Feedback Particle Filter with Data-Driven Gain-Function Approximation

Berntorp, K.; Grover, P.

TR2018-034 February 19, 2018

Abstract

This paper addresses the continuous-discrete time nonlinear filtering problem for stochastic dynamical systems using the feedback particle filter (FPF). The FPF updates each particle using feedback from the measurements, where the gain function that controls the particles is the solution of a Poisson equation. The main difficulty in the FPF is to approximate this solution using the particles that approximate the probability distribution. We develop a novel Galerkin-based method inspired by high-dimensional data-analysis techniques. Based on the time evolution of the particle cloud we determine basis functions for the gain function and compute values of it for each individual particle. Our method is completely adapted to the recorded history of the particles and the update of the particles do not require further intermediate approximations or assumptions. We provide an extensive numerical evaluation of the proposed approach and show that it compares favorably compared to baseline FPF and particle filters based on the importancesampling paradigm.

IEEE transactions on aerospace and electronic systems

Feedback Particle Filter with Data-Driven Gain-Function Approximation

Karl Berntorp[1] and Piyush Grover[1]

Abstract—**This paper addresses the continuous-discrete time nonlinear filtering problem for stochastic dynamical systems using the feedback particle filter (FPF). The FPF updates each particle using feedback from the measurements, where the gain function that controls the particles is the solution of a Poisson equation. The main difficulty in the FPF is to approximate this solution using the particles that approximate the probability distribution. We develop a novel Galerkin-based method inspired by high-dimensional data-analysis techniques. Based on the time evolution of the particle cloud we determine basis functions for the gain function and compute values of it for each individual particle. Our method is completely adapted to the recorded history of the particles and the update of the particles do not require further intermediate approximations or assumptions. We provide an extensive numerical evaluation of the proposed approach and show that it compares favorably compared to baseline FPF and particle filters based on the importance-sampling paradigm.**

I. INTRODUCTION

Particle filters (PFs) [1]–[3] are popular for estimation of nonlinear systems. Traditional PFs based on importance sampling generate random state trajectories and assign a weight to each state trajectory according to how well it predicts the observations. PFs have been successful in numerous applications, see [4]–[8] for some examples, and PFs have also been shown to be an integral part in nonlinear system identification [9], [10]. One problem with traditional PFs is the inevitable particle degeneracy [11] (i.e., only a few particles, or even one, have nonzero weight). Degeneracy leads to decreased performance, or even filter divergence. To mitigate this, PFs include a resampling step where trajectories are either kept or discarded, depending on their weight. The resampling step makes PFs practically useful, but introduces other negative effects, such as sample impoverishment and increased variance [2].

The feedback particle filter (FPF) has been introduced in a series of papers as a control-oriented, resampling-free, variant of the PF [12]–[15]. The FPF applies a feedback structure to each particle, visualized in Fig. 1. It can be viewed as a generalization of the Kalman filter to PFs. The measurement update is implemented as a gradual transition from prior to posterior, instead of the one-step multiplication of Bayes' rule in conventional importance-sampling based PFs. Numerical studies in [16], [17] have demonstrated significant performance improvements over conventional PFs. The gain function that is present in the feedback structure is in general nonlinearly dependent on the state and found as a solution to a constrained Poisson's equation [18]. Usually, approximate solutions are necessary, because closed-form expressions can only be computed in certain special cases.

The contribution in this paper is a data-driven approach for computation of the gain function in the FPF, which is applicable to a range of estimation problems. Our approach is motivated by the following observation: the ensemble of particles accumulated over time describes how the system evolves, and therefore gives information about how the particles should be controlled to explain the measurements. Inspired by proper orthogonal decomposition (POD) as a high-dimensional data-analysis technique, we approximate the gain function based on a series expansion of basis functions that are extracted from the time evolution of the particle cloud. We leverage the Galerkin approach [14], which is a method for converting problems involving continuous operators (such as boundary value problems) to the discrete domain, by converting the equation to the *weak domain* and characterize the solution by a set of basis functions.

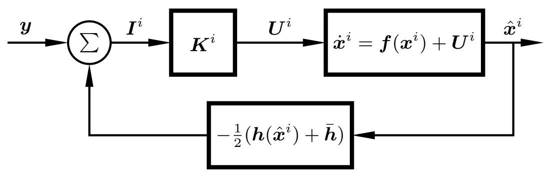
A. Related Work

Several papers have addressed gradual transitioning of the prior to posterior. In [19], a framework for gradual transition from prior to posterior was introduced—see also [20], [21]. The particle-flow filter has been introduced and improved in a series of papers, see, for example, [22]–[24], and [25], [26] consider particle flow in an importance-sampling based framework.

There are few papers addressing the control-gain approximation in the FPF. In [16], we demonstrated how a sensible approximation of the gain function can increase performance compared with baseline FPF for a specific system. Gain computation for an artificial, scalar example was considered in [14], and [27] reported on an initial study using a kernel-based approach for gain function approximation, which does not use basis functions. Furthermore, [28] makes a connection between the FPF and optimal transport, although restricted to linear systems.

A preliminary version of this work was presented in [29]. This paper contains several extensions. Specifically, we give a more detailed and rigorous exposition of the proposed method, and we provide additional numerical examples to illustrate the efficiency of the approach. In particular, using a linear Gaussian example, we show that our gain-computation method approaches the optimal gain as the number of particles $N \to \infty$.

[1] The authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA. Email: `karl.o.berntorp@ieee.org`, `grover@merl.com`



y
Σ
I^i
K^i
U^i
$\dot{x}^i = f(x^i) + U^i$
$\hat{x}^i$
$-\frac{1}{2}(h(\hat{x}^i) + \bar{h})$

Fig. 1. Simplified block diagram of the FPF. It uses feedback gains $\{\boldsymbol{K}^i\}_{i=1}^N$ to control the particles $\{\boldsymbol{x}^i\}_{i=1}^N$. This is in contrast to the conventional PF, where only the particles' weights are changed in the measurement update.

B. Notation

Vectors and matrices are denoted with bold-face letters as $\boldsymbol{x}$ and $\boldsymbol{A}$, respectively, where $\boldsymbol{a}_j$ is the jth column of $\boldsymbol{A}$. The jth element of $\boldsymbol{x}$ is denoted by x_j and A_{ij} means the element of $\boldsymbol{A}$ on row i, column j. The variables t and k are reserved for continuous time and discrete time step, respectively. With $\delta(\boldsymbol{x}-\boldsymbol{y})$ we mean the Dirac delta mass, which is one when $\boldsymbol{x}=\boldsymbol{y}$ and zero elsewhere. The conditional probability density function of $\boldsymbol{x}$ given $\boldsymbol{y}$ is denoted by $p(\boldsymbol{x}|\boldsymbol{y})$, $\mathbb{E}(\boldsymbol{x}) = \int \boldsymbol{x} p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, and $\bar{\boldsymbol{x}} = 1/N \sum_{i=1}^N \boldsymbol{x}^i$ for a finite positive integer N. Let $L^2(\mathbb{R}^n, p)$ mean the Hilbert space of square-integrable functions with respect to p at a given time and let $X := L^2(\mathbb{R}^n)$. Furthermore, $\nabla \boldsymbol{f}$ is the gradient of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$. The notation $H^1(\mathbb{R}^n, p)$ means the function space where the function and its first derivative (defined in expectation) are in $L^2(\mathbb{R}^n, p)$. The inner product between $\boldsymbol{u} := \boldsymbol{u}(\boldsymbol{x})$ and $\boldsymbol{v} := \boldsymbol{v}(\boldsymbol{x})$ is $\langle \boldsymbol{u}, \boldsymbol{v}\rangle := \int \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}$. The induced norm is $\|\boldsymbol{u}\| := \sqrt{\langle \boldsymbol{u}, \boldsymbol{u}\rangle}$. In $\mathbb{R}^n$, $\|\boldsymbol{u}\|_2 := \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}$. Finally, $\boldsymbol{0}_{n\times 1}$ is the $n \times 1$ zero matrix.

C. Outline

We state the problem scope in Sec. II and give a short background on the FPF in Sec. III. Sec. IV outlines the proposed method for gain computation, which is numerically evaluated in Sec. V. Finally, Sec. VI summarizes and draws conclusions.

II. PROBLEM FORMULATION

This paper is concerned with continuous-discrete time systems of the form

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t))\mathrm{d}t + \mathrm{d}\boldsymbol{\beta}(t), \tag{1a}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k, \tag{1b}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state at time $t \in \mathbb{R}$; $\boldsymbol{y}_k := \boldsymbol{y}(t_k) \in \mathbb{R}^m$ is the discrete-time measurement at time t_k; $\boldsymbol{f}_t := \boldsymbol{f}(\boldsymbol{x}(t))$ and $\boldsymbol{h}_k := \boldsymbol{h}(\boldsymbol{x}_k)$ are the drift and measurement function, respectively; and $\boldsymbol{\beta}(t)$ and $\boldsymbol{e}_k$ are process and measurement noise, respectively. The aim in continuous-discrete time Bayesian filtering is to estimate the posterior filtering density $p(\boldsymbol{x}(t)|\mathcal{Y}_k)$, or at least the relevant moments, at each time $t \geq t_k$. Here, $\mathcal{Y}_k := \{\boldsymbol{y}_0, \dots, \boldsymbol{y}_k\}$ denotes the set of measurements, obtained at discrete time steps. Throughout, both process and measurement noise are assumed independent, Gaussian distributed with zero mean and known covariance matrices $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$.

In this paper, we focus our efforts on the Bayesian filtering problem using the FPF. More specifically, given the prior $p(\boldsymbol{x}_k|\mathcal{Y}_{k-1})$ and a current measurement $\boldsymbol{y}_k$, our goal is to determine the feedback gains $\boldsymbol{K}^i$ in Fig. 1 that control the particles to approximate the posterior $p(\boldsymbol{x}_k|\mathcal{Y}_k)$.

