A Comparative Study of Nonlinear Filtering Techniques

Adam K. Tilton, Shane Ghiotto and Prashant G. 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 paper 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.

A secondary purpose of this paper is to provide a summary of the FPF algorithm, that can aid practitioners to rapidly implement the algorithm. A detailed algorithm (pseudo-code) is included, and compared against an EKF algorithm. Such comparisons also help highlight the feedback structure of the FPF algorithm.

I. Introduction

In a recent work, we introduced a new feedback control-based formulation of the particle filter for the nonlinear filtering problem [20], [18]. The aim of this paper is to describe, in a comparative manner, some results on theory and applications of the feedback particle filter (FPF).

The problem under consideration is a standard continuous-time nonlinear filtering problem:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = a(X_t) + \dot{B}_t, \tag{1a}$$

$$Y_t = h(X_t) + \dot{W}_t, \tag{1b}$$

where $X_t \in \mathbb{R}^d$ is the state at time t, $Y_t \in \mathbb{R}^m$ is the observation , $a(\,\cdot\,)$, $h(\,\cdot\,)$ are C^1 functions, and $\{\dot{B}_t\}$, $\{\dot{W}_t\}$ are mutually independent white noise processes of appropriate dimension. The covariance matrix of the observation noise $\{\dot{W}_t\}$, and that of the process noise $\{\dot{B}_t\}$ are both assumed to be positive definite. These matrices are denoted as R and Q, respectively (see Table I). The function h is a column vector whose j-th coordinate is denoted as h_j (i.e., $h = (h_1, h_2, \ldots, h_m)^T$).

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

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

Financial support from the AFOSR grant FA9550-09-1-0190 and the NSF CPS grant 0931416 is gratefully acknowledged.

A. K. Tilton, S. Ghiotto and P. G. Mehta are with the Coordinated Science Laboratory and the Department of Mechanical Science and Engineering at the University of Illinois at Urbana-Champaign (UIUC) {atilton2, ghiotto1, mehtapg}@illinois.edu

TABLE I. Notation for signal and observation models

Signal model		Observation model	
State process	X_t	Observation process	Y_t
Process noise	$\dot{B}_t$	Observation noise	$\dot{W}_t$
Noise covariance	Q	Noise covariance	R

The filter is infinite-dimensional since it defines the evolution, in the space of probability measures, of $\{p^*(\,\cdot\,,t) : t \ge 0\}$. If $a(\,\cdot\,)$, $h(\,\cdot\,)$ are linear functions, the solution is given by the finite-dimensional Kalman filter. The article [2] surveys methods to approximate the nonlinear filter. Two approaches described in this survey are the extended Kalman filter and the particle filter.

Feedback particle filter (FPF) is a novel algorithm for nonlinear filtering that is based on the principle of feedback (as the EKF algorithm is). The FPF algorithm, however, is applicable to a general class of nonlinear non-Gaussian filtering problems. Nonlinearity here refers to the nonlinearity of the functions $a(\cdot)$ and $h(\cdot)$ in the signal and the observation models. The 'non-Gaussianity' refers to the posterior distribution being non-Gaussian. The FPF algorithm was introduced in [20], [18]. In the present paper, we discuss theory and applications of the FPF algorithm, by providing comparisons with both the EKF and the conventional resampling-based particle filters.

The comparison with the EKF is used to highlight the feedback structure of the FPF algorithm. Also computational cost estimates are discussed, in terms of number of operations relative to EKF.

The comparison with the conventional particle filtering approaches is based on a numerical example taken from the survey article [2]. Comparisons are provided for both computational cost and accuracy. Consistent with the conclusions of [2], all particle filters are able to accurately estimate the state. However, the FPF is seen to have better accuracy at lower computational cost, relative to the other particle filtering approaches.

Apart from the FPF algorithm presented in this paper, there has been growing interest in control-based approaches to nonlinear filtering. Some related approaches appear in [3], [9], [4], [8], [10], [12].

The remainder of this paper is organized as follows: Sec II describes the basic FPF algorithm. Sec III provides some comparisons with the EKF, vis-a-vis the feedback structure. Sec IV provides a summary of the resampling based approaches to particle filtering. The numerical example appears in Sec V, where comparisons between the various particle filtering approaches are provided. The conclusions appear in Sec VI.

