed as the *Guided Sequential Monte Carlo* or GSMC. The crux of this new idea is to gradually move the samples from the prior density towards the regions of higher probability mass of the posterior density, such that the re-sampling of particles is not needed. A continuous deformation of the prior density into the posterior is made via gradual introduction of measurements using a deterministic function, also called a transport map, until all the measurements have been included. In the case of the existence of such a continuous transformation, the prior and the posterior densities are related as follows,

$$p(\mathbf{x}, 1) = \frac{1}{|\mathcal{J}\nabla\psi(\mathbf{x})|} g(\mathbf{x}) \tag{2.37}$$

where $\mathcal{J}$ refers to the Jacobian matrix, formed by taking the spatial derivative of the transformed vector $\nabla\psi(\mathbf{x})$. $\nabla\psi(\mathbf{x})$ defines the transport map between the prior and the posterior density. Equation (2.37) is a nonlinear elliptical PDE, and its exact solution for a single dimensional case can be formulated in terms of cumulative density functions of the involved densities. However, for a general N-dimensional nonlinear/non-Gaussian problem, an analytical solution cannot be found. Therefore, approximations are made by introducing the idea of deterministic coupling between the prior and the posterior state variables. This is done dynamically, by describing the state transition through an ODE w.r.t a fictitious time parameter λ such that,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\lambda} = -\frac{1}{p(\mathbf{x}, \lambda)} \nabla\varrho(\mathbf{x}) \tag{2.38}$$

with the intermediate density $p(\mathbf{x}, \lambda)$ formed by linearly interpolating between the prior and the posterior densities,

$$p(\mathbf{x}, \lambda) = (1 - \lambda) g(\mathbf{x}) + \lambda \left(\frac{g(\mathbf{x}) h(\mathbf{x})}{\int g(\mathbf{x}) h(\mathbf{x}) \mathrm{d}\mathbf{x}} \right) \qquad \lambda \in [0, 1] \tag{2.39}$$

and an associated Poisson's equation involving potential ϱ,

$$\nabla \cdot (\nabla\varrho) = -g(\mathbf{x}) + \frac{g(\mathbf{x}) h(\mathbf{x})}{\int g(\mathbf{x}) h(\mathbf{x}) \mathrm{d}\mathbf{x}} \tag{2.40}$$

The desired map is a first order ODE, given by (2.38) which describes the gradual change of the state variable w.r.t. λ. Its solution yields the estimated state, after the assimilation of the measurement data. Remaining true to the original formulation of EnKF, a particle/ensemble based approach is used by the authors, i.e. prior and the posterior distributions are approximated by the set of weighted particles $\{(\bar{w}^{(i)}, \bar{\mathbf{x}}^{(i)})\}_{i=1}^{N}$ and $\{(\hat{w}^{(i)}, \hat{\mathbf{x}}^{(i)})\}_{i=1}^{N}$ respectively. It is shown

in [Rei11] that under the Gaussian approximation, (2.38) admits the following exact solution for the *ith* particle,

$$\frac{\mathrm{d}\mathbf{x}^{(i)}}{\mathrm{d}\lambda} = -\frac{1}{2}\bar{\mathbf{P}}\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{H}\mathbf{x}^{(i)} + \mathbf{H}\bar{\mathbf{x}} - 2\mathbf{z}\right) \tag{2.41}$$

where $\bar{\mathbf{x}}$, $\bar{\mathbf{P}}$ are the empirical mean vector and the prior covariance matrix respectively, $\mathbf{H}$ the measurement matrix, $\mathbf{R}$ the noise covariance matrix while $\mathbf{z}$ is the latest set of measurements. With the prior set of particles forming the initial condition, an iterative discretized solution of (2.41) can be formed,

$$\mathbf{x}^{(i)}_{\lambda_j} = \mathbf{x}^{(i)}_{\lambda_{j-1}} - \frac{1}{2}\mathbf{P}\mathbf{H}\mathbf{R}^{-1}\left(\left[\mathbf{H}\mathbf{x}^{(i)}_{\lambda_{j-1}} + \mathbf{H}\bar{\mathbf{x}} - 2\mathbf{z}\right]\Delta\lambda\right) \tag{2.42}$$

such that $\mathbf{x}^{(i)}_{\lambda_M} = \hat{\mathbf{x}}^{(i)}$. Since the Gaussian assumption was made while deriving (2.42), the approximate prior density is given by $\hat{g}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \bar{\mathbf{P}})$. This is strictly true if the system is linear and Gaussian. If either of these conditions is not true, the transport map gets suboptimal, which could result in mis sampling of the important state space regions. To counter this discrepancy, a weight correction step followed by the re-sampling might be required. Weight correction requires a better approximation of the prior density. While several choices might be available, in [Rei13] a Gaussian kernel density estimator (GKDE) is chosen,

$$\bar{g}(\mathbf{x}) \approx \sum_{i=1}^{N} \bar{w}^{(i)}\mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}^{(i)}, \mathrm{h}\mathbf{P}_N) \tag{2.43}$$

where $\mathbf{P}_N$ is the empirical covariance matrix with the bandwidth h. Given the set of prior particles (2.42) and a finer approximation of the prior density (2.43), the weight update can finally be given as,

$$\hat{w}^{(i)} \propto \bar{w}^{(i)} \times \frac{\hat{g}(\bar{\mathbf{x}}^{(i)})}{\hat{g}(\hat{\mathbf{x}}^{(i)})} \times \frac{\bar{g}(\hat{\mathbf{x}}^{(i)})}{\bar{g}(\bar{\mathbf{x}}^{(i)})} \tag{2.44}$$

While quite intuitive, the GSMC based particle flow is based on the Gaussian assumption. For cases where the prior density and/or likelihood cannot be modeled sufficiently well by Gaussians, this flow could struggle in accurately moving the particles to their correct posterior locations. Also, the choice of the bandwidth parameter h could be tricky, as it might require heuristic tuning.

2.4.5 Gibbs transport particle flow

Another interesting approach for deriving an approximate transport map has been reported in [HDP15], which is based on conditional distributions of the intermediate density. One major contribution of the work is the derivation of the flow ODE which does not require solving any PDE. For the one dimensional case, the flow ODE is given as,

$$\tilde{f}(x,\lambda) = \frac{\gamma'_\lambda}{g(x)h(x)^{\gamma_\lambda}}\left[\mathrm{F}_\lambda(x)\int_{-\infty}^{\infty}\log h(y)g(y)h(y)^{\gamma_\lambda}dy - \int_{-\infty}^{x}\log h(y)g(y)h(y)^{\gamma_\lambda}dy\right] \tag{2.45}$$

where F_λ is the cumulative distribution function (CDF). The likelihood is introduced into the flow at the rate γ'_λ. For dimensions greater than one, a straightforward extension of (2.45) does

not work. This is due to the fact that the flow thus derived does not obey the *vanishing* property, i.e the flow does not vanish in the tails. However, the regularization helps to resolves the issue. Flow equation in higher dimension requires the evaluation of multi-dimensional integrals, which due to the curse of dimensionality becomes progressively harder to approximate. The task becomes somehow easier if the intermediate density can be factorized as $p(\mathbf{x}, \lambda) = \prod_{l=1}^{d} p(x_i, \lambda)$. However, as for most of the examples encountered in the real world, this is a rather strong assumption. In order to get around this problem, an approximate flow equation is derived that is based on conditioned densities as opposed to the joint distribution. Hence, the flow is termed as the *Gibbs flow* or GbF. The flow ODE for the *Gibbs flow* is described as,

$$\begin{aligned} \tilde{f}_l(\mathbf{x}, \lambda) &= \frac{\gamma'_\lambda}{g(\mathbf{x})h(\mathbf{x})^{\gamma_\lambda}} \left[\mathrm{F}_\lambda(x_l|\mathbf{x}_{-l}) \int_{-\infty}^{\infty} \log h(y_l, \mathbf{x}_{-l}) g(y_l, \mathbf{x}_{-l}) h(y_l, \mathbf{x}_{-l})^{\gamma_\lambda} dy_l \right. \\ &- \left. \int_{-\infty}^{x_l} \log h(y_l, \mathbf{x}_{-l}) g(y_l, \mathbf{x}_{-l}) h(y_l, \mathbf{x}_{-l})^{\gamma_\lambda} dy_l \right] \end{aligned} \tag{2.46}$$

Notationally, $p(y_i, \mathbf{x}_{-i})$ means $p(x_1, x_2, \cdots, x_{l-1}, y_l, x_{l+1}, \cdots, x_d)$, while $\mathrm{F}_\lambda(x_l|\mathbf{x}_{-l})$ represents the conditional CDF. It can be noted that (2.46) requires the evaluation of one dimensional integrals, which are relatively easy to compute. The term γ_λ, also referred to as the temperature function, is chosen to minimize the integrated absolute squared error.
For the numerical implementation of the Gibbs flow, an appropriate λ schedule needs to be chosen and the integrals involved in (2.46) be approximated. The authors suggest the use of Newton-Cotes quadrature for the latter. Particles are sampled from the prior density. Once the integrals in the flow equations are approximated, the flow ODE for the *ith* particle is solved using the Euler's method,

$$\mathbf{x}_{\lambda_j}^{(i)} = \mathbf{x}_{\lambda_{j-1}}^{(i)} + \mathbf{f}(\mathbf{x}_{\lambda_{j-1}^{(i)}}, \lambda_{j-1}) \Delta\lambda_j = \mathbf{\Phi}_{\lambda_j}(\mathbf{x}_{\lambda_{j-1}}^{(i)}) \tag{2.47}$$

which can also be described as $\mathbf{x}_{\lambda_j}^{(i)} = \mathbf{\Psi}_{\lambda_j}(\mathbf{x}_0^{(i)}) = \mathbf{\Phi}_{\lambda_j} \circ \mathbf{\Phi}_{\lambda_{j-1}} \circ \cdots . \mathbf{\Phi}_{\lambda_1}(\mathbf{x}_0^{(i)})$. Using this construction, it is possible to relate the intermediate density with the prior density in the following way,

$$p(\mathbf{x}_{\lambda_j}^{(i)}, \lambda_j) = g(\mathbf{\Psi}_{\lambda_j}^{-1}(\mathbf{x}_{\lambda_j}^{(i)})) \left| \det \mathcal{J}_{\mathbf{\Psi}_{\lambda_j}}(\mathbf{x}_{\lambda_j}^{(i)}) \right| \tag{2.48}$$

where $\mathcal{J}_{\mathbf{\Psi}_{\lambda_j}}$ is the Jacobian matrix of the transport maps $\mathbf{\Psi}_{\lambda_j}(\mathbf{x}_0^{(i)})$ and needs to be estimated on the run. Since approximations are used for the flow equation, its Jacobian is also dependent on the accuracy of the estimates. Finally, weights are calculated according to the following equation,

$$w_{\lambda_j}^{(i)} \propto w_{\lambda_{j-1}}^{(i)} \times \frac{1}{\left| \det \mathcal{J}_{\mathbf{\Psi}_{\lambda_{j-1}}}(\mathbf{x}_{\lambda_{j-1}}^{(i)}) \right|} \times \frac{g(\mathbf{x}_{\lambda_j}^{(i)}) h(\mathbf{x}_{\lambda_j}^{(i)})^{\lambda_j}}{g(\mathbf{x}_{\lambda_{j-1}}^{(i)}) h(\mathbf{x}_{\lambda_{j-1}}^{(i)})^{\lambda_{j-1}}} \tag{2.49}$$

Conceptually, the Gibbs flow is very sound as it is based on the *vanishing* property, which ensures that the flow is always convergent. Also, in the light of the studied numerical examples, the authors have shown the flow to work reasonably well for non-Gaussian distributions featuring

multi-modality. However, there are a couple of points to be noted. First, the Gibbs flow is based on estimation of the flow components individually along each dimension, which ignores the possible inter-coupling. This is the result of a simplification of the more complex flow, which requires the evaluation of multi-dimensional integrals (see section 3.3 of [HDP15]). This raises questions about the applicability of GbF for the scenarios with strongly coupled/correlated dimensions. Secondly, the integrals involved in the flow equations need to be numerically approximated. With the increasing number of dimensions, this could become prohibitively expansive even if Monte Carlo schemes are used for the approximation. Yet other possible sources of the performance degradation could come from the use of Euler's scheme for the numerical integration and the subsequent evaluation of the Jacobian matrix required for the weight update step. Unless the posterior density retains a conditionally independent structure, the overall measurement update procedure could be quite time consuming and prone to numerical errors.

2.4.6 Gaussian particle flow with importance sampling

Authors in [BG13], [BG16] and [BG14] mention that although particle flow and optimal transport methods developed in [Rei11]- [HDP15] are theoretically quite elegant, their performance suffers from a series of approximations made during the practical implementation. These include approximations made while deriving the flow, approximation made for the prior density and the numerical integration of the flow ODE. These could lead to the introduction of bias and loss of asymptotic consistency. Authors in [BG14] have derived an alternative formulation of progressive measurement update, termed as the *Gaussian particle flow with Importance sampling* or GFIS. This approach combines the particle flow concept with that of the importance sampling. This main derivation starts from a linear Gaussian case, and is modified to be used for the nonlinear/ non-Gaussian models. Numerical integration is not required as the flow equation can be used to sample the state at any discrete moment in the pseudo-time. Also, the processed particles are not directly used in the formation of posterior point estimates (e.g. mean, covariance), instead, they are considered samples coming from an importance density. Therefore, particle weights are calculated which theoretically allows for the corrections of the errors incurred due to the approximations made. Another highlight of GFIS is the use of both deterministic and stochastic terms within the flow equation, essentially making it a stochastic differential equation (SDE), instead of an ODE.
Starting with a linear Gaussian model, the intermediate density is given exactly as $p(\mathbf{x}, \lambda) = \mathcal{N}(\mathbf{x}_\lambda | \mathbf{m}_\lambda, \mathbf{P}_\lambda)$ such that,

$$\begin{aligned} \mathbf{P}_\lambda &= \left[\mathbf{P}_0^{-1} + \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\right]^{-1} \\ \mathbf{m}_\lambda &= \mathbf{P}_\lambda \left[\mathbf{P}_0^{-1} \mathbf{m}_0 + \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{z}\right]^{-1} \end{aligned} \tag{2.50}$$

where $\mathbf{m}_0$ and $\mathbf{P}_0$ represent the prior mean and covariance respectively. Rest of the terms have the same meaning as in the description of the Kalman filter, described in the section 2.2. Actually, these equations (2.50) are similar to the Kalman filter measurement update equations, given in the information fusion form. The variable $\mathbf{x}_\lambda$ is defined as a Orstein-Uhlenbeck (OU) process with $p(\mathbf{x}, \lambda)$ being its stationary density. Therefore, it can be sampled and its mean and covariance matrix be determined for any value of λ. An SDE defines the progressive change for the OU process,

$$d\mathbf{x}_\lambda = \mathbf{f}(\mathbf{x}, \lambda) d\lambda + \boldsymbol{\sigma}(\mathbf{x}, \lambda) d\boldsymbol{\epsilon}_\lambda \tag{2.51}$$

Here $\mathbf{f}(\mathbf{x}, \lambda)$ represents the deterministic drift term, while $\boldsymbol{\epsilon}_\lambda$ is the standard Wiener process with intensity $\boldsymbol{\sigma}(\mathbf{x}, \lambda)$ and is called the diffusion term, characterizing the stochastic part of the flow. The main task is to derive expressions for the drift and the diffusion terms and solve the SDE recursively.

When the measurement function ψ is nonlinear, Taylor series based approximations are made such that $\hat{\mathbf{H}}_\lambda = \frac{\partial \psi}{\partial \mathbf{x}}\big|_{x_\lambda}$ and $\mathbf{z}_\lambda \approx \mathbf{z} - \psi(\mathbf{x}_\lambda) + \hat{\mathbf{H}}_\lambda \mathbf{x}_\lambda$. Since a numerical solution is sought after, λ is discretized into non-overlapping intervals and SDE (2.51) is solved for each such interval. The flow starts with the $\hat{\mathbf{m}}_{\lambda_0} = \mathbf{m}_0$ and $\hat{\mathbf{P}}_{\lambda_0} = \mathbf{P}_0$, and continues to sample the approximate Gaussian intermediate density for the *ith* particle. The updates for the state (drift and diffusion terms) , mean and the covariance matrices during the pseudo-time interval $[\lambda_i, \lambda_{i-1}]$ are given below,

$$\begin{aligned}
\hat{\mathbf{m}}^{(i)}_{\lambda_j} &= \hat{\mathbf{m}}^{(i)}_{\lambda_{j-1}} + \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} \hat{\mathbf{H}}^T_{\lambda_{j-1}} \Big(\hat{\mathbf{H}}_{\lambda_{j-1}} \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} \hat{\mathbf{H}}^T_{\lambda_{j-1}} + \frac{\mathbf{R}}{\lambda_j - \lambda_{j-1}} \Big)^{-1} \\
&\times \Big(\hat{\mathbf{z}}_{\lambda_{j-1}} - \hat{\mathbf{H}}_{\lambda_{j-1}} \hat{\mathbf{m}}_{\lambda_{j-1}} \Big) \\
\hat{\mathbf{P}}^{(i)}_{\lambda_j} &= \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} - \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} \hat{\mathbf{H}}^T_{\lambda_{j-1}} \Big(\hat{\mathbf{H}}_{\lambda_{j-1}} \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} \hat{\mathbf{H}}^T_{\lambda_{j-1}} + \frac{\mathbf{R}}{\lambda_j - \lambda_{j-1}} \Big)^{-1} \hat{\mathbf{H}}_{\lambda_{j-1}} \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}} \\
\hat{\mathbf{x}}^{(i)}_{\lambda_j} &= \hat{\mathbf{x}}^{(i)}_{\lambda_{j-1}} + \boldsymbol{\Gamma}_{\lambda_{j-1}, \lambda_j} \Big(\hat{\mathbf{x}}^{(i)}_{\lambda_{j-1}} - \hat{\mathbf{m}}^{(i)}_{\lambda_{j-1}} \Big) + \boldsymbol{\Omega}_{\lambda_{j-1}, \lambda_j}{}^{\frac{1}{2}} \boldsymbol{\xi}_{\lambda_{j-1}, \lambda_j}
\end{aligned} \tag{2.52}$$

where $\boldsymbol{\Gamma}_{\lambda_{j-1}, \lambda_j}, \boldsymbol{\Omega}_{\lambda_{j-1}, \lambda_j}$ and $\boldsymbol{\xi}_{\lambda_{j-1}, \lambda_j}$ are given by,

$$\begin{aligned}
\boldsymbol{\Gamma}_{\lambda_{j-1}, \lambda_j} &= \exp\Big\{ -\frac{1}{2} \gamma (\lambda_j - \lambda_{j-1}) \Big\} (\hat{\mathbf{P}}^{(i)}_{\lambda_j})^{\frac{1}{2}} (\hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}})^{-\frac{1}{2}} \\
\boldsymbol{\Omega}_{\lambda_{j-1}, \lambda_j} &= \Big[1 - \exp\Big\{ \gamma (\lambda_j - \lambda_{j-1}) \Big\} \Big] \hat{\mathbf{P}}^{(i)}_{\lambda_i} \\
\boldsymbol{\xi}_{\lambda_{j-1}, \lambda_i} &= \frac{\int_{\lambda_{j-1}}^{\lambda_j} \gamma^{\frac{1}{2}} \exp\Big\{ -\frac{1}{2} \gamma (\lambda_j - \lambda_{j-1}) \Big\} d\boldsymbol{\epsilon}_\lambda}{\Big[1 - \exp\Big\{ \gamma (\lambda_i - \lambda_{j-1}) \Big\} \Big]^{\frac{1}{2}}} \sim \mathcal{N}(.|0, I)
\end{aligned}$$

γ is the scaling parameter for OU process and it controls the behavior of the system; flow is completely deterministic for γ = 0 and stochastic otherwise. λ can be discretized using a constant or adaptive step size. If the prior density and the likelihood are non-Gaussian, the flow equations can still be used. This is done by using the Laplace approximation which fits a Gaussian distribution to an arbitrary density at one of the modes. Since the flow involves approximations for the measurement model and/or the noise density, the particles are not considered to be samples from the posterior distribution, and hence not used directly while forming the point estimates. Instead, it is thought as if the particle stream is coming out from an importance sampler. Hence, the weight correction is needed. The weight update formulae for the deterministic and stochastic cases are respectively given as,

$$\begin{aligned}
w^{(i)}_{\lambda_j} &\propto w^{(i)}_{\lambda_{j-1}} \times \frac{p(\mathbf{x}^{(i)}_{\lambda_j}) h(\mathbf{x}^{(i)}_{\lambda_j})^{\lambda_j}}{p(\mathbf{x}^{(i)}_{\lambda_{j-1}}) h(\mathbf{x}^{(i)}_{\lambda_{j-1}})^{\lambda_{j-1}}} \times \sqrt{\frac{|\hat{\mathbf{P}}_{\lambda_j^{(i)}}|}{|\hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}}|}} \\
w^{(i)}_{\lambda_j} &\propto w^{(i)}_{\lambda_{j-1}} \times \frac{p(\mathbf{x}^{(i)}_{\lambda_j}) h(\mathbf{x}^{(i)}_{\lambda_j})^{\lambda_j}}{p(\mathbf{x}^{(i)}_{\lambda_{j-1}}) h(\mathbf{x}^{(i)}_{\lambda_{i-1}})^{\lambda_{j-1}}} \times \frac{\mathcal{N}(\mathbf{x}^{(i)}_{\lambda_{j-1}} | \hat{\mathbf{m}}^{(i)}_{\lambda_{j-1}}, \hat{\mathbf{P}}^{(i)}_{\lambda_{j-1}})}{\mathcal{N}(\mathbf{x}^{(i)}_{\lambda_j} | \hat{\mathbf{m}}^{(i)}_{\lambda_j}, \hat{\mathbf{P}}^{(i)}_{\lambda_j})}
\end{aligned} \tag{2.53}$$

Algorithm performance improves with the reduction in the λ step size, which can also be chosen adaptively. Resampling might be required after the particles have been flown to their final locations, i.e at λ=1. Resampling with replacement from a set of finite particles introduces dependencies amongst particles. In order to mitigate this effect, a MCMC step might be used to move particles independent of each other, nudging them towards the posterior density, i.e. a resample-move step as described in [BG01]. Authors in [BG14] have used the GFIS particle flow to approximate Optimal Importance Density within the standard particle filtering framework.
The GFIS framework is based on extending a flow, which is exact for a linear Gaussian system. The extension entails linearization of the observation model as well as the Laplace approximation for non-Gaussian densities. Stochasticity has been added to sample the intermediate densities in an efficient manner. However, as shown in [LZC15], the performance of a filter based on GFIS could suffer in practice. This could be due to several approximation made while deriving the flow. The inclusion of the stochastic term is also a matter of uncertain utility. Although it could lead to a more diverse set of samples, but since the flow is only approximate, it is possible that a flow without the noise term could outperform the one having it, as amply demonstrated in numerical examples provided in [BG14].

2.4.7 Stochastic particle flow

In [EMFD15], a new method type for particle flow has been presented. The main inspiration for the work comes from two sources: the GFIS particle flow [BG14] and the Riemann Manifold Metropolis adjusted Langevin algorithm *mMALA* [GCC11]. The first influence comes through the use of the stochastic term in the particle flow. Also, similar to [BG14], the particles are weighted to form posterior density estimate. However, weighting is carried out at the end of the flow, as opposed to being done recursively through out the λ loop. The use of tensor metric associated with Riemann manifold, as done in *mMALA*, seems to be the second major source of inspiration. The main intuition behind the *mMALA* is the exploitation of the local structure of the target density when proposing new samples to improve the overall mixing of the chain in a MCMC setting. By merging the two concepts, authors in [EMFD15] have devised a new flow, termed as the *Stochastic Particle flow* or SPF. Below, we briefly summarize the key aspects of SPF.
A set of particles $\{\mathbf{x}_{\lambda}^{(i)} \in \mathcal{R}^d : \mathrm{i} = 1, \cdots, \mathrm{N}\}$ dependent on continuous pseudo-time variable $\lambda \in [0, \infty)$ are assumed such that $\mathbf{x}_0^{(j)} = \bar{\mathbf{x}}_{k+1}^{(i)}$ and $\mathbf{x}_{\infty}^{(i)} = \hat{\mathbf{x}}_{k+1}^{(j)}$. Please note that the λ here does not carry the same notation as in all of the previously described methods. Though still a synthetic time, it does not define an exact homotopic relation between the prior and the posterior density. Given this, the state dynamics of the particles obey the Itô SDE similar to (2.51). The main task becomes to find expressions for the drift and diffusion components of the flow. Based on the argument of the vanishing probability current for a stationary Fokker-Planck equation, the following form of the drift vector is derived,

$$\mathbf{f}(\mathbf{x}, \lambda) = \frac{1}{2}\mathbf{D}(\lambda)\nabla \log p(\mathbf{x}, \lambda) \tag{2.54}$$

The SDE governing the state transition can thus be written as,

$$d\mathbf{x} = \frac{1}{2}\mathbf{D}(\lambda)\nabla \log p(\mathbf{x}, \lambda)d\lambda + \mathbf{D}^{\frac{1}{2}}(\lambda)d\boldsymbol{\epsilon}_{\lambda} \tag{2.55}$$

with the yet unknown diffusion matrix $\mathbf{D}(\lambda)$. In [GCC11], the discretized stochastic Langevin equation for a flat Riemann manifold with a constant curvature has the diffusion matrix given by the inverse of a position dependent tensor $\mathbf{G}(\mathbf{x})$. In the context of Bayesian estimation, the tensor is set to be the observed FIM as $\mathbf{G}(\mathbf{x}(\lambda)) = -\mathcal{H}_x \left[\log p(\mathbf{x})\right]$, where $\mathcal{H}_x$ represents the Hessian matrix. This results in the following form of the locally approximated diffusion matrix,

$$\mathbf{D}(\lambda) = \left[-\mathcal{H}_x \left[\log g(\mathbf{x}) + \log h(\mathbf{x})\right]\right]^{-1}_{\mathbf{x}=\mathbf{x}_\lambda} \tag{2.56}$$

Next, an integration method is devised for solving the SDE in (2.55), which is based on Ozaki's discretization scheme. As λ is not restricted to the interval $[0, 1]$ and can have any positive real value, the selection of a proper integration time horizon T and the step size $\Delta\lambda$ is critical for the performance. This requires a priori knowledge of certain parameters, which in turn are functions of the maximum a posteriori estimate that has to be estimated before the particle flow is initiated. As a result, the λ is discretized into L distinct values such that $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_{L-1} < T$. Unlike [BG14] where a Gaussian density approximates the intermediate densities, here $p(\mathbf{x}, \lambda_j)$ is approximated through a Gaussian mixture where one component is associated with each particle.

$$p(\mathbf{x}, \lambda_j) = \sum_{i=1}^{N_p} w^{(i)} \mathcal{N}(\mathbf{x}_{\lambda_j} | \mathbf{m}^{(i)}_{\lambda_j}, \mathbf{P}^{(i)}_{\lambda_j}) \tag{2.57}$$

Equations for the state, mean and covariance matrix update for each component are shown below,

$$\mathbf{x}^{(i)}_{\lambda_j} = \mathbf{x}^{(i)}_{\lambda_{j-1}} + \frac{1}{2} \int_{\lambda_{j-1}}^{\lambda_j} \mathbf{D}(\mathbf{x}^{(i)}_{\lambda_{j-1}}) \nabla \log p(\mathbf{x}^{(i)}_{\lambda_{j-1}}) d\lambda + \int_{\lambda_{j-1}}^{\lambda_j} \mathbf{D}^{\frac{1}{2}}(\mathbf{x}^{(i)}_{\lambda_{j-1}}) d\boldsymbol{\epsilon}_\lambda \tag{2.58}$$

$$\mathbf{m}^{(i)}_{\lambda_j} = \mathbf{m}^{(i)}_{\lambda_{j-1}} + \int_{\lambda_{j-1}}^{\lambda_j} \left[\mathbf{A}(\mathbf{x}^{(i)}_{\lambda_{j-1}}) \mathbf{m}^{(i)}_{\lambda_{j-1}} + \mathbf{b}(\mathbf{x}^{(i)}_{\lambda_{j-1}})\right] d\lambda \tag{2.59}$$

$$\mathbf{P}^{(i)}_{\lambda_j} = \mathbf{P}^{(i)}_{\lambda_{j-1}} + \int_{\lambda_{j-1}}^{\lambda_j} \left[\mathbf{A}(\mathbf{x}^{(i)}_{\lambda_{j-1}}) \mathbf{P}^{(i)}_{\lambda} + \mathbf{P}^{(j)}_{\lambda} \mathbf{A}^T(\mathbf{x}^{(i)}_{\lambda_{j-1}})\right] d\lambda + \int_{\lambda_{j-1}}^{\lambda_j} \mathbf{D}(\mathbf{x}^{(i)}_{\lambda_{j-1}}) d\boldsymbol{\epsilon}_\lambda \tag{2.60}$$

where,

$$\mathbf{A}(\mathbf{x}^{(i)}_{\lambda_j}) = -\frac{1}{2} \mathbf{D}(\mathbf{x}^{(i)}_{\lambda_j}) \left[\mathbf{P}_0^{-1} + \mathbf{H}^T_{\lambda_j} \mathbf{R}^{-1}_{k+1} \mathbf{H}_{\lambda_j}\right] \tag{2.61}$$

$$\mathbf{b}(\mathbf{x}^{(j)}_{\lambda_i}) = \frac{1}{2} \mathbf{D}(\mathbf{x}^{(i)}_{\lambda_j}) \left[\mathbf{P}_0^{-1} \phi(\mathbf{x}^{(i)}_{\lambda_j}) + \mathbf{H}^T_{\lambda_i} \mathbf{R}^{-1}_{k+1} \mathbf{z}_{\lambda_j}\right] \tag{2.62}$$

with $\mathbf{z}_{\lambda_j} = \mathbf{z}_{k+1} + \mathbf{H}_k \mathbf{x}^{(i)}_{\lambda_j} - \psi(\mathbf{x}^{(i)}_{\lambda_j})$. $\mathbf{P}_0$ is the prior density covariance estimate, $\mathbf{R}_k$ the measurement noise covariance matrix, $\phi(\mathbf{x}_{\lambda_j})$ the transition function while $\mathbf{H}_{\lambda_j}$ is the Jacobian matrix of the measurement function $\psi(\mathbf{x}_{\lambda_j})$. Weights are associated with the particles which are evaluated at the end of the pseudo-time recursion, based on the following equation,

$$w^{(i)}_{k+1} \propto w^{(i)}_k \int p(\mathbf{z}_{k+1} | \mathbf{x}'_{k+1}) p_j(\mathbf{x}'_{k+1} | \mathbf{Z}_k) d\mathbf{x}'_{k+1} \tag{2.63}$$

All integrals are approximated using Ozaki's discretization scheme. Resampling with replacement might be needed in order reduce the weight variance. SPF has been used to derive two types of particle filters: a Gaussian sum particle filter and a marginal particle filter.

SPF scheme builds further on the GFIS flow. One of the major differences is that posterior density is approximated by a mixture of Gaussians, as opposed to a single multivariate Gaussian (MVG). Also, no explicit assumption on the Gaussianity of the models is made, which make SPF quite attractive. Moreover, the flow takes into account the local curvature of the prior and the likelihood in form of the Hessian matrices. Altogether, SPF seems to offer a superior alternative to the existing particle flow methods. But this must be understood in the light of the fact that the flow is based on some a posteriori knowledge. Also, it is quite demanding from an implementation point of view. This is evident from the fact that the pseudo-time has to be adaptively resolved every time and instead of one, N Gaussian components need to be updated through out the pseudo-time loop. Finally, the integration method chosen is more intensive than a simple first order Euler integration, as suggested for many of the earlier discussed methods.

2.5 Summary

The aim of the chapter was to describe the main methods employed for nonlinear state estimation. This included traditional as well non-traditional methods based on gradual inclusion of the measurements. Both types of methods are approximate in the sense that each of the derived solution relies on a set of assumptions and approximations. In the case of EKF, the assumption is that the prior density and the likelihood can be approximately modeled as being Gaussian. This lessens the generality of Kalman filter based methods. While SMC and SMCMC make no such assumption, they suffer from the degeneration of the particles and also the curse of dimensionality. To alleviate these problems, especially the former, a novel concept has been worked out by many researchers in the field of nonlinear filtering, which aims for a gradual or step by step inclusion of the observational data in the measurement update step. It is hoped that this would help lessen the degeneracy, and could also help against the curse of dimensionality to a certain extent. Even though the idea is rather recent, a plethora of literature has already been published on new filters based thereof. We have described seven such methods in this chapter. All of these methods come with their own set of specific assumptions and simplifications, and they have garnered varied level of success. The overarching concept behind all these methods is that instead of introducing the likelihood in its full form in a single step, its effect is injected gradually by morphing a flat function into the actual form of the likelihood.

Bridging densities can be regarded as one of the earlier attempts to deal with the problem of degeneracy using progressive update. Though quite elegant, the actual implementation might require a large number of intermediate sampling stages. Each of such stages might require running many MCMC iterations. Also, the optional resampling step might be necessary to reduce the weight variance. All of these issues limits its practicality. GFIS and GSMC, on the other hand are quite easy to implement, but they are limited in the performance owing to the underlying Gaussian assumption, though post correction is made to cater for the inherent discrepancies. Methods like Progressive Bayes, Optimal homotopy based parameterization, SPF and GbF could result in much improved accuracy, but the actual implementation is rather involved and comes at much higher processing cost. The first two methods are limited in terms of their applicability, since they require the solution of a large set of ODEs even for a one dimensional example. Also, the state space discretization could increase exponentially as the system dimensionality increases. The last two methods involve the estimation of a large number of integrals, which might require very fine space and pseudo-time discretization to achieve the desired accuracy. Better levels of accuracy can of course be achieved, albeit at much higher processing cost. After all, as it is said, there is no *free lunch* to be found anywhere.

Chapter 3

Log Homotopy based particle flow filters

In the previous chapter, existing methods for nonlinear state estimation were discussed in detail. The focus, in particular, was on the so called *progressive measurement inclusion* methods, in which the measurement update step is carried out gradually. It was highlighted that most of such methods are either based on the modification of a basic Gaussian framework which lessened their applicability, or they require solution of several integrals over adjoint state space regions which could become progressively hard to compute with the increase in the state space dimensionality.

Therefore, we look for a different method which, though similar in concept, strikes a nice trade-off between the complexity of the method and the accuracy of the results. In our search, we find one such very compelling methodology, which is also based on the gradual inclusion of the measurements. This has been suggested by Daum and Huang in a series of papers [DH07]-[DHN16]. The method is based on the particle representation of the involved probability densities. The key idea is to model the transition of particles from the prior to the posterior density as a physical flow under the influence of an external force (measurements). Particles are sampled from the state transition density and a notion of synthetic time also called the pseudo-time is introduced, in which particles flow until they reach *correct* posterior locations. An ordinary differential equation defines the flow of particles in pseudo-time, while the a partial differential equation (Fokker-Planck equation or FPE) describes the density evolution. A homotopy is defined between the log prior and the log posterior densities, through the likelihood. The likelihood term is introduced gradually, with the pseudo-time parameter controlling the inclusion rate. By combining the log-homotopy equation with the FPE under different assumptions, a series of ODE characterizing the flow vector is obtained. The flow ODE is then integrated numerically to yield the updated states of particles. The new filter is termed as log-homotopy based particle flow filter or simply Daum-Huang filter (DHF) after the developers. Amongst many flow solutions that have been derived, the famous ones are the incompressible flow [DH07], zero diffusion exact flow [DH09b], Coulomb's law flow [DHN11a], zero-curvature flow [DH13b] and non zero diffusion flow [DH13a].

In this chapter, we first describe the theory behind the DHF. The most common variants of the log-homotopy based flow will be derived. Then, we study the typical implementation of DHF and highlight the critical steps. Each of those steps are studied in detail and recommendations for improvements are made, culminating into a modified DHF implementation. Finally, using

nonlinear and non-Gaussian examples, we study the effects of the proposed changes in the implementation. The outline of the chapter is given as follows: We start by a brief mention of the background concepts in the section 3.1. Next, we present a description of homotopy based particle flow together with the derivation of the generic homotopy based flow equation and specific flow solutions in section 3.2. In the section 3.3, we present a generic algorithm for the DHF implementation, and highlight the important steps in the section 3.4. In there, we describe, in detail, a number of possible schemes that could be employed for each of critical steps mentioned in the previous section. The improved implementation is described in section 3.5. Section 3.6 starts with the description of the two models used in the numerical analysis, followed by a sub-section on the parameter settings and the simulation methodology. Results for proposed alternative methods are described in section 3.7. Finally, the chapter is concluded by section 3.8.

3.1 Background

Before delving into the log-homotopy based particle flow filters, we would like to introduce some basic concepts required in the forthcoming discussion.

3.1.1 Stochastic differential equations

An *Ordinary Differential equation* (ODE) describes the evolution of a non-stochastic dynamic system over the time. The system is described in terms of its state vector, such that the ODE is defined by the following functional equation,

$$g(\mathbf{x}(t), \mathbf{x}'(t), \mathbf{x}''(t), \cdots, \mathbf{x}^{(N)}(t)) = 0 \tag{3.1}$$

A simple first order ODE can be expressed as,

$$\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{x}(t)) \tag{3.2}$$

Given the initial state $\mathbf{x}(0) = \mathbf{x}_0$, the value at anytime t>0 can be uniquely specified by the following

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_{\tau=0}^{\tau=t} \mathbf{f}(t, \mathbf{x}(t))\mathrm{d}\tau \tag{3.3}$$

Generally speaking, analytical solutions of ODEs can only be found for some exceptional cases. Hence numerical methods have to be relied upon. A stochastic differential equation (SDE) is a generalization of ODE, in the sense that in addition to the deterministic term $\mathbf{f}(t, \mathbf{x}(t))$, a second term $\mathbf{D}(\mathbf{x}, t)$ is also added to equation (3.2), representing the stochastic part. A first order SDE is given by,

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(t, \mathbf{x}(t))\mathrm{d}t + \boldsymbol{\sigma}(\mathbf{x}(t), t)\mathrm{d}\mathbf{W}_t \tag{3.4}$$

Here $\mathbf{W}_t$ represents a M-dimensional *Wiener* or *Brownian process*, while $\mathbf{f}(t, \mathbf{x}(t)) \in \mathbb{R}^{d\times 1}$ and $\boldsymbol{\sigma}(\mathbf{t}, t) \in \mathbb{R}^{d\times M}$ are deterministic terms called as the drift vector and the diffusion matrix respectively. A *Brownian process* is a stochastic process which has continuous sample paths,

with stationary independent increments. Also for t>0, $\mathbf{W}_t$ has a normal distribution $\mathcal{N}(\mathbf{0}, t)$. The solution of (3.4) can be expressed as,

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_{\tau=0}^{\tau=t} \mathbf{f}(t, \mathbf{x}(t))\mathrm{d}\tau + \int_{\tau=0}^{\tau=t} \boldsymbol{\sigma}(\mathbf{x}, \tau)\mathrm{d}\mathbf{W}_t \tag{3.5}$$

The first is the standard *Riemann* integral, while the second one is an *Itô stochastic* integral. Necessary proofs for the existence and uniqueness of the solution can be found in [Øk14]. Like their deterministic counterparts, exact solutions for SDEs can only be found in certain special cases. Typically, an SDE can be numerically solved using a variety of methods, such as the *Euler-Maruyama* approximation, *Milstein* approximation and *Stochastic Runge-Kutta* method etc.

3.1.2 Fokker-Planck equation

Given some initial description of a probability density for some stochastic process, a pertinent question could be, how does the density evolves over the passage of time. As discussed before, an arbitrary stochastic process can have both deterministic and probabalistic parts. Therefore, the evolution of the density with time would require a framework that includes both aspects of the governing SDE. One of the first studies describing the time evolution of probability density was carried out by Einstein in 1905. It appeared in the context of describing the *Brownian motion*, i.e. random motion of tiny particles suspended in a fluid, using the statistical concepts where the governing SDE was a simple diffusion equation (see Sect. 1.2 of [Gar04]). The resulting Partial Differential Equation (PDE) was very similar to the heat diffusion equation as proposed by Fourier in 1827, though the connections were not made until done by Einstein. Later on, the concept was generalized to include a drift component as well. The resulting PDE equation describing the time evolution of a PDF is termed as the Fokker-Planck equation or FPE. An FPE in one dimension is given as:

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}[f(x,t)p(x,t)] + \frac{1}{2}\frac{\partial^2}{\partial x^2}[\sigma^2(x,t)p(x,t)] \tag{3.6}$$

where $f(x,t)$ and $\sigma(x,t)$ are the drift and the diffusion components of the SDE, respectively. The equation was jointly named after *Adrian Fokker* and *Max Planck*, as they both worked on the foundational aspects of a complete theory of particle fluctuations inside a radiation field. The FPE is also called the *Kolmogorov forward equation* and the *Smoluchkowski equation*. FPE can be written as,

$$\frac{\partial p(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[-f(x,t)p(x,t) + \frac{1}{2}\frac{\partial}{\partial x}[\sigma^2(x,t)p(x,t)]\right] = \frac{\partial}{\partial x}J(x,t) \tag{3.7}$$

Here $J(x,t)$ is termed as the probability current, as its integral over any surface defines the net flow of probability across it. For a stochastic process admitting a stationary PDF, the probability current progressively goes to zero with the passage of time. The stationary solution is found by putting $\frac{\partial}{\partial x}J(x,t)$ to zero and solving the resultant equation under appropriate boundary conditions. Few examples of processes admitting a stationary solution include the diffusion of particles in a force field (e.g. gravitational), Rayleigh and the Orstein-Uhlenbeck processes etc. We study the concept of stationary FPE in detail in the chapter 5 of this thesis. As discussed

in the Sect. 3.5.2 of [Gar04], FPE can be seen as a special case of the more general *Differential Chapman Kolmogorov equation*. In d-dimension, the FPE corresponding to the stochastic process governed by SDE (3.4) is given by ,

$$\frac{\partial p(\mathbf{x},t)}{\partial t} = -\sum_{i=1}^{d}\frac{\partial}{\partial x_i}[f_i(\mathbf{x},t)p(\mathbf{x},t)] + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2}{\partial x_i \partial x_j}\left[Q_{i,j}(\mathbf{x},t)p(\mathbf{x},t)\right] \tag{3.8}$$

where $\mathbf{Q}(\mathbf{x},t)$ is the diffusion tensor, defined as,

$$Q_{i,j}(\mathbf{x},t) = \sum_{l=1}^{M}\sigma_{i,l}(\mathbf{x},t)\sigma_{l,j}(\mathbf{x},t) \tag{3.9}$$

3.2 Log-homotopy based particle flow

Now we can formally start with describing the details of the log-homotopy based particle flow. Filters based on such flows have been termed in the literature as the Daum-Huang particle flow filters or simply the Daum-Huang filters DHF, after the original developers. Conceptually, the log-homotopy based particle flow methodology bears a great deal of similarity with the other particle based progressive data assimilation methods described in the previous chapter. A homotopic relationship is defined between the prior and posterior densities through the likelihood, albeit in the logarithmic domain. The likelihood inclusion is controlled by a certain parameter, which also serves the role of an artificial or pseudo time. It is assumed that the particles, starting from their prior locations, obey an Itô stochastic differential equation in their motion w.r.t. that pseudo time parameter, until they reach their final (posterior) locations. Also, the evolution of the prior density into the posterior is described by an FPE. The SDE is termed as the flow equation, and the main task is to derive an expression for it. This is done by combining the log homotopic relationship between the densities with the FPE defined in the pseudo-time. This leads to the so called *generic flow equation*, which under specific set of assumptions yields a number of particular flow solutions. These can be solved to update the particles states, forming the progressive measurement update step.

We now formally start with mathematical description of this concept. Let λ be the pseudo time parameter, controlling the likelihood inclusion. A log-homotopy function $\log p(\mathbf{x}_k,\lambda)$ which bridges the prior and the posterior densities, can then defined using the homotopy parameter λ,

$$\log p(\mathbf{x}_{k+1},\lambda) = \log g(\mathbf{x}_{k+1}) + \lambda \log h(\mathbf{x}_{k+1}) - \log K(\lambda). \tag{3.10}$$

where $g(\mathbf{x}_{k+1})$ represents the prior $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$, $h(\mathbf{x}_{k+1})$ the likelihood $p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})$ and λ assumes the role of a pseudo-time varying from 0 to 1. $K(\lambda)$ is the normalization constant for the posterior density independent of $\mathbf{x}_{k+1}$. λ = 0 sets $p(\mathbf{x}_{k+1},\lambda)$ equal to the prior density while with λ = 1 the transformation is completed to the normalized posterior density. Therefore, $p(\mathbf{x}_{k+1},\lambda)$ serves the same roles as the intermediate densities in the other progressive methods. Please note that the λ is an artificial/synthetic time construct, as there is no real time flow involved between the change of prior to the posterior distribution. From now on we drop the time index k for the sake of convenience.
Now given the above formulation of the log homotopic relationship, it is further supposed that the flow of particle obey the Itô SDE defined in the pseudo-time,

$$d\mathbf{x} = \mathbf{f}(\mathbf{x},\lambda)d\lambda + \boldsymbol{\sigma}(\mathbf{x},\lambda)d\mathbf{W}_\lambda \tag{3.11}$$

where $\mathbf{f}(\mathbf{x}, \lambda)$ is the flow vector, $\mathbf{w}$ is the M-dimensional Wiener process with diffusion matrix $\boldsymbol{\sigma}(\mathbf{x}, \lambda)$. For a flow characterized as in (3.11), the evolution of the density $p(\mathbf{x}, \lambda)$ w.r.t the parameter λ is given by the Fokker-Planck equation, defined w.r.t. the pseudo-time λ,

$$\frac{\partial p(\mathbf{x}, \lambda)}{\partial \lambda} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}[f_i(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)] + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}[Q_{i,j}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)] \tag{3.12}$$

where $\mathbf{Q}(\mathbf{x}, \lambda)$ is the diffusion tensor. It can also be expressed in short hand notion as,

$$\frac{\partial p(\mathbf{x}, \lambda)}{\partial \lambda} = -\nabla \cdot (\mathbf{f}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)) + \frac{1}{2}\nabla^T \mathbf{Q}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)\nabla \tag{3.13}$$

where ∇ is the spatial vector differentiation operator. From (3.10), the pseudo-time derivative of the density $p(\mathbf{x}, \lambda)$ can be formulated.

$$\frac{\partial p(\mathbf{x}, \lambda)}{\partial \lambda} = p(\mathbf{x}, \lambda)\left(\log h(\mathbf{x}) - \frac{\partial \log K(\lambda)}{\partial \lambda}\right) \tag{3.14}$$

By combining equations (3.13) and (3.14) we get,

$$p(\mathbf{x}, \lambda)\left(\log h(\mathbf{x}) - \frac{\partial \log K(\lambda)}{\partial \lambda}\right) = -\nabla \cdot (\mathbf{f}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)) + \frac{1}{2}\nabla^T \mathbf{Q}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)\nabla \tag{3.15}$$

Using the vector calculus identity,

$$\nabla \cdot (\mathbf{a}\mathrm{b}) \ = (\nabla \cdot \mathbf{a})\,\mathrm{b} + \mathbf{a} \cdot (\nabla \mathrm{b})$$

Equation 3.15 can be further expanded,

$$\log h(\mathbf{x}) - \frac{\partial \log K(\lambda)}{\partial \lambda} = -\mathbf{f}^T(\mathbf{x}, \lambda) \cdot \nabla \log p(\mathbf{x}, \lambda) - \nabla \cdot \mathbf{f}(\mathbf{x}, \lambda) + \frac{1}{2p(\mathbf{x}, \lambda)}\left(\nabla^T \mathbf{Q}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)\nabla\right) \tag{3.16}$$

Then, the objective becomes to solve the generic flow equation (3.16) for the yet unknown flow $\mathbf{f}(\mathbf{x}, \lambda)$. Various flow solutions have been obtained by solving (3.16) under different assumptions. Here we discuss four such flows derived by F. Daum and J. Huang in their series of papers.

Incompressible flow

The first solution of (3.16) appeared in [DH07], which was based on two distinct assumptions. Firstly, the diffusion term $\boldsymbol{\sigma}(\mathbf{x}, \lambda)$ in (3.11) is ignored. Secondly, the flow is considered incompressible, i.e. $\nabla \cdot \mathbf{f}(\mathbf{x}, \lambda) = 0$. Also the derivative of the logarithm of normalization constant $\frac{\partial \log K(\lambda)}{\partial \lambda}$ is assumed to be very small, and therefore neglected. This leads to the flow equation,

$$\mathbf{f}(\mathbf{x}, \lambda) = -\log h(\mathbf{x}) \ \frac{\nabla \log p(\mathbf{x}, \lambda)}{||\nabla \log p(\mathbf{x}, \lambda)||^2} \tag{3.17}$$

The flow described by (3.17), has the magnitude proportional to the $\log h(\mathbf{x})$ and acts in the direction of the gradient of the log-homotopy. According to [DH07], the implementation needs

care, otherwise outliers can result from the direct application of equation (3.17). The gradient in (3.17) can be found by taking the derivative of log-homotopy equation (3.10),

$$\nabla \log p(\mathbf{x}, \lambda) = \nabla \log g(\mathbf{x}) + \lambda \nabla \log h(\mathbf{x}) \tag{3.18}$$

Given the measurement model, $\nabla \log h(\mathbf{x})$ can be found analytically. This might not be the case for $\nabla \log g(\mathbf{x})$ because here the function g(**x**) is represented by set of randomly distributed points (particles) in d-dimensional space. Authors in [DHKK09] and [DHNK09] have suggested *seventeen dubious methods* for gradient estimation of the log-homotopy function. Out of these seventeen methods two are said to be most general and overall best.

The first method approximates the gradient as a solution of k linear equations where each equation corresponds to the directional derivative approximated by a simple first difference. It involves finding *k* approximate neighbors (as opposed to the *k* real neighbors) to each particle in order to reduce the computational complexity. The search for nearest particles is split in to two phases,

1. First, all particles are projected along a line and *M* closest 1-dimensional points are found for each particle. That line is chosen to be the eigenvector corresponding to the largest eigenvalue of the error covariance matrix of particles. Normally, the number of particles required for DHF are quite low, therefore it is advised to run a EKF/UKF in parallel to get an estimate of prior covariance matrix.

2. Second, the *M* particle are projected back and *k* nearest points in the original d-dimensional space are searched. These *k* points are the approximated nearest neighbors. This algorithm is termed as *fast k-NN* algorithm. These points can be used to find the gradient.

An implementation for incompressible particle flow DHF has been mentioned in [CWDH11]. Another detailed analysis for one dimensional incompressible flow is presented in [CM10], where it is shown that for certain initial conditions (prior values), a solution for the flow equation might not exist. This is caused by the existence of singularities, which are also the reason behind the phenomenon of outliers. Several methods to deal with the singularity are discussed, such as moving only those particles which have a flow solution, stopping the flow entirely when reaching a singularity or temporarily stopping the particles and restart in the next iteration. It has also been reported that the incompressible flow DHF is a good mode estimator. The filter based on the incompressible flow is termed as DHF-IC.

Exact flow

If the diffusion term is still assumed to be zero and $\frac{\partial \log K(\lambda)}{\partial \lambda}$ is neglected, but the flow is allowed to be compressible, the following equation can be derived from (3.16)

$$\log h(\mathbf{x}) + \mathbf{f}^T(\mathbf{x}, \lambda) \cdot \nabla \log p(\mathbf{x}, \lambda) = -\nabla \cdot \mathbf{f}(\mathbf{x}, \lambda) \tag{3.19}$$

Different flows have been derived in [DH10a] based on solutions of (3.19). One particular solution relates to the case when $\log g(\mathbf{x})$ and $\log h(\mathbf{x})$ are bilinear in the components of the vector **x**, e.g. Gaussian prior and likelihood.

$$\log g(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{P}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \tag{3.20}$$

$$\log h(\mathbf{x}) = -\frac{1}{2}(\mathbf{z} - \psi(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x})) \tag{3.21}$$

Then, the gradient of the two densities can be written as,

$$\nabla \log p(\mathbf{x}, \lambda) = -\mathbf{P}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) + \lambda \mathbf{H}^T \mathbf{R}^{-1}(\mathbf{z} - \psi(\mathbf{x})) \tag{3.22}$$

where $\mathbf{H} = \frac{\partial \psi}{\partial \mathbf{x}}\big|_{\mathbf{x}_\lambda}$. More generally speaking, if the additive noise processes w_k and v_k belong to the exponential family, then an analytical solution termed as the *exact flow* can be derived. This is given as,

$$\mathbf{f}(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda) \tag{3.23}$$

After some calculation, $\mathbf{A}(\lambda)$ and $\mathbf{b}(\lambda)$ turn out to be,

$$\mathbf{A}(\lambda) = -\frac{1}{2}\mathbf{P}\mathbf{H}^T(\lambda \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H} \tag{3.24}$$

$$\mathbf{b}(\lambda) = (I + 2\lambda\mathbf{A})[(I + \lambda\mathbf{A})\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} + \mathbf{A}\bar{\mathbf{x}}] \tag{3.25}$$

For nonlinear systems, the measurement model can be linearized by a Taylor series expansion up to the first order term, such that $\mathbf{z} \approx \mathbf{z} - \psi(\mathbf{x}_\lambda) + \mathbf{H}\mathbf{x}_\lambda$. The detailed derivation, as described in [DC12], is given in the Appendix 3.A. We abbreviate this filter type as DHF-EF.

Coulomb's law based flow

Yet another solution can be developed in which the flow of particles in the pseudo-time is derived from the gradient of Poisson's equation [DHN11a]. The diffusion term in (3.16) is again assumed to be zero, but the derivative of the normalization constant is not ignored. Instead, it is derived and an exact expression is found,

$$\frac{\partial \log K(\lambda)}{\partial \lambda} = \mathbb{E}\left(\log(h(\mathbf{x}))\right) \tag{3.26}$$

Then the equation (3.16) is written in the form

$$\nabla \cdot \mathbf{q}(\mathbf{x}, \lambda) = -\eta(\mathbf{x}, \lambda) \tag{3.27}$$

where, $\mathbf{q}(\mathbf{x}, \lambda) = \mathbf{f}(\mathbf{x}, \lambda)p(\mathbf{x}, \lambda)$ and $\eta(\mathbf{x}, \lambda) = -p(\mathbf{x}, \lambda)\left(\log h(\mathbf{x}) - \frac{\partial \log K(\lambda)}{\partial \lambda}\right)$ It is noticed that the integral of $\eta(\mathbf{x}, \lambda)$ w.r.t. x along the flow is zero,

$$\int_\Omega \eta(\mathbf{x}, \lambda) = 0 \tag{3.28}$$

where Ω is the relevant volume of the state space. This is analogous to a zero divergence of the electric flux density out of an enclosed region without any charge (first Maxwell equation). Next, it is reasoned that if the function $\mathbf{q}(\mathbf{x}, \lambda)$ can be assumed to be the gradient of scalar potential function $V(\mathbf{x}, \lambda)$, then (3.27) can be expressed as Poisson's equation for the potential $V(\mathbf{x}, \lambda)$.

$$\Delta V(\mathbf{x}, \lambda) = \eta(\mathbf{x}, \lambda) \tag{3.29}$$

such that,

$$f(\mathbf{x}, \lambda) = \frac{\nabla V(\mathbf{x}, \lambda)}{p(\mathbf{x}, \lambda)} \tag{3.30}$$

where Δ is the Laplacian operator. The solution of (3.29) can be expressed in terms of the convolution integral for d≥3,

$$V(\mathbf{x},\lambda) = -\int_{\Omega} \eta(\mathbf{y},\lambda)\frac{c}{||\mathbf{x}-\mathbf{y}||^{d-2}}d\mathbf{y} \tag{3.31}$$

where $c = (4\pi)^{-\frac{d}{2}}\Gamma(\frac{d}{2}-1)$ and $\mathbf{y}$ is the running variable. The above equation gives the solution of the scalar potential $V(\mathbf{x},\lambda)$, whereas our quantity of interest is its gradient. Taking the gradient of (3.31) we get,

$$\nabla V(\mathbf{x},\lambda) = \mathbb{E}\left[\left(\log h(\mathbf{y}) - \frac{\partial \log K(\lambda)}{\partial\lambda}\right)\frac{c\,(2-d)\,(\mathbf{x}-\mathbf{y})^T}{||\mathbf{x}-\mathbf{y}||^d}\right] \tag{3.32}$$

Using the Monte Carlo approximation for the integrals, (3.32) can be approximated,

$$\nabla V(\mathbf{x}_i,\lambda) \approx \frac{1}{k}\sum_{j\in S_i}\left(\log h(\mathbf{x}_j) - \frac{1}{k}\sum_{j\in S_j}\log h(\mathbf{x}_j)\right)\cdot\left(\frac{c\,(2-d)\,(\mathbf{x}_i-\mathbf{x}_j)^T}{||\mathbf{x}_i-\mathbf{x}_j||^d+\alpha}\right) \tag{3.33}$$

The expression for the gradient $\nabla V(\mathbf{x},\lambda)$ is similar to the electromagnetic force equation given by Coulomb's law, hence the name of the flow. In order to reduce the computational complexity, the outer summation is carried out over the subset of *k* nearest neighbors of the *ith* particle $\mathbf{x}_i$, which is denoted here by S_i. This is motivated by the fact that as the state space dimensionality increases, the contribution of particles far apart approaches zero exponentially. α is set to $\frac{1}{\beta}Tr(P)^{\frac{d}{2}}$, where both α and β are the design parameters. Their purpose is to regularize the expression for $\nabla V(\mathbf{x},\lambda)$. Also, P can be approximated by a prior covariance matrix estimate. Fast Multipole Method (FMM) is suggested as an alternative method for solving the Poisson's equation. The filter based on this type of flow is referred as DHF-CLF.

Non zero diffusion constrained flow

The last type of flow we consider can be derived by not ignoring the diffusion term in equation (3.16) as suggested in [DH13a]. It starts by taking the gradient of the generic flow equation (3.16), which yields,

$$\begin{aligned}\nabla\log h(\mathbf{x}) = &-\nabla\log p(\mathbf{x},\lambda)^T\cdot\nabla\mathbf{f}(\mathbf{x},\lambda)\\ &-\mathbf{f}^T(\mathbf{x},\lambda)\cdot\nabla^2\log p(\mathbf{x},\lambda) - \nabla\left(\nabla\cdot\mathbf{f}(\mathbf{x},\lambda)\right)\\ &+\nabla\left(\frac{1}{2p(\mathbf{x},\lambda)}\nabla^T\mathbf{Q}(\mathbf{x},\lambda)p(\mathbf{x},\lambda)\nabla\right)\end{aligned} \tag{3.34}$$

An analytical evaluation of the equation (3.34) for the flow $\mathbf{f}(\mathbf{x},\lambda)$ is not possible, though numerical methods could be employed for this purpose. Depending on the dimensionality of the state-space, this can become computationally demanding. However, a little trick can lead to a closed form solution for the flow, if the following constraint holds,

$$\begin{aligned}\nabla\left(\frac{1}{2p(\mathbf{x},\lambda)}\nabla^T\mathbf{Q}(\mathbf{x},\lambda)p(\mathbf{x},\lambda)\nabla\right) = &\nabla\log p(\mathbf{x},\lambda)^T\cdot\nabla\mathbf{f}(\mathbf{x},\lambda)\\ &+\nabla\left(\nabla\cdot\mathbf{f}(\mathbf{x},\lambda)\right)\end{aligned} \tag{3.35}$$

This results in a simple formula for the flow equation, given by

$$\mathbf{f}(\mathbf{x}, \lambda) = -\Big(\nabla^2 \log p(\mathbf{x}, \lambda)\Big)^{-1} \big(\nabla \log h(\mathbf{x})\big)^T \tag{3.36}$$

The flow derivation does not involve neglecting the diffusion term, instead it appears in the constraint equation. Hence this flow is termed as non-zero diffusion constrained flow (NZDCF), and the DHF with this particular flow is termed as DHF-NZDCF.