III. BACKGROUND: FEEDBACK PARTICLE FILTER

This section gives a brief overview of the main steps in the FPF and a popular implementation of the FPF. For details about the general formulation of the FPF, see [14], [15], and [13] for the continuous-discrete formulation.

A. Feedback Particle Filter

The key step in Bayesian filtering is Bayes' rule, which states that

$$p(\boldsymbol{x}_k|\mathcal{Y}_k) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\mathcal{Y}_{k-1}). \tag{2}$$

In conventional PFs, the measurement update is implemented as a point-wise multiplication between likelihood and prior, where the prior is represented by a set of N weighted particles, where the weights are computed using the likelihood conditioned on the respective particle. The FPF approximates the posterior with N unweighted samples, or particles, $\boldsymbol{x}_t^i$ as

$$p(\boldsymbol{x}_k|\mathcal{Y}_k) \approx \hat{p}(\boldsymbol{x}_k|\mathcal{Y}_k) = \frac{1}{N}\sum_{i=1}^N \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \tag{3}$$

Note the difference to conventional PFs, where weights are used to select the importance of the particles. The FPF treats the evolution of the posterior distribution as a controlled system, where the state evolves in two alternating steps. For increasing $k = 0, 1, \dots,$

1) the continuous-time dynamics (1a) govern the evolution of the state between two measurement times $t \in [t_{k-1}, t_k)$.
2) the Bayesian update (2) is simulated using a closed-loop system model at discrete time t_k.

Fig. 1 provides the conceptual structure of the FPF, which is similar to that of the Kalman filter. The main differences to a Kalman filter are that N particles are controlled instead of only the mean, that $\boldsymbol{K}$ in general is a nonlinear function of the state, and that the error combines local and global information.

At time t_k, a new observation $\boldsymbol{y}_k$ arrives. To incorporate $\boldsymbol{y}_k$, a particle flow $\{\boldsymbol{S}_k^i(\lambda)\}_{i=1}^N$ defined by differential equations is introduced,

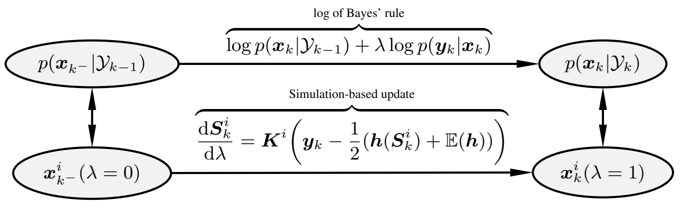
$$\frac{\mathrm{d}\boldsymbol{S}_k^i(\lambda)}{\mathrm{d}\lambda} = \boldsymbol{K}(\boldsymbol{S}_k^i(\lambda), \lambda)\boldsymbol{I}_k^i + \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{S}_k^i(\lambda), \lambda), \tag{4}$$

with initial condition $\boldsymbol{S}_k^i(0) = x_{k^-}^i$, for $i = 1, \dots, N$, where

$$x_{k^-} := \lim_{t \to t_k} x(t), \tag{5}$$

and where the limit approaches from below. The parameter $\lambda \in [0, 1]$ is the *pseudo-time*. The term $\boldsymbol{\Omega} \in \mathbb{R}^n$ is referred to as the *Wong-Zakai correction term* [30] and is for each element Ω_s calculated as

$$\Omega_s(\boldsymbol{x}, \lambda) = \frac{1}{2}\sum_{d=1}^n \sum_{j=1}^m K_{dj}(\boldsymbol{x}, \lambda)\frac{\partial K_{sj}}{\partial x_d}(\boldsymbol{x}, \lambda), \tag{6}$$



log of Bayes' rule
$\log p(\boldsymbol{x}_k|\mathcal{Y}_{k-1}) + \lambda \log p(\boldsymbol{y}_k|\boldsymbol{x}_k)$
$p(\boldsymbol{x}_{k^-}|\mathcal{Y}_{k-1})$
$p(\boldsymbol{x}_k|\mathcal{Y}_k)$
Simulation-based update
$\frac{\mathrm{d}\boldsymbol{S}_k^i}{\mathrm{d}\lambda} = \boldsymbol{K}^i\Big(\boldsymbol{y}_k - \frac{1}{2}(\boldsymbol{h}(\boldsymbol{S}_k^i) + \mathbb{E}(\boldsymbol{h}))\Big)$
$\boldsymbol{x}_{k^-}^i(\lambda = 0)$
$\boldsymbol{x}_k^i(\lambda = 1)$

Fig. 2. Illustration of how the pseudo-time update corresponds to the prior and posterior.

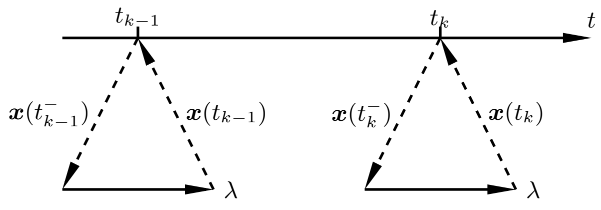


Fig. 3. Illustration of the measurement update in the FPF [13]. The state $\boldsymbol{x}$ is predicted up to t_{k-1}. When $\boldsymbol{y}_{k-1}$ arrives, the predicted state estimate is corrected using a simulation-based update, going from $\lambda = 0$ to $\lambda = 1$, yielding $\boldsymbol{x}(t_{k-1})$. The process is similar for t_k.

where K_{sj} is element sj of $\boldsymbol{K}$. The term $\boldsymbol{I}_k^i$ is the innovation error and equals

$$\boldsymbol{I}_k^i = \boldsymbol{y}_k - \frac{1}{2}\left(\boldsymbol{h}(\boldsymbol{S}_k^i(\lambda)) + \mathbb{E}(\boldsymbol{h}(\boldsymbol{S}_k(\lambda)))\right), \tag{7}$$

where

$$\mathbb{E}(\boldsymbol{h}(\boldsymbol{S}_k(\lambda)) \approx \bar{\boldsymbol{h}}_k = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}(\boldsymbol{S}_k^i(\lambda)). \tag{8}$$

The innovation process (7) includes the predicted measurement of particle i and the average of all particles in the particle flow. Corresponding to the particle flow, a density function $\rho_k(\boldsymbol{x}, \lambda)$ that defines the distribution of $\{\boldsymbol{S}_k^i(\lambda)\}_{i=1}^N$ is also introduced. The particle-flow update (4) is made possible by a log-homotopy transformation [13], which transforms the discrete-time Bayesian measurement update to a continuously evolving process. Fig. 2 illustrates how the homotopy transformation enables continuously moving from prior to posterior. It was shown in [13] that the gain function is a solution to a certain Euler-Lagrange boundary-value problem. Specifically, for each fixed $\lambda \in [0, 1]$, the gain function

$$\boldsymbol{K} = \begin{bmatrix}\nabla\phi_1(\boldsymbol{x}, \lambda) & \cdots & \nabla\phi_m(\boldsymbol{x}, \lambda)\end{bmatrix} \tag{9}$$

is obtained as the solution to

$$\begin{aligned} \nabla^{\mathrm{T}}(\rho\nabla\phi_j) &= -\frac{1}{R_{jj}}(h_j - \bar{h}_j)\rho, \\ \int \phi_j(\boldsymbol{x}, \lambda)\rho(\boldsymbol{x}, \lambda)\,\mathrm{d}\boldsymbol{x} &= 0, \end{aligned} \tag{10}$$

for $j = 1, \dots, m$, where R_{jj} is the variance of the jth element in $\boldsymbol{y}_k$, h_j is the jth element of $\boldsymbol{h}$, and similarly for $\bar{h}_j$. Note that a diagonal covariance for the measurement noise is assumed. The following consistency result holds for the FPF.

Theorem 1: Suppose that $\boldsymbol{K}$ is obtained according to (10) and that the particle flow is updated as (4), initiated as $\{\boldsymbol{S}_k^i(0)\}_{i=1}^N = \{x_{k^-}^i\}_{i=1}^N$, $\rho(\boldsymbol{x}, 0) = \hat{p}(\boldsymbol{x}_{k^-}|\mathcal{Y}_{k-1})$. If the estimated and true posterior are equal at time t_k^-, $\hat{p}(\boldsymbol{x}_{k^-}, \mathcal{Y}_{k-1}) = p(\boldsymbol{x}_{k^-}, \mathcal{Y}_{k-1})$, at $\lambda = 1$, it holds that $\rho(\boldsymbol{x}, 1) = p(\boldsymbol{x}_k|\mathcal{Y}_k)$.

Proof 1: See [13]. ■

From Theorem 1, it follows that the measurement update is exact and that the FPF provides the true posterior for an infinite number of particles and for a consistent initialization of the filter. In particular, for linear and Gaussian systems, the gain function in the FPF becomes the Kalman gain and the Wong-Zakai term vanished [13]. The two-step process of the FPF is illustrated in Fig. 3.

The main difficulty in the FPF is to find $\boldsymbol{K}$, and only in limited cases can an exact solution be computed. In the remainder of this section, we will discuss a Galerkin-type method based on the weak formulation of (10) [31] that has been developed in [14].

B. Galerkin Approximation of Gain Function

The consistency result in Theorem 1 only holds for an exact expression of the feedback gain. In fact, the main difficulty in the implementation of the FPF is to find solutions to (10). This equation can only be solved exactly for restricted types of systems, such as when (1) is linear and Gaussian. In other cases, numerical techniques are required.

