Machine Learning (2019) 108:2113–2157
https://doi.org/10.1007/s10994-019-05816-z

The kernel Kalman rule

Efficient nonparametric inference by recursive least-squares and subspace projections

Gregor H. W. Gebhardt[1] · Andras Kupcsik[2] · Gerhard Neumann[2,3]

Received: 4 September 2017 / Revised: 16 May 2019 / Accepted: 20 May 2019 / Published online: 18 June 2019

Abstract
Enabling robots to act in unstructured and unknown environments requires versatile state estimation techniques. While traditional state estimation methods require known models and make strong assumptions about the dynamics, such versatile techniques should be able to deal with high dimensional observations and non-linear, unknown system dynamics. The recent framework for nonparametric inference allows to perform inference on arbitrary probability distributions. High-dimensional embeddings of distributions into reproducing kernel Hilbert spaces are manipulated by kernelized inference rules, most prominently the kernel Bayes' rule (KBR). However, the computational demands of the KBR do not scale with the number of samples. In this paper, we present two techniques to increase the computational efficiency of non-parametric inference. First, the kernel Kalman rule (KKR) is presented as an approximate alternative to the KBR that estimates the embedding of the state based on a recursive least squares objective. Based on the KKR we present the kernel Kalman filter (KKF) that updates an embedding of the belief state and learns the system and observation models from data. We further derive the kernel forward backward smoother (KFBS) based on a forward and backward KKF and a smoothing update in Hilbert space. Second, we present the subspace conditional embedding operator as a sparsification technique that still leverages from the full data set. We apply this sparsification to the KKR and derive the corresponding sparse KKF and KFBS algorithms. We show on nonlinear state estimation tasks that our approaches provide a significantly improved estimation accuracy while the computational demands are considerably decreased.

Keywords Nonparametric inference · Kernel methods · State estimation · Model learning

Editor: Scott Sanner.

✉ Gregor H. W. Gebhardt
mail@gregor-gebhardt.de

Extended author information available on the last page of the article

1 Introduction

The ability to reason about past, current and future states in continuous, partially observable stochastic processes is a fundamental stepstone towards fully autonomos and intelligent systems. Such models are required in many applications as for example state estimation in case of incomplete sensory data, smoothing noisy data from mediocre sensors, or predicting future states from past and current observations.

Traditional state estimation techniques usually require analytical models of the underlying system, are often limited to a set of models with a special structure, and require knowledge about the moments of the stochstic processes. When assuming linear Gaussian models with known mean and covariance for instance, the Kalman filter (Kalman 1960) yields the optimal solution. However, the required linear Gaussian models with known statistics impose a strong limitation to the applicability of this method. For more complex processes, approximate solutions have to be used instead. Examples are the extended Kalman filter (McElhoe 1966; Smith et al. 1962) or the unscented Kalman filter (Julier and Uhlmann 1997; Wan and Van Der Merwe 2000). These solution inherit the Gaussian representation of the belief state to which they apply the non-linear system dynamics. However, the Gaussian distribution with its unimodal nature is a strong assumption about the belief state which leads to poor results for systems that require a more complex distribution over possible states. Moreover, both the Kalman filter but also it's extensions to non-linear systems, require that the dynamics of the systems are given as analytical models. Yet, these analytical models are often hard to obtain or make simplifying assumptions about the system.

The recently introduced framework for nonparametric inference (Song et al. 2013; Fukumizu et al. 2013) alleviates the problems of traditional state estimation methods for nonlinear systems. The basic idea of these methods is to embed the probability distributions into reproducing kernel Hilbert spaces (RKHS). These embeddings allow the representation of arbitrary probability distributions using empirical estimators. Inference on the embedded distribution can then be performed efficiently and entirely in the RKHS using the kernelized versions of the sum rule, the chain rule, and the Bayes' rule. Additionally, Song et al. (2013) use the kernel sum rule and the kernel Bayes' rule to construct the kernel Bayes' filter (KBF). The KBF learns the transition and observation models from observed samples and can be applied to nonlinear systems with high-dimensional observations. However, the computational complexity of the KBR update does not scale well with the number of samples such that hyper-parameter optimization becomes prohibitively expensive. Moreover, the KBR requires mathematical tricks that may cause numerical instabilities and also render the objective that is optimized by the KBR unclear.

In this paper, we present two approaches to overcome the limitations named above. First, we introduce the *subspace conditional embedding operator*. In contrast to the conditional embedding operator (Song et al. 2009), this operator allows to estimate its empirical estimator with a much larger data set while maintaining computational efficiency. We further apply the subspace conditional embedding operator to the kernel sum rule, kernel chain rule and kernel Bayes rule to derive their subspace versions. We have presented these results at the large-scale kernel learning workshop at ICML 2015 (Gebhardt et al. 2015).

Furthermore, we present the *kernel Kalman rule* (KKR) as an approximate alternative to the kernel Bayes' rule. Our derivations closely follow the derivations of the innovation update used in the Kalman filter and are based on a recursive least squares minimization objective in a reproducing kernel Hilbert space (RKHS). The KKR does not perform an exact Bayesian update as it uses a regularization term in the least squares objective and assumes constant noise on the conditioning variable. While the update equations are formulated in a potentially infinite dimensional RKHS, we derive through application of the kernel trick and by virtue of the representer theorem an algorithm that uses only operations of finite kernel matrices and vectors. We employ the kernel Kalman rule together with the kernel sum rule for filtering, which results in the *kernel Kalman filter* (KKF). In contrast to filtering techniques that rely on the KBR, the KKF allows to precompute expensive matrix inversions which significantly reduces the computational complexity and which also allows us to apply hyper-parameter optimization for the KKF. This work has been presented at AAAI 2017 (Gebhardt et al. 2017).

In addition to the KKF, we introduce the *kernel forward backward smoother* (KFBS) which computes the embedding of the belief state given all available observations from the past and the future. The kernel forward backward smoother combines the belief state embeddings of a forward pass and a backward pass into smoothed embeddings using Hilbert space operations. Both, the forward and the backward pass are realized by a KKF, where the backward KKF operates backwards in time starting at the last observation. To scale gracefully with larger data sets, we rederive the KKR, the KKF and the KFBS with the subspace conditional operator (Gebhardt et al. 2015).

We compare our approach to different versions of the KBR and demonstrate its improved estimation accuracy and computational efficiency. Furthermore, we evaluate the KKR on a simulated 4-link pendulum task, on a human motion capture data set (Wojtusch and von Stryk 2015) and on data from a table-tennis setup (Gomez-Gonzalez et al. 2016).

1.1 Related work

To the best of our knowledge the kernel Bayes' rule exists in three different versions. It was first introduced in its original version by Fukumizu et al. (2013). Here, the KBR is derived, similar to the conditional operator, using prior modified covariance operators. These prior-modified covariance operators are approximated by weighting the feature mappings with the weights of the embedding of the prior distribution. Since these weights are potentially negative, the covariance operator might become indefinite, and thus, rendering its inversion impossible. To overcome this drawback, the authors have to apply a form of the Tikhonov regularization that decreases accuracy and increases the computational costs. A second version of the KBR was introduced by Song et al. (2013) in which they use a different approach to approximate the prior-modified covariance operator. In the experiments conducted for this paper, this second version often leads to more stable algorithms than the first version. Boots et al. (2013) introduced a third version of the KBR where they apply only the simple form of the Tikhonov regularization. However, this rule requires the inversion of a matrix that is often indefinite, and therefore, high regularization constants are required, which again degrades the performance. In our experiments, we refer to these different versions with KBR(b) for the first, KBR(a) for the second (order adapted from the literature), and KBR(c) for the third version. Song et al. (2013) propose in their framework for nonparametric inference to combine the KBR with the kernel sum rule to obtain the kernel Bayes filter (KBF). The kernel Kalman filter presented in this work is closely related to this, as we simply replace the KBR with the KKR. We compare to the KBF in our experiments. Nishiyama et al. (2016) recently proposed the nonparametric kernel Bayes smoother. This approach builds on top of the kernel Bayes filter, which is used to compute the estimates of a normal forward pass. The smoothing update is then obtained by propagating the embeddings backwards in time without performing a second filtering pass.