A closer look at (3.36) reveals that it requires Hessian matrices of the log-prior and the likelihood densities, as well as the gradient of the log-likelihood. The gradient $\nabla \log h(\mathbf{x})$ and the Hessian of the log-likelihood, $\nabla^2 \log h(\mathbf{x})$ can be calculated analytically in most cases. On the other hand, there is no single method for the evaluation of the Hessian of the prior density, $\nabla^2 \log g(\mathbf{x})$. The most straightforward method is to approximate the prior density by a multivariate Gaussian density, and use the negative of the inverse of its covariance matrix $-\mathbf{P}^{-1}$. This leads to the following,

$$\begin{aligned} \nabla^2 \log p(\mathbf{x}, \lambda) \quad &= \nabla^2 \log g(\mathbf{x}) + \lambda \nabla^2 \log h(\mathbf{x}) \\ &\approx -\mathbf{P}^{-1} + \lambda \nabla^2 \log h(\mathbf{x}) \end{aligned} \tag{3.37}$$

It has been suggested in the paper by F.Daum and J.Huang that the matrix $\mathbf{P}$ can be set to the prior covariance matrix of a parallel running EKF/UKF. Another method suggested by the same authors is to use the fast k-NN algorithm to compute the hessian of the prior, similar to the incompressible flow. Alternatively, an approximation can be used instead, where $\mathbf{P}$ is the state covariance matrix computed directly from the prior position of particles.

The non-zero diffusion flow equation is more general in its context and applicability. This is because it does not assume any particular form of the prior density and likelihood function. However, it imposes two requirements. Firstly, the two functions should be no-where vanishing, and secondly, they should be sufficiently smooth. Both of these conditions guarantee that the gradient and the Hessian of the logarithm of densities always exist. Some regularization might be necessary if either of these two assumption does not remain valid.

Due its generality, we primarily focus on analyzing the NZDCF in the rest of this chapter. Since the flow can be ill-posed due to the possible violation of two necessary conditions, we suggest improvements in the implementation architecture to regularize the flow and to improve the overall performance of the filter.

3.3 Typical DHF implementation

Numerical results for the DHF have been presented in [DH10b]. DHF based on the incompressible and exact flows have been implemented by Choi. et al. in [CWDH11] for nonlinear scalar and linear vector system models. Exact flow DHF implementations for multi-target tracking have been reported in [DC12], where mobile targets are tracked based on the their received signal strength at a fixed sensor. In [BS14], joint probabilistic data association (JPDA) and maximum a posteriori penalty function (MAP-PF) algorithms based on the exact flow DHF have been derived. Recently, many researchers have carried out the comparative analysis for the DHF-NZDCF, in different application. This includes a comparison of DHF performance against more traditional methods for angle only filtering in 3D by Gupta et.al. [DGYM$^+$15], comparing the tracking performance of DHF vs. other methods for super-maneuverable targets by Kreucher et.al. in [KBS15], and the comparison of multi-sensor fusion using DHF against the particle filters in Mostagh and Chan in [MC15]. Results show a varying degree of success for DHF. While in some applications particle flow filters are shown to outperform the competitors, in others they do not perform quite well. The main issue is that while particle flow filters are theoretically quite elegant, their performance suffer from approximations made, both in theory and in the practical implementation. This includes the approximations made while deriving the flow, in the estimation of the prior density and also in the use of numerical integration techniques. This leads to the introduction of bias and loss of asymptotic consistency [BG14].

In Algorithm 4, we outline the implementation as described by T. Ding and M. Coates in [DC12].

Algorithm 4 Generic implementation of DHF

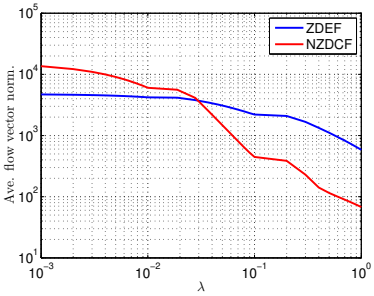
```
1:  procedure DHF
2:      Initialize DHF: Generate initial set of particles
3:      Initialize EKF/UKF: Initial mean and covariance
4:      Pseudo-time grid discretization
5:      for Loop over the time do
6:          Propagate particles using the dynamic model
7:          Prior covariance matrix estimate from EKF/UKF
8:          for Loop over the pseudo-time do
9:              for Loop over individual particles do
10:                 Integration of the flow equation
11:             end for
12:         end for
13:         Measurement Update for EKF/UKF
14:         Redraw particles (Optional)
15:     end for
16: end procedure
```

Particles are generated by sampling the transition density. An EKF/UKF is run in parallel to the main algorithm. This is done in order to approximate the prior covariance matrix. Next, the flow equation is solved in pseudo-time for all particles. The flow equation uses the prior covariance estimate from the parallel running EKF/UKF. Once done, the mean state vector is estimated and the measurement update is carried out for EKF/UKF. This process is repeated until the end of

the simulation time. The steps colored in red are the crucial factors in the performance of the DHF.

The first is the pseudo-time λ discretization strategy together with the numerical integration method. As the DHF flow is described by an ordinary differential equation, a suitable discretization is essential to capture the flow dynamics. Then the flow equation is integrated w.r.t. λ. While the exact implementation details in [DH13a], [CWDH11], [BS14] are not clear, authors in [DC12] have used single step Euler integration, as mentioned in the pseudo-code.The method is based on first order Taylor series expansion. It is simple to implement and is fairly quick. But care has to be taken as the flow ODE can exhibit stiffness. In that case a straight forward λ discretization together with the single step Euler integration might not work. For example, it has been mentioned in [DH13a] that the incompressible flow and ZDEF-DHF can implemented with uniform step size $\Delta\lambda = 0.1$, but not the NZDCF-DHF. For a comparison between the two flows, we refer to the multi-target tracking example studied in [KU14], with a special process model, termed as the *Coupled motion model*. Also the measurement model has non-Gaussian noise contribution. These specific models are mentioned in detail in section 3.6.

We plot the average of the flow vector norm against the pseudo-time λ for the zero diffusion exact flow (ZDEF) and the non-zero diffusion constrained flow (NZDCF), for the *strongly coupled* motion model.



(a)

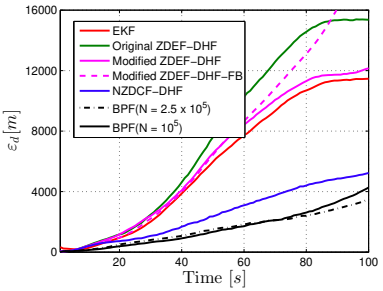
Figure 3.1: Log homotopy based flows as a function of λ

Here the averaging is done across the number of particles (N_p). It can be seen that the NZDCF has relatively high value at very small values of λ. Also it is more dynamic than the ZDEF as it exhibits three order of magnitude change in the same interval. This implies that, if the flow is coarsely sampled, say uniformly with $\delta\lambda$ = 0.1, the loss in performance for NZDCF-DHF can be far greater than for ZDEF-DHF as the flow dynamics would not be captured sufficiently well. The Jacobian of the NZDCF for this example has two zero eigenvalues. Also its Lipschitz constant is quite large, which points to the stiffness of this flow. EF on other hand is a non-stiff flow as its Jacobian is well-conditioned, and its Lipschitz constant is also relatively small. Therefore, the proper discretization of the pseudo time λ is very important.

Secondly, the non zero diffusion flow requires an estimate of the prior covariance matrix. While prior covariance estimate from parallel running EKF/UKF can be used as an approximation,

this makes the DHF accuracy dependent on that of the EKF/UKF. On the other hand, a sample covariance estimate can often be ill-conditioned. The question then becomes, whether there is a better method to estimate the prior covariance matrix. Finally, the re-generation of a new set of particles is an important step. Unlike a standard particle filter, the re-sampling/re-drawing step is not mandatory in the DHF, but optional. Instead, it has been mentioned that the homotopy flow moves the particles to their correct locations in the state space. But due to the approximations made in the derivations, the flow may not be accurate, which could reduce the accuracy of the estimates. Hence, the effect of particle re-generation is worth investigating.

To compare the performance of a typically implemented DHF, we again refer to the [KU14]. In the following figure, the root average mean square error (RAMSE) vs. time is plotted for different variants of DHF together with the EKF and SIR-PF.



(a)

Figure 3.2: Estimation error for the model discussed in [KU14] with the traditional DHF implementation.

As mentioned earlier, the measurement model used in this example is a nonlinear and non-Gaussian one. Hence, it presents challenges to all estimation schemes based on Gaussian assumptions. This includes the ZDEF-DHF and the EKF. Since the NZDCF is more general in its working, its expected performance should be better. Also, the prior covariance matrix is taken from a parallel running Kalman filter and no redrawing takes place. From the plot above, it can be seen that although NZDCF-DHF performs better than all other versions of DHF and the EKF, does not reach the performance of standard particle filters with the similar execution time. This points towards the inadequacy or ineffectiveness of the traditional DHF implementation methodology.

In the following section of the chapter, we look for improvements in the DHF performance by considering changes in the existing implementation architecture, as mentioned in Algorithm 4. Please note that the important or the key steps in the implementation are highlighted in red.

3.4 Important factors in DHF Implementation

In this section we individually discuss the aforementioned key factors affecting the DHF performance.

3.4.1 Pseudo-time discretization

In the previous section, it was shown that non-zero diffusion flow is considerably stiffer when compared to the exact flow. Hence, in order to efficiently solve the flow ODE, λ has to be finely grated. This has been mentioned in [DH13a] where the usage of exponentially spaced time steps or higher order integration schemes is recommended. In this paper we consider both uniform and non-uniform grid discretization. The idea is to analyze the effect of a particular grid discretization strategy and the numerical integration scheme on the filter performance, in terms of the estimation error and the processing time. While a coarse λ discretization would not result in the correct solution, a fine discretization on the other hand would lead to a substantial increase in the computational cost. Therefore, a middle point has to be chosen such that the flow dynamics at very small λ values are maximally captured, while only moderately increasing the processing cost.

3.4.2 Numerical solution of the flow ODE

The homotopy flow is defined by a vector ODE. In the current work, we seek for the numerical solution of the ODE. Broadly speaking, ODEs can be categorized into being stiff and non-stiff. While there is no precise definition of the stiffness, in the literature two criteria are generally mentioned for describing a stiff ODE. First, the condition number of the Jacobian matrix $\mathbf{J}(x,\lambda) = \frac{\partial \mathbf{f}(x,\lambda)}{\partial x}$ of a stiff ODE is quite large. As a consequence, multiple timescales exist in the ODE. Time scales, often referred to as modes, are defined by the inverse absolute eigenvalues of the Jacobian $J(x,\lambda)$. Secondly, in the Lipschitz inequality $||f(x_2,\lambda_2) - f(x_1,\lambda_1)|| \leq L||\lambda_2 - \lambda_1||$, the Lipschitz constant L is typically very large for a stiff ODE. Non-zero diffusion ODE can be characterized as stiff according to both criteria. Therefore, care has to be taken when choosing the numerical integration scheme for solving the flow ODE.
The standard Euler's method has been used for solving the flow ODE in earlier works. It is a first order method with a truncation error in the order of $\mathcal{O}(h^2)$. In this paper, we intend to compare the performance of some other numerical integration (NI) schemes for solving stiff ODEs alongside the Euler's method. There are several choices available. In the following, we mention some of the common NI methods for solving stiff ODEs.

3.4.2.1 Forth-order Runge-Kutta method

The forth-order Runge-Kutta method (RK4) is our second integration method. RK4 method has local truncation error in the order of $\mathcal{O}(h^5)$,while the total accumulated error is of order $\mathcal{O}(h^4)$.

3.4.2.2 Rosenbrock method

The Rosenbrock method belongs to the family of multistage procedures to solve stiff ODEs. The Jacobian matrix appears in the integration formula. Like the Runge-Kutta methods, Rosenbrock method successively form intermediate results. If the Jacobian matrix is ignored then

the method turns into the explicit Runge-Kutta scheme. Therefore, they are also called Runge-Kutta-Rosenbrock methods. Rosenbrock method preserve exact conservation properties due to the use of the analytic Jacobian matrix, and possess optimal linear stability properties for stiff problems.

3.4.2.3 Gear's method

Gear's method [C.G69] belongs to the class of methods known as backward differentiation formulae. It is an implicit integration method and uses the first and higher order derivatives. It is a predictor-corrector type scheme where each time step is initiated by prediction. Corrector iterations are then carried out until prescribed convergence criteria are achieved or non-convergence is deemed to have occurred.

3.4.3 Prior covariance shrinkage estimation

The evaluation of the flow equation (3.37) requires the availability of the prior covariance estimate. The covariance estimate can be derived in several ways. The simplest way is to estimate the covariance matrix using the prior particles. This is referred to as the sample covariance estimate S. S is an unbiased estimator of the true prior covariance P, and is the maximum likelihood estimate if the data is Gaussian distributed. But for nonlinear models / non-Gaussian noises, the Gaussian assumption does not remain valid. Also S could progressively get ill-conditioned, i.e. the spread of the eigenvalues gets larger with the time. This is especially the case, when the $\frac{d}{N_p}$ ratio is non-negligible, where d is the state vector dimension and N_p is the number of particles. As a consequence, the matrix inversion could lead to stability problems. For the case $d > n$, the resulting covariance matrix is not even full rank and hence not invertible. An alternative method suggested by the authors in [DH09a] is to run an EKF/UKF in parallel to DHF, and to use the prior covariance matrix generated by those filters. We refer to such a matrix as P_{XKF}, where XKF could be an extended or unscented versions of the KF. While this method is better than using the raw data based covariance estimate, it ties the DHF estimation accuracy to that of the EKF/UKF. P_{XKF} could also exhibit a wide spread of the eigenvalues.
Therefore, we look for an alternative method of covariance matrix estimation. That method should have two properties: the resulting matrix should always be positive definite (PD) and the matrix should be well-conditioned [SS05b]. One approach could be to start with the sample covariance, and ensure that the matrix is always PD. Such a matrix might not be well-conditioned. Alternatively, variance reduction techniques could be used to get a well-conditioned matrix, but this could be computationally expensive [SS05a]. There is another approach used in the multivariate statistics literature for the estimation of the covariance matrices, known as the *shrinkage estimation*. The use of such methods dates back to work of Stein [Ste56]. The main idea is to merge the raw estimate (S) which is unbiased but normally with high variance, together with a more structured but typically biased target (B) through a scale factor, to get the combined estimate (P_*). The objective is to reduce the estimation error, typically in a mean squared sense (MSE), by achieving an optimal trade off between the biased (B) and the unbiased (S) estimators. The scale factor is also called shrinkage intensity ρ as it shrinks the eigenvalues of S optimally towards the mean of eigenvalues of the true covariance matrix P [LW04b]. The resulting covariance matrix (P_*) will be biased, but will improve w.r.t. the two aforementioned properties, and is expected to lower the estimation error. There are several shrinkage estimators mentioned in the literature, with different target covariance matrices. In the current work, we

describe some of the more established shrinkage estimators. In subsections 3.4.3.1 to 3.4.3.3, shrinkage estimators are defined through a convex combination of the matrices B and S. The objective becomes to find an optimal shrinkage intensity that minimizes the cost function,

$$\min_{\rho} \quad E[||P_* - P||^2] \tag{3.38}$$

where $P_* = \rho B + (1-\rho)S$.

3.4.3.1 Shrinkage towards the Identity matrix

Shrinkage towards the Identity matrix is described in [LW04b]. The two main objectives defined are, 1) to get an asymptotically consistent estimator that is more accurate than the sample covariance matrix S, and is 2) also well-conditioned. No prior structure is assumed for the target matrix B, as it could lead to an increased bias. Instead, a simple matrix with same covariance terms and zero cross-variances (scaled identity) is chosen as the target. The shrinkage estimator has the following form

$$P_* = \frac{\alpha^2}{\delta^2}\mu^2 I + \frac{\beta^2}{\delta^2}S, \tag{3.39}$$

The estimator P_* asymptotically shrinks its eigenvalues towards the mean eigenvalue of the true covariance matrix P, in quadratic mean sense i.e. The terms α, β, δ and μ depend on the unobserved true covariance matrix P. Therefore, a consistent estimator of P_* is derived under the assumptions of general asymptotics. We term this estimator as P_{LW0}, and it has the following form,

$$P_{LW0} = \frac{a_n^2}{d_n^2}m_n I + \frac{b_n^2}{d_n^2}S, \tag{3.40}$$

where $m_n = tr(S)/d$ and,

$$d_n^2 = ||S - m_n I||^2 \qquad \bar{b}_n^2 = \frac{1}{n^2}\sum_{i=1}^{N_p}[||X_n^i (X_n^i)^T - S||^2]$$

$$b_n^2 = \min(\bar{b}_n^2, d_n^2) \qquad a_n^2 = d_n^2 - b_n^2.$$

$||.||$ is the squared *Frobenius norm* and X_n^i is the *ith* particle. Also the shrinkage intensity ρ is given by $\frac{a_n^2}{d_n^2}$. It is shown that the MSE for P_{LW0} asymptotically approaches that of P_* i.e. $E\big[||P_* - P||_n^2\big] - E\big[||P_{LW0} - P||_n^2\big] \rightarrow 0$. One main advantage of this estimator is that it does not assume any particular distribution for the data, and therefore is *distribution-free*.

3.4.3.2 Shrinkage towards the constant correlation matrix

This estimator is derived in [LW04a], in the context of portfolio optimization. The target matrix is chosen according to the constant correlation model. It means that pairwise correlations are identical, which is given by the average of all the sample correlations. We denote this estimator by P_{LW1}. The target matrix B is given by

$$B = \begin{cases} S_{ii} & : i = j \\ \bar{r}\sqrt{S_{ii}S_{jj}} & : i \neq j \end{cases}$$

where $\bar{r}$ is the average sample correlation. It is defined as

$$\bar{r} = \frac{2}{d(d-1)} \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{S_{ij}}{\sqrt{S_{ii}S_{jj}}} \tag{3.41}$$

The shrinkage intensity is defined as $\rho = \max\{0, \min\{1, \frac{\kappa}{d}\}\}$, with κ given as $\kappa = \frac{\hat{\pi}-\hat{\varrho}}{\hat{\gamma}}$. $\hat{\pi}$ denotes the sum of asymptotic variances of the entries of the sample covariance matrix S, while $\hat{\varrho}$ denotes the sum of asymptotic covariances of the entries of the shrinkage target B with the entries of the sample covariance matrix. $\hat{\gamma}$ gives a measure of the misspecification of the shrinkage target. The hats ($\hat{.}$) on top of terms indicate the fact that these are the estimates of the true values, which are not known. $\hat{\pi}$ are $\hat{\varrho}$ are given by,

$$\begin{aligned}
\hat{\pi} &= \frac{1}{d} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{d} \{(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j) - s_{ij}\}^2 \\
\hat{\varrho} &= \sum_{i=1}^{n} \hat{\pi}_{ii} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\bar{r}}{2} \Big(\sqrt{\frac{S_{jj}}{S_{ii}}} \vartheta_{ii,ij} + \sqrt{\frac{S_{ii}}{S_{jj}}} \vartheta_{jj,ij} \Big),
\end{aligned} \tag{3.42}$$

where

$$\begin{aligned}
\vartheta_{ii,ij} &= \frac{1}{d} \sum_{k=1}^{d} \{(x_{ik} - \bar{x}_i)^2 - \bar{x}_i) - S_{ii}\}\{(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j) - S_{ij}\} \\
\vartheta_{jj,ij} &= \frac{1}{d} \sum_{k=1}^{d} \{(x_{jk} - \bar{x}_j)^2 - \bar{x}_j) - S_{jj}\}\{(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j) - S_{ij}\}.
\end{aligned} \tag{3.43}$$

Finally $\hat{\gamma}$ is given by

$$\hat{\gamma} = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} (B_{ij} - S_{ij})^2. \tag{3.44}$$

3.4.3.3 Shrinkage towards the perfect costive correlation matrix

The authors in [LW03] suggest single-factor matrix as the shrinkage target. The paper is concerned with estimating the structure of the risk in the stock market and the modeling of the stock returns. The fact that stock returns are positively correlated with each other, is exploited. The shrinkage target is given by,

$$B_{ij} = \begin{cases} S_{ii} & : i = j \\ \sqrt{S_{ii}S_{jj}} & : i \neq j \end{cases}$$

The resulting linear estimator is denoted as P_{LW2}. The shrinkage intensity has the same form as for P_{LW1}, but with slightly different formula for $\hat{\varrho}$, which is given below.

$$\hat{\varrho} = \sum_{i=1}^{n} \hat{\pi}_{ii} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{2} \Big(\sqrt{\frac{S_{jj}}{S_{ii}}} \vartheta_{ii,ij} + \sqrt{\frac{S_{ii}}{S_{jj}}} \vartheta_{jj,ij} \Big) \tag{3.45}$$

3.4.3.4 Empirical Bayesian Estimator

In [Haf80], an estimator for multivariate Gaussian data is derived. It is given by the linear combination of the sample covariance matrix S and the scaled identity matrix. The scaling factor is estimated from the data. We denote this estimator by P_{EB} and it is given by

$$P_{EB} = \frac{N_p d - 2N_p - 2}{N_p^2 d}[det(S)]^{\frac{1}{d}} I + \frac{N_p}{N_p + 1} S \tag{3.46}$$

3.4.3.5 Stein Haff Estimator

This estimator is described in [Ste75]. The general form of the estimator is $V(S)\Phi(l(S))V(S)^T$, where matrix $V(S)$ contains the eigenvectors of the sample covariance matrix S while $\Phi(l(S))$ is a matrix that is a function of the eigenvalues $l(S)$ of the S. The data is assumed to be normally distributed. and the sample covariance estimate S is therefore Wishart distributed $S \sim \mathcal{W}(P, d)$. The Stein-Haff estimator is denoted by P_{SH}. It is constructed by leaving the eigenvectors of S unchanged while replacing the eigenvalues l by $\tilde{\tilde{l}}_i = nl_i / \Big(n - p + 1 + 2l_i \sum_{j=1, j\neq i}^{N_p} \frac{1}{l_i - l_j}\Big)$. Eigenvalues can get disordered by the transformation and might become negative, which could lead to the covariance estimate losing its positive definiteness. Therefore another algorithm called *Isotonic regression* is used in conjunction with the transformation [LP85]. This leads to eigenvalues $\tilde{\mathbf{l}} = \{\tilde{l_1}, \tilde{l_2}...\tilde{l_p}\}^T$. Hence the estimate P_{SH} is given by $V(S)D(\tilde{\mathbf{l}})V(S)^T$.

3.4.3.6 Minimax Estimator

The final shrinkage estimator considered is derived in [Ste82]. Again, Gaussian assumption is made. This estimator is termed Minimax because under a certain loss function, it has the lowest worst case error [LW04b]. Its structure is similar to P_{SH}, but sample eigenvalues are replaced by $\tilde{l}_i = \frac{n}{n+p-1-2i} l_i$. This estimator is denoted here by P_{MX}. Isotonizing regression is not applied in this case.

There is another interesting covariance estimator by Ledoit and Wolf [LW12] in which nonlinear transformation of the sample eigenvalues is considered. Also, it requires solving a nonlinear optimization problem using sequential linear programming. It is shown that the new nonlinear estimator outperforms the linear shrinkage estimators described earlier in this section. In the current work we don't consider this method.

3.4.4 Re-generating the particles set

In the standard particle filter, a new set of particles is generated after the measurement inclusion step. This is done in order to avoid the particle degeneracy. A measure of the particle degeneracy is the effective number of particles N_{eff}.When N_{eff} falls below a certain threshold, resampling of the particles is carried out. Depending on the number of particles, this can be computationally expensive. Homotopy based particle flow filters try to avoid the particle degeneracy by the gradual inclusion of the measurements. Unlike standard particle filters, resampling is not a mandatory step in the DHF according to [DH09a], as it moves the particles to the correct region of the state-space. However due to the inexactness of the homotopy flow ODE, the particle state update itself is imperfect. Hence the generation of a new particle set could potentially help in relocating/confining the particles to the correct region. Instead of the conventional resampling, an optional redrawing of the particles is hinted out by Daum and Huang in their papers. We have found a single paper which describes the particles redrawing method. In [DC12], it is suggested to redraw a new set of posterior particles by sampling a Gaussian distribution. The mean of the distribution is estimated using particles, while the filtered covariance matrix is provided by the EKF. In the current work we consider two redrawing schemes, one using a single multivariate Gaussian distribution(MVG), and other using a Gaussian mixture model (GMM) that is estimated through the Kernel density estimation (KDE).

3.4.4.1 MVG

Our first technique is inspired by the one described in the pseudo-code in [DC12]. The main difference is that we don't re-draw the whole set of particles. Instead, only those particles are re-drawn which are deemed too *wayward*. A multivariate Gaussian distribution is fitted to the posterior particles. This amounts to the estimation of the mean and variance of the MVG given the particles. New particles are generated from this MVG.

3.4.4.2 KDE-GMM

Our next re-drawing scheme is based on the intuition that a Gaussian fit to the posterior distribution might not be well suited for all cases. Hence we look for a non-Gaussian approximation to the filtered particles. The next most intuitive approach is to fit a Gaussian mixture model (GMM) to the data. The key-factor in the GMM approximation is the number of components, which can be set to a fixed value or could be data driven. The textbook approach to estimate the GMM parameters is the expectation-maximization or EM method [Bis06]. Alternatively, non-parametric methods like Kernel Density Estimation (KDE) may be employed for the estimation of the probability density, which is given by the sum of estimation kernels with a certain smoothing factor, centered at data points. Smoothing factor is also called bandwidth. In this paper we use the online KDE approach described by Kristan et.al. in [KLS11], in which a new method for online KDE is described. The method enables the construction of a multivariate probability density estimate by observing only a single sample at a time. The KDE of the target distribution is estimated using the sample distribution which is constructed by online clustering of the data points. Each new observation is treated as a distribution in the form of Dirac delta functions, and the sample distribution is modeled by a mixture of Gaussian and Dirac delta functions. Let $p_s(\mathbf{x})$ define the sample distribution, modeled as d-dimensional data as an

N-component Gaussian mixture model,

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} w_i^{KDE} \varphi_{\Sigma_{si}}(\mathbf{x} - \mathbf{x}_i) \tag{3.47}$$

where,

$$\varphi_{\Sigma_{si}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right) \tag{3.48}$$

is the Gaussian kernel. The kernel density estimate (KDE) is defined as the convolution of $p_s(\mathbf{x})$ with the kernel of covariance matrix *H*, also called the bandwidth.

$$\hat{p}_{KDE}(\mathbf{x}) = \varphi_H(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} w_i^{KDE} \varphi_{H+\Sigma_{si}}(\mathbf{x} - \mathbf{x}_i) \tag{3.49}$$

Next, the bandwidth of the kernel is estimated. *Asymptotic means integrated squared error* (AMISE) is used as a metric for measuring the closeness of the estimated density $p_{KDE}(\mathbf{x})$ with the unknown underlying density.

$$\mathcal{AMISE} = (4\pi)^{-\frac{d}{2}} |H|^{-\frac{1}{2}} N_w^{-1} + \frac{1}{4} d^4 \int tr^2\{H \mathbb{G}_p(\mathbf{x})\} d\mathbf{x} \tag{3.50}$$

where,

$$N_w = \frac{1}{\sum_{i=1}^{N} (w_i^{KDE})^2}$$

$$G = \hat{\Sigma}_{smp} \frac{4}{N_\alpha (d+2)}^{\frac{2}{d+4}} \tag{3.51}$$

The sample distribution is continuously refined and compressed in order to keep the algorithm complexity low. This is done by replacing clusters of components in ample distribution by a single Gaussian. The main idea is to identify clusters, such that each cluster can be sufficiently well approximated by a single component. The number of identified clusters are chosen such that the total error does not exceeds the threshold D_{th}. This threshold is a free parameter and can be chosen to achieve the desired level of accuracy.

$$\hat{M} = \arg\min_{M} \ \mathbb{E}(\Xi(M)), \ \ s.t. \ \mathbb{E}(\Xi(\hat{M})) \leq D_{th} \tag{3.52}$$

Hence, D_{th} controls the degree of approximation i.e. the number of GMM components fitted to the data. A lower value of D_{th} results in an increase in the number of components, and vice-versa. Another important step is re-vitalization which aims to counter the ill-effects of over-compression. This is done by maintaining an additional Gaussian mixture model (termed as detailed model) for each component in the $p_s(\mathbf{x})$, and comparing the errors for both models. Components having higher contribution to the total error are then replaced by two components of its detailed model. The process is repeated at the arrival of each data sample. Even though the original method is meant for online applications where the data is assumed to arrive in a sequential manner, we use this for our off-line estimation scenario (all filtered particle are used at once).

The output of the KDE in our contexts is a GMM, fitted to the posterior particles. We term this scheme as KDE-GMM.

3.4.4.3 Redrawing Algorithm

The purpose of re-drawing is to reduce the spread of the particles and relocate them in the appropriate region of the state space. We use the *Mahalanobis distance* (δ_M) for deciding the *waywardness* of the particles. Given $\mathcal{N}(\mathbf{x}_{k+1}|\bar{\boldsymbol{\mu}}_{k+1}, \bar{\mathbf{P}}_{k+1})$, the MVG approximation to the posterior, the distance (δ_M) for the posterior particle $\bar{\mathbf{x}}^i_{k+1}$ is given by,

$$\delta_M^{MVG}(i) = (\bar{\mathbf{x}}^i_{k+1} - \bar{\boldsymbol{\mu}}_{k+1})^T \bar{\mathbf{P}}^{-1}_{k+1} (\bar{\mathbf{x}}^i_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}) \tag{3.53}$$

We define a similar measure for the GMM model with K components $\sum_{l=1}^{K} w^l \mathcal{N}(\mathbf{x}_{k+1}|\bar{\boldsymbol{\mu}}^l_{k+1}, \bar{\mathbf{P}}^l_{k+1})$, such that,

$$\delta_M^{GMM}(i) = \sum_{l=1}^{K} w^l \left(\bar{\mathbf{x}}^i_{k+1} - \bar{\boldsymbol{\mu}}^l_{k+1}\right)^T [\bar{\mathbf{P}}^l_{k+1}]^{-1} \left(\bar{\mathbf{x}}^i_{k+1} - \bar{\boldsymbol{\mu}}^l_{k+1}\right) \tag{3.54}$$

where w^l , $\bar{\boldsymbol{\mu}}^l$ and $\bar{\mathbf{P}}^l$ are the weight, mean and covariance of the *kth* component of the GMM estimated through the online-KDE.

Algorithm 5 Redrawing Algorithm

procedure REDRAWPARTICLES($\{\bar{\mathbf{x}}^i_{k+1}\}_{i=1}^{N_p}$)
  $\varsigma(i) = \frac{1}{\delta^*_M(i)} \quad \forall i$
  $\Upsilon = \frac{1}{\left(\sum_{i=1}^{N_p} \frac{\varsigma(i)}{\sum_{i=1}^{N_p} \varsigma(i)}\right)^2}$
  if $\Upsilon \leq \nu_M \cdot N_p$ **then**
    if $\delta^*_M(i) \geq \sqrt{\frac{\Upsilon}{N_p}} \cdot \max \delta^*_M \quad \forall i$ **then**
      Redraw from $\mathcal{N}(\mathbf{x}_{k+1}|\bar{\boldsymbol{\mu}}_{k+1}, \bar{\mathbf{P}}_{k+1})$ or $\sum_{l=1}^{K} w^l \mathcal{N}(\mathbf{x}_{k+1}|\bar{\boldsymbol{\mu}}^l_{k+1}, \bar{\mathbf{P}}^l_{k+1})$
    else
      NoRedraw
    end if
  else
    NoRedraw
  end if
  return $\{\bar{\mathbf{x}}^{i,redraw}_{k+1}\}_{i=1}^{N_p}$
end procedure

The inverse of the distance $\varsigma = 1/\delta^*_M$ is a measure of the closeness of a particle to the estimated mean value. We use this value as a sort of weight ascribed to the particle, such that a particle close to the mean value is assigned a higher weight and vice versa. These weights are then normalized. Next, the particle *assemblage*, denoted as Υ is calculated. Υ has the same form as the effective sample size (ESS), in the traditional particle filter, and is a measure of the particle spread about the mean value. A higher value of Υ indicates an relatively even spread of the particles about the mean, whereas a lower value might suggest fragmentation of the

particles into sub-clusters. A detailed analysis of this measure is presented in the Appendix 3.B. Redrawing takes place only when the assemblage falls below a certain value, which equals $\nu_M \cdot N_p$. We call ν_M as the *redrawing intensity* and its value can be set to any value between 0 and 1. When ν_M is 0, redrawing never takes place, while redrawing happens surely for the value 1. In [KU15], we redrew the whole set of posterior particles, when the redrawing criterion was met. Here we make a small change and redraw only certain particles, which are deemed too off the main cluster(s). For that purpose, we compare δ_M^* for each particle against a certain threshold which is dependent on the assemblage. If the criterion is met, the particle is redrawn from the MVG or GMM. The procedure is summarized in Algorithm 5.

3.5 Improved DHF implementation

Finally, we describe the revised DHF implementation. The whole procedure is shown in the algorithmic form in the Algorithm 6.

Algorithm 6 Improved DHF implementation

procedure LOGHOMOTOPYFLOWFILTER($\{\hat{\mathbf{x}}_0\}_{i=1}^{N_p}$)
  for k = 0 : $k_{max} - 1$ **do**
    for i = 1 : N_p **do**
      $\hat{\mathbf{x}}_{k+1}^i = \phi_{k+1}(\bar{\mathbf{x}}_k^i) + \mathbf{w}_{k+1}$ ▷ Propagate the particles in real time
    end for
    $\hat{\mathbf{P}}_{k+1}$ = SHRINKAGEESTIMATOR($\{\hat{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$) ▷ Prior cov. estimation
    $\{\Delta\lambda_j, \lambda_j\}_{j=1}^{N_\lambda}$ = PSEUDOTIMEDISCRETIZATION(N_λ) ▷ λ grid
    for i = 1 : N_p **do**
      $\mathbf{y}_0 = \hat{\mathbf{x}}_{k+1}^i$ ▷ Temporary variable
      for j = 1 : N_λ **do**
        $\mathbf{y}_j$ = FLOWINTEGRATION($\mathbf{y}_{j-1}, \lambda_j, \mathbf{z}_{k+1}, \Delta\lambda_j$) ▷ Num. integration
      end for
      $\bar{\mathbf{x}}_{k+1}^i = \mathbf{y}_{N_\lambda}$
    end for
    Approximate $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ either through a MVG or a GMM
    REDRAWPARTICLES($\{\bar{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$) ▷ Redraw particle
  end for
end procedure

The procedure starts with an initial particle set, which is propagated through the real time. After that, the shrinkage estimator is used to form the Hessian estimate of the log prior density, while gradient and Hessian of the log of likelihood are also evaluated. The next task then becomes to numerically solve the flow equation over the pseudo-time range [0,1], to get the filtered particle set. This is further followed by an optional particle redrawing step, after which one iteration of the particle flow based filtering is completed. Here, $\{\hat{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$ and $\{\bar{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$ are the set of prior and posterior particles respectively.

3.6 Numerical Example: Model description

Here, we consider a scenario similar to the one described in [KU14], namely the tracking of multiple targets in a 2D space using range and bearing measurements, in order to study the effects of the methods proposed in the previous sections. States of targets are interdependent, therefore resulting in a nonlinear coupled dynamical model. Furthermore, the target association is assumed to be perfectly known and hence we do not use any data association algorithm. The state vector for the target *i* at time instant *k* is $\mathbf{x}_k^{(i)} = (x_k^{(i)}, y_k^{(i)}, \dot{x}_k^{(i)}, \dot{y}_k^{(i)})$, where $x_k^{(i)}$ and $y_k^{(i)}$ represent the position while $\dot{x}_k^{(i)}$ and $\dot{y}_k^{(i)}$ represent velocity components along the x and y-axis respectively. The overall state vector is formed by concatenating the individual target state vectors $\mathbf{x}_k = [\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} \dots \mathbf{x}_k^{(N)}]$. Also, the measurement vector for the target *i* is given by $\mathbf{z}_k^{(i)} = (r_k^{(i)}, \theta_k^{(i)})$, where $r_k^{(i)}$ is the range to the target while $\theta_k^{(i)}$ is the target bearing. The overall measurement vector at time *k* is generated in a similar way. The process model is described in equations (D1), where $a_{x_{k+1}}$ and $a_{y_{k+1}} \sim \mathcal{N}(0, \sigma_a^2)$, Δt is the time discretization step size and N is the total number of targets. The intuition behind the model is to make the motion of the targets coupled to each other. The target ($i = 1$) is pursued by all other targets ($i > 1$). The changes in the speed and direction of the targets depend on their relative distances to each other. κ_1, κ_2 and κ_3 are the coupling constants in the model. $\Pi_{x_k}^1$ and $\Pi_{y_k}^1$ control the speed/direction change for the pursued target and is inversely proportional to the sum of its relative distances to the all others. As pursuers come close, the pursued target speed is increased followed by a directional change. The direction change is realized through terms $\frac{v_t^2}{r_t}\cos(\frac{v_t}{r_t}k)$ and $\frac{v_t^2}{r_t}\sin(\frac{v_t}{r_t}k)$. r_t and v_t are the turning radius and velocity respectively and δ is a small offset. Similarly, the speed and direction changes for the pursuers are controlled by the terms $\Pi_{x_k}^i$ and $\Pi_{y_k}^i$.

$$
\begin{aligned}
x_{k+1}^i &= x_k^i + \dot{x}_k^i \Delta t + \frac{1}{2} a_{x_{k+1}} \Delta t^2 \\
y_{k+1}^i &= y_k^i + \dot{y}_k^i \Delta t + \frac{1}{2} a_{y_{k+1}} \Delta t^2 \\
\dot{x}_{k+1}^i &= \dot{x}_k^i + \Pi_{x_k}^i \Delta t + a_{x_{k+1}} \Delta t \\
\dot{y}_{k+1}^i &= \dot{y}_k^i + \Pi_{y_k}^i \Delta t + a_{y_{k+1}} \Delta t
\end{aligned} \tag{3.55}
$$

$$
\begin{aligned}
\Pi_{x_k}^1 &= \frac{1}{N-1} \sum_{i=2}^{N} \Big(\frac{\kappa_1}{\sqrt{(x_k^1 - x_k^i)^2 + (y_k^1 - y_k^i)^2} + \delta} \Big) \frac{v_t^2}{r_t} \cos(\frac{v_t}{r_t}k) \\
\Pi_{y_k}^1 &= -\frac{1}{N-1} \sum_{i=2}^{N} \Big(\frac{\kappa_1}{\sqrt{(x_k^1 - x_k^i)^2 + (y_k^1 - y_k^i)^2} + \delta} \Big) \frac{v_t^2}{r_t} \sin(\frac{v_t}{r_t}k) \\
\Pi_{x_k}^i &= \kappa_2 (x_k^1 - x_k^i) - \kappa_3 \dot{x}_k^i \\
\Pi_{y_k}^i &= \kappa_2 (y_k^1 - y_k^i) - \kappa_3 \dot{y}_k^i
\end{aligned} \tag{3.56}
$$

If κ_1, κ_2 and κ_3 are set to zero, then state dynamics corresponds to the standard Discrete white noise acceleration (DWNA) model. Measurements consist of ranges and angles of the two

object types, target and the pursuer.

$$\begin{aligned} r^i_{k+1} &= \sqrt{(x^{(i)}_{k+1})^2 + (y^{(i)}_{k+1})^2} + v^i_{r_{k+1}} \\ \theta^i_{k+1} &= \tan^{-1}\Big(\frac{y^{(i)}_{k+1}}{x^{(i)}_{k+1}}\Big) + v^i_{\theta_{k+1}} \end{aligned} \tag{3.57}$$

We consider two measurement models, one with uncorrelated Gaussian noises for both range and the angle, while the other has correlated Gaussian range noise and exponentially distributed angle noise. For the first model, the likelihood is given by (3.58). We assume that both range and bearing measurement noise $v_{\theta_{k+1}}$ vectors are mutually independent at each time step. Also, both noises are uncorrelated within themselves such that $\mathbb{E}[v^i_{r_{k+1}} v^j_{r_{k+1}}] = 0$ and $\mathbb{E}[v^i_{\theta_{k+1}} v^j_{\theta_{k+1}}] = 0$ for $i \neq j$, In the non-Gaussian measurement model given in (3.59), range measurement noises $v_{r_{k+1}} \sim \mathcal{N}(\mathbf{0}, \mathbf{R_r})$ are mutually correlated but are independent w.r.t.the bearing measurement noises $v_{\theta_{k+1}}$. Bearing measurement noise elements $v^i_{\theta_{k+1}}$ are exponentially distributed with the scale parameter β, such that $\mathbb{E}[(v^i_{\theta_{k+1}})^2] = \beta^2$ and $\mathbb{E}[v^i_{\theta_{k+1}} v^j_{\theta_{k+1}}] = 0$. $\mathbf{R_r}$ represents the covariance matrix of $v_{r_{k+1}}$ with $\sigma_r^2 = \mathbb{E}[(v^i_{r_{k+1}})^2]$ and $\sigma^2_{r_x} = \mathbb{E}[v^i_{r_{k+1}} v^j_{r_{k+1}}]$. $\sigma^2_{r_x}$ is assumed to be the same for any two targets. Measurement noises are chosen in this particular way in order to create a challenging estimation scenario, in which the relative strength of the particle flow method can be tested against the more traditional solutions like the EKF and the particle filter.

Gaussian noise:

$$\begin{aligned} p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) &= p(\mathbf{r}_{k+1}|\mathbf{x}_{k+1}) p(\theta_{k+1}|\mathbf{x}_{k+1}) \\ &= \frac{1}{\Big(2\pi\sigma_r\sigma_\theta\Big)^N} \prod_{i=1}^{N} \exp\Big\{ -\frac{1}{2\sigma_r^2}\Big(r^{(i)}_{k+1} - \sqrt{(x^{(i)}_{k+1})^2 + (y^{(i)}_{k+1})^2}\Big)^2 \\ &\quad - \frac{1}{2\sigma_\theta^2}\Big(\theta^{(i)}_{k+1} - \tan^{-1}(\frac{y^{(i)}_{k+1}}{x^{(i)}_{k+1}})\Big)^2 \Big\} \end{aligned} \tag{3.58}$$

Non-Gaussian noise:

$$\begin{aligned} p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) &= p(\mathbf{r}_{k+1}|\mathbf{x}_{k+1}) p(\theta_{k+1}|\mathbf{x}_{k+1}) \\ &= \frac{1}{\Big(2\pi\beta^2\Big)^{\frac{N}{2}} |R_r|^{\frac{1}{2}}} \exp\Big\{ -\frac{1}{2}\big(\mathbf{r}_{k+1} - \tilde{\mathbf{r}}_{k+1}\big)^T R_r^{-1} \big(\mathbf{r}_{k+1} - \tilde{\mathbf{r}}_{k+1}\big) \Big\} \\ &\quad \times \prod_{i=1}^{N} \exp\Big\{ -\frac{1}{\beta}\Big(\theta^{(i)}_{k+1} - \tan^{-1}(\frac{y^{(i)}_{k+1}}{x^{(i)}_{k+1}})\Big) \Big\} \end{aligned} \tag{3.59}$$

where the range measurement vector is given as,

$$\tilde{\mathbf{r}}_{k+1} = \Big[\sqrt{(x^{(1)}_{k+1})^2 + (y^{(1)}_{k+1})^2} \;\; \sqrt{(x^{(2)}_{k+1})^2 + (y^{(2)}_{k+1})^2} \cdots \sqrt{(x^{(N)}_{k+1})^2 + (y^{(N)}_{k+1})^2}\Big]^T$$

with the associated covariance matrix,

$$R_r = \begin{bmatrix} \sigma_r^2 & \sigma_{r_x}^2 & \cdots & \sigma_{r_x}^2 \\ \sigma_{r_x}^2 & \sigma_r^2 & \cdots & \sigma_{r_x}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{r_x}^2 & \sigma_{r_x}^2 & \cdots & \sigma_r^2 \end{bmatrix}$$

3.6.1 Parameters setting

We simulate two targets (N =2) in our analysis. Δt is set to 1, σ_a^2 to 0.5 $\mathrm{m}s^{-2}$, σ_r^2 is set to $2000m^2$, $\sigma_{r_x}^2$ to $\frac{3}{10}\sigma_r^2$, while β^2 is set to $\frac{1}{10}\mathrm{rad}^2$. We note that $\sigma_r < D_{i,k}\sigma_\theta \ \ \forall i, k$, where $D_{i,k}$ represents the distance of *ith* target from the radar location at time instant k.

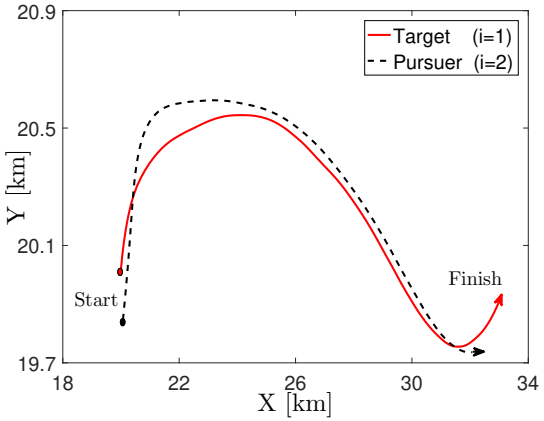


Figure 3.3: Sample trajectory

In this paper, we work only with the strongly coupled model with coupling constants κ_1, κ_2 and κ_3 set to 8000, 0.01 and 0.1 respectively. The turn radius r_t and turn speed v_t are set to 200 m and 10 $\mathrm{m}s^{-1}$ while δ is set to 0.001. We use 100 DHF particles (N_p = 100). DHF and SIR-PF particles are initialized by sampling a Gaussian distribution with mean of 20000 m and variance of 5000 m^2 for position elements, while their velocities are sampled from a Gaussian distribution with mean and variance of 5 ms^{-1} and 25 m^2s^{-2} respectively. EKF is initialized by sampling the Gaussian with initial state vector as mean and with variances 10^4 and 1 for the position and the velocity respectively. In Figure 3.3, we show a sample trajectory generated by using these parameters. We note that the target object (i=1) is pursued by the pursuing object (i=2). The target turns and increases speed as it is approached by the pursuer. The trajectory has segments of straight run as well as turns in the middle and at the end. Turning, in particular is challenging for the estimation algorithm, as this in addition to the nonlinearity in the measurements, introduces nonlinearity in the process model as well.

3.7 Numerical Example: Results

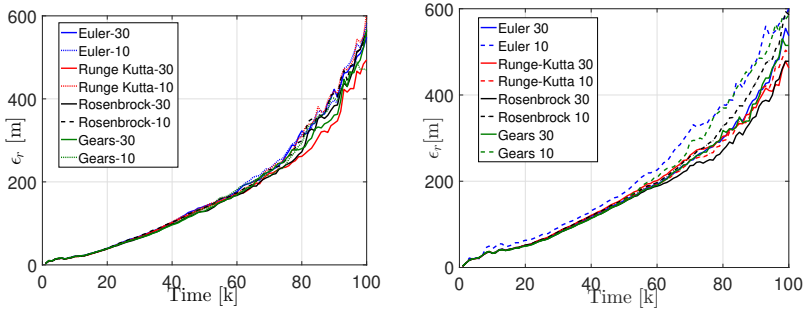
We use root average mean square error (RAMSE) as the performance metric. It is defined as follows: Let M be the total number of simulation runs for a particular scenario, $x_k^{i,m}$ and $y_k^{i,m}$ denote the positions of the *ith* target along X and Y-axis respectively, at time instant k in the mth trial. Likewise, let $\hat{x}_k^{i,m}$ and $\hat{y}_k^{i,m}$ denote the estimated positions for the *ith* target. The RAMSE ε_r is then defined as,

$$\epsilon_r(k) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{1}{2N}\sum_{i=1}^{N}\left(\left(x_k^{i,m}-\hat{x}_k^{i,m}\right)^2+\left(y_k^{i,m}-\hat{y}_k^{i,m}\right)^2\right)\right]}$$

We simulate each scenario for a total of fifty times ($M = 50$). First, we describe the effect of the numerical integration schemes.

3.7.1 Effect of numerical integration schemes

We compare the performance of the four methods mentioned in subsection 3.4.2, namely Euler's method, Runge-Kutta scheme of fourth order, Rosenbrock formula of second order and Gear's method. While we wrote scripts for the first two methods, MATLAB provided functions *ode23s* and *ode15s* were used for the Rosenbrock and the Gear's methods respectively. We also compare the effect of grid discretization on the performance of the above schemes. We use two specific cases, 10 uniformly spaced pseudo-time points (coarse discretization) and 30 exponentially spaced points (fine discretization). We plot the RAMSE ϵ_r for different schemes in Figures 3.4a and 3.4b.



(a) (b)

Figure 3.4: Comparison of numerical integration schemes for the (a) Gaussian noise (b) non-Gaussian noise.

We see a general increasing trend in the RAMSE vs. time for all methods. This is due to the specific process model used, which results in peculiar target trajectories involving rapid accelerations and sharp turns. It can be observed that the difference in the performance of different integration schemes is more pronounced in the case with non-Gaussian noise, as evident by the wider spread in the error curves. For the Gaussian case, we note that the Runge-Kutta method with 30 λ points has the lowest error. Among the integration methods with 30 discretization

points, Euler's method has the highest error. We can also note that Gears-10 has the lowest error for all methods employing 10 discretization points. The largest error is exhibited by the Euler-10 method, which happens to be the fastest. On the other hand, Rosenbrock-30 is the slowest of all the methods. Euler-30 ranks second in the processing speed, as it is almost 1.5 times faster than its nearest competitor Runge-Kutta-10, while being 3 times as fast as Gears-10 though slightly inferior in the performance. Next, we discuss the results for the model with non-Gaussian measurement noise. As discussed earlier, the error curves show more spread.

Gaussian		
Method	Avg. ϵ_r [m]	Proc.time (pp) [ms]
Euler-30	178.45	6.6
Euler-10	181.70	2.3
Runge-Kutta-30	163.06	27.4
Runge-Kutta-10	180.60	9.1
Rosenbrock-30	169.66	80.5
Rosenbrock-10	178.04	62.8
Gear-30	169.42	27.4
Gear-10	172.37	19.3
Non-Gaussian		
Method	Avg. ϵ_r [m]	Proc.time (pp) [ms]
Euler-30	186.69	5.6
Euler-10	223.07	1.8
Runge-Kutta-30	181.68	38.5
Runge-Kutta-10	184.39	12.7
Rosenbrock-30	173.17	71.9
Rosenbrock-10	196.30	55.8
Gear-30	184.49	26.4
Gear-10	186.69	17.9

Table 3.1: Comparison for different integration schemes

We note that the Rosenbrock method with 30 λ points has the lowest RAMSE, while the Euler's scheme with 10 λ points is the worst performer followed closely by the Gear-10. Runge-Kutta methods with both 10 and 30 points are the second best. In fact, the difference in the performance between the two is very small. This is followed by the Gear-30 and the Euler-30 methods. We tabulate the time averaged RAMSE and the average processing time per particle

for all methods in Table 3.1. Note that the time values mentioned only represent the time spent while solving the homotopy ODE for a single particle. The largest and the smallest values are highlighted in red and green respectively. It can be seen that while the Rosenbrock-30 is the best method, it is also computationally most expensive. On the other hand, the Euler-10 is the fastest but the worst performer of all methods. Euler-30 represents a reasonable trade-off between the performance and the processing time. While it is slightly worse than the Runge-Kutta and the Gear-30, it is approximately six times faster than the Runge-Kutta-30 and more than twice as fast as Runge-Kutta-10. In the proceeding analysis, we use Euler-30 as the default integration scheme.

3.7.2 Effect of shrinkage covariance estimation

Next we analyze the effect of shrinkage estimation schemes. We compare the performance of the six methods mentioned in subsection 3.4.3, together with that of sample covariance and the prior covariance matrices S and P_{EKF} respectively. We denote the DHF estimate generated using a particular covariance estimation scheme X as DHF-X. We use four metrics to judge the effectiveness of these methods. The first and the foremost is the RAMSE of the DHF estimates. This is the central criterion for judging the effectiveness of the shrinkage schemes, in terms of the accuracy of the DHF estimates. The second metric is the relative accuracy of the covariance matrix estimates themselves. In the context of shrinkage estimation, we use the percentage relative improvement in average loss or PRIAL as the measure for the exactness of any shrinkage covariance estimate, as defined in [LW04b],

$$\text{PRIAL} = \left(1 - \frac{\mathbb{E}[||P_{(.)} - P||^2]}{\mathbb{E}[||S - P||^2]}\right) \times 100 \tag{3.60}$$

where $||(.)||$ represents the Frobenius norm, S is the sample covariance matrix estimate, while $P_{(.)}$ and P are the shrinked covariance and the true covariance estimates, respectively.

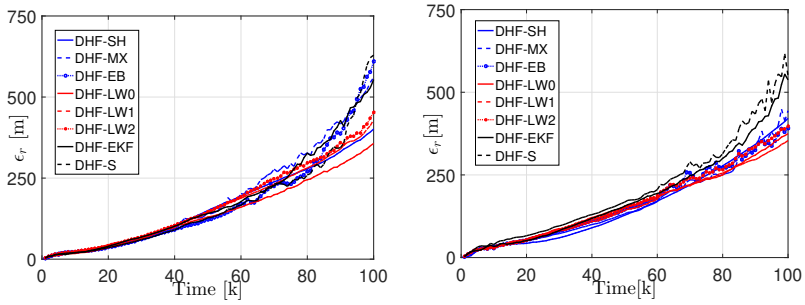


Figure 3.5: Comparison of covariance estimation schemes for the (a) Gaussian noise (b) non-Gaussian noise.

As P is not known, in the current scenario it is approximated by the covariance estimate from a sampling importance resampling particle filter (SIR-PF) with 25000 particles. The third metric is the shrinkage intensity ρ, which indicates the compromise between the unbiased but more

variant sample based estimate and the biased but less variant target. A lower value of ρ represents the closeness of the covariance estimate to the sample covariance matrix S. On the other hand, a higher value highlights a stronger influence of the target matrix B. At last, we use the condition number k_{cond} to analyze the spread in the eigenvalues of covariance estimates over the time. Figures 3.5a and 3.5b show the comparison of RAMSE for different covariance estimation schemes for the Gaussian and non-Gaussian noises, respectively. First, we discuss the RAMSE for DHF with covariance estimates from all methods, for the Gaussian noise model. DHF-MX (Minimax) has the highest error. This can be explained as follows: the Minimax estimator scales the eigenvalues of the sample covariance matrix in a nonlinear fashion. The highest $\lfloor \frac{p-1}{2} \rfloor$ eigenvectors have their eigenvalues shrinked, while for the others the eigenvalues are expanded. Scaling is done just based on the order of the sorted eigenvalues and it does not take into account any other possible information in the structure of the matrix S. This simplicity renders the estimator performing worse as compared to the others. Next in the line is the DHF-EKF. As can be seen in Figures 3.5 a & b, the error increases sharply after about 80s. Although each simulated trajectory is not exactly the same, this is roughly the time when the targets start turning in our coupled motion model in most of those runs. Hence this is a critical point, as this tend to increase the nonlinearity in our motion model. We see that for the DHF based on the EKF prior covariance, rising error indicates a failure in the proper tracking.

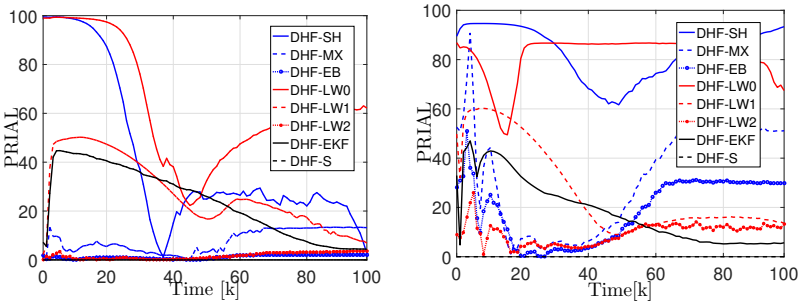


Figure 3.6: PRIAL of covariance estimation schemes for the (a) Gaussian noise (b) non-Gaussian noise.

This also proves to be a very strong motivation for the search of a covariance estimation scheme that is better than P_{EKF}. Interestingly, the performance of sample covariance based DHF is better than many other schemes. In fact, for most of the simulation time it has an error comparable to the better performing DHF-LW0. It starts to increase only when targets start turning. After that time, the DHF-S fails to properly cope with the induced process nonlinearity and the filter diverges rapidly. All variants of Ledoit-Wolf covariance estimators perform better, with LW0 based DHF outperforming all other filters. This can be attributed to the optimal convex combination (asymptotically) of the sample covariance matrix S and the scaled identity matrix I. This structure of the estimator results in a well-conditioned covariance estimator, that is more stable (from an inversion point of view). This property can be critical when considering the turning motion of the targets, as DHF particles can be flung far and wide if the flow is incorrect which of course depends on inverting the prior covariance matrix. DHF with the other

two covariance estimators by Ledoit and Wolf perform a little inferior relative to the DHF-LW0. P_{LW1} and P_{LW2} were derived for special problems in portfolio estimation and have very special structures. This lessens their generality and makes them very application specific. Next we discuss the non-Gaussian case. We note that DHF-S is the worst method. DHF-EKF comes next as its error also shows steeply diverging trend. This can be explained as follows: given that the measurements are nonlinear functions of state variables, and bearing noise is exponentially distributed, the EKF is not a good approximation for the resulting nonlinear and non-Gaussian scenario. Hence the covariance estimates generated by the EKF will not be accurate. DHF-LW0 has the lowest average error amongst all methods. This is because P_{LW0} is a distribution free estimator, and hence produces good estimates even in this non-Gaussian scenario. It is followed by the Stein-Haff and Minimax estimators. Compared with the DHF-EKF, all estimators except the sample covariance DHF-S have lower average RAMSE.

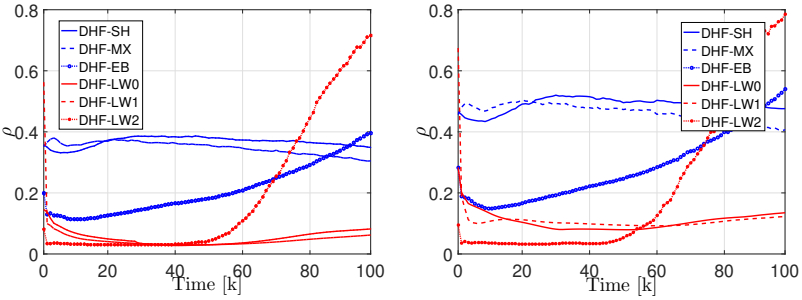


Figure 3.7: Shrinkage intensity of covariance estimation schemes for the (a) Gaussian noise (b) non-Gaussian noise.

