MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Data-Driven Gain Computation in the Feedback Particle Filter

Berntorp, K.; Grover, P.

TR2016-050 July 2016

Abstract

The recently introduced feedback particle filter (FPF) is a control-oriented particle filter (PF) aimed at estimation of nonlinear/non-Gaussian systems. The FPF controls each particle using feedback from the measurements and is resampling free, which is in contrast to conventional PFs based on importance sampling. The control gains are computed by solving boundary value problems. In general, numerical approximations are required and it is an open question how to properly compute the approximate solution. This paper outlines a novel method inspired by high-dimensional data-analysis techniques. Based on the time evolution of the particle cloud, we compute values of the gain function for each particle. We exemplify applicability and highlight the benefit of the approach on a planar two-body problem.

2016 American Control Conference (ACC)

Data-Driven Gain Computation in the Feedback Particle Filter

Karl Berntorp[1] and Piyush Grover[1]

***Abstract*— The recently introduced feedback particle filter (FPF) is a control-oriented particle filter (PF) aimed at estimation of nonlinear/non-Gaussian systems. The FPF controls each particle using feedback from the measurements and is resampling free, which is in contrast to conventional PFs based on importance sampling. The control gains are computed by solving boundary value problems. In general, numerical approximations are required and it is an open question how to properly compute the approximate solution. This paper outlines a novel method inspired by high-dimensional data-analysis techniques. Based on the time evolution of the particle cloud, we compute values of the gain function for each particle. We exemplify applicability and highlight the benefit of the approach on a planar two-body problem.**

I. INTRODUCTION

This paper is concerned with systems of the form

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

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

where $\boldsymbol{x} := \boldsymbol{x}(t) \in \mathbb{R}^n$ is the state; $\boldsymbol{y}_k := \boldsymbol{y}(t_k) \in \mathbb{R}^m$ is the discrete-time measurement at time t_k; $\boldsymbol{f}$ and $\boldsymbol{h}$ are the drift and measurement function, respectively; and $\boldsymbol{\beta}$ and $\boldsymbol{e}$ are process and measurement noise, respectively. The aim in continuous-discrete time Bayesian filtering is to estimate the posterior filtering density $p(\boldsymbol{x}|\mathcal{Y}_k)$, or at least the relevant moments, at each time $t \in \mathbb{R}$. Here, $\mathcal{Y}_k := \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_k\}$ denotes the set of measurements, obtained at discrete time steps. Sometimes a discretized counterpart to (1a) is used, resulting in (possibly with different $\boldsymbol{f}$)

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, k) + \boldsymbol{w}_k, \tag{2a}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{e}_k. \tag{2b}$$

Particle filters (PFs) [1], [2] are popular for estimation of nonlinear systems. PFs generate random state trajectories and assign a weight to them according to how well they predict the observations. PFs have been successful in numerous applications, see [3]–[5] for some examples. One problem with PFs is the inevitable particle degeneracy [6] (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].

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

The feedback particle filter (FPF) has been introduced in a series of papers as a control-oriented, resampling-free, variant of the PF [7]–[10]. The FPF applies a feedback structure to each particle. 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 [11], [12] have demonstratred 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 boundary value problem. Usually, approximate solutions are necessary, because closed-form expressions can only be computed in certain special cases.

This paper outlines a data-driven approach for computation of the gain function in the FPF, which should be 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 the time evolution of the particle cloud.

There are few papers addressing the control-gain approximation in the FPF. In [11], we demonstrated how a sensible approximation of the gain function can increase performance compared with baseline FPF for a specific system. The approach in this paper is more general, as we show in Sec. IV, and still compares favorably. Gain computation for an artificial, scalar example was considered in [9].

