COMPARISON OF NONLINEAR FILTERING TECHNIQUES

BY

SHANE T. GHIOTTO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Associate Professor Prashant Mehta

ABSTRACT

In a recent work it is shown that importance sampling can be avoided in the particle filter through an innovation structure inspired by traditional nonlinear filtering combined with optimal control formalisms. The resulting algorithm is referred to as *feedback particle filter*.

The purpose of this thesis is to provide a comparative study of the feedback particle filter (FPF). Two types of comparisons are discussed: i) with the extended Kalman filter, and ii) with the conventional resampling-based particle filters. The comparison with Kalman filter is used to highlight the feedback structure of the FPF. Also computational cost estimates are discussed, in terms of number of operations relative to EKF. Comparison with the conventional particle filtering approaches is based on a numerical example taken from the survey article on the topic of nonlinear filtering [2]. Comparisons are provided for both computational cost and accuracy.

To my family, friends, and professors who have supported me and helped me reach this point

ACKNOWLEDGMENTS

I would like to thank Professor Prashant Mehta for the opportunities he has continued to provide me to work on challenging and interesting problems over the last few years, as well as his constant guidance in helping me learn and discover countless concepts.

I would also like to thank my fellow labmate, Adam Tilton, the drive and passion he showed in his work everyday consistently inspired me to better myself and push myself to learn new skills way beyond the scope of what I thought I was capable of.

Finally, a general thank you to the rest of the research group I worked with. They made coming into the lab for those countless hours something worth looking forward to.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

EKF	Extended Kalman Filter
FPF	Feedback Particle Filter
PF	Particle Filter
EL-BVP	Euler-Lagrange Boundary Value Problem

CHAPTER 1

INTRODUCTION

Filtering and state estimation from noisy measurements is a concept that feeds into many fields such as signal processing, navigation, and control. From global positioning systems (GPS) and target tracking to applications in computer vision and even economics, the applications of filtering are diverse and widespread.

The primary objective in filtering is to provide an estimate of the hidden state of a dynamical system. This state estimate can then be used within feedback control laws, or to display information to a user. A higher quality estimate can result in better and more desirable behavior of control laws, or a richer, more accurate interface to a user.

In the real-time setting, the quality of the state estimate provided by the filter is not the only thing that must be taken into consideration. The computational efficiency of the filter is also a key determining factor in whether a particular filter can be utilized for a real-time problem. Oftentimes, for these type of problems, a trade-off study usually must be conducted to find the proper balance between tracking accuracy and computational load across the various filtering techniques.

1.1 Filtering Problem

In this thesis, the standard model of the continuous-time nonlinear filtering problem is considered:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = a(X_t) + \dot{B}_t, \tag{1.1a}$$

$$Y_t = h(X_t) + \dot{W}_t, \tag{1.1b}$$

where $X_t \in \mathbb{R}^d$ is the state at time t, $Y_t \in \mathbb{R}^m$ is the observation on the state, $a(\cdot)$ and $h(\cdot)$ are C^1 functions, and $\{\dot{B}_t\}$, $\{\dot{W}_t\}$ are mutually independent white noise processes of appropriate dimension.

The mathematical objective of the filtering problem is to estimate the posterior distribution of X_t given the history of observsations, $\mathscr{Z}_t := \sigma(Z_s : s \leq t)$. The posterior is denoted by p^*, so that for any measurable set $A \subset \mathbb{R}^d$,

$$\int_{x \in A} p^*(x,t)\, \mathrm{d}x = \mathrm{Prob}\{X_t \in A \mid \mathscr{Z}_t\}. \tag{1.2}$$

In the case that $a(\cdot)$ and $h(\cdot)$ are linear functions, the optimal solution is given by the Kalman filter. If $a(\cdot)$ or $h(\cdot)$ are nonlinear then a nonlinear filtering algorithm is required. Many approaches exist to approximate the nonlinear filter. A common approach is to extend the solution provided by the Kalman filter (EKF) to the nonlinear problem by linearizing the functions $a(\cdot)$ and $h(\cdot)$ about the state estimate and applying the Kalman algorithm [9]. Another common approach is to use a particle filter. Various resampling techniques are available to deal with the traditional particle filter issues such as particle degeneracy where only a few particles will have significant weights. Recent approaches, such as the feedback particle filter (FPF) in [16, 14], seek to utilize the concept of feedback which has made the Kalman filter so successful in the linear problem setting.

1.2 Contribution of Work

In this thesis, various nonlinear filtering techniques are evaluated on the criteria of tracking accuracy and computational efficiency. In particular, two cases are presented: the "feedback methods" (EKF and FPF) are compared on a benchmark problem to evaluate the similarities and differences between the methods. The other case evaluated is a comparison of the resampling based particle filtering techniques with the feedback particle filter.

The remainder of the thesis is organized as follows: chapter 2 provides background on the extended Kalman filter and feedback particle filter. Chapter 3 provides background on the conventional resampling based particle filters compared to the feedback particle filter. Chapter 4 then provides a series of benchmark

problems comparing these filtering methods. Chapter 5 ends with conclusions and directions for future work.

CHAPTER 2

COMPARISON OF FEEDBACK BASED FILTERING APPROACHES

This chapter provides an introduction to the algorithms used for the feedback particle filter and the extended Kalman filter. A computational comparison of the two algorithms is made by calculating estimated operation counts. The outline of the remainder of this chapter is as follows: in section 2.1 the algorithm for the feedback particle filter is introduced, section 2.2 provides an introduction to the extended Kalman filter, and section 2.3 then draws comparisons between the two filtering methods, particularly highlighting the feedback structure present in both.

2.1 Feedback Particle Filter

In [16, 14], the algorithm for the feedback particle filter (FPF) was introduced. FPF is a novel algorithm for nonlinear filter and employs the principle of feedback (as the algorithm for the EKF does as well). Unlike the EKF though, the FPF is applicable to a general class of nonlinear filtering problems with non-Gaussian posterior distributions. The EKF, as such, is unable to handle these non-Gaussian distributions.

2.1.1 Feedback Particle Filter Algorithm

Feedback particle filter (FPF) is a system of N controlled particles. The state of the filter is $\{X_t^i : 1 \leq i \leq N\}$: The value $X_t^i \in \mathbb{R}^d$ is the state for the i^{th} particle at time t. The dynamics of the i^{th} particle have the following gain feedback form,

$$\frac{\mathrm{d}X_t^i}{\mathrm{d}t} = a(X_t^i) + \dot{B}_t^i + \underbrace{\mathsf{K}_t\, I_t^i}_{\text{(control)}} \tag{2.1}$$

where $\{\dot{B}_t^i\}$ are mutually independent white noise processes with covariance matrix Q, and I_t^i is similar to the innovation error found in the extended Kalman filter,

$$I_t^i := Y_t - \frac{1}{2}(h(X_t^i) + \hat{h}), \tag{2.2}$$

where $\hat{h} := \mathsf{E}[h(X_t^i)|\mathscr{Z}_t]$. In a numerical implementation, this is approximated $\hat{h} \approx N^{-1}\sum_{i=1}^N h(X_t^i) =: \hat{h}^{(N)}$.

The gain function K is found as the solution to an Euler-Lagrange boundary value problem (E-L BVP): for $j = 1, 2, \ldots, m$, the function ϕ_j is a solution to the second-order partial differential equation,

$$\begin{aligned} &\nabla \cdot (p(x,t)\nabla\phi_j(x,t)) = -(h_j(x) - \hat{h}_j)p(x,t), \\ &\int_{\mathbb{R}^d} \phi_j(x,t)p(x,t)\,\mathrm{d}x = 0, \end{aligned} \tag{2.3}$$

where p denotes the conditional distribution of X_t^i given $\mathscr{Z}_t$. In terms of these solutions, the gain function is given by,

$$[\mathsf{K}]_{lj}(x,t) = \sum_{s=1}^m (R^{-1})_{sj}\frac{\partial\phi_s}{\partial x_l}(x,t)\,. \tag{2.4}$$

Denoting $[D\phi] := [\nabla\phi_1, \ldots, \nabla\phi_m]$, where $\nabla\phi_j$ is a column vector for $j \in \{1, \ldots, m\}$, the gain function is succinctly expressed as a matrix product,

$$\mathsf{K} = [D\phi]R^{-1}.$$

It is shown in [16, 13] that the FPF is consistent with the nonlinear filter, given consistent initializations $p(\cdot, 0) = p^*(\cdot, 0)$. Consequently, if the initial conditions $\{X_0^i\}_{i=1}^N$ are drawn from the initial distribution $p^*(\cdot, 0)$ of X_0, then, as $N \to \infty$, the empirical distribution of the particle system approximates the posterior distribution $p^*(\cdot, t)$ for each t.

The main computational burden of the algorithm is the computation/approximation of the gain function at each time t. For the following examples, the gain is restricted to the so-called *constant gain approximation* described in the following section. A more general class of Galerkin algorithms appears in [13].

2.1.2 Constant Gain Approximation

The gain function needs to be computed at each time. For a fixed time t and $j \in \{1,\dots,m\}$, a vector-valued function $\nabla\phi_j(x,t)$ is said to be a weak solution of the BVP (2.3) if

$$\mathsf{E}\left[\nabla\phi_j \cdot \nabla\psi\right] = \mathsf{E}[(h_j - \hat{h}_j)\psi] \tag{2.5}$$

holds for all $\psi \in H^1(\mathbb{R};p)$ where $\mathsf{E}[\cdot] := \int_{\mathbb{R}^d} \cdot\, p(x,t)\,\mathrm{d}x$ and H^1 is a certain Sobolev space (see [13]). The existence-uniqueness result for the weak solution of (2.5) also appears in [13].

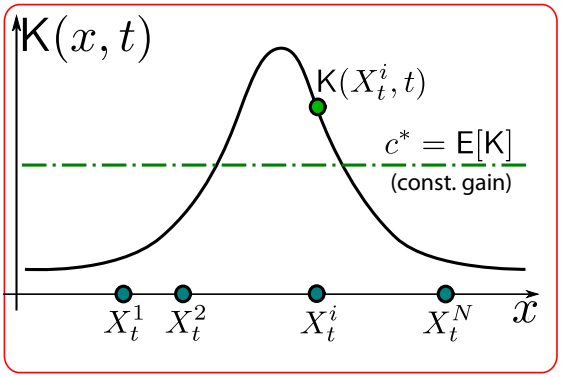

Figure 2.1: Approximation of the function K by its expected value $\mathsf{E}[\mathsf{K}]$ (For illustrative ease, the comparison is shown for the scalar ($d = 1$) case).