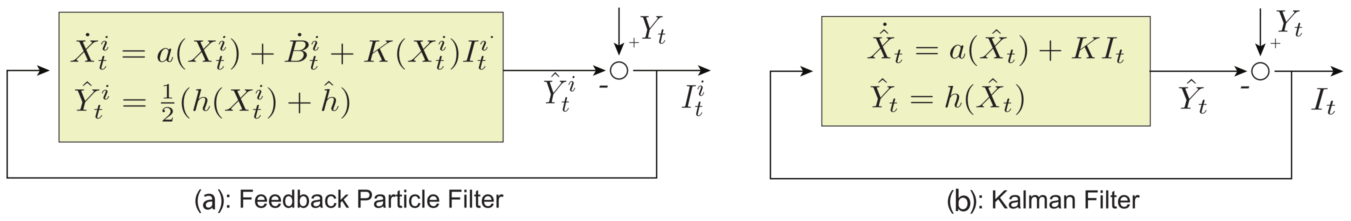

$\dot{X}_t^i = a(X_t^i) + \dot{B}_t^i + K(X_t^i) I_t^i$
$\hat{Y}_t^i = \frac{1}{2}(h(X_t^i) + \hat{h})$
Y_t
+
-
$\hat{Y}_t^i$
I_t^i
(a): Feedback Particle Filter
$\dot{\hat{X}}_t = a(\hat{X}_t) + K I_t$
$\hat{Y}_t = h(\hat{X}_t)$
Y_t
+
-
$\hat{Y}_t$
I_t
(b): Kalman Filter

Fig. 1. Comparison of feedback structure: (a) feedback particle filter and (b) Kalman filter.

II. Feedback Particle Filter

A. Feedback Particle Filter Algorithm

Feedback particle filter (FPF) is a controlled system. The state of the filter is $\{X_t^i : 1 \le i \le N\}$: 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,

$$\frac{\mathrm{d}X_t^i}{\mathrm{d}t} = a(X_t^i) + \dot{B}_t^i + \underbrace{\mathsf{K}_t\, I_t^i}_{\text{(control)}} \tag{2}$$

where $\{\dot{B}_t^i\}$ are mutually independent white noise processes with covariance matrix Q, and I_t^i is a modified version of the *innovation process* that appears in the nonlinear filter,

$$I_t^i := Y_t - \frac{1}{2}(h(X_t^i) + \hat{h}), \tag{3}$$

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

The gain function K is obtained as a 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,

$$\begin{aligned} &\nabla\cdot(p(x,t)\nabla\phi_j(x,t)) = -(h_j(x) - \hat{h}_j)p(x,t), \\ &\int_{\mathbb{R}^d} \phi_j(x,t)p(x,t)\,\mathrm{d}x = 0, \end{aligned} \tag{4}$$

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,

$$[\mathsf{K}]_{lj}(x,t) = \sum_{s=1}^m (R^{-1})_{sj} \frac{\partial\phi_s}{\partial x_l}(x,t)\,. \tag{5}$$

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 [20], [17] 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. In this paper, we restrict ourselves to the so-called *constant gain approximation* described in the following section. A more general class of Galerkin algorithms appears in [17].

B. Constant Gain Approximation

The gain function needs to be computed at each time. For a fixed time t and $j \in \{1,\ldots,m\}$, a vector-valued function $\nabla\phi_j(x,t)$ is said to be a weak solution of the BVP (4) if

$$\mathsf{E}[\nabla\phi_j\cdot\nabla\psi] = \mathsf{E}[(h_j - \hat{h}_j)\psi] \tag{6}$$

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 [17]). The existence-uniqueness result for the weak solution of (6) also appears in [17].

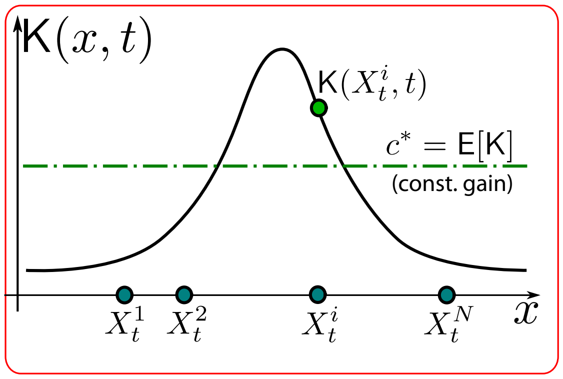


Fig. 3. Approximation of the function K by its expected value $\mathsf{E}[\mathsf{K}]$ (For illustrative ease, the comparison is shown for the scalar ($d = 1$) case).