Approximations of varying complexity can be computed based on the weak formulation of (10) [31], leading to a Galerkin-based approximation. A function $\nabla\phi_j$ is said to be a weak solution to (10) if

$$\mathbb{E}((\nabla\phi_j)^{\mathrm{T}}\nabla\psi) = \mathbb{E}\left(\frac{1}{R_{jj}}(h_j - \bar{h}_j)\psi\right) \tag{11}$$

for all *test functions* ψ belonging to $H^1(\mathbb{R}^n, p)$ [14]. By restricting ψ to belong to the subspace of $H^1(\mathbb{R}^n, p)$ spanned by $\{\psi_l\}_{l=1}^L$, ϕ_j is approximated as

$$\phi_j = \sum_{l=1}^{L} \kappa_{j,l}\psi_l, \tag{12}$$

that is, (12) is a weighted finite sum of L basis functions $\{\psi_l\}_{l=1}^L$, where $\{\kappa_{j,l}\}_{l=1}^L$ are constants for a fixed t_k. This implies that the gain function for each column becomes

$$\boldsymbol{k}_j = \sum_{l=1}^{L} \kappa_{j,l}\nabla\psi_l. \tag{13}$$

Eq. (13) leads to a finite-dimensional approximation of (11):

$$\sum_{l=1}^{L} \kappa_{j,l}\mathbb{E}\left((\nabla\psi_l)^{\mathrm{T}}\nabla\psi\right) = \mathbb{E}\bigg(\frac{1}{R_{jj}}(h_j - \bar{h})\psi\bigg). \tag{14}$$

In practical implementations, by substituting ψ with each ψ_l and approximating the expectation using the particle distribution, (14) becomes a linear matrix equation for each $j = 1, \dots, m$,

$$\boldsymbol{A}\boldsymbol{\kappa}_j = \boldsymbol{b}_j, \tag{15}$$

where

$$\boldsymbol{\kappa}_j = \begin{bmatrix}\kappa_{j,1} & \cdots & \kappa_{j,L}\end{bmatrix}^{\mathrm{T}}. \tag{16}$$

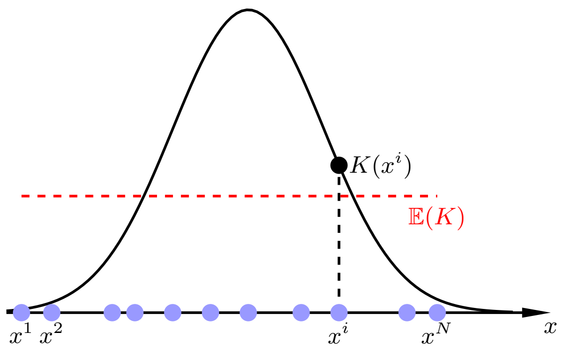


Fig. 4. An illustration of the constant-gain approximation in the one-dimensional case, corresponding to (18). The nonlinear gain function is approximated by a weighted expected value over the particles.

Note that the equation system is the same for all particles. In (15), element sl of $\boldsymbol{A}$, A_{sl}, and element s of $\boldsymbol{b}_j$, $b_{j,s}$, are

$$A_{sl} = \frac{1}{N}\sum_{i=1}^{N}(\nabla\psi_l^i)^{\mathrm{T}}\nabla\psi_s^i,$$

$$b_{j,s} = \frac{1}{R_{jj}N}\sum_{i=1}^{N}(h_j^i - \bar{h}_j)\psi_s^i.$$

1) Constant-Gain Approximation: A computationally simple approximation of the gain function is found by choosing the coordinates in the Galerkin approach as basis functions, that is, $\{\psi_l\}_{l=1}^{L} = \{x_l\}_{l=1}^{n}$. Fig. 4 gives an illustration in the scalar case; if the states are chosen as test functions, we have

$$\nabla\psi_l = \begin{bmatrix}\mathbf{0}_{1\times l-1} & 1 & \mathbf{0}_{1\times L-l+1}\end{bmatrix}^{\mathrm{T}}. \tag{17}$$

Hence, $\boldsymbol{A}$ in (15) becomes the identity matrix, and we end up with an approximation that is the same for all particles, the constant-gain approximation:

$$\begin{aligned}\boldsymbol{K} &\approx \begin{bmatrix}\boldsymbol{c}_1 & \cdots & \boldsymbol{c}_m\end{bmatrix}\boldsymbol{R}^{-1},\\ \boldsymbol{c}_j &:= \frac{1}{N}\sum_{i=1}^{N}\left(h_j^i - \bar{h}_j\right)\boldsymbol{S}_k^i(\lambda).\end{aligned} \tag{18}$$

Using (18) for gain approximation is so far the most common way to find an expression of the gain function. The resulting FPF is hereafter denoted by FPF. The constant-gain approximation is the best constant approximation of $\boldsymbol{K}$ in the mean-square sense, but it is not individualized for each particle.

The FPF using the constant-gain approximation has complexity $\mathcal{O}(N)$ and is summarized in Algorithm 1 for a discretization of the pseudo-time with step size $\Delta\lambda$. Note that the term $\boldsymbol{\Omega}$ as defined by (6) is zero in Algorithm 1.

IV. DATA-DRIVEN GAIN COMPUTATION

In this section, we outline our proposed data-driven approach for gain-function computation. We start from the Galerkin-based approximation outlined in Sec. III-B. Hence, we can formulate the problem we seek to solve as follows.

Problem 1: Find L basis functions $\nabla\psi_l$ that for each column $\boldsymbol{k}_j$ in $\boldsymbol{K}$ approximates the gain function according to (13).

How to choose the basis functions and thereby solve Problem 1 is nontrivial. We propose an approach for choosing the basis functions from observed data. Specifically, our method relies on the observation that the time evolution of the particle cloud and previous corrections due to the measurements describes the global system behavior. As a consequence, the particle cloud contains information about how to locally adjust the particles. We adapt POD [32] to find basis functions for the weak formulation in Sec. III-B. POD is widely used in computational fluid dynamics and structural vibrations, to mention two applications. In image processing it is known as principal component analysis, and is extensively used as a data-extraction method.

Algorithm 1 FPF with constaint-gain approximation

```
    Initialize: Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$
 1: for $k \leftarrow 1$ to $T$ do
 2:     Set $t = t_{k-1}$
 3:     while $t < t_k$ do
 4:         Simulate $\mathrm{d}\boldsymbol{x}^i = \boldsymbol{f}^i\mathrm{d}t + \mathrm{d}\boldsymbol{\beta}$, for $i \in \{1,\dots,N\}$
 5:         Set $t = t + \Delta t$
 6:     end while
 7:     Set $\{\boldsymbol{S}_k^i\}_{i=1}^N = \{\boldsymbol{x}_k^i\}_{i=1}^N$ and $\lambda = 0$.
 8:     while $\lambda \leq 1$ do
 9:         Compute $\boldsymbol{K}$ from (18).
10:         Compute $\boldsymbol{I}_k^i$ from (7), (8), for $i \in \{1,\dots,N\}$.
11:         Compute $\boldsymbol{S}_k^i$ using (4), for $i \in \{1,\dots,N\}$.
12:         Set $\lambda = \lambda + \Delta\lambda$.
13:     end while
14:     Set $\{\boldsymbol{x}_k^i\}_{i=1}^N = \{\boldsymbol{S}_k^i\}_{i=1}^N$
15: end for
```

A. Proper Orthogonal Decomposition

In this section we describe the background on POD necessary to understand our approach and the intuition behind it. The objective in POD is to obtain compact representations of high-dimensional data, such as in large-scale dynamical systems. Suppose the goal is to approximate a vector field $\boldsymbol{\theta}(\boldsymbol{x}, t)$. The field is decomposed as

$$\boldsymbol{\theta}(\boldsymbol{x}, t) = \bar{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\theta}'(\boldsymbol{x}, t),$$

where $\bar{\boldsymbol{\theta}}$ is a steady-state flow and $\boldsymbol{\theta}'$ is the time-varying part. The goal is to represent $\boldsymbol{\theta}'$ as a sum of orthonormal basis functions, that is,

$$\boldsymbol{\theta}' = \sum_{j=1}^{\infty} a_j(t)\boldsymbol{\varphi}_j(\boldsymbol{x}),$$

where a_j are time-dependent coefficients and $\{\boldsymbol{\varphi}_j\}_{j=1}^{\infty} \in X$ is the basis. The coefficients are uncorrelated and computed as $a_j = \langle\boldsymbol{\theta}', \boldsymbol{\varphi}_j\rangle$. In POD, we seek an optimal basis in the sense that if $\boldsymbol{\theta}'$ is projected onto $\{\boldsymbol{\varphi}_j\}_{j=1}^{L}$, the average energy content retained is greater than if projected onto any other set of L basis functions. This can be formulated as

$$\underset{\boldsymbol{\varphi}\in X}{\text{maximize}} \quad \frac{\overline{|\langle\boldsymbol{\theta}', \boldsymbol{\varphi}\rangle|^2}}{\|\boldsymbol{\varphi}\|^2}. \tag{19}$$

Using a first-order variation of the cost function, it can be shown that solving (19) amounts to solving the integral eigenvalue problem

$$\int \overline{R}(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{\varphi}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' = \alpha \boldsymbol{\varphi}(\boldsymbol{x}), \tag{20}$$

where R is the auto-correlation function and α is the eigenvalue. Typically, discretization is performed both in space and time. The discretized version of $\bar{R}$ in (20) is the covariance matrix $\boldsymbol{\Sigma}$, and (20) amounts to solve a matrix eigenvalue problem. For sufficiently many discretization points, the sample covariance matrix is a reliable approximation of $\boldsymbol{\Sigma}$. Assuming a subtracted mean, the sample covariance matrix is given by

$$\boldsymbol{\Sigma} = \frac{1}{M-1}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}, \tag{21}$$

where $\boldsymbol{X}$ is the matrix containing the data and M is the number of time-discretization points. For further details, see [32] and references therein.

An interpretation of POD is that the proper orthogonal modes (POMs, the eigenvectors) define the direction of optimal distribution of energy or power, and the corresponding proper orthogonal values (POVs, the eigenvalues) represent the power associated with each of the POMs [33]. The mass distribution of the particle cloud is corrected whenever a measurement arrives. Consequently, the intuition behind our approach is that if the basis functions are chosen based on the POMs of the time evolution of the particle cloud, the gain function will move the particles according to the directions of most mass concentration of the posterior distribution.

The remainder of this section explains how to incorporate POD for the gain computation.