In general, the weak solution $\nabla\phi_j(\cdot,t)$ of the BVP (2.5) is some nonlinear vector-valued function of the state (see Fig. 2.1). The idea behind the *constant gain approximation* is to find a constant vector $c_j^* \in \mathbb{R}^d$ to approximate this function (see Fig. 2.1). Precisely,

$$c_j^* = \arg\min_{c_j \in \mathbb{R}^d} \mathsf{E}[|\nabla\phi_j - c_j|^2].$$

By using a standard sum of square argument, the result is:

$$c_j^* = \mathsf{E}[\nabla\phi_j].$$

Even though $\nabla\phi_j$ is unknown, the constant vector c_j^* can be obtained using (2.5).

Specifically, by substituting $\psi(x) = x = (x_1, x_2, \ldots, x_d)$ in (2.5):

$$\mathsf{E}[\nabla\phi_j] = \mathsf{E}[(h_j - \hat{h}_j)\psi] = \int_{\mathbb{R}^d} (h_j(x) - \hat{h}_j)\, x\, p(x,t)\, \mathrm{d}x.$$

In simulations, the last term is approximated using particles:

$$\mathsf{E}[\nabla\phi_j] \approx \frac{1}{N}\sum_{i=1}^{N} \big(h_j(X_t^i) - \hat{h}_j\big)\, X_t^i,$$

which gives the following constant gain approximation:

$$\nabla\phi_j \approx \frac{1}{N}\sum_{i=1}^{N} (h_j(X_t^i) - \hat{h}_j) X_t^i =: c_j^{(N)}. \tag{2.6}$$

Denoting $C := [c_1^{(N)}, \ldots, c_m^{(N)}]$, where $c_j^{(N)}$ is a column vector for $j \in \{1, \ldots, m\}$, the gain function is succinctly expressed as a matrix product,

$$\mathsf{K} = CR^{-1}.$$

It is interesting to note that for the linear-Gaussian case, this constant approximation for the gain function yields the same result as the Kalman gain.

2.1.3 Extensions of the Feedback Particle Filter

There are two extensions of the feedback particle filter:

1. PDA-FPF: In [12, 8], the classical Kalman filter-based probabilistic data association filter (PDAF) is generalized to the nonlinear filtering problem with data association uncertainty. The resulting filter is referred to as the PDA-FPF.

2. IMM-FPF: In [11], the classical Kalman filter-based interacting multiple model filter (IMMF) is generalized to the nonlinear filtering problem with model association uncertainty. The resulting filter is referred to as the IMM-FPF.

The remarkable conclusion of both these papers is that the FPF-based implementations retain the innovation error-based feedback structure even for the nonlinear problem. This structure is expected to be useful because of the coupled nature of the filtering and the data/model association problems. The theoretical

results are illustrated with numerical examples for target tracking applications. For additional details, see [11, 12, 8].

2.2 Extended Kalman Filter

Extended Kalman filter (EKF) is an extension of the Kalman filter algorithm. The algorithm is used to obtain an approximate solution to the nonlinear filtering problem. The EKF approximates the posterior distribution by a Gaussian distribution, parameterized by its mean $\hat{X}_t$ and the covariance matrix P_t.

To perform the update step, the EKF uses linearizations of the signal model $a(\cdot)$ and the observation model $h(\cdot)$, evaluated at the mean $\hat{X}_t$. The respective Jacobian matrices are denoted by $A := \frac{\partial a}{\partial x}(\hat{X}_t)$ and $H := \frac{\partial h}{\partial x}(\hat{X}_t)$.

The EKF algorithm is given by,

$$\frac{\mathrm{d}\hat{X}_t}{\mathrm{d}t} = a(\hat{X}_t) + \mathsf{K}_t \left(Y_t - h(\hat{X}_t)\right), \tag{2.7}$$

$$\frac{\mathrm{d}P_t}{\mathrm{d}t} = AP_t + P_t A^T + Q - \mathsf{K}_t H P_t. \tag{2.8}$$

where the Kalman gain

$$\mathsf{K}_t = P_t H^T R^{-1}. \tag{2.9}$$

Under the assumptions that the signal and the observation models are linear and the posterior distribution is Gaussian, the Kalman filter is the optimal solution. For non-Gaussian and strongly nonlinear problems, the EKF algorithm is known to perform poorly, and can suffer from divergence issues; cf., [7].

2.3 Comparisons Between EKF and FPF

Fig. 2.2 provides a comparison of the feedback structure of the EKF and FPF algorithms. A pseudo-code for the two algorithms is given in Fig. 2.3. Although the FPF pseudo-code is for the constant gain approximation, the two algorithms are quite close even in the general case. The only difference is that in the gen-

eral case, the gain is a function also of the state, as given by the solution of the BVP (2.3), or its weak form (2.5).

2.3.1 Comparison of the Feedback Structure

In recent decades, there have been many important advances in importance sampling based approaches for particle filtering; cf., [4, 2, 10]. A crucial distinction in the feedback particle filter algorithm is that there is no resampling of particles.

It is believed that the introduction of control in the feedback particle filter has several useful features/advantages:

Innovation error. The innovation error-based feedback structure is a key feature of the feedback particle filter (2.1). The innovation error in (2.1) is based on the average value of the prediction $h(X_t^i)$ of the i^{th}-particle and the prediction $\hat{h}^{(N)}$ due to the entire population.

The feedback particle filter thus provides for a generalization of the Kalman filter to nonlinear systems, where the innovation error-based feedback structure of the control is preserved (see Fig. 2.2). For the linear case, the optimal gain function is the Kalman gain. For the nonlinear case, the Kalman gain is replaced by a nonlinear function of the state (see Fig. 2.1).

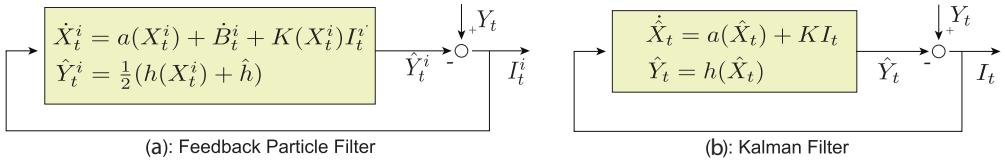

Figure 2.2: Comparison of feedback structure: (a) feedback particle filter and (b) Kalman filter.

Feedback structure. Feedback is important on account of the issue of *robustness*. A filter is based on an idealized model of the underlying dynamic process that is often nonlinear, uncertain and time-varying. The self-correcting property of the feedback provides robustness, allowing one to tolerate a degree of uncertainty inherent in any model.

In contrast, a conventional particle filter is based upon importance sampling. Although the innovation error is central to the Kushner-Stratonovich's stochastic

partial differential equation (SPDE) of nonlinear filtering, it is conspicuous by its absence in a conventional particle filter.

Arguably, the structural aspects of the Kalman filter have been as important as the algorithm itself in design, integration, testing and operation of the overall system. Without such structural features, it is a challenge to create scalable cost-effective solutions.

The "innovation" of the feedback particle filter lies in the (modified) definition of innovation error for a particle filter. Moreover, the feedback control structure that existed thusfar only for Kalman filter now also exists for particle filters (compare parts (a) and (b) of Fig. 2.2).

Does not require resampling. There is no resampling required as in the conventional particle filter. This property allows the feedback particle filter to be flexible with regards to implementation and does not suffer from sampling-related issues.

Variance reduction. Feedback can help reduce the high variance that is sometimes observed in the conventional particle filter. Numerical results in [15] support this claim, where a comparison of the feedback particle filter and the bootstrap filter is provided.

Ease of design, testing and operation. On account of structural features, feedback particle filter-based solutions are expected to be more robust, cost-effective, and easier to debug and implement. Computationally, it is seen that the resampling step in conventional particle filters is very expensive, whereas implementing feedback is seen to be rather efficient.

2.3.2 Comparison of the Computation Time

In this section, a comparison of the number of operations required to implement an FPF algorithm, relative to an EKF algorithm is provided. For the FPF algorithm, a constant gain approximation is assumed.

The comparison, tabulated in Table 1.1, is obtained by counting the number of operations – addition, multiplication and function evaluation – to implement a single update step for a simple scalar valued observation.

With a constant gain approximation, the number of operations scales linearly with the number of particles N, and also with the state dimension d. This is in contrast to EKF where the number of operations scale as d^3, in the number of dimensions.

By counting the addition, multiplication and function evaluation operations, the total number of operations required to implement the update step (per observation) in EKF is,

$$\mathrm{O}^{EKF} = 4d^3 + 12d^2 + 3d + 4. \tag{2.10}$$

For the FPF, it is

$$\mathrm{O}^{FPF} = 3Nd + 6N + 2d + 1. \tag{2.11}$$

Setting the total operation counts in (2.10) and (2.11) equal to each other gives the critical number of particles,

$$N_{\text{crit}} = \frac{4d^3 + 12d^2 + d + 3}{3d + 6}, \tag{2.12}$$

where the FPF and EKF implement the same number of operations.

If one assumes that additions, multiplication and function evaluation each take the same computation time, for $N = N_{\text{crit}}$, the two algorithms have identical computation time requirement. Since the number of operations scale linearly with N, the computation time to implement the update step in FPF then is a factor $\frac{N}{N_{\text{crit}}}$ more than the computation time to implement the update step in EKF.

The accuracy of the FPF improves as the number of particles N increases. The computation time analysis, presented in this section, can be used to carry out performance-computation time trade-off studies, relative to EKF.