Next we discuss the PRIAL for the covariance estimates. The expectation in formula (3.60) is calculated by averaging over all simulation runs. A value of 100 means perfect estimation accuracy, while 0 means accuracy as good as the sample covariance matrix S. Again, we discuss the Gaussian case first based on the results in Figures 3.6 a & b . We note that the PRIAL for P_{LW0} is highest while it is lowest for P_{LW2} Again, this can be attributed to the very specific structure of this estimator. For the non-Gaussian case, we note that the PRIAL for P_{SH} is the highest on the average , while is lowest for the P_{LW2}. One noteworthy thing is to compare the PRIAL of the estimators in the Gaussian vs. non-Gaussian case. We see that the PRIAL, on average, is lower for the Gaussian case. This can be explained by the fact that PRIAL represents how better a particular estimator is compared to the sample covariance estimator S. In non-Gaussian case, DHF-S is worse performer, which points to the fact that S is not a well-suited estimator. In fact, all DHF are better than DHF-S. Hence we see that the PRIAL for the estimators in the non-Gaussian case is significantly higher. On the other hand in the case of Gaussian noise, S is not the worst estimator. This increases the ratio $\frac{\mathbb{E}[||P_{(.)}-P||^2]}{\mathbb{E}[||S-P||^2]}$, which results in the lower values of PRIAL. Shrinkage intensities ρ are shown in Figures 3.7 a & b. We note that the lowest shrinkage intensity in both cases is exhibited by P_{LW0}. This suggests more contribution of the sample covariance than the scaled identity matrix. P_{SH} has the highest shrinkage intensity on average and is also the most consistent. Shrinkage intensities in the non-

Gaussian case are higher, again suggesting the inadequacy of the sample covariance matrix in the nonlinear/non-Gaussian scenario. Finally we discuss the average logarithmic condition number $\log k_{cond}$. As expected, P_{LW0} has the lowest condition number over time, at least two orders of magnitude smaller than all other estimators. Also, S has the highest condition number. For the subsequent analysis, we consider P_{LW0} as the default covariance estimation scheme.

Gaussian				
Method	Ave. ϵ_r [m]	Ave.PRIAL	Ave. ρ	Ave. k_{cond}
Stein-Haff	164.29	41.34	0.36	38620
Minimax	188.03	7.86	0.34	272820
Emp.Bayesian	170.94	1.20	0.20	181380
Ledoit-Wolf-0	144.77	63.13	0.05	170
Ledoit-Wolf-1	163.05	27.26	0.05	55610
Ledoit-Wolf-2	171.10	1.63	0.19	60370
EKF covariance	179.23	23.79	0	71460
Sample covariance	168.09	0	0	139760
Non-Gaussian				
Method	Ave. ϵ_r [m]	Ave.PRIAL	Ave. ρ	Ave. k_{cond}
Stein-Haff	161.22	83.30	0.40	45080
Minimax	166.58	32.0711	0.38	55490
Emp.Bayesian	171.28	18.78	0.23	46730
Ledoit-Wolf-0	153.32	81.71	0.09	180
Ledoit-Wolf-1	161.92	27.01	0.09	53220
Ledoit-Wolf-2	171.27	9.38	0.21	48470
EKF covariance	189.80	15.15	0	67770
Sample covariance	213.41	0	0	142260

Table 3.2: Comparison for different covariance estimation schemes

3.7.3 Effect of Redrawing

Once decided upon the pseudo-time discretization, flow integration and prior covariance estimation schemes, we now study the effect of redrawing on the performance of the DHF. As mentioned in section 3.4.4, we consider two methods for regenerating the particles. The first method is redrawing from a multivariate Gaussian (MVG), and the other from a Gaussian mixture model fitted to the posterior particles, that is estimated using an online kernel density estimation method. First, we discuss the redrawing from MVG.

3.7.3.1 Multivariate Gaussian

We follow the Algorithm 5 mentioned in section 3.4.4 for redrawing. The main parameter in that algorithm is the redrawing intensity ν_M. We vary ν_M between 0 and 1 and use five distinct values. First we study the effect of ν_M on the estimation accuracy, for which we plot the time averaged RAMSE for both Gaussian and the non-Gaussian cases in Figure 3.8a.

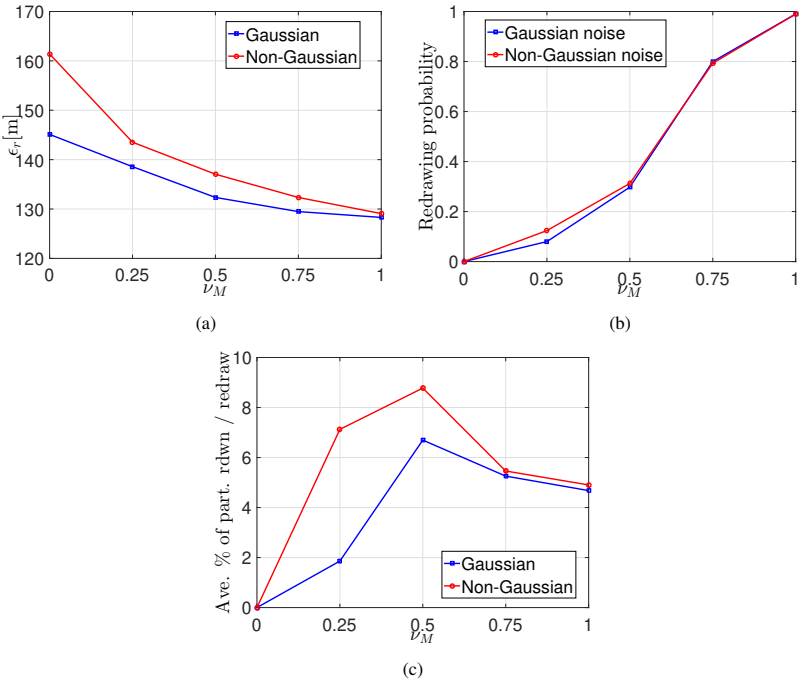


Figure 3.8: (a) Time averaged RAMSE vs. ν_M for Gaussian and non-Gaussian models (b) Redrawing probability vs. ν_M for Gaussian and non-Gaussian models, (c)Average percentage of particles redrawn vs. ν_M for Gaussian and non-Gaussian models.

We see that as the redrawing threshold is increased the error decreases monotonically: the lowest error is for $\nu_M = 1$. We note that the improvement in the performance by increasing ν_M is stronger in the non-Gaussian case than in the Gaussian one. This suggests the presence of

more wayward particles in the non-Gaussian case, which are subsequently moved to the right regions after being redrawn. Next, in Figure 3.8b, we plot the redrawing probability vs. redrawing intensity. Redrawing probability is defined as the number of times particles are redrawn in the simulation divided by the total simulation time. So if particles are redrawn for half of the whole simulation duration, the redrawing probability is 0.5. The value is averaged over all simulation runs. A higher value indicates a higher chance for particles to be redrawn during the simulation. We note a monotonically increasing relation between ν_M and the redrawing probability, which assumes a value of 1 for ν_M equals 1. This plot can also be used to infer about the assemblage Υ. As ν_M is increased, the probability of finding Υ below the threshold $\nu_M \cdot N_p$ increases. e.g. from Figure 3.8b, it can be inferred that almost 30% of the time Υ value is below $0.5N_p$. This suggests that the probability of having fragmentation of particles about the mean into two sub-groups of equal sizes (or any other equivalent scenario resulting in Υ = 0.5) is non-negligible. Also almost 50% of the time the value of the assemblage is between 50 and 75, while it is between 75 and 100 for almost 20% of the time. In relation to the RAMSE, we can conclude that the redrawing frequency has a direct positive effect on the estimation error. A higher redrawing probability leads to the reduced estimation error. We note that both the Gaussian and non-Gaussian cases have a similar trend.

But how many particles, on average, are redrawn at any time instance during the whole simulation time? While several metrics can be used for this effect, we use in particular the average percentage of particles redrawn, further averaged over the simulation time as plotted in Figure 3.8c. We see an interesting trend: The percentage of particles redrawn increases with the increase in the intensity ν_M up to 0.5, at which it hits the maximum 7%-9% of the particles for both cases. Then this value decreases. This can be explained in the light of the redrawing probability. For ν_M between 0 and 0.5, the redrawing probability increases and so does the percentage of redrawn particles. This suggests that even though the assemblage can be expected to be below $0.5N_p$ about 30% of the time, at times there is a significant number of particles satisfying the redrawing condition $\delta^*_M(i) \geq \sqrt{\frac{\Upsilon}{N_p}} \cdot \max \delta^*_M$. That is why the redrawing criterion $\Upsilon \leq \nu_M \cdot N_p$ is met in the first place, given the low value for ν_M. As ν_M is increased beyond 0.5, the redrawing probability increases, but the average number of particles satisfying the redrawing conditions decreases. That also points to the increase in the assemblage. We note that, on average, more particles are redrawn in the case of non-Gaussian noise than in the Gaussian case. This result is expected as estimation under the non-Gaussian noise is more challenging.

When seen together with the estimation error, we note that although the average rate of particles redrawn at any given time is not more than 10%, but redrawing those particles amounts to a significant reduction in the error. Also, the particles redrawn for ν_M equal 1 have the maximum effect on the estimation error as they are the few well separated from the rest of the particle cluster(s). If redrawn, they are moved to the correct region of the state-space, and hence contributing effectively to the point estimates.

3.7.3.2 Kernel Density Estimation

Now, we discuss the effect of redrawing particles from a GMM, estimated through the online KDE (oKDE) as described in [KLS11], using Algorithm 5. We have used the source code for the oKDE provided by the authors at [KLS16]. Although the method is general and can be used with any estimation kernel, the authors have used a multivariate Gaussian kernel in their work.

The oKDE method fits a GMM to the online data, which is supposed to arrive sequentially. In our context, we use the oKDE method to approximate the density of the particles after they move through the pseudo-time loop. Hence those particles can be thought of as coming from an importance sampler, and the task is to estimate the corrected posterior distribution. As a result we get an ensemble of weights, mean and covariances, $\{w_k, \mu_k, \Sigma_k\}\}_{k=1}^{K}$. Next, the averaged distance of each particle given the estimated GMM is calculated, and those particles which are thought to be too wayward are redrawn. As in the MVG case, we vary the redrawing threshold ν_M between 0 and 1.

There are two parameters that control the degree of estimation accuracy: the error threshold D_{th}, which controls the number of Gaussian components fitted to the data, and N_{init} which defines the number of data samples used for the initialization. Through experiments, we have found out that N_{init}, above a certain value, does not strongly influence the estimation accuracy. Therefore in our study we have kept N_{init} fixed to 33 (one third of total number of particles), while the threshold D_{th} is varied between 0.3 and 0.7, in the steps of 0.1. In Figure 3.9a, we plot the average number of GMM components (K) vs. the error threshold D_{th}. We note that as D_{th} is increased, K decreases exponentially. This can be attributed to the particular implementation method used by the authors in [KLS16]. Next, in Figure 3.9b, we show results for RAMSE vs ν_M for various values of threshold D_{th}, for both Gaussian and non-Gaussian cases.

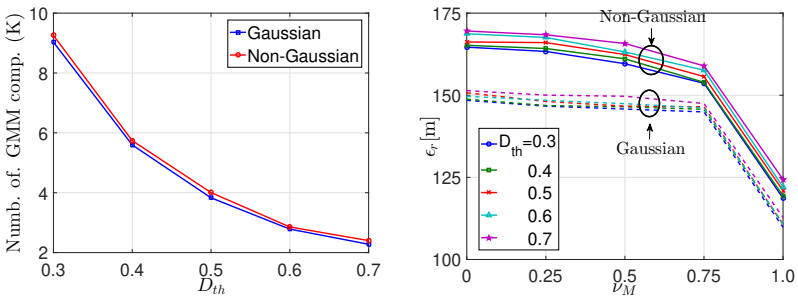


Figure 3.9: (a) Average number of GMM components vs. ν_M for Gaussian and non-Gaussian models, (b) RAMSE vs. ν_M for different values of D_{th} for Gaussian and non-Gaussian models.

There are a number of noteworthy things. First, we note that the error for the Gaussian cases is less than that for the non-Gaussian, for all values of ν_M. We saw a similar behavior in the previous section, where the redrawing was done using a MVG. This suggests that the posterior distribution is modeled more accurately by using the GMM for the Gaussian as compared to the non-Gaussian case. Secondly, we see that the error only slightly decreases with increasing ν_M up to 0.75. After that we observe a significant reduction in the error for both cases. This is explained in the following way: in contrast to redrawing from a MVG where the particles far from the estimated mean value had lower weight defined by the ς, here such particles can be softly assigned to more than one Gaussian components. And due to the relative weights of the GMM components, the contribution of those particles is lessened. This results in a higher assemblage Υ value, and hence the redrawing criterion is rarely met. But when ν_M is

sufficiently high, such that Υ is below $\nu_M.N_p$, redrawing takes place. Particles which meet the redrawing condition are redrawn using the GMM. Statistically, particles are more likely to be redrawn from the components with the higher weights, and hence making those components even stronger while the opposite happens to the original low weight components. As a result, one can expect a significant reduction in the particle spread after redrawing is done in this manner. Lastly, we observe that the error for a lower value of D_{th}(hence higher K) is lower for both cases, for all values of ν_M. Again this is intuitive, as a higher number of GMM components is suggestive of a better accuracy of the fitted distribution to the posterior particles.

Figure 3.10a shows the redrawing probability vs. ν_M.

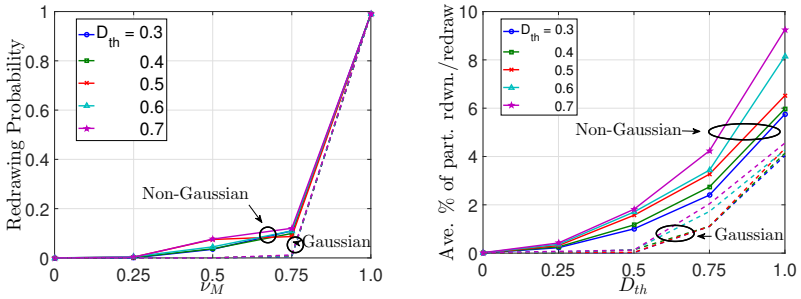


Figure 3.10: (a) Redrawing probability vs. ν_M for Gaussian and non-Gaussian models, (b) Average number of particles redrawn per redraw vs. ν_M for Gaussian and non-Gaussian models.

We use the same definition for this probability as used in the previous section. We note that the redrawing probability for both noise cases is almost zero for ν_M less than or equal to 0.25. Between ν_M 0.25 and 0.75, we see a slight increase for the non-Gaussian case while it is still very close to zero for the Gaussian case. E.g. at ν_M = 0.75, the redrawing probability is 10% for the case with non-Gaussian noise. A sharp rise can be seen for both cases between 0.75 and 1. Also the redrawing probability is higher for lower values of D_{th}. This trend has been explained in the previous paragraph, where it was mentioned that for higher assemblage values, the probability of redrawing is quite low. Hence, the redrawing probability also reveals information about the distribution of the assemblage. In contrast to the MVG case, the assemblage values are significantly larger but less spread. Therefore redrawing is only expected to happen for larger values of ν_M. Also, a higher D_{th} (smaller K) tends to make the assemblage lower, and therefore increasing the redrawing probability.

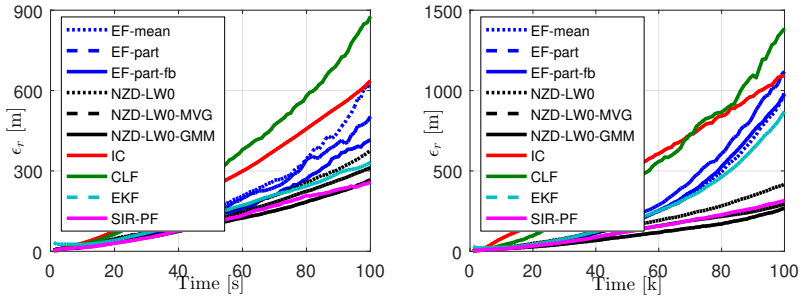
The average percentage of particles drawn per redraw is shown in Figure 3.10b. We observe a monotonically increasing trend for both Gaussian and non-Gaussian noises. We note that while the assemblage Υ value affects the redrawing probability, it is the distribution of the Mahalanobis distance itself that influences the average percentage of particles drawn per redraw. From the results we can infer that Mahalanobis distance distributions for both Gaussian and non-Gaussian noises are similar, although for the latter it is more skewed towards the right, as evident from the higher percentage of redrawn particles. The average percentage of particles drawn per redraw rises sharply for ν_M between 0.75 and 1, hence more particles are redrawn

for these values. This is correlated with the large drop in the estimation error. Altogether, it can be inferred that the redrawing done for ν_M between 0.75 and 1 significantly increases the estimation accuracy. It can also be concluded that the GMM provides a more accurate description for the posterior distribution. A higher percentage of particles is expected to be redrawn for higher values of D_{th} as the estimated GMM has fewer components, hence it is not accurate enough.

3.7.4 Comparison against other filters

In this subsection, we compare the performance of our modified DHF against the other versions of DHF mentioned in section 3.2, together with the EKF and sampling importance resampling particle filter (SIR-PF) with 25000 particles. In total, we present the results for eight different variants for DHF, out of which three are different flavors of exact flow filter (EF), three are variants of non-zero diffusion constrained flow based filters (NZDCF), while the other two are based on the incompressible flow (IC) and the Coulomb's law flow DHF (CLF) respectively. The most basic version of the exact flow based DHF is reported in [CWDH11], where the flow equation is solved by linearizing the measurement model about the estimated prior mean value. We call this implementation EF-mean. The second implementation of the exact flow has been reported by Ding and Coates in [DC12], and a pseudo-code is also provided. Two distinct changes are made to the EF-mean. In the first modification, the linearization of the measurement equation is carried out for individual particles, as opposed to being done only at the prior mean location. The second modification is related to the feedback of the DHF state estimates to the EKF, making the two filters coupled. In this study we consider these two cases individually i.e. the first modification alone, and it together with the feedback. We call these implementations EF-part and EF-part-fb respectively.

For the incompressible flow filter (IC), the flow equation (3.17) is solved for individual particles by assuming a Gaussian prior. Finally for the Coulomb's law based DHF (CLF), we use the parameters settings mentioned by the authors in [DHN11b]. One third of nearest neighbors are used in the evaluation of equation 3.33. We have found that this filter is very sensitive to the parameters settings, and in general is very hard to tune. First we plot the RAMSE for the different filters for the Gaussian case in Figure 3.11a.



(a) (b)

Figure 3.11: RAMSE comparison of different estimation methods with the (a) Gaussian noise, (b) non-Gaussian noise.

We note that the CLF has the largest error. The issue with this filter is the estimation of the probability density $p(\mathbf{x}, \lambda)$ for all particles throughout the pseudo-time, which is used in evaluating the flow equation $f(\mathbf{x}, \lambda) = \nabla \mathbf{V}(\mathbf{x}, \lambda)/p(\mathbf{x}, \lambda)$. As this is done using the few available particles, the resulting density estimate is not accurate enough and the filter is prone to diverge. This is also the issue with the Monte-Carlo approximation of the integral for the gradient $\nabla \mathbf{V}(\mathbf{x}, \lambda)$. Next, we see that DHF based on IC although being better than with CLF, still fares worse compared with all other filters. The filter is based on the assumption of zero-divergence, which appears to be a quite strict condition. Also, the flow might encounter singularities which can make the filter diverge. Among the three variants of the exact flow, the EF-part-fb is the best. This is expected as for this filter linearization is done about each particle, and also the filter is coupled to a parallel running EKF. The best among all DHF variants are the ones based on NZDCF, all of which use Euler integration with 30 time steps and LW0 covariance estimation scheme. We denote the DHF-NZDCF without redrawing by NZD-LW0, with redrawing from MVG by NZD-LW0-MVG, and the one with redrawing from GMM as NZD-LW0-GMM. For the redrawing we set the threshold ν_M equal to 1. Also for the oKDE, we set D_{th} to 0.5, which on average fits 4 GMM components to the posterior distribution. We note that the NZD-LW0-GMM is the best of all the schemes, even surpassing the SIR-PF with 25000. NZD-LW0-MVG is a little worse in performance to the SIR-PF, but is still better than the EKF.

Next we discuss the results for the non-Gaussian measurement noise, as plotted in Figure 3.11b. We note that all filters, except the variants of DHF-NZDCF and SIR-PF, perform poorly. DHF-IC and DHF-CLF fail to track the targets, with the latter being the worst in the performance. All variants of DHF-EF show a diverging error trend. This is due to the fact that EF hinges on the Gaussian assumption, which is not valid in the current case. As a part of the measurement noise is non-Gaussian, we see that these filters are unable to properly track targets. The same reasoning can be applied to EKF. NZD-LW0-GMM, NZD-LW0-MVG and SIR-PF are the first, second and the third best performer respectively. The error for all filters is generally larger when compared with the case with the Gaussian noise.

Next, we compare the execution time τ for a single update, including both the time and the measurement update steps. MATLAB simulations were performed on a computer with Intel

Core2 Quad with 2.66 GHz processors and 4 GB RAM. Table 3.3 shows the processing time per time step in seconds. We note that the EKF is the fastest of all methods. Next in the line are the EF based DHF, with DHF-EF-mean being the fastest. DHF with IC flow and NZD flow have quite similar processing time. We can also note that the covariance estimation (LW0) and redrawing from MVG do not incur any significant processing overhead. The oKDE, on the other hand takes quite a while to compute the GMM components. The processing time is the highest for D_{th} = 0.3 and it drops exponentially with increasing threshold. Redrawing with threshold 0.5 takes almost 1.2 seconds per time step, which is 6 times the processing time of the DHF-NZD-LW0. Hence the redrawing with KDE takes significant amount of time. The particle filter with 25000 particles takes 4.5 seconds, which makes it almost 4 times slower than the DHF-NZD-LW0-GMM. Finally, the slowest method is the DHF-CLF taking almost 8.5 second per time step. We note that the processing time for the model with non-Gaussian noise is little higher in general for most of the schemes.

Method	Processing time τ [s] (Gaussian)	Processing time τ [s] (Non-Gaussian)
EKF	0.0004	0.0004
EF-mean	0.004	0.005
EF-part	0.10	0.10
EF-part-fb	0.105	0.105
IC	0.19	0.20
CLF	8.34	8.57
NZD	0.195	0.20
NZD-LW0	0.202	0.205
NZD-LW0-MVG	0.205	0.21
NZD-LW0-GMM (D_{th} = 0.3)	1.77	1.83
NZD-LW0-GMM (D_{th} = 0.4)	1.33	1.36
NZD-LW0-GMM (D_{th} = 0.5)	1.19	1.21
NZD-LW0-GMM (D_{th} = 0.6)	1.12	1.13
NZD-LW0-GMM (D_{th} = 0.7)	1.09	1.11
SIR-PF (N_p = 25000)	4.34	4.65

Table 3.3: Comparison of processing time for different filters

3.8 Conclusion

DHF filters, even though not new in the literature, are still not fully explored in detail. They lack the in-depth theoretical and numerical analysis that the other contemporary filters have gone through. Especially, the implementation details are very application specific. In this chapter, we have tried to point out the key factors affecting the performance of a generic DHF. Highlighted factors have been studied individually in detail, with several possible methods provided for each of them. This include the different methods for pseudo-time discretization, different integration schemes, estimation of the prior covariance matrix and the redrawing. We have compared the results for those different schemes under on a challenging, nonlinear simulated scenario, both with Gaussian and non-Gaussian noises. The first critical step in the DHF implementation is the pseudo-time discretization and integration of the flow equation. The Euler based numerical integration scheme is quite simple, but together with a clever pseudo-time discretization, can perform quite well. The Euler method with exponentially spaced pseudo-time points, provides a nice trade off between the performance and the complexity. The next important step is the prior covariance estimation. We have analyzed different shrinkage covariance estimation schemes in this regard. Some of them have been derived for specific scenarios. The most general one is shrinkage towards identity matrix where no prior structure of the target matrix is assumed. It is a distribution free scheme and is shown to have outperformed other shrinkage estimators used in our analysis. DHF with shrinkage estimation methods is shown to have outperformed the DHF with the sample covariance matrix and the with the EKF estimate. Finally, we studied the effect of redrawing on the quality of the filter estimates. We choose two redrawing schemes, a single MVG based re-drawing and one based on the estimation of the posterior distribution via GMM, based on the oKDE. The estimated density is then used to redraw particles which are considered too off the main cluster. The re-drawing algorithm uses the Mahalanobis distance of particles to calculate the assemblage Υ. When Υ falls below a certain threshold determined by the redrawing intensity ν_M, particles deemed too wayward are redrawn. We show that the redrawing, when combined with the shrinkage estimation reduces the error even further. Redrawing from a GMM gives better estimation accuracy than from MVG. It has been shown that the properly done redrawing together with the shrinkage estimation could outperform a bootstrap particle filter, with considerably lesser execution time.

Appendix

3.A Derivation of the exact flow $\mathbf{f}_{EF}(\mathbf{x}, \lambda)$

We start with (3.19),

$$\log h(\mathbf{x}) + \mathbf{f}_{EF}^{T}(\mathbf{x}, \lambda) \cdot \nabla \log p(\mathbf{x}, \lambda) = -\nabla \cdot \mathbf{f}_{EF}(\mathbf{x}, \lambda)$$

It is assumed that the prior density and the likelihood are given as Gaussian i.e. $g(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \mathbf{P})$ and $h(\mathbf{x}) = \mathcal{N}(\mathbf{z}|\psi(\mathbf{x}), \mathbf{R})$. This leads to the following,

$$\begin{aligned} \log g(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^{T} \mathbf{P}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \\ \log h(\mathbf{x}) &= -\frac{1}{2}(\mathbf{z} - \psi(\mathbf{x}))^{T} \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x})) \end{aligned}$$

Given this, the gradient of the two densities then can be written as,

$$\nabla \log p(\mathbf{x}, \lambda) = -\mathbf{P}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) + \lambda \mathbf{H}^{T} \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x}))$$

where, $\mathbf{H} = \frac{\partial \psi}{\partial \mathbf{x}}$.
Next, it is assumed that the flow can be expressed linearly as,

$$\mathbf{f}_{EF}(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda)$$

We note that for this particular choice of flow $\nabla \cdot \mathbf{f}_{EF}(\mathbf{x}, \lambda) = \text{Tr}(\mathbf{A}(\lambda))$. Plugging the all terms in the equation (3.19), we get,

$$\begin{aligned} [\mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda)]^{T} \left[-\mathbf{P}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) + \lambda \mathbf{H}^{T} \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x})) \right] \\ - \frac{1}{2}(\mathbf{z} - \psi(\mathbf{x}))^{T} \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x})) = \text{Tr}(\mathbf{A}(\lambda)) \end{aligned}$$

For nonlinear systems, the measurement model can be linearized by the Taylor series expansion around the point $\mathbf{x}^{*}$ up to the first term,

$$\psi(\mathbf{x}) = \psi(\mathbf{x}^{*}) + \mathbf{H}(\mathbf{x} - \mathbf{x}^{*}) + \mathcal{O}(\Delta \mathbf{x}^{2})$$

This leads to ,

$$\begin{aligned} \psi(\mathbf{x}) &\approx \mathbf{H}\mathbf{x} + (\psi(\mathbf{x}^{*}) - \mathbf{H}\mathbf{x}^{*}) \\ &= \mathbf{H}\mathbf{x} + \boldsymbol{\gamma}^{*} \end{aligned}$$

And therefore,

$$[\mathbf{A}(\lambda)\mathbf{x}+\mathbf{b}(\lambda)]^T\left[-\mathbf{P}^{-1}\left(\mathbf{x}-\bar{\mathbf{x}}\right)+\lambda\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z}-\mathbf{H}\mathbf{x}-\boldsymbol{\gamma}^*\right)\right] - \frac{1}{2}\left(\mathbf{z}-\mathbf{H}\mathbf{x}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\left(\mathbf{z}-\mathbf{H}\mathbf{x}-\boldsymbol{\gamma}^*\right)=\mathrm{Tr}(\mathbf{A}(\lambda))$$

We drop λ for now onwards. Following with the derivation, we get,

$$\begin{aligned}&-\mathbf{x}^T\mathbf{A}^T\mathbf{P}^{-1}\mathbf{x}-\mathbf{x}^T\mathbf{A}^T\mathbf{P}^{-1}\bar{\mathbf{x}}+\lambda\mathbf{x}^T\mathbf{A}^T\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)-\lambda\mathbf{x}^T\mathbf{A}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}\\&\quad-\mathbf{b}^T\mathbf{P}^{-1}\mathbf{x}-\mathbf{b}^T\mathbf{P}^{-1}\bar{\mathbf{x}}+\lambda\mathbf{b}^T\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)-\lambda\mathbf{b}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}\\&\quad-\frac{1}{2}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)-\frac{1}{2}\mathbf{x}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}+\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}=\mathrm{Tr}(\mathbf{A})\end{aligned}$$

With the assumption that $\mathbf{P}^T\mathbf{A}^T=\mathbf{A}\mathbf{P}$ and by combining similar terms we get,

$$\begin{aligned}&\frac{1}{2}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)+\mathrm{Tr}(\mathbf{A})=-\mathbf{x}^T\left(\mathbf{P}^{-1}\mathbf{A}+\lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A}+\frac{1}{2}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)\mathbf{x}\\&+\left(\bar{\mathbf{x}}\mathbf{P}^{-1}\mathbf{A}+\lambda\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A}+\left(\mathbf{z}-\boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}-\lambda\mathbf{b}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}-\mathbf{b}^T\mathbf{P}^{-1}\mathbf{H}\right)\mathbf{x}\end{aligned}$$

The above is a quadratic equation, which can be expressed as,

$$\mathbf{x}^T\Upsilon\mathbf{x}+\delta^T\mathbf{x}+e=0$$

The next step is to set the coefficients of the monomial (quadratic and linear) terms to zero. This can be justified as equation must hold for all values of $\mathbf{x}$, which can be ensured by setting the coefficients to zeros. By setting the quadratic term to zero, we get,

$$\mathbf{P}^{-1}\mathbf{A}+\lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A}+\frac{1}{2}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}=0$$

This can be solved for yet unknown matrix $\mathbf{A}$,

$$SIRparticlefilter\mathbf{A}=-\frac{1}{2}\left[\mathbf{P}^{-1}+\lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}$$

Further simplifications can be made using the *Woodbury*'s lemma,

$$\left[\mathbf{A}+\mathbf{B}\mathbf{C}\mathbf{D}\right]^{-1}\mathbf{B}\mathbf{C}=\mathbf{A}^{-1}\mathbf{B}\left[\mathbf{C}^{-1}+\mathbf{D}\mathbf{A}^{-1}\mathbf{B}\right]^{-1}$$

where $\mathbf{A}=\mathbf{P}^{-1}$, $\mathbf{B}=\mathbf{H}^T$, $\mathbf{C}=\mathbf{R}^{-1}$ and $\mathbf{D}=\lambda\mathbf{H}$. This leads to the following simplified form for $\mathbf{A}$,

$$\boxed{\mathbf{A}=-\frac{1}{2}\mathbf{P}\mathbf{H}^T\left[\lambda\mathbf{H}\mathbf{P}\mathbf{H}^T+\mathbf{R}\right]^{-1}\mathbf{H}}$$

Next, we turn our attention towards $\mathbf{b}$. Putting the coefficient of the linear term to zero we get,

$$\bar{\mathbf{x}}\mathbf{P}^{-1}\mathbf{A} + \lambda\left(\mathbf{z} - \boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A} + \left(\mathbf{z} - \boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H} - \lambda\mathbf{b}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} - \mathbf{b}^T\mathbf{P}^{-1}\mathbf{H} = 0$$

$$\mathbf{b}^T\left[\mathbf{P}^{-1} + \lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right] = \bar{\mathbf{x}}\mathbf{P}^{-1}\mathbf{A} + \lambda\left(\mathbf{z} - \boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A} + \left(\mathbf{z} - \boldsymbol{\gamma}^*\right)^T\mathbf{R}^{-1}\mathbf{H}$$

Taking transpose and multiplying with $\mathbf{P}$, we get

$$\left[\mathbf{I} + \lambda\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]\mathbf{b} = \mathbf{A}\bar{\mathbf{x}} + \lambda\mathbf{A}\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z} - \boldsymbol{\gamma}^*\right) + \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z} - \boldsymbol{\gamma}^*\right)$$

This leads to,

$$\mathbf{b} = \left[\mathbf{I} + \lambda\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\left[\left(\mathbf{I} + \lambda\mathbf{A}\right)\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z} - \boldsymbol{\gamma}^*\right) + \mathbf{A}\bar{\mathbf{x}}\right]$$

We apply the *Woodbury*'s lemma yet again, but in a slight different form

$$\left[\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{D}\right]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}\right]^{-1}\mathbf{D}\mathbf{A}^{-1}$$

Recognizing, $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = \lambda\mathbf{P}\mathbf{H}^T$, $\mathbf{C} = \mathbf{R}^{-1}$ and $\mathbf{D} = \mathbf{H}$, we can solve for the first term on the R.H.S as the following,

$$\begin{aligned}\left[\mathbf{I} + \lambda\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]^{-1} &= \mathbf{I} - \lambda\mathbf{I}\mathbf{P}\mathbf{H}^T\left[\mathbf{R} + \lambda\mathbf{H}\mathbf{P}\mathbf{H}^T\right]^{-1}\mathbf{H}\mathbf{I} \\ &= \mathbf{I} + 2\lambda\mathbf{A}\end{aligned}$$

This yields a simpler formula for $\mathbf{b}$,

$$\boxed{\mathbf{b} = \left(\mathbf{I} + 2\lambda\mathbf{A}\right)\left[\left(\mathbf{I} + \lambda\mathbf{A}\right)\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\left(\mathbf{z} - \boldsymbol{\gamma}^*\right) + \mathbf{A}\bar{\mathbf{x}}\right]}$$

3.B Assemblage Υ

Let **D** be the vector containing the Mahalanobis distances of the particles. We assume that the particles can be divided into *L* distinct sub-clusters, each cluster has the same distance to the estimated mean. In that case $\mathbf{D} = [d_1, d_2, \dots d_L]$. This could either mean that the particles lie on hyper-balls in $\mathbb{R}^d$ with radii d_i concentric around the estimated mean, or each cluster is small enough, and far apart from others, such that it can be approximated by individual hyper-balls. Let the *ith* cluster has N_i number of particles such that $\sum_{i=1}^{L} N_i = N_p$. Let the vector $\boldsymbol{\Phi}$ contain the inverse of Mahalanobis distances

$$\boldsymbol{\Phi} = \left[\left(\frac{1}{d_1}\right)_{\times N_1}, \left(\frac{1}{d_2}\right)_{\times N_2}, \cdots, \left(\frac{1}{d_L}\right)_{\times N_L} \right]$$

The sum of the vector $\boldsymbol{\Phi}$ is given by,

$$\begin{aligned} \sum_{i=1}^{L} \Phi_i &= \frac{N_1}{d_1} + \frac{N_2}{d_2} + \cdots + \frac{N_L}{d_L} \\ &= \frac{\sum_{i=1}^{L} \left(\prod_{j=1, j\neq i}^{L} d_j \right) N_i}{\prod_{j=1}^{L} d_j} \end{aligned}$$

Therefore the normalized vector $\tilde{\boldsymbol{\Phi}}$ is given by

$$\tilde{\boldsymbol{\Phi}} = \left[\left(\frac{1}{d_1}\right)_{\times N_1}, \left(\frac{1}{d_2}\right)_{\times N_2}, \cdots, \left(\frac{1}{d_L}\right)_{\times N_L} \right] \times \frac{\prod_{j=1}^{L} d_j}{\sum_{i=1}^{L} \left(\prod_{j=1, j\neq i}^{L} d_j \right) N_i}$$

Next the sum of squares of the above vector is evaluated,

$$\begin{aligned}
\sum_{i=1}^{L}\tilde{\Phi}_i^2 &= \left[\frac{N_1}{(d_1)^2}+\frac{N_2}{(d_2)^2}+\cdots+\frac{N_L}{(d_L)^2}\right]\times\left(\frac{\prod_{j=1}^{L}d_j}{\sum_{i=1}^{L}\left(\prod_{j=1,j\neq i}^{L}d_j\right)N_i}\right)^2 \\
&= \frac{N_1\prod_{j=1,j\neq 1}^{L}(d_j)^2}{\sum_{i=1}^{L}\left(\prod_{j=1,j\neq i}^{L}(d_j)^2\right)N_i}+\frac{N_2\left(\prod_{j=1,j\neq 2}^{L}d_j\right)^2}{\left(\sum_{i=1}^{L}\left(\prod_{j=1,j\neq i}^{L}d_j\right)N_i\right)^2}+ \\
&\quad \cdots+\frac{N_L\left(\prod_{j=1,j\neq L}^{L}d_j\right)^2}{\left(\sum_{i=1}^{L}\left(\prod_{j=1,j\neq i}^{L}d_j\right)N_i\right)^2} \\
&= \sum_{i=1}^{L}N_i\left(\frac{\prod_{j=1,j\neq i}^{L}d_j}{\sum_{k=1}^{L}\left(\prod_{j=1,j\neq k}^{L}d_j\right)N_k}\right)^2
\end{aligned}$$

and finally the assemblage Υ is given by,

$$\Upsilon=\frac{1}{\sum_{i=1}^{L}\tilde{\Phi}_i^2}=\frac{\left(\sum_{k=1}^{L}N_k\left(\prod_{j=1,j\neq k}^{L}d_j\right)\right)^2}{\sum_{i=1}^{L}N_i\left(\prod_{j=1,j\neq i}^{L}d_j\right)^2}$$

Below, we consider few special cases for the assemblage.

3.B.1 Number of clusters equals N_p

Each particle is consider as single clusters, hence each clusters has one particle with distinct distance d_i. assemblage in that case is given by,

$$\Upsilon=\frac{\left(\sum_{k=1}^{N_p}\left(\prod_{j=1,j\neq k}^{N_p}d_j\right)\right)^2}{\sum_{i=1}^{N_p}\left(\prod_{j=1,j\neq i}^{N_p}d_j\right)^2}$$

3.B.2 All particles equidistant

If $d_i \approx d$

$$\Upsilon = \frac{\left(\sum_{k=1}^{N_p} d^{N_p-1}\right)^2}{\sum_{k=1}^{N_p} \left(d^{N_p-1}\right)^2} = \frac{\left(N_p\ d^{N_p-1}\right)^2}{N_p\ d^{2(N_p-1)}} = \frac{N_p^2\ d^{2(N_p-1)}}{N_p\ d^{2(N_p-1)}}$$

which leads to,

$$\Upsilon = N_p$$

3.B.3 Two dominant clusters

Now suppose that there are two main sub-cluster i.e. L =2.

$$\Upsilon = \frac{\left(\sum_{k=1}^{2} N_k \left(\prod_{j=1,j\neq k}^{2} d_j\right)\right)^2}{\sum_{i=1}^{2} N_i \left(\prod_{j=1,j\neq i}^{2} d_j\right)^2} = \frac{\left(d_2 N_1 + d_1 N_2\right)^2}{d_2^2 N_1 + d_1^2 N_2}$$

Now assume that $d_1 >> d_2$. In that case we can say in the limiting sense,

$$\begin{aligned}\lim_{d_1\to\infty} \Upsilon &= \lim_{d_1\to\infty} \frac{\left(d_2 N_1 + d_1 N_2\right)^2}{d_2^2 N_1 + d_1^2 N_2} \\ &= \frac{N_2^2}{N_2} = N_2\end{aligned}$$

Likewise for $d_2 >> d_1$, $\lim_{d_2\to\infty} \Upsilon = N_1$.

Chapter 4

Bayesian processing of Massive sensor data with log-homotopy based particle flow

In the presence of a large number of measurements, as known as the *massive sensor data*, most estimator face serious computational challenges. This problem is more pronounced for particle filters, SMCMC methods and the DHF. The problem can be pin pointed to the computation of likelihood term or its derivatives, which become a bottle neck. Dimensionality reduction may be employed, for example through data clustering, but it remains unclear whether the samples generated thereof still belong to the true posterior distribution. An interesting solution has been provided in [FSM15] based on [BDH15], where probabilistic subsampling also termed as the *confidence sampling*, has been employed to reduce the number of likelihood evaluations in the context of SMCMC. A major benefit of this approach is that it comes with a theoretical guarantee regarding the generated samples i.e.the sampled distribution always lie with in a user specified tolerance of the true posterior distribution. On the other hand, a better suited proposal distributions is one of the key requirements for the algorithm. While for some problems the selection of proposal distribution could be straight forward e.g. moderately nonlinear or Gaussian models, for others the choice may not be that obvious.

In this chapter, we present a novel approach for the Bayesian processing of massive sensor data, by combining the log-homotopy based particle flow together with SMCMC. We propose an initial measurement clustering, after which the log-homotopy flow is applied. The samples after the flow are assumed to be approximately in the vicinity of their true posterior locations, though not there exactly. Hence, they can form an excellent proposal to be used within the subsequent confidence sampling driven MCMC procedure. The main purpose of the last step is to have the convergence guarantee that comes associated with later procedure. In this way, we essentially bring the strength of both methods under one banner.

The chapter is organized as the following. The problem formulation is done in section 4.1. Section 4.2 revisits the basics of SMCMC. Next, in section 4.3, possible approaches for massive sensor data processing via SMCMC are highlighted. This is followed by section 4.4, where the probabilistic subsampling i.e.confidence sampling methodology is discussed in the detail. Potential issues with the choice of proposal distribution for SMCMC are mentioned in section 4.5. The use of DHF together with the data clustering to form a better proposal is also advocated in the same section. Section 4.6 describes our devised DHF based, confidence sampling driven

SMCMC algorithm. Mathematical models and simulation setup for test scenario used in the evaluation of the new scheme are mentioned in section 4.7. Section 4.8 provide results for the new method, followed by the conclusion in section 4.9.

4.1 Problem formulation

Recalling the standard formalism of recursive Bayesian estimation, as defined earlier in section 2.1,

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_k) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_k)d\mathbf{x}_k \tag{4.1}$$

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) = \frac{p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{Z}_k)}{p(\mathbf{z}_{k+1}|\mathbf{Z}_k)} \tag{4.2}$$

For a massive sensor data scenario, the vector $\mathbf{z}_k$ comprises of multiple measurements at each time step *k*. In case all measurements are independent, the likelihood can be written as,

$$p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}) = \prod_{i=1}^{M_{k+1}} p(\mathbf{z}_{k+1}^i|\mathbf{x}_{k+1}) \tag{4.3}$$

where M_{k+1} is the number of measurements present at time *k*+1. For massive sensor data, $M_{k+1} \gg 1$. In addition to the non-availability of exact solution for nonlinear/non-Gaussian systems, the presence of a hugh number of measurements itself poses severe computational challenges. In the current work, we are primarily interested in a SMCMC based method for the evaluation of (4.2) in the presence of a large number of measurements. We will employ *confidence sampling* to reduce the computational complexity, and use DHF based proposal to increase the efficiency of the MCMC sampling procedure.

4.2 Sequential Markov chain Monte Carlo (SMCMC) - A recap

Markov chain Monte Carlo method were invented to simulate dynamics of gaseous system in equilibrium [MRR$^+$53]. It was realized that instead of simulating the exact system dynamics and run it until the equilibrium, a simulation of a Markov chain was enough which had the equilibrium distribution as its stationary distribution. Initially, their usage was limited to the solution of lower dimensional problems , but with the increase in the computational power this hurdle was largely overcome. It was not until 80's that the statistical community got involved, after which a plethora of literature got published regarding MCMC methods.

The use of sequential MCMC in target tracking problems has its roots in the inadequacy of importance sampling in higher dimensional spaces. SMCMC, as used in the target tracking applications [KBD05], [LSSM15] and [SPCG09], differs from SMC in that they do not sample from the posterior distribution directly. Instead, at each time instance *k*, a stationary, reversible jump, Markov chain is constructed through a Markov transition kernel $q(\mathbf{x}_{k+1}^y|\mathbf{x}_{k+1}^x)$. Here, x and y indicate the initial and the proposed states and the kernel is referred to as the *proposal distribution* or simply the proposal. The chain is started at an arbitrary location and is

continuously lengthened by appending samples. The chain is assumed to have the posterior distribution $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ as its stationary distribution. Every new sample generated through the proposal distribution is either accepted or rejected, based on the Metropolis Hastings procedure, as described in the Algorithm 7.

Algorithm 7 Pseudo-code for Metropolis Hastings (MH) algorithm

procedure MH($\mathbf{x}_{k+1}^0$)
  for $m = 0, \cdots, N_{\text{iter}}$ **do**
    $\mathbf{x}_{k+1}^* \sim q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)$
    $u \sim \mathcal{U}_{(0,1)}$
    $\alpha(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) = \frac{p(\mathbf{x}_{k+1}^*|\mathbf{Z}_{k+1}) \times q(\mathbf{x}_{k+1}^m|\mathbf{x}_{k+1}^*)}{p(\mathbf{x}_{k+1}^m|\mathbf{Z}_{k+1}) \times q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)}$
    if $u < \alpha(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*)$ **then**
      $\mathbf{x}_{k+1}^{m+1} \leftarrow \mathbf{x}_{k+1}^*$ ▷ Accept
    else
      $\mathbf{x}_{k+1}^{m+1} \leftarrow \mathbf{x}_{k+1}^m$ ▷ Reject
    end if
  end for
end procedure

Here $\mathbf{x}_{k+1}^0$ refers to the initial sample of the MCMC chain. Proposed samples are always accepted when $\alpha(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) \geq 1$ (moving to a higher probability region), whereas for $\alpha(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) < 1$ samples are accepted with the corresponding probability. The chain is assumed to have the posterior distribution $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$ as its stationary distribution. After a certain burn-in period has been expired, every *pth* sample is kept. This procedure is called thinning out. *N* such samples are kept, which are assumed to have been approximately distributed according to the posterior distribution. MCMC applied in a sequential setting, while generally better than particle filters for the estimation of higher dimensional state spaces, does still require a lot of processing due to extensive sampling of the involved densities [KBD05].

4.3 Massive sensor data processing using MCMC

Massive sensor data describes the situation where a large number of observations / measurements are available to be processed at any time instant. This can occur in several situations. In the most typical scenario, massive sensor data could arise in the tracking of a single or multiple targets using the measurement gathered through a multitude of sensors. In that scenario, each target could have several measurements that have to be processed together. Few such examples are bearing only estimation in the presence of clutter [MMBs02], [BSDH09], and in the presence of position biases and offsets [TTKM16]. Alternatively, massive sensor data cold occur when the tracked target(s) can no longer be modeled as point source object(s) due to the enhanced sensor resolution. Instead, each target can have multiple scattering centers, which could also leads to more than one measurements per target, thereby constituting the realm of the extended object tracking [Koc08]. The presence of multiple objects and/or clutter exacerbate the problem, as the identity resolution could become critical. This could render the employment of traditional state estimation methods like EKF/UKF inadequate, while the other methods like

PF and SMCMC could simply be too expensive for estimation in such higher dimensional state spaces.

Massive sensor data, sometimes also referred to as the *Big data* in the literature [FSMG15], has been a subject of continued research. The proposed MCMC methods for such data can largely be categorized into two main classes. In the first class of methods, also termed as the divide and conquer approach, measurements are divided into non-overlapping batches or blocks. Each block is processed by individual processing nodes [SWC$^+$10], resulting in batch posterior MCMC based estimates. The main question then becomes how to optimally combine these estimates. Typical solutions involve approximating local densities as Gaussian or Gaussian kernel density estimates, which results in forming the overall posterior approximation by the multiplication of batch densities [SBB$^+$16], [NWX13] and [CCL15]. Divide and conquer methods though quite simple in terms of tractability and implementation, rely on the underlying Gaussian assumption. Their performance degrades when the assumption of local Gaussianity is violated.

The other class of methods use the idea of subsampling or decimation of the measurement set, such that MCMC is applied only to a subset of the whole data. This include methods like pseudo-marginal MH which relies on using an unbiased non-negative estimator of the likelihood [AR09], MH independent approach based on stochastic gradient Langevin dynamics (SGLD) [MW11] and approximate subsampling approaches like naive subsampling [BDH15] etc. In [AR09], the proposed algorithm results in estimates with high variance and slow convergence of the Markov chain. SGLD based method appears quite similar to *Metropolis adjusted Langevin algorithm* (MALA) [RC04] and provides consistent estimates of the posterior density, though it also suffers from a slow convergence rate. One can increase the speed by choosing larger step sizes for solving the relevant stochastic differential equation, but this could lead the chain away from the support of the target distribution [BDH15]. In naive subsampling, measurements are subsampled at random to yield a lower sized data set of fixed cardinality. This has a *broadening* effect on the likelihood as fewer samples contribute to the final estimate. Naive subsampling while trivial to implement, does not guarantee that the Markov chain samples out of the MH step are indeed from the correct target distribution. It also suffers from a slow convergence speed. In [QVK14], authors propose an exact but expensive approach to reduce the variance of the estimates, where instead of subsampling, importance sampling is performed by assigning weights proportional to their likelihood. It uses *Gaussian processes* or splines trained on small subset of the data. This method, while demonstrably efficient, still relies on the Gaussian assumption and the learning of good proxies for the likelihood terms. The *likelihood proxies* have to be cheap to compute and should approximate the likelihood reasonably well.

4.4 Confidence sampler: Probabilistic subsampling within MCMC framework

An interesting approach has been proposed in [RAC14], where the probabilistic subsampling of the data has been proposed. It relies on the use of the so called *concentration inequalities*, which provide a bound on the maximum absolute deviation of the average likelihood ratio. The method automatically select a subset of the measurement data based on the evaluation of a stopping criterion. The MH accept-reject decision is based on a user defined probability 1-δ. This leads to its convergence speed within $\mathcal{O}(\delta)$ of that of the full MH, where all of the measure-

ments are taken into account. Also, the resulting chain is uniformly ergodic provided that the original chain also had the said property. Most importantly, the algorithm provides a theoretical guarantee that the sampled density is with the $\mathcal{O}(\delta)$ of the true posterior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$. The disadvantage of the approach is that the evaluation of the stopping criterion uses a measure of the range of log-likelihood ratio set, which except for a few simple cases, requires likelihood calculation for the whole data set. The concentration inequalities are the *worst case* assurances but they carry with them an additional processing cost. Since the method utilizes a confidence bound, it is also termed as the *confidence sampler*. Next, we briefly mention the highlights of the confidence sampling procedure.

We begin with the reformulation of the MH step as described in the step 5 of the Algorithm 7.

$$u < \frac{p(\mathbf{x}^*_{k+1}|\mathbf{Z}_{k+1})q(\mathbf{x}^m_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{x}^m_{k+1}|\mathbf{Z}_{k+1})q(\mathbf{x}^*_{k+1}|\mathbf{x}^m_{k+1})} \tag{4.4}$$

Now assuming that the likelihood can be decomposed into individual terms i.e.ȧssuming independence of measurements, we can re-write as,

$$u < \frac{p(\mathbf{x}^*_{k+1}|\mathbf{Z}_{k})q(\mathbf{x}^m_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{x}^m_{k+1}|\mathbf{Z}_{k})q(\mathbf{x}^*_{k+1}|\mathbf{x}^m_{k+1})} \prod_{i=1}^{M_{k+1}} \frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})} \tag{4.5}$$

Further manipulation of the equation leads to,

$$\begin{aligned} \frac{1}{M_{k+1}} \log\left[u\frac{p(\mathbf{x}^*_{k+1}|\mathbf{Z}_{k})q(\mathbf{x}^m_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{x}^m_{k+1}|\mathbf{Z}_{k})q(\mathbf{x}^*_{k+1}|\mathbf{x}^m_{k+1})}\right] &< \frac{1}{M_{k+1}} \sum_{i=1}^{M_{k+1}} \log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right] \\ \psi(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) &< \Lambda_{M_{k+1}}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) \end{aligned} \tag{4.6}$$

where the left side of the inequality is independent of the data, while the right hand side exclusively depends on the measurements. We define the average log-likelihood ratio using full data set, $\Lambda_{M_{k+1}}$. When using a subset of measurements of size N_m, the average log-likelihood ratio can be defined as,

$$\Lambda_{N_m} = \frac{1}{N_m} \sum_{i=1}^{N_m} \log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right] \tag{4.7}$$

Concentration inequalities can be used to obtain a bound on the Λ_{N_m}.

$$P(|\Lambda^{M_{k+1}}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) - \Lambda^{N_m}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})| \leq c_{N_m}) \geq 1 - \delta_{N_m} \tag{4.8}$$

where δ_{N_m} is the used-defined threshold and c_{N_m} is dependent on the particular form of the inequality used. In [BDH15], the *Bernstein's* inequality has been used which results in:

$$c_{N_m} = \sqrt{\frac{2V_{S_m}\log(3 \times \delta^{-1}_{N_m})}{N_m}} + \frac{3\mathcal{R}\log(3 \times \delta^{-1}_{N_m})}{N_m} \tag{4.9}$$

In the formula above, V_{S_m} is the sample variance of the subsampled log-likelihood ratios and $\mathcal{R}$ is the range given by,

$$\mathcal{R}_{k+1} = \max_{1\leq i\leq M_{k+1}} \left\{\log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right]\right\} - \min_{1\leq i\leq M_{k+1}} \left\{\log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right]\right\} \tag{4.10}$$

Now referring back to the standard MCMC method, we note that the accept reject decision is based on the evaluation of (4.6). The average log-likelihood ratio based on the whole data set is not of our interest, instead, we would like to base our decision on $\Lambda_{N_m}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})$. It results in a stopping criterion $|\Lambda^{N_m}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) - \psi(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})| > c_{N_m}$. This, when seen in the light of the concentration inequality, can be interpreted as taking the right decision with probability at least 1-δ, if the stopping criterion is met.

We start with the user-defined parameter $\delta_s \in (0,1)$. The algorithm starts with an empty set $\mathcal{Z}_k$, for the subsampled measurements. At each iteration, measurements are added to this subset and the stopping criterion is re-checked. The data aggregation stops as soon as $|\Lambda^{N_m}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) - \psi(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})| > c_{N_m}$ holds true, or all measurements have been added to the set $\mathcal{Z}_{k+1}$, in which case the accept-reject decision is based on the evaluation of the full data set. For some $p_s > 1$, we set $\delta_{S_m} = \frac{p_s - 1}{p_s N^{p_s}_{m,k+1}} \delta_s$ which leads to $\Sigma_{N_{m,k+1}>1} \delta_{S_m} \leq \delta_s$. The event,

$$\mathcal{E} = \bigcap_{N_{m,k+1} \geq 1} \{|\Lambda^{M_{k+1}}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) - \Lambda^{N_m}(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})| \leq c_{N_m}\} \tag{4.11}$$

therefore, holds with the probability at least $1 - \delta$. It can be seen that (4.10) requires the evaluation of log-likelihood ratios for the whole data set. While for some problems range can be straightforwardly computed e.g. in the case of a Gaussian likelihood, it might not be generally the case. Thus any potential gain achieved by subsampling the data is lost.

To alleviate the problem of the still high processing cost, an approximate method has been presented in [BDH15] which makes use of the so called *proxies*. Proxies are likelihood mimicking terms which are cheap to evaluate, but at the same time should approximate the actual function sufficiently well. The proxy based algorithm yield empirical gains, but still keeps the guarantees of the original scheme. Additionally, proxy terms act as control variates, therefore reducing the variance of the estimates. In [BDH15], three desirable criterion are described for any potential proxy term $\mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})$:

1. The proxy term should well approximate the actual log-likelihood ratio $\mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) \approx \log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right]$
2. $\Sigma_{i=1}^{M_{k+1}} \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})$ can be cheaply computed.
3. The range $\mathcal{R}_{k+1}$ can be uniformly bounded and cheap to compute.

The introduction of proxy terms leads to the modification of Λ_{N_m},

$$\Lambda_{N_m} = \frac{1}{N_m} \sum_{i=1}^{N_m} \left\{ \log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right] - \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) \right\} \tag{4.12}$$

Amongst the several choices available for proxies, the simplest is provided by the first order Taylor series expansion, [FSM15]. Given, $\ell_i(\mathbf{x}_{k+1}) = \log p(\mathbf{z}^i_{k+1}|\mathbf{x}_{k+1})$, the linearized log likelihood can be expressed as,

$$\hat{\ell}_i(\mathbf{x}_{k+1}) = \ell_i(\mathbf{x}^+_{k+1}) + (\nabla \ell_i\ )^T_{\mathbf{x}^+_{k+1}} . (\mathbf{x}_{k+1} - \mathbf{x}^+_{k+1}) \tag{4.13}$$

where $\nabla \ell_i$ is the gradient of the log-likelihood with the linearization carried about some point $\mathbf{x}^+_{k+1}$.