B. POD for Finding Dominant Modes of the Particles

As mentioned, our approach relies on the observation that the set of particles, when simulated forward in time, gives information about the time evolution of the system. Hence, it is intuitive to suggest that particles that do not follow the behavior of the particle cloud are less likely to be a significant contributor to the posterior estimate and should be corrected accordingly to be consistent with the mass distribution of the particle cloud. To this end, we want to determine how each particle relates to the POMs of the particle cloud.

Assume that we predict N particles $\{\boldsymbol{x}^i\}_{i=1}^N$ using the dynamical system (1a) in open loop up to time t_k^-, when a measurement arrives. At each simulation step (i.e., time-discretization point), the particles are stacked in a column matrix as

$$\boldsymbol{x}' := \begin{bmatrix}(\boldsymbol{x}^1)^{\mathrm{T}} & \cdots & (\boldsymbol{x}^N)^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{nN}. \tag{22}$$

Eq. (22) is a snapshot of the state space using the particle cloud from the FPF. We store the M latest snapshots of the particle cloud, that is, M snapshots in the form of (22). In accordance with the POD approach for generating the data $\boldsymbol{X}$ in (21), we subtract the average from (22) *for each snapshot.* This ensures that the point cloud is centered at its origin. Then we stack the resulting data column wise, leading to

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1^1 & \cdots & \boldsymbol{x}_M^1 \\ \vdots & & \vdots \\ \boldsymbol{x}_1^N & \cdots & \boldsymbol{x}_M^N\end{bmatrix} \in \mathbb{R}^{nN \times M}. \tag{23}$$

To find the principal directions of the set of particle clouds in (23), we employ singular value decomposition (SVD) [34]. Hence, $\boldsymbol{X}$ is decomposed as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}, \tag{24}$$

where $\boldsymbol{U} \in \mathbb{R}^{nN \times nN}$ is an orthonormal matrix containing the left singular vectors of $\boldsymbol{X}$, $\boldsymbol{S} \in \mathbb{R}^{nN \times M}$ consists of $\min(nN, M)$ nonnegative singular values σ_j in decreasing order on the diagonal, and $\boldsymbol{V} \in \mathbb{R}^{M \times M}$ is orthonormal and contains the right singular vectors. Only the POMs corresponding to the most significant singular values are used. Thus, we extract the first $r \leq \min(nN, M)$ columns from $\boldsymbol{U}$ to form $\hat{\boldsymbol{U}}$ and decompose it as

$$\hat{\boldsymbol{U}} = \begin{bmatrix}\boldsymbol{u}_1^1 & \cdots & \boldsymbol{u}_r^1 \\ \vdots & & \vdots \\ \boldsymbol{u}_1^N & \cdots & \boldsymbol{u}_r^N\end{bmatrix} \in \mathbb{R}^{nN \times r}, \tag{25}$$

where the matrix $\boldsymbol{S}$ containing the singular values is truncated similarly. The decomposition (25) gives r orthonormal eigenvectors of the data. Multiplying $\boldsymbol{Q} = \hat{\boldsymbol{U}}\hat{\boldsymbol{S}}$ results in

$$\boldsymbol{Q} = \begin{bmatrix}\sigma_1\boldsymbol{u}_1^1 & \cdots & \sigma_r\boldsymbol{u}_r^1 \\ \vdots & & \vdots \\ \sigma_1\boldsymbol{u}_1^N & \cdots & \sigma_r\boldsymbol{u}_r^N\end{bmatrix} = \begin{bmatrix}\boldsymbol{q}_1^1 & \cdots & \boldsymbol{q}_r^1 \\ \vdots & & \vdots \\ \boldsymbol{q}_1^N & \cdots & \boldsymbol{q}_r^N\end{bmatrix}. \tag{26}$$

The interpretation of $\boldsymbol{V}$ in POD is that column m, $\boldsymbol{v}_m$, determines the time modulation of eigenvector m; that is, element j in $\boldsymbol{v}_m$ is the time modulation of $\boldsymbol{u}_m^i$ at time index $k - M + j$, and the last (Mth) element, v_{mM}, of $\boldsymbol{v}_m$ gives the time modulation at time step k, that is, the current time. Hence, the product $\boldsymbol{q}_m^i v_{mM}$ indicates how much each POM affects the direction of motion of the ith particle. We choose the dominant mode to represent the direction of motion, that is,

$$\bar{\boldsymbol{q}}^i = \boldsymbol{q}_1^i v_{1M} \tag{27}$$

denotes the direction of the dominant POM for the ith particle. Note that in general the direction of motion could also be chosen as an average of the POMs. However, in our results we have not seen any major performance differences. The directional vector (27) is in the next section used to choose the basis functions for approximating the gain function.

C. Gain Computation with POD in Feedback Particle Filter

In the constant-gain approximation, the test functions are chosen as the n state coordinates. This implies through (17) that the lth basis function is a unit step along the lth coordinate axis. On the other hand, the vector (27) obtained from POD represents how the particles are moving in the particle cloud. Hence, to adjust in what direction the measurements should move the particles, we can use (27). Motivated by this, we add $\bar{\boldsymbol{q}}^i$ to the unit step for each particle. In this way, each particle is adjusted locally based on global information from the ensemble of particles. Thus, for particle i, the lth basis function equals

$$\nabla \psi_l^i = \begin{bmatrix} \mathbf{0}_{1\times l-1} & 1 & \mathbf{0}_{1\times L-l+1} \end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i, \tag{28}$$

where the first term on the right-hand side corresponds to the constant-gain approximation (17). The test function ψ_l^i corresponds to the integration of (28) and equals

$$\psi_l^i = x_l^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}} \boldsymbol{x}^i, \tag{29}$$

where x_l^i is the lth element of $\boldsymbol{x}^i$. Note that because the test function (29) is expressed per particle, the test function is in general a nonlinear function of the state. The coefficients $\boldsymbol{\kappa}_j$ in (13) are found by inserting (28) and (29) into (15), which for each measurement y_j, $j = 1, \dots, m$, in $\boldsymbol{y}_k$ results in

$$\begin{aligned} A_{sl} &= \frac{1}{N} \sum_{i=1}^{N} \left(\|\bar{\boldsymbol{q}}^i\|_2^2 + \bar{q}_s^i + \bar{q}_l^i + \delta(s-l) \right), \\ b_s &= \frac{1}{R_{jj} N} \sum_{i=1}^{N} (h_j^i - \bar{h}_j) \left(x_s^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}} \boldsymbol{x}^i \right), \end{aligned} \tag{30}$$

where A_{sl} is the element of $\boldsymbol{A}$ on row s, column l. and where b_s is element s of $\boldsymbol{b}_j$. The resulting gain function becomes

$$\boldsymbol{K}_k^i = \begin{bmatrix} \boldsymbol{k}_1^i & \cdots & \boldsymbol{k}_m^i \end{bmatrix}, \tag{31}$$

where $\boldsymbol{k}_j^i$ is computed using (28) as

$$\boldsymbol{k}_j^i = \sum_{l=1}^{n} \kappa_{j,l} \left(\begin{bmatrix} \mathbf{0}_{1\times l-1} & 1 & \mathbf{0}_{1\times n-l+1} \end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i \right). \tag{32}$$

From (4) and (32), it follows that the correction for particle $\boldsymbol{x}^i$ consists of a feedforward term and a feedback term that is nonlinear in particle $\boldsymbol{x}^i$ and where the gain function $\boldsymbol{K}_k^i$ is adjusted individually for each of the particles through (28). However, the adjustment is also based on global information, both through the term (8) in the innovation error (7) and through the POD-based gain function.

The rationale for why choosing POD for computing the basis functions in the Galerkin approach is that POD acts directly on the system response to extract basis functions, often for subsequent use in Galerkin projections [35]. The goal of the feedback gain $\boldsymbol{K}$ is to drive the particles towards the response of the system given by the measurements. Thus, when using a Galerkin approach for approximating the gain function, there is a close connection to the interpretation of POD. Fig. 5 provides a geometric interpretation of our approach. The filter formulation is summarized in Algorithm 2. Similar to the constant-gain approximation, in this paper for simplicity we ignore the term $\boldsymbol{\Omega}$ defined by (6), essentially assuming that it is negligible compared to the feedback correction term. This assumption is expected to be reasonable when the system dynamics and measurements behave sufficiently smooth. However, analysis is required for determining whether this assumption is valid for a given problem.

Remark 1: The left singular vectors in POD are optimal in the sense that they capture more energy in the L^2 sense for the data along a given direction than any other fixed number of vectors [32]. In other words, the first r columns of $\boldsymbol{U}$ (i.e., $\hat{\boldsymbol{U}}$ in (25)) give an optimal orthonormal basis for approximating the data contained in $\boldsymbol{X}$.

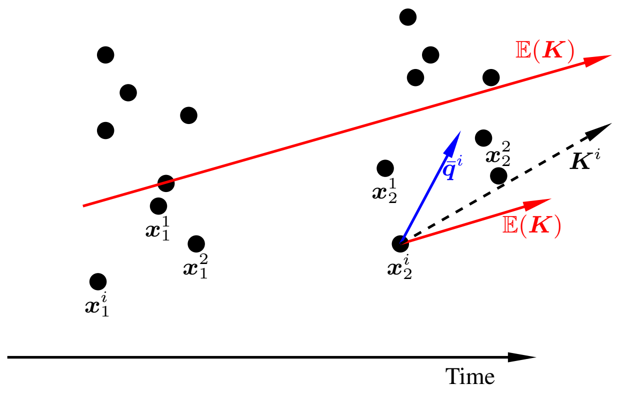


Fig. 5. Schematic of the POD-based gain-function approximation for illustration purposes when $M = 2$. The POD can be interpreted as adding a correction term to the constant-gain approximation. The constant-gain approximation gives the expected value of $\boldsymbol{K}$. Then, the POD adjusts each particle by adding a correction vector $\bar{\boldsymbol{q}}^i$ to each particle, resulting in a corrected $\boldsymbol{K}^i$ for each particle.