Table 2.1: Operation count (per observation) for the FPF

FPF			
Calculation	Adds	Multiplies	Func Eval
$h(X_t^i)$	0	0	N
$\hat{h}^{(N)} = \frac{1}{N}\sum_{i=1}^{N} h(X_t^i)$	N	1	0
$I_t^i = Y_t - \frac{1}{2}(h(X_t^i) + \hat{h}^{(N)})$	2N	N	0
$\mathsf{K} = \frac{1}{N}\sum_{i=1}^{N}(h(X_t^i) - \hat{h}^{(N)})X_t^i$	Nd+N	Nd+2d	0
$U_t^i = \mathsf{K} I_t^i$	0	Nd	0
Total	Nd+4N	2Nd+N+2d+1	N

Table 2.2: Operation count (per observation) for the EKF

EKF			
Calculation	Adds	Multiplies	Func Eval
$A = \frac{\partial a}{\partial x}(\hat{X}_t)$	0	0	d^2
$H = \frac{\partial h}{\partial x}(\hat{X}_t)$	0	0	d
$I_t = Y_t - h(\hat{X}_t)$	1	1	1
$\mathsf{K} = P_t H^T$	d^2	$d^2 + d$	0
$U_t = \mathsf{K} I_t$	0	d	0
P_t	$2d^3 + 6d^2$	$2d^3 + 3d^2$	1
Total	$2d^3 + 7d^2 + 1$	$2d^3 + 4d^2 + 2d + 1$	$d^2 + d + 2$

Algorithm 1 FPF

1: **Iteration** At each time-step t
2: Calculate

$$\hat{h}^{(N)} := \frac{1}{N}\sum_{i=1}^{N} h(X_t^i)$$

3: Calculate the const. approx. of gain function

$$\mathsf{K}_t = \frac{1}{N}\sum_{i=1}^{N}\left(h(X_t^i) - \hat{h}^{(N)}\right) X_t^i,$$

4: **for** $i := 1$ to N **do**
5: Calculate the innovation error

$$I_t^i = Y_t - \frac{h(X_t^i) + \hat{h}^{(N)}}{2}$$

6: Propagate the particle according to the SDE

$$\frac{\mathrm{d}X_t^i}{\mathrm{d}t} = a(X_t^i) + \dot{B}_t^i + \mathsf{K}_t I_t^i$$

7: **end for**

Algorithm 2 EKF

1: **Iteration** At each time-step t
2: Evaluate the Jacobians at $\hat{X}_t$

$$A := \frac{\partial a}{\partial x}(\hat{X}_t) \quad H := \frac{\partial h}{\partial x}(\hat{X}_t)$$

3: Calculate the gain function:

$$\mathsf{K}_t = P_t H^T$$

4: Calculate the innovation error

$$I_t = Y_t - h(\hat{X}_t)$$

5: Propagate the mean

$$\frac{\mathrm{d}\hat{X}_t}{\mathrm{d}t} = a(\hat{X}_t) + \mathsf{K}_t I_t$$

6: Propagate the covariance

$$\frac{\mathrm{d}P_t}{\mathrm{d}t} = AP_t + P_t A^T + Q - \mathsf{K}_t H P_t$$

Figure 2.3: Comparison of the update step for the feedback particle filter, and the extended Kalman filter (For notational ease, a scalar-valued measurement is assumed with observation covariance $R = 1$).

CHAPTER 3

RESAMPLING BASED PARTICLE FILTERS

This chapter provides an introduction to the algorithms used for the resampling based particle filters In particular, the algorithms for particle filters based on sampling importance resampling (SIR), occasional resampling, deterministic resampling, and regularization are discussed.

3.1 Conventional Particle Filters

A conventional particle filter is a simulation-based algorithm to approximate the filtering task. At time t, the state of the filter is $\{(X_t^i, w_t^i) : 1 \leq i \leq N\}$: The value $X_t^i \in \mathbb{R}^d$ is the state and $w_t^i \in [0,1]$ is the weight, for the i^{th} particle at time t. The weights are assumed normalized, i.e., $\sum_{i=1}^N w_t^i = 1$. In terms of these particles,

$$\text{Prob}\{X_t \in A \mid \mathscr{Z}_t\} = \sum_{i=1}^N w_t^i 1\{X_t^i \in A\}.$$

for any measurable set $A \subset \mathbb{R}^d$, where 1 denotes the indicator function. The initial set of particles $\{X_0^i\}_{i=1}^N$ may be drawn i.i.d. from the initial distribution $p^*(\cdot, 0)$ of X_0. In this case the weights are uniform, $w_0^i = \frac{1}{N}$.

3.1.1 Sampling Importance Resampling (SIR)

A conventional particle filter is an algorithm for evolution of the ensemble,

$$(X_t^i, w_t^i) \longrightarrow (X_{t+\delta}^i, w_{t+\delta}^i),$$

as new measurements are obtained; here δ is the time-step. This evolution is carried out in two steps:

1. Prediction: Prediction involves using the SDE model (1.1a) to *push-forward* the particles, $X_t^i \longrightarrow X_{t+\delta}^{i-}$. This is accomplished by numerically integrating the SDE. The weights $w_{t+\delta}^{i-} = w_t^i$.

At the end of prediction step, the particles are denoted as $(X_{t+\delta}^{i-}, w_{t+\delta}^{i-})$.

2. Update: The update step involves application of the Bayes' formula to update the weights. Given a new observation, Y_t, the unnormalized weights are obtained as,

$$\tilde{w}_{t+\delta}^{i-} = w_{t+\delta}^{i-} \, L(Y_t | X_{t+\delta}^{i-}), \tag{3.1}$$

where $L(y|x)$ is the likelihood function, conditional probability of observing $Y_t = y$ given $X_t = x$. The likelihood function may be obtained by using the observation model (1.1b). The weights at time $t+\delta$ are then obtained by normalization,

$$w_{t+\delta}^{i} = \frac{\tilde{w}_{t+\delta}^{i-}}{\sum_{j=1}^{N} \tilde{w}_{t+\delta}^{j-}}. \tag{3.2}$$

This basic algorithm, known at least since 1960s (see [6]), is known to suffer from the issue of *particle degeneracy*. whereby only a few particles have insignificant weight values. This is a problem because it reduces the effective sampling size. The remedy is to occasionally resample in order to 'rejuvenate' the particle population: That is, eliminate particles that have small weights and reproduce particles that have larger weights.

There are several methods for resampling (see [5] for an early reference), some of which are discussed next.

3.1.2 Occasional Resampling

Resampling is carried out periodically, every l^{th} (discrete) time-step; the parameter l is referred to as the *lag parameter*.

In the simplest form of the algorithm, one resamples new particles from the discrete distribution specified by the ensemble $\{(X_t^1, w_t^1), \ldots (X_t^N, w_t^N)\}$. The weights

$\{w_t^1, \cdots w_t^N\}$ are interpreted as a probability mass function for a discrete random variable taking values $\{X_t^1, \cdots X_t^N\}$. After the resampling step, the particles have identical weight $\frac{1}{N}$.

One drawback of this simple algorithm is that random resampling introduces additional noise into the simulation. This is a problem because it can lead to large variance and in some cases, numerical instabilities [3].

To address this issue, one may chose to do resampling in a deterministic manner. An algorithm for this is described next.

3.1.3 Deterministic Resampling

As the name suggests, the particles are resampled in a (partially) deterministic manner.

At each resampling time-step, the i-th particle is 'branched' n_i times, where $n_i = \lfloor N w_t^i \rfloor$. This means: If $n_i > 0$, one creates n_i copies of X_t^i, and if $n_i = 0$ then X_t^i is removed. After this deterministic step, one has $\tilde{N} = \sum_{i=1}^{N} n_i$ particles. Then, $N_{\text{res}} = N - \tilde{N}$ more particles are obtained by using random resampling. For this purpose, the residual weights are defined as $w_t^{i,\text{res}} := w_t^i - \frac{n_i}{N}$. The N_{res} particles are obtained by random sampling from the discrete distribution specified by the ensemble $\{(X_t^1, cw_t^{1,\text{res}}), \ldots (X_t^N, cw_t^{N,\text{res}})\}$, where c is a normalizing constant.

In summary, after each resampling time-step, one obtains a total of N particles of which $\tilde{N}$ are deterministically obtained from the ensemble and N_{res} are randomly generated. The particles have identical uniform weight after the resampling step.

Although these two algorithms alleviate the problem of particle degeneracy, they can introduce the problem of *sample impoverishment* related to the loss of particle diversity. The problem occurs if particles with large weights are selected for resampling many times. In simulations, this can lead to nearly identical values for all particles, particularly if the process noise is small. This problem is addressed by a regularization procedure, which is presented next.

3.1.4 Regularization

A regularized particle filter refers to the algorithm where resampling from the discrete distribution (specified by the ensemble $\{(X_t^1, w_t^1), \ldots (X_t^N, w_t^N)\}$) is replaced by resampling from an absolutely continuous distribution. The continuous distribution is generated by using a kernel density smoother, the details for which can be found in [2, 1].

In the simulations described next, a Gaussian kernel is used. That is the density at time t is approximated as,

$$p(x,t) \approx \sum_{i=1}^{N} w_t^i q^{\varepsilon}(x; X_t^i) =: \tilde{p}(x,t)$$

in which $q^{\varepsilon}(x;\mu) := \frac{1}{\sqrt{2\pi\varepsilon}} \exp(-\frac{(x-\mu)^2}{2\varepsilon})$; in numerical simulations, $\varepsilon = 0.1$ is used.

The regularization may be done at the update or the prediction step. In the simulation, regularization is done at the update step. For this regularized filter, an algorithm for acceptance/rejection of particles is also implemented as described in [2].

CHAPTER 4

APPLICATIONS

In this chapter, comparisons on tracking accuracy and computational efficiency are made between the various filtering methods discussed in earlier chapters. The outline of this chapter is as follows: section 4.1 provides a comparison between the FPF and EKF on a constant velocity target tracking problem, section 4.2 provides a comparison of the FPF and EKF on a maneuvering target tracking problem, and section 4.3 provides a comparison with the particle filters on a benchmark problem defined in [2].

4.1 Constant Velocity Target Tracking

Starting with a basic two-dimensional problem with a simple white noise acceleration model, the target is modeled as,

$$\frac{d}{dt}\begin{pmatrix} x_1(t) \\ x_2(t) \\ v_1(t) \\ v_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ v_1(t) \\ v_2(t) \end{pmatrix}$$

with the target constrained to $x_2 = 0$ and $v_2 = 0$.