A. Notation

With $\delta(\boldsymbol{x} - \boldsymbol{y})$ we mean the Dirac delta function, which is one when $\boldsymbol{x} = \boldsymbol{y}$ and zero otherwise. The conditional probability density function of column matrix $\boldsymbol{x}$ given $\boldsymbol{y}$ is denoted by $p(\boldsymbol{x}|\boldsymbol{y})$. For notational brevity and to be consistent with [9], we will sometimes write p instead. Define the expectation as $\mathbb{E}(\boldsymbol{x}) := \int \boldsymbol{x} p \,\mathrm{d}\boldsymbol{x}$ and denote the average of $\boldsymbol{x}$ as $\bar{\boldsymbol{x}}$. 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 $\|\boldsymbol{u}\| := \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle}$. Finally, $\mathbf{0}_{n\times 1}$ is the $n \times 1$ matrix with zeros everywhere.

II. FEEDBACK PARTICLE FILTER

This section gives a brief overview of the main steps in the FPF. The key step in Bayesian filtering is Bayes' rule, which states that

$$p(\boldsymbol{x}_k|\mathcal{Y}_k) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\mathcal{Y}_{k-1}). \tag{3}$$

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 particles. Each particle is weighted using the likelihood. The key differences between the FPF and conventional PFs are that in the FPF,

- the ensemble of particles is a controlled system;
- Bayes' rule (3) is implemented using a continuous transformation from prior to posterior.

The FPF approximates the posterior p with N unweighted samples, or particles, $\boldsymbol{x}^i$ as

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

Note the difference to conventional PFs, where weights are used to select the importance of the particles. The FPF treats the measurement update of the ith particle as a controlled system:

$$\mathrm{d}\boldsymbol{x}^i = \boldsymbol{f}^i\mathrm{d}t + \mathrm{d}\boldsymbol{\beta}^i + \boldsymbol{U}_k^i.$$

As seen from Fig. 1, the structure of the FPF is similar to that of the Kalman filter. To incorporate $\boldsymbol{y}_k$, a particle flow $\boldsymbol{S}_k^i := \boldsymbol{S}_k^i(\lambda)$ and a control input $\boldsymbol{U}_k^i := \boldsymbol{U}_k^i(\lambda)$ are introduced:

$$\frac{\mathrm{d}\boldsymbol{S}_k^i}{\mathrm{d}\lambda} = \boldsymbol{U}_k^i, \tag{4}$$

where $\lambda \in [0,1]$ is the *pseudo-time*. $\boldsymbol{S}_k^i$ is initialized ($\lambda = 0$) to equal the ith particle before the measurement update and $\boldsymbol{U}_k^i$ is designed such that the distribution generated by $\{\boldsymbol{S}_k^i\}_{i=1}^N$ approximates the posterior at $\lambda = 1$. This leads to a simulation-based implementation of Bayes' rule, unlike traditional PFs. This approach is made possible by a log-homotopy transformation, which transforms the discrete-time Bayesian measurement update to a continuously evolving process. In (4), $\boldsymbol{U}_k^i$ has the form

$$\boldsymbol{U}_k^i(\lambda) = \boldsymbol{K}_k^i\boldsymbol{I}_k^i + \frac{1}{2}\boldsymbol{\Omega}_k^i,$$

where $\boldsymbol{K}_k^i = \begin{bmatrix}\boldsymbol{k}_1^i & \cdots & \boldsymbol{k}_m^i\end{bmatrix} := \boldsymbol{K}^i(\boldsymbol{S}_k^i,\lambda) \in \mathbb{R}^{n\times m}$ is the feedback gain function, $\boldsymbol{\Omega}_k^i := \boldsymbol{\Omega}(\boldsymbol{S}_k^i,\lambda)$ is the *Wong-Zakai correction term* [13]. In the remainder, $\boldsymbol{\Omega} \approx \boldsymbol{0}$. The innovation error $\boldsymbol{I}^i$ equals

$$\boldsymbol{I}_k^i = \boldsymbol{y}_k - \frac{1}{2}(\boldsymbol{h}^i + \hat{\boldsymbol{h}}), \tag{5}$$

where $\hat{\boldsymbol{h}} = \mathbb{E}(\boldsymbol{h}(\boldsymbol{x}))$ is expressed in terms of particles as

$$\hat{\boldsymbol{h}} \approx \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{h}(\boldsymbol{S}_k^i).$$