D. Computational Complexity

The main computational burden of Algorithm 2 lies in the measurement update, corresponding to lines 7–17. When computing $\bar{\boldsymbol{q}}^i$ for $i \in \{1, \dots, N\}$ using (24)–(26) on Line 8, we need to compute an SVD, which has complexity $\mathcal{O}((nN)^2 + M^3) \approx \mathcal{O}((nN)^2)$ since typically $N \gg M$. The gain computation (32) results in the same number of test functions as the dimension of the state vector. Hence, $\boldsymbol{A}$ in (15) has dimension $n \times n$ and finding the coefficient vector $\boldsymbol{\kappa}$ is independent on N so the SVD on Line 8 is for a reasonably small n where most time is spent. The basis functions should ideally be updated in the measurement update as the pseudo-time increases. However, this would imply an SVD for each time step in the measurement update, which would be computationally prohibitive.

The quadratic complexity mostly renders the approach suitable for a relatively small number of particles. However, when compared to the constant-gain approximation, which is $\mathcal{O}(N)$, the number of particles can often be drastically decreased while still achieving better performance. In a practical implementation, the relative importance between performance and computational resources will decide which algorithm to use.

V. Numerical Study

We evaluate the proposed method for gain computation in the FPF on three different examples and compare it against several different nonlinear filters. All filters are implemented in MATLAB and we have performed Monte-Carlo simulations for all examples. We will highlight different aspects in the different examples. We will sometimes make use of the (time-averaged) root-mean-square error (RMSE) to compare performance. The different methods used throughout this section are:

FPF: FPF with the constant-gain approximation, see Sec. III-B1 and Algorithm 1.

Algorithm 2 FPF with POD-Based Gain Computation

```
    Initialize: Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$
 1: for $k \leftarrow 1$ to $T$ do
 2:     Set $t = t_{k-1}$.
 3:     while $t < t_k$ do
 4:         Simulate $\mathrm{d}\boldsymbol{x}^i = \boldsymbol{f}^i \mathrm{d}t + \mathrm{d}\boldsymbol{\beta}$, for $i \in \{1, \dots, N\}$.
 5:         Set $t = t + \Delta t$.
 6:     end while
 7:     Construct $\boldsymbol{X}$ according to (22), (23).
 8:     Compute $\bar{\boldsymbol{q}}^i$ for $i \in \{1, \dots, N\}$ using (24)–(27).
 9:     Set $\{\boldsymbol{S}_k^i\}_{i=1}^N = \{\boldsymbol{x}_k^i\}_{i=1}^N$ and $\lambda = 0$.
10:     while $\lambda \leq 1$ do
11:         Compute ${}_{l=1}^{n}\{\nabla\psi_l^i, \psi_l^i\}_{i=1}^N$ using (28) and (29).
12:         Compute $\boldsymbol{A}, \boldsymbol{b}_j$ using (30), for $j \in \{1, \dots, m\}$.
13:         Compute $\boldsymbol{\kappa}_j$ using (15), for $j \in \{1, \dots, m\}$.
14:         Compute $\boldsymbol{K}_k^i$ using (31)–(32) for $i \in \{1, \dots, N\}$.
15:         Simulate $\boldsymbol{S}_k^i$ using (4) for $i \in \{1, \dots, N\}$.
16:         Set $\lambda = \lambda + \Delta\lambda$.
17:     end while
18:     Set $\{\boldsymbol{x}_k^i\}_{i=1}^N = \{\boldsymbol{S}_k^i\}_{i=1}^N$.
19: end for
```

ALG2: The proposed FPF in Algorithm 2.

RBPF: A Rao-Blackwellized particle filter (RBPF) based on importance sampling [36].

PF: A particle filter based on importance sampling with optimal proposal [37].

UKF: The continuous-discrete time UKF in [38].

A. Linear Scalar Example

This example has previously been used in [13]. Consider the system

$$\begin{aligned} \mathrm{d}x(t) &= ax(t)\mathrm{d}t + \sigma_\beta \mathrm{d}\beta(t), \\ y_k &= hx_t + \sigma_e e_k, \end{aligned} \tag{33}$$

where $a = -0.5$, $h = 3$, $\sigma_\beta = 1$, $\sigma_e = 2$. The measurements arrive at time instants $t_k = 0.5, 1.0, 1.5, \dots, 10$. In [13], it was shown that the solution to the boundary value problem (10), which for linear systems with Gaussian noise can be expressed in closed form, equals the Kalman gain. We denote this filter with FPFKF. In this example we set the discretization of the particle flow to $\Delta\lambda = 0.05$, which according to the results in [13] offers a good compromise between performance and computational complexity. Furthermore, the dynamics (33) is discretized with a sampling time $\Delta t = 0.005$.

Fig. 6 shows the true mean μ_t and the conditional means obtained using the proposed POD-based FPF (ALG2) and the FPF with exact gain computation (FPFKF) for $N = 50$ particles. Our proposed FPF performs well compared to the FPF with exact gain computation, implying that our approach is close to optimal for linear systems.

Fig. 7 illustrates the gain computation as a function of the particles for a snapshot taken at $t = 15$ s at the last step of the homotopy when using $N = 50, 500, 5000$ particles, respectively. It is seen that our approach computes gains that are very similar to the Kalman gain and the exact formulation.

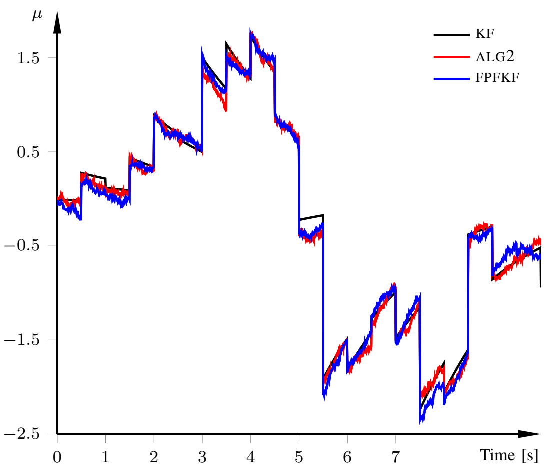


Fig. 6. Comparison of the Kalman filter (KF), the FPF with exact gain computation (FPFKF), and our proposed POD-based gain function approximation (ALG2) for the linear system (33).

This is further illustrated by Fig. 8, which shows the time-averaged RMSE as function of particles taken over 500 Monte-Carlo simulations.

B. Coordinated Turn Problem

This example has been used previously in different contexts [39]–[42]. A target moves in a plan according to a clockwise coordinated turn [41] of radius 500 m with constant velocity 200 km/h. The initial position is $\boldsymbol{p}_0 = [-500\ 500]^{\mathrm{T}}$, starting in the y-direction. The geometric path forms a circle of radius 500 m. The target motion is modeled by a five-state coordinated turn model with unknown constant turn rate and velocity. The continuous-time model of the coordinated turn is

$$\begin{bmatrix} \dot{p}^X \\ \dot{p}^Y \\ \dot{v}^X \\ \dot{v}^Y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p^X \\ p^Y \\ v^X \\ v^Y \\ \omega \end{bmatrix} + \boldsymbol{w}, \tag{34}$$

where p, v, ω denote the position, velocity, and turn rate, respectively. By introducing

$$\boldsymbol{x}_k = \begin{bmatrix} p_k^X & p_k^Y & v_k^X & v_k^Y & \dot{\omega}_k \end{bmatrix}^{\mathrm{T}},$$

the corresponding discrete-time model [42] can be written as

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & \frac{\sin(\dot{\omega}_k \Delta t)}{\dot{\omega}_k} & -\frac{1-\cos(\dot{\omega}_k \Delta t)}{\dot{\omega}_k} & 0 \\ 0 & 1 & \frac{1-\cos(\dot{\omega}_k \Delta t)}{\dot{\omega}_k} & \frac{\sin(\dot{\omega}_k \Delta t)}{\dot{\omega}_k} & 0 \\ 0 & 0 & \cos(\dot{\omega}_k \Delta t) & -\sin(\dot{\omega}_k \Delta t) & 0 \\ 0 & 0 & \sin(\dot{\omega}_k \Delta t) & \cos(\dot{\omega}_k \Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \Delta t \boldsymbol{w}_k.$$

where $\Delta t = 0.01$ is the sampling time. The FPFs FPF and ALG2 use $\Delta\lambda = 0.01$ in the measurement updates and ALG2 uses the last 40 data points for basis computation (see Algorithm 1 and Algorithm 2). These choices are set somewhat arbitrarily, and especially the discretization length heavily influences the computational demands [13]. The process noise $\boldsymbol{w}$ is zero mean Gaussian with covariance $\boldsymbol{Q} = \mathrm{diag}([30^2, 30^2, 0.1^2, 10^2, 10^2])$. We set the initial