Taking bearing-only measurements on this target, after rotating the coordinate frame as,

$$\begin{pmatrix} R_1(t) \\ R_2(t) \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x_1(t) - x_1^b(t) \\ x_2(t) - x_2^b(t) \end{pmatrix}$$

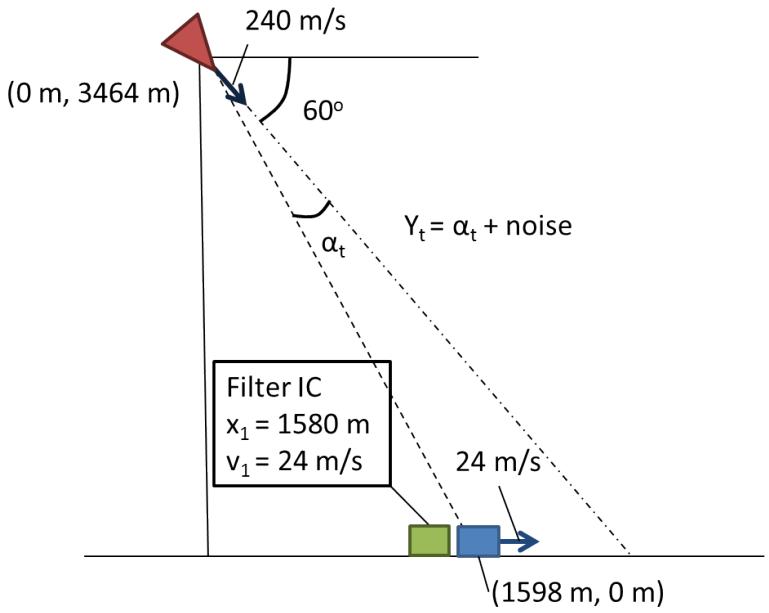

Figure 4.1: Constant velocity target problem setup

and remembering that Y_t is of the form:

$$Y_t = h(X_t, X_t^b) + \sigma_W W_t$$

so here,

$$Y_t = \arctan\left(\frac{R_2(t)}{R_1(t)}\right) + \sigma_W W_t$$

Fig. 4.1 depicts the basic set-up of this constant velocity problem: a target moves along the x direction at a constant velocity, and a sensor moves at a constant velocity towards the target taking bearing-only measurements.

For this experiment a mean-squared error metric is used on position,

$$MSE = \sqrt{\frac{1}{T}\sum_{TI}\left(\frac{x_1 - \hat{x_1}}{x_1}\right)^2}$$

and on velocity,

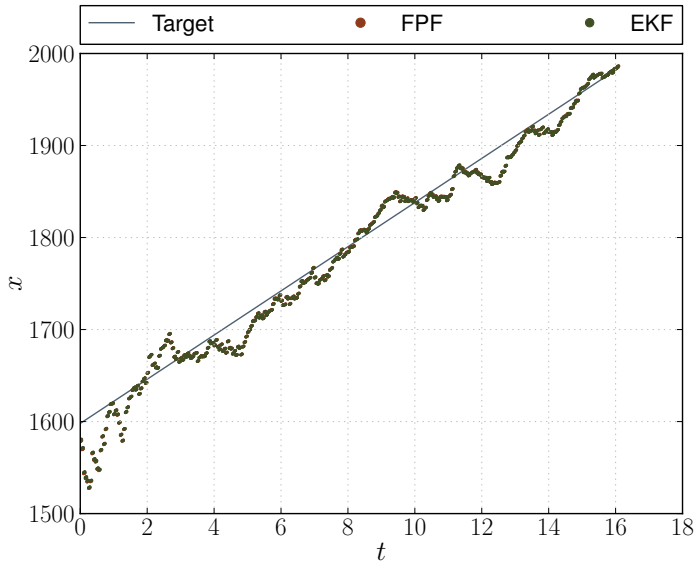

Figure 4.2: Typical position tracking results for constant velocity target

$$MSE = \sqrt{\frac{1}{T}\sum_{TI}\left(\frac{v_1 - \hat{v_1}}{v_1}\right)^2}$$

Given a sampling rate of 20 Hz and an observation noise of 7°, typical results on position tracking as in Fig. 4.2 and for velocity tracking in Fig. 4.3 are given for both the FPF and EKF.

From the typical simulations, it is seen that the FPF and EKF perform nearly identically in tracking accuracy for the constant velocity target problem. This is further backed up by the results from a series of Monte Carlo runs across a range of number of particles for the FPF, sampling rates and observation noises. These results are presented in Fig. 4.4, Fig. 4.5, Fig. 4.6 and Fig. 4.7.

The computation time between the EKF and FPF. is also compared Tables 2.1 and 2.2 show the timings(in microseconds) achieved for this problem from our PYTHON implementation of each filter. Fig. 4.9 and Fig. 4.8 demonstrate the correlation between the timing equilibrium point (where the computational load of the FPF and EKF is equal) found during simulations (approximately 33 particles) and the theoretical operation count equilibrium point (50 particles) found from (1.14).