The innovation process (5) includes the predicted measurement of particle i and the average of all particles. The control synthesis is done by solving an optimal-control problem at each pseudo-time, with the Kullback-Leibler divergence as the cost [10]. By defining $(\boldsymbol{\phi})^{\mathrm{T}} = \begin{bmatrix}\phi_1 & \cdots & \phi_m\end{bmatrix} := \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x},\lambda)$, $\boldsymbol{k}_j := \nabla\phi_j(\boldsymbol{x},\lambda)$ is a solution to the boundary value problem

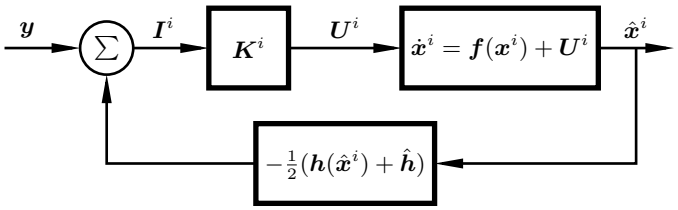

Fig. 1. Simplified block diagram of the FPF. It uses feedback gains $\{\boldsymbol{K}^i\}_{i=1}^N$ to control the particles $\{\boldsymbol{x}^i\}_{i=1}^N$. This is in contrast to the conventional PF, where only the particles' weights are changed in the measurement update.

$$\nabla^{\mathrm{T}}(p\nabla\phi_j) = -\frac{1}{R_{jj}}(h_j - \hat{h}_j)p, \tag{6}$$

for $j = 1,\ldots,m$ and each time t_k [9], where R_{jj} is the variance of the jth element in $\boldsymbol{y}_k$. In analogy with the PF, the FPF is consistent (i.e., $\hat{p}(\boldsymbol{x}|\mathcal{Y}_t) = p(\boldsymbol{x}|\mathcal{Y}_t)$ for all t given correct initial distribution) when $N \to \infty$. A key feature with the simulation-based update in the FPF is that it removes the need for resampling, which is present in conventional PFs.

The main difficulty in the FPF is to find $\boldsymbol{K}$. In the following, we will discuss a Galerkin-based method based on the weak formulation of (6) [14].

A. Galerkin Approximation of Gain Function

The consistency result for the FPF 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 (6). 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 (6) [14], leading to a Galerkin-based approximation. A function $\nabla\phi_j$ is said to be a weak solution to (6) if

$$\mathbb{E}((\nabla\phi_j)^{\mathrm{T}}\nabla\psi) = \mathbb{E}\left(\frac{1}{R_{jj}}(h_j - \hat{h}_j)\psi\right) \tag{7}$$

for all *test functions* ψ belonging to $H^1(\mathbb{R}^n,p)$ [9]. 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

$$\phi_j = \sum_{l=1}^{L}\kappa_j^l\psi_l, \tag{8}$$

that is, (8) 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

$$\boldsymbol{k}_j = \sum_{l=1}^{L}\kappa_j^l\nabla\psi_l. \tag{9}$$

Eq. (9) leads to a finite-dimensional approximation of (7):

$$\sum_{l=1}^{L} \kappa_j^l \mathbb{E}\left((\nabla\psi_l)^{\mathrm{T}}\nabla\psi\right) = \mathbb{E}\left(\frac{1}{R_{jj}}(h_j - \hat{h})\psi\right). \tag{10}$$

In practical implementations, by substituting ψ with each ψ_l and approximating the expectation using the particle distribution, (10) becomes a linear matrix equation

$$\boldsymbol{A}\boldsymbol{\kappa}_j = \boldsymbol{b}_j. \tag{11}$$

Note that the equation system is the same for all particles. Hence, element sl of $\boldsymbol{A}$, A_{sl}, and element l of $\boldsymbol{b}_j$, b_j^l, are found as