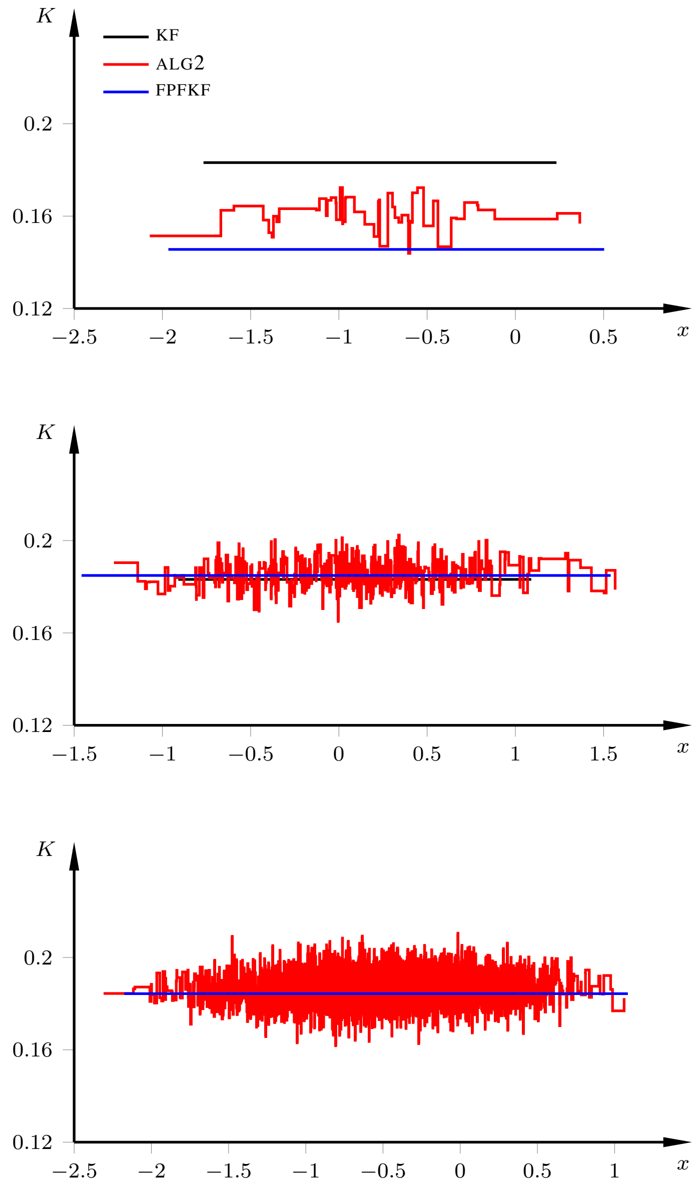


Fig. 7. Comparison of the computed gains from the Kalman filter (KF), the FPF with exact gain computation (FPFKF), and our proposed POD-based gain function approximation (ALG2) for the linear system (33). The number of particles in the respective plot from top to bottom are $N = 50, 500, 5000$.

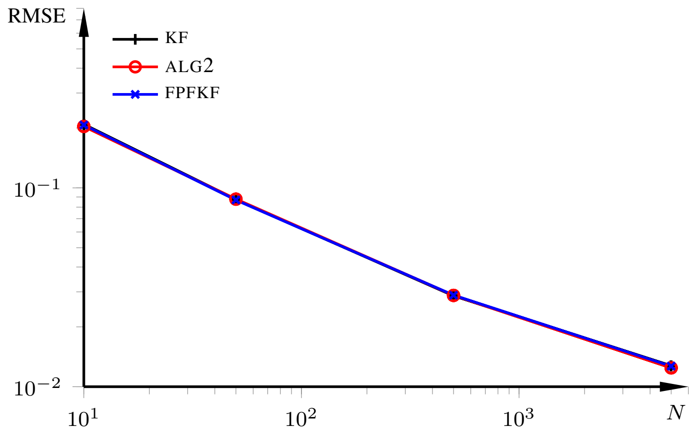


Fig. 8. Time-averaged RMSE for the Kalman filter (KF), the FPF with exact gain computation (FPFKF), and our proposed POD-based gain function approximation (ALG2) for the linear system (33).

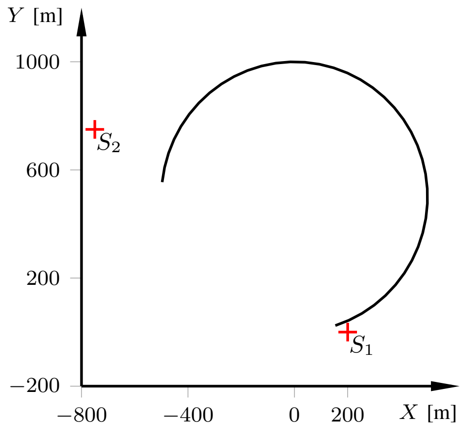


Fig. 9. The geometric path and the sensor locations (red +) used in the simulation example. The path is taken from a 40 s simulation.

estimate for all filters to $\boldsymbol{x}_0 = \mathbf{0}$, with initial covariance $\boldsymbol{P}_0 = \text{diag}([250^2, 250^2, 0.1^2, 30^2, 30^2])$, that is, we know very little about the initial state of the target. Two sensors measure the bearing of the target with sampling time $T_s = 1$ s. The sensors are located at $S_1 = (200, 0)$ and $S_2 = (-750, 750)$. The geometric path for a simulation and the sensor locations are shown in Fig. 9. The measurement model is

$$h_{k,j} = \arctan\left(\frac{p_k^Y - S_j^Y}{p_k^X - S_j^X}\right), \quad j = 1, 2.$$

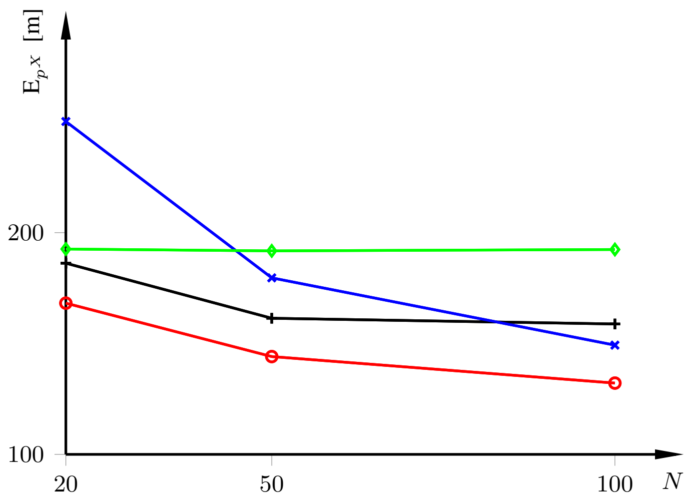
The measurement noise for each sensor is Gaussian zero mean with standard deviation $\sigma_j = 0.1$.

1) Results: We use the RMSE as performance measure, and the results are for 5000 Monte-Carlo simulations. Figs. 10 and 11 show the time-averaged RMSE of the position and velocity, respectively, as the number of particles varies. ALG2 again clearly outperforms the other particle filters. The unscented Kalman filter is competitive for small N, but as the number of particles increase, the proposed FPF performs much better.

In Fig. 12, we display the mean execution time (per update step) for each filter when varying the number of particles. The $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ lines are included for comparison. We see that ALG2 has a larger computational cost than the others, but that the difference is small for small N, which is also where our approach has the largest error improvements. In a practical implementation, the error improvements obviously have to be related to the available computational power.

C. Two-Body Problem

Here we assess the performance of Algorithm 2 using a planar two-body problem, which involves estimating the motion of a satellite that orbits around earth, and compare against a Rao-Blackwellized particle filter (RBPF) [36]. A more detailed comparison of baseline FPF against several PFs and the UKF is found in [16]. Simplified two-dimensional



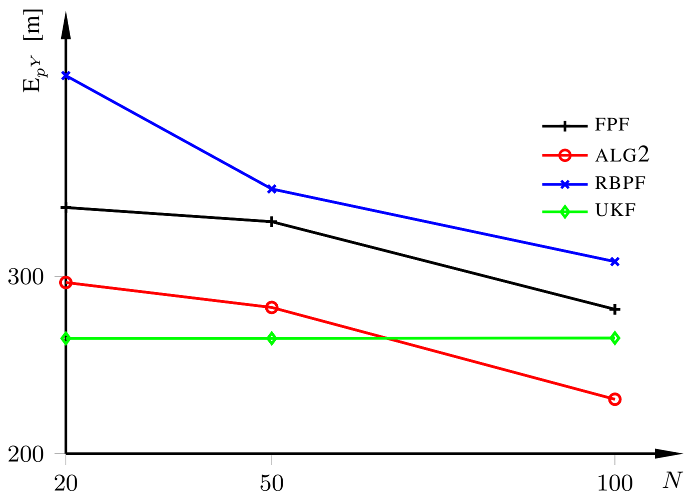
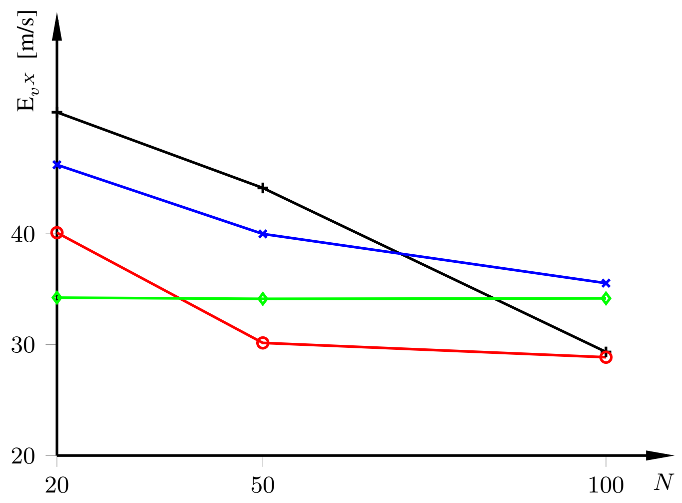


Fig. 10. Time-averaged RMSEs for the position for varying number of particles for the coordinated turn problem. The RMSE values are computed for $N = 20, 50, 100$.