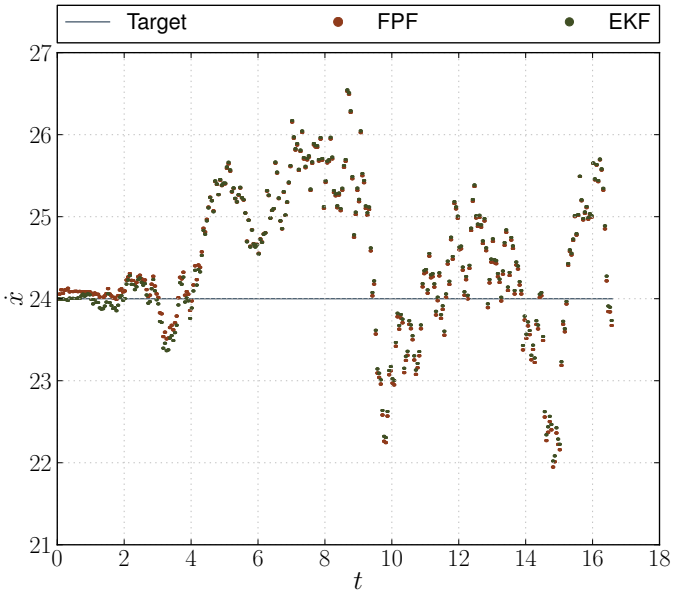

Figure 4.3: Typical velocity tracking results for constant velocity target

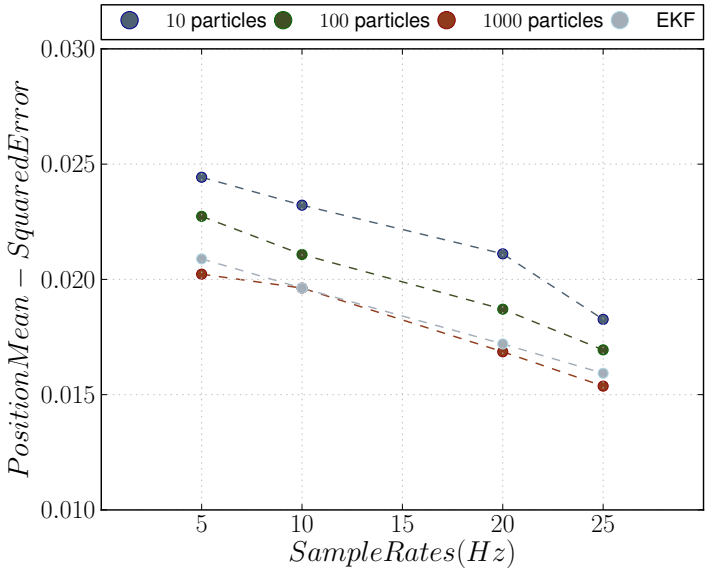

Figure 4.4: Position MSE results across sampling rates

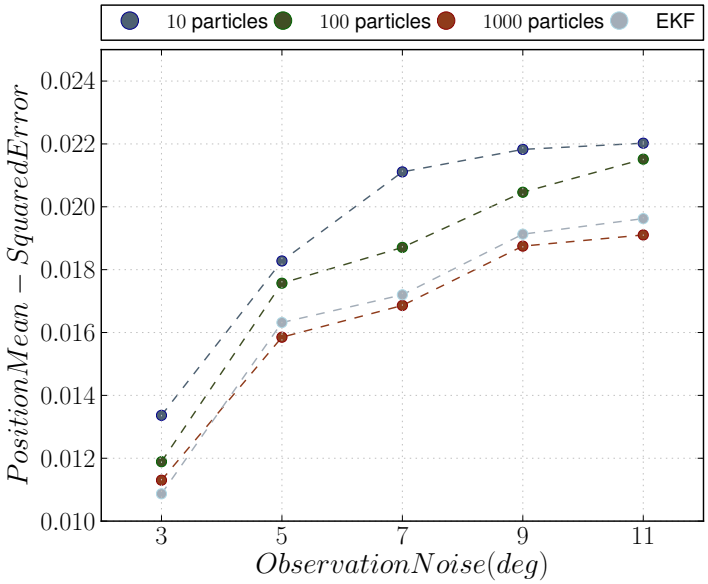

Figure 4.5: Position MSE results across observation noises

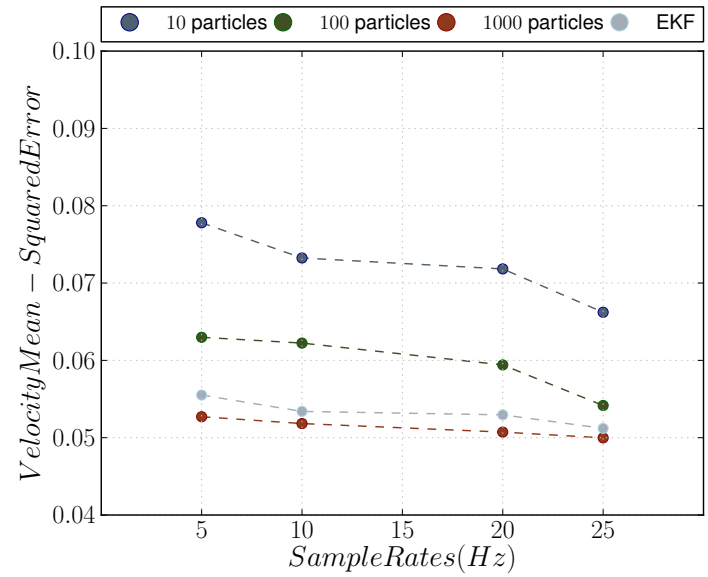

Figure 4.6: Velocity MSE results across sampling rates

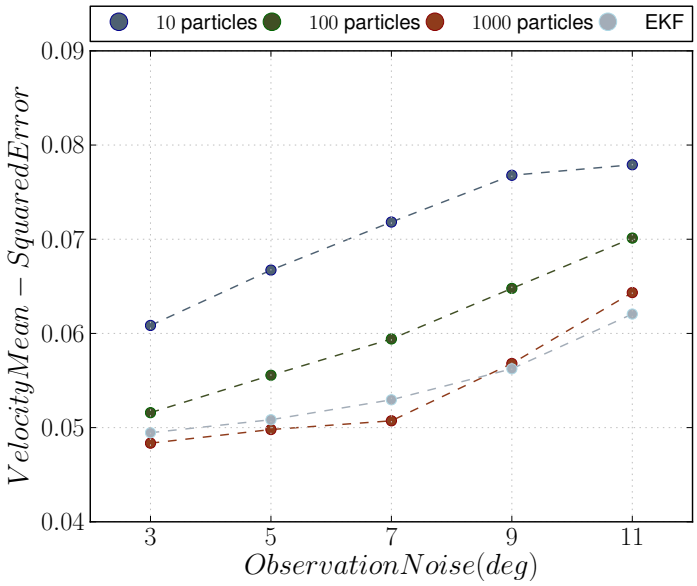

Figure 4.7: Velocity MSE results across observation noises

Table 4.1: FPF Timings

FPF	Equation	10 particles	100 particles	1000 particles
Calculate h^i	$h^i = \arctan\frac{x_2}{x_1}$	32.4	46.5	152.4
Calculate $\hat{h}$	$\hat{h} = \frac{1}{N}\sum_{i=1}^{N} h^i$	68.6	98.9	395.4
Gain	$K = \frac{1}{N}\frac{1}{\sigma_W^2}\sum_{i=1}^{N}(h^i - \hat{h})X^i$	94.1	123.7	465.9
Mean of h^i and $\hat{h}$	$h_{mean} = \frac{h^i + \hat{h}}{2}$	98.8	162.7	797.4
Innovation Error	$I^i = Y_t - h_mean$	35.1	68.4	388.8
Control	$U^i = K * I^i$	8.3	9.9	21.1
Total		337.3	510.1	2221.0

Table 4.2: EKF Timings

EKF	Equation	Time
Predict Covariance	$P = P + FP + PF^T + Q$	18.0
Jacobian	H= $C^b(t)[\frac{-x_2}{x_1^2+x_2^2}, \frac{x_1}{x_1^2+x_2^2}, 0]$	283.9
Kalman Gain	$K = \frac{1}{\sigma_W^2}PH^T$	14.4
Innovation Error	$I = Y^t - h_x$	47.6
Control	$U = K * I$	6.7
Update Covariance	$P = P - KHP$	11.2
Total		381.8

For additional comparison, two more test cases were investigated. The first case

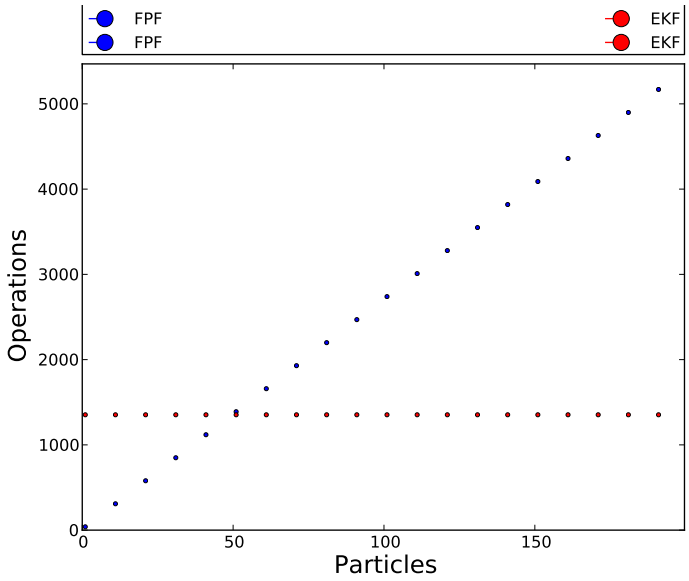

Figure 4.8: Total operations for FPF (scaled with particles) and EKF

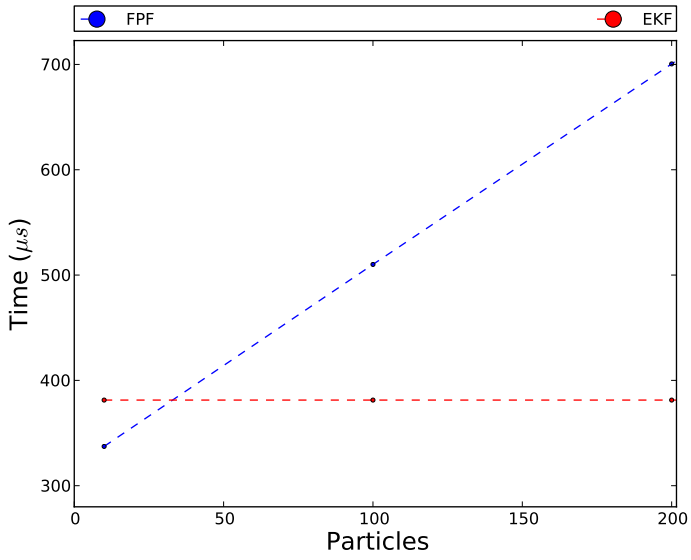

Figure 4.9: Total times for FPF (linear fit between particles) and EKF

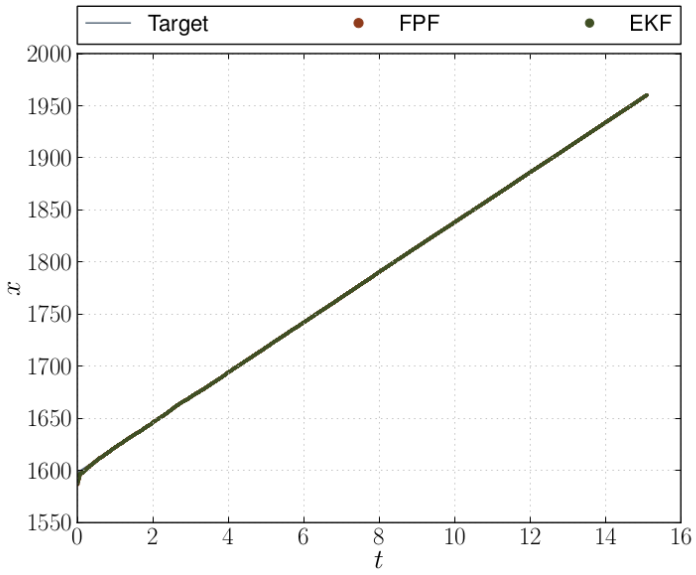

Figure 4.10: Position tracking results on a constant velocity target with a low noise bearing sensor

was the same simulation set-up as before, but the noise on the bearing sensor was reduced to 0.3°. The improved results here (Fig. 4.10, Fig. 4.11 demonstrate that the poor velocity tracking in the previous case was likely due to lack of information/observability from the higher noise sensor.

The second additional case investigated is the same scenario as above, with the addition of a range sensor providing measurements to the filter. As with the low noise bearing sensor case, the improved results in this case (Fig. 4.12, Fig. 4.13), demonstrate that the lack of information/observability in the bearing-only measurement case is likely what was causing some of the poor velocity tracking results.

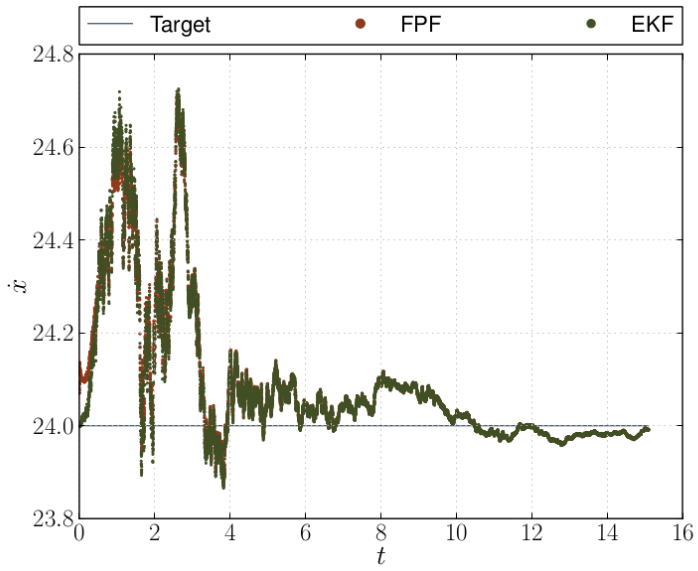

Figure 4.11: Velocity tracking results on a constant velocity target with a low noise bearing sensor

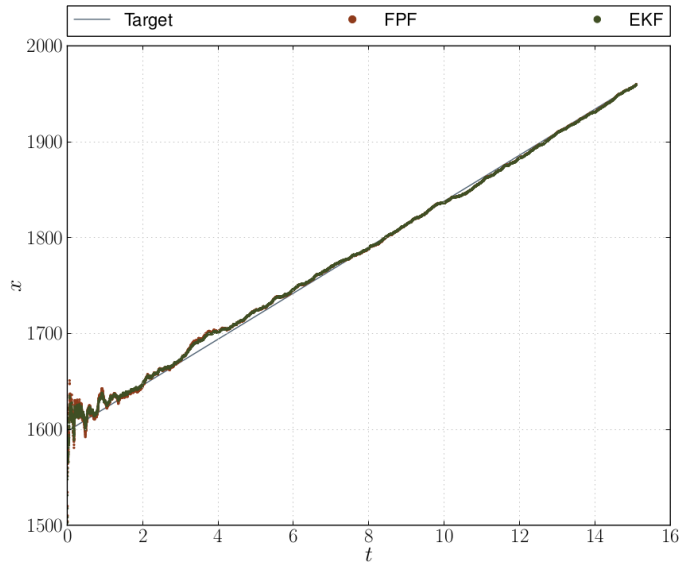

Figure 4.12: Position tracking results on a constant velocity target with range and bearing sensors