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

$$b_j^s = \frac{1}{R_{jj}N}\sum_{i=1}^{N}(h_j^i - \hat{h}_j)\psi_s^i.$$

1) Constant-Gain Approximation: A computationally simple approximation of $\boldsymbol{K}_k^i$ is found by choosing the coordinates as basis functions, that is, $\{\psi_l\}_{l=1}^L = \{x_l\}_{l=1}^n$. If the states are chosen as test functions, we have

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

Hence, $\boldsymbol{A}$ in (11) becomes the identity matrix, and we end up with an approximation that is the same for all particles, the constant-gain approximation:

$$\boldsymbol{K}_k \approx \begin{bmatrix}\boldsymbol{c}_1 & \cdots & \boldsymbol{c}_m\end{bmatrix}\boldsymbol{R}^{-1}, \quad \boldsymbol{c}_j := \frac{1}{N}\sum_{i=1}^{N}\left(h_j^i - \hat{h}_j\right)\boldsymbol{S}_k^i. \tag{12}$$

Using (12) 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.

III. DATA-DRIVEN GAIN COMPUTATION

Our method for gain-function approximation relies on the observation that the time evolution of the particle cloud describes the global system behavior. As a consequence, the particle cloud contains information about how to locally adjust the particles. We adapt POD [16] to find basis functions for the weak formulation in Sec. II-A. 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.

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 $\{\boldsymbol{\varphi}\}_{j=1}^{\infty} \in X$ 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 $\{\boldsymbol{\varphi}\}_{j=1}^{L}$, the average energy content retained is greater than if projected onto any other set of L basis functions. This can be formulated as

$$\underset{\boldsymbol{\varphi}\in X}{\text{maximize}} \quad \frac{\overline{|\langle\boldsymbol{\theta}', \boldsymbol{\varphi}\rangle|^2}}{\|\boldsymbol{\varphi}\|^2}. \tag{13}$$

Using a first-order variation of the cost function, it can be shown that solving (13) amounts to solving an integral eigenvalue problem:

$$\int \overline{R(\boldsymbol{x}, \boldsymbol{x}')}\boldsymbol{\varphi}(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}' = \lambda\boldsymbol{\varphi}(\boldsymbol{x}), \tag{14}$$

where R is the auto-correlation function. Typically, discretization is performed both in space and time. The discretized version of $\bar{R}$ in (14) is the covariance matrix $\boldsymbol{\Sigma}$, and (14) 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

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

where $\boldsymbol{X}$ is the matrix containing the data and M is the number of time-discretization points. For further details, see [16] and references therein. The remainder of this section explains how to incorporate POD for gain computation.

A. Using POD in Feedback Particle Filter

Our goal is to utilize the motion equations to choose a suitable basis for the gain function. To this end, assume that we simulate (1a) in open loop until time index k, when a measurement arrives. At each simulation step (i.e., time-discretization point), $\boldsymbol{x}$ is discretized in N points in $\mathbb{R}^n$, $\{\boldsymbol{x}^i\}_{i=1}^N$. These points are stacked in a column matrix as

$$\boldsymbol{x}' := \begin{bmatrix}(\boldsymbol{x}^1)^{\mathrm{T}} & \cdots & (\boldsymbol{x}^N)^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{nN}. \tag{15}$$

Eq. (15) is a discretization of the state space using the particle cloud from the FPF. We store the M latest snapshots of the particle cloud, normalize, subtract the average, and stack them column wise, leading to

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1^1 & \cdots & \boldsymbol{x}_M^1 \\ \vdots & & \vdots \\ \boldsymbol{x}_1^N & \cdots & \boldsymbol{x}_M^N\end{bmatrix} \in \mathbb{R}^{nN\times M}. \tag{16}$$

We now use singular value decomposition (SVD) of $\boldsymbol{\Sigma}$ [17]. Hence, decompose $\boldsymbol{X}$ as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}, \tag{17}$$