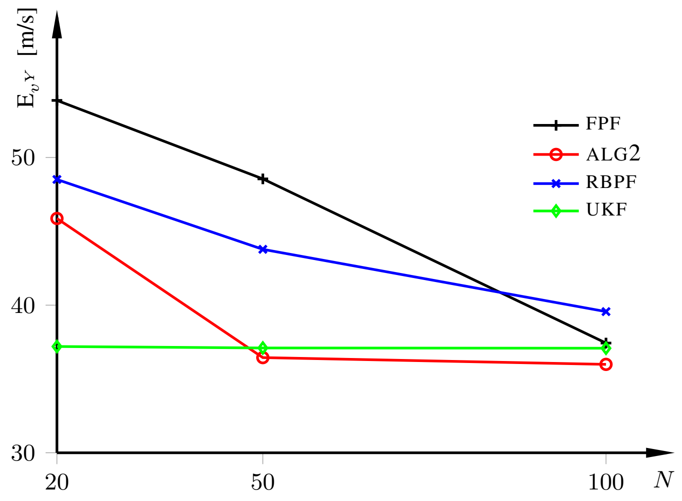


Fig. 11. Time-averaged RMSEs for the velocity for varying number of particles for the coordinated turn problem. The RMSE values are computed for $N = 20, 50, 100$.

equations of motion relative to the earth-fixed, earth-centered, inertial frame are given by

$$\begin{aligned}\dot{p}_X &= v_X, \\ \dot{p}_Y &= v_Y, \\ \dot{v}_X &= -\mu \frac{p_X}{r^3} + \frac{1}{m} F_X + w_3, \\ \dot{v}_Y &= -\mu \frac{p_Y}{r^3} + \frac{1}{m} F_Y + w_4,\end{aligned} \tag{35}$$

where p_X, p_Y are the longitudinal and lateral positions in the earth-fixed frame, respectively, and v_X, v_Y are the corresponding velocities. F_X and F_Y are the external forces applied to the satellite to correct for the perturbation accelerations w_3 and w_4, $r = \sqrt{p_X^2 + p_Y^2}$, $\mu = 398601.2$ is the earth's gravitational constant, and m is the satellite mass. For simplicity, $F_X = F_Y = 0$ in what follows. The perturbations w_3 and w_4 are both Gaussian distributed with zero mean and standard deviation 0.1 m/s^2. The initial conditions are $\boldsymbol{x}_0 = [7000 \;\; 0 \;\; 0 \;\; -7.54]^{\mathrm{T}}$, in km and km/s, respectively, corresponding to a low-earth orbit with period time around

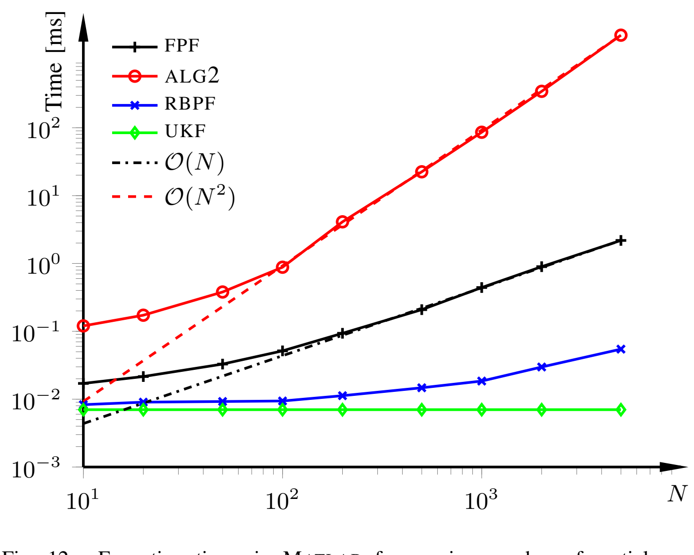


Fig. 12. Execution times in MATLAB for varying number of particles. The time is for one measurement update and the predictions between two measurements and averaged over 50 Monte-Carlo simulations.

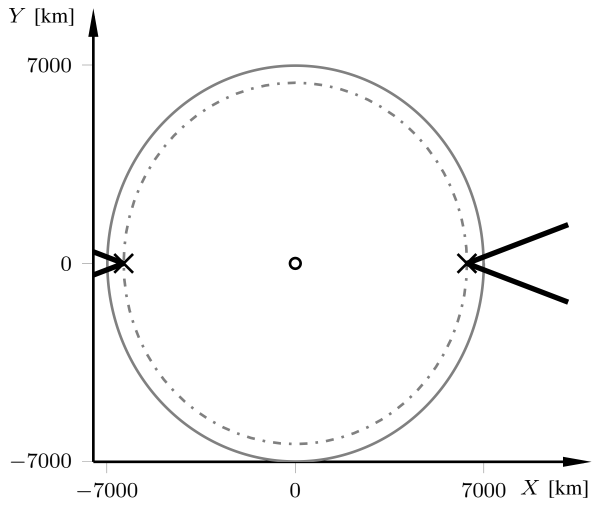


Fig. 13. The two-body problem with two bearing sensors (crosses) that measure the respective angle to the satellite from the earth-fixed X-axis. The earth surface is indicated with the dash-dotted circle and the small solid circle indicates the earth center. The true satellite path for one orbit realization is the gray circle.

97 min. The initial orbit is assumed uncertain for all filters, with covariance matrix $\boldsymbol{P}_0 = \operatorname{diag}([4, 4, 0.04, 0.04])$, where $\operatorname{diag}(\cdot)$ is the diagonal matrix.

Two bearing sensors measure the angle of the satellite relative to the earth-fixed inertial frame. The sensors are located at $S_1 = (r_0, 0)$, $S_2 = (-r_0, 0)$, where $r_0 = 6374$ km. The measurement model is

$$\boldsymbol{y}_k = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \arctan\left(\dfrac{p_Y}{p_X - r_0}\right) \\ \arctan\left(\dfrac{p_Y}{p_X + r_0}\right) \end{bmatrix} + \boldsymbol{e}_k,$$

and both sensors have Gaussian distributed, independent noise, with zero mean and standard deviation 1 deg. Each sensor is only able to track objects that reside in a cone with 40 deg opening angle. When the satellite is within the respective X-axis aligned cone, the sensor provides measurements at 0.1 Hz. Fig. 13 shows a schematic of the setup. Note that the sensors never provide measurements simultaneously. Furthermore, because the measurements are infrequent, there can be a severe mismatch between actual measurement and predicted measurement.

The simulated data is generated by propagating (35) using the Euler-Maruyama scheme with step size $\Delta t = 0.01$ s. The filters are discretized with step size $\Delta t = 0.1$ s (Line 5 in Algorithm 2), and each simulation lasts for 500 min, corresponding to approximately 5.5 orbits. In the FPFs, $\Delta\lambda = 0.001$ (Line 16 in Algorithm 2).

1) Results: The mean-square error is often a useful measure, but does not necessarily describe how well the posterior is estimated. In this problem, the dynamics is governed by an approximately circular orbit; hence, combined with the measurements, we conclude that the posterior should be approximately directed along the orbit. Fig. 14 displays particle clouds for FPF and ALG2 at two time instants. The first snapshot is during prediction phase (when no measurements are available), after roughly 450 min (left part of the figure). The second snapshot is after five orbits, when the satellite is within the visibility cone of the first sensor (right part of the figure). The actual position and estimated mean, respectively, are also shown. We use $N = 100$ in this simulation. The mean estimate of the POD-based FPF is close to the actual position, and the particle cloud aligns along the true orbit during both time instants. The constant-gain FPF predicts a skewed particle cloud during prediction phase, and it is also more scattered. When measurements are available, FPF accurately predicts the posterior to be located along the orbit. However, the particle cloud covers almost a quarter of the orbit, whereas the estimated posterior for ALG2 is more concentrated around the true mean.

To validate against ground truth, Fig. 15 compares the particle clouds after five orbits for ALG2 ($N = 100$) with a Rao-Blackwellized particle filter (RBPF) using $N = 1000$ particles. The posteriors are similar in size and shape. In this particular realization, the mean is slightly more accurate with the RBPF. This can partly be explained by the resulting coarse discretization when only using 100 particles in ALG2. Note that in [16], we showed that the RBPF with $N = 100$ was severely biased, but for 1000 particles it performed well.

VI. Conclusion

We proposed a data-driven approach based on POD for choosing basis functions that approximate the gain function present in the FPF, which is the main difficulty when implementing the FPF. The key idea is that the evolution of the particle cloud gives information about how to locally adjust the particles. Because the method is data driven, it is applicable to a range of estimation problems. To verify this, we applied the proposed method to a linear scalar example and three different benchmark problems, and the proposed method compared favorably in the examples when comparing against other filters from the literature.

The results also indicate that the FPF can be used as an off-the-shelf algorithm for performing parameter estimation. Particle filters based on importance sampling often resort to either roughening of the process noise or rely on marginalization to being able to perform successful parameter estimation. To the contrary, our findings indicate that the FPF, similar to the Kalman filter, can estimate parameters by adding them to the state vector. It is future work to further explore this finding.

References

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, 1993.

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.

[3] A. Doucet and A. M. Johansen, “A Tutorial on Particle Filtering and Smoothing: Fifteen years Later,” in *Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds. Oxford University Press, 2009.

[4] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao-Blackwellized particle filters,” *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, 2007.

