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

Comparison of Gain Function Approximation Methods in the Feedback Particle Filter

Berntorp, K.

TR2018-102 July 13, 2018

Abstract

This paper is concerned with a study of the different proposed gain-function approximation methods in the feedback particle filter. The feedback particle filter (FPF) has been introduced in a series of papers as a control-oriented, resampling-free, variant of the particle filter. The FPF applies a feedback gain to control each particle, where the gain function is found as a solution to a boundary value problem. approximate solutions are usually necessary, because closed-form expressions can only be computed in certain special cases. By now there exist a number of different methods to approximate the optimal gain function, but it is unclear which method is preferred over another. This paper provides an analysis of some of the recently proposed gain-approximation methods. We discuss computational and algorithmic complexity, and compare performance using well-known benchmark examples.

International Conference on Information Fusion (FUSION)

Comparison of Gain Function Approximation Methods in the Feedback Particle Filter

Karl Berntorp[1]

***Abstract*—This paper is concerned with a study of the different proposed gain-function approximation methods in the feedback particle filter. The feedback particle filter (FPF) has been introduced in a series of papers as a control-oriented, resampling-free, variant of the particle filter. The FPF applies a feedback gain to control each particle, where the gain function is found as a solution to a boundary value problem. Approximate solutions are usually necessary, because closed-form expressions can only be computed in certain special cases. By now there exist a number of different methods to approximate the optimal gain function, but it is unclear which method is preferred over another. This paper provides an analysis of some of the recently proposed gain-approximation methods. We discuss computational and algorithmic complexity, and compare performance using well-known benchmark examples.**

I. Introduction

The feedback particle filter (FPF) has been introduced in a series of papers as a control-oriented, resampling-free, variant of the particle filter (PF) [1]–[4]. 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 [5]. Numerical studies (e.g., [6], [7]) 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 [8]. Usually, approximate solutions are necessary, because closed-form expressions can only be computed in certain special cases.

The FPF provides a numerical solution to the continuous-time filtering problem

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

$$\mathrm{d}\boldsymbol{y}_t = \boldsymbol{h}(\boldsymbol{x}_t)\mathrm{d}t + \mathrm{d}\boldsymbol{e}_t, \tag{1b}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the state at time t, $\boldsymbol{y}_t \in \mathbb{R}^m$ is the observation, and $\{\boldsymbol{\beta}_t\}$, $\{\boldsymbol{e}_t\}$ are mutually independent Wiener processes with covariances $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. The mappings $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are given C^1 functions.

The FPF estimates the posterior distribution $p(\boldsymbol{x}_t|\mathcal{Y}_t)$, $\mathcal{Y}_t = \{\boldsymbol{y}_s : s \leq t\}$ by N particles $\{\boldsymbol{x}_t^i\}_{i=1}^N$, similar to the PF [9], [10]. For each time t, an approximation to the posterior distribution is given by

$$p(\boldsymbol{x}_t|\mathcal{Y}_t) \approx \hat{p}(\boldsymbol{x}_t|\mathcal{Y}_t) = \frac{1}{N}\sum_{i=1}^N \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^i), \tag{2}$$

where $\delta(\cdot)$ is the Dirac delta function.

Instead of relying on importance sampling, where particles are generated according to their importance weight [10], [11], the FPF is a controlled system consisting of N particles. The ith particle has the gain feedback form

$$\mathrm{d}\boldsymbol{x}_t^i = \boldsymbol{f}(\boldsymbol{x}_t^i)\mathrm{d}t + \mathrm{d}\boldsymbol{\beta}_t^i + \boldsymbol{K}(\boldsymbol{x}_t^i, t) \circ \mathrm{d}\boldsymbol{I}_t^i, \tag{3}$$

where

$$\mathrm{d}\boldsymbol{I}_t^i = \mathrm{d}\boldsymbol{y}_t - \frac{1}{2}(\boldsymbol{h}(\boldsymbol{x}_t^i) + \bar{\boldsymbol{h}})\mathrm{d}t, \tag{4}$$

$$\bar{\boldsymbol{h}} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{h}(\boldsymbol{x}_t^i), \tag{5}$$

and $\circ$ indicates that the differential equation is expressed in its Stratonovich form [4].

The gain function

$$\boldsymbol{K} = \begin{bmatrix}\nabla\phi_1(\boldsymbol{x}_t, t) & \nabla\phi_2(\boldsymbol{x}_t, t) & \dots & \nabla\phi_m(\boldsymbol{x}_t, t)\end{bmatrix} \tag{6}$$

is obtained from a solution to the boundary value problem

$$\begin{aligned}\nabla^{\mathrm{T}}(p(\boldsymbol{x}_t|\mathcal{Y}_t)\nabla\phi_j) &= -\frac{1}{R_{jj}}(h_j - \bar{h}_j)p(\boldsymbol{x}_t|\mathcal{Y}_t),\\ \int \phi_j(\boldsymbol{x}_t, t)p(\boldsymbol{x}_t|\mathcal{Y}_t)\,\mathrm{d}\boldsymbol{x} &= 0,\end{aligned} \tag{7}$$

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 $\bar{h}_j$. Given $\boldsymbol{K}$, the FPF is consistent: If the particles initially are sampled from the correct prior, then as $N \to \infty$, $\hat{p}(\boldsymbol{x}_t|\mathcal{Y}_t) \to p(\boldsymbol{x}_t|\mathcal{Y}_t)$. Algorithm 1 provides the general FPF algorithm, assuming a sampling period Δt.