For filtering tasks with known linear system equations and Gaussian noise, the Kalman filter (KF) yields the solution that minimizes the squared error of the estimate to the true state.

Two widely known and applied approaches to extend the Kalman filter to non-linear systems are the *extended Kalman filter* (EKF) (McElhoe 1966; Smith et al. 1962) and the *unscented Kalman filter* (UKF) (Wan and Van Der Merwe 2000; Julier and Uhlmann 1997). Both, the EKF and the UKF, assume that the non-linear system dynamics are known and use them to update the prediction mean. Yet, updating the prediction covariance is not straightforward. In the EKF the system dynamics are linearized at the current estimate of the state, and in the UKF the covariance is updated by applying the system dynamics to a set of sample-points (sigma points). While these approximations make the computations tractable, they can significantly reduce the quality of the state estimation, in particular for high-dimensional systems.

Hsu et al. (2012) recently proposed an algorithm for learning Hidden Markov Models (HMMs) by exploiting the spectral properties of observable measures to derive an observable representation of the HMM (Jaeger 2000). An RKHS embedded version thereof was presented by Song et al. (2010). While this method is applicable for continuous state spaces, it still assumes a finite number of discrete hidden states.

Other closely related algorithms to our approach are the kernelized version of the Kalman filter and Kalman smoother by Ralaivola and d'Alche Buc (2005) and the kernel Kalman filter based on the conditional embedding operator (KKF-CEO) by Zhu et al. (2014). The former approach formulates the Kalman filter in a sub-space of the infinite feature space that is defined by the kernels. Hence, this approach does not fully leverage the kernel idea of using an infinite feature space. In contrast, the KKF-CEO approach embeds the belief state also in an RKHS. However, they require that the observation is a noisy version of the full state of the system, and thus, they cannot handle partial observations. Moreover, they also deviate from the standard derivation of the Kalman filter, which—as our experiments show—decreases the estimation accuracy. The full observability assumption is needed in order to implement a simplified version of the innovation update of the Kalman filter in the RKHS. The KKF does not suffer from this restriction. It also provides update equations that are much closer to the original Kalman filter and outperforms the KKF-CEO algorithm as shown in our experiments. Another approach to state estimation is presented in (Kawahara et al. 2007), where the authors propose to estimate low-dimensional state vectors based on kernel canonical correlation analysis and then regress a linear transition model of the estimated state vectors and the nonlinear features of the input.

Learning predictors in the space of predictive state representations to perform filtering has been proposed in Sun et al. (2016b) and later extended to smoothing in Sun et al. (2016a). They introduce predictive state inference machines (PSIM) which are (nonlinear) regressors on predictive states learned from data to perform filtering. With the smoothing machine (SMACH) they extend this concept for smoothing.

1.2 Structure of the paper

The remainder of the paper is structured as follows: in Sect. 2, we discuss the framework for non-parametric inference (Song et al. 2013) and the Kalman filter as foundations for the work we present in this paper; in Sect. 3 we introduce the *subspace conditional embedding operator* and show its application to the framework for non-parametric inference; in Sect. 4 we present the *kernel Kalman rule* and the *subspace kernel Kalman rule*; in Sect. 5 we introduce the *(subspace) kernel Kalman filter* (Sect. 5.1) and the *(subspace) kernel forward backward smoother* (Sect. 5.4) before we conclude the paper in Sect. 6. Experimental evaluations of all proposed methods are shown and discussed directly in the respective sections.

2 Preliminaries

Our work is based on the recent formulations of embedding distributions into reproducing kernel Hilbert spaces (Smola et al. 2007; Song et al. 2013). These embeddings allow to represent arbitrary probability distributions non-parametrically by a potentially infinite dimensional feature vector. Through the application of derived operators (Song et al. 2009; Fukumizu et al. 2013) it is furthermore possible to apply inference rules entirely in the Hilbert space. In the first part of this section, we want to give the reader an introduction into this technology and define the notation we will use throughout this article.

One of the main contributions of our paper is a novel method for performing approximate Bayesian updates on a distribution embedded into an RKHS. The derivations of this update rule are based on a least-squares objective and inspired by the derivations of the Kalman filter update, thus we name this method *kernel Kalman rule*. In the second part of this section, we will recapitulate the classical Kalman filter equations and review the derivations of the innovation update based on the least-squares objective.

2.1 Nonparametric inference with Hilbert space embeddings of distributions

Intuitively, a Hilbert space is an extension of the well known two- or three-dimensional Euclidean vector space to arbitrary many dimensions, specifically including infinite dimensional vector spaces. Such infinite dimensional Hilbert spaces include spaces where the single elements are functions, i.e., infinite dimensional vectors that contain for each element of the domain the corresponding function value in the image. In addition, a Hilbert space has an inner product that allows to measure distances and angles between its elements. For a reproducing kernel Hilbert space $\mathcal{H}_k$, this inner product $\langle \cdot, \cdot \rangle$ is implicitly defined by a reproducing kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}') \rangle$, where $\varphi(\boldsymbol{x})$ is a feature mapping into a possibly infinite dimensional space, intrinsic to the kernel function. For example the Gaussian kernel computes the inner product of the feature mappings of its inputs where the feature mappings itself cannot be written down explicitly as they are into an infinite dimensional space. Due to the reproducing property of the kernel, all elements f of the RKHS can be reproduced by k in the sense that the outcome $f(x)$ of the function for a specific value x can obtained by an evaluation of the kernel function (Aronszajn 1950), i.e., $f(\boldsymbol{x}) = \langle f, \varphi(\boldsymbol{x}) \rangle$ for any $f \in \mathcal{H}_k$.

In a practical setting, we want to embed probability distributions in an RKHS spanned by samples $\mathcal{D}_X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Based on the representer theorem (Schölkopf et al. 2001) and the reproducing property, the elements f of an RKHS $\mathcal{H}_k$ can then be written as

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \cdot) = \sum_{i=1}^{n} \alpha_i \langle \varphi(\boldsymbol{x}_i), \varphi(\cdot) \rangle = \boldsymbol{\alpha}^\intercal \boldsymbol{\Upsilon}_{\boldsymbol{x}}^\intercal \varphi(\cdot), \tag{1}$$

with the weights $\alpha_i \in \mathbb{R}$ and where we denote the feature matrix of samples $\boldsymbol{x}_i$ by $\boldsymbol{\Upsilon}_{\boldsymbol{x}} = [\varphi(\boldsymbol{x}_1), \ldots, \varphi(\boldsymbol{x}_n)]$. In the following paragraphs we will show how probability distributions can be represented as an embedding in such a reproducing kernel Hilbert space and how the operators for performing inference in the RKHS can be derived.