Algorithm 8 Confidence sampler with proxy

procedure CONFIDENCESAMPLER($\mathbf{x}^m_{k+1}$,$\mathbf{x}^*_{k+1}$,$\mathbf{z}_{k+1}$,δ,$\psi(.)$, γ_{mcmc})
  $N_{m,k+1} = 0$ ▷ Number of subsampled measurements
  $\Lambda_{N_m} = 0$ ▷ Subsampled log-likelihood
  $\mathcal{Z}^*_{k+1} = \emptyset$ ▷ Set of subsampled measurements
  $l_N = 1$ ▷ Batch size
  i = 0 ▷ Loop counter
  FLAG = UP ▷ Flag variable
  Compute $\mathcal{R}^B_{k+1}$ according to (4.18) ▷ Range
  while FLAG == UP **do**
    i = i + 1
    $\{\mathbf{z}^{N_{m,k+1}+1,*}_{k+1}, \cdots, \mathbf{z}^{b,*}_{k+1}\} \sim_{w/repl.} \mathcal{Z}_{k+1} \backslash \mathbf{z}_{k+1}$
    $\mathcal{Z}^*_{k+1} = \mathcal{Z}^*_{k+1} \cup \{\mathbf{z}^{N_{m,k+1}+1,*}_{k}, \cdots, \mathbf{z}^{b,*}_{k+1}\}$
    $\Lambda_{N_m} = \frac{1}{l_N}\left(N_{m,k+1} \times \Lambda_{N_m} + \Sigma_{j=N_{m,k+1}+1}^{l_N}\left[\log \frac{p(\mathbf{z}^{j,*}_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^{j,*}_{k+1}|\mathbf{x}^m_{k+1})} - \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})\right]\right)$
    $N_{m,k+1} = l_N$
    $\delta_i = \frac{p_s - 1}{p_s i^{p_s}}\delta$
    Compute c according to (4.9) ▷ Confidence bound
    $l_N = \gamma_{mcmc} N_{m,k+1} \wedge M_{k+1}$ ▷ Geometrically increase the batch size
    if $|\Lambda_{N_m} + \frac{1}{M_{k+1}} \sum_{i=1}^{M_{k+1}} \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) - \psi(.)| \geq c$ **or** $N_{m,k+1} == M_k$ **then**
      FLAG = DOWN
    end if
  end while
  return $N_{m,k+1}, \Lambda_{N_m}$
end procedure

Finally, the proxy can be written as,

$$\begin{aligned}\mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) &= \hat{\ell}_i(\mathbf{x}^*_{k+1}) - \hat{\ell}_i(\mathbf{x}^m_{k+1}) \\ &= (\nabla \ell_i \ )^T_{\mathbf{x}^+_{k+1}} . (\mathbf{x}^*_{k+1} - \mathbf{x}^m_{k+1}) \end{aligned} \tag{4.14}$$

Also,

$$\frac{1}{M_{k+1}} \sum_{i=1}^{M_{k+1}} \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) = \frac{1}{M_{k+1}} \sum_{i=1}^{M_{k+1}} (\nabla \ell_i \ )^T_{\mathbf{x}^+_{k+1}} (\mathbf{x}^*_{k+1} - \mathbf{x}^m_{k+1}) \tag{4.15}$$

This leads to the following form of the range,

$$\begin{aligned}\mathcal{R}_{k+1} = &\max_{1 \leq i \leq M_{k+1}} \left\{\log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right] - \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})\right\} - \\ &\min_{1 \leq i \leq M_{k+1}} \left\{\log\left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})}\right] - \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})\right\} \end{aligned} \tag{4.16}$$

Now we can derive an upper bound on the range $\mathcal{R}^B_{k+1}$, such that as $\mathcal{R}^B_{k+1} \geq \mathcal{R}_{k+1}$, which can be computed efficiently.

$$\begin{aligned}\mathcal{R}^B_{k+1} &= 2 \max_{1\leq i\leq M_{k+1}} \left\{ \left| \log \left[\frac{p(\mathbf{z}^i_{k+1}|\mathbf{x}^*_{k+1})}{p(\mathbf{z}^i_{k+1}|\mathbf{x}^m_{k+1})} \right] - \mathscr{P}_i(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1}) \right| \right\} \\ &= 2 \max_{1\leq i\leq M_{k+1}} \left\{ \left| \ell_i(\mathbf{x}^*_{k+1}) - \ell_i(\mathbf{x}^m_{k+1}) - \hat{\ell}_i(\mathbf{x}^*_{k+1}) + \hat{\ell}_i(\mathbf{x}^m_{k+1}) \right| \right\} \\ &= 2 \max_{1\leq i\leq M_{k+1}} \left| B_i(\mathbf{x}^m_{k+1}) - B_i(\mathbf{x}^*_{k+1}) \right| \end{aligned} \tag{4.17}$$

where, $B_i(\mathbf{x}_{k+1}) = \ell_i(\mathbf{x}_{k+1}) - \hat{\ell}_i(\mathbf{x}_{k+1})$ is the residual error of the Taylor series approximation bounded as, $|B_i(\mathbf{x}_{k+1})| \leq 0.5||\nabla^2 \log l(\mathbf{x}_{k+1})||_\infty ||\mathbf{x}_{k+1} - \mathbf{x}^+_{k+1}||^2_2$. We can further upper bound the range,

$$\mathcal{R}^B_{k+1} = 2 \max_{1\leq i\leq M_{k+1}} \left| |B_i(\mathbf{x}^m_{k+1})| + |B_i(\mathbf{x}^*_{k+1})| \right| \tag{4.18}$$

The last equation follows after the application of the triangle inequality. The main advantage of using Taylor series based proxies is the ease in their computation and that of the range measure $\mathcal{R}$. The full confidence sampling algorithm using proxies is described in the Algorithm 8.

4.5 A better proposal distribution for MH step

The confidence sampler requires the term $\psi(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})$ as an input, which in turn depends on the proposal distribution $q(\mathbf{x}^*_{k+1}|\mathbf{x}^m_{k+1})$. In its simplest form, a proposal can simply be based on the random walk. The resulting scheme is termed as the random walk Metropolis (RWM) and was first described in [MRR+53]. The proposed value for a d-dimensional Markov chain is chosen randomly from a pre-specified Lebesgue density, with a given step size. The acceptance rate for RWM algorithm is reasonable for a 1D state space, but as the dimensionality increases, it decreases dramatically. Also, if the target distribution is multi modal with far separated modes, the Markov chain can get *stuck* at one of the modes and might take very long before it can get to the others. This problem is of course exacerbated in higher dimensions. Therefore, it makes sense to propose samples from a distribution that is well suited to the problem i.e.it has support that is wide enough, so that any region with non-zero mass can be reached. A sufficient condition is that $q(\mathbf{x}^*_{k+1}|\mathbf{x}^m_{k+1})$ is positive everywhere. The accept-reject step makes sure that as the chain reaches stationarity, it is sampling from the correct target distribution indeed i.e from $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$.

The next most obvious choice for the proposal distributions is the prior density $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$. If chosen so, $\psi(\mathbf{x}^m_{k+1}, \mathbf{x}^*_{k+1})$ reduces to $\log(u)/M$, making it independent of the state values. While quite simple, this choice is often not very, especially when the prior and posterior distributions have significant probability masses in well separated regions of the state space. This could be the case when the likelihood is quite peaked, resulting in a very low acceptance rate. As a possible remedy, one could sample from regions of the posterior with significant probability mass to improve the acceptance rate.

As alluded to in the introduction, the log-homotopy based particle flow can be used to form a better proposal. This is owing to the fact that the flow incrementally moves the particles towards their posterior locations by gradually incorporating measurements. This helps solve the issue of *degeneracy* in a standard estimation problem. DHF, if carefully implemented can also be computationally cheaper than a standard particle filter [KUK17]. Hence, it comes naturally to use

the particles out of the DHF to form the proposal distribution for the subsequent MCMC step. Below we describe some basics of the homotopy based particle flow and its implementation methodology.

4.5.1 Log homotopy based particle flow

The whole procedure is shown in an algorithmic form in the Algorithm 9, where $\{\hat{\mathbf{x}}^i_{k+1}\}^{N_p}_{i=1}$ and $\{\bar{\mathbf{x}}^i_{k+1}\}^{N_p}_{i=1}$ are the set of prior and posterior particles, respectively. We plan to use the DHF based approximation for the posterior density as the proposal in the confidence sampling based SMCMC i.e. $q(\mathbf{x}_{k+1}|.) \approx \hat{p}_{DHF}(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$. But before this can be done, there are two main issues to be resolved. The first one is the processing time of the DHF. As the main focus of the work is to propose a MCMC based method that can handle massive sensor data, the dimensionality of the measurement space becomes a critical factor here. As it can be noted that non zero diffusion constrained flow equation requires the Hessian of the log-likelihood function. A direct application of the DHF, therefore, can be prohibitively expensive. The question becomes, how to use the DHF while still maintaining a reasonably low processing cost. One answer to this problem is to decimate or sub-sample the measurement set. The second question relates with the finding of an analytical approximation for $\hat{p}_{DHF}(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$, for its sampling and evaluation.

Algorithm 9 Log homotopy flow based measurement update

procedure LOGHOMOTOPYFLOWUPDATE($\{\hat{\mathbf{x}}_{k+1}\}^{N_p}_{i=1}$,$\{\Delta\lambda_j, \lambda_j\}^{N_\lambda}_{j=1}$,$\mathbf{z}_{k+1}$)
  $\hat{\mathbf{P}}_{k+1}$ = SHRINKAGEESTIMATOR($\hat{\mathbf{x}}^i_{k+1}$) ▷ Estimate the prior covariance matrix
  for i = 1 : N_p **do**
    $\mathbf{y}_0 = \hat{\mathbf{x}}^i_{k+1}$ ▷ Temporary variable
    for j = 1 : N_λ **do**
      $\mathbf{H}_\lambda$ = GETHESSIANMATRIX($\log h(\mathbf{z}_k|\mathbf{y}_{j-1})$)
      $\mathbf{h}_\lambda$ = GETGRADIENTVECTOR($\log h(\mathbf{z}_k|\mathbf{y}_{j-1})$)
      $\mathbf{m}(\mathbf{y}_j) = -\left[\hat{\mathbf{P}}_k^{-1} + \lambda_j \mathbf{H}_\lambda\right]^{-1} \mathbf{h}_\lambda^T$ ▷ Non zero diffusion constrained flow
      $\mathbf{y}_j = \mathbf{y}_{j-1} + \mathbf{m}(\mathbf{y}_j)\Delta\lambda_j$ ▷ Propagate the particles in pseudo time
    end for
    $\bar{\mathbf{x}}^i_{k+1} = \mathbf{y}_{N_\lambda}$
  end for
  Evaluate the posterior mean $\bar{\boldsymbol{\mu}}_{k+1}$ and covariance matrix $\bar{\mathbf{P}}_{k+1}$
  REDRAWPARTICLES($\{\bar{\mathbf{x}}^i_{k+1}\}^{N_p}_{i=1}$) ▷ Redraw particle (Optional)
  return $\{\mathbf{x}^i_{k+1}\}^{N_p}_{i=1}$, $\bar{\boldsymbol{\mu}}_{k+1}$, $\hat{\mathbf{P}}_{k+1}$
end procedure

4.5.2 Data reduction

Firstly, we tackle the issue of dimensionality reduction. Below, we list some of the methods which could be employed for reducing the number of measurements.

4.5.2.1 Naive subsampling

This is the most basic method for shrinking the measurement set, which can be done by throwing out elements at random keeping the overall set cardinality fixed. Alternatively, it can be done by taking every *mth* measurement. This method does not take the structure present in the data set into account. Though one of the simplest method, it could deteriorate the performance of the filter if the not enough samples are chosen.

4.5.2.2 Data Clustering

The next approach is based on clustering of the data points in the measurement space. Clustering turns out to be a quite effective means of dimensionality reduction. This has been a thoroughly studied topic, with applications in areas like image processing, computer vision, machine learning etc. We use two clustering methods, *K-means* clustering and *K-medoids* clustering with the partitioning done around medoids.

K-means clustering K-means clustering is essentially a vector quantization technique, with origins in voice compression, that eventually got popular for the data analysis. The basic idea in K-means clustering is to partition a fixed number of N-dimensional observations into K sets/clusters, where each observation belongs to the cluster whose centroid is closest to it. It is an iterative method consisting of two steps: *Expectation (E)* and the *Maximization (M)*. The process starts with choosing centroids as K random data points. In the E step, all data points are assigned to the one of the centroids. Usually the L2-norm is chosen as the metric for measuring inter-point distance. Next, the centroids locations are updated by averaging the points in their respective clusters. The procedure is carried out until the convergence or a fixed number of iterations have been carried out.

K-medoids clustering This is the second method we employ for the data clustering. A medoids is a point within a cluster whose average dissimilarity to all other points in the cluster is minimal. i.e.it is a most centrally located point in the cluster. While in K-mean clustering the centroid is usually not point with the cluster set (average of points), a medoids is always a point with in the cluster. Similar to the previous method, E and M steps are also iteratively followed in the K-medoids algorithm. K-medoids clustering is said to be more robust to noise and outliers as compared to K-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

4.5.3 Proposal density representation

As discussed before, the output of the DHF are the approximated posterior samples, represented through a Dirac-delta approximation. For them to be used as a proposal density within a MCMC step, they have to be further approximated by some closed form probability density expression. As described in the [KU15], the redrawing step in the Algorithm 9 (step 14) returns an approximated density, either as a single multivariate Gaussian (MVG) or as a Gaussian mixture model (GMM). In the current work, we follow a similar approach and use a MVG approximated form for the proposal density.

4.6 Sequential MCMC with DHF based proposal for massive sensor data processing

In [FSM15], MCMC is used together with the confidence sampler to estimate a high-dimensional non-Gaussian state space. The overall procedure is termed as adaptive Sequential Markov chain Monte Carlo (ASMCMC).

Algorithm 10 Adaptive SMCMC with particle flow based proposal

procedure SMCMCWITHPARTICLEFLOW($\{\mathbf{Z}_k\}_{k=0}^{k_{max}}$)
  Initialize the particle $\{\bar{\mathbf{x}}_0^i\}_{i=1}^{N_p}$ ▷ Initialize particles
  for k = 0 : $k_{max} - 1$ **do**
    $\mathbf{z}_{k+1}^c$ = CLUSTERMEASUREMENTS($\mathbf{z}_{k+1}$) ▷ Primary data decimation
    $\mathbf{z}_{k+1}^{\hat{c}}$ = LIKELIHOODBASEDCOMPRESSION($\mathbf{z}_{k+1}^c$) ▷ Sec. data decimation
    $\{\hat{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$ = PRIORSAMPLING($\{\bar{\mathbf{x}}_k^i\}_{i=1}^{N_p}$) ▷ Time update
    $\{\bar{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$ = LOGHOMOTOPYFLOWUPDATE($\{\hat{\mathbf{x}}_{k+1}^i\}_{i=1}^{N_p}$,$\{\Delta\lambda_j, \lambda_j\}_{j=1}^{N_\lambda}$,$\mathbf{z}_{k+1}^{c/\hat{c}}$)
    $q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)$ = GETPROPOSALDENSITY($\{\mathbf{x}_{k+1}^i\}_{i=1}^{N_p}$) ▷ Form the proposal
    Markov chain Monte Carlo
    Initialize the Markov chain: $\mathbf{x}_{k+1}^0$
    for m = 1 : $N_c + N_b$ **do**
      if m = 1 $\vee N_b$ **then**
        $\left[\{\mathscr{P}_i(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*)\}_{i=1}^{M_{k+1}}, \mathcal{R}_k^B\right]$ = UPPRXYRNG($\{\mathbf{x}_{k+1}^i\}_{i=1}^{N_p}$, $\mathbf{z}_{k+1}$)
      end if
      $\mathbf{x}_{k+1}^* \sim q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)$ ▷ Draw from the proposal distribution
      $\psi(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*) = \frac{1}{M}\log\left[u\frac{p(\mathbf{x}_{k+1}^*|\mathbf{Z}_k)q(\mathbf{x}_k^m|\mathbf{x}_{k+1}^*)}{p(\mathbf{x}_{k+1}^m|\mathbf{Z}_k)q(\mathbf{x}_{k+1}^*|\mathbf{x}_{k+1}^m)}\right]$
      $[N_{m,k}, \Lambda_{N_m}]$ =CONFIDENCESAMPLER($\mathbf{x}_{k+1}^m$,$\mathbf{x}_{k+1}^*$,$\mathbf{z}_{k+1}$,δ,$\psi(.)$)
      if $\Lambda_{N_m} > \psi(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*)$ - $\frac{1}{M_{k+1}}\sum_{i=1}^{n}\mathscr{P}_i(\mathbf{x}_{k+1}^m, \mathbf{x}_{k+1}^*)$ **then** ▷ Modified MH
        $\mathbf{x}_{k+1}^m = \mathbf{x}_{k+1}^*$
      else
        $\mathbf{x}_{k+1}^m = \mathbf{x}_{k+1}^m$
      end if
    end for
    $\bar{p}(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\delta(\mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}^i)$ ▷ MCMC based posterior density estimate
  end for
end procedure

The algorithm is based on two main steps: a initial joint drawing of the $\mathbf{x}_{k+1}, \mathbf{x}_k$ with the target density,

$$p(\mathbf{x}_{k+1}, \mathbf{x}_k|\mathbf{Z}_k) \propto p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_k) \tag{4.19}$$

and a secondary refinement step redrawing both of these variables individually. The latter step is composed of two further substeps. All three sub stages use a Metropolis-Hastings step, with

the first and the third employing the confidence sampling since the likelihood evaluations are involved. Variants of transitional density are used as proposals distributions in the MH steps. ASMCMC has been shown to sample the posterior density reasonably well, with lesser execution time when compared to the crude MCMC, while still achieving reasonable compression gain. The confidence sampling is the key in reducing the computational burden. Proxy terms and the upper-bounded range are calculated only twice; once at the start of MCMC, and secondly, after the burn-in phase has expired. In the current work, we make a distinction from [FSM15], in that we specifically use a DHF based proposal distribution with in the Sequential MCMC. Since all components have been described in the earlier sections, the task at hand is to embed all of them within a unified framework. We call this scheme adaptive SMCMC with particle flow based proposal or ASMCMC-DHF, and is described in the Algorithm 10.

4.7 Model & Simulation setup

In order to test the performance of our algorithm, we consider multi-target tracking scenario in the presence of clutter, similar to the one in [FSM15]. However, a major distinction is that we use a nonlinear measurement model. Observations are generated by a sensor located at the origin, and consist of range and bearing of targets. State vector for the target i at time instant k is $\mathbf{x}_k^{(i)} = (x_k^{(i)}, y_k^{(i)}, \dot{x}_k^{(i)}, \dot{y}_k^{(i)})$, where $x_k^{(i)}$ and $y_k^{(i)}$ represent the position while $\dot{x}_k^{(i)}$ and $\dot{y}_k^{(i)}$ representing velocity components along the x and y-axis respectively. The overall state vector is formed by concatenating the individual target state vectors $\mathbf{x}_k = [\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} \dots \mathbf{x}_k^{(N_T)}]$. We assume a discrete white noise acceleration model (DWNA) $p(\mathbf{x}_{k+1}|\mathbf{x}_k^i) = \mathcal{N}(\mathbf{x}_{k+1}|\mathbf{F}\mathbf{x}_k^i, \mathbf{Q})$ with transition matrix F given by,

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_2 & T_s\mathbf{I}_2 \\ \mathbf{0}_2 & T_s\mathbf{I}_2 \end{bmatrix} \tag{4.20}$$

and the measurement noise covariance matrix,

$$\mathbf{Q} = \sigma_x^2 \begin{bmatrix} (T_s^3/2)\mathbf{I}_2 & (T_s^2/2)\mathbf{I}_2 \\ (T_s^2/2)\mathbf{I}_2 & T_s\mathbf{I}_2 \end{bmatrix} \tag{4.21}$$

Since we are considering massive sensor data scenario, multiple measurements per target are generated. The number of measurements per targets are considered to be Poisson distributed with intensity λ_x. In addition to the target returns, there are clutter measurements whose number at any time instance is also Poisson distributed with intensity λ_c. Furthermore, target association is not assumed to be known, but we do not use any data association algorithm. This is justified as the main purpose of this work is to test the efficacy of the use of DHF together with ASMCMC. The total number of measurements received at the time instance k+1 is given by $\mu_{k+1} = N_T M_{k+1}^x + M_{k+1}^c$, where M_{k+1}^x represents the number of measurements per target, while M_{k+1}^c is the number of clutter measurements. The joint likelihood can then be expressed as,

$$l(\mathbf{x}_{k+1}) = \frac{e^{-\mu_{k+1}}}{M_{k+1}!} \prod_{i=1}^{M_{k+1}} \left[\lambda_c p_c(\mathbf{z}_{k+1}^i) + \lambda_x \sum_{j=1}^{N_T} p_x(\mathbf{z}_{k+1}^i|\mathbf{x}_{k+1,j}) \right] \tag{4.22}$$

The measurement vector for the target *i* is given by $\mathbf{z}_{k+1}^{(i)} = (r_{k+1}^{(i)}, \theta_{k+1}^{(i)})$, where $r_{k+1}^{(i)}$ is the range to the target while $\theta_{k+1}^{(i)}$ is the target bearing, resulting in the following form of the sub-likelihood,

$$p_x(\mathbf{z}_{k+1}^{i}|\mathbf{x}_{k+1,j}) = p_x^{i,j} = \mathcal{N}(\mathbf{z}_{k+1}^{i}|\mathbf{h}(\mathbf{x}_{k+1,j}), \mathbf{R}_{k+1}) \tag{4.23}$$

with,

$$\mathbf{h}(\mathbf{x}_{k+1,j}) = \left[\sqrt{x_{k+1,j} + y_{k+1,j}}, \tan^{-1}\left(\frac{y_{k+1,j}}{x_{k+1,j}}\right)\right]^T \tag{4.24}$$

where R_{k+1} being the measurement covariance matrix. The clutter measurements are independent of the target measurements and are uniformly distributed with in the surveillance area of sensor.

$$p_c(\mathbf{z}_{k+1}^{i}) = \mathcal{U}_x(\mu(z_{k+1,r}, z_{k+1,\theta}))\mathcal{U}_y(\mu(z_{k+1,r}, z_{k+1,\theta})) \tag{4.25}$$

We consider tracking of two targets (N_T=2), under two separate conditions: a moderately massive sensor data scenario with λ_x=50 and λ_c=200, and a extremely massive sensor data scenario with λ_x = 500 and λ_c = 2000. In the subsequent analysis, we will refer to the former case as the MSD1 (Massive Sensor Data 1), while the later as MSD2 (Massive Sensor Data 2). As the standard setting, σ_a^2 is set to 0.5 $\mathrm{m}s^{-2}$, σ_r^2 to $10m^2$ while σ_θ^2 is set to $0.01\mathrm{rad}^2$. The two targets start at position (-50 , -50) and (30 , 30), whereas initial velocities for the two targets are given by (-0.1 , -0.1) and (0.1 , 0.5) respectively. We note that $\sigma_r < D_{i,k}\sigma_\theta \ \ \forall i, k$, where $D_{i,k}$ represents the distance of *ith* target from the radar location at time instant k. As a consequence, the sub-likelihood functions have characteristic banana-like shapes due to the higher bearing measurement uncertainty. We consider 100 particles for DHF, $N_p = 100$, and 30 geometrically spaced pseudo-time points for solving flow ODE, as required in line 7 of the Algorithm 10. Also, we approximate the proposal density $q(\mathbf{x}_{k+1}^{*}|\mathbf{x}_{k+1}^{m})$ using the log-homotopy flow updated particles through a multivariate Gaussian density. We use root average mean square error (RAMSE) as the performance metric, which is defined as the following. Let N_{sim} be the total number of simulation runs for a particular scenario, $x_k^{i,m}$ and $y_k^{i,m}$ denote the positions of the *ith* target along X and Y-axis respectively, at time instant *k* in the mth trial. Likewise, let $\hat{x}_k^{i,m}$ and $\hat{y}_k^{i,m}$ denote estimated positions for the *ith* target. The RAMSE $\epsilon_r(k)$, where *k*>0, is then defined as,

$$\epsilon_r(k) = \sqrt{\frac{1}{N_{sim}}\sum_{m=1}^{N_{sim}}\left[\frac{1}{2N_T}\sum_{i=1}^{N_T}\left(\left(x_k^{i,m} - \hat{x}_k^{i,m}\right)^2 + \left(y_k^{i,m} - \hat{y}_k^{i,m}\right)^2\right)\right]}$$

We simulated each scenario a total of fifty times ($N_{sim} = 50$), with each simulation running for a total of 50 seconds. The standard parameter setting for our Algorithm is shown in Table 4.1,

Parameter	Value	Parameter	Value	Parameter	Value
N_p	100	N_{mcmc}	400	N_{burn}	$N_{mcmc}/4$
γ_{mcmc}	1.5	δ_s	0.1	p_s	2
T_s	1	λ_x	50/500	λ_c	200/2000
σ_x^2	0.5	σ_r^2	10	σ_θ^2	0.01
A_{meas}	400×400	T_{sim}	50	K_{clust}	30

Table 4.1: Algorithm parameters

The proxy update step, mentioned in the line 13 of the Algorithm 10, basically refers to calculation of the proxy terms and the range measure estimates as given by the (4.14) and (4.18) respectively. These estimates are based on the evaluation of the gradient and the Hessian of the log-likelihood terms. Its is hoped that the evaluation of these quantities is easier to carry out than the actual likelihood evaluation. Also, this is done only twice while running the chain; first in the beginning, and second after the burn-in phase has been expired. Also the linearization point $\mathbf{x}_{k+1}^{+}$ is chosen to be the mid point between the current and the proposed states $\mathbf{x}_{k+1}^{m}$ and $\mathbf{x}_{k+1}^{*}$.

Our analysis is twofold. First, we analyze the effect of some of the important parameters on the performance of ASMCMC-DHF by varying those within a certain range. Secondly, we compare the performance of ASMCMC-DHF against other estimation methods, namely the particle filter and ASMCMC.

4.8 Results

We study the results of changing parameters of sub-components of the Algorithm 10. This include: the primary data decimation (clustering), the secondary likelihood based data decimation step, and the MCMC step parameters N_{mcmc} and γ_{mcmc}. We also consider the effect of changing the system parameters like the process and measurement noise covariances (angle), σ_a^2 and σ_θ^2, respectively. Additionally, we also study the effect of increasing the state dimensionality on the estimation error. We gauge the overall performance by considering three variables: root average mean square error (RAMSE), acceptance rate and the compression ratio. Acceptance rates and the compression ratios are calculated after the confidence sampling based MCMC procedure has been carried out. The compression ratio is defined as, the total number of measurements available divided by the number of measurements actually used by the confidence sampler i.e. $|\mathcal{Z}_k|$; a higher value is indicative of a lesser amount of data usage and lower processing time and the vice-versa. Processing time is also considered when analyzing the algorithm sub-components.

4.8.1 Primary data decimation

The aim here is to study the effect of changing the number of measurements subsampled in the initial step, using methods mentioned in 4.5.2. Number of measurements after the initial stage data decimation via clustering is controlled by the parameter K_{clust}, for which we use three value: 10,20 and 30. The purpose is to significantly reduce the number of measurements used by the DHF while forming the proposal for the ASMCMC step. Clustering leads to a smaller set of measurements (depending on the value of K_{clust}), which requires a change to be made in measurement intensities λ_x and λ_c. The two intensities then become: $\hat{\lambda}_x = K_{clust} \times \frac{N_T \lambda_x}{\lambda_c + N_T \lambda_x}$ and $\hat{\lambda}_c = K_{clust} \times \frac{\lambda_c}{\lambda_c + N_T \lambda_x}$, respectively. Below, we plot typical measurement scans superimposed with the clustering centroids (K_{clust} = 30) for K-Means, for the two sets of λ_x and λ_c,

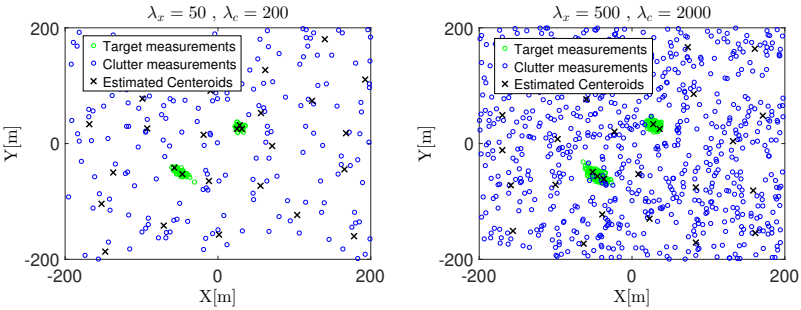


Figure 4.1: A typical measurement scan with K-Means clustering applied

We plot similar figures using K-medoids clustering, for both of the two sets of λ_x and λ_c,

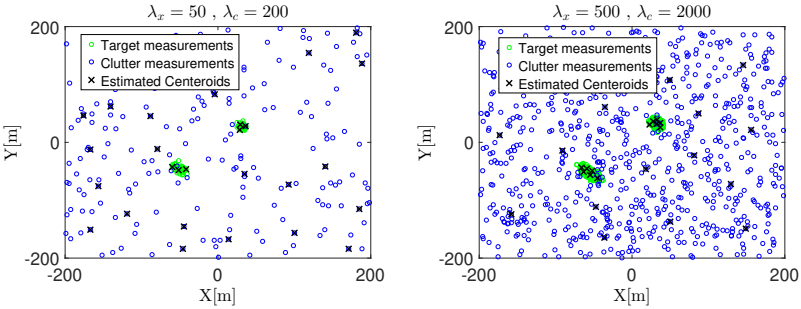


Figure 4.2: A typical measurement scan with K-Medoids clustering applied

For the first case (MSD1), we can see that centroids are less uniformly arranged, whereas for MSD2 we see a more regular pattern. Secondly, since clutter is uniformly spread with a density four times that of the target on the average, target centroids can be affected by the nearby clutter measurements.

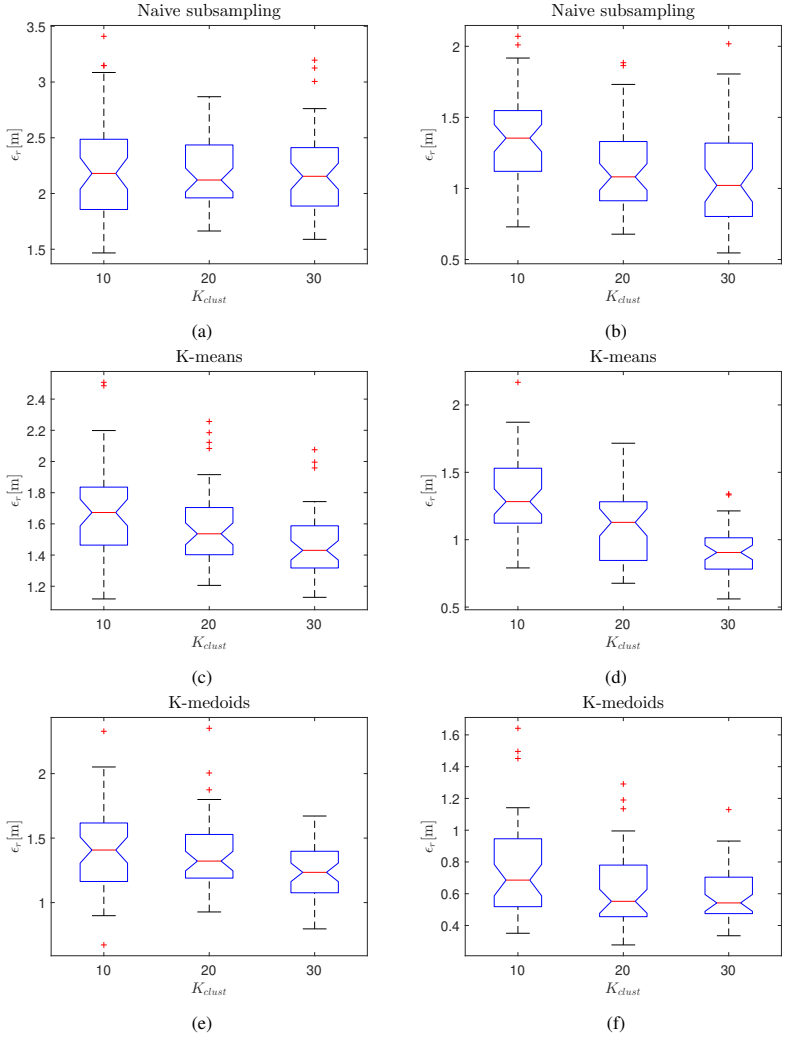


Figure 4.3: RAMSE for data reduction schemes with MSD1(λ_x = 50, λ_c=200) (a,c,e) and MSD2 (λ_x = 500, λ_c=2000) (b,d,f)

$\lambda_{\mathbf{x}} = \mathbf{50}$, $\lambda_{\mathbf{c}} = \mathbf{200}$				
Method	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
Naive (K_{clust}=10)	(1.47 , 2.32 , 3.41)	(0.13 , 0.17 , 0.24)	(1.30 , 1.69 , 2.45)	(13.35 , 52.44 , 89.21)
K_{clust} = 20	(1.66 , 2.12 , 2.87)	(0.13 , 0.20 , 0.28)	(1.31 , 1.72 , 2.59)	(23.62 , 61.81 , 94.95)
K_{clust} = 30	(1.59 , 2.18 , 3.20)	(0.17 , 0.22 , 0.28)	(1.43 , 1.77 , 2.40)	(33.19 , 70.54 , 106.93)
K-means (K_{clust}=10)	(1.12 , 1.67 , 2.51)	(0.10 , 0.16 , 0.20)	(1.55 , 1.89 , 2.24)	(14.75 , 52.67 , 89.48)
K_{clust} = 20	(1.21 , 1.54 , 2.26)	(0.13 , 0.19 , 0.24)	(1.78 , 2.11 , 2.44)	(24.98 , 63.40 , 100.90)
K_{clust} = 30	(1.13 , 1.43 , 2.08)	(0.12 , 0.22 , 0.26)	(1.84 , 2.29 , 2.51)	(36.27 , 70.90 , 104.50)
K-medoids (K_{clust}=10)	(0.90 , 1.44 , 2.33)	(0.13 , 0.19 , 0.28)	(1.79 , 2.26 , 2.71)	(14.97 , 56.15 , 90.94)
K_{clust} = 20	(0.93 , 1.32 , 2.35)	(0.14 , 0.21 , 0.30)	(1.83 , 2.47 , 2.88)	(23.97 , 65.45 , 99.05)
K_{clust} = 30	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 77.14 , 116.12)
$\lambda_{\mathbf{x}} = \mathbf{500}$, $\lambda_{\mathbf{c}} = \mathbf{2000}$				
Naive (K_{clust}=10)	(0.73 , 1.34 , 2.07)	(0.10 , 0.18 , 0.24)	(1.53 , 2.04 , 2.40)	(18.63 , 157.21 , 304.26)
K_{clust} = 20	(0.68 , 1.12 , 1.88)	(0.13 , 0.20 , 0.25)	(1.65 , 2.12 , 2.58)	(30.93 , 167.03 , 312.34)
K_{clust} = 30	(0.55 , 1.03 , 2.02)	(0.17 , 0.24 , 0.29)	(1.64 , 2.15 , 2.69)	(41.77 , 181.89 , 320.69)
K-means (K_{clust}=10)	(0.79 , 1.28 , 2.17)	(0.11 , 0.15 , 0.19)	(1.89 , 2.22 , 2.80)	(17.83 , 157.57 , 289.16)
K_{clust} = 20	(0.68 , 1.13 , 1.72)	(0.12 , 0.18 , 0.21)	(1.90 , 2.29 , 2.86)	(32.99 , 170.19 , 314.78)
K_{clust} = 30	(0.56 , 0.91 , 1.34)	(0.12 , 0.19 , 0.24)	(2.03 , 2.34 , 3.12)	(45.81 , 199.71 , 369.69)
K-medoids (K_{clust}=10)	(0.35 , 0.69 , 1.64)	(0.14 , 0.17 , 0.21)	(1.84 , 2.55 , 2.74)	(17.38 , 154.33 , 291.27)
K_{clust} = 20	(0.28 , 0.55 , 1.29)	(0.14 , 0.19 , 0.24)	(1.87 , 2.62 , 2.81)	(32.34 , 170.28 , 299.81)
K_{clust} = 30	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(43.47 , 179.37 , 307.93)

Table 4.1: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio for the primary data decimation

For the naive subsampling, K_{clust} number of measurements are chosen at random from the whole set. Roughly speaking, this results in the selection of target measurements with the probability $\frac{N_T\lambda_x}{\lambda_c+N_T\lambda_x}$, while the clutter measurements are chosen at random with probability $\frac{\lambda_c}{\lambda_c+N_T\lambda_x}$. Since the data set is quite large, we have given an approximation ignoring the effect of sampling without replacement. In Figure 4.3, we present the box plot for RAMSE for the two cases of massive sensor data considered, while further elaborating the results in Table 4.1. For the number of subsampled measurements considered, error for the naive subsampling seems not be affected much by the increase in K_{clust}. This can be attributed to the uniform measurement sampling, where the probability of choosing target measurements increases only slightly with the increase in K_{clust}. On the other hand, we see a more pronounced decreasing trend in error with the increase in number of clusters, for both K-means and K-medoids clustering. The latter has lesser error for all values of K_{clust}, which can be attributed to the closeness of the target measurement centroids to the true values i.e. ones with out noise. Another thing to be observed is that the error for MSD1 is higher than for MSD2. This is understandable, as although the number of measurements after the initial clustering are the same, the whole set of measurements is available to be sampled from in the later ASMCMC step. Secondly, the centroids tend to be closer to the true values for MSD2.
We note that as the value of K_{clust} increases, the acceptance ratio is decreased. On the other hand, we see a rising trend for the compression ratio. With more number of measurements per targets available, the quality of the proposal density get better. In other words, the proposal gets even closer to the target distribution (posterior), and therefore in the MH step, not many samples are likely to be accepted. This also has a positive effect on the probabilistic subsampling of measurements as fewer samples are required to meet the stopping criterion. Next, we compare the execution speed for all of the schemes by plotting the median processing time in Figure 4.4,

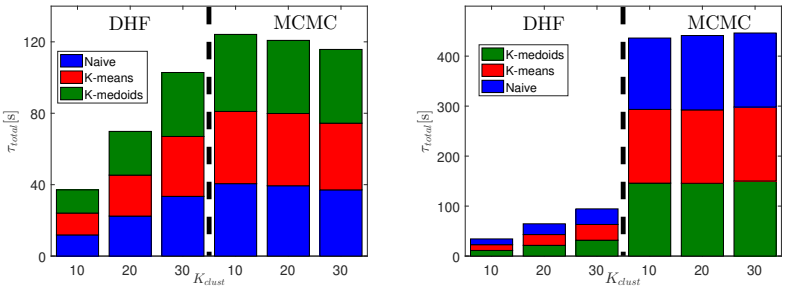


Figure 4.4: Median processing time for data reduction schemes for (a)(λ_x = 50, λ_c=200) and (b)(λ_x = 500, λ_c=2000)

We have further classified the processing time into the two categories; time for generating proposal through the DHF and the time taken by the confidence sampling based MCMC stage. Firstly, we discuss the results for MSD1. We note that the DHF processing time increases linearly with the increase in the value of K_{clust}, which is similar for all methods. Since the flow equation for individual particle requires the evaluation of the gradient and Hessian of the log-

likelihood (which involves summation over all measurements 4.A), the cost grows linearly with the increase in the number of measurements. On the other hand, we note a slight decrease in the processing time for MCMC step. This can be attributed to the better quality of the DHF based proposal used, resulting in higher compression ratios. DHF processing takes almost the same time as running the MCMC chain for $K_{clust} = 30$. In the case of MSD2, we observe a similar trend for the DHF processing times, while for MCMC they increase slightly with the increase in the number of measurements. This might be an indication of a comparatively weaker proposal, and potentially there is more to be gained by further increasing the number of clusters. But that would come with an added processing cost. In the current setting, DHF takes almost 4 times less to execute than the MCMC in the best case. A further increase in the number of clusters, though could result in a slightly decreased error, it will come at a cost of much increased processing time. Therefore based on the results, we choose K-medoids clustering with $K_{clust} = 30$ as the default method from now onwards.

4.8.2 Likelihood based data decimation

Next, we study the effect of likelihood based decimation of measurements. This is an optional secondary level data compression method, which is based on choosing the most likely samples from the set of the previously clustered measurements. It works like this; the previous posterior state estimate is used for predicting the next state vector. Given this estimate, likelihood is evaluated against all measurement medoids. The list is sorted and K_{scnd} number of measurements are chosen. Since, the objective is to reduce the computation time for the DHF even further, we have chosen $K_{scnd} = N_T$ i.e two most likely samples. It is hoped that in this way, the one most likely measurement per target is selected. RAMSE, with the secondary stage decimation (Double DD) applied is compared below against the single stage data decimation,

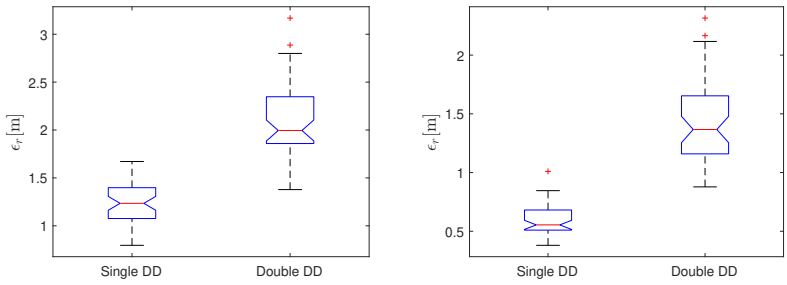


Figure 4.5: RAMSE for secondary data decimation step for (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

We note a higher estimation error with the likelihood based data decimation. Though seemingly inconsistent, this can be explained by considering the fact that only two measurements are fed to the DHF. Depending on the efficiency of the clustering process, these measurements may or may belong to the two targets e.g. in the scan with very strong clutter, target measurements could get masked. Therefore, clutter returns could be chosen instead, resulting in a bad proposal

for the ASMCMC and the subsequent posterior estimate. The problem can become exacerbated in the presence of an even stronger clutter, or in the case when the paths of the two targets cross-over.

$\lambda_{\mathbf{x}} = 50$, $\lambda_{\mathbf{c}} = 200$				
Data Dec.	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
Single DD	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 77.14 , 116.12)
Double DD	(1.38 , 2.11 , 3.17)	(0.11 , 0.17 , 0.19)	(1.48 , 1.81 , 2.13)	(5.59 , 42.78 , 81.04)
$\lambda_{\mathbf{x}} = 500$, $\lambda_{\mathbf{c}} = 2000$				
Single DD	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(32.34 , 170.28 , 299.81)
Double DD	(0.88 , 1.38 , 2.32)	(0.12 , 0.16 , 0.22)	(1.79 , 2.21 , 2.85)	(14.17 , 150.72 , 280.95)

Table 4.2: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio for the secondary data decimation

In Table 4.2, it can be seen that acceptance rate is lower for double DD for both MSD1 and MSD2. This is because, the constructed Markov chain is not long enough to sample from the high probability regions of the posterior density. Compression ratio is also lower for double DD as more samples are chosen by confidence sampler to get closer to the target distribution, although the processing time for double DD is slightly less. The reduction mainly is due to the use of far fewer measurements in the DHF. In the light of these results, we choose not to use the secondary data decimation in our subsequent analysis.

4.8.3 MCMC chain length N_{mcmc}

The Markov chain length is a very important parameter in the Algorithm 10. A shorter chain may not reach stationarity, while a longer chain length could be wasteful of the resources. Hence, an appropriate length has to be chosen to strike a right balance. We consider three values of the Markov chain length: 100 , 400, 1000 with a burn-in period of 25, 100 and 250 respectively. Below, we plot RAMSE against the chosen values of N_{mcmc} for both MSD1 and MSD2

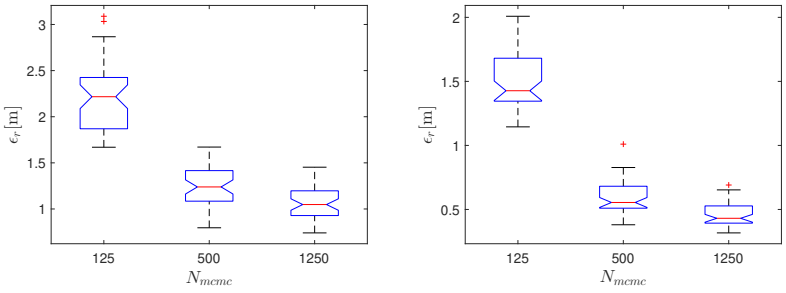


Figure 4.6: RAMSE for different values of Markov chain lengths N_{mcmc} with (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

We note that for $N_{mcmc} = 125$, the error is the highest. RAMSE drops considerably by using the chain of length 500, while a further increase to the length of 1250 eror is reduced even

further, although the difference is rather small. By interpolating the trend, it can be argued that yet longer Markov chains won't lead to a very significant drop in the error, in particular for MSD1. As will be shown later, the error has already reached very close to the Crámer-Rao Lower Bound (CRLB). RAMSE for MSD2, on other hand, can still undergo further decrease in error as CRLB is yet to be reached. The second consideration here is the processing time. As the chain length is increased, it takes more time to draw samples. Since the number of measurements are generated via Poisson processes and the SMCMC stage uses a probabilistically chosen measurement subset, the exact relation between the processing time and the chain length is not linear.

$\lambda_{\mathbf{x}} = 50, \lambda_{\mathbf{c}} = 200$				
N_{mcmc}	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
125	(1.65 , 2.19 , 3.09)	(0.16 , 0.27 , 0.32)	(1.79 , 1.97 , 2.34)	(33.03 , 42.92 , 55.07)
500	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 77.14 , 116.12)
1250	(0.89 , 1.17 , 1.55)	(0.11 , 0.23 , 0.28)	(1.83 , 2.67 , 2.88)	(40.34 , 125.11 , 212.11)
$\lambda_{\mathbf{x}} = 500, \lambda_{\mathbf{c}} = 2000$				
125	(0.89 , 1.18 , 1.76)	(0.05 , 0.22 , 0.38)	(2.05 , 2.42 , 2.94)	(41.59 , 74.28 , 112.13)
500	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(43.47 , 179.37 , 307.93)
1250	(0.31 , 0.49 , 0.8)	(0.13 , 0.21 , 0.24)	(1.88 , 2.77 , 2.93)	(63.02 , 411.47 , 751.84)

Table 4.3: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio vs. N_{mcmc}

It can be seen from Table 4.3 that the acceptance rate for the shortest chain length is the greatest and vice versa. This is intuitive, as more samples are expected to be accepted while stationarity has not been reached. The acceptance rate saturates around 0.234 as the chain length is sufficiently long [RC04]. Compression ratio, infact increases with the increase in the chain length. Again this is indicative of the chain reaching stationarity. It still has to be studied as to what value the compression ratio would saturate to.

4.8.4 Block length parameter γ_{mcmc}

The confidence based proxy sampler iteratively increase the measurement block size until the stopping criterion is met or whole set has been exhausted. One important parameter is the block length parameter γ_{mcmc}. It controls the size of the subsampled data set sampled from the existing set without replacement. Below, we plot RAMSE for four different values of γ_{mcmc}: 1.01 , 1.5 , 2.0 and 4.0. We note the a sharp drop in the error between first two values of γ_{mcmc} considered. For MSD1, the error stabilizes afterwards, while for MSD2 it continues to drop even further, albeit more slowly. We see a decreasing trend for the acceptance rate and the compression ratio with the increase in γ_{mcmc}. The later in particular decrease quite significantly, as the block length parameter is increased from 2 to 4. This can be explained in the following way: when γ is close to 1, the increase in the subsampled block size is very small. As the measurements are sampled at random, the chances of choosing the measurements belonging to the target (ones which make the stopping criterion more likely to be met) are almost equal to those mentioned for naive subsampling in the section4.8.1.

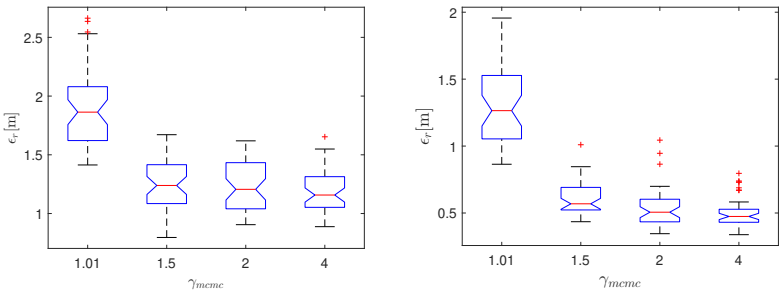


Figure 4.7: RAMSE for different values of γ_{mcmc} with (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

This could mean that very often, the stopping criterion is met quite soon into the iterations, leading to a significantly high compression ratio. RAMSE, on the other hand, is higher as a smaller subset of samples is used in the MH step and the target distribution is not evaluated properly. With the increase in the γ_{mcmc}, more samples could be chosen in one iteration of the confidence sampler. This implies that more target measurements are available more often for the likelihood evaluations, hence only better proposed samples are chosen i.e.lower RAMSE, acceptance rate and compression ratio.

$\lambda_{\mathbf{x}} = \mathbf{50}, \lambda_{\mathbf{c}} = \mathbf{200}$				
γ_{mcmc}	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
1.01	(1.41 , 1.86 , 2.66)	(0.21 , 0.30 , 0.42)	(2.12 , 3.03 , 3.34)	(35.28 , 65.72 , 109.86)
1.5	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 74.14 , 116.12)
2	(0.91 , 1.21 , 1.62)	(0.14 , 0.18 , 0.24)	(1.23 , 1.40 , 1.65)	(38.50 , 75.40 , 121.70)
4	(0.89 , 1.16 , 1.65)	(0.11 , 0.15 , 0.19)	(1.18 , 1.30 , 1.47)	(39.20 , 79.16 , 119.20)
$\lambda_{\mathbf{x}} = \mathbf{500}, \lambda_{\mathbf{c}} = \mathbf{2000}$				
1.01	(0.86 , 1.26 , 1.96)	(0.17 , 0.27 , 0.35)	(2.56 , 3.65 , 5.73)	(38.99 , 161.72 , 424.59)
1.5	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(43.47 , 179.37 , 307.93)
2	(0.34 , 0.53 , 0.88)	(0.09 , 0.14 , 0.18)	(1.66 , 1.95 , 2.38)	(46.00 , 179.47 , 300.26)
4	(0.34 , 0.47 , 0.80)	(0.09 , 0.12 , 0.16)	(1.63 , 1.82 , 2.01)	(64.75 , 194.96 , 280.97)

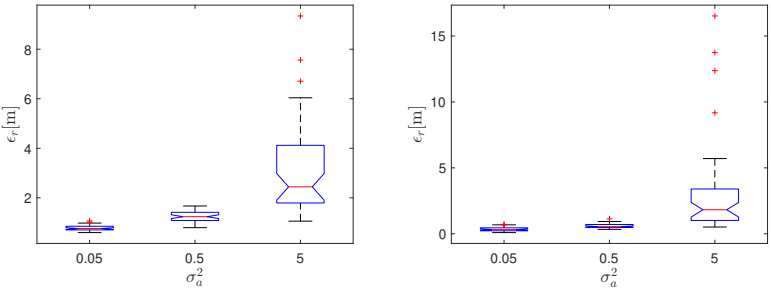
Table 4.4: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio vs. γ_{mcmc}

Again, we note that MSD2 has a slightly more potential for the decrease in error with the corresponding increase in γ_{mcmc}. Also more outlier are present in for MSD2.

Based on the results, we can infer that while a smaller value of γ_{mcmc} might be undesirable due to higher error, a larger value is also not suited because of the significant decrease in the compression ratio which directly translates into a higher number of likelihood evaluations, thereby defeating the very purpose of confidence sampling. Therefore, a middle ground is stuck and γ_{mcmc} is chosen to be 1.5 for the onward analysis.

4.8.5 Process noise covariance

Now that we have studied the effect of algorithm parameters, we turn our attention to the parameters of the system model considered in this exercise. Firstly, we discuss the effect of changing the process noise σ_a^2. We have considered three values: 0.05, 0.5 and 5 ms^{-2}.



(a) (b)

Figure 4.8: RAMSE for different values of σ_a^2 with (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

It can be readily seen that the error increase sharply with the increase in process noise covariance. In particular, we see that for σ_a^2 = 5, while the medians for both MSD1 and MSD2 still grow in a controlled manner, there are many outliers present with some having error 3 to 8 times the median. This is due to the fact that target trajectories are more likely to cross over in the case of stronger process noise. Since, no target association is considered, DHF particles can be wrongly updated leading to ill proposed states. Even the SMCMC, with the current choice of parameters (N_{mcmc} , γ_{mcmc}) is unable to properly sample the posterior distribution. Hence, the results are not very surprising.

$\lambda_x = 50$, $\lambda_c = 200$			
σ_a^2	**RAMSE**	**Acceptance rate**	**Compression ratio**
0.05	(0.61 , 0.78 , 1.07)	(0.14 , 0.26 , 0.35)	(2.54 , 2.90 , 3.73)
0.5	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)
5	(1.25 , 2.69 , 9.34)	(0.11 , 0.15 , 0.20)	(1.35 , 1.68 , 2.17)
$\lambda_x = 500$, $\lambda_c = 2000$			
0.05	(0.09 , 0.32 , 0.72)	(0.19 , 0.23 , 0.26)	(2.45 , 3.78 , 8.17)
0.5	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)
5	(0.51 , 1.82 , 16.52)	(0.04 , 0.07 , 0.15)	(1.55 , 2.28 , 4.27)

Table 4.5: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio vs. σ_a^2

We see a decrease in the acceptance rate and the compression ratio as well.

4.8.6 Measurement noise covariance

Next parameter considered in the series is the angular measurement noise covariance σ_θ^2. The reason of choosing angle instead of range is that the σ_θ^2 has a higher impact on the tracking

performance. We consider three values: 0.001 , 0.01 , 0.1 mrad^{-2}.

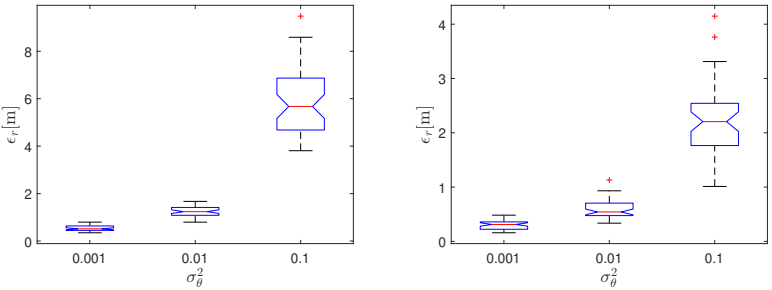


Figure 4.9: RAMSE for different values of σ^2_θ with (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

As can be seen from Figure 4.9, the estimation accuracy is quite severely affected by the increase in σ^2_θ. Infact, the median error grow almost exponentially with the corresponding increase in noise power. Therefore, it can be concluded that the increase in measurement noise covariance is more detrimental to the filter efficiency than a similar increase in the process noise.

$\lambda_{\mathbf{x}} = 50$, $\lambda_{\mathbf{c}} = 200$			
σ^2_θ	**RAMSE**	**Acceptance rate**	**Compression ratio**
0.001	(0.35 , 0.52 , 0.80)	(0.17 , 0.24 , 0.32)	(1.83 , 3.01 , 5.23)
0.01	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)
0.1	(3.81 , 5.67 , 9.48)	(0.08 , 0.12 , 0.15)	(1.09 , 1.24 , 1.42)
$\lambda_{\mathbf{x}} = 500$, $\lambda_{\mathbf{c}} = 2000$			
0.001	(0.16 , 0.31 , 0.48)	(0.18 , 0.23 , 0.30)	(1.91 , 2.87 , 6.37)
0.01	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)
0.1	(1.01 , 2.21 , 4.15)	(0.11 , 0.16 , 0.21)	(1.87 , 2.14 , 2.74)

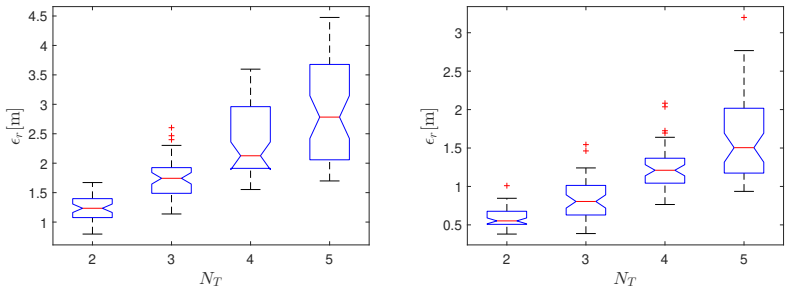
Table 4.6: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio vs. σ^2_θ

Interestingly, the compression ratio can be quite high for σ^2_θ = 0.001 at times, and the acceptance rate quite close to optimal.

4.8.7 Increase in the dimensionality

Higher dimensional state spaces present a major challenge to any estimation algorithm. This is infact due to the *curse of dimensionality*, which occurs due to not having enough particles to properly represent the relevant probability densities. In this subsection, we intend to study the effect of increasing the state dimensionality on the tracking performance. We had considered two targets in the preceding analysis. That corresponds to having the state vector of length 8. Here we vary the number of targets N_T between 2 to 5, corresponding to the state vector of length between 12 and 20, thereby presenting a challenging estimation scenario to our algorithm. The full initial state vector is chosen to be $\mathbf{x}_0^5 = (\mathbf{x}_0^1, \mathbf{x}_0^2, \mathbf{x}_0^3, \mathbf{x}_0^4, \mathbf{x}_0^5)^T$

where $\mathbf{x}_0^1 = (-50, -50, -0.5, -0.5)$, $\mathbf{x}_0^2 = (30, 30, 0.1, 0.5)$ $\mathbf{x}_0^3 = (-20, 20, -0.3, 0.1)$, $\mathbf{x}_0^4 = (50, -50, 0.2, -0.1)$ and $\mathbf{x}_0^5 = (0, 0, 0.01, 0.05)$, for $N_T = 5$. For the lesser number of targets, the initial state vector is derived by truncating $\mathbf{x}_0^5$. Below we present the RAMSE box error plot,



(a) (b)

Figure 4.10: RAMSE for different number of targets N_T with (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

We see that median error increase linearly with the increase in the state dimension, where as the maximum error shows a geometric trend. More outliers are to be found in the case of MSD2. This could mean two things: firstly, the proposal formed using the DHF gets less accurate and secondly, the Markov chain length more inadequate as the state dimensionality increases. The first issue can be tackled by increasing the number of DHF particles or by considering a better form of the proposal density. In the current analysis, the proposal density is approximated as being a Multivariate Gaussian. A better alternative could be approximating the DHF based proposal using the Kernel density estimator, as demonstrated in [KUK17]. For the second, longer chain lengths and higher value of γ_{mcmc} can be consider for the MCMC step.

$\lambda_{\mathbf{x}} = 50$, $\lambda_{\mathbf{c}} = 200$				
N_T	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
2	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 77.14 , 116.12)
3	(1.22 , 1.75 , 2.60)	(0.11 , 0.17 , 0.23)	(1.37 , 1.85 , 2.18)	(42.96 , 92.56 , 138.18)
4	(1.68 , 2.15 , 3.60)	(0.08 , 0.14 , 0.20)	(1.22 , 1.39 , 1.59)	(58.59 , 119.06 , 167.75)
5	(1.70 , 2.81 , 4.48)	(0.09 , 0.12 , 0.17)	(1.21 , 1.36 , 1.51)	(72.49 , 145.67 , 202.71)
$\lambda_{\mathbf{x}} = 500$, $\lambda_{\mathbf{c}} = 2000$				
2	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(43.47 , 179.37 , 307.93)
3	(0.45 , 0.80 , 1.54)	(0.08 , 0.15 , 0.19)	(1.75 , 2.25 , 2.54)	(49.99 , 217.03 , 378.75)
4	(0.77 , 1.21 , 2.08)	(0.05 , 0.10 , 0.13)	(1.77 , 2.04 , 2.36)	(74.43 , 313.74 , 514.19)
5	(0.94 , 1.48 , 3.20)	(0.04 , 0.06 , 0.08)	(1.78 , 2.01 , 2.24)	(123.08 , 511.09 , 826.15)

Table 4.7: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio vs. N_T

Acceptance rates and the compression ratios decrease with the increase in number of targets. From Table 4.7, it can be seen that the median processing time for MSD1 increases linearly

with the increase in the number of dimensions, while exponentially for the MSD2. This can be explained by the fact that the DHF proposal based on the current choice of parameters, gets progressively worse with the increase in the dimensionality. Hence, the MCMC step becomes the increasing important and it dominates the overall processing cost. The actual likelihood is very narrow (due to the product of higher number of terms), and full measurement block is used more often by the confidence sampler to nudge the particle cloud ever closer to the true posterior density. Though the same holds true for MSD2, but their the effect is not that prominent for the choice of the number of dimensions.