The challenging part in the FPF is to solve (7). For the linear Gaussian case the solution is given by the Kalman gain. In general, however, approximate solutions are necessary. Various different approaches to compute the approximate solution to (7) have been proposed recently. The perhaps most basic approximation method is to use a constant-gain approximation [2], [6], [12], where the gain function is computed based on the average of the particles. Learning techniques based on dynamic programming are found in [13] and a numerical Galerkin solution is proposed in [3], [4]. A difficulty with the Galerkin approach is how to choose the basis functions

[1]The author is with Mitsubishi Electric Research Laboratories (MERL), 02139 Cambridge, MA, USA. Email: karl.o.berntorp@ieee.org

Algorithm 1 FPF algorithm

Initialize: Set $\{\boldsymbol{x}_0^i\}_{i=1}^N \sim p_0(\boldsymbol{x}_0)$
while $t \leq T_{\text{final}}$ **do**
  Compute $\boldsymbol{K}(\boldsymbol{x}_t^i, t)$ for $i \in \{1, \ldots, N\}$.
  for $i \leftarrow 1$ **to** N **do**
    Compute $\Delta \boldsymbol{I}_t^i$ from (4).
    Set $\bar{\boldsymbol{x}} = \boldsymbol{x}_t^i + \boldsymbol{K}(\boldsymbol{x}_t^i, t)\Delta \boldsymbol{I}_t$.
    Draw $\Delta\beta \sim \mathcal{N}(\boldsymbol{0}, 1)$.
    Set $\boldsymbol{x}_{t+\Delta t}^i = \bar{\boldsymbol{x}} + \boldsymbol{f}(\bar{\boldsymbol{x}})\Delta t + \sqrt{Q\Delta t}\Delta\boldsymbol{\beta}$.
  end for
  Set $t = t + \Delta t$
end while

embedded in the Galerkin solution. In the original paper [4], the states were chosen as the basis functions. Continuation schemes appear in [14]. In recent work, we proposed a method based on proper orthogonal decomposition (POD) for basis functions selection [15], [16]. A recently proposed kernel-based algorithm seems promising, as it avoids the need for basis functions [17], [18]. Yet another approach is based on an optimal transport formulation, which currently applies to the linear FPF [19].

In this paper, we provide an overview, discussion, and comparison of some of the available gain-function approximation methods. In particular, we will focus on the recently proposed constant-gain approximation [3], the POD-based Galerkin solution [15], and the basis-free kernel algorithm [18]. There have been comparison studies involving the FPF before (e.g., [3], [4], [6], [7], [15]). However, these studies have been focused on showing the relative merits between various formulations of the FPF with conventional PFs. Instead, in this paper we provide a simulation study of different FPF formulations. Using three examples of varying complexity, we perform extensive Monte-Carlo simulations and discuss the relative merits of the methods, both in terms of tracking and computational performance. The Galerkin and kernel approaches rely on the same type of assumptions, such as the form of the posterior density and that the measurement function $\boldsymbol{h}$ and its gradient are in L^2 [3], [17]. We will not focus on the mathematical details but rather investigate how the approaches perform in practice.

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.

Outline: The outline of the remainder of the paper is as follows. Sec. II provides a summary of the Galerkin approach to gain-function approximation, whereas the kernel-based approach is treated in Sec. III. Sec. IV contains the evaluation of the different methods, and Sec. V concludes the paper.

II. Galerkin Approximation of Gain Function

In this section we summarize the Galerkin approach to gain-function approximation. For a richer account of the mathematical details and assumptions made, refer to [3]. The basis of the Galerkin method is the weak formulation of (7), from which approximations of varying complexity can be computed. A function $\nabla \phi_j$ is said to be a weak solution to (7) if

$$\mathbb{E}((\nabla \phi_j)^{\mathrm{T}} \nabla \psi) = \mathbb{E}\left(\frac{1}{R_{jj}} (h_j - \bar{h}_j) \psi \right) \tag{8}$$

for all *test functions* ψ belonging to $H^1(\mathbb{R}^n, p)$ [3]. 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

$$\phi_j = \sum_{l=1}^L \kappa_{j,l} \psi_l, \tag{9}$$

that is, (9) 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

$$\boldsymbol{k}_j = \sum_{l=1}^L \kappa_{j,l} \nabla \psi_l. \tag{10}$$

Eq. (10) leads to a finite-dimensional approximation of (8):

$$\sum_{l=1}^L \kappa_{j,l} \mathbb{E}\left((\nabla \psi_l)^{\mathrm{T}} \nabla \psi \right) = \mathbb{E}\left(\frac{1}{R_{jj}} (h_j - \bar{h}) \psi \right). \tag{11}$$

By substituting ψ with each ψ_l and approximating the expectation using the particle distribution, (11) becomes a linear matrix equation for each $j = 1, \ldots, m$,

$$\boldsymbol{A} \boldsymbol{\kappa}_j = \boldsymbol{b}_j, \tag{12}$$

where

$$\boldsymbol{\kappa}_j = \begin{bmatrix} \kappa_{j,1} & \cdots & \kappa_{j,L} \end{bmatrix}^{\mathrm{T}}. \tag{13}$$

Note that the equation system is the same for all particles. In (12), element sl of $\boldsymbol{A}$, A_{sl}, and element s of $\boldsymbol{b}_j$, $b_{j,s}$, are

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

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

A. Constant-Gain Approximation

A computationally cheap way to approximate the gain function is done by choosing the coordinates in the Galerkin approach as basis functions, that is, $\{\psi_l\}_{l=1}^{L} = \{x_l\}_{l=1}^{n}$. If the states are the test functions, we have

$$\nabla\psi_l = \begin{bmatrix}\mathbf{0}_{1\times l-1} & 1 & \mathbf{0}_{1\times L-l+1}\end{bmatrix}^{\mathrm{T}}. \tag{14}$$

Hence, $\boldsymbol{A}$ in (12) becomes the identity matrix, and we end up with an approximation that is the same for all particles, the constant-gain approximation:

$$\begin{aligned}\boldsymbol{K} &\approx \begin{bmatrix}\boldsymbol{c}_1 & \cdots & \boldsymbol{c}_m\end{bmatrix}\boldsymbol{R}^{-1},\\ \boldsymbol{c}_j &:= \frac{1}{N}\sum_{i=1}^{N}\left(h_j^i - \bar{h}_j\right)\boldsymbol{x}_t^i.\end{aligned} \tag{15}$$

The FPF resulting from using (15) 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 $\mathcal{O}(N)$. The gain computation is summarized in Algorithm 2, which should be used on Line 2 in Algorithm 1. Note that for the continuous-discrete time case, the measurement update in Algorithm 1 (i.e., Lines 4–5) should additionally be discretized [2].