2.1.1 Embeddings of marginal and joint distributions

The embedding of a marginal density $P(X)$ over the random variable X is defined as the expected feature mapping $\mu_X := \mathbb{E}_X[\varphi(X)]$, also called the *mean map* (Smola et al. 2007). Using a finite set of samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ from $P(X)$, the mean map can be estimated as

$$\hat{\mu}_X = \frac{1}{n}\sum_{i=1}^{n}\varphi(\boldsymbol{x}_i) = \frac{1}{n}\boldsymbol{\Upsilon}_x^\intercal \mathbf{1}_n, \tag{2}$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is an n dimensional vector of ones. Because of the reproducing property of the kernel function, computing the expectation of a function which is an element of the same RKHS resolves to simple matrix operations. On the other hand, the probability of a single outcome or higher order statistics of the distributions are not straight forward to obtain.

Alternatively, a distribution can be embedded in a tensor product RKHS $\mathcal{H}_k \times \mathcal{H}_k$ as the expected tensor product of the feature mappings (Smola et al. 2007)

$$\mathcal{C}_{XX} := \mathbb{E}_{XX}\left[\varphi(X)\otimes\varphi(X)\right] - \mu_X \otimes \mu_X, \tag{3}$$

where we use $\otimes$ to denote the tensor product (or outer product) of two vectors. This embedding is also called the *centered covariance operator*. The finite sample estimator is given by

$$\hat{\mathcal{C}}_{XX} = \frac{1}{m}\sum_{i=1}^{m}\varphi(\boldsymbol{x}_i)\otimes\varphi(\boldsymbol{x}_i) - \hat{\mu}_X \otimes \hat{\mu}_X. \tag{4}$$

Similarly, we can define the *uncentered cross-covariance operator* for a joint distribution $p(X, Y)$ of two variables X and Y as $\hat{\mathcal{C}}_{XY} = \frac{1}{m}\sum_{i=1}^{m}\varphi(\boldsymbol{x}_i)\otimes\phi(\boldsymbol{y}_i)$. Here, we have used a data set of tuples $\mathcal{D}_{XY} = \left\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\right\}$ sampled from $p(X, Y)$ and a second RKHS $\mathcal{H}_g$ with kernel function $g(\boldsymbol{y}, \boldsymbol{y}') =: \langle\phi(\boldsymbol{y}), \phi(\boldsymbol{y}')\rangle$.

2.1.2 The conditional embedding operator

The embedding of a conditional distribution $P(Y|X)$ is not like the mean map a single element of the RKHS, but rather a family of embeddings that yields a mean embedding for each realization of the conditioning variable X. To obtain the conditional distribution for a specific value $X = \boldsymbol{x}_*$, Song et al. (2009) defined the *conditional embedding operator* $\mathcal{C}_{Y|X}$ which, if applied to the feature mapping of $\boldsymbol{x}$, returns the embedding of $P(Y|X = \boldsymbol{x}_*)$

$$\mu_{Y|\boldsymbol{x}} := \mathbb{E}_{Y|\boldsymbol{x}}\left[\phi(Y)\right] = \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}). \tag{5}$$

Using the data set $\mathcal{D}_{XY}$ from the joint distribution, an estimator of the conditional embedding operator can be derived from a least-squares objective (Grünewälder et al. 2012) as

$$\hat{\mathcal{C}}_{Y|X} = \boldsymbol{\Phi}(\boldsymbol{K}_{xx} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{\Upsilon}_x^\intercal, \tag{6}$$

with the feature matrices $\boldsymbol{\Phi} := [\phi(\boldsymbol{y}_1), \ldots, \phi(\boldsymbol{y}_n)]$ and $\boldsymbol{\Upsilon}_x := [\varphi(\boldsymbol{x}_1), \ldots, \varphi(\boldsymbol{x}_n)]$, the Gram matrix $\boldsymbol{K}_{xx} = \boldsymbol{\Upsilon}_x^\intercal\boldsymbol{\Upsilon}_x \in \mathbb{R}^{n\times n}$, the regularization parameter λ, and the identity matrix $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$. With the feature mapping of the realization $\boldsymbol{x}_*$ this results in

$$\hat{\mu}_{Y|\boldsymbol{x}_*} = \hat{\mathcal{C}}_{Y|X}\varphi(\boldsymbol{x}_*) = \boldsymbol{\Phi}(\boldsymbol{K}_{xx} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{\Upsilon}_x^\intercal\varphi(\boldsymbol{x}_*) = \boldsymbol{\Phi}(\boldsymbol{K}_{xx} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{k}_{\boldsymbol{x}_*}, \tag{7}$$

where $\boldsymbol{k}_{\boldsymbol{x}_*} = [k(\boldsymbol{x}_1, \boldsymbol{x}_*), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}_*)]^\intercal$ is the kernel vector of the samples $\boldsymbol{x}_i$ and the realization $\boldsymbol{x}_*$. As the kernel matrices in the inverse and the kernel vector of the realization are finite, the embedding of the conditional distribution can be represented as a weighted sum of feature mappings

$$\hat{\mu}_{Y|\boldsymbol{x}_*} = \boldsymbol{\Phi}\boldsymbol{\alpha} = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{y}_i), \tag{8}$$

with the finite weight vector $\boldsymbol{\alpha} \in \mathbb{R}^n$. Based on the two definitions for Hilbert space embeddings of probability distributions and the conditional embedding operator discussed above, all the rules of the framework for non-parametric inference (Song et al. 2013) can be derived.

2.1.3 The kernel sum rule

The embedding of $Q(Y) = \sum_X P(Y|X)\pi(X)$ can be obtained from the *kernel sum rule* (Song et al. 2013). To that end, the conditional operator is applied to the embedding $\hat{\mu}_X^\pi = \boldsymbol{\Upsilon}_x \boldsymbol{\alpha}_\pi$ of the prior distribution $\pi(X)$,

$$\hat{\mu}_Y^\pi = \hat{\mathcal{C}}_{Y|X}\hat{\mu}_X^\pi = \boldsymbol{\Phi}(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\boldsymbol{\alpha}_\pi. \tag{9}$$

Again, the result can be represented as a weighted sum over feature mappings. In order to obtain the distribution $Q(Y)$ as a covariance operator instead of a mean map, Song et al. (2013) also proposed the kernel sum rule for tensor product features which yields the prior modified covariance operator $\mathcal{C}_{YY}^\pi$ as

$$\mathcal{C}_{YY}^\pi = \mathcal{C}_{(YY)|X}\mu_X^\pi \tag{10}$$

$$\hat{\mathcal{C}}_{YY}^\pi = \boldsymbol{\Phi}\mathrm{diag}((\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\boldsymbol{\alpha})\boldsymbol{\Phi}^\intercal, \tag{11}$$

where $\mathcal{C}_{(YY)|X}$ is the conditional operator for tensor product features.

2.1.4 The kernel chain rule

The *kernel chain rule* (Song et al. 2013) yields an embedding of the joint distribution $Q(X, Y) = P(Y|X)\pi(X)$ as a prior modified covariance operator. There are two versions of the kernel chain rule. Both apply the conditional embedding operator of $P(Y|X)$ to an embedding of the prior distribution $\pi(X)$. In the first version the conditional operator is applied to a covariance embedding of the prior distribution. This covariance operator is not estimated directly from samples but approximated from the weight vector $\boldsymbol{\alpha}_\pi$ of the embedding of the prior distributions as $\hat{\mathcal{C}}_{XX}^\pi = \boldsymbol{\Upsilon}_x\mathrm{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^\intercal$. This yields Version (a) of the kernel chain rule as

$$\hat{\mathcal{C}}_{YX}^\pi = \hat{\mathcal{C}}_{Y|X}\hat{\mathcal{C}}_{XX}^\pi = \boldsymbol{\Phi}(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\mathrm{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^\intercal. \tag{12}$$

Version (b) of the kernel chain rule first computes the mean map μ_Y^π conditioned on the prior distribution $\pi(X)$ by applying the conditional embedding operator to the mean map μ_X^π. Afterwards, the prior-modified covariance operator of the joint distribution is constructed from the resulting weight vector which results in

$$\hat{\mathcal{C}}_{YX}^\pi = \boldsymbol{\Phi}\mathrm{diag}((\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^\intercal. \tag{13}$$

Both versions of the kernel chain rule have been used to derive different versions of the kernel Bayes' rule as we will depict below.