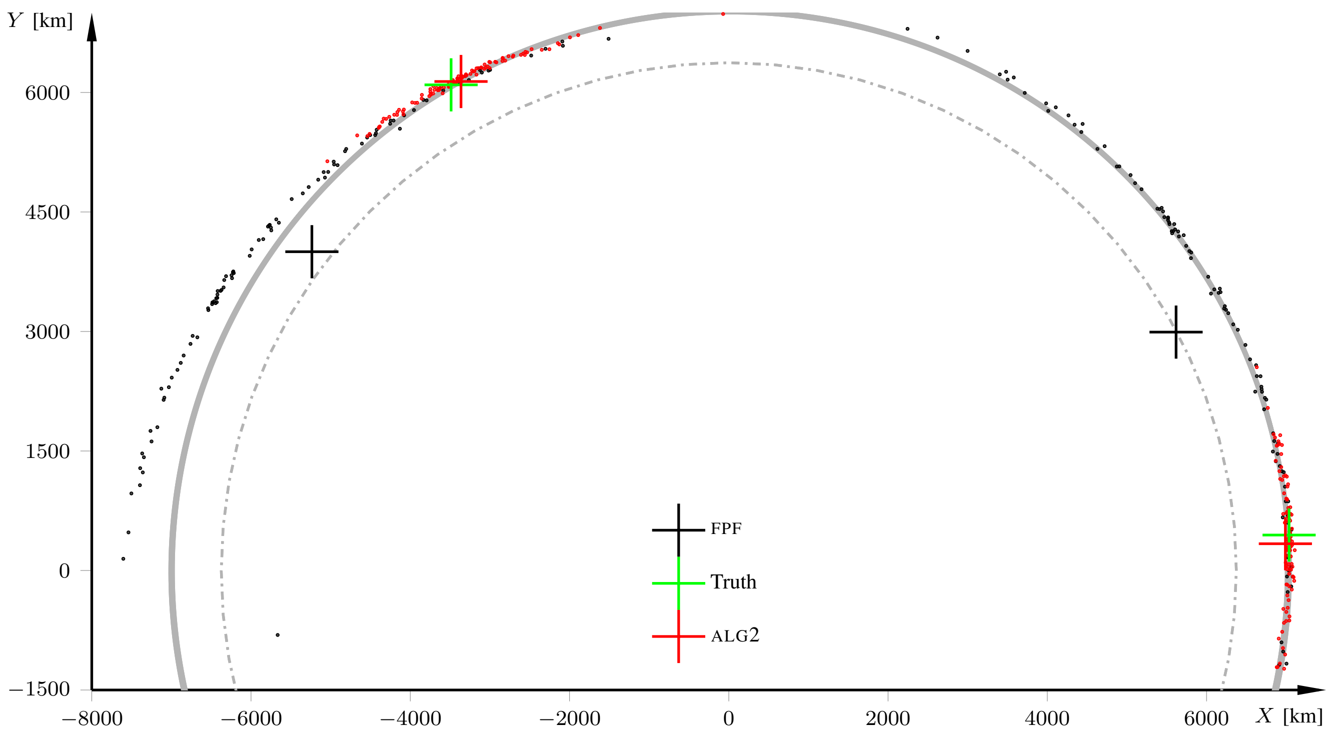


Fig. 14. Particle clouds ($N = 100$) and mean values (+) after roughly four and a half orbits (left) and five orbits (right), respectively. The true path is in solid gray and the earth surface is in dash-dotted gray. The proposed FPF in ALG2 predicts posteriors that are aligned with the orbit, concentrated around the true mean. FPF predicts posteriors misaligned with the orbit and overestimates the uncertainty.

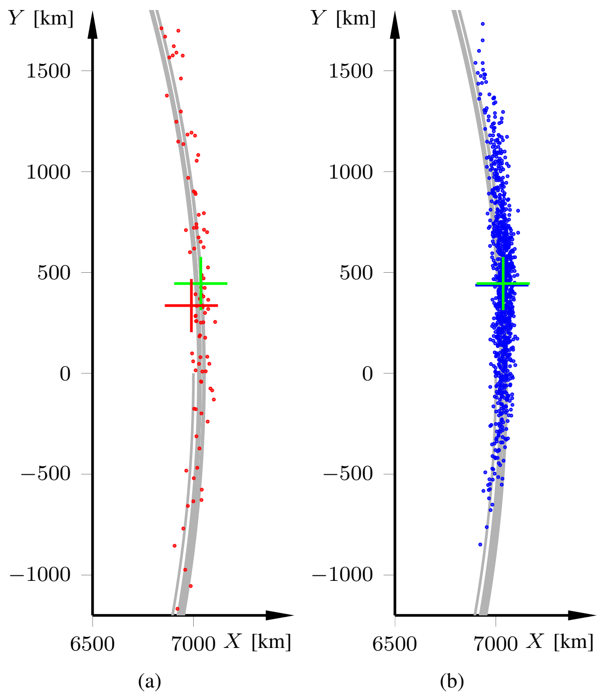


Fig. 15. POD-based FPF (left) for $N = 100$ compared with RBPF (right) using $N = 1000$ after five orbits. Same notation as in Fig. 14. The estimated posteriors are similar in shape and size. Since the RBPF uses ten times more particles, the resulting particle cloud is denser.

[5] F. Gustafsson, “Particle filter theory and practice with positioning applications,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 53–82, 2010.

[6] K. Berntorp, “Particle filter for combined wheel-slip and vehicle-motion estimation,” in *Am. Control Conf.*, Chicago, IL, Jul. 2015.

[7] ——, “Joint wheel-slip and vehicle-motion estimation based on inertial, GPS, and wheel-speed sensors,” *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 1020–1027, 2016.

[8] K. Berntorp and S. Di Cairano, “Process-noise adaptive particle filtering with dependent process and measurement noise,” in *Int. Conf. Decision and Control*, Las Vegas, NV, Dec. 2016.

[9] F. Lindsten, M. I. Jordan, and T. B. Schön, “Particle Gibbs with ancestor sampling.” *J. Machine Learning Res.*, vol. 15, no. 1, pp. 2145–2184, 2014.

[10] T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson, and L. Dai, “Sequential Monte Carlo methods for system identification,” in *IFAC Symp. System Identification*, Bejing, China, Oct. 2015.

[11] A. Doucet, S. Godsill, and C. Andrieu, “On sequential Monte Carlo sampling methods for Bayesian filtering,” *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[12] T. Yang, P. G. Mehta, and S. P. Meyn, “A mean-field control-oriented approach to particle filtering,” in *Am. Control Conf.*, San Francisco, CA, Jun. 2011.

[13] T. Yang, H. Blom, and P. Mehta, “The continuous-discrete time feedback particle filter,” in *Amer. Control conf.*, Portland, OR, Jun. 2014.

[14] T. Yang, R. S. Laugesen, P. G. Mehta, and S. P. Meyn, “Multivariable feedback particle filter,” *Automatica*, vol. 71, no. Supplement C, pp. 10 – 23, 2016.

[15] T. Yang, P. Mehta, and S. Meyn, “Feedback particle filter,” *IEEE Trans. Autom. Control*, vol. 58, no. 10, pp. 2465–2480, 2013.

[16] K. Berntorp, “Feedback particle filter: Application and evaluation,” in *Int. Conf. Information Fusion*, Washington, DC, Jul. 2015.

[17] A. K. Tilton, S. Ghiotto, and P. G. Mehta, “A comparative study of nonlinear filtering techniques,” in *Int. Conf. Information Fusion*, Istanbul, Turkey, Jul. 2013.

[18] R. S. Laugesen, P. G. Mehta, S. P. Meyn, and M. Raginsky, “Poisson’s equation in nonlinear filtering,” *SIAM J. Control and Optimization*, vol. 53, no. 1, pp. 501–525, 2015.

[19] U. D. Hanebeck, K. Briechle, and A. Rauh, “Progressive Bayes: a new framework for nonlinear state estimation,” in *Proc. SPIE*, vol. 5099, 2003.

[20] M. F. Huber, F. Beutler, and U. D. Hanebeck, “Semi-analytic Gaussian assumed density filter,” in *Amer. Control Conf.* San Francisco, CA, Jun. 2011.

[21] P. Bunch and S. Godsill, “Particle filtering with progressive Gaussian approximations to the optimal importance density,” in *Int. Workshop Computational Adv. Multi-Sensor Adaptive Process.*, Saint Martin, Dec. 2013.

[22] F. Daum and J. Huang, “Nonlinear filters with particle flow,” in *Proc. SPIE*, vol. 7445, 2009.

[23] ——, “Particle flow with non-zero diffusion for nonlinear filters,” in *Proc. SPIE*, vol. 8745, 2013.

[24] T. Ding and M. J. Coates, “Implementation of the Daum-Huang exact-flow particle filter,” in *IEEE Statistical Signal Process. Workshop*, Ann Arbor, MI, Aug. 2012.

[25] Y. Li, L. Zhao, and M. Coates, “Particle flow auxiliary particle filter,” in *Int. Workshop Computational Advances in Multi-Sensor Adaptive Processing*, Dec. 2015, pp. 157–160.

[26] P. Bunch and S. Godsill, “Approximations of the optimal importance density using Gaussian particle flow importance sampling,” *J. American Statistical Association*, vol. 111, no. 514, pp. 748–762, 2016.

[27] A. Taghvaei and P. G. Mehta, “Gain function approximation in the feedback particle filter,” in *Int. Conf. Decision and Control*, Las Vegas, NV, 2016.

[28] ——, “An optimal transport formulation of the linear feedback particle filter,” in *Amer. Control Conf.*, Boston, MA, Jul. 2016.

[29] K. Berntorp and P. Grover, “Data-driven gain computation in the feedback particle filter,” in *Amer. Control Conf.*, Boston, MA, Jul. 2016.

[30] G. Tessitore and J. Zabczyk, “Wong-Zakai approximations of stochastic evolution equations,” *J. Evolution Eqs.*, vol. 6, no. 4, pp. 621–655, 2006.

[31] A. Ern, *Theory and practice of finite elements*. Springer, 2004.

[32] G. Kerschen, J.-C. Golinval, A. F. Vakakis, and L. A. Bergman, “The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview,” *Nonlinear Dynamics*, vol. 41, no. 1-3, pp. 147–169, 2005.

[33] B. Feeny and R. Kappagantu, “On the physical interpretation of proper orthogonal modes in vibrations,” *J. Sound and Vibration*, vol. 211, no. 4, pp. 607–616, 1998.

[34] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, Maryland: The Johns Hopkins University Press, 1996.

[35] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Current science*, vol. 78, no. 7, pp. 808–817, 2000.

[36] T. B. Schön, F. Gustafsson, and P.-J. Nordlund, “Marginalized particle filters for mixed linear nonlinear state-space models,” *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2279–2289, 2005.

[37] O. Cappé, S. J. Godsill, and E. Moulines, “An overview of existing methods and recent advances in sequential Monte Carlo,” *Proc. IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[38] S. Särkkä, “On unscented Kalman filtering for state estimation of continuous-time nonlinear systems,” *IEEE Trans. Autom. Control*, vol. 52, no. 9, pp. 1631–1641, 2007.

[39] U. Orguner and F. Gustafsson, “Storage efficient particle filters for the out of sequence measurement problem,” in *Int. Conf. Information Fusion*, Cologne, Germany, Jun. 2008.

[40] K. Berntorp, K.-E. Årzén, and A. Robertsson, “Storage efficient particle filters with multiple out-of-sequence measurements,” in *Int. Conf. Information Fusion*, Singapore, Jul. 2012.

[41] X. Rong Li and V. P. Jilkov, “Survey of maneuvering target tracking . part I: dynamic models,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1333–1364, 2003.

[42] F. Gustafsson, *Statistical Sensor Fusion*. Lund, Sweden: Utbildningshuset/Studentlitteratur, 2010.