Algorithm 2 Constant-gain approximation

Input: $\{\boldsymbol{x}_t^i\}_{i=1}^N$, $\{\boldsymbol{h}(\boldsymbol{x}_t^i)\}_{i=1}^N$
Output: $\boldsymbol{K}$
1: Determine $\bar{\boldsymbol{h}}$ using (5).
2: Compute $\boldsymbol{K}$ from (15).

B. Proper Orthogonal Decomposition for Gain Computation

In this section, we describe the POD-based gain-function approximation method. The objective in POD is to obtain compact representations of high-dimensional data, such as in large-scale dynamical systems [20]. Suppose the goal is to approximate a vector field $\boldsymbol{\theta}(\boldsymbol{x}, t)$. The vector field $\boldsymbol{\theta}$ is represented by a sum of orthonormal basis functions,

$$\boldsymbol{\theta} = \sum_{j=1}^{\infty} a_j(t)\boldsymbol{\varphi}_j(\boldsymbol{x}),$$

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

$$\underset{\boldsymbol{\varphi}\in X}{\text{maximize}} \quad \frac{\overline{|\langle\boldsymbol{\theta}, \boldsymbol{\varphi}\rangle|^2}}{\|\boldsymbol{\varphi}\|^2}. \tag{16}$$

Using a first-order variation of the cost function, it can be shown that solving (16) amounts to solving the integral eigenvalue problem

$$\int \overline{R}(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{\varphi}(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}' = \alpha\boldsymbol{\varphi}(\boldsymbol{x}), \tag{17}$$

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 (17) is the covariance matrix $\boldsymbol{\Sigma}$, and (17) 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

$$\boldsymbol{\Sigma} = \frac{1}{M-1}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}, \tag{18}$$

where $\boldsymbol{X}$ is the matrix containing the data and M is the number of time-discretization points.

The POD approach relies on the observation that the set of particles, when simulated forward in time, gives information about the time evolution of the system. Consequently, at each simulation step (i.e., time-discretization point), the N particles are stacked in a column matrix as

$$\boldsymbol{x}' := \begin{bmatrix}(\boldsymbol{x}^1)^{\mathrm{T}} & \cdots & (\boldsymbol{x}^N)^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{nN}. \tag{19}$$

Eq. (19) is a snapshot of the state space using the particle cloud from the FPF. The POD approach uses the M latest snapshots of the particle cloud. The average for each snapshot is subtracted from (19) and the resulting data is stacked column wise, leading to

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1^1 & \cdots & \boldsymbol{x}_M^1\\ \vdots & & \vdots\\ \boldsymbol{x}_1^N & \cdots & \boldsymbol{x}_M^N\end{bmatrix} \in \mathbb{R}^{nN\times M}. \tag{20}$$

Singular value decomposition (SVD) [21] is used to find the principal directions of the set of particle clouds in (20). $\boldsymbol{X}$ is decomposed as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}, \tag{21}$$

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 eigenvectors corresponding to the most significant singular values are used. Thus, the first $r \leq \min(nN, M)$ columns from $\boldsymbol{U}$ are extracted to form $\hat{\boldsymbol{U}}$, which is decomposed as

$$\hat{\boldsymbol{U}} = \begin{bmatrix}\boldsymbol{u}_1^1 & \cdots & \boldsymbol{u}_r^1\\ \vdots & & \vdots\\ \boldsymbol{u}_1^N & \cdots & \boldsymbol{u}_r^N\end{bmatrix} \in \mathbb{R}^{nN\times r}, \tag{22}$$

where the matrix $\boldsymbol{S}$ containing the singular values is truncated similarly. The decomposition (22) gives r orthonormal eigenvectors of the data. Multiplying $\boldsymbol{Q} = \hat{\boldsymbol{U}}\hat{\boldsymbol{S}}$ results in

$$\boldsymbol{Q} = \begin{bmatrix}\sigma_1\boldsymbol{u}_1^1 & \cdots & \sigma_r\boldsymbol{u}_r^1\\ \vdots & & \vdots\\ \sigma_1\boldsymbol{u}_1^N & \cdots & \sigma_r\boldsymbol{u}_r^N\end{bmatrix} = \begin{bmatrix}\boldsymbol{q}_1^1 & \cdots & \boldsymbol{q}_r^1\\ \vdots & & \vdots\\ \boldsymbol{q}_1^N & \cdots & \boldsymbol{q}_r^N\end{bmatrix}. \tag{23}$$

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. The direction of motion is represented by the dominant mode,

$$\bar{\boldsymbol{q}}^i = \boldsymbol{q}_1^i v_{1M}. \tag{24}$$

1) Gain Computation with POD in FPF: In the constant-gain approximation, the test functions are the n state coordinates. This implies through (14) that the lth basis function is a unit step along the lth coordinate axis. On the other hand, the vector (24) 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 (24). 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

$$\nabla\psi_l^i = \begin{bmatrix}\boldsymbol{0}_{1\times l-1} & 1 & \boldsymbol{0}_{1\times L-l+1}\end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i, \tag{25}$$

where the first term on the right-hand side corresponds to the constant-gain approximation (14). The test function ψ_l^i corresponds to the integration of (25) and equals

$$\psi_l^i = x_l^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}}\boldsymbol{x}^i, \tag{26}$$

where x_l^i is the lth element of $\boldsymbol{x}^i$. Note that because the test function (26) is expressed per particle, the test function is in general a nonlinear function of the state. The coefficients $\boldsymbol{\kappa}_j$ in (10) are found by inserting (25) and (26) into (12), which for each measurement y_j, $j=1,\dots,m$, in $\boldsymbol{y}_k$ results in

$$\begin{aligned} A_{sl} &= \frac{1}{N}\sum_{i=1}^{N}\left(\|\bar{\boldsymbol{q}}^i\|_2^2 + \bar{q}_s^i + \bar{q}_l^i + \delta(s-l)\right), \\ b_s &= \frac{1}{R_{jj}N}\sum_{i=1}^{N}(h_j^i - \bar{h}_j)\left(x_s^i + (\bar{\boldsymbol{q}}^i)^{\mathrm{T}}\boldsymbol{x}^i\right), \end{aligned} \tag{27}$$