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 numbers 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. Extract the first $r \leq \min(nN, M)$ columns from $\boldsymbol{U}$ to form $\hat{\boldsymbol{U}}$ and decompose it as

$$\hat{\boldsymbol{U}} = \begin{bmatrix} \boldsymbol{u}_1^1 & \cdots & \boldsymbol{u}_r^1 \\ \vdots & & \vdots \\ \boldsymbol{u}_1^N & \cdots & \boldsymbol{u}_r^N \end{bmatrix} \in \mathbb{R}^{nN\times r}. \tag{18}$$

From the decomposition (18), we know have r orthonormal eigenvectors for each of the N particles. The singular values as usual determine the magnitude of each eigenvector. Multiplying $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{S}$ results in

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_1\boldsymbol{u}_1^1 & \cdots & \sigma_L\boldsymbol{u}_r^1 \\ \vdots & & \vdots \\ \sigma_1\boldsymbol{u}_1^N & \cdots & \sigma_L\boldsymbol{u}_r^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1^1 & \cdots & \boldsymbol{q}_r^1 \\ \vdots & & \vdots \\ \boldsymbol{q}_1^N & \cdots & \boldsymbol{q}_r^N \end{bmatrix}. \tag{19}$$

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 $k - M + j$. Thus, since we store the particle cloud up to time index k, we multiply $\boldsymbol{q}_m^i$ with the last element of $\boldsymbol{v}_m$. The average of the resulting vectors gives the direction of motion. We denote the result with $\bar{\boldsymbol{q}}^i$.

B. 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 that the lth basis function is a unit step along the lth coordinate axis. On the other hand, the eigenvectors obtained from POD represent the direction of motion (i.e., the gradient) for each particle. 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

$$\nabla\psi_l^i = \begin{bmatrix}\boldsymbol{0}_{1\times l-1} & 1 & \boldsymbol{0}_{1\times L-l+1}\end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i \tag{20}$$

and the corresponding test function, which is linear in the particle, becomes

$$\psi_l^i = x_l^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}}\boldsymbol{x}^i. \tag{21}$$

Then, (20) and (21) are used to compute $\boldsymbol{\kappa}_j$ in (11), where

$$\begin{aligned} A_{sl} &= \frac{1}{N}\sum_{i=1}^{N}\left(\|\bar{\boldsymbol{q}}^i\|_2^2 + \bar{q}_s^i + \bar{q}_l^i + \delta(s-l)\right), \\ b_j^s &= \frac{1}{R_{jj}N}\sum_{i=1}^{N}(h_j^i - \hat{h}_j)\left(x_s^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}}\boldsymbol{x}^i\right). \end{aligned} \tag{22}$$

The resulting gain function becomes

$$\boldsymbol{K}_k^i = \begin{bmatrix}\boldsymbol{k}_1^i & \cdots & \boldsymbol{k}_m^i\end{bmatrix}, \tag{23}$$

where $\boldsymbol{k}_j^i$ are computed as

$$\boldsymbol{k}_j^i = \sum_{l=1}^{n}\kappa_j^l\left(\begin{bmatrix}\boldsymbol{0}_{1\times l-1} & 1 & \boldsymbol{0}_{1\times n-l+1}\end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i\right). \tag{24}$$

The filter formulation is summarized in Algorithm 1.

Remark 1: The left singular vectors in POD are optimal in the sense that they capture more energy along a given direction than any other set of basis functions [16]. In other words, the first r columns of $\boldsymbol{U}$ (i.e., $\hat{\boldsymbol{U}}$ in (18)) give an optimal orthonormal basis for approximating the data contained in $\boldsymbol{X}$.

Remark 2: The gain computation (24) results in test functions that are linear in the particles. Hence, $\boldsymbol{A}$ in (11) has dimension $n \times n$. The main computational overhead compared with the constant-gain approximation is instead the computation of the SVD. Note, however, that the SVD only is computed once per measurement update, not at each step in the particle-flow simulation (4). Often it is sufficient to choose the first few eigenvectors to accurately explain the modes of the considered system, implying that $r \ll \min(nN, M)$.