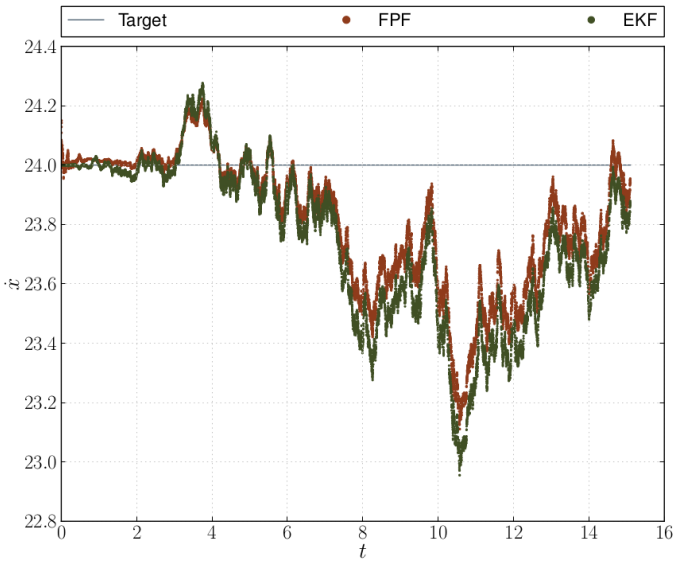

Figure 4.13: Velocity tracking results on a constant velocity target with range and bearings sensors

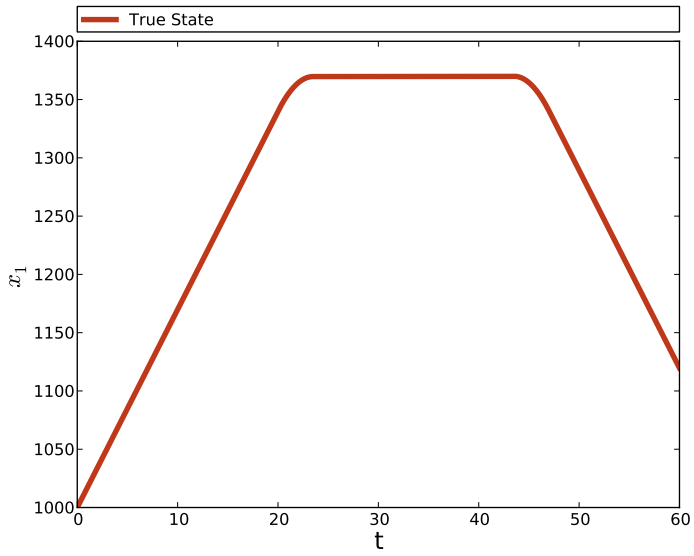

Figure 4.14: Position path taken by target with maneuvers

4.1.1 Maneuvering Target Tracking

Here, a similar set-up as the constant velocity model is used, except the target is able to make a series of maneuvers: in this case two turns. It is assumed that the maneuvers are unknown to the filters. The position and velocity profiles used for the simulations are seen in Fig. 4.14 and Fig. 4.15.

In a naive approach, the same constant velocity models as the previous problem were used and the results can be observed in Fig. 4.18, Fig. 4.19. It can be seen that tracking accuracy has diminished severely for both filters, so a new approach must be taken to regain some of the performance. As a secondary approach to the maneuvering target problem, a constant acceleration model was implemented without much improvement over the constant velocity model (Fig. 4.16, Fig. 4.17).

For the extended Kalman filter, the constant velocity model is replaced with the Singer jerk model:

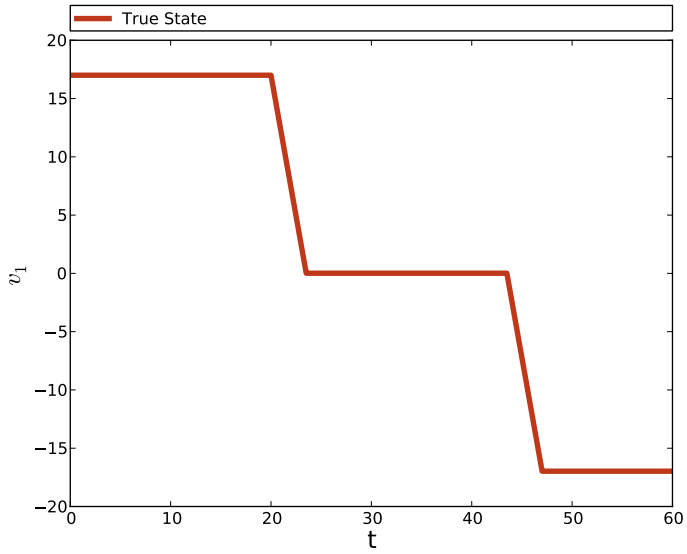

Figure 4.15: Velocity profile of target with maneuvers

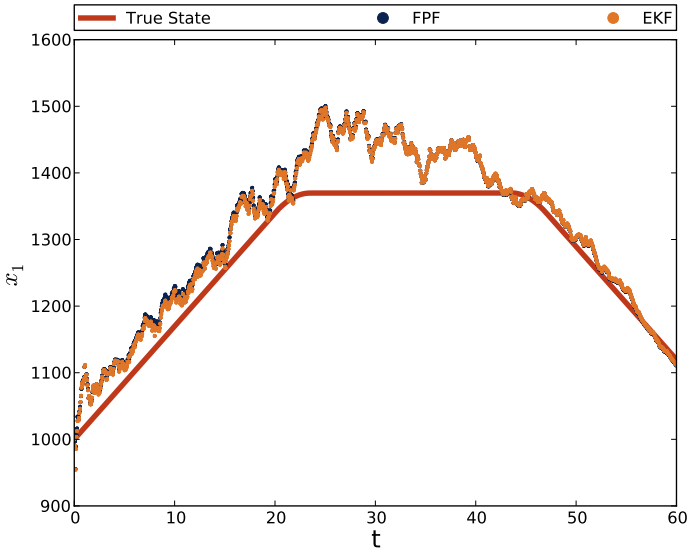

Figure 4.16: Position tracking results on a maneuvering target with a constant acceleration model for the filter

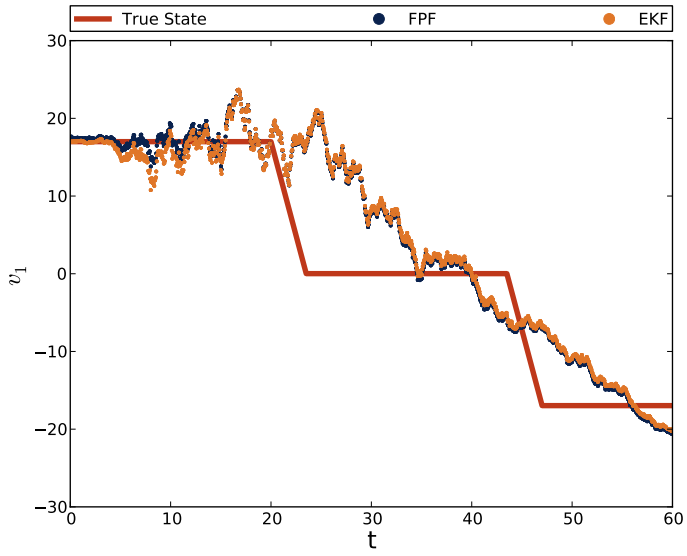

Figure 4.17: Velocity tracking results on a maneuvering target with a constant acceleration model for the filter

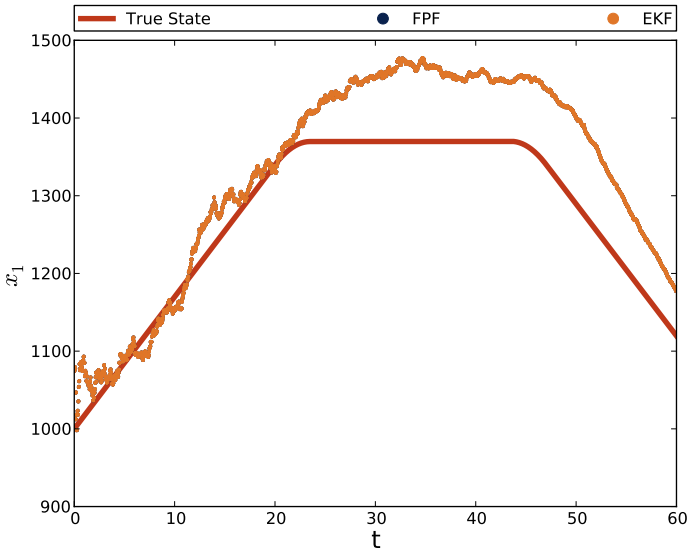

Figure 4.18: Position tracking results on a maneuvering target with a constant velocity model

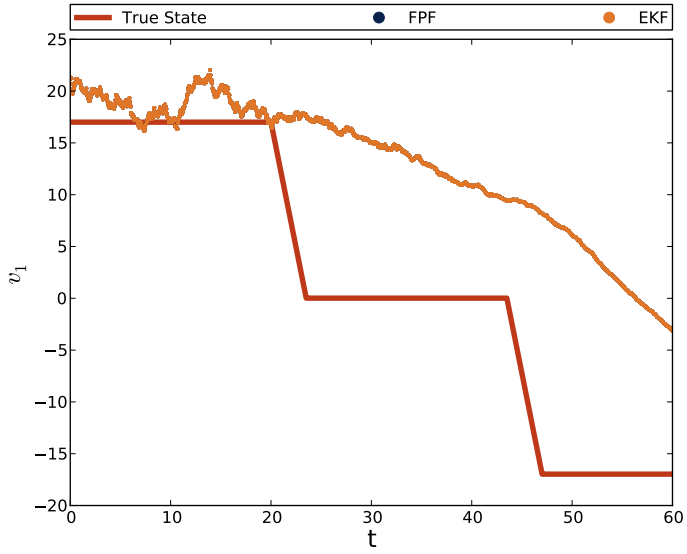

Figure 4.19: Velocity tracking results on a maneuvering target with a constant velocity model

$$\frac{d}{dt}\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ v_1(t) \\ v_2(t) \\ v_3(t) \\ a_1(t) \\ a_2(t) \\ a_3(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ v_1(t) \\ v_2(t) \\ v_3(t) \\ a_1(t) \\ a_2(t) \\ a_3(t) \end{pmatrix} + noise$$

Here τ is chosen such that it is equal to the expected length of the maneuver. In the Singer model the variance on the noise is given by

$$\sigma_m^2 = \frac{a_{max}^2}{3}[1 + 4P_{max} - P_0]$$

Where a_{max} is the maximum acceleration, P_{max} is the probability the target moves with that acceleration, and P_0 is the probability the target moves with no acceleration.