2.1.5 The kernel Bayes' rule

Given the embedding of a prior distribution $\pi(X)$ and the feature mapping of an observation $\phi(\boldsymbol{y}_*)$, the *kernel Bayes' rule* (KBR) infers the mean embedding of the posterior distribution $Q_\pi(X|Y = \boldsymbol{y}_*)$. The idea is to construct a prior-modified conditional embedding operator that yields the mean map of the posterior if applied to the feature mapping of the observation (Fukumizu et al. 2013)

$$\mu^{\pi}_{X|\boldsymbol{y}} = \mathcal{C}^{\pi}_{X|Y}\phi(\boldsymbol{y}_*). \tag{14}$$

This prior-modified conditional operator is constructed from two prior-modified covariance operators $\mathcal{C}^{\pi}_{YY}$ and $\mathcal{C}^{\pi}_{YX}$ obtained from the kernel sum and the kernel chain rule, respectively, using the relation

$$\mathcal{C}^{\pi}_{X|Y} = \mathcal{C}^{\pi}_{XY}\left(\mathcal{C}^{\pi}_{YY}\right)^{-1}. \tag{15}$$

In the first version, which we denote by KBR(b) following the notation of Song et al. (2013), Fukumizu et al. (2013) derived the kernel Bayes' rule using the tensor product conditional operator in the kernel chain rule (c.f. Eq. 13) and arrived at

$$\hat{\mu}^{\pi}_{X|y} = \boldsymbol{\Upsilon}_x \boldsymbol{D}\boldsymbol{G}_{yy}\left((\boldsymbol{D}\boldsymbol{G}_{yy})^2 + \kappa \boldsymbol{I}_n\right)^{-1} \boldsymbol{D}\boldsymbol{g}_{\boldsymbol{y}_*}, \tag{16}$$

with the diagonal $\boldsymbol{D} := \operatorname{diag}((\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\boldsymbol{\alpha}_\pi)$, the gram matrix $\boldsymbol{G}_{yy} = \boldsymbol{\Phi}^\intercal\boldsymbol{\Phi}$, the kernel vector $\boldsymbol{g}_{\boldsymbol{y}_*} = [g(\boldsymbol{y}_1, \boldsymbol{y}_*), \ldots, g(\boldsymbol{y}_n, \boldsymbol{y}_*)]^\intercal$ and κ as regularization parameter. Song et al. (2013) derived the KBR using the first formulation of the kernel chain rule shown in Eq. 12 which results in

$$\hat{\mu}^{\pi}_{X|y} = \boldsymbol{\Upsilon}_x \boldsymbol{\Lambda}^\intercal\left((\boldsymbol{D}\boldsymbol{G}_{yy})^2 + \kappa \boldsymbol{I}_n\right)^{-1} \boldsymbol{G}_{yy}\boldsymbol{D}\boldsymbol{g}_{\boldsymbol{y}_*}, \tag{17}$$

with $\boldsymbol{\Lambda} := (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{K}_{xx}\operatorname{diag}(\boldsymbol{\alpha}_\pi)$. This second version of the kernel Bayes' rule is denoted by KBR(a). As the matrix $\boldsymbol{D}\boldsymbol{G}_{yy}$ is typically not invertible, both of these versions of the KBR use a form of the Tikhonov regularization in which the matrix in the inverse is squared. Boots et al. (2013) use a third form of the KBR which is derived analogously to the first version but does not use the squared form of Tikhonov regularization, i.e.,

$$\hat{\mu}^{\pi}_{X|y} = \boldsymbol{\Upsilon}_x\left(\boldsymbol{D}\boldsymbol{G}_{yy} + \kappa \boldsymbol{I}_n\right)^{-1} \boldsymbol{D}\boldsymbol{g}_{\boldsymbol{y}_*}. \tag{18}$$

Since the product $\boldsymbol{D}\boldsymbol{G}_{yy}$ is often not positive definite, a strong regularization parameter is required to make the matrix invertible. We denote this third version of the kernel Bayes' rule consequently by KBR(c).

2.2 The Kalman filter

The Kalman filter is a well known technique for state estimation, prediction, and smoothing in environments with linear system dynamics that are subject to zero-mean Gaussian noise with known covariances (Kalman 1960). The system equations can be formulated as

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{v}_t, \quad \boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{w}_t, \tag{19}$$

where $\boldsymbol{x}_t$ is the latent state of the system at time t and $\boldsymbol{y}_t$ is the corresponding observation. The linear Gaussian model is defined by the system matrix $\boldsymbol{F}$, the observation matrix $\boldsymbol{H}$, and noise vectors $\boldsymbol{v}_t$ and $\boldsymbol{w}_t$ which are sampled from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{P})$ and $\mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$, respectively.

From the assumption of Gaussian transition noise and Gaussian observation noise, it follows that the belief state over the latent state $\boldsymbol{x}_t$ is as well a Gaussian distribution with mean $\boldsymbol{\eta}_{\boldsymbol{x},t}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{x},t}$. The Kalman filter applies iteratively two update procedures to the belief state to which we will refer to as *prediction* and *innovation* update. During the prediction update the Kalman filter propagates the belief state in time by applying the transition model, i.e.,

$$\boldsymbol{\eta}^{-}_{\boldsymbol{x},t+1} = \boldsymbol{F}\boldsymbol{\eta}^{+}_{\boldsymbol{x},t}, \quad \boldsymbol{\Sigma}^{-}_{\boldsymbol{x},t+1} = \boldsymbol{F}\boldsymbol{\Sigma}^{+}_{\boldsymbol{x},t}\boldsymbol{F}^{\intercal} + \boldsymbol{P}. \tag{20}$$

On new observations $\boldsymbol{y}_t$, the innovation update applies Bayes' theorem to the *a-priori* belief state $\{\boldsymbol{\eta}^{-}_{\boldsymbol{x},t}, \boldsymbol{\Sigma}^{-}_{\boldsymbol{x},t}\}$ to obtain the *a-posteriori* mean and covariance as

$$\boldsymbol{\eta}^{+}_{\boldsymbol{x},t} = \boldsymbol{\eta}^{-}_{\boldsymbol{x},t} + \boldsymbol{Q}_t(\boldsymbol{y}_t - \boldsymbol{H}\boldsymbol{\eta}^{-}_{\boldsymbol{x},t}), \tag{21}$$

$$\boldsymbol{\Sigma}^{+}_{\boldsymbol{x},t} = \boldsymbol{\Sigma}^{-}_{\boldsymbol{x},t} - \boldsymbol{Q}_t\boldsymbol{H}\boldsymbol{\Sigma}^{-}_{\boldsymbol{x},t}, \tag{22}$$

with the Kalman gain matrix

$$\boldsymbol{Q}_t = \boldsymbol{\Sigma}^{-}_{\boldsymbol{x},t}\boldsymbol{H}^{\intercal}(\boldsymbol{H}\Sigma^{-}_{\boldsymbol{x},t}\boldsymbol{H}^{\intercal} + \boldsymbol{R})^{-1}.$$

Another approach to derive the Kalman filter equations follows from a least-squares objective between the state estimator a-priori and a-posteriori to the observation (Simon 2006). This second approach does not make the explicit assumption that the belief state can be represented as a Gaussian random variable. Rather this representation follows from the objective to minimize the variance of the error between the a-priori and a-posteriori estimators. We will take this second approach as inspiration to derive the kernel Kalman rule in Sect. 4.