4.8.8 Comparison against other methods

Finally, we compare the performance of our proposed ASMCMC-DHF scheme against other methods. In the current analysis, we have used two such methods: the sampling importance resampling particle filter (SIR-PF) with 1000, 10000 and 25000 particles, and the the ASMCMC method as described in [FSM15] with 500 and 1250 MCMC chain lengths. The effort is made to make the comparison fair in the sense of similar execution times for all procedures. Simulation were run on a server running MATLAB version 7.9 with 2x Intel Xeon E5530 2.40 GHz processors and with 12GB of RAM. The operating system was Windows Server 2008 R2 x64 Std. In Figures 4.11 a&b, we plot the RAMSE for all schemes under comparison together with the Crámer-Rao Lower Bound (CRLB), while a tabulated description is provided in Table 4.8.

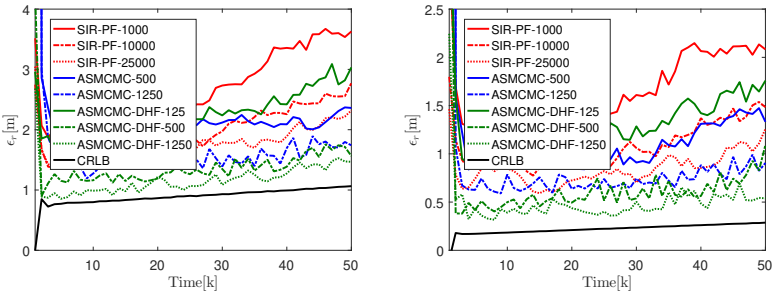


Figure 4.11: Comparison of ASMCMC-DHF with other schemes for (a) λ_x = 50, λ_c=200, (b) λ_x = 500, λ_c=2000

SIR-PF with 1000 particles is the worst performing method, showing a degree of divergence towards the end. This reflects the inadequacy of the number of samples to properly approximate the involved densities. The approximation gets progressively better as more samples are added. As can be seen for SIR-PF with 25000 particles, the RAMSE for both cases (MSD1 and MSD2) is still not close enough to the CRLB, thereby suggesting potential improvements can be achieved by further increasing the number of particles. Error in the case of MSD2 is expectedly less than for MSD1. Next, we discuss the results for ASMCMC. Again, we note significant drop in RAMSE by increasing the MCMC chain length from 500 to 1250. ASMCMC-500 seems to have performance similar to SIR-PF-10000. Increasing the chain length to 1250 makes its per-

formance somewhat close SIR-PF-25000. The difference is more noticeable in RAMSE for the two schemes for MSD1 than MSD2. Although, ASMCMC based schemes exhibit quite decent median acceptance rate they have rather insignificant compression ratios. This means that in order to achieve better performance, the whole data needs to be exhausted, therefore, defeating the very purpose of using the confidence sampling. This happens due to the use of rather simple Gaussian proposal densities in the two sampling stages. While this could work fine in some cases e.g. one with linear measurement model, for nonlinear measurements the suggested proposal is not the best choice. The compression ratio gets slightly better for MSD2.

Finally, we discuss the results for ASMCMC-DHF i.e.ùsing a DHF based proposal together with the confidence sampling for MCMC. We plot RAMSE for three values of N_{mcmc} considered in section 4.8.3 i.e.ì25, 500 and 1250. It is to be noted that 20% of initial samples of the MCMC chain are discarded. For chain of length 125, we note that the error is quite high, coming next only to SIR-PF-1000. This illustrates that although the clustering based DHF proposal is better than a simple particle filter, when used with shorter MCMC chains it could be detrimental to the overall performance. Since the MCMC chain is yet to reach stationarity, estimates formed thereof could be biased and might not be accurate. As a possible remedy, a higher value of γ_{mcmc} can be chosen when using shorter chains but that would lead to a lowering of the compression gain, thus nullifying potential benefits. We note drastic reduction in the error with the use of a moderate chain length of 500. With a further increase, we note that the error approaches CRLB limit for the MSD1. For MSD2, there is still further room for improvements.

At the very last, we discuss the processing time for a single update step (both time and measurement) for all procedures. Since the measurements are generated randomly for each scan using two independent Poisson processes, we observe a large variation in the processing times. SIR-PF-1000 is the quickest of all methods, while ASMCMC-1250 being the slowest. The later is because of the double use of confidence sampling. Furthermore, we note that SIR-PF-25000 and ASMCMC-DHF-1250 have execution times comparable to that of ASMCMC-500, although the later has higher error. ASMCMC-1250 can be seen as the most optimal method offering a right trade-off between the estimation accuracy and the execution speed. In the retrospect, it can be seen that the choice of the proposal density quite significantly affects the performance; a better choice e.g. based on DHF particles not only decreases the error but also takes lesser time for sampling the posterior density in the MCMC step.

4.9 Conclusion

The *massive sensor data* provides the possibility for extraction of more information content, given a large measurement set, thus increasing the estimation accuracy. However, this comes with enhanced computational requirements, hence limiting the use of many standard estimation methods such as MCMC. The source of the problem can be pinpointed to the evaluation of likelihood, which even in factorized format, presents formidable processing challenge. Many solution have been proposed to solve/by-pass this bottle-neck. One such approximate method, namely the confidence sampling is of particular interest. Confidence sampling squeezes the original observational data set into a smaller one to be processed by an MCMC sampler, while still maintaining theoretical guarantees for the sampled distribution. It is based on using the so called concentration inequalities, which can be used to theoretically bound the maximum

deviation of the approximated target density. When used in an MCMC setting, the use of such inequalities yields a stopping criterion for the sampling procedure. Though the target density is still approximated, there are potential processing gains to be achieved by limiting the evaluation of likelihood to fewer terms, together with the guarantee ensuring that the sampled density is always within a specified distance from the actual target density. The work done for this chapter is an extension of [FSM15]. We have combined the idea of confidence sampling based MCMC together with the log-homotopy based particle flow filters (DHF), in a way that the later is used to construct a better proposal distribution used within the former. Since the processing time for DHF grows linearly with the number of measurements, we choose a sub-sampled set of measurements from the full set. This is done by employing standard clustering algorithms. Next, using the clustered measurements, we run DHF and form a proposal distribution to be used in the Metropolis-Hastings step within the ASMCMC. We have termed our newly proposed method as the adaptive SMCMC with particle flow based proposal or ASMCMC-DHF. We have thoroughly analyzed the performance of ASMCMC-DHF for the processing of massive sensor data under different settings of algorithm and system parameters. We have noted that our new scheme can handle the effect of increasing dimensionality in a graceful manner. Also, it has been shown that our method not only outperforms the more well established method like the particle filter, but also performs better than its parent algorithm, the ASMCMC.

$\lambda_{\mathbf{x}} = \mathbf{50}$, $\lambda_{\mathbf{c}} = \mathbf{200}$				
Method	**RAMSE**	**Acceptance rate**	**Compression ratio**	**Processing time**
SIR-PF-1000	(1.87 , 2.42 , 3.67)	-	-	(3.21 , 4.31 , 6.87)
SIR-PF-10000	(1.38 , 1.83 , 2.77)	-	-	(40.89 , 57.27 , 62.86)
SIR-PF-25000	(1.36 , 1.73 , 2.32)	-	-	(101.71 , 144.02 , 165.00)
ASMCMC-500	(1.90 , 2.10 , 2.37)	(0.01 , 0.27 , 0.71)	(1.00 , 1.01 , 4.04)	(37.86 , 128.07, 184.48)
ASMCMC-1250	(1.19 , 1.52 , 1.89)	(0.02 , 0.25 , 0.66)	(1.00 , 1.05 , 4.31)	(129.72 , 372.48 , 482.15)
ASMCMC-DHF-125	(1.65 , 2.19 , 3.09)	(0.16 , 0.27 , 0.32)	(1.79 , 1.97 , 2.34)	(33.03 , 42.92 , 55.07)
ASMCMC-DHF-500	(0.80 , 1.24 , 1.67)	(0.15 , 0.24 , 0.31)	(1.91 , 2.52 , 2.74)	(37.36 , 77.14 , 116.12)
ASMCMC-DHF-1250	(0.89 , 1.17 , 1.55)	(0.11 , 0.23 , 0.28)	(1.83 , 2.67 , 2.88)	(40.34 , 125.11 , 212.11)
$\lambda_{\mathbf{x}} = \mathbf{500}$, $\lambda_{\mathbf{c}} = \mathbf{2000}$				
SIR-PF-1000	(1.19 , 1.57 , 2.15)	-	-	(22.31 , 33.86 , 39.11)
SIR-PF-10000	(0.77 , 1.05 , 1.54)	-	-	(253.33 , 324.52 , 355.37)
SIR-PF-25000	(0.59 , 0.82 , 1.26)	-	-	(459.17 , 676.21 , 723.25)
ASMCMC-500	(0.84 , 0.98 , 1.47)	(0.01 , 0.20 , 0.67)	(1.06 ,2.01 , 5.15)	(156.80 , 486.28 , 686.44)
ASMCMC-1250	(0.59 , 0.70 , 0.99)	(0.01 , 0.18 , 0.59)	(1.18 , 2.57 , 4.58)	(367.31 , 1249.43 , 1576.67)
ASMCMC-DHF-125	(0.89 , 1.18 , 1.76)	(0.05 , 0.22 , 0.38)	(2.05 , 2.42 , 2.94)	(41.59 , 74.28 ,112.13)
ASMCMC-DHF-500	(0.34 , 0.54 , 1.13)	(0.14 , 0.21 , 0.25)	(1.83 , 2.69 , 2.89)	(43.47 , 179.37 , 307.93)
ASMCMC-DHF-1250	(0.31 , 0.49 , 0.8)	(0.13 , 0.21 , 0.24)	(1.88 , 2.77 , 2.93)	(63.02 , 411.47 , 751.84)

Table 4.8: Minimum, median and maximum values for RAMSE, Acceptance rate and Compression ratio for different filtering schemes

Appendix

4.A Appendix

For the time index *k*, the likelihood function is given by,

$$l(\mathbf{x}_k) = \frac{e^{-\mu_k}}{M_k!} \prod_{i=1}^{M_k} \left[\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right] \tag{4.26}$$

$$\log l(\mathbf{x}_k) = \log \frac{e^{-\mu_k}}{M_k!} + \sum_{i=1}^{M_k} \log \left[\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right] \tag{4.27}$$

where,

$$\mathbf{x}_{k,j} = [x_{k,j} y_{k,j} \dot{x}_{k,j} \dot{y}_{k,j}]^T$$

$$\nabla \log l(\mathbf{x}_k) = \sum_{i=1}^{M_k} \frac{\lambda_x \nabla p_x^i}{\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum\limits_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j})} \tag{4.28}$$

also,

$$\nabla^2 \log l(\mathbf{x}_k) = \sum_{i=1}^{M_k} \left[\frac{\lambda_x \left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum\limits_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right) \nabla^2 p_x^i - \lambda_x^2 (\nabla p_x^i)(\nabla p_x^i)^T}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum\limits_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right)^2} \right] \tag{4.29}$$

where,

$$\nabla p_x^i = \left[\nabla_{(1)} p_x^{i,1}, \nabla_{(2)} p_x^{i,2}, \cdots, \nabla_{(N_T)} p_x^{i,N_T} \right]^T$$

$$\nabla^2 p_x^i = \begin{bmatrix} \nabla^2_{(1,1)} p_x^{i,1} & \nabla^2_{(1,2)} p_x^{i,1} & \cdots & \nabla^2_{(1,N_T)} p_x^{i,1} \\ \nabla^2_{(2,1)} p_x^{i,2} & \nabla^2_{(2,2)} p_x^{i,2} & \cdots & \nabla^2_{(2,N_T)} p_x^{i,2} \\ \vdots & \vdots & \cdots & \vdots \\ \nabla^2_{(N_T,1)} p_x^{i,N_T} & \nabla^2_{(N_T,2)} p_x^{i,N_T} & \cdots & \nabla^2_{(N_T,N_T)} p_x^{i,N_T} \end{bmatrix}$$

$$\nabla^2 \log l(\mathbf{x}_k) = \sum_{i=1}^{M_k} \left[\frac{\lambda_x \nabla^2 p_x^i}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right)} - \frac{\lambda_x^2 (\nabla p_x^i)(\nabla p_x^i)^T}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) \right)^2} \right] \tag{4.30}$$

$$p_c(\mathbf{z}_k^i) = \mathcal{U}_x(\mu(z_{k,r}, z_{k,\theta})) \mathcal{U}_y(\mu(z_{k,r}, z_{k,\theta})) \tag{4.31}$$

$$p_x(\mathbf{z}_k^i | \mathbf{x}_{k,j}) = p_x^{i,j} = \mathcal{N}(\mathbf{z}_k^i | \mathbf{h}(\mathbf{x}_{k,j}), \mathbf{R}_k) \tag{4.32}$$

$$\mathbf{h}(\mathbf{x}_{k,j}) = \left[\sqrt{x_{k,j} + y_{k,j}}, \tan^{-1} \left(\frac{y_{k,j}}{x_{k,j}} \right) \right]^T \tag{4.33}$$

$$\nabla_{(j)} p_x^{i,j} = \frac{\partial p_x^{i,j}}{\partial \mathbf{x}_{k,j}} = \mathcal{N}(z_k^i | \mathbf{h}(\mathbf{x}_{k,j}), \mathbf{R}_k) \left[\mathbf{H}_{k,j}^T \mathbf{R}_k^{-1} \left(z_k^i - \mathbf{h}(\mathbf{x}_{k,j}) \right) \right] \tag{4.34}$$

$$\begin{aligned} \nabla^2_{(j,j)} p_x^{i,j} = \frac{\partial^2 p_x^{i,j}}{\partial^2 \mathbf{x}_{k,j}} = \mathcal{N}(z_k^i | \mathbf{h}(\mathbf{x}_{k,j}), \mathbf{R}_k) \Big[\mathcal{H}_{k,j} \circledast \mathbf{R}_k^{-1} \left(z_k^i - \mathbf{h}(\mathbf{x}_{k,j}) \right) - \mathbf{H}_{k,j}^T \mathbf{R}_k^{-1} \mathbf{H}_{k,j} \\ + \mathbf{H}_{k,j}^T \mathbf{R}_k^{-1} \left(z_k^i - \mathbf{h}(\mathbf{x}_{k,j}) \right) \left(z_k^i - \mathbf{h}(\mathbf{x}_{k,j}) \right)^T \mathbf{R}_k^{-1} \mathbf{H}_{k,j} \Big] \end{aligned} \tag{4.35}$$

$$\nabla^2_{(j,m)} p_x^{i,j} = \frac{\partial^2 p_x^{i,j}}{\partial \mathbf{x}_{k,j} \partial \mathbf{x}_{k,m}} = \mathbb{O}^{4 \times 4} \tag{4.36}$$

where,

$$\mathbf{H}_{k,j} = \begin{bmatrix} \frac{x_{k,j}}{\sqrt{x_{k,j}+y_{k,j}}} & \frac{y_{k,j}}{\sqrt{x_{k,j}+y_{k,j}}} & 0 & 0 \\ -\frac{y_{k,j}}{\sqrt{x_{k,j}+y_{k,j}}} & \frac{x_{k,j}}{\sqrt{x_{k,j}+y_{k,j}}} & 0 & 0 \end{bmatrix}$$

and $\mathcal{H}_{k,j} \circledast \mathbf{R}_k^{-1} \left(z_k^i - \mathbf{h}(\mathbf{x}_{k,j}) \right)$ refers to the tensor vector product, resulting into a matrix. The four matrices slices of the tensor $\mathcal{H}_{k,j}$ are given by,

$$\mathcal{H}_{k,j}^{(1)} = \begin{bmatrix} \frac{y_{k,j}^2}{(\sqrt{x_{k,j}+y_{k,j}})^3} & \frac{2x_{k,j}y_{k,j}}{(x_{k,j}+y_{k,j})^2} \\ -\frac{x_{k,j}y_{k,j}}{\sqrt{x_{k,j}+y_{k,j}}} & \frac{y_{k,j}^2-x_{k,j}^2}{(x_{k,j}+y_{k,j})^2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{H}_{k,j}^{(2)} = \begin{bmatrix} -\frac{x_{k,j}y_{k,j}}{(\sqrt{x_{k,j}+y_{k,j}})^3} & \frac{y_{k,j}^2-x_{k,j}^2}{(x_{k,j}+y_{k,j})^2} \\ \frac{x_{k,j}^2}{\sqrt{x_{k,j}+y_{k,j}}} & -\frac{2x_{k,j}y_{k,j}}{(x_{k,j}+y_{k,j})^2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$\mathcal{H}_{k,j}^3$ and $\mathcal{H}_{k,j}^4$ are given by $\mathbb{O}^{4\times 2}$.

The hessian of the log-likelihood is given by a block diagonal matrix such that,

$$\boldsymbol{\Xi} = \begin{bmatrix} \Xi_{1,1} & \Xi_{1,2} & \cdots & \Xi_{1,N_T} \\ \Xi_{2,1} & \Xi_{2,2} & \cdots & \Xi_{2,N_T} \\ \vdots & \vdots & \cdots & \vdots \\ \Xi_{N_T,1} & \Xi_{N_T,2} & \cdots & \Xi_{N_T,N_T} \end{bmatrix}$$

The diagonal elements are given by,

$$\Xi_{l,l} = \sum_{i=1}^{M_k} \left[\frac{\lambda_x \nabla_{(l,l)}^2 p_x^{i,l}}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x^{i,j} \right)} - \frac{\lambda_x^2 (\nabla_{(l)} p_x^{i,l})(\nabla_{(l)} p_x^{i,l})^T}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x^{i,j} \right)^2} \right]$$

whereas the off-diagonal elements are given by,

$$\Xi_{l,m} = -\sum_{i=1}^{M_k} \left[\frac{\lambda_x^2 (\nabla_{(l)} p_x^{i,l})(\nabla_{(m)} p_x^{i,m})^T}{\left(\lambda_c p_c(\mathbf{z}_k^i) + \lambda_x \sum_{j=1}^{N_T} p_x^{i,j} \right)^2} \right]$$

Chapter 5

Tensors and the Log homotopy based particle flow

Bayesian estimation is carried out recursively, typically consisting of a prediction and a correction step. As discussed in the previous chapters, the prediction step is usually carried out by solving or approximating the Chapman Kolmogorov equation. In fact, there exist a triad of methods that can be employed to this effect: solving a stochastic differential equation for the samples of the previous posterior density, solving CK integral equation or solving the Fokker-Planck equation (FPE). In the Bayesian estimation context, FPE can be used for the propagation of the posterior density at time step t_k in order to get the prior density at time t_{k+1} for a system with a drift and/or a diffusion component. Analytical solutions to the Fokker-Planck equation are available in only few special cases. Therefore numerical methods have to be employed in order to find an approximate solution. Any such method might face the following set of problems,

- **Positivity and normality**: The FPE describes the time evolution of a probability density function (PDF). Therefore any numerical solution of the FPE has to be non-negative over the whole domain and it should integrate to unity.

- **Domain truncation**: Domain of a PDF is usually infinite. Since the numerical solution has to be formed over a finite region of the state space, the truncation of the domain has to be done judiciously in order to minimize the error incurred thereof.

- **Curse of dimensionality**: The most severe problem faced by any numerical solution for a FPE is the exponential increase in the number of free variables with the increase in the problem dimensionality. This is referred to as the curse of dimensionality. A related issue is that for problems with higher dimensions, the time dimension has to be grated much more finely, particularly at the start of the simulation [MB05].

In this chapter, we employ the FPE to not only predict the probability density into the future times but also to perform the measurement update. We include the measurements using the log-homotopy flow by solving the tensorized FPE in pseudo-time to get the posterior density, hence completing one cycle of the Bayesian recursion. Since the multi-dimensional probability density function, when discretized, leads to a tensor or an n-way array, a tensor based approach

for solving FPE will be used. We elaborate on the tensor based approach and provide results for the regularized optimization problem.

We start by defining the basics of FPE based prediction step and the Bayesian measurement incorporation in section 5.1. The most common numerical methods employed in solving the FPE are described in the section 5.2. Since the FPE solution in higher dimensions entails solving a tensor equation, we discuss the basics of tensors in the section 5.3 while the related concepts are elaborated in the section 5.4. Based on the notations and methodology developed in the two previous sections, we study two problems admitting stationary solutions and derive equations describing the FPE in tensorized format in the section 5.5. We further convert those equations into matrix-vector format and define appropriate constraints. Next, in section 5.6, we provide the details for the tensorization based unified framework for the solution of the FPE, as described by Y.Sun and M.Kumar in their series of paper [SK14], [SK15b], [SK15c] and [SK15d] which is followed by the description of our devised nonlinear filtering algorithm in the section 5.7 termed as the *tensorized filter*. Next, we present the numerical results for the two stationary problems in the section 5.8, together with those for a dynamical problem. This constitutes a nonlinear filtering scenario, which is solved using the *tensorized filter*. Future works make up the section 5.9, which is followed by the conclusion in the section 5.10.

5.1 Continuous-discrete filtering

We consider a general N dimensional nonlinear system with drift $\mathbf{f}(t, \mathbf{x})$. The system is perturbed by white noise defined by a P dimensional Brownian process $\mathbf{B}$(t).

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + \mathbf{g}(\mathbf{x}, t)d\mathbf{B}, \quad \mathbf{x} \in \mathbb{R}^N \tag{5.1}$$

$\mathbf{B}$(t) has zero mean and variance given by $\mathbf{Q}$t. Also, $\mathbf{f}(\mathbf{x}, t) : \mathbb{R}^N \times [0, \infty) \longrightarrow \mathbb{R}^N$. Likewise, $\mathbf{g}(\mathbf{x}, t) : [0, \infty) \times \mathbb{R}^N \longrightarrow \mathbb{R}^{N \times P}$. Let the multivariate function $\mathsf{U}(\mathbf{x}, \mathrm{t})$ define the PDF for the state vector $\mathbf{x}$ at time t. Given the initial PDF $\mathsf{W}_0(\mathbf{x}, \mathrm{t}_0)$, the $\mathsf{U}(\mathbf{x}, \mathrm{t}_k)$ density at any time $t_k \geq t_0$ can be obtained by solving the FPE.

$$\frac{\partial}{\partial t}\mathsf{U}(\mathbf{x}, \mathrm{t}) = \mathcal{L}_{FP}\{\mathsf{U}(\mathbf{x}, \mathrm{t})\} \tag{5.2}$$

where $\mathcal{L}_{FP}$ is called the Fokker-Planck operator (FPO) and is defined as,

$$\mathcal{L}_{FP} = -\sum_{i=1}^{N} \frac{\partial}{\partial x_i} f_i(\mathbf{x}, t) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \partial x_j}(\Upsilon(\mathbf{x}, t))_{i,j}, \tag{5.3}$$

with $\Upsilon(\mathbf{x}, t)$ being the diffusion matrix defined as $\mathbf{g}(\mathbf{x}, \mathbf{t})\mathbf{Q}\mathbf{g}(\mathbf{x}, \mathbf{t})^T$. Hence, the FPE describes the time evolution of the PDF $\mathsf{W}_0(\mathbf{x}, \mathrm{t}_0)$ through the system defined by (5.1). Equation (5.2) can describe the change of PDF between any two time instants t_k and t_{k+1}. Next, we consider a discrete time measurement process defined as,

$$\mathbf{z}_{\mathbf{t_k}} = \psi(\mathbf{x}, t_k) + \nu_{\mathbf{t_k}}, \quad \mathbf{z}_{\mathbf{t_k}} \in \mathbb{R}^M \tag{5.4}$$

where, $\psi(\mathbf{x}_{t_k}, t_k) : \mathbb{R}^N \longrightarrow \mathbb{R}^M$ is called the measurement function and $\nu_{\mathbf{t_k}}$ the measurement noise. Measurements are supposed to be generated at discrete time instances $t_1, t_2, \cdots t_k$.

Let $\mathbf{Z}_{t_k}$ denote the set of measurements up to time t_k, then according to Bayes theorem, the posterior density at time t_k i.e. the density after the inclusion of measurements is given as,

$$\mathsf{U}(\mathbf{x}, \mathrm{t_k}|\mathbf{Z}_{\mathrm{t_k}}) = \frac{\mathsf{p}(\mathbf{z}_{\mathrm{t_k}}|\mathbf{x}, \mathbf{t_k})\mathsf{U}(\mathbf{x}, \mathrm{t_k}|\mathbf{Z}_{\mathrm{t_{k-1}}})}{p(\mathbf{z}_{t_k}|\mathbf{Z}_{t_{k-1}})}, \tag{5.5}$$

where $\mathsf{U}(\mathbf{x}, \mathrm{t_k}|\mathbf{Z}_{\mathrm{t_{k-1}}})$ is prior density at time t_k and $\mathsf{p}(\mathbf{z}_{\mathrm{t_k}}|\mathbf{x}, \mathbf{t_k})$ the measurement likelihood. The conditional density $p(\mathbf{z}_{t_k}|\mathbf{Z}_{t_{k-1}})$ appears as a normalization constant in the measurement update formula, and it describes the distribution of measurement at time t_k, conditioned on the set of all previous measurements. The PDF $\mathsf{U}(\mathbf{x}, \mathrm{t_0})$ is used as the starting point for the first prediction step given by the solution of FPE (5.2), followed by the measurement inclusion step as given in (5.5). $\mathsf{U}(\mathbf{x}, \mathrm{t_1}|\mathbf{Z}_{\mathrm{t_1}})$ becomes the input to the next prediction step and hence the process is continued.

Since the process/time update is carried by solving a continuous time PDE, while the measurement update is done by introducing a discrete time measurement process, the whole approach is termed as the *Continuous Discrete filtering*.

5.2 Numerical methods for the solution of FPE

There have been several numerical methods, proposed in the literature, for solving FPE. These include higher order finite difference method (FDM) and finite element method (FEM) applied to two and four dimensional problems [KN06], homotopic Galerkin approach [Cha06], meshless partition of unity finite element method (PUFEM) by Kumar et al. in [KCJ07]. Recently, a tensor based approach has been introduced by Y.Sun and M.Kumar in the series of papers, [SK14], [SK15a], [SK15b], which will become the basis for the construction of our tensor based nonlinear filtering algorithm. The key concept idea is to express the multidimensional PDF in separable form, which is then tensorized together with the Fokker Planck operator (FPO). Both the sought after PDF and the Fokker Planck operator are expressed in the CANDECOMP-PARAFAC decomposition (CPD) form. Expressing them in CPD form and the subsequent tensorization has been described as instrumental in dealing with the curse of dimensionality. Once done, the individual components are computed using alternating least squares (ALS) algorithm. Details of CP decomposition and the ALS algorithm can be found in [Bro97]. This leads to separation of dimensions, as high dimensional operations are broken down into a series of single dimensional ones, which in turn lead to a linear rather than an exponential rise in the problem complexity. Another key aspect of that framework is the decoupling of the temporal domain from the spatial ones, thereby considering it as an additional dimension. By doing this the benefits of the tensor approach can be used to the fullest. Furthermore, the authors use the Chebyshev spectral method for the domain discretization and differentiation, as opposed to the finite difference method. In this way the occurrence of the so called Runge phenomenon [Epp87] is avoided.

But we before start describing the tensor decomposition based approach to solve FPE and our developed nonlinear filtering algorithm, we will first explain some of the important concepts involved.

5.3 Tensors: An N-dimensional extension of arrays

In many cases, scientific data can be arranged in tables, indicating some sort of dependence or relationship between variables. For two dimensional data, the table is referred to as an array or a matrix. Matrix analysis is a quite well studied branch of mathematics. Tensors, also called *Multidimensional* or *N-way* arrays, are the generalization of matrices to higher dimension [Tuc64]. The use of tensor decomposition first started to appear in experimental fields like chemometrics and psychometrics as early as 1920s. Since then, their use has been proliferated to many other disciplines like signal processing [DD96], computer vision [VT02], neuro sciences [BS05], spectroscopy [JMKB96] etc. Tensor processing appears to have been extensively used in the area of digital communications e.g. the blind source separation DS-CDMA [SD01], Joint detection and estimation of OFDM systems subject to CFO [JS03], Multiple-invariance sensor array processing [SBG00] and Blind coding of linear space-time codes [SB02]. In the current work, the focus is on solving the discretized FPE to get an approximation of the probability density function. Since in higher dimension state spaces, a discretized PDF is represented by an n-way array or a tensor, it becomes imperative to study the basics of tensor analysis. Therefore, the current section and the one following it are devoted to the description of important concepts involved therein.
A rank-1 tensor is a generalization of a rank-1 matrix.

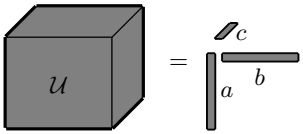


Figure 5.1: A 3D rank-1 tensor

Such a tensor in three dimensions $\mathcal{U} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ can be defined as

$$\mathcal{U} = \mathbf{a} \circledast \mathbf{b} \circledast \mathbf{c} \tag{5.6}$$

where $\circledast$ refers to the outer product between two vectors. An individual element of the tensor $\mathcal{U}$ is defined by the following product:

$$u_{\eta_1,\eta_2,\eta_3} = a_{\eta_1} b_{\eta_2} c_{\eta_3}$$

In N dimensions, a rank-1 tensor $\mathcal{U} \in \mathbb{R}^{P_1 \times P_2 \times \cdots \times P_N}$ is defined as

$$\mathcal{U} = \mathbf{u}_1 \circledast \mathbf{u}_2 \circledast \cdots \circledast \mathbf{u}_N \tag{5.7}$$

The element $u_{\eta_1,\eta_2,\cdots,\eta_N}$ of the tensor is defined as

$$u_{\eta_1,\eta_2,\cdots,\eta_N} = u_{1,\eta_1} u_{2,\eta_2} \cdots u_{N,\eta_N}$$

Generalizing on this, a general tensor in 3 dimensions can be written as a sum of R_U rank-1 tensors,

$$\mathcal{U} = \sum_{i=1}^{R_U} \mathbf{a}_i \circledast \mathbf{b}_i \circledast \mathbf{c}_i \tag{5.8}$$

Vectors $\mathbf{a}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ are referred to as the *loading vectors*. They are also referred to as the *basis vectors*. Stacking these vectors next to each other leads to the so called *factor matrices*. E.g. the first factor matrix of the tensor described in (5.17) is given by, $\mathbf{U}_1 = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{R_U}]$. *Factor matrices* along other dimensions can be described in a similar way. The tensor $\mathcal{U}$ can be represented into a matrix and a vector. These processes are referred to as the matricization and the vectorization of the tensor. For a 3 dimensional tensor, we have the following three matrix unfoldings i.e. tensor data expanded into matrices along the individual dimensions,

$$\begin{aligned}\mathbf{U}^{(1)} &= \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T \\ \mathbf{U}^{(2)} &= \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T \\ \mathbf{U}^{(3)} &= \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\end{aligned}$$

where $\mathbf{A}$, $\mathbf{B}$ and the $\mathbf{C}$ are three factor matrices, and $\odot$ is the Khatri-Rao product between the two matrices and is defined as,

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 | \mathbf{a}_2 \otimes \mathbf{b}_2 | \cdots | \mathbf{a}_N \otimes \mathbf{b}_N] \tag{5.9}$$

Here $\otimes$ refers to the Kronecker product between two matrices, such that

$$\mathbf{a}_1 \otimes \mathbf{b}_1 = [a_1^{(1)}\mathbf{b}_1 \| a_1^{(2)}\mathbf{b}_1 \| \cdots \| a_1^{(P_1)}\mathbf{b}_1] \tag{5.10}$$

where $|$ and $\|$ denote the horizontal and the vertical concatenation, respectively. Also $a_i^{(j)}$ refers to the jth element of the vector $\mathbf{a}_i$. Kronecker product between two matrices $\mathbf{A} \in \mathbb{R}^{P_1 \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times P_2}$ is given by $\mathbf{C} \in \mathbb{R}^{mP_1 \times nP_2}$ and is defined in a similar way such that,

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{n,1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{P_1,1}\mathbf{B} & \cdots & a_{P_1,n}\mathbf{B} \end{bmatrix} \tag{5.11}$$

For the case where the rank-1 tensor represents the data in N-dimensional space, there will be *N* such matrices and vectors that the tensor $\mathcal{U}$ can be folded into. The exact relationships are given below,

$$\mathbf{U}^{(i)} = \mathbf{U}_i \left(\mathbf{U}_N \odot \mathbf{U}_{N-1} \odot \cdots \odot \mathbf{U}_{i+1} \odot \mathbf{U}_{i-1} \odot \cdots \odot \mathbf{U}_1\right)^T \tag{5.12}$$

The following identity can be used for the further unfolding of the matrix into a vector,

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B}) \tag{5.13}$$

5.4 Preliminaries for the Tensor Decomposition based analysis

Owing to their size, tensors in high dimensions are not easy to work with. Hence alternative representations are sought after. One such representation is based on the concept of seperation of the variables. This will become the foundation of the unified framework for the solution of FPE in discretized form, as described later on in this chapter. Therefore, before undertaking that task we would like to explain some key concepts.

5.4.1 Variable separation

Representing a function in dimensions greater than 2 gets increasingly complex in terms of computation. It is because the number discretized points making up the function in a hypercubic space grows geometrically with the increase in the dimension d i.e $N = n_1 \times n_2 \times \cdots \times n_d$. In the context of numerics, this phenomenon, like in the Sequential Monte Carlo, is referred to as the *curse of dimensionality*. Separation of variable representation has been noted as one of the key factor in effectively handling this issue for solving high dimensional discretized problems. A very important contribution in this regard is [BM05], where authors have developed a framework to efficiently represent functions and operator in such scenarios and have applied it to compute the wavefunction for the multiparticle Schrödinger equation with one-dimensional particles and simplified potentials. In our work, we will heavily borrow from [BM05], as it is required to solve the discretized Fokker-Planck equation in higher spaces.
We start with a very basic separated representation for a d dimensional function $u(x_1, x_2, \cdots, x_d) : \mathbb{R}^d \to \mathbb{R}$,

$$u(x_1, x_2, \cdots, x_d) \approx u_1(x_1)u_2(x_2)\cdots u_d(x_d) \tag{5.14}$$

As it can be seen the function is separated into its constituent one dimensional functions. However, this is a very simple expression and in most of the cases it is not sufficient for the accurate approximation. It is natural to extend the expression containing the sum of R_U such terms,

$$\begin{aligned} u(x_1, x_2, \cdots, x_d) &\approx \sum_{j=1}^{R_U} \theta_j u_1^j(x_1)u_2^j(x_2)\cdots u_d^j(x_d) \\ &= \sum_{j=1}^{N}\prod_{i=1}^{d} \theta_j u_i^j(x_i) \end{aligned} \tag{5.15}$$

This expression is more general and depending on the approximation rank R_U, can approximate an arbitrary function much better. By having the function $u(x_1, x_2, \cdots, x_d)$ in the form above, many linear operations in d dimensional spaces decompose into the corresponding operations in one dimensional space. The main task is to keep the approximation rank as low as possible.

5.4.2 Tensor decomposition

Once the function has been approximated in the above form, the next step is to discretize it,

$$u(x_1(\eta_1), x_2(\eta_2), \cdots, x_d(\eta_d)) = \sum_{j=1}^{R_U} \theta_j u_1^j(x_1^{\eta_1})u_2^j(x_2^{\eta_2})\cdots u_d^j(x_d^{\eta_d}) \tag{5.16}$$

where $\{\eta_i\}_{i=1}^d$ are the indices of the data points along the d dimensions. Each dimension is discretized using n_i points. Since each axis is orthogonal, the discretized representation of the function in (5.15) becomes a d-dimensional tensor,

$$\mathcal{U} = \sum_{j=1}^{R_U} \theta_j \mathbf{u}_1^j \circledast \mathbf{u}_2^j \circledast \cdots \circledast \mathbf{u}_d^j \tag{5.17}$$

Discretized functions $\mathbf{u}_d$ become the loading/basis vectors. Therefore, the tensor $\mathcal{U}$ is written as a sum of R_U rank-1 tensor, each composed of the outer products of d loading vectors. In

[BM05], the expression in (5.17) is also termed as the vector in d dimension. Representing a tensor in 2D (array) by a summation of R_U rank-1 bears a special name, the singular value decomposition. For a real matrix $\mathbf{U} \in \mathbb{R}^{P_1 \times P_2}$, $\mathbf{a}_i \in \mathbb{R}^{P_1 \times 1}$ and $\mathbf{b}_i \in \mathbb{R}^{P_2 \times 1}$ can be called the left and right singular values vectors if there exist a non-negative number σ_i such that $\mathbf{U}\mathbf{a}_i = \sigma_i \mathbf{b}_i$ and $\mathbf{U}^T\mathbf{b}_i = \sigma_i \mathbf{a}_i$. In that case the matrix $\mathbf{U}$ may be expressed as,

$$\begin{aligned} \mathbf{U} &= \mathbf{A\Sigma B} \\ &= \sum_{i=1}^{R_U} \sigma_i \tilde{\mathbf{a}}_i \mathbf{b}_i^T = \sum_{i=1}^{R_U} \mathbf{a}_i \mathbf{b}_i^T \end{aligned}$$

where R_U is the decomposition rank. If such decomposition exist, it is referred to as the singular value decomposition (SVD) of the matrix $\mathbf{U}$. The matrices $\mathbf{A}$ and $\mathbf{B}$ then hold the left and right singular vectors of $\mathbf{U}$, which form a set of orthonormal basis functions.

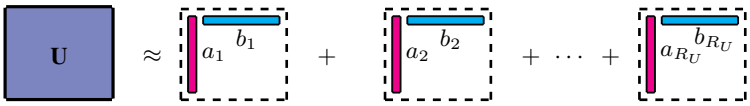


Figure 5.1: A SVD representation for 2D arrays

For a three or a higher dimensional tensor, the similar decomposition is referred to as the Canonical decomposition or the Parallel factor decomposition (CANDECOMP-PARAFAC Decomposition or CPD) and is defined by the equation 5.17. There are some interesting properties of CP decomposition. First, the approximation rank defined as the smallest number of rank-1 tensors that exactly generate the full tensor, can only be known within certain bounds. That is, assuming such a sum actually exist in the first place. Secondly, as opposed to SVD, CPD can be uniquely determined under a much milder set of assumptions, except for the inherent permutation and scaling indeterminacies. Typically, the CPD is computed using the Alternating Least Squares (ALS) algorithm. For a very lucid description of tensor representation, decomposition and related topics, please refer to [KB09]. A typical CPD representation of a three dimensional tensor is shown below,

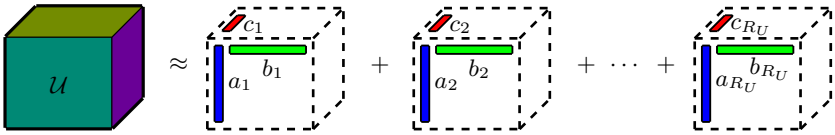


Figure 5.2: A CPD representation for a 3D tensor

5.4.3 Tensorization of a linear operator

A linear operator $\mathbb{A}$ in dimension d can be defined as a linear map $\mathbb{A} : \mathsf{F} \rightarrow \mathsf{F}$, where F being the space of functions in dimension d. A matrix $\mathbb{A}$ in dimension d is a defined to be the discrete representation of a linear operator in dimension d. In the way similar to the representation of

the function in dimension d, the operator $\mathbb{A}$ is first represented with the indices,

$$\mathbb{A} = \sum_{j=1}^{N_A} \mu_j A_1^j(x_1^{\eta_1,\eta_1'}) A_2^j(x_1^{\eta_2,\eta_2'},) \cdots A_d^j(x_d^{\eta_d,\eta_d'},) \tag{5.18}$$

which is further discretized to yield the following representation,

$$\mathbb{A} = \sum_{j=1}^{N_A} \mu_j \mathbf{A}_1^j \circledast \mathbf{A}_2^j \circledast \cdots \circledast \mathbf{A}_d^j \tag{5.19}$$

Next, two important operations are defined. The first is the inner product between two d dimensional vectors (tensors),

$$\langle \mathcal{U}, \mathcal{V} \rangle = \sum_{j_1=1}^{N_{U_1}} \sum_{j_2=1}^{N_{U_2}} \theta_{j_1} \theta_{j_2} \langle \mathbf{u}_1^{j_1} \mathbf{v}_1^{j_2} \rangle \langle \mathbf{u}_2^{j_1} \mathbf{v}_2^{j_2} \rangle \cdots \langle \mathbf{u}_d^{j_1} \mathbf{v}_d^{j_2} \rangle \tag{5.20}$$

Vectors along each dimensions are multiplied such that the result is a scaler. Next, the operation of a tensorized operator on a tensorized function is defined such that,

$$\langle \mathbb{A}, \mathcal{V} \rangle = \sum_{i=1}^{N_A} \sum_{j=1}^{N_U} \mu_i \theta_j \left(\mathbf{A}_1^i \mathbf{v}_1^j \right) \otimes \left(\mathbf{A}_2^i \mathbf{v}_2^j \right) \otimes \cdots \otimes \left(\mathbf{A}_d^i \mathbf{v}_d^j \right) \tag{5.21}$$

It can be seen that the result of this product is another tensor, albeit with increased approximation rank $N_U' = N_A \times N_U$. The important thing to note is the operator has been decomposed into its constituent matrices, each of which operates along its dimension individually. In other words, a multi-dimensional operator has been decomposed into a sequence of single dimensional operators. This is a very major advantage of using the separated representation for functions and operators. This will be put to fullest of use in the description of a unified framework for the solution of FPE as per [SK15b], in the section 5.6.

5.5 An *Ab initio* approach for numerically solving FPE

In the previous section, we described the Separable representation of real valued functions and the tensor decomposition based representation for such functions and linear operators. We can now discuss the solution of the FPE based on these two concepts. We start by studying two cases, each of which admits a stationary solution to the FPE. The main idea is to demonstrate the key steps involved in the derivation of equations describing the tensorized density. The equations for the factor matrices are derived, which are later solved via ALS. This main aim of this section is to provide an insight into the complexities involved in solving for the FPE via *Ab initio* method for rather simple cases. That also becomes the motivation for the search for a more unified framework for numerically solving the Fokker Planck equation.

5.5.1 2 dimensional nonlinear harmonic oscillator

In the first example, we consider a two dimensional harmonic oscillator defined by the following SDE,

$$\ddot{x} + b\dot{x} + x + a(x^2 + \dot{x}^2)\dot{x} = g\eta(t) \tag{5.22}$$

where $\eta(t)$ is the white noise process with variance $\mathbb{E}[\eta^2] = \sigma_\eta^2$. The process in (5.22) admits a stationary solution which is given as,

$$P_s(x, y) = K \exp\left(-\frac{1}{g^2\sigma_{\eta^2}}\left[\frac{a}{2}\left(x^2 + y^2\right)^2 + b\left(x^2 + y^2\right)\right]\right) \tag{5.23}$$

where K is the normalization constant. Now, the above SDE is converted into a system of SDE such that,

$$\begin{aligned} x_2 &= \dot{x}_1 \\ \dot{x}_2 &= -bx_2 - x_1 - a(x_1^2 + x_2^2)y + g\eta(t) \end{aligned} \tag{5.24}$$

which can be expressed in vectorized form,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -bx_2 - x_1 - a(x_1^2 + x_2^2)x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} g\eta(t) \tag{5.25}$$

Since the system described by (5.25) is two dimensional, the solution of the related FPE is derived via singular value decomposition. As will be seen in the next example, the methodology for the solution of a higher dimensional FPE via tensor decomposition follows a very similar approach. Hence, the current example serves as a nice illustration for grasping the key concepts. There are two main parts of the solution: expressing the individual components of the FPE in the separable form and the subsequent discretization of the drift and diffusion terms together with the basis functions along individual axis. Below, we start with the first phase of the derivation. From (5.25), it can readily be seen that the drift terms are already in the separable form,

$$\begin{aligned} f_1(x_1, x_2) &= \prod_{d=1}^{2} f_{1_d}(x_i) = (1)(x_2) \\ f_2(x_1, x_2) &= \sum_{j=1}^{3}\prod_{d=1}^{2} f_{2_d}^j(x_i) = -x_2(b + ax_2^2) - ax_1 - ax_1^2x_2 \\ &= (1)(-x_2(b + ax_2^2)) + (ax_1)(1) + (-ax_1^2)(x_2) \end{aligned} \tag{5.26}$$

while the diffusion matrix is given by,

$$\mathbf{D} = g^2\sigma_\eta^2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.27}$$

Therefore, the FPE describing the evolution of the density for the system described in (5.22) can be expressed as,

$$\begin{aligned} \frac{\partial}{\partial t}\mathsf{U}(x_1, x_2) = &-\frac{\partial}{\partial x_1} f_1(x_1, x_2)\mathsf{U}(x_1, x_2) - \frac{\partial}{\partial x_2} f_2(x_1, x_2)\mathsf{U}(x_1, x_2) \\ &+ \frac{1}{2}\frac{\partial}{\partial x_2^2} g^2\sigma_\eta^2 \mathsf{U}(x_1, x_2) \end{aligned} \tag{5.28}$$

Also, we assume that the density can be likewise expressed in the separable form,

$$\mathsf{U}(x_1, x_2) = \sum_{i=1}^{R_U} u_1^i(x_1) u_2^i(x_2) \tag{5.29}$$

Now, with the drift and diffusion terms of the stochastic process together with the density expressed in separable form, we can proceed to expand the individual terms in FPE (5.28). First, we describe the two drift terms,

$$\begin{aligned} \frac{\partial}{\partial x_1} f_1(x_1, x_2)\mathsf{U}(x_1, x_2) &= \frac{\partial}{\partial x_1}\left[x_2 \sum_{i=1}^{R_U} u_1^i(x_1) u_2^i(x_2)\right] &(5.30)\\ &= x_2 \sum_{i=1}^{R_U} \dot{u}_1^i(x_1) u_2^i(x_2) &(5.31)\end{aligned}$$

and,

$$\begin{aligned} \frac{\partial}{\partial x_2} f_2(x_1, x_2)\mathsf{U}(x_1, x_2) &= \frac{\partial}{\partial x_2}\left[\left(\sum_{j=1}^{3}\prod_{d=1}^{2} f_{2_d}^j(x_i)\right)\left(\sum_{i=1}^{R_U} u_1^i(x_1) u_2^i(x_2)\right)\right] \\ &= \left(\sum_{j=1}^{3} f_{2_1}^j(x_1) f_{2_2}^j(x_2)\right)\left(\sum_{i=1}^{R_U} u_1^i(x_1)\dot{u}_2^i(x_2)\right) + \\ &\quad \left(\sum_{j=1}^{3} f_{2_1}^j(x_1)\dot{f}_{2_2}^j(x_2)\right)\left(\sum_{i=1}^{R_U} u_1^i(x_1) u_2^i(x_2)\right)\end{aligned}$$

The derivatives of the individual terms $\dot{f}_{2_2}^j(x_2)$ are given below,

$$\begin{aligned} \dot{f}_{2_2}^1(x_2) &= \frac{\partial}{\partial x_2}\left(-x_2\left(a + bx_2^2\right)\right) = -a - 3by^2 \\ \dot{f}_{2_2}^2(x_2) &= \frac{\partial}{\partial x_2}(1) = 0 \\ \dot{f}_{2_2}^3(x_2) &= \frac{\partial}{\partial x_2}(-x_2) = 1\end{aligned}$$

The diffusion part can be expressed as,

$$\begin{aligned} \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\partial^2}{\partial x_i x_j} D_{i,j}\mathsf{U}(x_1, x_2) &= \sigma_\eta^2 \frac{\partial^2}{\partial x_2^2}\left(\sum_{i=1}^{R_U} u_1^i(x_1) u_2^i(x_2)\right) \\ &= g^2\sigma_\eta^2 \sum_{i=1}^{R_U} u_1^i(x_1)\ddot{u}_2^i(x_2)\end{aligned}$$

putting every thing together, the Fokker-Planck equation can be written as,

$$\begin{aligned}\frac{\partial}{\partial t}\sum_{i=1}^{R_U} u_1^i(x_1)u_2^i(x_2) &= -x_2\sum_{i=1}^{R_U}\dot{u}_1^i(x_1)u_2^i(x_2)\\ &\quad -\left[\sum_{j=1}^{3} f_{2_1}^j(x_1)f_{2_2}^j(x_2)\right]\left[\sum_{i=1}^{R_U} u_1^i(x_1)\dot{u}_2^i(x_2)\right]\\ &\quad -\left[(-a-3bx_2^2)+ax_2^2\right]\left[\sum_{i=1}^{R_U} u_1^i(x_1)u_2^i(x_2)\right]\\ &\quad +g^2\sigma_\eta^2\sum_{i=1}^{R_U} u_1^i(x_1)\ddot{u}_2^i(x_2) \end{aligned} \tag{5.32}$$

Now, we start with the second part of the procedure i.e. discretization of the individual terms. Let $\mathbf{x}_1 = \{x_1^p\}_{p=1}^{P_1}$ and $\mathbf{x}_2 = \{x_2^q\}_{q=1}^{P_2}$ be the vectors of P discretized points along x_1 and x_2 axis. Thus the partial differential operators can be expressed using the finite difference / Chebyshev differentiation matrices such that $\frac{\partial}{\partial x_i} \longmapsto \mathbf{D}_{x_i}$ and likewise $\frac{\partial}{\partial x_i^2} \longmapsto \mathbf{D}_{x_i}^2$. Likewise, the *ith* basis vectors evaluated at the points set $\mathbf{x}_1$ and $\mathbf{x}_2$ be vectorized, yielding the basis/loading vectors $\mathbf{u}_1^i$, $\mathbf{u}_2^i$. Overall, discretization of all such vector yield the factor matrices $\mathbf{U}_1$ and $\mathbf{U}_2$. Also, let $\mathbf{G}_1 \in \mathbb{R}^{P_1\times 3}$ and $\mathbf{G}_2 \in \mathbb{R}^{P_2\times 3}$ represent the matricized drift components $\mathbf{f}_{2_1}$ and $\mathbf{f}_{2_2}$, such that *ith* matrix row corresponds to the particular function values evaluated at points $x_{1,i}$ and $x_{2,i}$ respectively. Given all this, the discretized drift part of the FPE can be expressed as,

$$\begin{aligned} D_{p,q}^{(1)} &= \frac{\partial}{\partial x_1} f_1(x_1,x_2)\mathsf{U}(x_1,x_2)\Big|_{\substack{x_1=x_{1,p}\\ x_2=x_{2,q}}} = x_{2,q}\sum_{i=1}^{R_U}\frac{\partial}{\partial x_1}u_1^i(x_{1,q})u_2^i(x_{2,p})\\ &= x_{2,q}\sum_{i=1}^{R_U} u_1^i(x_{1,p})\left(\mathbf{D}_{x_1}\mathbf{U}_1\right)_{p,i} \end{aligned} \tag{5.33}$$

and,

$$\begin{aligned} D_{p,q}^{(2)} &= \frac{\partial}{\partial x_1} f_2(x_1,x_2)\mathsf{U}(x_1,x_2)\Big|_{\substack{x_1=x_{1,p}\\ x_2=x_{2,q}}}\\ &= \left(\sum_{j=1}^{3} f_{2_1}^j(x_{1,p})f_{2_2}^j(x_{2,q})\right)\left(\sum_{i=1}^{R_U} u_1^i(x_{1,p})\frac{\partial}{\partial x_2}u_{2,q}^i(x_{2,q})\right)\\ &+ \left(\sum_{j=1}^{3} f_{2_1}^j(x_{1,p})\frac{\partial}{\partial x_2}f_{2_2}^j(x_{2,q})\right)\left(\sum_{i=1}^{R_U} u_1^i(x_{1,p})u_2^i(x_{2,q})\right)\\ &= \left(\sum_{j=1}^{3} f_{2_1}^j(x_{1,p})f_{2_2}^j(x_{2,q})\right)\left(\sum_{i=1}^{R_U} u_1^i(x_{1,p})\left(\mathbf{D}_{x_2}\mathbf{U}_2\right)_{q,i}\right)\\ &+ \left(\sum_{j=1}^{3} f_{2_1}^j(x_{1,p})\left(\mathbf{D}_{x_2}\mathbf{G}_2\right)_{q,i}\right)\left(\sum_{i=1}^{R_U} u_1^i(x_{1,p})u_2^i(x_{2,q})\right) \end{aligned} \tag{5.34}$$

Similarly, the diffusion term can be expressed as,

$$\begin{aligned} D^{(3)}_{p,q} &= g^2\sigma_\eta^2 \frac{\partial^2}{\partial x_2^2}\mathsf{U}(x_1,x_2)\Big|_{\substack{x_1=x_{1,p}\\x_2=x_{2,q}}} = \sum_{i=1}^{R_U} u_1^i(x_{1,q})\frac{\partial^2}{\partial x_1^2}u_2^i(x_{2,p}) \\ &= x_{2,q}\sum_{i=1}^{R_U} u_1^i(x_{1,p})\left(\mathbf{D}^2_{x_2}\mathbf{U}_2\right)_{p,q} \end{aligned} \tag{5.35}$$

In the full matrix form, (5.35) and (5.34) can be given as,

$$\begin{aligned} \mathbf{D}^{(1)} &= \left(\mathbf{D}_{x_1}\mathbf{U}_1\right)\left(\mathrm{Diag}(\mathbf{x}_2)\mathbf{U}_2\right)^T \\ &= \mathbf{D}_{x_1}\mathbf{U}_1\mathbf{U}_2^T\mathrm{Diag}(\mathbf{x}_2) \end{aligned} \tag{5.36}$$

$$\begin{aligned} \mathbf{D}^{(2)} &= \left(\mathbf{G}_1\mathbf{G}_2^T\right)\circ\left(\mathbf{U}_1\left(\mathbf{D}_{x_2}\mathbf{U}_2\right)^T\right) + \left(\mathbf{G}_1\left(\mathbf{D}_{x_2}\mathbf{G}_2\right)^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\right) \\ &= \left(\mathbf{G}_1\mathbf{G}_2^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\mathbf{D}_{x_2}^T\right) + \left(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\right) \end{aligned} \tag{5.37}$$

$$\mathbf{D}^{(3)} = g^2\sigma_\eta^2\mathbf{U}_1\mathbf{U}_2^T\left(\mathbf{D}^2_{x_2}\right)^T \tag{5.38}$$

where $\circ$ represents the Hadamard or pair-wise product between the two matrices and $\mathrm{Diag}(\mathbf{x}_2)$ refers to the diagonal matrix with $\mathbf{x}_2$ forming the diagonal. When looking for the stationary solution of the FPE, it is assumed that the probability current has reached zero. Hence, the L.H.S. of the (5.32) can be put to zero. This lead to the following matricized form of the FPE,

$$\frac{1}{2}\mathbf{D}^{(3)} = \mathbf{D}^{(1)} + \mathbf{D}^{(2)} \tag{5.39}$$

which leads to,

$$\begin{aligned} \frac{1}{2}g^2\sigma_\eta^2\mathbf{U}_1\mathbf{U}_2^T\left(\mathbf{D}^2_{x_2}\right)^T &= \mathbf{D}_{x_1}\mathbf{U}_1\mathbf{U}_2^T\mathrm{Diag}(\mathbf{x}_2) \\ &+ \left(\mathbf{G}_1\mathbf{G}_2^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\mathbf{D}_{x_2}^T\right) \\ &+ \left(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\right) \end{aligned} \tag{5.40}$$

The task now becomes to solve for the unknown factor matrices $\mathbf{U}_1$ and $\mathbf{U}_2$. It is done here in two steps. First, with the help of following vector identities, we convert (5.40) into a vector equation.

$$\begin{aligned} \mathrm{vec}(\mathbf{ABC}) &= (\mathbf{C}^T\otimes\mathbf{A})\mathrm{vec}(\mathbf{B}) \\ \mathrm{vec}(\mathbf{AB}) &= (\mathbf{I}^T\otimes\mathbf{A})\mathrm{vec}(\mathbf{B}) \\ \mathrm{vec}(\mathbf{A}\circ\mathbf{B}) &= \mathrm{vec}(\mathbf{A})\circ\mathrm{vec}(\mathbf{B}) \end{aligned} \tag{5.41}$$

Applying these identities, the equation (5.40) can be transformed into a in the following way,

$$\begin{aligned} \frac{1}{2}g^2\sigma_\eta^2\mathrm{vec}(\mathbf{U}_1\mathbf{U}_2^T\left(\mathbf{D}^2_{x_2}\right)^T) &= \mathrm{vec}(\mathbf{D}_{x_1}\mathbf{U}_1\mathbf{U}_2^T\mathrm{Diag}(\mathbf{x}_2)) \\ &+ \mathrm{vec}\left(\left(\mathbf{G}_1\mathbf{G}_2^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\mathbf{D}_{x_2}^T\right)\right) \\ &+ \mathrm{vec}\left(\left(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T\right)\circ\left(\mathbf{U}_1\mathbf{U}_2^T\right)\right) \end{aligned} \tag{5.42}$$

Solving for $\mathbf{U}_1$, one gets,

$$\begin{aligned}\frac{1}{2}g^2\sigma_\eta^2((\mathbf{D}_{x_2}^2\mathbf{U}_2)\otimes\mathbf{I})\text{vec}(\mathbf{U}_1) &= ((\text{Diag}(\mathbf{x}_1)\mathbf{U}_2)\otimes\mathbf{D}_{x_1})\,\text{vec}(\mathbf{U}_1)\\ &+ \text{vec}(\mathbf{G}_1\mathbf{G}_2^T)\circ((\mathbf{D}_{x_2}\mathbf{U}_2)\otimes\mathbf{I})\,\text{vec}(\mathbf{U}_1)\\ &+ \text{vec}(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T)\circ(\mathbf{U}_2\otimes\mathbf{I})\,\text{vec}(\mathbf{U}_1) \qquad (5.43)\end{aligned}$$

which can be further simplified as,

$$\begin{aligned}\big[\tfrac{1}{2}g^2\sigma_\eta^2(\mathbf{D}_{x_2}^2\mathbf{U}_2)\otimes\mathbf{I} - (\text{Diag}(\mathbf{x}_1)\mathbf{U}_2)\otimes\mathbf{D}_{x_1} - \text{vec}(\mathbf{G}_1\mathbf{G}_2^T)\circ((\mathbf{D}_{x_2}\mathbf{U}_2)\otimes\mathbf{I})\\ + \text{vec}(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T)\circ(\mathbf{U}_2\otimes\mathbf{I})\big]\,\text{vec}(\mathbf{U}_1) = \mathbf{0}\\ \mathbf{A}_{u_1}\text{vec}(\mathbf{U}_1) = \mathbf{0} \qquad (5.44)\end{aligned}$$

The vector equation for $\mathbf{U}_2$ can be derived in the similar way

$$\begin{aligned}\big[\tfrac{1}{2}g^2\sigma_\eta^2(\mathbf{D}_{x_2}^2\otimes\mathbf{U}_1) - \text{Diag}(\mathbf{x}_1)^T\otimes(\mathbf{D}_{x_1}\mathbf{U}_1) - \text{vec}(\mathbf{G}_1\mathbf{G}_2^T)\circ(\mathbf{D}_{x_2}\otimes\mathbf{U}_1)\\ + \text{vec}(\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T)\circ(\mathbf{I}\otimes\mathbf{U}_1)\big]\,\text{vec}(\mathbf{U}_2^T) = \mathbf{0}\\ \mathbf{A}_{u_2}\text{vec}(\mathbf{U}_2^T) = \mathbf{0} \qquad (5.45)\end{aligned}$$

where $\mathbf{A}_{u_1}$ and $\mathbf{A}_{u_2}$ are represent the two modes of the matricized FPO. The second step in getting the factor matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ is to recursively solve (5.44) and (5.45). Both matrices are initialized with random values and iterations are run until the convergence. One last thing is the use of appropriate boundary value and normality constraints. The first is enforced by simply removing the outer most rows and column of the matrices $\mathbf{A}_{u_1}$ and $\mathbf{A}_{u_2}$ and of the two loading matrices $\mathbf{U}_1$ and $\mathbf{U}_2$. The normalization constraint is imposed in order to have

$$\int_\Omega \mathbf{U}(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = 1 \qquad (5.46)$$

Due to the separable form for the PDF, the double integral is decoupled into two one dimensional integrals. The integrals are numerically approximated through the some quadrature e.g. the Clenshaw-Curtis quadrature $\mathbf{w}_C$ such that (5.46) is approximated as $\mathbf{w}_C^T\mathbf{U}_1\mathbf{U}_2^T\mathbf{w}_C \approx 1$. Therefore, the constrained ALS equations are given by,

$$\begin{bmatrix}\mathbf{A}_{u_1}\\ (\mathbf{w}_C^T\mathbf{U}_2)\otimes\mathbf{w}_C\end{bmatrix}\text{vec}(\mathbf{U}_1) = \begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix} \qquad (5.47)$$

and for the second dimension we have,

$$\begin{bmatrix}\mathbf{A}_{u_2}\\ \mathbf{w}_C^T\otimes(\mathbf{w}_C^T\mathbf{U}_1)\end{bmatrix}\text{vec}(\mathbf{U}_2) = \begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix} \qquad (5.48)$$

The simulation results for this example will be mentioned in the section 5.8.