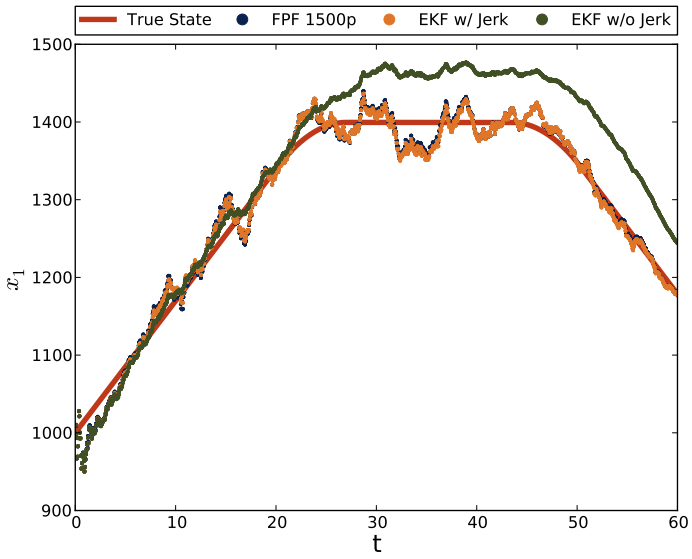

Figure 4.20: Position tracking results on a maneuvering target with a Singer model on EKF and synchronized FPF

For the FPF, a technique referred to as synchronization is introduced. The concept of synchronization in the context of the FPF involves distributing particles over a variety of potential maneuver models; by then taking advantage of the feedback structure present in the FPF, the mean state of the particles will *synchronize* around the particles maneuvering most like the actual target is. For the purpose of the experiments presented here, a uniform distribution over the accelerations [-5,5] was used. However, the flexibility of using particles allows many different options when it comes to selecting a distribution (multi-modal Gaussian, delta functions at specific maneuvers, etc.).

Fig. 4.20 and Fig. 4.21 show typical tracking results on a maneuvering target for the EKF with a constant velocity model, EKF with a Singer model, and synchronized FPF. As before, the constant velocity shows poor tracking capabilities across both position and velocity. Interestingly, the EKF with Singer model and the synchronized FPF show nearly identical and much improved tracking performance across both position and velocity. Fig. 4.22 and Fig. 4.23 further exemplify this improvement by showing a factor of nearly two reduction in the average MSE over a set of Monte Carlo runs.

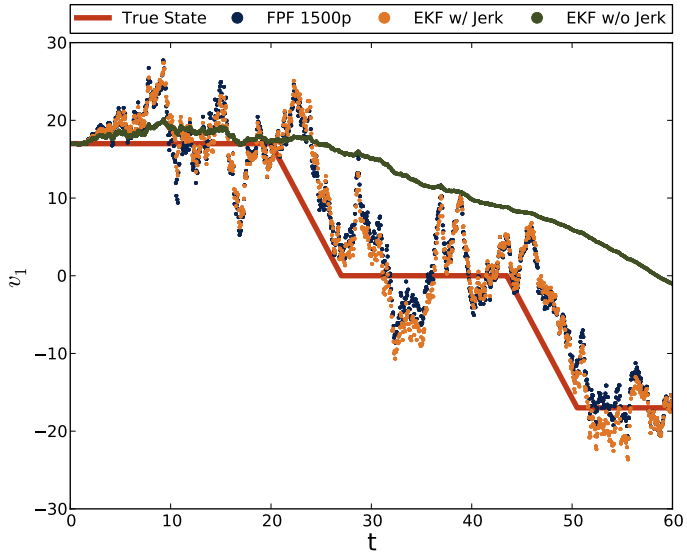

Figure 4.21: Velocity tracking results on a maneuvering target with a Singer model on EKF and synchronized FPF

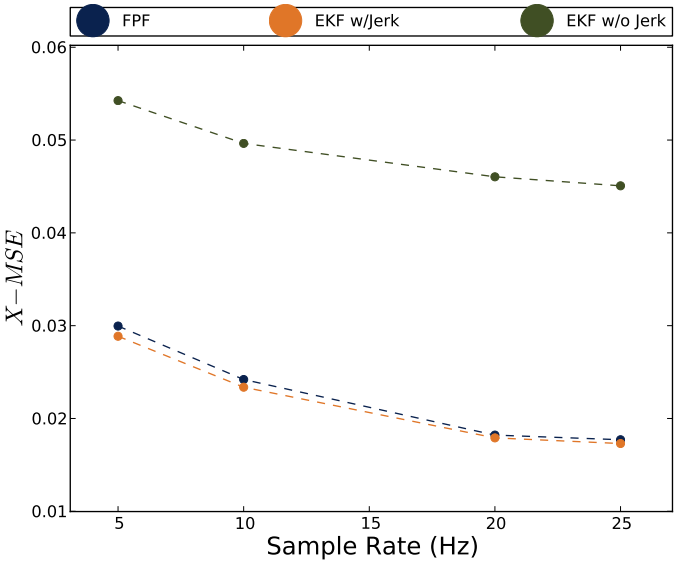

Figure 4.22: Position MSE results on a maneuvering target with a Singer model on EKF and synchronized FPF

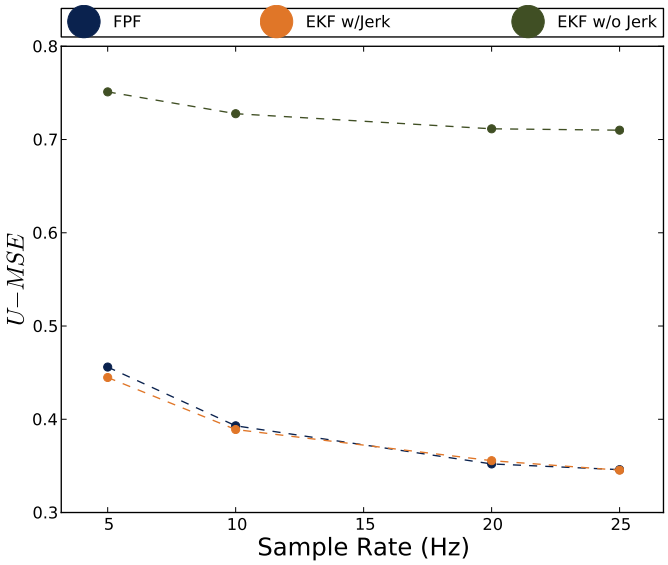

Figure 4.23: Velocity MSE results on a maneuvering target with a Singer model on EKF and synchronized FPF

4.2 Ship Tracking

For this example problem, the FPF and EKF are compared to the conventional, resampling based particle filters. The example problem is taken from the survey article [2].

In the example problem, the dynamics describe the motion of a ship. The ship moves with a constant radial and angular velocity, perturbed by white noise, when it is within some distance of the origin. If the ship drifts too far away from the origin, a restoring force pushes it back towards the origin. The signal model for the state process $X_t = [X_{t,1}, X_{t,2}]^T \in \mathbb{R}^2$ is described by,

$$\begin{aligned}\frac{\mathrm{d}X_{t,1}}{\mathrm{d}t} &= -X_{t,2} + f_1(X_{t,1}, X_{t,2}) + \dot{B}_{t,1},\\ \frac{\mathrm{d}X_{t,2}}{\mathrm{d}t} &= X_{t,1} + f_2(X_{t,1}, X_{t,2}) + \dot{B}_{t,2},\end{aligned}$$

where $\dot{B}_{t,1}, \dot{B}_{t,2}$ are independent white noise processes, and

$$f_i(x) \doteq \gamma \frac{x_i}{|x|^2} - \Theta \frac{x_i}{|x|} 1_{(\rho,\infty)}(|x|), \quad i = 1,2,$$

where $|x| = \sqrt{x_1^2 + x_2^2}$, $1_{(\rho,\infty)}$ denotes the indicator function on the set $(\rho,\infty) \subset \mathbb{R}$,

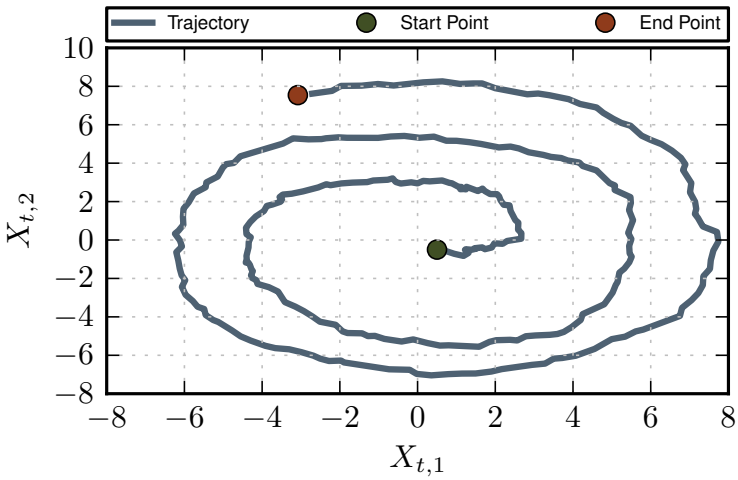

Figure 4.24: Typical trajectory for ship

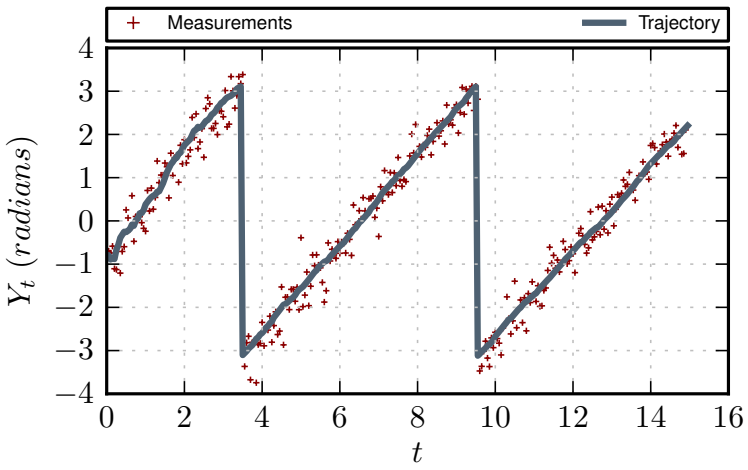

Figure 4.25: Typical angle measurements seen versus time

and γ, Θ, and ρ are real-valued parameters.