In general, the weak solution $\nabla\phi_j(\cdot,t)$ of the BVP (6) is some nonlinear vector-valued function of the state (see Fig. 3). 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. 3). 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, we have

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

Even though $\nabla\phi_j$ is unknown, the constant vector c_j^* can be obtained using (6). Specifically, by substituting $\psi(x) = x = (x_1,x_2,\ldots,x_d)$ in (6):

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

In simulations, we approximate the last term using particles:

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

```
Algorithm 1 Feedback Particle Filter
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 Extended Kalman Filter
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
       $\frac{\mathrm{d}P_t}{\mathrm{d}t} = AP_t + P_t A^T + Q - \mathsf{K}_t H P_t$
```

Fig. 2. 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 $R = 1$).

which gives the following constant gain approximation:

$$\nabla\phi_j \approx \frac{1}{N}\sum_{i=1}^{N}(h_j(X_t^i) - \hat{h}_j)X_t^i =: c_j^{(N)}. \tag{7}$$

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,

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

In the remainder of this paper, we will consider only this constant gain approximation of the gain function.

C. Extensions of the Basic FPF Algorithm

There are two extensions of the feedback particle filter:

1. PDA-FPF: In [16], [13], we generalized the classical Kalman filter-based probabilistic data association filter (PDAF) to the nonlinear filtering problem with data association uncertainty. The resulting filter is referred to as the PDA-FPF.

2. IMM-FPF: In [15], we generalized the classical Kalman filter-based interacting multiple model filter (IMMF) 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 [15], [16], [13].

III. Comparison with Extended Kalman Filter

A. 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,

$$\frac{\mathrm{d}\hat{X}_t}{\mathrm{d}t} = a(\hat{X}_t) + \mathsf{K}_t\left(Y_t - h(\hat{X}_t)\right), \tag{8}$$

$$\frac{\mathrm{d}P_t}{\mathrm{d}t} = AP_t + P_t A^T + Q - \mathsf{K}_t H P_t. \tag{9}$$

where the Kalman gain

$$\mathsf{K}_t = P_t H^T R^{-1}. \tag{10}$$

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., [11].

TABLE II. NOTATION

Feedback particle filter		Kalman filter	
Particle state	X_t^i	EKF estimate	$\hat{X}_t$
FPF gain (7)	K_t	Kalman gain	K_t
Innovation error	I_t^i	Innovation error	I_t
		Covariance	P_t

Figure 1 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, with notation in Table II. 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 general case, the gain is a function also of the state, as given by the solution of the BVP (4), or its weak form (6).

B. Comparison of the Feedback Structure

In recent decades, there have been many important advances in importance sampling based approaches for particle filtering; cf., [5], [2], [14]. A crucial distinction in the feedback particle filter algorithm is that there is no resampling of particles.

We believe 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). The innovation error in (2) 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. 1). 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. 3).

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. 1).

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 [19] 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.

C. Comparison of the Computation Time

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

The comparison, tabulated in Table III, 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,

$$\mathrm{O}^{EKF} = 4d^3 + 12d^2 + 3d + 4. \tag{11}$$

For the FPF, it is

$$\mathrm{O}^{FPF} = 3Nd + 6N + 2d + 1. \tag{12}$$

Setting the total operation counts in (11) and (12) equal to each other gives the critical number of particles,

$$N_{\text{crit}} = \frac{4d^3 + 12d^2 + d + 3}{3d + 6}, \tag{13}$$

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 III. COMPARISON OF THE OPERATION COUNT (PER OBSERVATION) BETWEEN FPF AND EKF

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

IV. SAMPLING BASED APPROACHES

A conventional particle filter is a simulation-based algorithm to approximate the filtering task. At time t, the state of the filter is $\{(X_t^i, w_t^i) : 1 \le i \le N\}$: 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, $w_0^i = \frac{1}{N}$.

A. 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 (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,

$$\tilde{w}_{t+\delta}^{i-} = w_{t+\delta}^{i-} L(Y_t | X_{t+\delta}^{i-}), \tag{14}$$

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 (1b). The weights at time $t + \delta$ are then obtained by normalization,

$$w_{t+\delta}^i = \frac{\tilde{w}_{t+\delta}^{i-}}{\sum_{j=1}^N \tilde{w}_{t+\delta}^{j-}}. \tag{15}$$

This basic algorithm, known at least since 1960s (see [7]), 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 [6] for an early reference), some of which are discussed next.