5.5.2 P dimensional linear harmonic oscillator

We further demonstrate the efficacy of the use of separable representation and the tensor decomposition for solving FPE by considering a generalized linear oscillator. Here, again the objective is to derive equations to be solved for the factor matrices via ALS. The following 2P dimensional, linear and separable system is considered,

$$\begin{aligned} \dot{x}_{2i-1} &= x_{2i} \\ \dot{x}_{2i} &= -a_{2i}x_{2i} - a_{2i-1}\frac{\partial V}{\partial x_{2i-1}} + \xi_i, \qquad i = 1, \cdots, P \end{aligned} \tag{5.49}$$

where $V(x_1, x_3, \cdots, x_{2P-1}) = \frac{1}{2}\sum_{i=1}^{P} x_{2i-1}^2$ and ξ_i is uncorrelated white noise such that $\mathbb{E}[\xi_i, \xi_j] = \sigma_{i,j}^2 = 0$ and $\mathbb{E}[\xi_i^2] = \sigma_{i,i}^2 = 2a_{2i-1}a_{2i}T$. Given this, the stochastic process in the (5.49) admits the following stationary solution,

$$P_s(x_1, x_2, \cdots, x_{2P}) = \prod_{i=1}^{P}\left(\frac{1}{16\pi^2T^2\sqrt{a_{2i-1}}}\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{4T}\sum_{i=1}^{P}\left(x_{2i-1}^2 + \frac{x_{2i}^2}{a_{2i-1}}\right)\right] \tag{5.50}$$

For simplicity, we restrict the system dimensionality to four i.e. $P = 2$. The resulting system can be written in the following form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -a_1 & -a_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -a_3 & -a_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} \tag{5.51}$$

Once again, we assume a separable form for the stationary density,

$$\mathsf{U}(x_1, y_1, x_2, y_2) = \sum_{l=1}^{R_U} u_1^l(x_1)u_2^l(x_2)u_3^l(x_3)u_4^l(x_4) \tag{5.52}$$

Next, we express the drift and diffusion components individually. The drift components can

already be seen to be in a separable form,

$$
\begin{aligned}
D^{(1)} &= \frac{\partial}{\partial x_1} f_1(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) = x_2 \sum_{l=1}^{R_U} \dot{u}_1^l(x_1) u_2^l(x_2) u_3^l(x_3) u_4^l(x_4) \\
D^{(2)} &= \frac{\partial}{\partial x_2} f_2(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) \\
&= (-a_1 x_1 - a_2 x_2) \sum_{l=1}^{R_U} u_1^l(x_1) \dot{u}_2^l(x_2) u_3^l(x_3) u_4^l(x_4) \\
&- a_2 \sum_{l=1}^{R_U} u_1^l(x_1) u_2^l(x_2) u_3^l(x_3) u_4^l(x_4) \\
D^{(3)} &= \frac{\partial}{\partial x_3} f_3(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) = x_4 \sum_{l=1}^{R_U} u_1^l(x_1) u_2^l(y_2) \dot{u}_3^l(x_3) u_4^l(x_4) \\
D^{(4)} &= \frac{\partial}{\partial x_2} f_4(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) \\
&= (-a_3 x_3 - a_4 x_4) \sum_{l=1}^{R_U} u_1^l(x_1) u_2^l(x_2) u_3^l(x_3) \dot{u}_4^l(x_4) \\
&- a_4 \sum_{l=1}^{R_U} u_1^l(x_1) u_2^l(x_2) u_3^l(x_3) u_4^l(x_4)
\end{aligned}
\tag{5.53}
$$

Similarly, the diffusion terms are given as,

$$
\begin{aligned}
D^{(5)} &= \frac{\partial}{\partial x_1} f_1(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) = \gamma_1 \sum_{l=1}^{R_U} u_1^l(x_1) \ddot{u}_2^l(y_2) u_3^l(x_3) u_4^l(x_4) \\
D^{(6)} &= \frac{\partial}{\partial x_3} f_3(x_1, y_1, x_2, y_2)\mathsf{U}(x_1, y_1, x_2, y_2) = \gamma_2 \sum_{l=1}^{R_U} u_1^l(x_1) u_2^l(y_2) u_3^l(x_3) \ddot{u}_4^l(x_4)
\end{aligned}
\tag{5.54}
$$

where $\gamma_1 = 2a_1a_2T$ and $\gamma_2 = 2a_3a_4T$, respectively. Next, we consider the discretization of the dimensions, such that each dimension is discretized with P_i points i.e. $\mathbf{x}_i = \{x_i^p\}_{p=1}^{P_i}$. Again, we use the notation of $\mathbf{U}_d$ to represent the factor matrix along the *dth* dimension. Also the discretized differential operator along the same dimension will be represented as $\mathbf{D}_{x_d}$. Given

this, equations 5.53 and 5.54 can be expressed as,

$$
\begin{aligned}
D^{(1)}_{p_1,p_2,p_3,p_4} &= x_{2,p_2}\sum_{l=1}^{R_U} u^l_2(x_{2,p_2})u^l_3(x_{3,p_3})u^l_4(x_{4,p_4})(\mathbf{D}_{x_1}\mathbf{U}_1)_{p_1,i} \\
D^{(2)}_{p_1,p_2,p_3,p_4} &= \sum_{j=1}^{2} f^j_{2_1}(x_{1,p_1})f^j_{2_2}(x_{1,p_2})\sum_{l=1}^{R_U} u^l_{1,p_1}(x_1)u^l_{3,p_3}(x_3)u^l_{4,p_4}(x_4)(\mathbf{D}_{x_2}\mathbf{U}_2)_{p_2,i} \\
&+ \sum_{j=1}^{2} f^j_{2_1}(x_{1,p_1})(\mathbf{D}_{x_2}f^j_{2_2}(x_{1,p_2}))_{p_2,j}\sum_{l=1}^{R_U} u^l_{1,p_1}(x_1)u^l_{2,p_2}(x_2)u^l_{3,p_3}(x_3)u^l_{4,p_4}(x_4) \\
D^{(3)}_{p_1,p_2,p_3,p_4} &= x_{4,p_4}\sum_{l=1}^{R_U} u^l_1(x_{1,p_1})u^l_2(x_{2,p_2})u^l_4(x_{4,p_4})(\mathbf{D}_{x_3}\mathbf{U}_3)_{p_3,i} \\
D^{(4)}_{p_1,p_2,p_3,p_4} &= \sum_{j=1}^{2} f^j_{4_1}(x_{1,p_3})f^j_{4_2}(x_{1,p_4})\sum_{l=1}^{R_U} u^l_{1,p_1}(x_1)u^l_{2,p_2}(x_2)u^l_{3,p_3}(x_3)(\mathbf{D}_{x_4}\mathbf{U}_4)_{p_4,i} \\
&+ \sum_{j=1}^{2} f^j_{3_1}(x_{3,p_3})(\mathbf{D}_{x_4}f^j_{4_2}(x_{1,p_4}))_{p_4,j}\sum_{l=1}^{R_U} u^l_{1,p_1}(x_1)u^l_{2,p_2}(x_2)u^l_{3,p_3}(x_3)u^l_{4,p_4}(x_4) \\
D^{(5)}_{p_1,p_2,p_3,p_4} &= \gamma_1\sum_{l=1}^{R_U} u^l_1(x_{1,p_1})u^l_3(x_{3,p_3})u^l_4(x_{4,p_4})(\mathbf{D}^2_{x_2}\mathbf{U}_2)_{p_2,j} \\
D^{(6)}_{p_1,p_2,p_3,p_4} &= \gamma_2\sum_{l=1}^{R_U} u^l_1(x_{1,p_1})u^l_2(x_{4,p_2})u^l_3(x_{3,p_3})(\mathbf{D}^2_{x_4}\mathbf{U}_4)_{p_2,j}
\end{aligned}
\tag{5.55}
$$

Tensorizing the above series of equations leads to,

$$
\begin{aligned}
\mathcal{D}^{(1)} &= \sum_{l=1}^{R_U}(\mathbf{D}_{x_1}\mathbf{U}_1)_l \otimes (\mathrm{Diag}(\mathbf{x}_2)\mathbf{U}_2)_l \otimes (\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \\
\mathcal{D}^{(2)} &= \sum_{l=1}^{R_U}\left([(\mathbf{U}_1)_l \otimes (\mathbf{D}_{x_2}\mathbf{U}_2)_l] \circ \mathbf{G}_1\mathbf{G}_2^T\right) \otimes (\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \\
&+ \sum_{l=1}^{R_U}\left([(\mathbf{U}_1)_l \otimes (\mathbf{U}_2)_l] \circ \left[\mathbf{G}_1\mathbf{G}_2^T\mathbf{D}_{x_2}^T\right]\right) \otimes (\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \\
\mathcal{D}^{(3)} &= \sum_{l=1}^{R_U}(\mathbf{U}_1)_l \otimes (\mathbf{U}_2)_l \otimes (\mathbf{D}_{x_3}\mathbf{U}_3)_l \otimes (\mathrm{Diag}(\mathbf{x}_4)\mathbf{U}_4)_l \\
\mathcal{D}^{(4)} &= \sum_{l=1}^{R_U}(\mathbf{U}_1)_l \otimes (\mathbf{U}_2)_l \otimes \left([(\mathbf{U}_3)_l \otimes (\mathbf{D}_{x_4}\mathbf{U}_4)_l] \circ \mathbf{G}_3\mathbf{G}_4^T\right) \\
&+ \sum_{l=1}^{R_U}(\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \otimes \left([(\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l] \circ \left[\mathbf{G}_3\mathbf{G}_4^T\mathbf{D}_{x_4}^T\right]\right)
\end{aligned}
\tag{5.56}
$$

and,

$$\begin{aligned}\mathcal{D}^{(5)} &= \gamma_1 \sum_{l=1}^{R_U} (\mathbf{U}_1)_l \otimes (\mathbf{D}_{x_2}^2 \mathbf{U}_2)_l \otimes (\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \\ \mathcal{D}^{(6)} &= \gamma_2 \sum_{l=1}^{R_U} (\mathbf{U}_3)_l \otimes (\mathbf{U}_4)_l \otimes (\mathbf{U}_3)_l \otimes (\mathbf{D}_{x_4}^2 \mathbf{U}_4)_l \end{aligned} \tag{5.57}$$

Here $(\mathbf{U}_i)_l$ refers to the *lth* column of the corresponding factor matrix, while $\mathbf{G}_i$ refers to the matricized drift elements along the *ith* dimension. Similar to the last example, we look for the stationary solution of the Fokker Planck equation. In the tensor notation, this can be expressed as below,

$$\mathcal{D}^1 + \mathcal{D}^2 + \mathcal{D}^3 + \mathcal{D}^4 - \frac{1}{2}\mathcal{D}^5 - \frac{1}{2}\mathcal{D}^6 = 0 \tag{5.58}$$

The next task is to find expressions for the four unknown factor matrices. This is done by matricization and the subsequent vectorization of the tensor equation 5.58, along the different four modes. Lets define the following matrices,

$$\begin{aligned} \Lambda_1^1 &= \text{vec}(\mathbb{I})^T \otimes (\mathbf{G}_1 \mathbf{G}_2^T) & \Lambda_2^1 &= \text{vec}(\mathbb{I})^T \otimes (\mathbf{G}_1 \mathbf{D}_{x_2}^T \mathbf{G}_2^T) \\ \Pi_1^1 &= \text{vec}(\mathbf{G}_1 \mathbf{G}_2^T)^T \otimes \mathbb{I} & \Pi_2^1 &= \text{vec}(\mathbf{G}_1 \mathbf{D}_{x_2}^T \mathbf{G}_2^T)^T \otimes \mathbb{I} \\ \Lambda_1^2 &= \text{vec}(\mathbb{I})^T \otimes (\mathbf{G}_1 \mathbf{G}_2^T)^T & \Lambda_2^2 &= \text{vec}(\mathbb{I})^T \otimes (\mathbf{G}_1 \mathbf{D}_{x_2}^T \mathbf{G}_2^T)^T \\ \Pi_1^2 &= \text{vec}(\mathbf{G}_1 \mathbf{G}_2^T)^T \otimes \mathbb{I}^T & \Pi_2^2 &= \text{vec}(\mathbf{G}_1 \mathbf{D}_{x_2}^T \mathbf{G}_2^T)^T \otimes \mathbb{I}^T \\ \Lambda_1^3 &= \mathbb{I} \otimes \text{vec}(\mathbf{G}_3 \mathbf{G}_4^T)^T & \Lambda_2^3 &= \mathbb{I} \otimes \text{vec}(\mathbf{G}_3 \mathbf{D}_{x_4}^T \mathbf{G}_4^T)^T \\ \Pi_1^3 &= (\mathbf{G}_3 \mathbf{G}_4^T) \otimes \text{vec}(\mathbb{I})^T & \Pi_2^3 &= (\mathbf{G}_3 \mathbf{D}_{x_4}^T \mathbf{G}_4^T) \otimes \text{vec}(\mathbb{I})^T \\ \Lambda_1^4 &= (\mathbb{I})^T \otimes \text{vec}(\mathbf{G}_3 \mathbf{G}_4^T)^T & \Lambda_2^4 &= (\mathbb{I})^T \otimes \text{vec}(\mathbf{G}_3 \mathbf{D}_{x_4}^T \mathbf{G}_4^T)^T \\ \Pi_1^4 &= (\mathbf{G}_3 \mathbf{G}_4^T)^T \otimes \text{vec}(\mathbb{I})^T & \Pi_2^4 &= (\mathbf{G}_3 \mathbf{D}_{x_4}^T \mathbf{G}_4^T)^T \otimes \text{vec}(\mathbb{I})^T \end{aligned}$$

where $\mathbb{I}$ is the matrix containing all ones, of appropriate dimensions. Using (5.12) and (5.41), the matricizing and vectorization along the individual dimensions are given by,

Mode 1 unfolding

$$\begin{aligned} &\mathbf{D}_{x_1}\mathbf{U}_1[\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{U}_2 \text{Diag}(\mathbf{x}_2))]^T + \mathbf{U}_1[(\mathbf{U}_4 \text{Diag}(\mathbf{x}_4)) \odot (\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_2]^T + \\ &\Lambda_1^1 \circ [\mathbf{U}_1(\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}_{x_2}\mathbf{U}_2))^T] + \Lambda_2^1 \circ [\mathbf{U}_1(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2)^T] + \\ &\Pi_1^1 \circ [\mathbf{U}_1((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_2)^T] + \Pi_2^1 \circ [\mathbf{U}_1(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2)^T] - \\ &\tfrac{1}{2}\gamma_1 \mathbf{U}_1(\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}_{x_2}^2\mathbf{U}_2))^T - \tfrac{1}{2}\gamma_2 \mathbf{U}_1((\mathbf{D}_{x_4}^2\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_2)^T = 0 \end{aligned}$$

$$\Big([\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{U}_2\text{Diag}(\mathbf{x}_2))] \otimes \mathbf{D}_{x_1} + [(\mathbf{U}_4\text{Diag}(\mathbf{x}_4)) \odot (\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_2] \otimes \mathbf{I} +$$
$$\text{Diag}(\text{vec}(\Lambda_1^1))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}_{x_2}\mathbf{U}_2)) \otimes \mathbf{I}] + \text{Diag}(\text{vec}(\Lambda_2^1))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}] +$$
$$\text{Diag}(\text{vec}(\Pi_1^1))[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}] + \text{Diag}(\text{vec}(\Pi_2^1))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}] -$$
$$\tfrac{1}{2}\gamma_1[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}^2_{x_2}\mathbf{U}_2)) \otimes \mathbf{I}] -$$
$$\tfrac{1}{2}\gamma_2[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}]\Big)\text{vec}(\mathbf{U}_1) = \mathbf{0} \tag{5.59}$$

Mode 2 unfolding

$$\text{Diag}(\mathbf{x}_2)\mathbf{U}_2[\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}_{x_1}\mathbf{U}_1)]^T + \mathbf{U}_2[(\mathbf{U}_4\text{Diag}(\mathbf{x}_4)) \odot (\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_1]^T +$$
$$\Lambda_1^2 \circ [\mathbf{D}_{x_2}\mathbf{U}_2(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T] + \Lambda_2^2 \circ [\mathbf{U}_2(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T] +$$
$$\Pi_1^2 \circ [\mathbf{U}_2((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T] + \Pi_2^2 \circ [\mathbf{U}_2(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T] -$$
$$\tfrac{1}{2}\gamma_1\mathbf{D}^2_{x_2}\mathbf{U}_2(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T - \tfrac{1}{2}\gamma_2\mathbf{U}_2((\mathbf{D}^2_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_1)^T = 0$$

$$\Big([\mathbf{U}_4 \odot \mathbf{U}_3 \odot (\mathbf{D}_{x_1}\mathbf{U}_1)] \otimes \text{Diag}(\mathbf{x}_2) + [(\mathbf{U}_4\text{Diag}(\mathbf{x}_4)) \odot (\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_1] \otimes \mathbf{I} +$$
$$\text{Diag}(\text{vec}(\Lambda_1^2))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{D}_{x_2}] + \text{Diag}(\text{vec}(\Lambda_2^2))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{I}] +$$
$$\text{Diag}(\text{vec}(\Pi_1^2))[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{I}] + \text{Diag}(\text{vec}(\Pi_2^2))[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{I}] -$$
$$\tfrac{1}{2}\gamma_1[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{D}^2_{x_2}] -$$
$$\tfrac{1}{2}\gamma_2[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}]\Big)\text{vec}(\mathbf{U}_2) = \mathbf{0} \tag{5.60}$$

Mode 3 unfolding

$$\mathbf{U}_3[\mathbf{U}_4 \odot (\mathbf{U}_2\text{Diag}(\mathbf{x}_2)) \odot (\mathbf{D}_{x_1}\mathbf{U}_1)]^T + \mathbf{D}_{x_3}\mathbf{U}_3[(\mathbf{U}_4\text{Diag}(\mathbf{x}_4)) \odot \mathbf{U}_2 \odot \mathbf{U}_1]^T +$$
$$[\mathbf{U}_3(\mathbf{U}_4 \odot (\mathbf{D}_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1)^T] \circ \Lambda_1^3 + [\mathbf{U}_3(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Lambda_2^3 +$$
$$[\mathbf{U}_3((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Pi_1^3 + [\mathbf{U}_3(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Pi_2^3 -$$
$$\tfrac{1}{2}\gamma_1\mathbf{U}_3(\mathbf{U}_4 \odot (\mathbf{D}^2_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1)^T - \tfrac{1}{2}\gamma_2\mathbf{U}_3((\mathbf{D}^2_{x_4}\mathbf{U}_4) \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T = 0$$

$$\Big([\mathbf{U}_4 \odot (\mathbf{U}_2\text{Diag}(\mathbf{x}_2)) \odot (\mathbf{D}_{x_1}\mathbf{U}_1)] \otimes \mathbf{I} + [(\mathbf{U}_4\text{Diag}(\mathbf{x}_4)) \odot \mathbf{U}_2 \odot \mathbf{U}_1] \otimes \mathbf{D}_{x_3} +$$
$$[(\mathbf{U}_4 \odot (\mathbf{D}_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Lambda_1^3)) + [(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Lambda_2^3)) +$$
$$[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Pi_1^3)) + [(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Pi_2^3)) -$$
$$\tfrac{1}{2}\gamma_1[(\mathbf{U}_4 \odot (\mathbf{D}^2_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1) \otimes \mathbf{I}] -$$
$$\tfrac{1}{2}\gamma_2[((\mathbf{D}_{x_4}\mathbf{U}_4) \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\Big)\text{vec}(\mathbf{U}_3) = \mathbf{0} \tag{5.61}$$

Mode 4 unfolding

$$\mathbf{U}_4[\mathbf{U}_3 \odot (\text{Diag}(\mathbf{x}_2)\mathbf{U}_2) \odot (\mathbf{D}_{x_1}\mathbf{U}_1)]^T + \mathbf{U}_4\text{Diag}(\mathbf{x}_4)[(\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_2 \odot \mathbf{U}_1]^T +$$
$$[\mathbf{U}_4(\mathbf{U}_3 \odot (\mathbf{D}_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1)^T] \circ \Lambda_1^4 + [\mathbf{U}_4(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Lambda_2^4 +$$
$$[\mathbf{D}_{x_4}\mathbf{U}_4(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Pi_1^4 + [\mathbf{U}_4(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T] \circ \Pi_2^4 -$$
$$\tfrac{1}{2}\gamma_1\mathbf{U}_4(\mathbf{U}_3 \odot (\mathbf{D}^2_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1)^T - \tfrac{1}{2}\gamma_2\mathbf{D}^2_{x_4}\mathbf{U}_4(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1)^T = 0$$

$$\Big([\mathbf{U}_3 \odot (\text{Diag}(\mathbf{x}_2)\mathbf{U}_2) \odot (\mathbf{D}_{x_1}\mathbf{U}_1)] \otimes \mathbf{I} + [(\mathbf{D}_{x_3}\mathbf{U}_3) \odot \mathbf{U}_2 \odot \mathbf{U}_1] \otimes \text{Diag}(\mathbf{x}_4) + [(\mathbf{U}_3 \odot (\mathbf{D}_{x_2}\mathbf{U}_2) \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Lambda_1^4)) + [(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Lambda_2^4)) + [(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{D}_{x_4}]\text{Diag}(\text{vec}(\Pi_1^4)) + [(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{Diag}(\text{vec}(\Pi_2^4)) - \frac{1}{2}\gamma_1[(\mathbf{U}_3 \odot (\mathbf{D}_{x_2}^2\mathbf{U}_2) \odot \mathbf{U}_1) \otimes \mathbf{I}] \frac{1}{2}\gamma_2[(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{D}_{x_4}]\Big)\text{vec}(\mathbf{U}_4) = \mathbf{0} \tag{5.62}$$

The objective is to solve (5.59)-(5.62) iteratively for the unknown vectorized factor matrices. As in the previous example, the boundary value and the normality constraints have to be imposed before solving the equations. The first is done in the way similar to the earlier case. The normality constraint yields,

$$\int_{\Omega_1}\int_{\Omega_2}\int_{\Omega_3}\int_{\Omega_4} p(x_1, x_2, x_3, x_4)\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3\mathrm{d}x_4 = 1 \tag{5.63}$$

Approximating (5.63) using some quadrature and plugging the separable form for the density yields,

$$\begin{aligned}
&\approx \sum_{i=1}^{P_1}\sum_{j=1}^{P_2}\sum_{k=1}^{P_3}\sum_{l=1}^{P_4} w_{1,i}w_{2,j}w_{3,k}w_{4,l} \times p(x_{1,i}, x_{2,j}, x_{3,k}, x_{4,l}) \\
&= \sum_{i=1}^{P_1}\sum_{j=1}^{P_2}\sum_{k=1}^{P_3}\sum_{l=1}^{P_4} w_{1,i}w_{2,j}w_{3,k}w_{4,l} \times \sum_{m=1}^{R_U} (\mathbf{U}_1)_m \otimes (\mathbf{U}_2)_m \otimes (\mathbf{U}_3)_m \otimes (\mathbf{U}_4)_m \\
&= (\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T \text{vec}\left[\text{Mat}\left(\sum_{m=1}^{R_U} (\mathbf{U}_1)_m \otimes (\mathbf{U}_2)_m \otimes (\mathbf{U}_3)_m \otimes (\mathbf{U}_4)_m\right)\right] \\
&= (\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T \text{vec}[\mathbf{U}_1(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2)^T]
\end{aligned}$$

where vec(.) and Mat(.) refer to the vectorization and matricization operations. Therefore, we get,

$$(\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_2) \otimes \mathbf{I}]\text{vec}(\mathbf{U}_1) = 1 \tag{5.64}$$

Similarly, for other dimensions we have,

$$\begin{aligned}
(\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T[(\mathbf{U}_4 \odot \mathbf{U}_3 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{vec}(\mathbf{U}_2) &= 1 \\
(\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T[(\mathbf{U}_4 \odot \mathbf{U}_2 \odot \mathbf{U}_2) \otimes \mathbf{I}]\text{vec}(\mathbf{U}_3) &= 1 \\
(\mathbf{w}_1 \otimes \mathbf{w}_2 \otimes \mathbf{w}_3 \otimes \mathbf{w}_4)^T[(\mathbf{U}_3 \odot \mathbf{U}_2 \odot \mathbf{U}_1) \otimes \mathbf{I}]\text{vec}(\mathbf{U}_4) &= 1
\end{aligned} \tag{5.65}$$

As in the previous, we describe the results for this case in the section 5.8 as well.

5.6 A unified framework for the solution of non-stationary FPE

In the previous section, we presented two cases to demonstrate the key steps required in solving a discretized FPE. As it can seen that even for simple two and four dimensional examples, the full procedure is quite involved, and if not done carefully, could easily result in mistakes. Hence, we look for a framework that is compact enough to be employed for the solution of a general FPE. There have been several numerical methods, proposed in the literature, for solving FPE. Recently, a tensor based approach has been introduced by Y.Sun and M.Kumar in the series of papers, [SK14], [SK15a], [SK15b]. Their main idea is to express the multidimensional PDF in separable form, which is then tensorized together with the Fokker Planck operator (FPO).
In this section, we summarize the method for solving FPE via tensorization, as described in [SK15b]. The main objective here is to solve the tensorized version of (5.2) via ALS. The first step is to approximate the multivariate PDF $\mathsf{W}(\mathbf{x},\mathrm{t})$ as the sum of the product of uni-variate functions spatial and temporal basis functions $u_d^i(x_d)$ and $\tau^i(t)$.

$$\mathsf{U}(\mathbf{x},\mathrm{t}) \approx \sum_{i=1}^{R_U} \left[\left(\prod_{d=1}^{N} u_d^i(x_d) \right) \tau^i(t) \right] \tag{5.66}$$

Above, $\mathsf{U}(\mathbf{x},\mathrm{t})$ stands for the approximate representation with R_U as the rank of the approximation of the true PDF $\mathsf{W}(\mathrm{t},\mathbf{x})$. From now on, we only refer to the approximated density $\mathsf{U}(\mathbf{x},\mathrm{t})$. Note that the temporal dimension has also been resolved into R_U basis functions in this representation and is not ignored like in the *Ab-initio* cases. This signifies the fact that non-stationary solutions are being sought after. As will be clear in the subsequent analysis, time is treated just like an additional dimension. Next, all basis functions are discretized along their respective dimensions. This results in R_U basis vectors of length n_d for each of the spatial dimension *d*, and vectors of length n_t for the time. Along every dimension, basis vectors can be arranged into factor matrix form as $\mathbf{U}_d = \left[\mathbf{u}_d^1, \mathbf{u}_d^2, \cdots, \mathbf{u}_d^{R_U}\right]$ or in case of time as $\mathbf{\Gamma} = \left[\boldsymbol{\tau}^1, \boldsymbol{\tau}^2, \cdots, \boldsymbol{\tau}^{R_U}\right]$.

5.6.1 Chebyshev spectral differentiation

The numerical solution of the FPE is sought in a spatio-temporal hypercube, where each dimension is discretized with sufficient number of points. One of the main advantage of the solution via tensor decomposition is that the higher dimensional operations like the first and second order partial derivatives are decoupled into the corresponding single dimensional ones. Generally, a finite differentiation matrix (FDM) is used to approximate the first order derivatives, while the second order partial derivative is approximated by its square. One issue with FDM is that for higher order derivatives, the number of points required for an adequate and stable approximation grows much faster. As noted in [CBS00], the time domain discretization has to be much fine for the solution to be stable, if the explicit FDM is employed. Another problem with FDM is the so called *Runge phenomenon*, which refers to the problem of oscillation at the edges of an discretized interval. It occurs when using the polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points. E.g. when approximating the function $f(x)$ using n points $\{x_i\}_{i=0}^n$ together with the Lagrange polynomials ℓ_i, the resulting error can be expressed as,

$$f(y) - \sum_{i=0}^{n} f(x_i)\ell_i(y) = \frac{f^{(n+1)(\xi)}}{(n+1)!} \prod_{i=0}^{n} (y - x_i) \tag{5.67}$$

Two factors affect the approximation accuracy, the (n+1)th derivative $f^{(n+1)(\xi)}$ and the position of the points x_i. Since the first term is problem dependent, it leaves the judicious choice of the second to manage the approximation error. In [SK15b], Chebyshev spectral differentiation to generate the optimally spaces interpolation points. This is because among all the polynomials of degree n with leading coefficient 1 ($\prod_{i=0}^{n}(y-x_i) = y^n + a_{n-1}y^{n-1} + \cdots + a_0 y^0$), the unique polynomial which has the smallest maximum on interval [-1,1] is the Chebyshev polynomial. This leads to the use of its extrema as the interpolation locations,

$$x_{d_j} = \cos\left(\frac{j\pi}{n_d}\right) \qquad j = 0, 1, \cdots, n_d \tag{5.68}$$

These points are the projections onto the x-axis of equally spaced points on the unit circle. Point density is higher close to the end of the axis than in the middle.

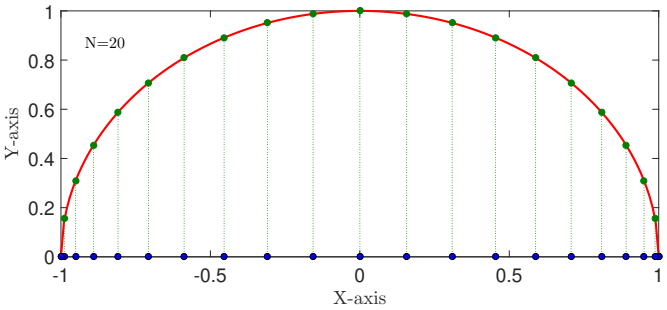


Figure 5.1: Chebyshev distributed points

As opposed to the finite difference matrix, the Chebyshev differentiation matrix has a dense structure. One such matrix along the *dth* dimension is defined as following,

$$D_d(i,j) = \begin{cases} \frac{2{n_d}^2+1}{6}, & i,j = 0 \\ -\frac{2{n_d}^2+1}{6}, & i,j = n_d \\ \frac{-x_j}{2(1-x_j^2)}, & i,j = 1, \cdots, n_d - 1, \\ \frac{c_i(-1)^{i+j}}{c_j(x_i - x_j)}, & i \neq j, i,j = 1, \cdots, n_d - 1 \end{cases}$$

where

$$c_i = \begin{cases} 2 & i = 0 \text{ or } n_d \\ 1 & otherwise \end{cases}$$

5.6.2 Tensorization of the Fokker Planck operator

It is now assumed that the discretization of the individual dimensions in (5.66) has yielded the density tensor $\mathcal{U}(\mathrm{t}, \mathbf{x})$ in the CPD form as per (5.17),

$$\mathcal{U}(\mathbf{x}, \mathrm{t}) = \sum_{i_u=1}^{R_U} \left(\bigotimes_{d=1}^{N} \mathbf{u}_d^{i_u} \right) \boldsymbol{\tau}^i, \tag{5.69}$$

Here, we abuse the notation a bit and represent the outer product between the basis vectors and factor matrices with $\bigotimes$. It will be explicitly mentioned when the sign is used to denote a Kronecker product. Having the PDF in the CPD format is a key factor in combating the curse of dimensionality. The next step is to tensorize the FPO $\mathcal{L}_{FP}$ in the form mentioned in (5.19).

$$\mathcal{L}_{FP} = \mathbb{A} \approx \sum_{i_A=1}^{R_A} \bigotimes_{d=1}^{N} \mathbf{A}_d^{i_A} \tag{5.70}$$

It is assumed that the scalars θ_i and μ_i have been absorbed into one of the loading vector/factor matrix.

The first task now is to express the individual terms in the drift vector $\mathbf{f}(\mathbf{x}, t)$ and the diffusion matrix $\Upsilon(\mathbf{x}, t)$ in the separable form as in the (5.66), which is followed by their discretization as done previously. Below, one such operation for the drift term $f_i((\mathbf{x}, t))$ is shown,

$$f_i(\mathbf{x}, \mathrm{t}) = \sum_{j=1}^{R_{1i}} \left[\left(\prod_{d=1}^{N} f_{i_d}^j(x_d) \right) f_{i_t}^j \right] \tag{5.71}$$

If the drift and diffusions are not be analytically expressed in the CPD format, they are then approximated using ALS or Fourier series. The next step is to apply the differential operator $\frac{\partial}{\partial x_i}$, given in form of the Chebyshev differentiation matrix D_N to this representation. This can be expressed as,

$$\begin{aligned} \frac{\partial}{\partial x_i} f_i(\mathbf{x}, t) \approx \sum_{j=1}^{R_{1i}} & \left(\bigotimes_{d=1}^{i-1} \mathrm{diag}(f_{i_d}^j) \right) \\ & \otimes \left(\mathrm{diag}(\mathbf{D}_d f_{i_d}^j) + \mathrm{diag}(f_{i_d}^j \mathbf{D}_d) \right) \\ & \left(\bigotimes_{d=i+1}^{N} \mathrm{diag}(f_{i_d}^j) \right) \otimes \mathrm{diag}(f_{i_t}^j), \end{aligned} \tag{5.72}$$

where diag represents the diagonalization of the vector under operation and R_{1i} the CPD approximation rank for $f_i(\mathbf{x}, t)$. This procedure is applied to all components of the drift vector and the diffusion matrix. For the later, the second derivative is given by squaring the matrix D_N. This results in the following tensorized form of the Fokker-Planck operator,

$$\mathbb{A} = \sum_{i_A}^{R_{A_0}} \left[\left(\bigotimes_{d=1}^{N} \mathbf{A}_d^{i_A} \right) \otimes \mathbf{A}_t^{i_A} \right], \tag{5.73}$$

where $\mathbf{A}_d$ and $\mathbf{A}_t$ are $n_d \times n_d$ and $n_T \times n_T$ matrices respectively. Generally, the drift and the diffusion terms do not have explicit time dependency. Hence $\mathbf{A}_t$ is usually replaced by I_t. The left hand side of (5.2) can also be expressed in the tensorized form as,

$$\frac{\partial}{\partial t} \approx \left(\bigotimes_{d=1}^{N} I_d \right) \otimes \mathbf{D}_t = \mathbb{A}_t. \tag{5.74}$$

where, $\mathbf{D}_t$ is the temporal differentiation matrix. Finally the whole equation can be written down as,

$$\mathbb{A}'\mathcal{U} = 0, \tag{5.75}$$

where $\mathbb{A}' = \mathbb{A} - \mathbb{A}_t$, with $R_A = (R_{A_0} + 1) \times (N + 1)$ as its overall approximation rank . Also please note that the time is treated as another dimension in the analysis.

5.6.3 Solution of the discretized FPE via R-ALS

Alternating Least squares method can be used to solve the tensor equation (5.75), but it would lead to a trivial solution. Additional constraint terms have to be added in order to have a non-trivial solution. Three such constraints have been identified in [SK15b], one for initial value, boundary value and normality each. This leads to the new formulation of the minimization objective function R,

$$\min_{\{\mathbf{u}_k^r, \boldsymbol{\tau}^r\}} ||\mathbb{A}'\mathcal{U}||_F^2 \;+\; \alpha||\mathbb{N}\mathcal{U} \;-\; \mathbb{U}_0||_F^2 \;+\; \beta||\mathbb{M}\mathcal{U}||_F^2 \;+\; \gamma||\mathbb{B}\mathcal{U} \;-\; \mathbb{Q}||_F^2. \quad (5.76)$$

α, β and γ refer to the penalties associated with the three constraints. The first penalty term refers to the initial value constraint, where $\mathcal{U}_0$ is the initial value tensor,

$$\mathbb{U}_0 = \sum_{i_u=1}^{R_{U_0}} \left[\left(\bigotimes_{d=1}^{N} u_{0,d}^{i_u} \right) \otimes [1] \right] \quad (5.77)$$

and $\mathcal{N}$ is the projection tensor given by,

$$\mathbb{N} = \left(\bigotimes_{d=1}^{N} \mathbf{I}_{\mathbf{x}_d}^{i} \right) \otimes e \quad (5.78)$$

and e is given by the row vector $[1, 0, \cdots, 0]$. Likewise the boundary value constraint is realized using the tensor $\mathcal{M}$,

$$\mathbb{M} = \sum_{i_M=1}^{N} \left[\left(\bigotimes_{d=1}^{i_M-1} \mathbf{I}'_{\mathbf{x}_d} \right) \otimes \mathbf{I}''_{\mathbf{x}_{i_M}} \left(\bigotimes_{d=i_M+1}^{N} \mathbf{I}_{\mathbf{x}_d} \right) \otimes \mathbf{I}_t \right] \quad (5.79)$$

where $\mathbf{I}'_{\mathbf{x}_d} = \text{diag}([0, 1, \cdots, 1, 0])$, $\mathbf{I}''_{\mathbf{x}_d} = \text{diag}([1, 0, \cdots, 0, 1])$ and $\mathbf{I}_{\mathbf{x}_d}$ is the identity matrix. Finally the normality constraint term contains tensors,

$$\mathbb{B} = \left(\bigotimes_{d=1}^{N} \mathbf{b}_d \right) \otimes \mathbf{I}_t, \qquad \mathbb{Q} = \left(\bigotimes_{d=1}^{N} [1] \right) \otimes \mathbb{1}. \quad (5.80)$$

with $\mathbb{1}$ being the unit vector and b_d the Clenshaw-Curtis quadrature for the *dth* dimension. The root mean squared error R_{rms} is given as, $\sqrt{R/(n_T \prod_{d=1}^{N} n_d)}$.

Algorithm 11 Regularized Alternating Least Squares

procedure R-ALS(α,β,γ,δ,ϵ_1,ϵ_2,ω_0,j_{max},R_U^0,R_U^{max},$\mathcal{U}_0$)
  Initialize:
  Get $\mathbf{x}_k, \mathbf{D}_{\mathbf{x}_k} \forall$ k
  Get $\mathbf{t}, \mathbf{D}_\mathbf{t}$
  Tensorize the FPO as in (5.73)
  R_U=R_U^0
  Set $\mathcal{U}$ equal to $\mathcal{U}_0$
  while $R_{rms} > \epsilon_1$ & $R_U \leq R_U^{max}$ **do**
    $\omega = \omega_0$
    while $\Delta R_{rms} > \epsilon_2$ & $j \leq j_{max}$ **do**
      for k=1 : N+1 **do**
        Solve for $\overrightarrow{\mathbf{u}_k}$ or $\overrightarrow{\tau}$ using regularized eqn.(5.82)
      end for
      $\omega = \delta \,.\, \omega$
      j=j+1
    end while
    if $R_U < R_U^{max}$ **then**
      Initialize: $\mathbf{u}_k^{R_U+1}, \mathbf{t}^{R_U+1}$
      $R_U = R_U + 1$
    else
      break
    end if
  end while
  Reshape $\overrightarrow{\mathbf{u}_k}$ / $\overrightarrow{\boldsymbol{\tau}}$ into $\mathbf{U}_k$ / $\boldsymbol{\Gamma}$
  return Factor matrices $\mathbf{U}_d$
end procedure

In [SK15b], (5.76) is solved directly yielding,

$$[\mathbf{M} + \alpha\mathbf{M}_I + \beta\mathbf{M}_B + \gamma\mathbf{M}_N]\,\overrightarrow{\mathbf{u}_k} = \alpha\mathbf{v}_I + \gamma\mathbf{v}_N \tag{5.81}$$

where $\overrightarrow{\mathbf{u}_k}$ represents the vectorized factor matrix $\mathbf{U}_k$ i.e $\overrightarrow{\mathbf{u}_k} = \text{vec}(\mathbf{U}_k)$. A similar expression can be derived for the time factor matrix $\boldsymbol{\Gamma}$, yielding $\overrightarrow{\tau}$. $\mathbf{M}$, $\mathbf{M}_I$, $\mathbf{M}_B$ and $\mathbf{M}_N$ are all block matrices with $R_U \times R_U$ sub-matrices. We provide the description of these sub-matrices in the Appendix. We make a small change and instead of solving the problem directly, we regularize the objective function and then solve the ALS problem. This leads to the following,

$$\overrightarrow{\mathbf{u}_k} = [\mathbf{M} + \alpha\mathbf{M}_I + \beta\mathbf{M}_B + \gamma\mathbf{M}_N + \omega I_R]^{-1} \times \left(\alpha\mathbf{v}_I + \gamma\mathbf{v}_N + \omega\overrightarrow{\mathbf{u}_k}\right) \tag{5.82}$$

ω is the regularization parameter and the I_R is identity matrix of appropriate dimensions. It is recommended to gradually reduce ω after each successive iteration [LKN13]. Equation (5.82) is a single step of a series of iterations, also called refinements. We describe the implementation of the regularized ALS algorithm for solving (5.82) in Algorithm 11, where R_U^{max} is the maximum approximation rank, j_{max} is the maximum number of refinement iterations, δ is the regularization control parameter. while ϵ_1 and ϵ_2 are user defined tolerances.

5.7 Tensorized Filter: A Tensor decomposition based nonlinear filtering algorithm

Our main contribution is to combine the tensor based solution for FPE with the log-homotopy based flow method for the measurement inclusion, and thus defining a recursive Bayesian filtering strategy. We start with the initial density tensor $\mathcal{U}_0$ at t_0. In the time propagation step, we solve the FPE with the real time flow $\mathbf{f}(\mathbf{x}, \lambda)$ and diffusion $\Upsilon(\mathbf{x}, \lambda)$ to get the tensorized prior density $\mathcal{U}_{\text{prior}}$ at time step t_1. Next using this as the initial condition, we again solve the FPE but this time based on the homotopy flow $\mathbf{f}_H(\mathbf{x}, \lambda)$ as given in (6.1.2) in order to get the tensorized posterior density $\mathcal{U}_{\text{post}}$.

$$\mathbf{f}_H(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda) \tag{5.83}$$

with $\mathbf{A}(\lambda)$ and $\mathbf{b}(\lambda)$ given by,

$$\mathbf{A}(\lambda) = -\frac{1}{2}\mathbf{P}\mathbf{H}^T(\lambda\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H} \tag{5.84}$$

$$\mathbf{b}(\lambda) = (I + 2\lambda\mathbf{A})[(I + \lambda\mathbf{A})\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} + \mathbf{A}\hat{\mathbf{x}}]. \tag{5.85}$$

In this manner we incorporate measurements, which completes one cycle of the recursive Bayesian estimation. We term our new filter as the Tensor-Homotopy based filtering, or simply the *tensorized filter*. The full procedure is described in the Algorithm 12.

Algorithm 12 Tensorized filter

procedure TENSORIZED DHF(Process model (5.1) & Measurement model (5.4))
  Initialize: $\mathcal{U}_0$,k_{max},α,β,γ,δ,ϵ_1,ϵ_2,ω_0,j_{max},R_U^{max}
  Measurements: $\{\mathbf{z}_{t_1}$,$\mathbf{z}_{t_2}$,...,$\mathbf{z}_{t_{k_{max}}}\}$
  Tensorize the realtime and pseudo-time FPOs
  Set R_U^0 = 1
  for k=1:k_{max} **do**
    Prediction step
    Resolve $t \in (t_{k-1}, t_k]$
    Apply Algorithm 11 with the drift $\mathbf{f}(\mathbf{x}, t)$ and diffusion $\boldsymbol{\Upsilon}(\mathbf{x}, t)$ to get prior tensor $\mathcal{U}_{\text{prior}}$ at $t = t_k$, starting with $\mathcal{U}_{\text{post}}$ at $t = t_{k-1}$
    $R_U^0 = R_U^{max}$
    Measurement Update step
    Resolve $\lambda \in (0, 1]$
    Apply Algorithm 11 with the drift $\mathbf{f}_H(\mathbf{x}, \lambda)$ to get posterior tensor $\mathcal{U}_{\text{post}}$ at $\lambda = 1$ starting with
    $\mathcal{U}_{\text{prior}}$ at $t = t_k$
  end for
  return $\mathcal{U}_{\text{prior}}$ & $\mathcal{U}_{\text{post}}$ at t_k
end procedure

There are two important observations here. If the drift and the diffusion terms cannot be written in the separable form, then an approximation has to be made using some numerical method. This can be done using the ALS algorithm to find the factor matrices to define the FPO. Secondly, the homotopy drift terms, as given by (6.1.2) have explicit dependence on the pseudo-time λ. Thus in the tensorized representation of FPO, identity matrices for the time dimensions

cannot be used. Instead the appropriate matrices are chosen based on the CPD approximation of the homotopy drift terms.

5.8 Results

In this section, we describe the results for the studied examples. This comes in two parts. First, the results for the solution of stationary FPE via *ab-initio* approach will be given. This will be followed by an example describing a dynamic nonlinear filtering case where we use our newly devised *Tensorized filter* to estimate the probability density tensor.

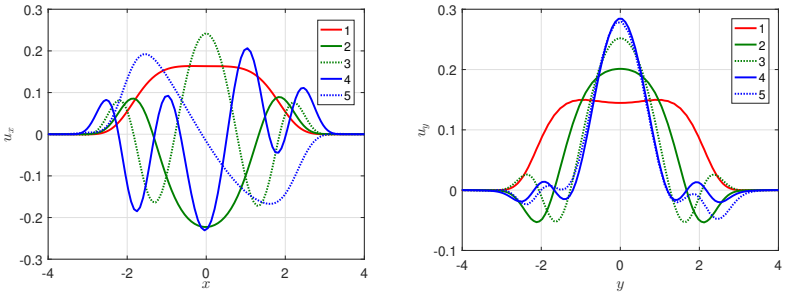
5.8.1 Solution for stationary FPE

5.8.1.1 2D density

We start with the solution for the test example mentioned in the section 5.5.1. The two dimensional state equations are given as,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -bx_2 - x_1 - a(x_1^2 + x_2^2)x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} g\eta(t)$$

In the current example, we set a=0.125, b=-0.5, g=1 while the noise variance σ_η is set to 0.4. We solve iteratively solve (5.47) and (5.48). We use discretize both axis in the interval [-4 , 4] with 69 Chebyshev points in each of them. Also we use five basis functions for each dimension, which are plotted in the following,



(a) (b)

Figure 5.1: Basis function along (a) x and (b) y

All basis functions have been L2-normalized. We note that all of them show significant mass in the interval [-3 , 3] and assume negligible values outside. Symmetry of the basis functions can also be noted in all but two, basis function 4 and 5 of along x axis, where they are both anti-symmetric and also opposed to each other as well.
The total number of degree of freedom is given by (number of nodes per dimension) $\times$ (number of dimension) $\times$ (number of basis function) = $69 \times 2 \times 5 = 690$. Next, we plot the 2D stationary density generated thereof and the difference w.r.t. the exact solution in figures 5.2 a & b. (5.23).

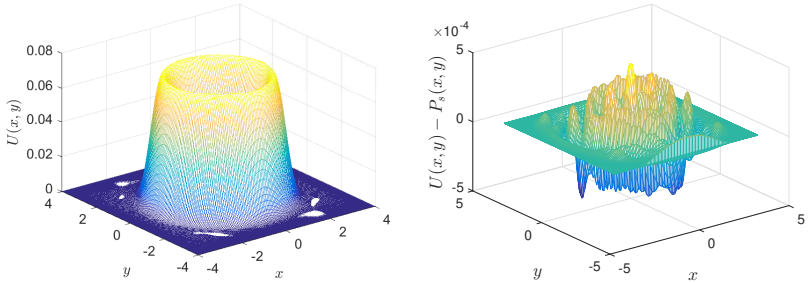


Figure 5.2: (a) Stationary density U(x,y) and (b) the error

We note that the 2D density appears like a mug that tapers towards the top instead to the bottom. Next, we see that error in the estimated solution is approximately of the order of 10^{-4}. As it can be seen that fairly decent result have been achieved with the current parameter setting.

5.8.1.2 4D density

Next, we give the results for the solution of the 4D FPE described in the section 5.5.2. The state space model is given as,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -a_1 & -a_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -a_3 & -a_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}$$

We set $a_1 = 1$, $a_2 = 1$, $a_3 = 2$, $a_4 = 1.2$, while $T = 0.5$. Also, the noise powers are given by $\mathbb{E}[\xi_1^2] = 2a_1a_2T = 1$ and $\mathbb{E}[\xi_2^2] = 2a_3a_4T = 2.4$. This leads to the following form of the stationary density,

$$\begin{aligned} P_s(x_1,x_2,x_3,x_4) &= \prod_{i=1}^{P} \left(\frac{1}{16\pi^2 T^2 \sqrt{a_{2i-1}}} \right)^{-\frac{1}{2}} \exp\left[-\frac{1}{4T} \sum_{i=1}^{P} \left(x_{2i-1}^2 + \frac{x_{2i}^2}{a_{2i-1}} \right) \right] \\ &= \left(\frac{1}{4\pi^2} \right)^{-\frac{1}{2}} \left(\frac{1}{8\pi^2} \right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2} \left(x_1^2 + x_3^2 + x_2^2 + (1/2)x_4^2 \right) \right] \end{aligned}$$

We discretize each axis with 100 Chebyshev points in the interval [-5 , 5] and have 10 basis functions along each dimension. We iteratively solve (5.59)-(5.62) together with (5.65). Below, we plot the basis functions along the four dimensions,

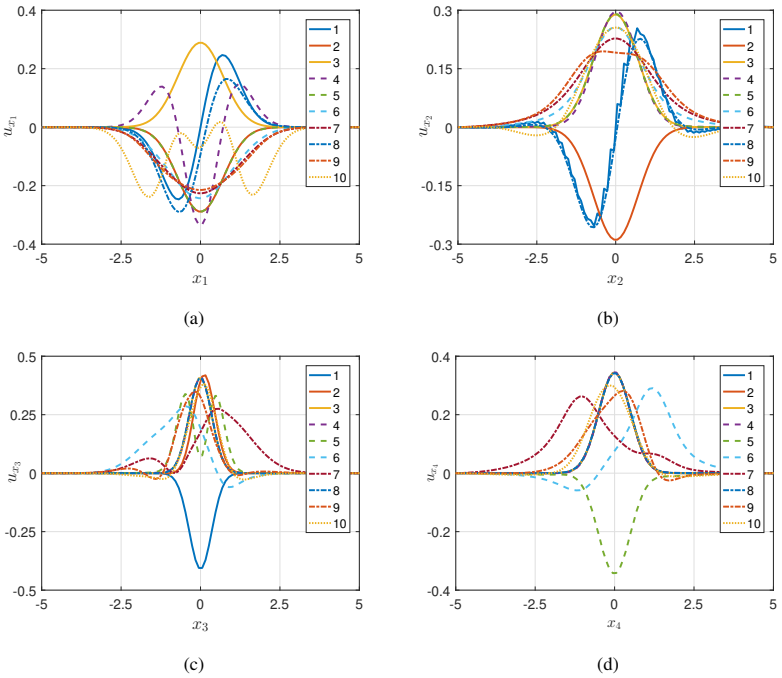


Figure 5.3: Basis function along (a) x_1, (b) x_2, (c) x_3 and (d) x_4.

We note that the spread of the basis functions is lowest for the third dimension, whereas the highest spread is shown by the basis function along the second dimension. The total degree of freedom for the discretized problem is $100 \times 4 \times 10 = 4000$ which is about $\frac{1}{400000}$ times that of the fully tensorized density if the same number of points are assumed along each dimension i.e. a total of 100^4. This shows that the scheme utilizing the tensorization of the FPO together with the CPD representation of the density can achieve a good solution while requiring significantly less number of discretization points. As will be seen, this result will be very useful when solving the FPE for a dynamic case in a nonlinear filtering setting.

Next, we plot the six 2D marginal densities.

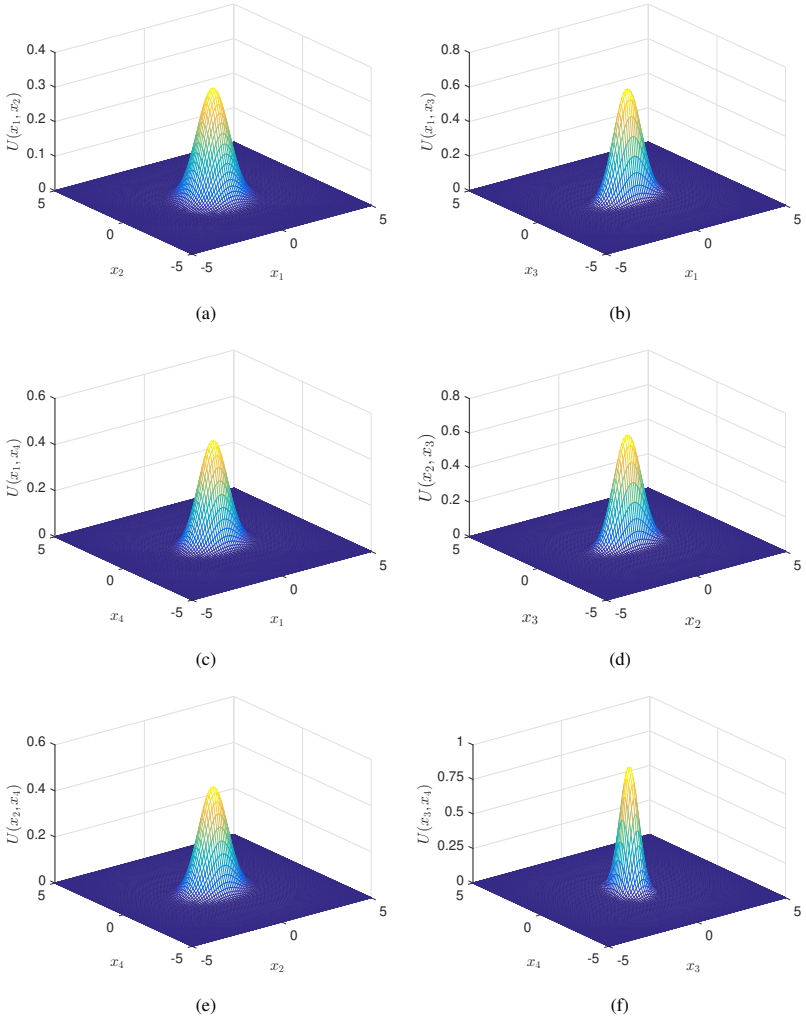


Figure 5.4: Marginalized densities along (a) x_1-x_2 , (b) x_1-x_3 , (c) x_1-x_4 , (d) x_2-x_3 , (e) x_2-x_4 and (f) x_3-x_4 axis.

5.8.2 Solution for non-stationary FPE: Nonlinear filtering

Finally, we discuss a nonlinear filtering example where our devised *Tensorized filter* will be employed to recursively solve for the prior and posterior densities. FPE will be solved twice using the R-ALS, once for the time update step and the second time for the measurement update using the *exact flow*. We consider a single target moving in a plane, represented by a four state nearly constant velocity model (NCV) for our numerical experiments.

$$d\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}dt + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} d\mathbf{w} \tag{5.86}$$

where the state vector $\mathbf{x} = [x, y, v_x, v_y]$, and the noise power spectral density $\mathbf{Q} = \sigma_w^2.I_{2\times 2}$. Time constitutes the fifth dimension. As can be seen, the drift components can easily be written in separable form. We use range and angle measurements from a radar sensor located at the origin.

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ \tan^{-1}(\frac{y}{x}) \end{bmatrix} + \begin{bmatrix} v_r \\ v_\theta \end{bmatrix} \tag{5.87}$$

with $v_r \sim \mathcal{N}(0, \sigma_r^2)$ and $v_\theta \sim \mathcal{N}(0, \sigma_\theta^2)$ representing the two mutually uncorrelated measurement noises. We choose time interval between two consecutive measurements to be 1 second. Domains for the x and y-axis are discretized with 150 points such that x , y $\in$ [4970 , 5130], while v_x and v_y have 150 points each in the interval [-10 , 10]. Both real time t and the pseudo-time λ are discretized for each prediction interval with 50 points. As evident the velocity axis is more finely discretized than the position ones. $\mathcal{U}_0$ is initialized with univariate Gaussian vectors for the two position and two velocity dimensions. The mean and variance of the initial position basis vectors are chosen randomly from $\mathcal{N}(5000, 50)$. The same is done for the two velocity vectors, but in this case the sampling density is $\mathcal{N}(1, 2)$. The t dimension is initialized with a vector consisting of all ones i.e. $\mathbb{1}$.

5.8.2.1 R-ALS parameter tuning

The next task is to decide upon the values of different parameters used in R-ALS, as described in the Algorithm 11. In order to do that, we simulate the time propagation for a total of 20 seconds. Since the process model is linear and Gaussian, we can exactly know about the form of the density given that the initial density is also Gaussian. E.g. given the initial density $\mathcal{N}(\mathbf{x}_0, \mathbf{P}_0)$, the change in the state density and the covariance matrix is given by the Time update step of the Kalman-Bucy filter,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{F}(t)\mathbf{x}(t) \tag{5.88}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}(t) = \mathbf{F}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}(t)^T + \mathbf{Q}(t) \tag{5.89}$$

Given the initial condition, we numerically solve this set of equations in the interval [0 20], to get the mean and the covariance of the Gaussian at t=20s. We also simulate the scenario using the R-ALS to get the numerical representation for the density. Then, we compute the integrated square error (ISE) between the densities according to the following

$$\epsilon_r \left(P_{KB}, \hat{P}_{ALS} \right) = \int \int \int \int \| P_{KB}(x, y, v_x, v_y) - \hat{P}_{ALS}(x, y, v_x, v_y) \|^2 \mathrm{d}x \mathrm{d}y \mathrm{d}v_x \mathrm{d}v_y \tag{5.90}$$

where $P_{KB}(x, y, v_x, v_y)$ is the solution found by numerically solving (5.88), while $\hat{P}_{ALS}(x, y, v_x, v_y)$ is the 2D marginal formed after solving the R-ALS. The idea is start with a base set of parameters values , and to find the optimal by changing one at a time. The value thus found will be used later on while using the filtering algorithm 12.