Noisy angular measurements are sampled at time-intervals of $\delta = 0.05$, according to the observation model,

$$Y_t = h(X_t) + \theta V_t, \tag{4.1}$$

where V_t are sampled i.i.d. from a standard Gaussian distribution $\mathcal{N}(0,1)$, independent of $(X_0, \dot{B}_t)$ and $h(x_1, x_2) \doteq \arctan(x_2/x_1)$. For the numerical simulations, choose $\theta = 0.32$, which represents approximately 18° of standard deviation for the (Gaussian) observation noise.

A single trajectory along with a sequence of measurements is depicted in Fig.(). The initial condition $X_0 = (0.5, -0.5) =: x_0$ and the parameters $\gamma = 2$, $\Theta = 50$, $\rho = 9$. This trajectory was obtained by using a predictor-corrector Euler scheme for time-discretization of the ODE. A fixed discretization time-step of $\delta = 0.05$ was used for this as well as for all other numerical simulations reported here.

Next, theresults of the numerical experiments are presented. In these experiments, 100 distinct signal trajectories and measurement profiles were generated over the time interval $[0, 8.25]$.

For each of the 100 Monte-Carlo runs, the initial condition of the ship's trajectory was randomly sampled from the prior distribution $p_0 = \mathcal{N}(x_0, 10)$, where $x_0 = [0.5, -0.5]$. The process and the observation noise was generated for each run in an independent manner.

The filters are initialized with a prior Gaussian distribution p_0. That is, the EKF is initialized with $\hat{X}_0 = x_0$ and $P_0 = 10$. For the particle filters, the initial conditions of the particles, $\{X_0^i\}_{i=1}^N$, are randomly sampled from the distribution p_0.

The following metrics are used for the comparisons:

Root mean square error (rmse): Computed as a measure of performance over all trajectories and over all time instants,

$$\text{rmse} := \frac{1}{100}\frac{1}{165}\sum_{j=1}^{100}\sum_{k=1}^{165} |X^j(k\delta) - \hat{X}^j(k\delta)|$$

where $X^j, j = 1, \cdots, 100$ represents the signal trajectory, $X^j(k\delta)$ is the true state

at time instant $k\delta$, $\hat{X}^j(k\delta)$ is the state estimate obtained as a mean, and 165 is the total number of time instances during each simulation run ($\delta = 0.05$, $T = 165\delta = 8.25$).

Mean computation time: Computed as the mean time (in milliseconds) it took to perform a single update step. The mean is obtained over the 100 Monte-Carlo runs. The computation times are obtained by using a numerical profiler in the PYTHON programming environment.

Fig. 4.26 shows a typical trajectory for the ship with the estimates provided by the FPF and the EKF also being displayed. The results are further quantified in Table 2.3. The trends shown in the Table 2.3 are consistent with the trends reported in [2] for the particle filters. The quantitative numbers are also quite close. The accuracy of the estimate, in terms of the rmse, improves by using sophisticated versions of resampling schemes. The penalty is the computation time, which increases as the rmse improves, with the FPF being the exception: it has the best rmse and the lowest computation time among the particle filters. The results for the regularization particle filter are not included because it became computationally prohibitive to evaluate the Gaussian kernels for the 100 Monte-Carlo runs.

The results with the EKF algorithm, as reported here, are not consistent with [2]. In that paper, the EKF was reported to have very poor performance. it was found instead that the performance of EKF was in fact comparable to FPF, and better than other particle filtering approaches.

Table 4.3: Performance Comparison

Filter Type	rmse	Comp. time
EKF	1.0143	0.062
PF_SIR	1.2902	0.690
PF_Resample	1.0991	1.448
PF_Resample_Lag	1.0856	1.077
PF_Deterministic	1.0677	1.557
Feedback PF	0.9901	0.202

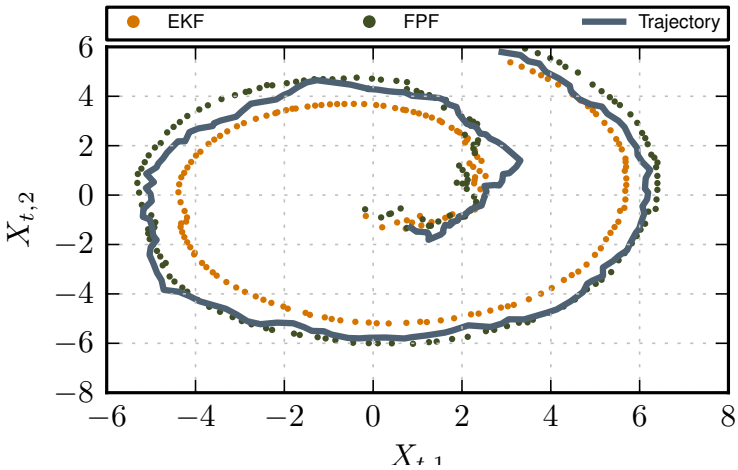

Figure 4.26: Typical ship trajectory and the estimates of the state provided by FPF and EKF in the $x_1 - x_2$ plane

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Comparative studies of various nonlinear filtering techniques were provided. FPF is shown to provide for a generalization of the Kalman filter to a general class of nonlinear non-Gaussian problems. FPF inherits many of the properties that has made the Kalman filter so widely applicable over the past five decades, including innovation error and the feedback structure (see Fig. 2.2).

Comparisons using the FPF with a constant approximation for the gain against the EKF, demonstrated a striking parity between the two filters. However, as the nonlinearity and non-Gaussianity of the problems increase and the performance of the EKF is expected to drop, it is suspected the FPF will continue to perform strongly through the use of alternative, more complex, approximations to the gain function and through intuitive use of particle distributions (as in the maneuvering target problem).

Comparisons with several particle filtering algorithms are also discussed. The results of the numerical example, taken from the survey paper [2], are encouraging. These numerical results show that – for this particular example problem – relative to conventional particle filtering algorithms, the FPF algorithm can provide better or comparable performance at a fraction of the computational cost.

Feedback is important on account of the issue of robustness. In particular, feedback can help reduce the high variance that is sometimes observed in the conventional particle filter. Even more significantly, the structural aspects of the Kalman filter have been as important as the algorithm itself in design, integration, testing and operation of a larger system involving filtering problems (e.g., navigation systems). It is expected for the FPF to similarly provide for an integrated framework, now for nonlinear non-Gaussian problems.

Future work will focus on further applications where the feedback structure of the FPF could potentially be beneficial, as well as further development of the algo-

rithms to work towards better approximations to the solution of the EL-BVP that provides the gain function for the FPF. Other work within the research group has focused on the use of filtering with oscillators with some interesting applications.

REFERENCES

[1] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174 –188, feb 2002.

[2] A. Budhiraja, L. Chen, and C. Lee. A survey of numerical methods for nonlinear filtering problems. *Physica D: Nonlinear Phenomena*, 230(1):27–36, 2007.

[3] F. Daum and J. Huang. Generalized particle flow for nonlinear filters. *In Proc. of SPIE*, 7698, 2010.

[4] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte-Carlo Methods in Practice*. Springer-Verlag, April 2001.

[5] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140(2):107–113, 1993.

[6] J. E. Handschin and D. Q. Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage nonlinear filtering. *International Journal of Control*, 9(5):547–559, 1969.

[7] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, Boston, MA, 2004.

[8] A. K. Tilton, T. Yang, H. Yin, and P. G. Mehta. Feedback particle filter-based multiple target tracking using bearing-only measurements. In *Proc. 15th Int. Conf. on Inf. Fusion*, pages 2058–2064, Singapore, July 2012.

[9] G. Welch and G Bishop. An introduction to the kalman filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, July 2006.

[10] J. Xiong. Particle approximations to the filtering problem in continuous time. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

[11] T. Yang, H. A. P. Blom, and P. G. Mehta. Interacting multiple model-feedback particle filter for stochastic hybrid systems. In *Submitted to* 52^{nd} *IEEE Conf. Decision and Control*, Firenze, Italy, December 2013.

[12] T. Yang, G. Huang, and P. G. Mehta. Joint probabilistic data association-feedback particle filter for multi-target tracking application. In *Proc. of the 2012 American Control Conference*, pages 820–826, Montréal, Canada, June 2012.

[13] T. Yang, R. S. Laugesen, P. G. Mehta, and S. P. Meyn. Multivariable feedback particle filter. In *Proc. of* 51^{st} *IEEE Conf. Decision and Control*, pages 4063–4070, Maui, HI, Dec 2012.

[14] T. Yang, P. G. Mehta, and S. P. Meyn. Feedback particle filter with mean-field coupling. *In Proc. of IEEE Conference on Decision and Control*, pages 7909–7916, December 2011.

[15] T. Yang, P. G. Mehta, and S. P. Meyn. Feedback particle filter with mean-field coupling. In *Proc. of* 50^{th}*IEEE Conf. Decision and Control*, pages 7909–7916, Orlanda, FL, December 2011.

[16] T. Yang, P. G. Mehta, and S. P. Meyn. A mean-field control-oriented approach to particle filtering. *In Proc. of American Control Conference*, pages 2037–2043, June 2011.