3 Efficient nonparametric inference in a subspace

A general drawback of kernel methods is that the complexities of the algorithms scale poorly with the number of samples in the kernel matrices. As the conditional embedding operator and the kernel inference rules require the inversion of a kernel matrix, the complexity scales cubically with the number of data points. To overcome this drawback, several approaches exist that aim to find a good trade-off between a compact representation and leveraging from a large data set. Examples are the sparse Gaussian processes that use pseudo-inputs (Snelson and Ghahramani 2006; Csató and Opper 2002), or a sparse subset of the data which is selected by maximizing the posterior probability (Smola and Bartlett 2001). Other techniques are based on approximating the kernel matrices using the Nyström method (Williams and Seeger 2000) or random Fourier features (Rahimi and Recht 2007). We approach this problem by proposing the subspace conditional embedding operators (Gebhardt et al. 2015). The basic idea is to use only a subset of the available training data as representation for the embeddings but the full data set to learn the conditional operators. In the following sections, we will recapitulate this approach and show how it can be applied to the framework for nonparamteric inference.

Given the feature matrices $\boldsymbol{\Phi} := [\phi(\boldsymbol{y}_1), \ldots, \phi(\boldsymbol{y}_n)]$ and $\boldsymbol{\Upsilon}_x := [\varphi(\boldsymbol{x}_1), \ldots, \varphi(\boldsymbol{x}_n)]$, we can define the respective subsets $\boldsymbol{\Psi} \subset \boldsymbol{\Phi}$ and $\boldsymbol{\Gamma} \subset \boldsymbol{\Upsilon}_x$, where $|\boldsymbol{\Psi}| = |\boldsymbol{\Gamma}| = m \ll n$. We assume that the subsets are representative for the embedded distributions. Similar to the conditional operator discussed in Sect. 2.1.2, we define the subspace conditional operator $\mathcal{C}^S_{Y|X}$ as the mapping from an embedding $\varphi(\boldsymbol{x}) \in \mathcal{H}_k$ to the mean embedding $\mu_{Y|\boldsymbol{x}} \in \mathcal{H}_g$ of the conditional distribution $P(Y|\boldsymbol{x})$ conditioned on the variate $\boldsymbol{x}$. To obtain this subspace conditional operator, we first introduce an auxiliary conditional operator $\mathcal{C}^{\text{aux}}_{Y|X}$ which maps from the subspace projection of the embedding $\boldsymbol{\Gamma}^{\intercal}\varphi(\boldsymbol{x})$ to the mean map of the conditional distribution, i.e.,

$$\hat{\mu}_{Y|\boldsymbol{x}} = \hat{\mathcal{C}}^{\text{aux}}_{Y|X}\boldsymbol{\Gamma}^{\intercal}\varphi(\boldsymbol{x}). \tag{23}$$

We can then derive this auxiliary conditional operator by minimizing the squared error on the full data set

$$\hat{\mathcal{C}}^{\text{aux}}_{Y|X} = \arg\min_{\mathcal{C}} \left\| \boldsymbol{\Phi} - \mathcal{C}\boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \right\|_2 \tag{24}$$

$$= \boldsymbol{\Phi}\boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \left(\boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} + \lambda \boldsymbol{I}_m \right)^{-1}, \tag{25}$$

with the identity $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$. Substituting this result for the auxiliary conditional operator in Eq. 23 gives us the subspace conditional operator as

$$\begin{aligned}\hat{\mathcal{C}}^{S}_{Y|X} &= \hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \\ &= \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}} \left(\boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}_m) \right)^{-1} \boldsymbol{\Gamma}^\intercal, \end{aligned} \tag{26}$$

where $\boldsymbol{K}_{x\bar{x}} = \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \in \mathbb{R}^{n\times m}$ is the kernel matrix of the sample feature set $\boldsymbol{\Upsilon}_x$ and the subset $\boldsymbol{\Gamma}$. Since we assume that $m \ll n$, the inverse in the subspace conditional operator is in $\mathbb{R}^{m\times m}$ and, thus, of a much smaller size than the inverse in the standard conditional operator shown in Eq. 6. Additionally, we can use the feature matrix $\boldsymbol{\Gamma}^\intercal$ on the right hand side in Eq. 26 to represent the mean embedding always in the subspace spanned by the features $\boldsymbol{\Gamma}$. This allows to avoid representations and computations in the high-dimensional space spanned by the features of the full sample set. Before we will rederive the non-parametric inference rules analogously to Song et al. (2013) but based on the subspace conditional embedding operator, we will discuss the selection of the samples for spanning the subspace and the relation of the subspace conditional operator to other sparsification approaches in the next sections.

3.1 Selecting the sample set to span the subspace

To learn the subspace conditional embedding operator, we need to choose m points for the representation of the embedding from a data set of n data points, where $m \ll n$. The selection of these data points is a crucial step as we want a subset that is descriptive enough to represent the belief state well. We propose two approaches to address this problem which aim at different characteristics of the subset.

The first approach simply samples uniformly without replacement from the full data set. The result is a subset that resembles statistically the full data set, i.e., regions that have a high density in the full data set will have a high density in the subset and vice versa.

The goal of the second approach is to get a subset with an optimal coverage of the sample space. We select the first sample randomly from the full data set into the subset. Afterwards, we iteratively extend the subset by adding samples according to the following criterion: we compute the maximum kernel activation for each sample in the full data set with the samples in the current subset, then we extend the current subset by taking the sample from the full data set with the minimal maximum activation. We call this second strategy the *kernel activation heuristic*.

3.2 Relation to other sparsification approaches

Many other sparsification techniques for kernel methods exist. The two most important techniques among these are probably the Nyström method (Williams and Seeger 2000; Drineas and Mahoney 2005) and the random Fourier features (Rahimi and Recht 2007). Both methods are closely related to our approach.

Williams and Seeger (2000) approximate the Gram matrix $\boldsymbol{K}$ based on the eigendecomposition $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$, where $\boldsymbol{U}$ are the eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal of the eigenvalues