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

$$\boldsymbol{K}_k^i = \begin{bmatrix}\boldsymbol{k}_1^i & \cdots & \boldsymbol{k}_m^i\end{bmatrix}, \tag{28}$$

where $\boldsymbol{k}_j^i$ is computed using (25) as

$$\boldsymbol{k}_j^i = \sum_{l=1}^{n}\kappa_{j,l}\left(\begin{bmatrix}\boldsymbol{0}_{1\times l-1} & 1 & \boldsymbol{0}_{1\times n-l+1}\end{bmatrix}^{\mathrm{T}} + \bar{\boldsymbol{q}}^i\right). \tag{29}$$

The correction for particle $\boldsymbol{x}^i$ consists of a 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 (25).

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 [22]. 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. The filter formulation is summarized in Algorithm 3.

C. Computational Complexity

The POD approach with the implementation in Algorithm 3 is $\mathcal{O}(N^2)$. The gain computation (29) results in the same number of test functions as the dimension of the state vector. Hence, $\boldsymbol{A}$ in (12) has dimension $n\times n$ and finding the coefficient vector $\boldsymbol{\kappa}$ is independent on N [16].

Algorithm 3 FPF with POD-Based Gain Computation

Input: $\{\boldsymbol{x}_{t_j}^i\}_{i=1}^N$ for $j\in[k-M+1,k]$, $\{\boldsymbol{h}(\boldsymbol{x}_{t_k}^i)\}_{i=1}^N$
Output: $\{\boldsymbol{K}(\boldsymbol{x}_{t_k}^i)\}_{i=1}^N$

1: Construct $\boldsymbol{X}$ according to (19), (20).
2: Compute $\bar{\boldsymbol{q}}^i$ for $i\in\{1,\dots,N\}$ using (21)–(24).
3: Compute $\overset{n}{_{l=1}}\{\nabla\psi_l^i,\psi_l^i\}_{i=1}^N$ using (25) and (26).
4: Compute $\boldsymbol{A},\boldsymbol{b}_j$ using (27), for $j\in\{1,\dots,m\}$.
5: Compute $\boldsymbol{\kappa}_j$ using (12), for $j\in\{1,\dots,m\}$.
6: Compute $\boldsymbol{K}^i$ using (28)–(29) for $i\in\{1,\dots,N\}$.

III. Kernel-based approach

We now outline the kernel-based gain-function approximation. We include the details necessary for implementing the method, but refer to the original papers [17], [18] for details.

The kernel-based algorithm is based on the fixed-point solution of the equation

$$\phi = e^{t\Delta_\rho}\phi + \int_0^t e^{s\Delta_\rho}(h-\bar{h})\mathrm{d}s, \tag{30}$$

where $\rho(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\mathcal{Y}_t)$ and scalar-valued measurements are assumed. In (30), the *semigroup* $e^{t\Delta_\rho}\phi$ is

$$e^{t\Delta_\rho}\phi = \sum_{m=1}^{\infty} e^{-t\lambda_m}\langle e_m,\phi\rangle e_m, \tag{31}$$

where λ_m is the eigenvalue of the eigenfunction e_m. The set of eigenfunctions forms a complete orthonormal basis on L^2 [17], [23]. It is shown in [17], [18] that the density ρ admits a spectral gap under certain assumptions, from which it can be deduced that there exists a unique solution to (30), and thereby to (7); that is, the solution to the fixed-point equation (30) solves the boundary value problem (7).

A kernel approximation is used to approximate the semigroup by an integral operator T_ϵ for $t=\epsilon$, that is, the fixed-point solution (30) is approximated as

$$\phi_\epsilon = T_\epsilon\phi_\epsilon + \int_0^\epsilon T_s(h-\bar{h})\mathrm{d}s, \tag{32}$$

where

$$\begin{aligned} T_\epsilon\phi_\epsilon &= \int_{\mathbb{R}^d} k_\epsilon(\boldsymbol{x},\boldsymbol{y})\phi_\epsilon(\boldsymbol{y})\rho(\boldsymbol{y}_k)\mathrm{d}\boldsymbol{y} \\ &\approx \frac{1}{N}\sum_{i=1}^{N} k_\epsilon(\boldsymbol{x},\boldsymbol{x}^i)\phi_\epsilon(\boldsymbol{x}^i). \end{aligned} \tag{33}$$

The exact shape of the Gaussian kernel k_ϵ is found in [18]. For our purposes, it suffices to say that it is based on a Gaussian

kernel and that the method of successive approximations is used to solve the discrete counterpart of (32),

$$\boldsymbol{\Phi} = \boldsymbol{T}\boldsymbol{\Phi} + \epsilon(\tilde{\boldsymbol{h}} - \bar{h}), \tag{34}$$

where $\boldsymbol{\Phi} = [\phi_\epsilon(\boldsymbol{x}^1)\ \phi_\epsilon(\boldsymbol{x}^2)\ \dots\ \phi_\epsilon(\boldsymbol{x}^N)]^{\mathrm{T}}$ is the unknown, $\tilde{\boldsymbol{h}} = [h(\boldsymbol{x}^1)\ h(\boldsymbol{x}^2)\ \dots\ h(\boldsymbol{x}^N)]^{\mathrm{T}}$, and $\boldsymbol{T}$ is an $N \times N$ Markov matrix that approximates the operator T_ϵ, with the ijth entry

$$T_{ij} = \frac{1}{N}\sum_{j=1}^{N} k_\epsilon(\boldsymbol{x}^i, \boldsymbol{x}^j). \tag{35}$$

Given $\boldsymbol{\Phi}$ and $\boldsymbol{T}$, the gain function $\boldsymbol{K}$ can be computed.