Number of points per dimension and basis vectors First, we describe the effect of changing the maximum number of the basis vectors RU_{max} and the number of points per dimension N_p,

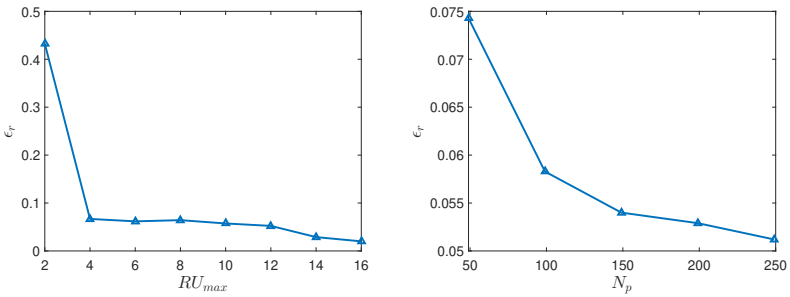
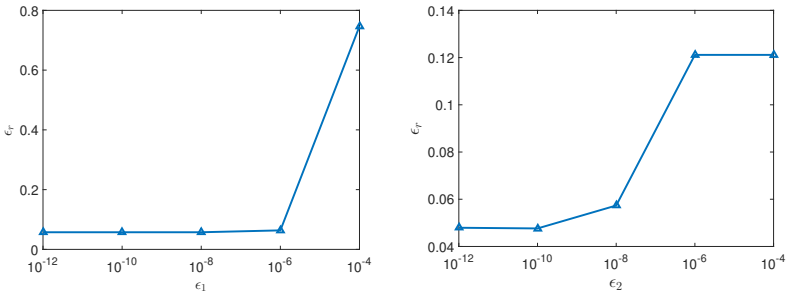


Figure 5.5: ISE v.s. (a) Number of basis functions (b) Number of points per dimension.

We note that ISE decrease significantly for RU_{max} between 2 and 4. Afterwards it stays constant more or less until the RU_{max} is 10. After that it drops again but the drop is shallow. In the second plot, we see a constant drop in the error against the value of number of points per dimensions considered, albeit the drop being less significant after $N_p > 150$. Processing time is another consideration that has to be taken into account. The processing time increases exponentially with the increase in both the number of basis functions and the number of the points per dimensions, though the rise is much steeper against the N_p. In particular, the processing time increases six fold with by changing N_p from 150 to 250. Against the RU_{max}, the increase seems almost linear up to 10, after which the exponential growth is more clearly visible. Therefore, we choose a middle ground by selecting RU_{max}=10 while N_p is set to be 150.

Error tolerances Next, we describe the effect of varying the two tolerances, ϵ_1 and ϵ_2, on the ISE.

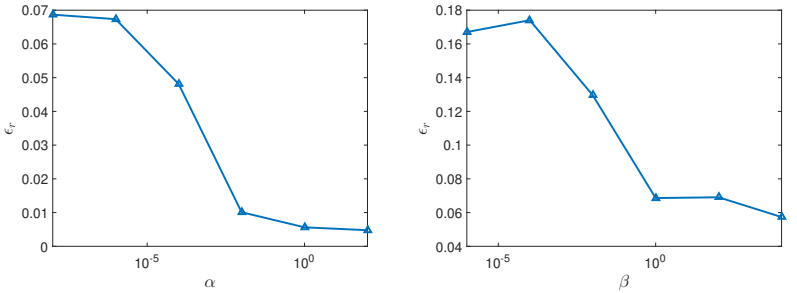


(a) (b)

Figure 5.6: ISE v.s. the two tolerances

The first of the two tolerances pertain to the absolute normalized error, whereas the second one is about the relative error. We note that the choice of ϵ_1 is not very critical, whereas the estimation accuracy is strongly dependent on the value of ϵ_2. Processing time increases exponentially with the lowering of ϵ_2, unless the iterations are aborted after a fixed number of times j_{max} have been run. The later is there to ensure that the loop is always terminated. Therefore, we set ϵ_1 and ϵ_2 to $10^{=10}$, while j_{max} to 50.

Constraint penalties Finally, we discuss the effect of changing the three penalty terms.



(a) (b)

Figure 5.7: ISE v.s. (a) Initial value and (b) the Boundary value constraint

We select the value of α from the vector $[10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1, 10^{2}]$ while β and γ from $[10^{-6}, 10^{-4}, 10^{-2}, 1, 10^{2}, 10^{4}]$. It should be reminded that α, β and γ refer to the penalty associated with the initial value, boundary value and the normality constraints respectively. It can be seen that the R-ALS algorithm is most sensitive to the choice of the penalty term enforcing the initial value constraint. α has to be set largest out of the three. Next, we see that β also has to be set to a fairly large value in order to minimize the estimation error. γ the on other hand, appears to have a detrimental role. Though its presence is necessary to ensure that the

density tensor is always normalized, a larger value could be contribute to a significant increase in the ISE.

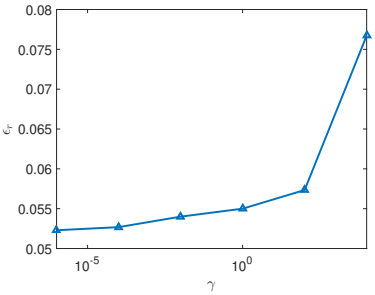


Figure 5.8: ISE v.s. Normality constraint

Therefore based on the results, we choose $\alpha = 10^4$, $\beta = 10^2$, while $\gamma = 10^{-6}$. Below, we summarize the values of the ALS algorithm parameters together with the values for the noise covariances.

Parameter	Value	Parameter	Value	Parameter	Value
α	10^4	β	10^2	γ	10^{-6}
ω	10^{-10}	δ	10^{-1}	ϵ_1	10^{-10}
ϵ_2	10^{-10}	k_{max}	50	RU_{max}	10
σ_w^2	0.01	σ_r^2	100	σ_θ^2	$\frac{\pi}{180}$

Table 5.1: R-ALS Parameters

5.8.2.2 Prediction

To show how our proposed method works, we first present the results for the prediction step. We have plotted the L2-normalized basis vectors and the two dimensional position marginal $U(x, y)$ at t=20s, in Figure 5.9.

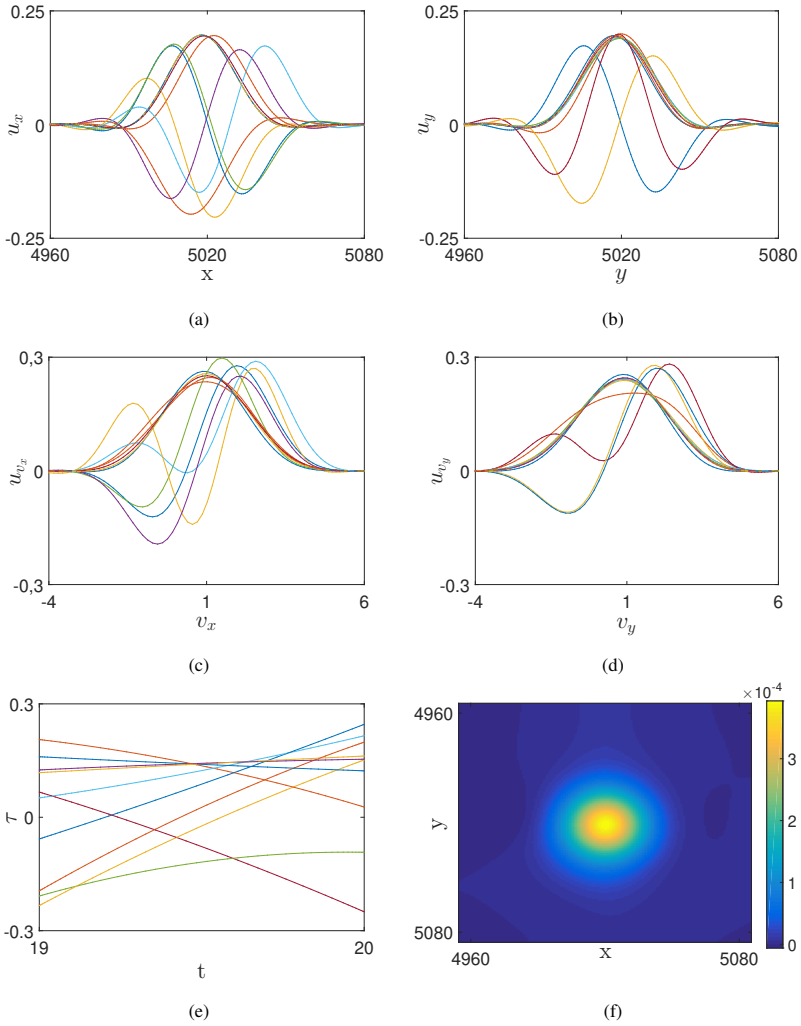


Figure 5.9: Basis function along (a) x, (b) y, (c) u_x, (d) u_y (e) t axis and the marginalized density $U(x, y)$ at t = 20s.

Note that these results are for pure time propagation and do not include the measurement inclusion. First the FPO is tensorized and represented in the form of equation (5.73). This yields 25

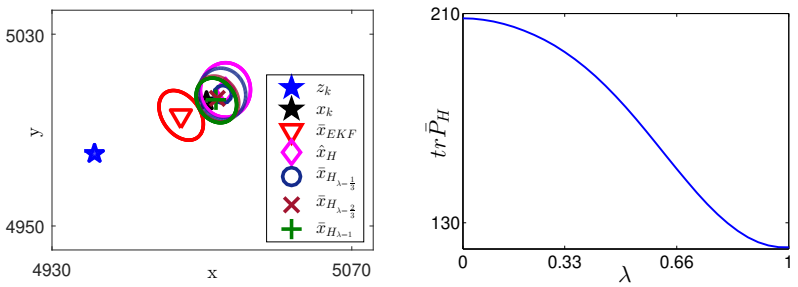
terms for the representation of $\mathcal{A}'$ in the equation (5.75) i.e. $R_A = 25$. Next, the prediction of the density is carried out in steps of one second as per the Algorithm 2. The maximum number of basis vectors R_U^{max} is capped at 8 for all dimensions. The results after each prediction step is used as the intial condition for the next. The last elements of all time vectors are absorbed in one of the four spatial dimensions to form the initial tensor for the next time step. This is possible as $\mathcal{U}_0$ does not involve the time dimension.

5.8.2.3 Measurement Inclusion via homotopy

The main step is to write the log-homotopy based flow equation in the separable form, which for the exact flow equation (6.1.2), is a straightforward step. For instance the *ith* flow component can be written as,

$$\begin{aligned} f_{H_i}(\mathbf{x},\lambda) &= a_{i1}(\lambda)x + a_{i2}(\lambda)y + a_{i3}(\lambda)v_x \\ &\quad + a_{i4}(\lambda)v_y + b_i(\lambda) \\ &= (x)(1)(1)(1)(a_{i1}(\lambda)) + (1)(y)(1)(1)(a_{i2}(\lambda)) \\ &\quad + (1)(1)(v_x)(1)(a_{i3}(\lambda)) + (1)(1)(1)(v_y)(a_{i4}(\lambda)) \\ &\quad + (1)(1)(1)(1)(b_i(\lambda)) \\ &= \sum_{j=1}^{5}\left[\left(\prod_{d=1}^{4} f_{i_d}^j(x_d)\right) f_{i_\lambda}^j\right] \end{aligned}$$

where a_{ij} and b_i are entries of the matrix $\mathbf{A}(\lambda)$ and vector $\mathbf{b}(\lambda)$ respectively. All components are expressed in the above form. The next step is to tensorize the Fokker-Planck operator in the pseudo-time λ, following the same procedure as for the real time. For the exact flow equation, $R_A = 105$. As for the progressive prediction mentioned before, the prior density tensor $\mathcal{U}_{prior}$ at time t_k serves as the starting point to the Algorithm 1.



(a) (b)

Figure 5.10: (a)Position error vs. time and (b) Velocity error vs. time.

It can be noted that the equation (6.1.2) required the prior covariance matrix P, and the evaluation of the measurement function ψ and its Jacobian H. In a typical implementation, P is taken from a parallel running EKF [DC12] or as discussed in the chapter 3, using some shrinkage estimation method. However, as we already have the prior density available in the tensorized

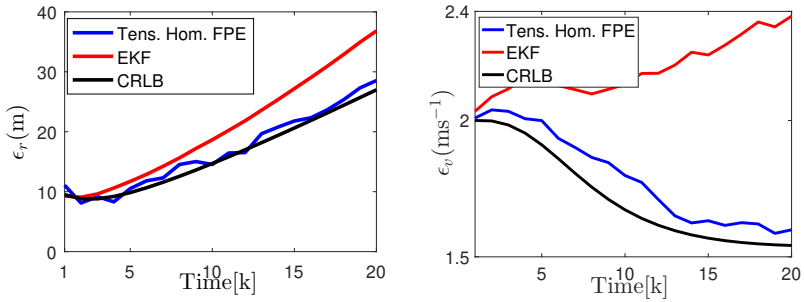
format, we can directly get this. In Figures 5.10a and 5.10b, we show the results for a generic measurement update step. The first plot shows the true position vector x_0, measurement vector z_k, EKF based position estimate $\bar{x}_{EKF}$, Tensor-homotopy based prior estimate $\hat{x}_H$, and estimates $\bar{x}_H(\lambda)$ for three values of λ with the last being the posterior estimate $\bar{x}_H$. Also shown are the associated error ellipses. It can be seen that the error ellipses shrink as λ approaches 1. This can also be inferred from the trace of the covariance matrix $P_H(\lambda)$. The purpose of this presentation is to show the effectiveness of homotopy based measurement update step.

5.8.2.4 Filtering results

Having analyzed the results of the individual prediction and measurement updates, we now analyze their combined effect in the Bayesian filtering. We run 20 simulations for the Tensor-Homotopy based filtering, with randomized intial conditions. For EKF we use 100 simulation runs. We compare our results with that of EKF and the Crámer-Rao Lower Bound (CRLB). We note that $\sigma_r < D_k\sigma_\theta \ \ \forall k$, where D_k represents the distance of the target from the radar location at time instant k. The considered example has simple linear dynamics, but the high cross range measurement error creates a challenging scenario from the filtering perspective. We use the root mean squared error (RMSE) as the performance metric. For the position, RMSE formula is shown below

$$\epsilon_r = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[(x_k^m - \bar{x}_k^m)^2 + (y_k^m - \bar{y}_k^m)^2\right]} \tag{5.91}$$

where M is the number of Monte Carlo simulation runs which equals 20 in the current case.



(a) (b)

Figure 5.11: (a) Position error vs. time and (b) Velocity error vs. time.

Similar equation is used to define the velocity error ϵ_v. In Figure 5.11a and 5.11b, we show the position and velocity error, ϵ_r and ϵ_v averaged over all runs. Finally we compare the execution time of both prediction and the measurement update steps. Simulations were performed on a computer with Intel Core2 Quad with 2.66 GHz processors and 4 GB RAM. The processing time depends on many factors like the problem dimensionality, number of points along dimensions n_d, the maximum approximation rank of the PDF R_U^{max} and of the FPO R_A, the tolerances ϵ_1 and ϵ_2, maximum number of iterations j_{max} etc. For the choice of parameters

mentioned in Table 5.1, the first prediction step for $t \in (t_0, t_1]$ takes about 50 seconds. This is because basis vectors are incrementally increased, starting from 1 up to R_U^{max}, each refined until the termination criterion is met. Subsequently, it takes about 20 seconds per prediction step as the number of basis vectors are kept constant. The computationally most expensive part is the measurement update. As the FPO in that case is comprised of 105 matrices, the R-ALS iterations take about 200 seconds. Please note that in [SK15a], authors have worked on a simple example with 1D linear measurement model. In such cases, the likelihood function can easily be expressed in the separable form, which leads to a straightforward measurement update. This is not the situation in our case, as likelihood separation would need an approximation. On the other hand we use the homotopy flow, for which it is not an issue, although the penalty is paid in terms of the extra processing cost. We can see that our method almost achieves the CRLB for both position and velocity errors, as opposed to the EKF based estimate. Apart from the tensorization based solution of FPE, the homotopy based particle flow is particularly important in achieving this level of performance.

Finally, we plot the 2D marginalized posterior density $U_{post}(x, y)$ at different time instances in Figure 5.12.

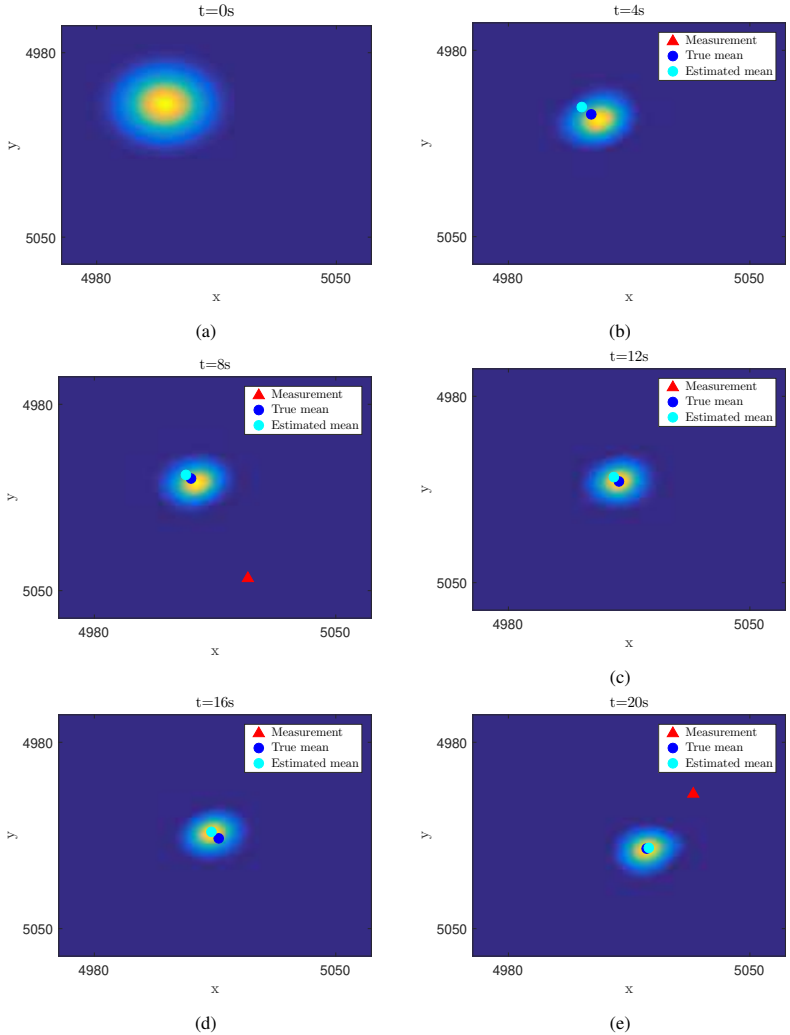


Figure 5.12: (a) Initial density $U_0(x, y)$ and the marginalized densities $U(x, y)$ at (b) 4s , (c) 8s , (d) 12s , (e) 16s and (f) 20.

Since high values for the measurement noise covariances have been chosen, measurements are spread over a large area. Hence, they do not show up in every plot i.e. measurements are out side the discretized area.

5.9 Possible avenue for Improvements

ALS based solution of the tensorized FPE is yielded by optimizing the multidimensional objective function, which could turn out to be non-convex. In such cases the ALS is not guaranteed to converge to a global minimum. Also it is well known to have a slow convergence rate. Regularization helps in the speeding up, but only up to a certain extent. There are other suboptimal methods described in the literature that approximately solve the constrained least squares for tensors. Some important contributions towards this end include the Enhanced Line Search (ELS) [RCH08], replacement of the ALS iterations by several joint diagonalization problems and the subsequent matching to get the best estimates of the factor matrices [RH08], a semi-algebraic framework for approximate CPD via Simultaneous Matrix Diagonalization and Generalized Unfolding (SMD) [RSH12] and a Semi-Algebraic Tensor decomposition (SALT) based on Joint Eigenvalue decomposition [LA11]. In particular, a cause of the slow convergence has been noted to be the so called *swampiness*. This occurs when two or more loading vectors in a factor matrix are collinear or nearly collinear. In [TPC16] a new algorithm for CPD has been proposed that is based on the random partitioning of matrices, where each of the portion is updated individually. It has been shown to achieve dramatic improvements in the convergence. Secondly, in our work we have used a static grid and therefore had to discretize a large domain. It can be conjectured that a significant speed up in the computation can be made by using dynamic grid adaptation. One idea is to use the concept described in [CBS00] for moving the grid by computing the significant domain based on the Chebyshev inequality. The other approach, which can be combined with it is the *Sparse grids*, as defined and used in [KS13] and [Gar01]. Thirdly, since the FPO for the *exact flow* contains 105 matrices, the processing time can potentially be lowered by using other flows in the FPE defining the measurement update step, provided the FPO is approximated using lesser terms. It is hoped that for higher dimensional nonlinear/non-Gaussian problems, the full power of the tensor based approach can be exploited and the extra computational cost can be justified by the higher level of accuracy achieved.

5.10 Conclusion

In this chapter, there two main objectives were pursued. The first objective was to describe a numerical framework for the solution of the Fokker-Planck equation. The second one was to propose a new nonlinear filtering algorithm that uses the log-homotopy based particle flow within the developed framework. We have studied problems pertaining to two separate classes: systems admitting stationary solution and nonlinear filter problems that do not admit stationary densities. An *Ab Initio* method was followed to demonstrate the basics of the solution of FPE via tensor methodology. The important take away lesson is that the step by step method though very lucid in explaining the intricacies of the tensorization based FPE solution, is very rigorous, and hence prone to errors. Therefore, we looked for a unified framework, which was found in the CPD-ALS based solution for FPE ,as developed by W.Sun and M.Kumar ([SK14], [SK15b] and [SK15a]). Expressing the probability density and the FPE in the CPD form has been identified as the key in fighting the curse of dimensionality in the discretized problems. A cost function comprising of the FPE and the associated constraints is formed and minimized. The resulting matrix-vector equation is solved via the regularized Alternating least squares (R-ALS), yielding the basis vectors. In the second part, we devised a nonlinear filtering algorithm, named as the *Tensorized filter*. This is achieved by combining the tensor framework with the log-

homotopy based particle flow. As opposed to the more widely used tensor multiplication based measurement update [DKG16] or particle filter type solutions (DHF), we use the homotopy flow for solving the tensorized FPE. Hence, for a single Bayesian recursive step, the FPE is solved twice, first w.r.t. the real time and the second w.r.t. the pseudo time. We study three examples: a 2D and a 4D case admitting stationary solution, and one five dimensional (four spatial and one temporal) nonlinear filtering example. The first two are solved using the equation derived via *Ab-Initio* method. For the nonlinear filtering example, we used the R-ALS and have demonstrated that our scheme not only works but in fact its estimation error approaches the the Crámer-Rao lower bound, albeit at the cost of significant computational time.

Appendix

The objective is to minimize the cost function formed in parts by the tensorized FPO and the initial value, boundary value and the normality constraint terms.

$$\min_{\{\mathbf{u}_k^r, \boldsymbol{\tau}^r\}} R = \min_{\{\mathbf{u}_k^r, \boldsymbol{\tau}^r\}} ||\mathbb{A}'\mathcal{U}||_F^2 + \alpha||\mathbb{N}\mathcal{U} - \mathbb{U}_0||_F^2 + \beta||\mathbb{M}\mathcal{U}||_F^2 + \gamma||\mathbb{B}\mathcal{U} - \mathbb{Q}||_F^2.$$

α, β and γ refer to the penalties associated with the three constraints. Also note that N refers to the maximum number of spatial dimensions, while (N+1)th is the temporal dimension. Minimization is to be done w.r.t. both the spatial and the temporal dimensions i.e.

$$\frac{\partial R}{\partial \mathbf{u}_d^k} = \mathbf{0}, \qquad \frac{\partial R}{\partial \boldsymbol{\tau}^k} = \mathbf{0}$$

Since the main term (one containing the tensorized FPO) and ones pertaining to specific constraints show up additively, therefore each one of them can be dealt separately.

5.A FPO

We start with the first term containing the tensorized FPO,

$$\mathbb{A}'\mathcal{U} = \sum_{i_A=1}^{R_A} \sum_{i_u=1}^{R_U} \left(\bigotimes_{d=1}^{N} \mathbf{A}_d^{i_A} \mathbf{u}_d^{i_u} \right) \mathbf{A}_t^{i_A} \boldsymbol{\tau}^{i_u}$$

Building on the concepts presented in the section 5.4, it follows,

$$||\mathbb{A}'\mathcal{U}||_F^2 = \langle \mathbb{A}'\mathcal{U}, \mathbb{A}'\mathcal{U} \rangle = \sum_{i_A=1}^{R_A} \sum_{j_A=1}^{R_A} \sum_{i_u=1}^{R_U} \sum_{j_u=1}^{R_U} \left(\prod_{d=1}^{N} \langle \mathbf{A}_d^{i_A} \mathbf{u}_d^{i_u}, \mathbf{A}_d^{j_A} \mathbf{u}_d^{j_u} \rangle \right) \langle \mathbf{A}_t^{i_A} \boldsymbol{\tau}^{i_u}, \mathbf{A}_t^{j_A} \boldsymbol{\tau}^{j_u} \rangle$$

Taking derivative w.r.t. the spatial loading vector $\mathbf{u}_k^r$ for $k = 1, \cdots, N$, and putting the terms to zero leads to the following equation,

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{u}_k^r} < \mathbb{A}'\mathcal{U}, \mathbb{A}'\mathcal{U} > &= \sum_{i_A=1}^{R_A}\sum_{j_A=1}^{R_A}\sum_{j_u=1}^{R_U}\left(\mathbf{A}_k^{j_A}\mathbf{u}_k^{j_u}\right)^T\mathbf{A}_k^{i_A}\left(\prod_{\substack{d=1\\d\neq k}}^{N}\langle\mathbf{A}_d^{i_A}\mathbf{u}_d^{i_u},\mathbf{A}_d^{j_A}\mathbf{u}_d^{j_u}\rangle\right)\\
&\times \langle\mathbf{A}_t^{i_A}\boldsymbol{\tau}^{i_u},\mathbf{A}_t^{j_A}\boldsymbol{\tau}^{j_u}\rangle\\
&+ \sum_{i_A=1}^{R_A}\sum_{i_u=1}^{R_U}\sum_{j_A=1}^{R_A}\left(\mathbf{A}_k^{i_A}\mathbf{u}_k^{i_u}\right)^T\mathbf{A}_k^{j_A}\left(\prod_{\substack{d=1\\d\neq k}}^{N}\langle\mathbf{A}_d^{i_A}\mathbf{u}_d^{i_u},\mathbf{A}_d^{j_A}\mathbf{u}_d^{j_u}\rangle\right)\\
&\times \langle\mathbf{A}_t^{i_A}\boldsymbol{\tau}^{i_u},\mathbf{A}_t^{j_A}\boldsymbol{\tau}^{j_u}\rangle = \mathbf{0}
\end{aligned}$$

which when put in the matrix form looks like

$$\begin{bmatrix} (\mathbf{M})_{1,1} & \cdots & (\mathbf{M})_{1,R_U}\\ \vdots & \ddots & \cdots\\ (\mathbf{M})_{R_U,1} & \cdots & (\mathbf{M})_{R_U,R_U}\end{bmatrix}\begin{bmatrix}\mathbf{u}_d^1\\ \vdots\\ \mathbf{u}_d^{R_U}\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \vdots\\ \mathbf{0}\end{bmatrix}$$

where the sub-matrix $(\mathbf{M})_{i,j}$ is defined as,

$$(\mathbf{M})_{i,j} = \sum_{i_A=1}^{R_A}\sum_{j_A=1}^{R_A}(\mathbf{A}_k^{j_A})^T\mathbf{A}_k^{i_A}\left(\prod_{\substack{d=1\\d\neq k}}^{N} < \mathbf{A}_d^{i_A}\mathbf{u}_d^j, \mathbf{A}_d^{j_A}\mathbf{u}_d^i >\right)\langle\mathbf{A}_t^{i_A}\boldsymbol{\tau}^i,\mathbf{A}_t^{j_A}\boldsymbol{\tau}^j\rangle$$

The same procedure when applied w.r.t. the spatial dimension $\boldsymbol{\tau}^r$ yields,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\tau}_k^r} < \mathbb{A}'\mathcal{U}, \mathbb{A}'\mathcal{U} > &= \sum_{i_A=1}^{R_A}\sum_{j_A=1}^{R_A}\sum_{j_u=1}^{R_U}\left(\mathbf{A}_t^{j_A}\boldsymbol{\tau}^{j_u}\right)^T\mathbf{A}_t^{i_A}\left(\prod_{d=1}^{N}\langle\mathbf{A}_d^{i_A}\mathbf{u}_d^{i_u},\mathbf{A}_d^{j_A}\mathbf{u}_d^{j_u}\rangle\right)\\
&+ \sum_{i_A=1}^{R_A}\sum_{i_u=1}^{R_U}\sum_{j_A=1}^{R_A}\left(\mathbf{A}_t^{i_A}\boldsymbol{\tau}^{i_u}\right)^T\mathbf{A}_t^{j_A}\left(\prod_{d=1}^{N}\langle\mathbf{A}_d^{i_A}\mathbf{u}_d^{i_u},\mathbf{A}_d^{j_A}\mathbf{u}_d^{j_u}\rangle\right) = \mathbf{0}
\end{aligned}$$

with the corresponding sub-matrices given by,

$$(\mathbf{M})_{i,j} = \sum_{i_A=1}^{R_A}\sum_{j_A=1}^{R_A}(\mathbf{A}_t^{j_A})^T\mathbf{A}_t^{i_A}\left(\prod_{d=1}^{N} < \mathbf{A}_d^{i_A}\mathbf{u}_d^j, \mathbf{A}_d^{j_A}\mathbf{u}_d^i >\right)$$

5.B Normality constraint term

Now turning to the normality constraint term, we can write

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{u}_k^r}||\mathbb{B}\mathcal{U} - \mathbb{Q}||_F^2 &= \frac{\partial}{\partial \mathbf{u}_k^r}\left(\langle\mathbb{B}\mathcal{U},\mathbb{B}\mathcal{U}\rangle - 2\langle\mathbb{B}\mathcal{U},\mathbb{Q}\rangle + \langle\mathbb{Q},\mathbb{Q}\rangle\right)\\
&= \frac{\partial}{\partial \mathbf{u}_k^r}\langle\mathbb{B}\mathcal{U},\mathbb{B}\mathcal{U}\rangle - 2\frac{\partial}{\partial \mathbf{u}_k^r}\langle\mathbb{B}\mathcal{U},\mathbb{Q}\rangle = \mathbf{0}
\end{aligned}$$

Now, as per the previous sections we have

$$\mathbb{BU} = \sum_{i_u=1}^{R_U} \left(\bigotimes_{d=1}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \right) \otimes \boldsymbol{\tau}^{i_u}$$

which further leads to,

$$\langle \mathbb{BU}, \mathbb{BU} \rangle = \sum_{i_u=1}^{R_U} \sum_{j_u=1}^{R_U} \left(\prod_{d=1}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \langle \mathbf{u}_d^{j_u}, \mathbf{b}_d \rangle \right) \langle \boldsymbol{\tau}^{i_u}, \boldsymbol{\tau}^{j_u} \rangle$$

$$\langle \mathbb{BU}, \mathbb{Q} \rangle = \sum_{i_u=1}^{R_U} \left(\prod_{d=1}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \right) \langle \boldsymbol{\tau}^{i_u}, \mathbb{1} \rangle$$

The two derivatives are given as,

$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}_k^r} \langle \mathbb{BU}, \mathbb{BU} \rangle &= \sum_{j_u=1}^{R_U} \mathbf{B}_k \mathbf{u}_k^{j_u} \left(\prod_{\substack{d=1 \\ d \neq k}}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \langle \mathbf{u}_d^{j_u}, \mathbf{b}_d \rangle \right) \langle \boldsymbol{\tau}^{i_u}, \boldsymbol{\tau}^{j_u} \rangle \\ &+ \sum_{i_u=1}^{R_U} \mathbf{B}_k \mathbf{u}_k^{i_u} \left(\prod_{\substack{d=1 \\ d \neq k}}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \langle \mathbf{u}_d^{j_u}, \mathbf{b}_d \rangle \right) \langle \boldsymbol{\tau}^{i_u}, \boldsymbol{\tau}^{j_u} \rangle \end{aligned}$$

and

$$\frac{\partial}{\partial \mathbf{u}_k^r} \langle \mathbb{BU}, \mathbb{Q} \rangle = \mathbf{b}_k \left(\prod_{\substack{d=1 \\ d \neq k}}^{N} \langle \mathbf{u}_d^{i_u}, \mathbf{b}_d \rangle \right) \langle \boldsymbol{\tau}^{i_u}, \mathbb{1} \rangle$$

The whole normality constraint in the matrix form is given by,

$$\begin{bmatrix} (\mathbf{M}_N)_{1,1} & \cdots & (\mathbf{M}_N)_{1,R_U} \\ \vdots & \ddots & \cdots \\ (\mathbf{M}_N)_{R_U,1} & \cdots & (\mathbf{M}_N)_{R_U,R_U} \end{bmatrix} \begin{bmatrix} \mathbf{u}_d^1 \\ \vdots \\ \mathbf{u}_d^{R_U} \end{bmatrix} = \begin{bmatrix} (\mathbf{v}_N)_1 \\ \vdots \\ (\mathbf{v}_N)_{R_U} \end{bmatrix}$$

where,

$$(\mathbf{M}_N)_{i,j} = \mathbf{B}_k \left(\prod_{\substack{d=1 \\ d \neq k}}^{N} \langle \mathbf{u}_d^i, \mathbf{b}_d \rangle \langle \mathbf{b}_d, \mathbf{u}_d^j \rangle \right) \langle \boldsymbol{\tau}^i, \boldsymbol{\tau}^j \rangle$$

$$(\mathbf{v}_N)_i = b_k \left(\prod_{\substack{d=1 \\ d \neq k}}^{N} \langle u_d^i b_d \rangle \right) \langle \boldsymbol{\tau}^i \boldsymbol{\tau}^j \rangle$$

Similarly, for derivative w.r.t. the temporal basis factor $\boldsymbol{\tau}^r$, we have the terms

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\tau}^r}\langle \mathbb{B}\mathcal{U}, \mathbb{B}\mathcal{U}\rangle &= \sum_{j_u=1}^{R_U}\left(\prod_{d=1}^{N}\langle \mathbf{u}_d^{i_u}, \mathbf{b}_d\rangle\langle \mathbf{u}_d^{j_u}, \mathbf{b}_d\rangle\right)(\boldsymbol{\tau}^{j_u})^T \\ &+ \sum_{i_u=1}^{R_U}\left(\prod_{d=1}^{N}\langle \mathbf{u}_d^{i_u}, \mathbf{b}_d\rangle\langle \mathbf{u}_d^{j_u}, \mathbf{b}_d\rangle\right)(\boldsymbol{\tau}^{i_u})^T\end{aligned}$$

and

$$\frac{\partial}{\partial \boldsymbol{\tau}^r}\langle \mathbb{B}\mathcal{U}, \mathbb{Q}\rangle = \left(\prod_{d=1}^{N}\langle \mathbf{u}_d^{i_u}, \mathbf{b}_d\rangle\right)\mathbb{1}$$

For the temporal derivative, the sub-matrices of the matrix equation are given by

$$(\mathbf{M}_N)_{i,j} = \mathbf{I}_k\left(\prod_{d=1}^{N}\langle \mathbf{u}_d^{i}, \mathbf{b}_d\rangle\langle \mathbf{b}_d, \mathbf{u}_d^{j}\rangle\right)$$

and the sub-vectors,

$$(\mathbf{v}_N)_i = \mathbb{1}\left(\prod_{d=1}^{N}\langle \mathbf{u}_d^{i}, \mathbf{b}_d\rangle\right)$$

where $\mathbb{1} = [1, 1, \cdots, 1]^T$ and $\mathbf{B}_k = \mathbf{b}_k \times \mathbf{b}_k^T$.

5.C Initial value constraint term

We give the expressions for the sub-matrices and the sub-vector without going into the whole derivation. Please note that for the initial value constraint, we have the initial value tensor $\mathcal{U}_0$ given by

$$\mathbb{U}_0 = \sum_{i_u=1}^{R_{U_0}}\left[\left(\bigotimes_{d=1}^{N} u_{0,d}^{i_u}\right)\otimes[1]\right]$$

and corresponding projection tensor $\mathcal{N}$ is given by,

$$\mathbb{N} = \left(\bigotimes_{d=1}^{N} I_{\mathbf{x}_d}^{i}\right)\otimes e$$

and e is given by the row vector $[1, 0, \cdots, 0]$. Given this, we can derive the following expressions for the spatial dimensions,

$$(\mathbf{M}_I)_{i,j} = \mathbf{I}_k\left(\prod_{\substack{d=1\\d\neq k}}^{N}\langle \mathbf{u}_d^{i}, \mathbf{u}_d^{j}\rangle\right)\langle \tau^i, \tau^j\rangle$$

$$(\mathbf{v}_I)_i = \sum_{j=1}^{RU_0}\mathbf{u}_{0,d}^{j}\left(\prod_{\substack{d=1\\d\neq k}}^{N}\langle \mathbf{u}_d^{i}, \mathbf{u}_{0,d}^{j}\rangle\right)\langle \boldsymbol{\tau}^i, \mathbf{e}\rangle$$

Similar terms for the temporal dimension are also given below,

$$(\mathbf{M}_I)_{i,j} = \mathbf{E}\left(\prod_{d=1}^{N}\langle \mathbf{u}_d^i, \mathbf{u}_d^j\rangle\right)$$

$$(\mathbf{v}_I)_i = \sum_{j=1}^{RU_0}\mathbf{e}\left(\prod_{d=1}^{k}\langle \mathbf{u}_d^i, \mathbf{u}_{0,d}^j\rangle\right)$$

where $\mathbf{I}_k$ is the identity matrix of appropriate dimensions and $\mathbf{E} = \mathbf{e} \times \mathbf{e}^T$.

5.D Boundary value constraint term

Given the boundary value tensor,

$$\mathbb{M} = \sum_{i_M=1}^{N}\left[\left(\bigotimes_{d=1}^{i_M-1} I'_{\mathbf{x}_d}\right)\otimes I''_{\mathbf{x}_{i_M}}\left(\bigotimes_{d=i_M+1}^{N} I_{\mathbf{x}_d}\right)\otimes I_t\right]$$

we can derive the corresponding sub-matrices. First, for the spatial dimensions,

$$(\mathbf{M}_B)_{i,j} = \sum_{r=1}^{N}\mathbf{I}_t\left(\prod_{d=1}^{N}\langle(\mathbf{J})_{r,d}\mathbf{u}_d^i, (\mathbf{J})_{r,d}\mathbf{u}_d^j\rangle\langle\boldsymbol{\tau}^i, \boldsymbol{\tau}^j\rangle\right)$$

while for the temporal dimension we have,

$$(\mathbf{M}_B)_{i,j} = \sum_{r=1}^{N}\mathbf{I}_t\left(\prod_{d=1}^{N}\langle(\mathbf{J})_{r,d}\mathbf{u}_d^i, (\mathbf{J})_{r,d}\mathbf{u}_d^j\rangle\langle\boldsymbol{\tau}^i, \boldsymbol{\tau}^j\rangle\right)$$

where $\mathbf{I}_t$ is the identity matrix corresponding to the time dimension and $\mathbf{J} \in \mathbb{R}^{\sum_{d=1}^{N} n_d \times \sum_{d=1}^{N} n_d}$ given by,

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}''_{\mathbf{x}_1} & \mathbf{I}_{\mathbf{x}_2} & \mathbf{I}_{\mathbf{x}_3} & \cdots & \mathbf{I}_{\mathbf{x}_N} \\ \mathbf{I}'_{\mathbf{x}_1} & \mathbf{I}''_{\mathbf{x}_2} & \mathbf{I}_{\mathbf{x}_3} & \cdots & \mathbf{I}_{\mathbf{x}_N} \\ \mathbf{I}'_{\mathbf{x}_1} & \mathbf{I}'_{\mathbf{x}_2} & \mathbf{I}''_{\mathbf{x}_3} & \cdots & \mathbf{I}_{\mathbf{x}_N} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{I}'_{\mathbf{x}_1} & \mathbf{I}'_{\mathbf{x}_2} & \mathbf{I}'_{\mathbf{x}_3} & \cdots & \mathbf{I}''_{\mathbf{x}_N} \end{bmatrix}$$

The sub-matrices constituting the block matrix $\mathbf{J}$ have been described in the section 5.6. Finally, all terms can be put together in the following equation,

$$[\mathbf{M} + \alpha\mathbf{M}_I + \beta\mathbf{M}_B + \gamma\mathbf{M}_N]\,\overrightarrow{\mathbf{u}_k} = \alpha\mathbf{v}_I + \gamma\mathbf{v}_N$$

where $\mathbf{M}$, $\mathbf{M}_I$, $\mathbf{M}_B$ and $\mathbf{M}_N$ are all block matrices and $\overrightarrow{\mathbf{u}_k}$ is the vectorized factor matrix $\mathbf{U}_k$.

Chapter 6

Flow solution for the sum of Gaussians based prior densities

We have discussed, implemented and analyzed several of the log-homotopy based particle flows in the previous chapters. Among all flow solutions, the so-called *exact flow* is of particular interest. The reason is that it has a closed form solution that is quite elegant and simple to implement. It is based on the Gaussian assumption for the prior density and the likelihood, which together with the assumption of zero diffusion term in the SDE, leads to a closed form analytical flow solution. This flow has been subject of many studies e.g. [KU14], [BS14], [DGYM$^+$15] etc. The Gaussian assumption for the prior density is a rather strong one. In particular, the prior density for highly nonlinear process/measurement models, or models with non-Gaussian noises can exhibit multi-modality, and hence the *exact flow* may not be suitable. In this chapter, we consider a more general scenario where the prior density may not be represented accurately by a single multivariate Gaussian. Therefore, in order to cater for the non-Gaussianity of the prior, we use a Gaussian mixture model (GMM). We solve the corresponding FPE for the unknown flow and derive analytical flow solutions. Finally, we implement our new flows and show that a filter based on one of the new flows outperforms the *exact flow* and the particle filter.

The outline of the chapter is given as follows: Section 6.1 contains the derivation of flow equations based on the Gaussian mixture assumption for the prior. Implementation methodology for our new flows is described in the section 6.2. Numerical simulation results are mentioned in the section 6.3, which is followed by the conclusion in the section 6.4.

6.1 Derivation of Gaussian Mixture Flow

If the diffusion term is assumed to be zero but the flow is allowed to be compressible, the following equation can be derived from (3.16),

$$\log h(\mathbf{x}) + \nabla \log p(\mathbf{x}, \lambda)^T \cdot \mathbf{f}(\mathbf{x}, \lambda) = -\nabla \cdot \mathbf{f}(\mathbf{x}, \lambda) \tag{6.1}$$

$\frac{\partial \log K(\lambda)}{\partial \lambda}$ represents the logarithmic change in the normalization constant. For a given value of λ, this term is constant. Hence, it is ignored in the subsequent analysis. One particular solution, termed as the *exact flow*, relates to the case of $\log g(\mathbf{x})$ and $\log h(\mathbf{x})$ being bilinear in the components of vector $\mathbf{x}$, e.g., assuming a Gaussian prior and likelihood.

$$\log g(\mathbf{x}) = \log c_P - \frac{1}{2} (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{P}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \tag{6.2}$$

$$\log h(\mathbf{x}) = \log c_R - \frac{1}{2} (\mathbf{z} - \psi(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{z} - \psi(\mathbf{x})) \tag{6.3}$$

where c_P and c_R are the normalization constants associated with the prior and the likelihood. The *exact flow* equation is then given by,

$$\mathbf{f}(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda) \tag{6.4}$$

with,

$$\mathbf{A}(\lambda) = -\frac{1}{2}\mathbf{P}\mathbf{H}^T(\lambda\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H} \tag{6.5}$$

$$\mathbf{b}(\lambda) = (I + 2\lambda\mathbf{A})[(I + \lambda\mathbf{A})\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} + \mathbf{A}\bar{\mathbf{x}}] \tag{6.6}$$

Here $\mathbf{P}$ refers to the prior covariance matrix, $\bar{\mathbf{x}}$ is the prior mean vector, and $\mathbf{H}$ is the Jacobian of the measurement function $\psi(\mathbf{x})$. For more details on the implementation and analysis of this type of flow, please refer to [KU14], [KU15] and [DC12]. In this work, we relax the Gaussian assumption for the prior density. Instead, we assume that the prior density cannot be modeled sufficiently well by a single Gaussian, and is rather approximated by a sum of Gaussian with M components i.e.

$$g(\mathbf{x}) = \sum_{i=1}^{M} \theta_i \mathcal{N}(\mathbf{x}|\mu_i, \mathbf{P}_i) \tag{6.7}$$

where θ_i, μ_i and $\mathbf{P}_i$ are the weight, mean and the covariance matrices of the *ith* component. The gradient of the log of the prior density then can be written as,

$$\nabla \log g(\mathbf{x}, \lambda) = \sum_{i=1}^{M} \alpha_i(\mathbf{x}) \mathbf{P}_i^{-1} (\mathbf{x} - \mu_i) \tag{6.8}$$

with α_i defined as,

$$\alpha_i(\mathbf{x}) = \frac{-\theta_i \mathcal{N}(\mathbf{x}|\mu_i, \mathbf{P}_i)}{\sum_{j=1}^{M} \theta_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{P}_j)} \tag{6.9}$$

The likelihood $\log h(\mathbf{x})$, on the other hand, is represented by a single component Gaussian. Its gradient is defined as,

$$\nabla \log h(\mathbf{x}, \lambda) \quad = \quad \mathbf{H}^T \mathbf{R}^{-1} \left(\mathbf{z} - \psi(\mathbf{x})\right) \tag{6.10}$$

Again we assume that the flow equation can be expressed in a linear form like in (6.4),

$$\mathbf{f}(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda) \tag{6.11}$$

The matrix $\mathbf{A}(\lambda)$ and the vector $\mathbf{b}(\lambda)$ are unknowns, and our task is to find analytical expressions for them. For this choice of flow, the divergence becomes $\nabla \cdot \mathbf{f}(\mathbf{x}, \lambda) = \text{Tr}(\mathbf{A}(\lambda))$. For the sake of brevity we drop λ from the arguments of both $\mathbf{A}$ and $\mathbf{b}$. Now, we refer back to (6.1) and plug in the values,

$$\left(\sum_{i=1}^{M} \alpha_i(\mathbf{x}) \mathbf{P}_i^{-1} \left(\mathbf{x} - \mu_i\right) + \lambda \mathbf{H}^T \mathbf{R}^{-1} \left(\mathbf{z} - \psi(\mathbf{x})\right)\right)^T (\mathbf{A}\mathbf{x} + \mathbf{b}) + \log c_R - \frac{1}{2} \left(\mathbf{z} - \psi(\mathbf{x})\right)^T \mathbf{R}^{-1} \left(\mathbf{z} - \psi(\mathbf{x})\right) = \text{Tr}(\mathbf{A}) \tag{6.12}$$

The measurement model can be linearized by the Taylor series expansion up to the first term about the point $\mathbf{x}_\lambda$, such that $\bar{\mathbf{z}} \approx \mathbf{z} - \psi(\mathbf{x}_\lambda) + \mathbf{H}\mathbf{x}_\lambda$, where $\mathbf{H} = \frac{\partial \psi}{\partial \mathbf{x}}\big|_{\mathbf{x}_\lambda}$. Linearization of the measurement model leads to the following expansion of the (6.12),

$$\begin{aligned} \sum_{i=1}^{M} \alpha_i(\mathbf{x}) \left(\mathbf{x} - \mu_i\right)^T \mathbf{P}_i^{-1} \mathbf{A}\mathbf{x} + \sum_{i=1}^{M} \alpha_i(\mathbf{x}) \left(\mathbf{x} - \mu_i\right)^T \mathbf{P}_i^{-1} \mathbf{b} \\ + \lambda (\bar{\mathbf{z}} - \mathbf{H}\mathbf{x})^T \mathbf{R}^{-1} \mathbf{H}\mathbf{A}\mathbf{x} + \lambda (\bar{\mathbf{z}} - \mathbf{H}\mathbf{x})^T \mathbf{R}^{-1} \mathbf{H}\mathbf{b} \\ + \log c_R - \frac{1}{2} \bar{\mathbf{z}} \mathbf{R}^{-1} \bar{\mathbf{z}} + \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{H}\mathbf{x} - \frac{1}{2} \mathbf{x}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\mathbf{x} = -\text{Tr}(\mathbf{A}) \end{aligned} \tag{6.13}$$

$\alpha_i(\mathbf{x})$ are the only nonlinear factors in the (6.13). If they can be approximated by a linear or a quadratic term, the resulting equation can be expressed as a polynomial in $\mathbf{x}$. Therefore, we expand the $\alpha_i(\mathbf{x})$ via Taylor series up to the first term about some point $\tilde{\mathbf{x}}$. Hence, $\alpha_i(\mathbf{x}) \approx \alpha_i(\tilde{\mathbf{x}}) + (\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{a}_i$ where $\mathbf{a}_i = \nabla \alpha_i(\mathbf{x})|_{\tilde{\mathbf{x}}}$, which is given by,

$$\mathbf{a}_i(\mathbf{x}) = -\alpha_i(\tilde{\mathbf{x}}) \left(\mathbf{P}_i^{-1} \left(\mathbf{x} - \mu_i\right) + \sum_{j=1}^{M} \alpha_j(\tilde{\mathbf{x}}) \mathbf{P}_j^{-1} \left(\mathbf{x} - \mu_j\right)\right) \tag{6.14}$$

For conciseness, we drop the $\tilde{\mathbf{x}}$ from the arguments of α and $\mathbf{a}$. With the linearization of $\alpha_i(\mathbf{x})$ at hand, we can open the summations in the (6.13). The first term then becomes,

$$\begin{aligned} \sum_{i=1}^{M} \left(\alpha_i + (\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{a}_i\right) \left(\mathbf{x} - \mu_i\right)^T \mathbf{P}_i^{-1} \mathbf{A}\mathbf{x} = \mathbf{x}^T \left(\sum_{i=1}^{M} \alpha_i \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} - \left(\sum_{i=1}^{M} \alpha_i \mu_i^T \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} \\ + \mathbf{x}^T \left(\sum_{i=1}^{M} \mathbf{a}_i \mathbf{x}^T \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} - \mathbf{x}^T \left(\sum_{i=1}^{M} \mathbf{a}_i \mu_i^T \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} - \mathbf{x}^T \left(\sum_{i=1}^{M} \tilde{\mathbf{x}}^T \mathbf{a}_i \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} \\ + \left(\sum_{i=1}^{M} \tilde{\mathbf{x}}^T \mathbf{a}_i \mu_i^T \mathbf{P}_i^{-1} \mathbf{A}\right) \mathbf{x} \end{aligned} \tag{6.15}$$

Likewise the second term can be expanded,

$$\sum_{i=1}^{M}\left(\alpha_i(\tilde{\mathbf{x}})+(\mathbf{x}-\tilde{\mathbf{x}})^T\mathbf{a}_i\right)(\mathbf{x}-\mu_i)^T\mathbf{P}_i^{-1}\mathbf{b}=\mathbf{x}^T\left(\sum_{i=1}^{M}\mathbf{a}_i\mathbf{b}^T\mathbf{P}_i^{-1}\right)\mathbf{x}+\left(\sum_{i=1}^{M}\alpha_i\mathbf{b}^T\mathbf{P}_i^{-1}\right)\mathbf{x}$$
$$-\left(\sum_{i=1}^{M}\mathbf{b}\mathbf{P}_i^{-1}\mu_i\mathbf{a}_i^T\right)\mathbf{x}-\left(\sum_{i=1}^{M}\tilde{\mathbf{x}}^T\mathbf{a}_i\mathbf{b}^T\mathbf{P}_i^{-1}\right)\mathbf{x}-\left(\sum_{i=1}^{M}\alpha_i\mu_i^T\mathbf{P}_i^{-1}\mathbf{b}\right)+\left(\sum_{i=1}^{M}\tilde{\mathbf{x}}^T\mathbf{a}_i\mu_i^T\mathbf{b}^T\mathbf{P}_i^{-1}\mathbf{b}\right) \tag{6.16}$$

Now we combine these two parts,

$$\mathbf{x}^T\Theta(\mathbf{x})\mathbf{x}+\mathbf{x}^T\Lambda\mathbf{x}+\beta^T\mathbf{x}+c \tag{6.17}$$

where,

$$\begin{aligned}
\Theta(\mathbf{x}) &= \sum_{i=1}^{M}\mathbf{a}_i\mathbf{x}^T\mathbf{P}_i^{-1}\mathbf{A}\\
\Lambda &= \sum_{i=1}^{M}\left[\left(\alpha_i\mathbf{I}-\mathbf{a}_i\mu_i^T-\tilde{\mathbf{x}}^T\mathbf{a}_i\mathbf{I}\right)\mathbf{A}+\mathbf{a}_i\mathbf{b}^T\right]\mathbf{P}_i^{-1}\\
\beta^T &= \sum_{i=1}^{M}\left(-\alpha_i\mu_i^T+\tilde{\mathbf{x}}^T\mathbf{a}_i\mu_i^T\right)\mathbf{P}_i^{-1}\mathbf{A}+\sum_{i=1}^{M}\left(\alpha_i\mathbf{b}^T-\mathbf{b}^T\mathbf{P}_i^{-1}\mu\mathbf{a}_i^T\mathbf{P}_i-\mathbf{a}_i^T\tilde{\mathbf{x}}\mathbf{b}^T\right)\mathbf{P}_i^{-1}\\
c &= \sum_{i=1}^{M}\left(\mathbf{a}_i^T\tilde{\mathbf{x}}-\alpha_i\right)\mu_i^T\mathbf{P}_i^{-1}\mathbf{b}
\end{aligned}$$

The remaining terms on the LHS of the (6.13) can be condensed into a similar form given as follows,

$$\mathbf{x}^T\Pi\mathbf{x}+\gamma^T\mathbf{x}+d \tag{6.18}$$

where,

$$\begin{aligned}
\Pi &= -\lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A}-\frac{1}{2}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\\
\gamma^T &= \lambda\bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{A}-\lambda\mathbf{b}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}+\bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H}\\
d &= \log c_R+\lambda\bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{b}-\frac{1}{2}\bar{\mathbf{z}}^T\mathbf{R}^{-1}\bar{\mathbf{z}}
\end{aligned}$$

Finally, (6.13) can be expressed as,

$$\mathbf{x}^T\Theta(\mathbf{x})\mathbf{x}+\mathbf{x}^T\Upsilon\mathbf{x}+\delta^T\mathbf{x}+e=0 \tag{6.19}$$

where,

$$\begin{aligned}
\Upsilon &= \Lambda+\Pi\\
\delta^T &= \beta^T+\gamma^T\\
e &= c+d+Tr(\mathbf{A})
\end{aligned}$$

The next step is to set the coefficients of the monomials (cubic, quadratic and linear) terms to zero. This can be justified as (6.19) must hold for all values of $\mathbf{x}$, which can be ensured by setting the coefficients to zeros. Now we have two choices here to start with.

6.1.1 Ignoring $\Theta(\mathbf{x})$

In the first case, we ignore the cubic term and just consider the quadratic and the linear terms. Then, Υ can be written as,

$$\Upsilon = \mathbf{QA} + \sum_{i=1}^{M} \mathbf{a}_i \mathbf{b}^T \mathbf{P}_i^{-1} - \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{HA} - \frac{1}{2} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \tag{6.20}$$

where,

$$\mathbf{Q}_i = \left(\alpha_i \mathbf{I} - \mathbf{a}_i \mu_i^T - \tilde{\mathbf{x}}^T \mathbf{a}_i \mathbf{I} \right) \mathbf{P}_i^{-1} \qquad \mathbf{Q} = \sum_{i=1}^{M} \mathbf{Q}_i$$

Setting Υ to zero leads to

$$\mathbf{JA} = \mathbf{K} - \sum_{i=1}^{M} \mathbf{a}_i \mathbf{b}^T \mathbf{P}_i^{-1} \tag{6.21}$$

with,

$$\mathbf{J} = \mathbf{Q} - 2\lambda \mathbf{K} \qquad , \qquad \mathbf{K} = \frac{1}{2} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}$$

Next, the same procedure is applied to δ,

$$\begin{aligned}
\delta^T = & \sum_{i=1}^{M} \left(-\alpha_i \mu_i^T + \tilde{\mathbf{x}}^T \mathbf{a}_i \mu_i^T \right) \mathbf{P}_i^{-1} \mathbf{A} \\
& + \sum_{i=1}^{M} \left(\alpha_i \mathbf{b}^T - \mathbf{b}^T \mathbf{P}_i^{-1} \mu_i \mathbf{a}_i^T \mathbf{P}_i - \mathbf{a}_i^T \tilde{\mathbf{x}} \mathbf{b}^T \right) \mathbf{P}_i^{-1} \\
& + \lambda \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{HA} - \lambda \mathbf{b}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} + \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{H} \\
= & \sum_{i=1}^{M} \mathbf{s}_i^T \mathbf{A} + \sum_{i=1}^{M} \mathbf{b}^T \mathbf{U}_i + \lambda \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{HA} - \lambda \mathbf{b}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \\
& + \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{H} \\
= & \mathbf{s}^T \mathbf{A} + \mathbf{b}^T \mathbf{U} + \lambda \bar{\mathbf{z}} \mathbf{R}^{-1} \mathbf{HA} - \lambda \mathbf{b}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \\
& + \bar{\mathbf{z}} \mathbf{R}^{-1} \mathbf{H} \\
= & \left(\mathbf{s}^T + \lambda \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{H} \right) \mathbf{A} + \mathbf{b}^T \left(\mathbf{U} - \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \right) \\
& + \bar{\mathbf{z}}^T \mathbf{R}^{-1} \mathbf{H}
\end{aligned} \tag{6.22}$$

where,

$$\mathbf{s}_i^T = \left(-\alpha_i \mu_i^T + \tilde{\mathbf{x}}^T \mathbf{a}_i \mu_i^T \right) \mathbf{P}_i^{-1} \qquad \mathbf{s}^T = \sum_{i=1}^{M} \mathbf{s}_i^T$$

$$\mathbf{U}_i = \left(\alpha_i \mathbf{I} - \mathbf{P}_i^{-1} \mu_i \mathbf{a}_i^T \mathbf{P}_i - \mathbf{a}_i^T \tilde{\mathbf{x}} \mathbf{I} \right) \mathbf{P}_i^{-1} \qquad \mathbf{U} = \sum_{i=1}^{M} \mathbf{U}_i$$

This yields,

$$\delta^T = \mathbf{f}^T\mathbf{A} + \mathbf{b}^T\mathbf{G} + \mathbf{m}^T \tag{6.23}$$

with,

$$\begin{aligned}\mathbf{f}^T &= \mathbf{s}^T + \lambda\mathbf{m}^T \\ \mathbf{G} &= \mathbf{U} - 2\lambda\mathbf{K} \\ \mathbf{m}^T &= \bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H}\end{aligned}$$

By setting δ^T equal to zero in (6.23), $\mathbf{b}^T$ can be written in terms of $\mathbf{A}$,

$$\boxed{\mathbf{b}^T = -\left(\mathbf{f}^T\mathbf{A} + \mathbf{m}^T\right)\mathbf{G}^{-1}} \tag{6.24}$$

By inserting (6.21) into (6.24), we get

$$\begin{aligned}\mathbf{J}\mathbf{A} &= \mathbf{K} + \sum_{i=1}^{M}\mathbf{a}_i\left(\mathbf{f}^T\mathbf{A} + \mathbf{m}^T\right)\mathbf{G}^{-1}\mathbf{P}_i^{-1} \\ &= \mathbf{K} + \sum_{i=1}^{M}\mathbf{a}_i\mathbf{f}^T\mathbf{A}\mathbf{G}^{-1}\mathbf{P}_i^{-1} + \sum_{i=1}^{M}\mathbf{a}_i\mathbf{m}^T\mathbf{G}^{-1}\mathbf{P}_i^{-1}\end{aligned} \tag{6.25}$$

Now using the vector identity,

$$\text{vec}\left(\mathbf{XYZ}\right) = \left(\mathbf{Z}^T \otimes \mathbf{X}\right)\text{vec}(\mathbf{Y}) \tag{6.26}$$

we can vectorize the equation(6.25) as below,

$$\begin{aligned}(\mathbf{I}\otimes\mathbf{J})\text{vec}(\mathbf{A}) = \text{vec}(\mathbf{K}) &+ \sum_{i=1}^{M}\left(\left(\mathbf{G}^{-1}\mathbf{P}_i^{-1}\right)^T \otimes \left(\mathbf{a}_i\mathbf{f}^T\right)\right)\text{vec}(\mathbf{A}) \\ &+ \sum_{i=1}^{M}\text{vec}\left(\mathbf{a}_i\mathbf{m}^T\mathbf{G}^{-1}\mathbf{P}_i^{-1}\right)\end{aligned}$$

which leads to,

$$\boxed{\text{vec}(\mathbf{A}) = \mathbf{E}^{-1}\left(\text{vec}(\mathbf{K}) + \sum_{i=1}^{M}\text{vec}\left(\mathbf{a}_i\mathbf{m}^T\mathbf{G}^{-1}\mathbf{P}_i^{-1}\right)\right)} \tag{6.27}$$

where,

$$\mathbf{E} = \left[(\mathbf{I}\otimes\mathbf{J}) - \sum_{i=1}^{M}\left(\left(\mathbf{G}^{-1}\mathbf{P}_i^{-1}\right)^T \otimes \left(\mathbf{a}_i\mathbf{f}^T\right)\right)\right]$$

The matrix $\mathbf{A}$ from (6.27) is in vectorized form. First, it needs to be reshaped back into matrix form. Once done, it can be inserted into (6.23) to get the vector $\mathbf{b}$. This constitutes our first flow equation, termed here as *Gaussian Mixture Particle Flow*-1 or *GMPF*-1.

$$\boxed{\mathbf{f}_{\text{GMPF-1}}(\mathbf{x},\lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda)} \tag{6.28}$$

By ignoring the cubic term, we have derived the flow with the matrix $\mathbf{A}$ and vector $\mathbf{b}$ being independent of the state $\mathbf{x}$, as originally assumed in the (6.4).