Algorithm 1 FPF with POD-Based Gain Computation

Initialize: Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$
for $k \leftarrow 0$ to T **do**
  Set $t = t_k$
  while $t \leq t_{k+1}$ **do**
    Simulate $\mathrm{d}\boldsymbol{x}^i = \boldsymbol{f}^i\mathrm{d}t + \mathrm{d}\boldsymbol{\beta}$, for $i \in \{1, \ldots, N\}$
  end while
  Construct $\boldsymbol{X}$ according to (16) using M particle clouds
  Compute $\bar{\boldsymbol{q}}^i$ using (17)–(19)
  Set $\{\boldsymbol{S}_k^i\}_{i=1}^N = \{\boldsymbol{x}_k^i\}_{i=1}^N$ and $\lambda = 0$
  while $\lambda \leq 1$ **do**
    Compute ${}_{l=1}^{n}\{\nabla\psi_l^i, \psi_l^i\}_{i=1}^N$ using (20) and (21)
    Compute $\boldsymbol{A}, \boldsymbol{b}_j$ using (22), for $j \in \{1, \ldots, m\}$
    Compute $\boldsymbol{\kappa}_j$ using (11), for $j \in \{1, \ldots, m\}$
    Compute $\boldsymbol{K}_k^i$ using (23)–(24) for $i \in \{1, \ldots, N\}$
    Simulate the particle flow according to (4) using $\boldsymbol{K}_k^i$ and $\boldsymbol{I}_k^i$ computed by (5) for $i \in \{1, \ldots, N\}$
  end while
  Set $\{\boldsymbol{x}_k^i\}_{i=1}^N = \{\boldsymbol{S}_k^i\}_{i=1}^N$
end for

IV. NUMERICAL STUDY

We assess the performance on 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) [18]. A more detailed comparison of FPF against several PFs and the UKF is found in [11]. Simplified two-dimensional equations of motion relative to the earth-fixed, earth-centered, inertial frame are given by

$$\begin{aligned} \dot{p}_X &= v_X, \\ \dot{p}_Y &= v_X, \\ \dot{v}_X &= -\mu\frac{p_X}{r^3} + \frac{1}{m}F_X + w_3, \\ \dot{v}_Y &= -\mu\frac{p_Y}{r^3} + \frac{1}{m}F_Y + w_4, \end{aligned} \tag{25}$$

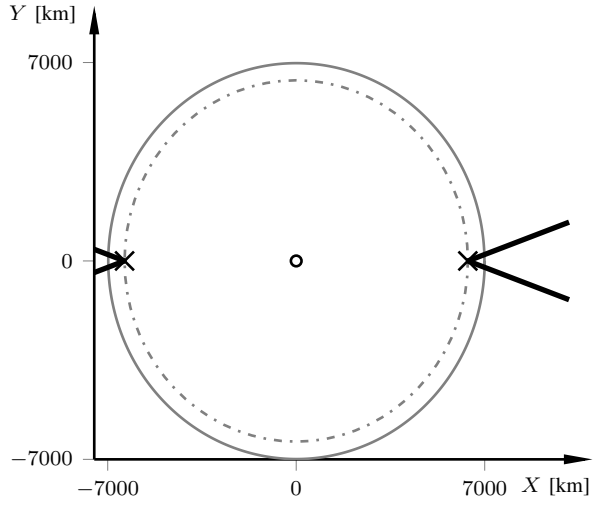

Fig. 2. The two-body problem with two bearing sensors (crosses) that measure the respective angle to the satellite. The earth surface is indicated with the dash-dotted circle and the small solid circle indicates the earth center. The true satellite path for one orbit realization is the gray circle.

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 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.[1] The initial conditions are

$$\boldsymbol{x}_0 = \begin{bmatrix} 7000 & 0 & 0 & -7.54 \end{bmatrix}^{\mathrm{T}}, \tag{26}$$

in km and km/s, respectively, corresponding to a low-earth orbit with period time around 97 min..

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

$$\boldsymbol{y}_k = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \arctan\left(\dfrac{p_Y}{p_X - R_0}\right) \\ \arctan\left(\dfrac{p_Y}{p_X + R_0}\right) \end{bmatrix} + \boldsymbol{e},$$

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. 2 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.