One possible implementation of the method is summarized in Algorithm 4, see [17], [18] for other possibilities. There are a number of implementation parameters to consider. For instance, the parameter $\epsilon > 0$ can be interpreted as a step length. As $\epsilon \to 0$ the approximation error decreases as $\|\boldsymbol{K}_\epsilon - \boldsymbol{K}\| \leq C\epsilon + \text{h.o.t}$, where the constant C depends on the function h and h.o.t stands for higher-order terms. However, as $\epsilon \to \infty$ one recovers the constant-gain approximation in Sec. II-A, which shows that there are indeed connections between the Galerkin and kernel approach. The iteration limit T_{iter} is another parameter. For the simulation studies in Sec. IV, with the solution from the previous time step as initialization, T_{iter} typically should be in the range $T_{\text{iter}} \in [1, 10]$.

Algorithm 4 Kernel-based gain-function approximation

Input: $\{\boldsymbol{x}_t^i\}_{i=1}^N$, $\tilde{\boldsymbol{h}}$, $\boldsymbol{\Phi}_0$
Output: $\{\boldsymbol{K}(\boldsymbol{x}_t^i)\}_{i=1}^N$
1: Set $g_{ij} = e^{(-\|\boldsymbol{x}_t^i - \boldsymbol{x}_t^j\|/(4\epsilon))}$ for $i, j \in \{1, \dots, N\}$.
2: Set $k_{ij} = \frac{g_{ij}}{\sqrt{\sum_l g_{il} \sum_l g_{jl}}}$ for $i, j \in \{1, \dots, N\}$.
3: Set $T_{ij} = \frac{k_{ij}}{\sum_l k_{il}}$ for $i, j \in \{1, \dots, N\}$.
4: Determine $\bar{h}$ using (5).
5: **for** $k = 1$ **to** T_{iter} **do**
6: Set $\boldsymbol{\Phi}_k = \boldsymbol{T}\boldsymbol{\Phi}_{k-1} + \epsilon(\tilde{\boldsymbol{h}} - \bar{h})$.
7: Set $\boldsymbol{\Phi}_k = \boldsymbol{\Phi}_k - \frac{1}{N}\sum_{i=1}^{N} \Phi_{k,i}$.
8: **end for**
9: Set $\boldsymbol{\Phi} = \boldsymbol{\Phi}_{T_{\text{iter}}}$.
10: Set

$$\boldsymbol{K}(\boldsymbol{x}_t^i) = \frac{1}{2\epsilon}\sum_{j=1}^{N}\left(T_{ij}(\Phi_j + \epsilon(\tilde{h}_j - \bar{h})(\boldsymbol{x}_t^j - \sum_{k=1}^{N} T_{ik}\boldsymbol{x}_t^k))\right)$$

A. Computational Complexity

The kernel-based approach with the implementation in Algorithm 4 is $\mathcal{O}(N^2)$ similar to Algorithm 3. The fixed-point iterations on Lines 5–8 are typically initialized with the solution $\boldsymbol{\Phi}_0$ from the previous time step. The computational complexity of the kernel and the POD-based method are in the same order, implying that the efficiency of the respective implementation and method in terms of allowing few particles, is going to determine which of the methods to prefer.

IV. Comparison Study

We compare the different methods using three different examples. We also include an EKF (EKF) and a bootstrap PF (PF) in the study. We denote the constant-gain FPF with CG, the POD-based approach with POD, and the kernel-based approach with KERNEL. There are several parameter choices in POD and KERNEL that will affect the performance. For POD, the number of snapshots M is the main parameter to tweak, and the parameter ϵ is the most important for KERNEL. We have not spent much effort in tuning these to each particular example, and it is likely that better performance can be achieved by adjusting the values of these parameters. The FPFs are based on a continuous-time dynamical model, which needs to be discretized in a digital implementation, and the measurements typically arrive at a fixed sampling frequency. To accommodate this and avoid too abrupt changes in the update, the FPFs additionally discretize the measurement update [2]; that is, the gain computation and subsequent measurement update (Algorithms 2–4) are performed in increments. The number of update steps will affect both tracking performance and computation times, and in what follows, the reported computation times include this discretization.

A. Linear Example

This example has previously been used in [2]. Consider the system

$$\begin{aligned} \mathrm{d}x(t) &= -0.5x(t)\mathrm{d}t + \mathrm{d}\beta(t), \\ y_k &= 3x_t + e_k. \end{aligned} \tag{36}$$

The measurements arrive at time instants $t_k = 0.5, 1.0, 1.5, \dots, 10$. In [2], it was shown that the solution to the boundary value problem (7), which for linear systems with Gaussian noise can be expressed in closed form, equals the Kalman gain. We denote the filter using the exact solution with FPFKF. We discretize the measurement update in 20 steps, which according to the results in [2] offers a good compromise between performance and computation time. Furthermore, the dynamics (36) is discretized with a sampling time $\Delta t = 0.005$. In the kernel approach, $\epsilon = 0.1$ and the maximum number of fixed-point iterations is $T_{\text{iter}} = 10$. The number of snapshots in the POD for performing the gain computation is $M = 5$.

Fig. 1 illustrates the gain computation as a function of the particles for a snapshot when using $N = 100$ particles. The mean of the gains for POD and KERNEL are also shown (dashed). The Kalman gain KF, constant-gain approximation CG, and POD POD approach are very similar. The gain for the kernel method is not consistent with the other ones, and the behavior is quite different. The gain function for KERNEL increases with the distance to the particle mean, which is not seen for POD. The POD gains oscillate around the particle mean, but the resulting gain function is mostly flat. However, the mean value of the gain functions are similar. This is consistent with [18], where the gain computation according to Algorithm 4 is shown to be unbiased as $N \to \infty$, which also is true for the other FPFs.