B. 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 [4].

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

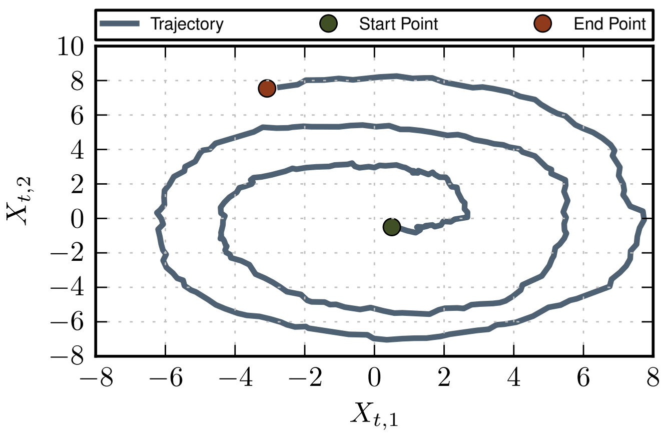
C. Deterministic Resampling

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

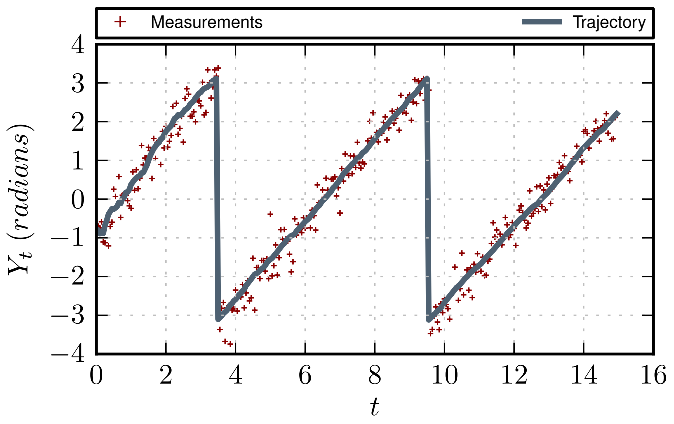
At each resampling time-step, the i-th particle is 'branched' n_i times, where $n_i = \lfloor N w_t^i \rfloor$. This means: If $n_i > 0$, one creates n_i copies of X_t^i, and if $n_i = 0$ then X_t^i is removed. After this deterministic step, one has $\tilde{N} = \sum_{i=1}^N n_i$ particles. Then, $N_{\text{res}} = N - \tilde{N}$ 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 $\{(X_t^1, c w_t^{1,\text{res}}), \ldots (X_t^N, c w_t^{N,\text{res}})\}$, 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*



(a)



(b)

Fig. 4. Numerical results for a typical simulation: (a) Signal trajectory in the (x_1, x_2)-plane; and (b) Angular measurements for the signal trajectory.

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.

D. 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].

V. Numerical Results

In this section, we describe the numerical results for a nonlinear example problem borrowed from the survey article [2]. In their survey article, the authors show that the particle filtering approaches can significantly outperform the EKF.

In the present paper, we extend the comparison to now include the FPF. We show that the FPF algorithm is both more accurate and more computationally efficient than the other particle filtering methods. The results are obtained using the constant gain approximation of the gain function.

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,

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

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 $(\rho, \infty) \subset \mathbb{R}$, 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,

$$Y_t = h(X_t) + \theta V_t, \tag{16}$$

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, we chose $\theta = 0.32$, 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. 4(a)-4(b). 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.

We next present the results of the numerical experiments. 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.

Table IV provides performance comparisons between the different particle filters, with $N = 500$ particles. 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.

TABLE IV. PERFORMANCE COMPARISON

Filter Type	rmse	Comp. time
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

The trends shown in the Table IV 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.

TABLE V. GAIN CAP COMPARISON

Gain Cap	FPF rmse	EKF rmse
8	1.0111	1.0386
9	0.9901	1.0143
10	1.0164	1.0497
15	1.0464	1.8380
20	1.0667	2.5390

We next discuss comparisons with the EKF algorithm. In numerical implementation it was found that the gain, for both EKF and FPF algorithm, was very large at the onset of simulation. This led to numerical instabilities for the chosen (fixed) discretization time-step $\delta = 0.05$. In order to avoid these numerical issues, the gains were capped at $\mathsf{K} = \pm 9$ for both the FPF and the EKF algorithms (The numbers reported in Table V for FPF are with this gain cap). Fig. 5(a) depicts the results from a typical simulation: part (a) provides a comparison of the estimate obtained using the EKF and the FPF algorithms, part (b) depicts the gains as a function of time. Note the gains are saturated for the first two seconds, and are within the $[-9,9]$ range thereafter.