[1]Keplerian orbits do not exist in practice because of perturbation forces. Satellites drift from their assigned orbital positions because of, e.g., solar radiation pressure and atmospheric drag, if not accounted for. These disturbances, i.e., w_3 and w_4, are modeled as Gaussians here.

The initial orbit is assumed uncertain for all filters, with mean (26) and covariance matrix

$$\boldsymbol{P}_0 = \mathrm{diag}([4, 4, 0.04, 0.04]).$$

The simulated data is generated by propagating (25) 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, and each simulation lasts for 500 min, corresponding to approximately 5.5 orbits. In the FPFs, $\Delta\lambda = 0.001$.

1) Results: RMSE 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, it is possible to conclude that the posterior should be approximately directed along the orbit. Fig. 3 displays particle clouds for FPF and ALG1 at two instants in time. The first snapshot is during prediction phase (when no measurements are available), after roughly 450 min (left part of figure). The second snapshot is after five orbits, when the satellite is within the visibility cone of the first sensor (right part of figure). Also shown is the actual position and estimated mean, respectively, at both time instants. 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 mesurements 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 ALG1 is more concentrated around the true mean.

To validate against ground truth, Fig. 4 compares the particle clouds after five orbits for ALG1 ($N = 100$) 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 ALG1. Note that in [11] we showed that the RBPF with $N = 100$ was severily biased, but for 1000 particles it performed well.

V. Conclusion

We proposed a data-driven approach for choosing basis functions that approximate the gain function present in the FPF, which is the main difficulty when implementing the FPF. This is the first paper that proposes a general method for how to choose the basis functions. 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, we believe that it can be applied to a range of estimation problems.

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.