$\lambda_1, \ldots, \lambda_n$. By taking only the first $m < n$ eigenvectors as $\boldsymbol{U}^{(m)}$ and the first m eigenvalues as diagonal $\boldsymbol{\Lambda}^{(m)}$, $\boldsymbol{K}$ can be approximated as $\boldsymbol{K} \approx \boldsymbol{U}^{(m)}\boldsymbol{\Lambda}^{(m)}(U^{(m)})^\intercal$. However, since the eigendecomposition is computationally costly and more efficient methods only significantly decrease the running time for $m \ll n$, Williams and Seeger (2000) propose to use instead the Nyström approximation of the eigenvectors which can be computed in only $O(m^2 n)$ instead of $O(n^3)$ for the true eigenvectors. The resulting approximation of the Gram matrix $\boldsymbol{K}$ has the form $\tilde{\boldsymbol{K}} = \boldsymbol{K}_{n,m}\boldsymbol{K}_{m,m}^{-1}\boldsymbol{K}_{m,n}$. Using the Nyström method to approximate the conditional embedding operator would result in the following equations

$$\hat{\mathcal{C}}_{Y|X}\hat{\mu}_X = \boldsymbol{\Phi}(\boldsymbol{K}_{n,m}\boldsymbol{K}_{m,m}^{-1}\boldsymbol{K}_{m,n} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{K}_{n,m}\boldsymbol{K}_{m,m}^{-1}\boldsymbol{K}_{m,X} \tag{27}$$

$$= \boldsymbol{\Phi}\boldsymbol{K}_{n,m}\boldsymbol{K}_{m,m}^{-1}(\boldsymbol{K}_{m,n}\boldsymbol{K}_{n,m}\boldsymbol{K}_{m,m}^{-1} + \lambda\boldsymbol{I}_m)^{-1}\boldsymbol{K}_{m,X}, \tag{28}$$

where $\boldsymbol{K}_{m,X}$ is the approximator of the kernel mean embedding μ_X using the m samples of the Nyström approximation. Thus, if we would assume $\boldsymbol{K}_{m,m} = \boldsymbol{I}$, the subspace conditional operator is equivalent to the Nyström approximation. This assumption requires that features of the selected data points are orthogonal, i.e., $\phi(\boldsymbol{x}_i)^\intercal\phi(\boldsymbol{x}_j) = \delta_{ij}$. Note that the kernel activation heuristic presented in the previous sections selects the data points by minimizing this inner product for all points that are already in the subset.

The idea of the random Fourier features (Rahimi and Recht 2007) is to compute the Fourier transform p of the kernel method k. Random samples are drawn from the distribution over frequencies p which are then used to construct a feature function $\boldsymbol{z}(\boldsymbol{x})$. Rather than using a projection of the high dimensional feater as in our approach, the random features directly transform the samples into a finite dimensional feature space whose inner product approximates the kernel function, i.e., $\boldsymbol{z}(\boldsymbol{x})^\intercal\boldsymbol{z}(\boldsymbol{y}) \approx k(\boldsymbol{x}, \boldsymbol{y})$. Let $\boldsymbol{Z} \in \mathbb{R}^{n\times m}$ be the feature matrix of the n data points with m random features. We could approximate the conditional embedding operator as

$$\hat{\mathcal{C}}_{Y|X}\hat{\mu}_X = \boldsymbol{\Phi}(\boldsymbol{Z}\boldsymbol{Z}^\intercal + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{Z}\boldsymbol{Z}^\intercal\boldsymbol{m}_X \tag{29}$$

$$= \boldsymbol{\Phi}\boldsymbol{Z}(\boldsymbol{Z}^\intercal\boldsymbol{Z} + \lambda\boldsymbol{I}_m)^{-1}\boldsymbol{Z}^\intercal\boldsymbol{m}_X. \tag{30}$$

Again, it is easy to observe the similarity to our approach if we replace $\boldsymbol{Z}$ by $\boldsymbol{K}_{x\bar{x}}$. Note that this representation, in contrast to our approach and to the Nyström method, does not allow to derive an operator in the reproducing kernel Hilbert space but directly uses finite vector/matrix representations.

3.3 The subspace kernel sum rule

Analogously to Song et al. (2013), the subspace kernel sum rule is the application of the subspace conditional operator to the embedding of a distribution $\pi(X)$, i.e.,

$$\hat{\mu}_Y^\pi = \hat{\mathcal{C}}_{Y|X}^S\hat{\mu}_X^\pi = \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\left(\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m\right)^{-1}\boldsymbol{\Gamma}^\intercal\boldsymbol{\Upsilon}_x\boldsymbol{\alpha}_\pi, \tag{31}$$

where $\mu_X^\pi = \boldsymbol{\Upsilon}_x\boldsymbol{\alpha}_\pi$ is the embedding of the prior distribution $\pi(X)$. We construct the subspace kernel sum rule for tensor product features differently than Song et al. (2013). Instead of applying the conditional operator to the mean embedding and then approximating the covariance operator with the resulting weights (c.f. Eq. 11), we first approximate the covariance operator $\mathcal{C}_{XX}^\pi$ from the weights $\boldsymbol{\alpha}_\pi$ and then apply the subspace conditional operator to both sides, i.e.,

$$\begin{aligned}\hat{\mathcal{C}}^{S,\pi}_{YY} &= \hat{\mathcal{C}}^{S}_{Y|X}\hat{\mathcal{C}}^{\pi}_{XX}\left(\hat{\mathcal{C}}^{S}_{Y|X}\right)^{\intercal} = \hat{\mathcal{C}}^{S}_{Y|X}\boldsymbol{\Upsilon}_x \operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^{\intercal}\left(\hat{\mathcal{C}}^{S}_{Y|X}\right)^{\intercal} \\ &= \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\boldsymbol{L}\operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{L}^{\intercal}\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{\Phi}^{\intercal}. \end{aligned} \tag{32}$$

Here, we denote $\boldsymbol{L} := \left(\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}_m\right)^{-1}\boldsymbol{K}_{x\bar{x}}^{\intercal} \in \mathbb{R}^{m\times n}$ to keep the notation uncluttered. This definition of the subspace kernel sum rule follows from the kernel chain rule for tensor product features, where the conditional operator $\mathcal{C}_{Y|X}$ is applied to the covariance embedding $\mathcal{C}^{\pi}_{XX}$ to obtain the covariance embedding $\mathcal{C}^{\pi}_{YX}$ (c.f. Eq. 12). The subspace kernel sum rule follows from applying the transpose of the condition operator a second time from the right-hand side.

3.4 The subspace kernel chain rule

The subspace kernel chain rule is a straight forward modification of the kernel chain rule by Song et al. (2013). We simply apply the subspace conditional operator $\mathcal{C}^{S}_{Y|X}$ from the left side to a covariance operator $\mathcal{C}^{\pi}_{XX} = \boldsymbol{\Upsilon}_x \operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^{\intercal}$ approximated from the weights $\boldsymbol{\alpha}_\pi$ of the prior mean map μ^{π}_X

$$\hat{\mathcal{C}}^{S,\pi}_{YX} = \hat{\mathcal{C}}^{S}_{Y|X}\hat{\mathcal{C}}^{\pi}_{XX} = \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\boldsymbol{L}\operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{\Upsilon}_x^{\intercal}. \tag{33}$$

With the subspace kernel sum rule and the subspace kernel chain rule we can now construct the subspace kernel Bayes' rule.