6.1.2 Merging $\Theta(\mathbf{x})$ with the Υ

A second flow equation can be derive if the $\Theta(\mathbf{x})$ is not ignored, but it is merged with the quadratic term such that,

$$\Upsilon(\mathbf{x}) = \Lambda + \Pi + \Theta(\mathbf{x}) \tag{6.29}$$

This makes the Υ matrix a function of the state $\mathbf{x}$, which is given as,

$$\Upsilon(\mathbf{x}) = \mathbf{Q}(\mathbf{x})\mathbf{A} + \sum_{i=1}^{M} \mathbf{a}_i \mathbf{b}^T \mathbf{P}_i^{-1} - \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\mathbf{A} - \frac{1}{2}\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \tag{6.30}$$

with

$$\begin{aligned}
\mathbf{Q}_i(\mathbf{x}) &= \left(\mathbf{a}_i \mathbf{x}^T \mathbf{I} + \alpha_i \mathbf{I} - \mathbf{a}_i \mu_i^T - \tilde{\mathbf{x}}^T \mathbf{a}_i \mathbf{I}\right) \mathbf{P}_i^{-1}, \\
\mathbf{Q}(\mathbf{x}) &= \sum_{i=1}^{M} \mathbf{Q}_i(\mathbf{x})
\end{aligned}$$

The rest of the derivation proceeds in the same way as in the previous case. The matrix $\mathbf{A}$ and the vector $\mathbf{b}$ in the resulting flow equation will have spatial dependency, i.e. they depend not only on the pseudo-time λ, but also on the state vector $\mathbf{x}$. We term this flow as the *Gaussian Mixture Particle flow*-2 or *GMPF*-2.

$$\boxed{\mathbf{f}_{\textit{GMPF-2}}(\mathbf{x}, \lambda) = \mathbf{A}(\mathbf{x}, \lambda)\mathbf{x} + \mathbf{b}(\mathbf{x}, \lambda)} \tag{6.31}$$

Checking the correctness of the derivation

The exact flow equation with Gaussian assumption is given by

$$\mathbf{f}_H(\mathbf{x}, \lambda) = \mathbf{A}(\lambda)\mathbf{x} + \mathbf{b}(\lambda)$$

with,

$$\begin{aligned}
\mathbf{A}(\lambda) &= -\frac{1}{2}\mathbf{P}\mathbf{H}^T(\lambda \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H} \\
\mathbf{b}(\lambda) &= (I + 2\lambda\mathbf{A})[(I + \lambda\mathbf{A})\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} + \mathbf{A}\bar{\mathbf{x}}].
\end{aligned} \tag{6.32}$$

Now we check the correctness of the newly derived flow. In the case of the prior being a single Gaussian i.e. $g(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \mathbf{P})$, we will have $\theta = [1, 0, 0]^T$, $\alpha = [-1, 0, 0]^T$ and $\mathbf{a} = [\mathbf{0}, \mathbf{0}, \mathbf{0}]^T$, $\mu = [\bar{\mathbf{x}}, 0, 0]$ and $\mathbf{P} = [\mathbf{P}, \mathbf{0}, \mathbf{0}]$.

$$\begin{aligned}
\mathbf{Q} &= -\mathbf{P}^{-1} \\
\mathbf{J} &= -\left(\mathbf{P}^{-1} + \lambda \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\right) \\
\mathbf{E} &= \mathbf{I} \otimes \mathbf{J}
\end{aligned} \tag{6.33}$$

Therefore,

$$\begin{aligned} \text{vec}(A) &= \mathbf{E}^{-1}\text{vec}(K) \\ (\mathbf{I} \otimes \mathbf{J})\,\text{vec}(A) &= \text{vec}(K) \\ \text{vec}\,(\mathbf{JA}) &= \text{vec}\,(\mathbf{K}) \\ \mathbf{A} &= \mathbf{J}^{-1}\mathbf{K} \end{aligned}$$

leading to,

$$\mathbf{A} = -\frac{1}{2}\left(\mathbf{P}^{-1} + \lambda \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \tag{6.34}$$

which by the application of the matrix inversion lemma, can be written in the same form as in (6.32). Similarly for **b**, we note that

$$\begin{aligned} \mathbf{U} &= -\mathbf{P}^{-1} \\ \mathbf{G} &= -\left(\mathbf{P}^{-1} + \lambda \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right) \\ \mathbf{f}^T &= \bar{\mathbf{x}}^T\mathbf{P}^{-1} + \lambda \bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H} \end{aligned}$$

which leads to,

$$\mathbf{b}^T = \left[\bar{\mathbf{x}}^T\mathbf{P}^{-1}\mathbf{A} + \bar{\mathbf{z}}^T\mathbf{R}^{-1}\mathbf{H}\,(\mathbf{I} + \lambda\mathbf{A})\right]\left[\mathbf{P}^{-1} + \lambda\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]^{-1} \tag{6.35}$$

Next, by taking the transpose of the equation (6.35) and with the assumption that $\mathbf{P}^T\mathbf{A}^T = \mathbf{AP}$, we can write,

$$\mathbf{b} = \left[\mathbf{I} + \lambda\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\left[(\mathbf{I} + \lambda\mathbf{A})\,\mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\bar{\mathbf{z}} + \mathbf{A}\bar{\mathbf{x}}\right]$$

Again the inversion lemma leads to the familiar form. Please note that the assumption about the symmetricity of the matrix product made above is also required for deriving the flow in (6.32), as highlighted in the Appendix 3A.

6.2 Implementation

In order to evaluate the performance of our new flow equations, we look for a scenario where the prior density can be expressed by a GMM. Such a situation may arise in the case when the process noise is non-Gaussian, but can be approximated through a Gaussian mixture, i.e.

$$p(\mathbf{w}_k) = \sum_{i=1}^{N_w} \omega_i \mathcal{N}(\mathbf{w}_k | \nu_i, \mathbf{Q}_i) \tag{6.36}$$

Also, we assume that the posterior density at the time k can be represented as $p(\mathbf{x}_k|\mathbf{Z}_k) = \sum_{j=1}^{M} \theta_{j_k} \mathcal{N}(\mathbf{x}_k | \bar{\mu}_{j_k}, \bar{\mathbf{P}}_{j_k})$. As both the posterior and the transition densities are given by Gaussian mixtures with M and N_w components respectively, the prior density at the time step $k+1$ will consist of $N_w M$ integrals such that,

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_k) = \sum_{i=1}^{N_w} \sum_{j=1}^{M} \omega_i \bar{\theta}_{j_k} \int \mathcal{N}(\mathbf{x}_{k+1} | \mathbf{f}(\mathbf{x}_k) + \nu_i, \mathbf{Q}_i) \times \mathcal{N}(\mathbf{x}_k | \bar{\mu}_{j_k}, \bar{\mathbf{P}}_{j_k}) \mathrm{d}\mathbf{x}_k \tag{6.37}$$

Here we have two issues at hand. First, as the integral is being carried out w.r.t. $\mathbf{x}_k$ which appears as an argument to the nonlinear function $\mathbf{f}$, the individual components of the transition density are no longer Gaussian. This means that the integrals in (6.37) generally can not be evaluated analytically and therefore have to be approximated. Second, the number of terms in the prior density increases geometrically with each time update step.

Algorithm 13 Gaussian Mixture Particle Flow - Time Update

1: **procedure** *GMPF*-TIMEUPDATE($p(\mathbf{x}_k|\mathbf{Z}_k)$)
2: **Main steps**:

1. Sample $\mathcal{N}(\mathbf{x}_k | \bar{\mu}_{j_k}, \bar{\mathbf{P}}_{j_k})$ to obtain $\{\mathbf{x}_k^{l,j}\}_{l=1}^{L}$ $\forall j$.
2. Sample $\mathcal{N}(\mathbf{x}_{k+1} | \mathbf{f}(\mathbf{x}_k^{l,j}) + \nu_i, \mathbf{Q}_i)$. to obtain $\hat{\mathbf{x}}_{k+1}^{l,m}$ $\forall i, j \mid m = (i-1).N_w + j$.
3. Evaluate the mean $\hat{\mu}_{m_{k+1}}$ and the variance $\hat{\mathbf{P}}_{m_{k+1}}$, $\forall m$ using sample based estimates.
4. Get the weight of the components as

$$\hat{\theta}_{m_{k+1}} = \frac{\omega_i \bar{\theta}_{j_k}}{\sum_{i=1}^{N_w} \sum_{j=1}^{M} \omega_i \bar{\theta}_{j_k}} \qquad \forall m.$$

3: **return** $\{\hat{\mathbf{x}}_{k+1}^{l,m}\}_{l=1|m=1}^{L|N_w.M} \sim p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$
4: **end procedure**

Therefore, some mechanism has to be adopted to keep the total number of components fixed or under a certain threshold. Integrals like the ones appearing in (6.37) can be handled in two ways. In the first approach, cubature rules can be used together with the deterministic sampling to perform the time update step e.g. in the Cubature Kalman Filter [AH09]. The other option is to use Monte Carlo sampling to evaluate the integrals. One such method has been described by

Kotecha and Djuric in their paper on Gaussian sum particle filtering [KD03]. In the following, we use the second approach to perform the time update step, which is summarized in Algorithm 13. Here L refers to the number of particles per component.
We see that after performing the time update, the number of components is increased from M to $N_w M$. Once samples from the prior density have been obtained, the measurement update is carried out via log-homotopy flow given by the (6.28) or (6.31). After the particles have been moved to their posterior locations, the next task becomes evaluating the posterior distribution parameters. Again, we follow the method used in the [KD03] for obtaining the posterior GMM parameters. The final step involves reducing the number of components. This can be accomplished in two ways, either by resampling the posterior Gaussian mixture such that only the M strongest components are kept, or by merging the components based on some distance based criterion. The first has been described in [KD03], while we find the second approach advocated in [ALS07]. The distance between two components $\{\theta_i, \mu_i, \mathbf{P}_i\}$ and $\{\theta_j, \mu_j, \mathbf{P}_j\}$ is defined as,

$$d_{i,j} = \frac{\theta_i \theta_j}{\theta_i + \theta_j} (\mu_i - \mu_j) \mathbf{P}^{-1} (\mu_i - \mu_j)^T \tag{6.38}$$

where $\mathbf{P}$ is the overall covariance. The procedure is started by merging the two components having the shortest distance, and is then continued. Merging preserves the overall mean and the covariance of the distribution. Therefore, the posterior density can be represented as,

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) = \sum_{i=1}^{M} \bar{\theta}_{m_{k+1}} \mathcal{N}(\mathbf{x}_k|\bar{\mu}_{m_{k+1}}, \bar{\mathbf{P}}_{m_{k+1}}) \tag{6.39}$$

The measurement update procedure is described in the Algorithm 14.

Algorithm 14 Gaussian Mixture Particle Flow - Measurement Update

1: **procedure** *GMPF*-MEASUREMENTUPDATE($\{\hat{\mathbf{x}}_{k+1}^{l,m}\}_{l=1|m=1}^{L|N_w.M}$)
2: **Main steps**:
1. Integrate flow ODE (6.28) or (6.31) over the pseudo-time $\lambda \in (0, 1]$, to move particles to posterior locations $\bar{\mathbf{x}}_{k+1}^{l,m}$.
2. Update the GMM parameters.
3. Reduce the number of components via re-sampling or merging based on the distance criterion (6.38).
3: **return** $p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1})$
4: **end procedure**

6.3 Numerical Results

In this section, we describe the working our new flow. We divide the description into two parts. First, using a simple example we gain more insight into the measurement update using our new flows. This will help understand the benefits and potential issues. Next, using a 1D example of a nonlinear process and measurement model, we present results for the GMPF filters compared to those of a DHF based on the *exact flow* and a SIR-PF with 500 particles. The particular

number for the SIR-PF particles is chosen, because the filter seem to saturate around this value. Increasing it further does not lead to any significant improvement in the performance.
First, we study the working of our two new flow equations. For that we consider the tri-modal form of the prior density,

$$p(x_{k+1}|Z_k) = \sum_{i=1}^{3} w_i \mathcal{N}(x_{k+1}|\mu_i, P_i) \tag{6.40}$$

where $\mathbf{w} = \{0.6, 0.25, 0.15\}$, $\boldsymbol{\mu} = \{-10, 4, 10\}$ and $\mathbf{P} = \{1, 0.5, 3\}$. Also the likelihood is given by,

$$p(z_{k+1}|x_{k+1}) = \frac{1}{\sqrt{2\pi}} \exp\left[-0.5\left(z_{k+1} - \frac{x_{k+1}^2}{20}\right)^2\right] \tag{6.41}$$

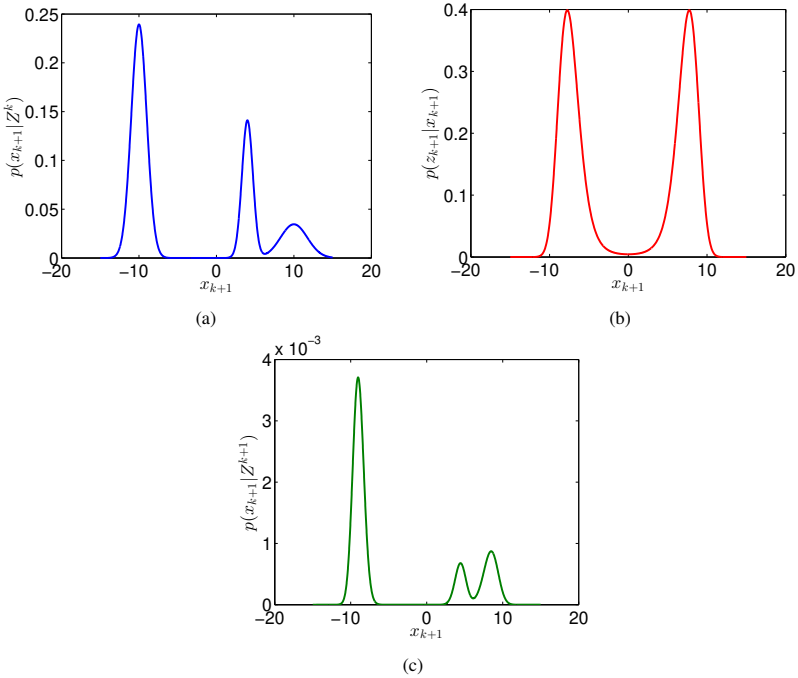


Figure 6.1: (a) Prior density, (b) Likelihood and (c) the Posterior density for the toy model.

where $z_{k+1} = 3$. We see that the prior density is given by a tri-modal GMM, while the likelihood is bi-model for the given set of parameters. We generate 100 samples from the prior density and apply equations (6.4), (6.28) and (6.31) separately to propagate the particles through the pseudo-time loop. We discretize the λ into 30 geometrically spaced points between 0 and 1. Below, we plot the three flow values for the particles vs. the pseudo-time values.

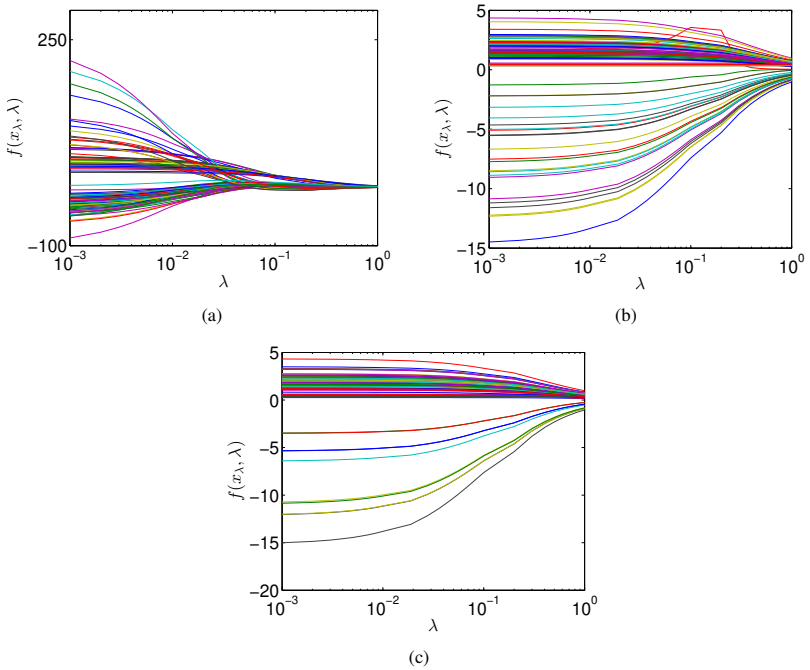


Figure 6.2: (a) Exact flow, (b) *GMPF*-1 and (c) *GMPF*-2 vs. λ

We see some interesting trends here. The *exact flow* has very large dynamic range when compared to the other two flows, thus suggesting a requirement for very fine λ discretization. The flow values for *GMPF*-2 are more concentrated for all value of λ, suggesting lesser variance of the particles. We see relatively large value of the flow for some particle, but they are not that many. Finally, we plot the posterior particles super-imposed on the posterior density.

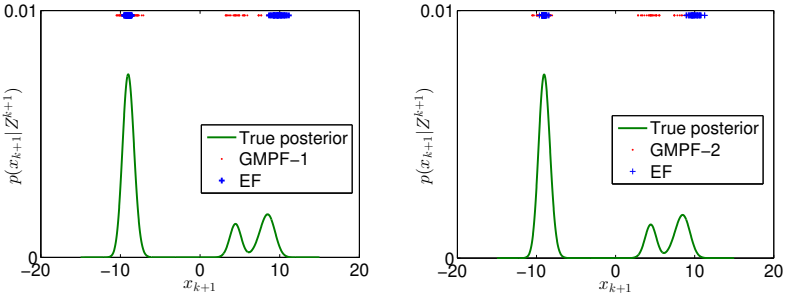


Figure 6.3: True posterior density with the particle spread for (a) *GMPF*-1 and (b) *GMPF*-2.

Here, we see the full power of our new flows. The *exact flow* clearly fails to capture the mode 2, located at the x=4. Also, the spread about the two captured mode less than what is actual, suggesting the inadequacy of the flow equation to fully capture the multi-modal dynamics. On the other hand, the other flows nicely capture the full structure of posterior density, by not only spreading the particles around the three modes but also encapsulating the information regarding the variance of the individual modes. *GMPF*-2 is better than 1 as its mode variance is close to the actual.

In the next part of this section, we consider a univariate non-stationary growth model, similar to the one studied in [KD03].

$$x_{k+1} = 0.5x_k + 25\frac{x_k}{1+x_k^2} + 8\cos(1.2k) + w_k \tag{6.42}$$

$$y_{k+1} = \frac{x_{k+1}^2}{20} + v_k \tag{6.43}$$

where w_k is the bi-modal process noise distribution, represented by a GMM such that $p(w_k) = 0.5\mathcal{N}(w_k|-2,3) + 0.5\mathcal{N}(w_k|2,3)$, while v_k is the measurement noise given by $\mathcal{N}(v_k|0,1)$. We compare the performance of DHF based on the variants of our new flows, employing both resampling and merging for reducing the number of components. They are named here as DHF-*GMPF*-1-resamp, DHF-*GMPF*-1-merge, DHF-*GMPF*-2-resamp and DHF-*GMPF*-2-merge. We use 3 components to represent the initial density $p(\mathbf{x}_0)$ i.e. M = 3. Mean and covariance for individual components are chosen such that the overall mean is zero and covariance is 25. The number of particles per component, L, is chosen to be 50. We use 30 geometrically spaced λ points between 0 and 1. Linearization of the process model $h(\mathbf{x})$ and the variable $\alpha(\mathbf{x})$ is carried about individual particles. In addition to the variants of our new flows, we also study the performance of the Gaussian prior based *exact flow* with 100 particles (DHF-EF), SIR particle filter with 500 particles (PF-500) and the Extended Kalman filter (EKF). We simulate the scenario for a total of 50 times, each running for 100 time instances. Increasing the number of particles beyond the values stated for the DHF-EF and SIR-PF, does not enhance the performance significantly. Also, the choice of parameters for our new flows is made in part to achieve a trade-off between the performance and the computational complexity. In Figure 6.4, we plot

the time averaged root mean square error (RMSE) against the realizations (simulation index).

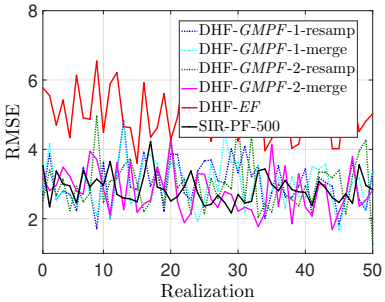


Figure 6.4: Time averaged RMSE vs. realizations

We note that the lowest error is given by the DHF-*GMPF*-2-merge, closely followed by PF-500. This is followed by DHF-*GMPF*-2-resamp and other the variants of DHF-*GMPF*-1. There are three observations to be made here. First, the filter based on the inclusion of the cubic term in the flow leads to a better performance. Secondly, merging of components seems to be a better solution as compared to the resampling, and thirdly, differences between all GMPF versions and PF-500 are not significant. Next we find that DHF-EF, in comparison to the *GMPF* and PF, exhibits a large error. This further supports our claim, that a flow based on the single Gaussian prior assumption might be inadequate for scenarios involving multi modal densities. EKF fails to track the state and has the highest error amongst all. RMSE for the EKF is not shown in Figure 6.4.

Method	Avg. RMSE	Var. RMSE
DHF-*GMPF*-1-resamp	3.04	7.02
DHF-*GMPF*-1-merge	2.99	6.93
DHF-*GMPF*-2-resamp	2.97	5.67
DHF-*GMPF*-2-merge	2.77	5.42
DHF-EF	5.08	6.20
EKF	19.2	14.24
PF-500	2.86	3.74

Table 6.1: Comparison for different filters

Furthermore, we note that the lowest value for the error standard deviation is given by the particle filter. DHF-*GMPF*-2-merge fares second, followed by the other variants of DHF-*GMPF*. This shows that error variability for our newly proposed filters is slightly higher when compared to the standard particle filter. EKF is the quickest of all filters, taking 0.004 second per iteration. Next comes the PF-500 which takes about 0.009 second, followed by the DHF-EF with 0.013

second, while the DHF-*GMPF*-2-merge takes about 0.04 second per iteration.

6.4 Conclusions

In the log-homotopy based particle flow filters, the prior density is usually approximated by a single Gaussian. This assumption may be inadequate for scenarios with highly nonlinear models and/or non-Gaussian noises. In this chapter, we have derived new Log-homotopy based flows, for which the Gaussian assumption is relaxed. Instead, we model the prior density as a Gaussian mixture. This leads to several new particle flow variants. We have studied one dimensional systems with nonlinear process and measurement models. The process noise is chosen to be non-Gaussian, represented by a mixture of two equally weighted Gaussian components. We show that the filters based on our new flows have same RMSE as the optimal particle filter. On the other hand the error variability for these filters is slightly high. Several factors might affect the overall performance e.g. the number of components M, the number of particles per components L, pseudo-time discretization $\Delta\lambda$, choice of discretization points for $h(\mathbf{x})$ and $\alpha(\mathbf{x})$, choice of numerical integration etc.

Chapter 7

Conclusion & Future works

State estimation of naturally occurring or man made dynamical systems is an ubiquitous problem. Recursive Bayesian estimation offers an intuitive framework for the inference of hidden states of a dynamical system based on measurements. RBE consist of two steps: prediction and correction. The correction step, also called the measurement update or the data assimilation step is of the main interest. Measurements generated from the most dynamical systems are typically related to states in a nonlinear manner. Additionally, they can also be corrupted by the presence of non-Gaussian noises. This presents a challenge to the traditional state estimation schemes like SMC, since they suffer from issues like the *weight degeneracy* and the *curse of dimensionality*. Weight degeneracy could occur when the prior density and the likelihood have very small overlapping region, e.g. the likelihood is peaked. The curse of dimensionality, on the other hand, occurs due to the geometric increase in the number of particles required to adequately represent probability densities, with a linear increase in the state space dimensionality. These two problems can get coupled for some particularly degenerate problems. The progressive inclusion of measurements has been cited in the literature as one of the methods to combat these problems. The main concept is a gradual or step by step inclusion of the observational data during measurement update step. It is hoped that this would help lessen the degeneracy, and could also help against the curse of dimensionality to a certain extent.

The idea is rather recent, a plethora of literature has already been published. All of these methods come with their own sets of specific assumptions and simplifications, and they have garnered varied level of success. While some are simple in their operation, other can be too burdensome in terms of the computation. The log-homotopy based particle flow class of method, also named DHF after their inventor, is a similar method of nonlinear/non-Gaussian data assimilation that also introduces the effect of measurements gradually. The main idea behind it, is the introduction of a synthetic time, in which particles are moved from their prior locations to their corresponding posterior ones. An ordinary differential equation, also called the *flow equation*, dictates the motion of the particle in the artificial time. DHF, even though not new in the literature, is still not fully explored in detail. It lacks the in-depth theoretical and numerical analysis that the other contemporary filters have gone through. Especially, the implementation details are very application specific. In this thesis, we have pointed out the key factors affecting the performance of a generic DHF. The highlighted factors have been studied individually in the detail, and possible suggestions for the improvement have been made with regards to each

of them. This includes different methods for the pseudo-time discretization, different integration schemes, estimation of the prior covariance matrix, and the particle redrawing. We have compared the results for these different schemes by simulating a challenging, nonlinear and non-Gaussian scenario. It has been shown that a DHF employing the shrinkage estimation and properly done redrawing outperform a simple SMC method, i.e. a bootstrap particle filter, with considerably lesser execution time.

Next, we looked into the Bayesian processing of massive data. The massive data approach provides the possibility for the extraction of more information content, given a larger set of measurements, thus increasing the estimation accuracy. However, this comes with enhanced computational requirements, hence limiting the use of many standard estimation methods such as SMCMC. The source of the problem can be pinpointed to the evaluation of the likelihood function, which even in the case of factorization, presents a significant processing challenge. Many solutions have been proposed to solve or bypass this bottleneck. One of these approximate method, namely *confidence sampling* is of particular interest. Confidence sampling squeezes the original observational data set to a smaller one to be processed by the MCMC sampler, while still maintaining theoretical guarantees. It is based on the use of the so called *concentration inequalities*, which can be used to theoretically bound the maximum deviation of the approximated target density from the true target. When used in an MCMC setting, the use of such inequalities yields a stopping criterion for the sampling procedure. Though the target density is still approximated, there are potential processing gains achieved by limiting the evaluation of the likelihood to fewer terms, together with the guarantee ensuring that the sampled density is always within a specified distance from the actual target density. In this work, we have expanded on an earlier work done by Freitas et.al. in [FSM15] and have combined the idea of confidence sampling based MCMC together with the log-homotopy based particle flow filters (DHF), in that the later is used to construct a better proposal density to be used within the former. We have termed our newly proposed method as the adaptive SMCMC with particle flow based proposal or ASMCMC-DHF. We have thoroughly analyzed the performance of ASMCMC-DHF for the processing of massive data under different settings of algorithm and system parameters. We have noted that our new scheme can handle the effect of the increasing dimensionality in a graceful manner. Also, it has been shown that our method not only outperforms the well established methods like the particle filter, but also performs better than its parent algorithm, ASMCMC.

Our next contribution is to develop a grid based nonlinear filtering algorithm, by combining the log-homotopy based flow with the tensor decomposition based solution of the Fokker-Planck equation (FPE), as developed by W. Sun and M. Kumar ([SK14], [SK15a] and [SK15b]). Expressing the probability densities and the FPE in the Canonical/Parallel factor Decomposition (CPD) form has been identified as the key in fighting the curse of dimensionality in discretized problems. A cost function comprising of FPE and the associated constraints is formed, which is then minimized. The resulting matrix-vector equation is solved via the regularized alternating least squares (R-ALS), yielding the basis vectors. The nonlinear filtering algorithm has the time update step based on the solution of the tensorized FPE w.r.t.the real time. The FPE is solved for the second time for the measurement update step, though this time w.r.t.the pseudo-time parameter λ. This is achieved by combining the tensor framework with the log-homotopy based particle flow. We have named our newly devised filter as, the *tensorized filter*. For a simulated scenario, we have shown that our newly devised filter approaches the Crámer-Rao

lower bound, albeit at the cost of significant computational time.

As our final contribution, we have expanded the existing mathematical framework of the log-homotopy based particle flows. This is done by generalizing the Gaussian assumption for one of the more elegant flows, namely the *exact flow*. We start by modeling the prior density as a sum of Gaussians. This leads to several new particle flow variants with the same form as the *exact flow*. Based on these new flows, we have derived two new variants of the DHF. We show that the filters based on our new flows have significantly improve the exact flow in the case of a non-linear filtering problem

7.1 Future works

In this section, we will point towards some of the future extension of the current work.

Inclusion of stochastic term in the flow equation

Particle flow equations derived by F.Ḋaum and J.Ḣuang, and the ones derived in Chapter 6 of this thesis are basically deterministic maps. Given an initial condition, one can exactly know about the final state of a particle. These equations are not mathematically exact because of the simplifying assumption made in their derivation. Hence, even though fulfilling the basic requirements for the Bayesian assimilation of the data and working fine for the studied examples, it would be interesting to see how the flows perform with the addition of a stochastic term. Theoretically the stochastic term could cater for the inherent imperfectness of the flows that could pose problems for extremely challenging real life applications. In a recent work [DHN16], the stochastic term has been used in context of the log-homotopy based particle flow. As the flow equation is no longer deterministic, this requires the use of stochastic integration methods. A very thorough discussion on the integration of stochastic differential equations can be found in [Cro15].

Improving the performance of the tensorized filter

In the discussion and implementation of the *tensorized filter*, we have only used the *exact flow*. This is because it can readily be decomposed into a separable form, unlike the other flows, where ALS algorithm might have to be run to get them into a similar format. This would lead to an increase in the processing time. However, as noted in Chapter 3, the non-zero diffusion constrained flow performs better than the exact flow, particularly in the non-Gaussian cases. Therefore, it would be worth investigating its performance when used in a tensorized framework. This might also necessitate the use of other methods to solve the quadratic optimization problem to reduce the processing time. We mentioned several such methods in Chapter 6, such as the enhanced line search (ELS) [RCH08], replacement of the ALS iterations by several joint diagonalization problems and the subsequent matching to get the best estimates of the factor matrices [RH08], a semi-algebraic framework for approximate CPD via simultaneous matrix diagonalization and generalized unfolding (SMD-GU) [RSH12] and a semi-algebraic tensor decomposition based on the joint eigenvalue decomposition [LA11]. In [TPC16] a new algorithm for solving ALS problems has been proposed that is based on the random partitioning of the factor matrices such that each of those portion is individually updated. It can be conjectured

that a significant speed up in the computation can also be achieved by using dynamic grid adaptation due to the relatively smaller domain involved. One idea is to use the concept described in [CBS00] for moving the grid by computing the significant domain based on the Chebyshev inequality. Another possible approach is to use *sparse grids*, as defined and used in [KS13] and [Gar01]. It is hoped that for higher dimensional nonlinear/non-Gaussian problems, the full power of the tensor based approach can be exploited and the extra computational cost can be justified by the higher level of accuracy achieved.

Better approximation of nonlinear functions in DHF-GMPF

We have used a first order linear approximation for the variable α(x) and the measurement functions ψ(x) in the derivation of the Gaussian mixture particle flow. It would definitely be interesting to introduce the quadratic terms in these approximations, as it could increase the performance of the new flows in systems with strong non-linearities.

Flow with Likelihood also modeled through a sum of Gaussians

Finally, as a natural extension of our latest work, a more general likelihood function could also be modeled as a sum of Gaussians and further new flow equations can be derived.

Bibliography

[AH09] I. Arasaratnam and S. Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.

[ALS07] Simo Ali-Löytty and Niilo Sirola. Gaussian Mixture Filter in Hybrid Navigation. In *Proceedings of The European Navigation Conference GNSS 2007*, pages 831–837, May 2007.

[AMGC02] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on Particle Filters for Online Non-linear/ Non-Gaussian Bayesian Tracking. *IEEE transaction on signal processing*, 50(2):174–188, February 2002.

[AR09] C. Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 03 2009.

[AS72] D.L. Alspach and H.W. Sorenson. Nonlinear Bayesian Estimation Using Gaussian Sum Approximation. *IEEE Transaction on Automatic Control*, 17(4):439–448, 1972.

[BDH15] R. Bardenet, A. Doucet, and C. Holmes. On Markov chain Monte Carlo methods for tall data. *ArXiv e-prints*, May 2015.

[BG01] C. Berzuini and W. Gilks. RESAMPLE-MOVE Filtering with Cross-Model Jumps. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 117–138. Springer New York, 2001.

[BG13] P. Bunch and S. Godsill. Particle Filtering with Progressive Gaussian Approximation to Optimal Importance Density. In *IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2013.

[BG14] P. Bunch and S. Godsill. The Progressive Proposal Particle Filter: Better Approximations to the Optimal Importance Density. *ArXiv e-prints*, January 2014.

[BG16] Pete Bunch and Simon Godsill. Approximations of the Optimal Importance Density using Gaussian Particle Flow Importance Sampling. *Journal of the American Statistical Association*, 111(514):748–762, 2016.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BL97] N. Bergman and L. Ljung. Point-mass filter and Cramer-Rao bound for terrain-aided navigation. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 1, pages 565–570, December 1997.

[BM05] Gregory Beylkin and Martin J. Mohlenkamp. Algorithms for Numerical Analysis in High Dimensions. *SIAM Journal on Scientific Computing*, 26(6):2133–2159, 2005.

[Bro97] R. Bro. PARAFAC Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2):149–171, October 1997.

[BS05] C.F. Beckmann and S.M. Smith. Tensorial extensions of independent component analysis for multisubject FMRI analysis. *NeuroImage*, 25(1):294–311, 2005.

[BS14] K.L. Bell and L.D. Stone. Implementation of the homotopy particle filter in the JPDA and MAP-PF multi-target tracking algorithms. In *Proceedings of the 17th International Conference on Information Fusion (FUSION)*, pages 1–8, July 2014.

[BSDH09] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82–100, December 2009.

[CBS00] S. Challa and Y. Bar-Shalom. Nonlinear filter design using Fokker-Planck-Kolmogorov probability density evolutions. *IEEE Transactions on Aerospace and Electronic Systems*, 36(1):309–315, January 2000.

[CCL15] R. Casarin, Radu V. Craiu, and F. Leisen. Embarrassingly Parallel Sequential Markov-chain Monte Carlo for Large Sets of Time Series. *ArXiv e-prints*, December 2015.

[C.G69] C.Gear. The automatic integration of stiff ordinary differential equations. In *Information processing 68 (Proc. IFIP Congress, Edinburgh, 1968)*, volume 1, pages 187–193, 1969.

[Cha06] Suman Chakravorty. A homotopic Galerkin approach to the solution of the Fokker-Planck-Kolmogorov equation. In *American Control Conference, 2006*, June 2006.

[CM10] L. Chen and R. Mehra. A study of nonlinear filters with particle flow induced by log-homotopy. In *Proc.SPIE*, 2010.

[Cro15] D. Crouse. Basic tracking using nonlinear continuous-time dynamic models [Tutorial]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):4–41, February 2015.

[CWDH11] S. Choi, P. Willet, F. Daum, and J. Huang. Discussion and Application of Homotopy filter. In *Proc.SPIE*, 2011.

[Dau86] F. Daum. Exact finite-dimensional nonlinear filters. *IEEE Transactions on Automatic Control*, 31(7):616–622, July 1986.

[DC12] T. Ding and M.J. Coates. Implementation of Daum-Huang Exact Flow Particle Filter. In *IEEE Statistical Signal Processing Workshop (SSP)*, 2012.

[DD96] L. DeLathauwer and B. DeMoor. From matrix to tensor: Multilinear algebra and signal processing. In *4th IMA International Conference on Mathematics in Signal Processing*, 1996.

[DGYM+15] Syamantak Datta Gupta, Jun Ye Yu, Mahendra Mallick, Mark Coates, and Mark Morelande. Comparison of angle-only filtering algorithms in 3d using EKF, UKF, PF, PFF, and ensemble KF. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 1649–1656, July 2015.

[DH03] F. Daum and J. Huang. Curse of Dimensionality and Particle Filters. In *IEEE Aerospace conference*, pages 1970–1993. IEEE, March 2003.

[DH07] F. Daum and J. Huang. Nonlinear filters with log-homotopy. In *Proc.SPIE*, 2007.

[DH09a] F. Daum and J. Huang. Nonlinear filters with particle flow. In *Proc.SPIE*, 2009.

[DH09b] F. Daum and J. Huang. Nonlinear filters with particle flow induced by log-homotopy. In *Proc.SPIE*, 2009.

[DH10a] F. Daum and J. Huang. Generalized particle flow for nonlinear filters. In *Proc.SPIE*, 2010.

[DH10b] F. Daum and J. Huang. Numerical experiments on nonlinear filters with exact particle flow induced by log-homotopy. In *Proc.SPIE*, 2010.

[DH13a] F. Daum and J. Huang. Particle flow with non-zero diffusion for non-linear filters. In *Proc.SPIE*, 2013.

[DH13b] F. Daum and J. Huang. Zero curvature particle flow for nonlinear filters. In *Proc.SPIE*, 2013.

[DHKK09] F. Daum, J. Huang, M. Krichman, and T. Kohen. Seventeen dubious methods to approximate the gradient for nonlinear filters with particle flow. In *Proc.SPIE*, 2009.

[DHN11a] F. Daum, J. Huang, and A. Noushin. Coulomb's law particle flow for nonlinear filters. In *Proc. SPIE*, 2011.

[DHN11b] F. Daum, J. Huang, and A. Noushin. Numerical experiments for Coulomb's law particle flow for nonlinear filters. In *Proc.SPIE*, 2011.

[DHN16] F. Daum, J. Huang, and A. Noushin. Gromov's method for Bayesian stochastic particle flow: a simple exact formula for Q. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, volume 3, pages 19–21, September 2016.

[DHNK09] F. Daum, J. Huang, A. Noushin, and M. Krichman. Gradient estimation for particle flow induced by log-homotopy for nonlinear filters. In *Proc.SPIE*, 2009.

[DKG16] B. Demissie, M. Altamash Khan, and F. Govaers. Nonlinear filter design using Fokker-Planck propagator in Kronecker tensor format. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 1–8, July 2016.

[Eid04] A. Eidehall. *An Automotive Lane Guidance System: Licentiate Thesis*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 2004.

[EMFD15] F. Eler de Melo, S. Maskell, M. Fasiolo, and F. Daum. Stochastic Particle Flow for Nonlinear High-Dimensional Filtering Problems. *ArXiv e-prints*, November 2015.

[Epp87] James F. Epperson. On the Runge Example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

[EU117] Integrated maritime surveillance. `https://ec.europa.eu/maritimeaffairs/policy/integrated_maritime_surveillance_en`, 2017.

[EUJ17] European Commission Blue Hub project - Maritime Data, Data Fusion & Tracking. `https://bluehub.jrc.ec.europa.eu/research_areas_maritime`, 2017.

[Eve94] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res*, 99:10143–10162, 1994.

[Far08] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2008.

[FSM15] A.De Freitas, F. Septier, and L. Mihaylova. Sequential Markov Chain Monte Carlo for Bayesian Filtering with Massive Data. *ArXiv e-prints*, December 2015.

[FSMG15] A. De Freitas, F. Septier, L. Mihaylova, and S. Godsill. How can subsampling reduce complexity in sequential MCMC methods and deal with big data in target tracking? In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 134–141, July 2015.

[Gar04] C. W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, volume 13 of *Springer Series in Synergetics*. Springer-Verlag, Berlin, 3rd edition, 2004.

[Gar01] Jochen Garcke. *Sparse Grids in a Nutshell*, pages 57–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 201.

[GC01] S. Godsill and T. Clapp. Improvement Strategies for Monte Carlo Particle Filters. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 139–158. Springer New York, 2001.

[GCC11] M. Girolima, B. Calderhead, and S.A. Chin. Riemann Manifold Langevin and Hamiltonian Monte Carlo. *Journal of Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):123–214, 2011.

[GDK+17] F. Govaers, B. Demissie, M. Altamash Khan, M. Ulmke, and W. Koch. Tensor Decomposition based Multi Target Tracking in Cluttered Environments. *Journal of Advances in Information Fusion (JAIF)*, 2017. submitted.

[GdlEA13] F. García, A. de la Escalera, and J. M. Armingol. Enhanced obstacle detection based on Data Fusion for ADAS applications. In *In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1370–1375, October 2013.

[GSS93] N. Gordon, D. Salmond, and A.. Smith. Novel approach to nonlinear/non-Gaussian bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing.*, pages 107–113, April 1993.

[Gus05] F. Gustafsson. Statistical signal processing for automotive safety systems. In *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pages 1428–1435, July 2005.

[Gus13] Fredrik Gustafsson. *Statistical sensor fusion*. Studentlitteratur AB, 2 edition, May 21, 2013.

[Haf80] L.R. Haff. Emprirical Bayes Estimation of the Multivariate Normal Covariance Matrix. *The Annals of Statistics*, 8(3):586–597, 1980.

[HDP15] J. Heng, A. Doucet, and Y. Pokern. Gibbs Flow for Approximate Transport with Applications to Bayesian Computation. *ArXiv e-prints*, September 2015.

[HF03] U.D. Hanebeck and O. Feiermann. Progressive Bayesian estimation for nonlinear discrete-time systems: the filter step for scalar measurements and multidimensional states. In *IEEE Conference on Decision and Control*, volume 5, pages 5366–5371. IEEE, December 2003.

[HJSM11] J. Hagmar, M. Jirstrand, L. Svensson, and M. Morelande. Optimal Parameterization of Posterior Densities Using Homotopy. In *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pages 1–8, 5-8 July, 2011. IEEE.

[HM98] P. L. Houtekamer and Herschel L. Mitchell. Data Assimilation Using an Ensemble Kalman Filter Technique. *Monthly Weather Review*, 126(3):796–811, 1998.

[Jan05] J. Jansson. *Collision Avoidance Theory with Applications to Automotive Collision Mitigation*. PhD thesis, Linköping University, Sweden, 2005.

[JMKB96] Jerome J. Workman JR., Paul R. Mobley, Bruce R. Kowalski, and Rasmus Bro. Review of Chemometrics Applied to Spectroscopy: 1985-95, part i. *Applied Spectroscopy Reviews*, 31(1-2):73–124, 1996.

[JS03] Tao Jiang and N. D. Sidiropoulos. A direct blind receiver for SIMO and MIMO OFDM systems subject to unknown frequency offset and multipath. In *2003 4th IEEE Workshop on Signal Processing Advances in Wireless Communications - SPAWC 2003*, pages 358–362, June 2003.

[JUDW95] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the 1995 American Control Conference*, volume 3, pages 1628–1632, June 1995.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KB09] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[KBD05] Zia Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1819, November 2005.

[KBS15] C. Kreucher, K. Bell, and D. Sobota. A comparison of tracking algorithms for supermaneuverable targets. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 534–541, July 2015.

[KCJ07] M. Kumar, Suman. Chakravorty, and J.L. Junkins. A semianalytic meshless approach to the Fokker-Planck equation with application to hybrid systems. In *In Proceedings of the 46th IEEE Conference on Decision and Control*, pages 3078–3083, December 2007.

[KD03] J.H. Kotecha and P. M. Djuric. Gaussian sum particle filtering. *Signal Processing, IEEE Transactions on*, 51(10):2602–2612, October 2003.

[KdFL+17] M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Bayesian Processing of Big Data using Log Homotopy Based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2017*, pages 1–6, October 2017.

[KdFL+18] M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Improving Adaptive Markov chain Monte Carlo using Log-Homotopy Particle Flow based proposal. To be submitted to Journal of Advances in Information Fusion (JAIF), 2018.

[KE97] V. Krishnamurthy and R. J. Elliott. Exact finite-dimensional filters for doubly stochastic auto-regressive processes. *IEEE Transactions on Automatic Control*, 42(9):1289–1293, September 1997.

[KLS11] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate Online Kernel Density Estimation with Gaussian Kernels. *Pattern Recognition*, 44:2630–2642, 2011.

[KLS16] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Online Kernel Density Estimation with Gaussian Kernels and Online Discriminative Kernel Density Estimation with Gaussian Kernels. `http://www.vicos.si/File:Maggot_v3.5.zip`, 2016.

[KN06] P. Kumar and S. Narayanan. Solution of Fokker-Planck equation by finite element and finite difference methods for nonlinear systems. *Sadhana*, 31(4):445–461, August 2006.

[Koc08] J. W. Koch. Bayesian approach to extended object and cluster tracking using random matrices. *IEEE Transactions on Aerospace and Electronic Systems*, 44(3):1042–1059, July 2008.

[KS13] C. Kalender and A. Schottl. Sparse Grid-Based Nonlinear Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2386–2396, October 2013.

[KU14] M. Altamash Khan and M. Ulmke. Non-linear and non-Gaussian state estimation using log-homotopy based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2014*, pages 1–6, October 2014.

[KU15] M. Altamash Khan and M. Ulmke. Improvements in the Implementation of Log-Homotopy based Particle Flow Filters. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 74–81, July 2015.

[KUD+16] M. Altamash Khan, M. Ulmke, B. Demissie, F. Govaers, and W. Koch. Combining Log-Homotopy Flow with Tensor decomposition based solution for Fokker-Planck equation. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 2229–2236, July 2016.

[KUK16] M. Altamash Khan, M. Ulmke, and W. Koch. A log homotopy based particle flow solution for mixture of Gaussian prior densities. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 546–551, September 2016.

[KUK17] M. Altamash Khan, M. Ulmke, and W. Koch. Analysis of Log-Homotopy based Particle Flow Filters. *Journal of Advances in Information Fusion (JAIF)*, 12(1), June 2017.

[LA11] X. Luciani and L. Albera. Semi-algebraic canonical decomposition of multiway arrays and Joint Eigenvalue Decomposition. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4104–4107, May 2011.

[LC16] Y. Li and M. Coates. Fast particle flow particle filters via clustering. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 2022–2027, July 2016.

[LKN13] N. Li, S. Kindermann, and C. Navasca. Some convergence results on the Regularized Alternating Least-Squares method for tensor decomposition. *Linear Algebra and its Applications*, 438(2):796 – 812, 2013.

[LP85] S.P. Lin and M.D. Perlman. A Monte Carlo Comparison of four estimators of a Covariance Matrix. *Journal of Multivariate Analysis*, 6(44):411–429, 1985.

[LSSM15] R. Lamberti, F. Septier, N. Salman, and L. Mihaylova. Sequential Markov Chain Monte Carlo for multi-target tracking with correlated RSS measurements. In *Proceedings of the 10th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2015.

[LW03] Olivier Ledoit and Michael Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5):603 – 621, 2003.

[LW04a] Olivier Ledoit and Michael Wolf. Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

[LW04b] Olivier Ledoit and Michael Wolf. Well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365 – 411, 2004.

[LW12] Olivier Ledoit and Michael Wolf. Nonlinear shrinkage estimation of large-dimensional covariance matrices. *The Annals of Statistics*, 40(2):1024–1060, April 2012.

[LZC15] Yunpeng Li, Lingling Zhao, and M. Coates. Particle flow auxiliary particle filter. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 157–160, December 2015.

[M.B63] M.Brown. A generalized error function in n dimensions. Technical Report Technical Memorandum No. NMC-TM-63-8, U. S. NAVAL MISSILE CENTER Point Mugu, California, April 1963.

[MB05] A. Masud and L.A. Bergman. Solution of the four dimensional Fokker-Planck equation: Still a challenge. In *Proceedings of the 9th International Conference on Structural Safety and Reliability ICOSSAR'05*, July 2005.

[MC15] N. Moshtagh and M.W. Chan. Multisensor fusion using homotopy particle filter. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 1641–1648, July 2015.

[MMBs02] A. Marrs, S. Maskell, and Y. Bar-shalom. Expected Likelihood for Tracking in Clutter with Particle Filters. In *Proceedings of SPIE Signal and Data Processing of Small Targets*, pages 230–239, 2002.

[MOLG01] C. Musso, N. Oudjane, and F. Le Gland. Improving Regularised Particle Filters. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 247–271. Springer New York, 2001.

[MRR+53] N. Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[MW11] M.Welling and Yee W.Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[Nwa93] C. Nwagboso. *Automotive Sensory Systems*. Springer Netherlands, Netherlands, 1993.

[NWX13] W. Neiswanger, C. Wang, and E. Xing. Asymptotically Exact, Embarrassingly Parallel MCMC. *ArXiv e-prints*, November 2013.

[Øk14] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications (Universitext)*. Springer, 6th edition, January 2014.

[PS99] M.K. Pitt and N. Shephard. Filtering via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446), 1999.

[QVK14] M. Quiroz, M. Villani, and R. Kohn. Speeding Up MCMC by Efficient Data Subsampling. *ArXiv e-prints*, April 2014.

[RAC14] R.Bardenet, A.Doucet, and C.Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 405–413. JMLR Workshop and Conference Proceedings, 2014.

[RC04] C. Robert and G. Casella. In *Monte Carlo Statistical Methods*, Springer Texts in Statistics. Springer-Verlag New York, 2 edition, 2004.

[RCH08] Myriam Rajih, Pierre Comon, and Richard A. Harshman. Enhanced Line Search: A Novel Method to Accelerate PARAFAC. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1128–1147, 2008.

[Rei11] S. Reich. A dynamical systems framework for intermittent data assimilation. *BIT Numerical Mathematics*, 51(1):235–249, 2011.

[Rei13] S . Reich. A Guided Sequential Monte Carlo Method for the Assimilation of Data into Stochastic Dynamical Systems. In Andreas Johann, Hans-Peter Kruse, Florian Rupp, and Stephan Schmitz, editors, *Recent Trends in Dynamical Systems*, volume 35 of *Springer Proceedings in Mathematics and Statistics*, pages 205–220. Springer Basel, 2013.

[RH08] F. Roemer and M. Haardt. A closed-form solution for Parallel Factor (PARAFAC) Analysis. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2365–2368, March 2008.

[RSH12] F. Roemer, C. Schroeter, and M. Haardt. A semi-algebraic framework for approximate CP decompositions via joint matrix diagonalization and generalized unfoldings. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 2023–2027, November 2012.

[SB02] N. D. Sidiropoulos and R. S. Budampati. Khatri-Rao space-time codes. *IEEE Transactions on Signal Processing*, 50(10):2396–2407, October 2002.

[SBB+16] Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and big data: the consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, 2016.

[SBG00] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388, August 2000.

[SD01] N. D. Sidiropoulos and G. Z. Dimic. Blind multiuser detection in W-CDMA systems with large delay spread. *IEEE Signal Processing Letters*, 8(3):87–89, March 2001.

[SFD02] D. Streller, K. Furstenberg, and K. Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Proceedings of the 5th International IEEE Conference on Intelligent Transportation Systems*, pages 118–123, 2002.

[SK14] Yifei Sun and Mrinal Kumar. Numerical solution of high dimensional stationary Fokker Planck equations via tensor decomposition and Chebyshev spectral differentiation. *Computers & Mathematics with Applications*, 67(10):1960 – 1977, 2014.

[SK15a] Y. Sun and M. Kumar. Nonlinear Bayesian filtering based on Fokker-Planck equation and tensor decomposition. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 1483–1488, July 2015.

[SK15b] Y. Sun and M. Kumar. A numerical solver for high dimensional transient Fokker-Planck equation in modeling polymeric fluids. *Journal of Computational Physics*, 289:149–168, 2015.

[SK15c] Y. Sun and M. Kumar. Solution of high dimensional transient Fokker-Planck equations by tensor decomposition. In *American Control Conference (ACC), 2015*, pages 1475–1480, July 2015.

[SK15d] Y. Sun and M. Kumar. Uncertainty propagation in orbital mechanics via tensor decomposition. *Celestial Mechanics and Dynamical Astronomy*, pages 1–26, 2015.

[SKS02] M. Simandl, J. Kralovec, and T. Soderstrom. Anticipative grid design in point-mass approach to nonlinear state estimation. *IEEE Transactions on Automatic Control*, 47(4):699–702, April 2002.

[SKS06] M. Simandl, J. Kralovec, and T. Soderstrom. Advanced point-mass method for nonlinear state estimation. *Automatica*, 42(7):1133 – 1145, 2006.

[SPCG09] F. Septier, S. K. Pang, A. Carmi, and S. Godsill. On MCMC-Based particle methods for Bayesian filtering: Application to multitarget tracking. In *3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 360–363, December 2009.

[SR08] S.Matzka and R.Altendorfer. A comparison of Track-to-Track Fusion Algorithms for Automotive Sensor Fusion. In *In Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea*, August 2008.

[SS05a] J. Schäfer and J. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21:754–764, 2005.

[SS05b] J. Schäfer and J. Strimmer. A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics, 2005.

[SSCT04] T Strobel, A Servel, C Coue, and T Tatschke. Compendium on sensors - state-of-the-art of sensors and sensor data fusion for automotive preventive safety. In *Applications, ProFusion IP Deliverable, PReVENT IP, European Commission, Retrieved 21 March 2005 from from PReVENT IP*, 2004.

[Ste56] C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical and Statistical Probability*, 1956.

[Ste75] C. Stein. Estimation of a covariance matrix, Rietz Lecture. Technical report, 39th Annual Meeting IMS, Atlanta, GA, 1975.

[Ste82] C. Stein. Series of lectures given at the University of Washington. Technical report, Seattle, 1982.

[SWC+10] Marc A. Suchard, Quanli Wang, Cliburn Chan, Jacob Frelinger, Andrew Cron, and Mike West. Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures. *Journal of Computational and Graphical Statistics*, 19(2):419–438, 2010. PMID: 20877443.

[SY16] R. Streubel and B. Yang. Fusion of stereo camera and MIMO-FMCW radar for pedestrian tracking in indoor environments. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 565–572, July 2016.

[TPC16] P. Tichavsḱy, A. H. Phan, and A. Cichocki. Partitioned Alternating Least Squares Technique for Canonical Polyadic Tensor Decomposition. *IEEE Signal Processing Letters*, 23(7):993–997, July 2016.

[TTKM16] E. Taghavi, R. Tharmarasa, T. Kirubarajan, and M. McDonald. Multisensor Multitarget Bearing Only Sensor Registration. *arxiv*, March 2016.

[Tuc64] L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology.*, pages 110–127. Holt, Rinehart and Winston, New York, 1964.

[UNC15] Review of Maritime Transport 2015. http://unctad.org/en/pages/PublicationWebflyer.aspx?publicationid=1374, 2015.

[VT02] M. Alex O. Vasilescu and Demetri Terzopoulos. *Computer Vision — ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I*, chapter Multilinear Analysis of Image Ensembles: Tensor Faces. Springer Berlin Heidelberg, 2002.

List of Figures

List of Tables

List of Algorithms

Index

List of Publications

- M. Altamash Khan and M. Ulmke. Non-linear and non-Gaussian state estimation using log-homotopy based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2014*, pages 1–6, October 2014

- M. Altamash Khan and M. Ulmke. Improvements in the Implementation of Log-Homotopy based Particle Flow Filters. In *Proceedings of the 18th International Conference on Information Fusion (FUSION)*, pages 74–81, July 2015

- M. Altamash Khan, M. Ulmke, B. Demissie, F. Govaers, and W. Koch. Combining Log-Homotopy Flow with Tensor decomposition based solution for Fokker-Planck equation. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 2229–2236, July 2016

- B. Demissie, M. Altamash Khan, and F. Govaers. Nonlinear filter design using Fokker-Planck propagator in Kronecker tensor format. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 1–8, July 2016

- M. Altamash Khan, M. Ulmke, and W. Koch. A log homotopy based particle flow solution for mixture of Gaussian prior densities. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 546–551, September 2016

- M. Altamash Khan, M. Ulmke, and W. Koch. Analysis of Log-Homotopy based Particle Flow Filters. *Journal of Advances in Information Fusion (JAIF)*, 12(1), June 2017

- M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Bayesian Processing of Big Data using Log Homotopy Based Particle Flow Filters. In *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2017*, pages 1–6, October 2017

- F. Govaers, B. Demissie, M. Altamash Khan, M. Ulmke, and W. Koch. Tensor Decomposition based Multi Target Tracking in Cluttered Environments. *Journal of Advances in Information Fusion (JAIF)*, 2017. submitted

- M. Altamash Khan, A. de Freitas, L.Mihaylova, M. Ulmke, and W. Koch. Improving Adaptive Markov chain Monte Carlo using Log-Homotopy Particle Flow based proposal. To be submitted to Journal of Advances in Information Fusion (JAIF), 2018