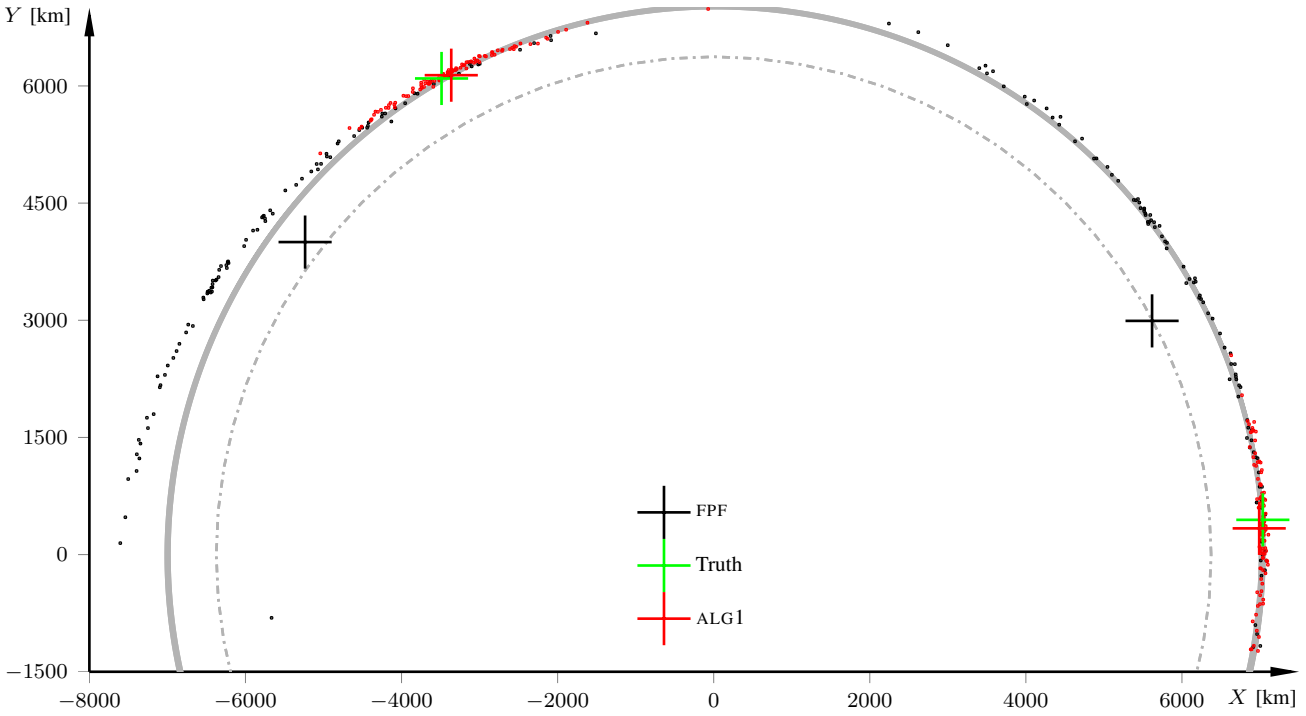

Fig. 3. Particle clouds ($N = 100$) and mean values (+) after roughly four and a half orbits (left) and five orbits (right), respectively. True path is in solid gray and earth surface is in dash-dotted gray. Our proposed FPF (ALG1) predicts posteriors that are aligned with the orbit, concentrated around the true mean. FPF predicts posteriors misaligned with the orbit (left) and overestimates the uncertainty.

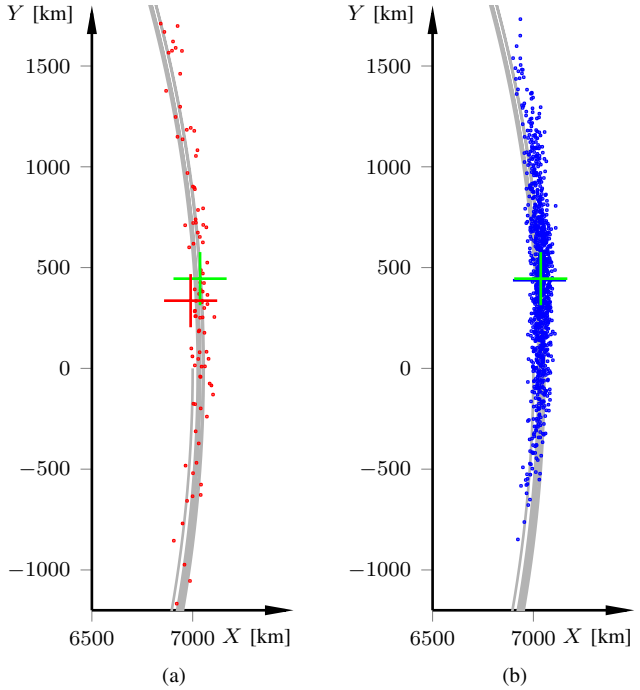

Fig. 4. POD-based FPF (left) for $N = 100$ compared with RBPF using $N = 1000$ after five orbits. Same notation as in Fig. 3. The estimated posteriors are similar in shape and size. Since the RBPF uses ten times more particles, the particle cloud is more dense for the RBPF.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] T. Yang, H. Blom, and P. Mehta, "The continuous-discrete time feedback particle filter," in *Am Control conf.*, Portland, OR, Jun. 2014.

[9] T. Yang, R. Laugesen, P. Mehta, and S. Meyn, "Multivariable feedback particle filter," in *51st Conf. Decision and Control*, Grand Wailea, Maui, Hawaii, Dec. 2012.

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

[11] K. Berntorp, "Feedback particle filter: Application and evaluation," in *18th Int. Conf. Information Fusion*, Washington, DC, Jul. 2015.

[12] A. K. Tilton, S. Ghiotto, and P. G. Mehta, "A comparative study of nonlinear filtering techniques," in *16th Int. Conf. Information Fusion*, Istanbul, Turkey, Jul. 2013.

[13] G. Tessitore and J. Zabczyk, "Wong-Zakai approximations of stochastic evolution equations," *J. evolution eqs.*, vol. 6, no. 4, pp. 621–655, 2006.

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

[15] 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.

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

[17] 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.