The gain cap of ± 9 was found to be the value that avoided the numerical instabilities and minimized the rmse for both the EKF and the FPF algorithms. This value was found through our investigation of the effect of the gain cap on filter performance, with fixed $\delta = 0.05$. The performance comparison, as a function of gain cap, are tabulated in Table V.

We also carried out a limited set of numerical simulations with an adaptive time-stepping scheme (without the gain cap). The rmse values with these simulations were approximately the same as those obtained using the gain cap of ± 9.

We also investigated the accuracy of the FPF algorithm as a function of the number of particles used. These results are tabulated in Table VI.

Note that all the results reported here (Table IV-VI) were obtained for the same set of 100 Monte-Carlo runs (trajectories and measurement profiles).

TABLE VI. PARTICLE COMPARISON

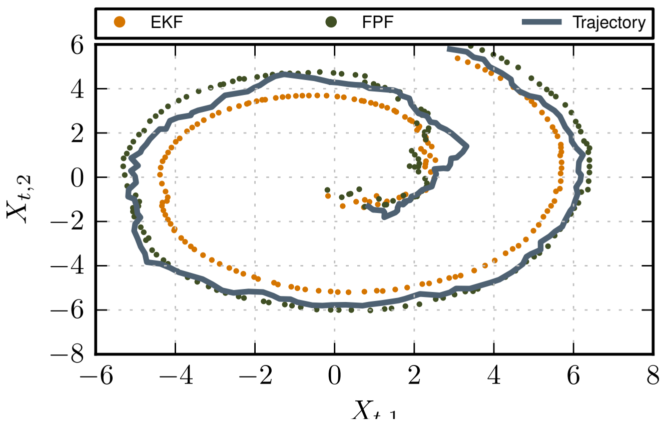
# of Particles	rmse	Comp. time
10	1.0906	0.056
50	0.9985	0.070
100	0.9981	0.086
500	0.9901	0.202
1000	0.9883	0.346

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. We found instead that the performance of EKF was in fact comparable to FPF, and better than other particle filtering approaches.

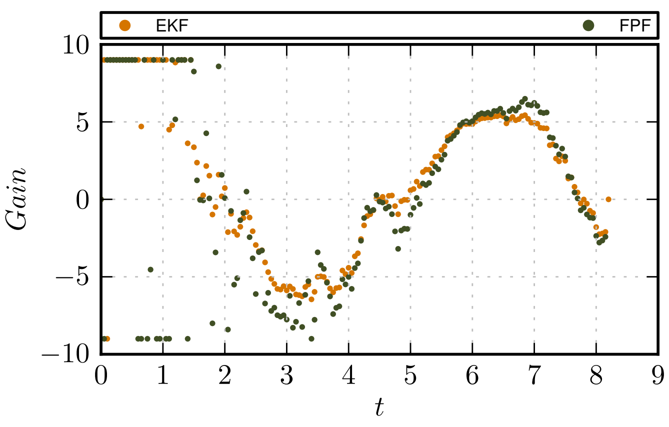
VI. CONCLUSIONS

In this paper, we provided some comparative studies with the feedback particle filter (FPF). 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. 1).

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.



(a)



(b)

Fig. 5. Numerical results for a typical simulation: (a) Signal, FPF, and EKF trajectories in the (x_1,x_2)-plane; and (b) Gain profiles with a gain cap of 9.

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). We expect FPF to similarly provide for an integrated framework, now for nonlinear non-Gaussian problems.

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] D. Crisan and J. Xiong. Approximate McKean-Vlasov representations for a class of SPDEs. *Stochastics: An International Journal of Probability and Stochastic Processes*, pages 1–16, 2009.

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

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

[6] 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.

[7] 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.

[8] R. Ma and T. P. Coleman. Generalizing the posterior matching scheme to higher dimensions via optimal transportation. In *Allerton Conference on Communication, Control, and Computing*, pages 96–102, 2011.

[9] S. K. Mitter and N. J. Newton. A variational approach to nonlinear estimation. *SIAM J. Cont. Opt.*, 42(5):1813–1833, 2003.

[10] S. Pequito, A. P. Aguiar, B. Sinopoli, and P. A. Gome. Nonlinear estimation using mean field games. In *NetGCOOP 2011 : International conference on Network Games, Control and Opt.*, 2011.

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

[12] G. Terejanu, P. Singla, T. Singh, and P. D. Scott. Uncertainty propagation for nonlinear dynamical systems using Gaussian mixture models. *AIAA Journal of Guidance, Control and Dynamics*, 31(6):1623–1633, Nov. 2008.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.