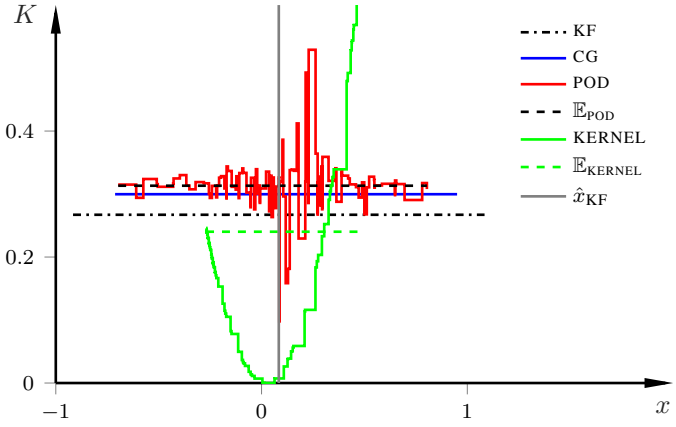

Fig. 1. Comparison of the computed gains from the Kalman filter (KF), the FPF with exact gain computation (FPFKF), the constant-gain approximation in Algorithm 2 (CG), the POD-based gain function approximation in Algorithm 3 (POD), and the kernel-based approach in Algorithm 4 (KERNEL). The number of particles is $N = 100$.

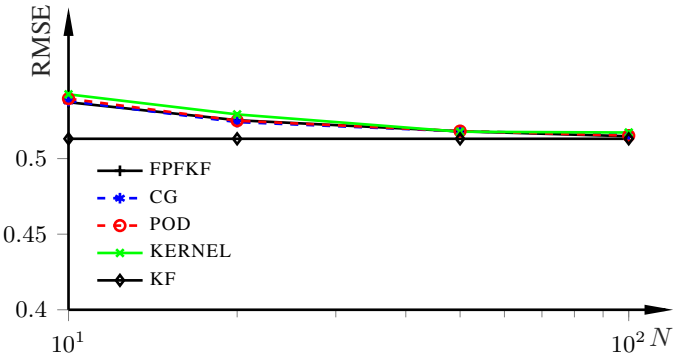

Fig. 2. Time-averaged RMSE for the linear system (36) for varying number of particles N. The data points are from $N = 10, 20, 50, 100$.

Fig. 2 displays the time-averaged RMSE as function of particles taken over 500 Monte-Carlo simulations, where we have also included a standard KF. All FPF approaches perform very similar to each other, and for $N = 100$ particles, the performance of the KF is approximately attained.

B. Benchmark Example

In this evaluation we use the first-order nonlinear model

$$x_{k+1} = 0.5x_k + 25\frac{x_k}{1+x_k^2} + 8\cos(1.2k) + w_k, \tag{37a}$$

$$y_k = 0.05x_k^2 + e_k, \tag{37b}$$

with $w_k \sim \mathcal{N}(0, 10)$, $e_k \sim \mathcal{N}(0, 1)$, $x_0 \sim \mathcal{N}(0.1, 2)$. This model was used in [9], and has since then turned into a benchmark problem for evaluating nonlinear estimators [24], [25]. Note that the state dynamics (37a) is highly nonlinear and that the state is squared in the measurement relation (37b), leading to a bimodal posterior.

The results are based on 500 Monte-Carlo simulations, with 30 time steps in each simulation. The model (37) is already in discrete-time form. Referring to Algorithm 1, the sampling period is set to $\Delta t = 1$ s. We set $M = 5$ in Algorithm 3 and $\epsilon = 0.01$, $T_{\text{iter}} = 10$ in Algorithm 4, and the FPFs perform the measurement update using 20 steps.

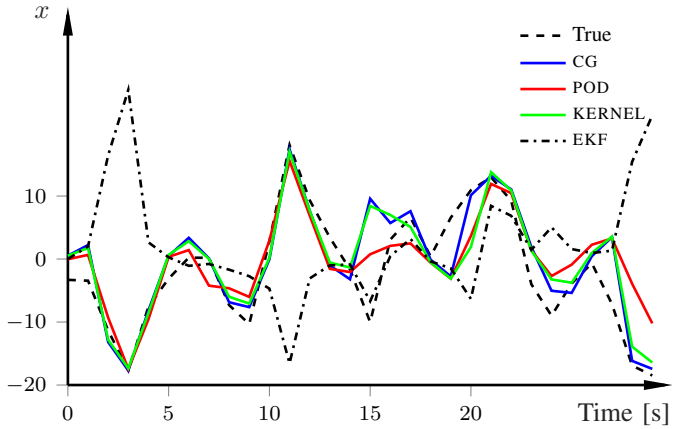

Fig. 3. Estimation performance for one realization of the example in Sec. IV-B. The PFs use 50 particles.

Fig. 3 shows the performance for the respective FPFs in one realization for $N = 50$, together with PF and EKF. The effect of the bimodality can be seen for the EKF, which on several occasions chooses the wrong sign to correct with the measurement.

Fig. 4 shows the RMSE values for 50 particles. KERNEL performs best among the different filters, slightly better than POD. As expected, the EKF performs very poorly in comparison the with the PFs, and using the prior as proposal density in the bootstrap PF leads to inefficiency. Furthermore, the FPF with constant gain, CG, underperforms relative to POD and KERNEL.

To illustrate the tracking performance versus computation time, Fig. 5 displays the time-averaged RMSE as function of average computation time for one prediction and measurement update, for a varying number of particles. The algorithms are C-coded, but no consideration has been taken to code optimization. It is preferable to be located in the lower-left corner of the figure.