3.5 The subspace kernel Bayes' rule

The Bayes' rule computes a posterior distribution $P(X|Y)$ from a prior distribution $\pi(X)$ and a likelihood function $P(Y|X)$. Fukumizu et al. (2013) derive a conditional operator $\mathcal{C}^{\pi}_{X|Y}$ from the prior modified covariance operators $\mathcal{C}^{\pi}_{XY}$ and $\mathcal{C}^{\pi}_{YY}$. We follow this approach and construct the subspace kernel Bayes' rule (subKBR) from the prior modified covariance operators $\mathcal{C}^{S,\pi}_{XY}$ and $\mathcal{C}^{S,\pi}_{YY}$ which we obtain from the subspace kernel chain rule and the subspace kernel sum rule for tensor product features, respectively. When applied to the embedding of a variate $\boldsymbol{y}_*$, the subspace kernel Bayes' rule returns the mean embedding of the conditional distribution $P(X|\boldsymbol{y}_*)$ as

$$\hat{\mu}^{\pi}_{X|y} = \hat{\mathcal{C}}^{S,\pi}_{X|Y}\phi(\boldsymbol{y}_*) \tag{34}$$

$$\hat{\mu}^{\pi}_{X|y} = \hat{\mathcal{C}}^{S,\pi}_{XY}\left(\hat{\mathcal{C}}^{S,\pi}_{YY}\right)^{-1}\phi(\boldsymbol{y}_*). \tag{35}$$

From the subspace kernel chain rule, we obtain

$$\hat{\mathcal{C}}^{S,\pi}_{XY} = \left(\hat{\mathcal{C}}^{S}_{Y|X}\hat{\mathcal{C}}^{\pi}_{XX}\right)^{\intercal} \tag{36}$$

$$= \boldsymbol{\Upsilon}_x \operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{L}^{\intercal}\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{\Phi}^{\intercal} \tag{37}$$

and from the subspace kernel sum rule

$$\mathcal{C}^{S,\pi}_{YY} = \mathcal{C}^{S}_{Y|X}\mathcal{C}^{\pi}_{XX}\left(\mathcal{C}^{S}_{Y|X}\right)^{\intercal} \tag{38}$$

$$= \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\boldsymbol{L}\operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{L}^{\intercal}\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{\Phi}^{\intercal}. \tag{39}$$

To keep the notation of the subspace kernel Kalman rule uncluttered, we define the following matrices

$$\bar{\boldsymbol{\Lambda}} := \operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{L}^\intercal \tag{40}$$

$$\bar{\boldsymbol{D}} := \boldsymbol{L}\operatorname{diag}(\boldsymbol{\alpha}_\pi)\boldsymbol{L}^\intercal \in \mathbb{R}^{m\times m}, \tag{41}$$

$$\boldsymbol{E} := \boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}} \in \mathbb{R}^{m\times m}, \tag{42}$$

where $\boldsymbol{G}_{yy} = \boldsymbol{\Phi}^\intercal\boldsymbol{\Phi}$ is the kernel matrix of the samples $\boldsymbol{y}_i$. Using the same form of the Tikhonov regularization as the kernel Bayes' rule in Fukumizu et al. (2013) and substituting the prior modified subspace covariance operators from Eqs. 37 and 39 results in

$$\hat{\mu}_{X|y}^{\pi} = \hat{\mathcal{C}}_{XY}^{S,\pi}\left[\left(\hat{\mathcal{C}}_{YY}^{S,\pi}\right)^2 + \gamma\boldsymbol{I}_m\right]^{-1}\hat{\mathcal{C}}_{YY}^{S,\pi}\phi(\boldsymbol{y}_*) \tag{43}$$

$$= \boldsymbol{\Upsilon}_x\bar{\boldsymbol{\Lambda}}\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{\Phi}^\intercal\left[\left(\boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\bar{\boldsymbol{D}}\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{\Phi}^\intercal\right)\boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\bar{\boldsymbol{D}}\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{\Phi}^\intercal + \gamma\boldsymbol{I}_m\right]\boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\bar{\boldsymbol{D}}\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{\Phi}^\intercal\phi(\boldsymbol{y}_*) \tag{44}$$

$$= \boldsymbol{\Upsilon}_x\bar{\boldsymbol{\Lambda}}\boldsymbol{E}\left[(\bar{\boldsymbol{D}}\boldsymbol{E})^2 + \gamma\boldsymbol{I}_m\right]^{-1}\bar{\boldsymbol{D}}\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{g}_{\boldsymbol{y}_*}, \tag{45}$$

with kernel vector $\boldsymbol{g}_{\boldsymbol{y}_*} = [g(\boldsymbol{y}_1, \boldsymbol{y}_*), \ldots, g(\boldsymbol{y}_n, \boldsymbol{y}_*)]^\intercal$, and where we apply the matrix identity $\boldsymbol{A}(\boldsymbol{B}\boldsymbol{A} + \lambda\boldsymbol{I})^{-1} = (\boldsymbol{A}\boldsymbol{B} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}$ with $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}$ to obtain a finite matrix in the inverse. Since $\boldsymbol{E}$ and $\bar{\boldsymbol{D}}$ are both in $\mathbb{R}^{m\times m}$, the matrix inversion has complexity $O(m^3)$ instead of $O(n^3)$. The entire subspace kernel Bayes' rule has complexity $O(nm^2)$ and, thus, scales linearly with the number of sample points (given a fixed reference set) instead of cubically as for the original kernel Bayes' rule.

3.6 Experimental evaluation

We compare the performance, learning time and run time of the subspace kernel Bayes' filter in comparison to the standard kernel Bayes' filter on a simple toy task. We simulate a pendulum which we randomly initialize in the ranges $[0.1\pi, 0.4\pi]$ for the angle θ and $[-0.5\pi, 0.5\pi]$ for the angular velocity $\dot{\theta}$. The pendulum has a mass of 5kg and a friction coefficient of 1. We apply Gaussian white noise to the system with a variance of 1, and to the observations with a variance of 0.1. Additionally, the observed angles are randomly perturbed by an offset of $\pi/4$. These random perturbations occur with a probability of 0.1 in every time step. Each episode consists of 30 time steps with $\Delta t = 0.1$.

Figure 1 shows that the subspace KBF has a slightly better performance when the training set equals the subspace set and maintains the performance of the standard KBF with an increasing number of training samples while the subspace set is fixed to 100 samples. However, at the same time the learning time of the subspace KBF increases at a much lower magnitude and the run time is nearly constant while the learning and run time of the standard KBR grow cubically. The samples for the subspace kernel Bayes rule are drawn uniformly without replacement from the full sample set.

4 The kernel Kalman rule

All three versions of the kernel Bayes' rule discussed in Sect. 2.1.5 have drawbacks. First, due to the approximation of the prior modified covariance operators, these operators are not guaranteed to be positive definite and, thus, their inversion requires either a harsh form of

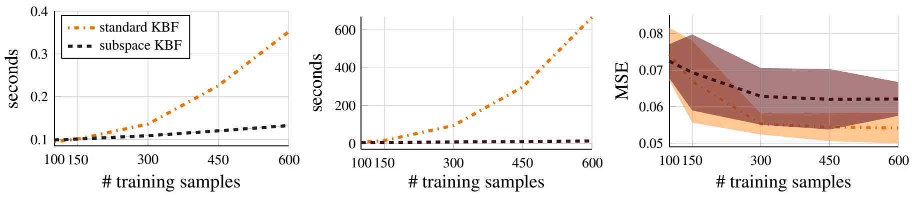


Fig. 1 The subspace variant of the kernel Bayes' filter outperforms the standard kernel Bayes' filter in both, the tarining time depicted in the left plot and the run time depicted in the middle plot, while maintaining a similar performance to the standard kernel Bayes' rule as depicted in the right plot. The size of the subspace is fixed to 100 samples. The plots show median and the [0.25, 0.75] quantiles over 20 evaluations

the Tikhonov regularization or a strong regularization factor and are often still numerically instable. Furthermore, the inverse is dependent on the embedding of the prior distribution and, hence, needs to be recomputed for every Bayesian update of the mean map. This recomputation significantly increases the computational costs, for example, if we want to optimize the hyper-parameter.