The kernel approach KERNEL performs best in terms of tracking error irrespective of the number of particles. However, this comes at a much higher computational cost. When taking the computational cost into consideration, the constant-gain approximation becomes an attractive option. The POD approach consistently performs well, even for few particles, whereas the constant-gain filter becomes inefficient as the computation time is decreased. This example shows that the POD and kernel has better tracking performance for highly nonlinear examples, both approaches make good use of the particles even for very few particles, but, especially for the kernel approach, the performance comes with a high computational cost.

C. Coordinated Turn

In this example, a target moves in a plane according to a clockwise coordinated turn [26] of radius 500 m with constant velocity 200 km/h. The initial position is $\boldsymbol{p}_0 = [-500\ 500]^{\text{T}}$,

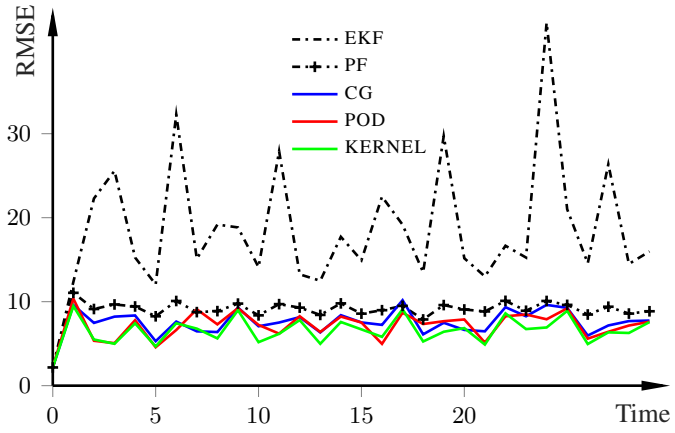

Fig. 4. RMSE values for the benchmark example in Sec. IV-B for $N = 50$.

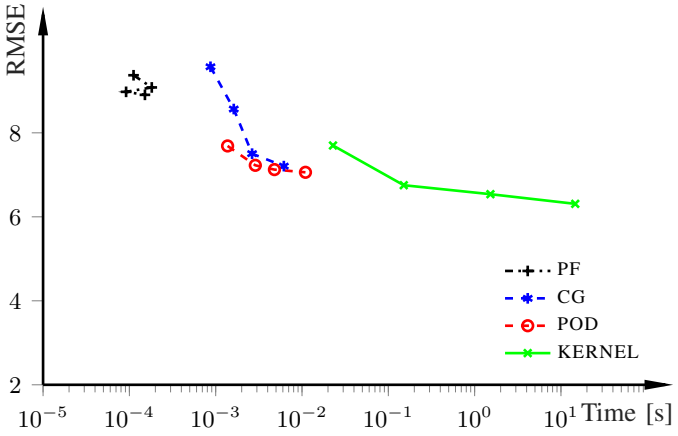

Fig. 5. Computation time versus time-averaged RMSE values for the benchmark example in Sec. IV-B. The data points correspond to $N = 10, 20, 50, 100$.

starting in the x-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 is

$$\begin{bmatrix} \dot{p}^X \\ \dot{p}^Y \\ \dot{v}^X \\ \dot{v}^Y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p^X \\ p^Y \\ v^X \\ v^Y \\ \omega \end{bmatrix} + \boldsymbol{w}, \qquad (38)$$

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 [25] 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 $\Delta t = 0.01$ s is the sampling period. The FPFs use 100 discretization steps in the measurement updates. POD uses the last $M = 40$ data points for basis computation (see Algorithm 3) and KERNEL uses $\epsilon = 10^2$. The process noise $\boldsymbol{w}$ is zero mean Gaussian with covariance $\boldsymbol{Q} = \mathrm{diag}([0.1^2, 0.1^2, 0.1^2, 0.1^2, 0.01^2])$. We set the initial estimate for all filters to $\boldsymbol{x}_0 = [-500, 500, 55, 0, 0]^{\mathrm{T}}$, with initial covariance $\boldsymbol{P}_0 = \mathrm{diag}([25^2, 25^2, 3^2, 3^2, 0.1^2])$, that is, we know very little about the initial state of the target. Two sensors measure the range to the target with sampling time $T_s = 1$ s. The sensors are located at $S_1 = (-200, 0)$ and $S_2 = (200, 0)$. The measurement model is

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

The measurement noise for each sensor is Gaussian zero mean with standard deviation $\sigma_j = 1$ m. The kernel approach assumes scalar-valued updates. This is here solved by computing the two columns in $\boldsymbol{K}$ separately for each measurements, that is, executing Algorithm 4 once for each measurement.

1) Results: Fig. 6 shows the estimation results for one realization, with $N = 200$. The feedback correction is clearly seen for the FPFs, whereas PF uses too few particles to recover when it has started to diverge. Fig. 7 show the time-averaged RMSE of the velocity as the number of particles varies. In this example, KERNEL performs worse than the other FPFs when using few particles.

V. Concluding Discussion

We compared three of the recently proposed gain-approximation methods in the FPF. We used three examples to assess the performance of the algorithms. Several conclusions can be drawn from the study. First, the constant-gain approximation seems to be a competitive approximation method. It is computationally fast ($\mathcal{O}(N)$) and performed similar to the POD-based and kernel-based approaches with respect to tracking performance in two out of the three examples. The POD and kernel approaches clearly outperformed the constant-gain approximation method on the highly nonlinear benchmark example, which indicates that the models need to be sufficiently nonlinear or multimodal for these methods to be preferred.

The constant-gain approximation is simpler to implement since it has fewer parameters to tune. The POD performance depends on the number of time steps to base the basis extraction on. A sensible choice of this parameter will depend on the time scales of the dynamics. Similarly, ϵ is an important parameter for the kernel approach, as clear from the evaluation, and it looks as if the performance is rather sensitive to the choice of ϵ. However, if tuned correctly, as seen from the benchmark example in Sec. IV-B, the performance benefits may be substantial.

References

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

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

[3] T. Yang, R. S. Laugesen, P. G. Mehta, and S. P. Meyn, "Multivariable feedback particle filter," *Automatica*, vol. 71, pp. 10–23, 2016.

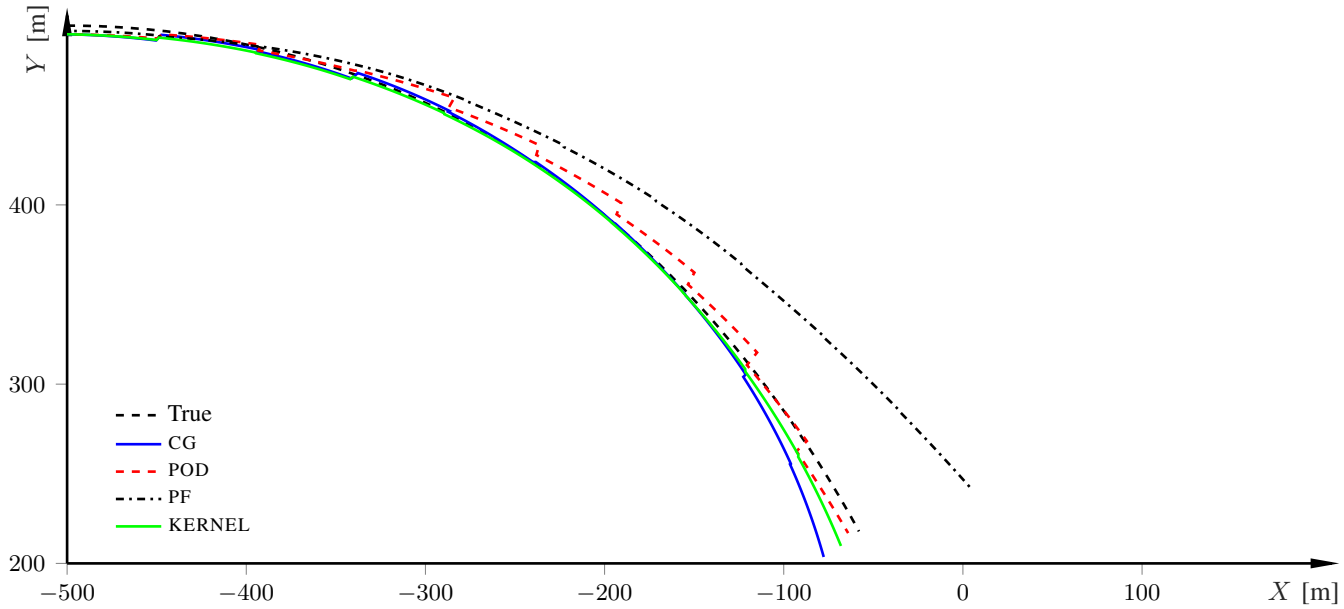

Fig. 6. Estimation results for one realization of the coordinated turn problem, with $N = 200$.

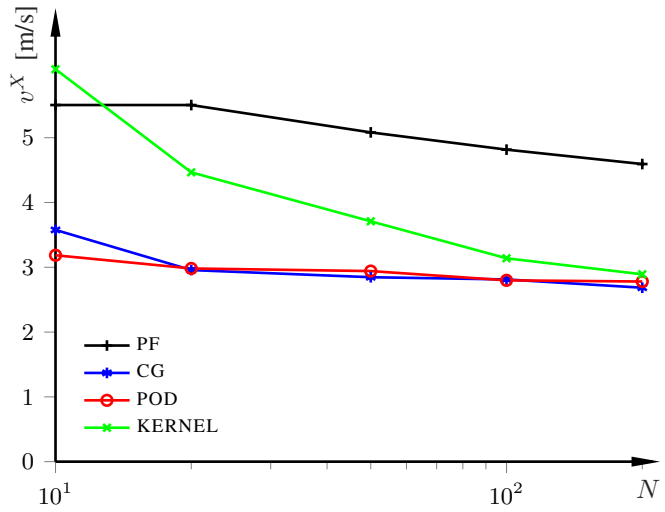

Fig. 7. Time-averaged RMSEs of the velocity for varying number of particles for the coordinated turn problem. The RMSE values are computed for $N = 20, 50, 100, 200$.

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

[5] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[13] A. Radhakrishnan, A. Devraj, and S. Meyn, "Learning techniques for feedback particle filter design," in *Conf. Decision and Control*, Las Vegas, NV, Dec. 2016.

[14] Y. Matsuura, R. Ohata, K. Nakakuki, and R. Hirokawa, "Suboptimal gain functions of feedback particle filter derived from continuation method," in *AIAA Guidance, Navigation, and Control Conf.*, San Diego, CA, Jan. 2016.

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

[16] ——, "Feedback particle filter with data-driven gain-function approximation," *IEEE Trans. Aerosp. Electron. Syst.*, 2018, in press.

[17] A. Taghvaei and P. G. Mehta, "Gain function approximation in the feedback particle filter," in *Conf. Decision and Control*, Las Vegas, NV, Dec. 2016.

[18] A. Taghvaei, P. G. Mehta, and S. P. Meyn, "Error estimates for the kernel gain function approximation in the feedback particle filter," in *Amer. Control Conf.*, Seattle, WA, May 2017.

[19] A. Taghvaei and P. G. Mehta, "An optimal transport formulation of the linear feedback particle filter," in *Amer. Control Conf.*, Boston, MA, Jul. 2016.

[20] 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.

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

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

[23] D. Bakry, I. Gentil, and M. Ledoux, *Analysis and geometry of Markov diffusion operators*. Springer, 2013, vol. 348.

[24] K. Berntorp, A. Robertsson, and K.-E. Årzén, "Rao-Blackwellized particle filters with out-of-sequence measurement processing," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6454–6467, 2014.

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

[26] X. R. Li and V. P. Jilkov, "A survey of maneuvering target tracking - part III: Measurement models," in *SPIE Conf. Signal and Data Process. of Small Targets*, San Diego, CA, 2001.