In this section we will present the *kernel Kalman rule* (KKR) as an approximate alternative to the kernel Bayes' rule (Fukumizu et al. 2013). We assume a prior belief state over a variable X embedded in a Hilbert space $\mathcal{H}_k$ as

$$\mu_{X,t}^{-} = \mathbb{E}_{X_t|\boldsymbol{y}_{1:t-1}}[\varphi(X)]$$

and new measurement $\boldsymbol{y}_t$ embedded in a Hilbert space $\mathcal{H}_g$ as $\phi(\boldsymbol{y}_t)$. With the kernel Kalman rule we want to infer the embedding of the posterior belief state

$$\mu_{X,t}^{+} = \mathbb{E}_{X_t|\boldsymbol{y}_{1:t}}[\varphi(X)] \in \mathcal{H}_k$$

from the prior belief and the new measurement. The derivations for the KKR are inspired by the ansatz from recursive least squares (Gauss 1823; Sorenson 1970; Simon 2006), and thus the resulting update equations follow from a clear optimality criterion.

4.1 Estimating the posterior mean embedding from a least squares objective

Let $\mathcal{C}_{Y|X}$ be a conditional embedding operator of the observation model $P(Y|X)$ that yields for a given belief state embedded in the Hilbert space $\mathcal{H}_k$ the distribution over possible observations embedded into the Hilbert space $\mathcal{H}_g$. We call this conditional embedding operator also *observation operator*. For a single sample $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, the observation operator yields the relation

$$\phi(\boldsymbol{y}_t) = \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t) + \zeta_t, \tag{46}$$

where ζ_t is zero mean noise with covariance $\mathcal{R}$. Let us assume that the distribution $p(\boldsymbol{x})$ is unknown and we can only observe the samples $\boldsymbol{y}_t$. The objective of the KKR is then to find the mean embedding μ_X that minimizes the squared error

$$L = \mathbb{E}_{XY}\left[\left(\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu_X\right)^{\intercal} \mathcal{R}^{-1} \left(\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu_X\right)\right]. \tag{47}$$

Note that the use of $\mathcal{R}^{-1}$ as metric for the least squares is somewhat arbitrary (it works with any invertible matrix). This definition becomes more important once we regularize the estimate of μ_X (see below). We assume that $\mathcal{R}$ is constant for all samples $\boldsymbol{x}_t$.[1] We do *not* assume that the noise is constant if we use a mean embedding on the operator $\mathcal{C}_{Y|X}$.

To show that $\mu_X = \mathbb{E}_X[\varphi(\boldsymbol{x})]$, i.e., that μ_X is indeed the mean embedding of the distribution $p(\boldsymbol{x})$, we can solve for μ_X by setting the derivative of L to zero, i.e.,

$$\frac{\mathrm{d}L}{\mathrm{d}\mu_X} = \mathbb{E}_{XY}\left[(\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu_X)^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X}\right] \tag{48}$$

$$= \mathbb{E}_Y\left[\phi(\boldsymbol{y}_t)^\intercal\right]\mathcal{R}^{-1}\mathcal{C}_{Y|X} - \mu_X^\intercal \mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X} \tag{49}$$

$$= 0. \tag{50}$$

Assuming that $\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X}$ is invertible, this yields

$$\mu_X = (\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X})^{-1}\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathbb{E}_Y\left[\phi(\boldsymbol{y}_t)\right] \tag{51}$$

$$= (\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X})^{-1}\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X}\mathbb{E}_X\left[\varphi(\boldsymbol{x}_t)\right] = \mathbb{E}_X\left[\varphi(\boldsymbol{x}_t)\right], \tag{52}$$

where we have used the kernel sum rule $\mathbb{E}_Y[\phi(\boldsymbol{y}_t)] = \mathcal{C}_{Y|X}\mathbb{E}_X[\varphi(\boldsymbol{x}_t)]$. Hence, under the assumption that $\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X}$ is invertible, μ_X indeed estimates the mean embedding of the unobserved distribution $p(\boldsymbol{x})$. Note that the derivations hold for a constant $\boldsymbol{x}$, i.e, $\boldsymbol{x}_t = \boldsymbol{x}$ as well as for samples $\boldsymbol{x}_t$ drawn from the distribution $p(\boldsymbol{x})$.

In practice, inverting $\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X}$ is, however, not always feasible. Hence, we can introduce an additional regularization objective, i.e.,

$$L_{\text{reg}} = L + (\mu_X - \mu_X^-)^\intercal (\mathcal{C}_{XX}^-)^{-1}(\mu_X - \mu_X^-), \tag{53}$$

where μ_X^- and $\mathcal{C}_{XX}^-$ denote a prior belief embedded as mean embedding and covariance operator, respectively. The solution of this optimization problem is given by

$$\mu_X = \left(\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathcal{C}_{Y|X} + (\mathcal{C}_{XX}^-)^{-1}\right)^{-1}\left(\mathcal{C}_{Y|X}^\intercal \mathcal{R}^{-1}\mathbb{E}_Y\left[\phi(\boldsymbol{y}_t)\right] + (\mathcal{C}_{XX}^-)^{-1}\mu_X^-\right). \tag{54}$$

Note that this regularization is the only approximation we make in the derivation of the kernel Kalman rule.

4.2 Using recursive least squares to estimate the posterior embedding

Since we want to update our estimate μ_X iteratively with each new observation $\boldsymbol{y}_t$, we assume a prior mean map $\mu_{X,t}^-$. In each iteration, we update the prior mean map with the measurement $\boldsymbol{y}_t$ to obtain the posterior mean map $\mu_{X,t}^+$. From the recursive least squares solution, we know that the update rule for obtaining the posterior mean map $\mu_{X,t}^+$ is

$$\mu_{X,t}^+ = \mu_{X,t}^- + \mathcal{Q}_t\left(\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu_{X,t}^-\right), \tag{55}$$

where $\mathcal{Q}_t$ is the Hilbert space *Kalman gain operator* that is applied to the correction term $\delta_t = \phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu_{X,t}^-$. We call this rule the *kernel Kalman rule* (KKR). It remains to find an optimal value for the Kalman gain operator. In the next section, we will show that this rule is an unbiased estimator of the posterior mean map. Thus, we cannot obtain the optimal $\mathcal{Q}_t$ by minimizing directly the error. Instead, we will show in Sect. 4.2.2, how we can find an optimal $\mathcal{Q}_t$ by minimizing the covariance of the error instead.

[1] This assumption can be relaxed to $\mathcal{R}$ being constant for a single sample $\boldsymbol{x}$ for the derivations on this page.

4.2.1 The kernel Kalman update is an unbiased estimator of the posterior mean map

The embedding of the observation $\phi(\boldsymbol{y}_t)$ in the correction term δ_t is a single-sample estimator of the embedding of the true distribution over observations $\mu^-_{Y|x,t} = \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t)$. Let us assume for now that we have access to the true embedding $\varphi(\boldsymbol{x}_t)$. Thus, we have for the embedding of the observation

$$\phi(\boldsymbol{y}_t) = \mu^-_{Y|x,t} + \zeta_t = \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t) + \zeta_t, \tag{56}$$

where ζ_t denotes the error of the single sample estimator $\phi(\boldsymbol{y}_t)$ to the embedding of the true distribution $\mu^-_{Y|x,t}$. By taking the expectation it is easy to show that the error of the single-sample estimator is zero-mean and thus $\phi(\boldsymbol{y}_t)$ is an unbiased estimator for $\mu^-_{Y|X,t}$. We refer to "Appendix B.1" for a more detailed derivation. We further assume that ζ_t is independent from the state x_t and has constant covariance $\mathcal{R}$. Following from the delta method (Agresti 2002), this assumption is a reasonable choice as we assume i.i.d, zero mean observation noise. Similarly, the error of the a-posteriori mean embedding to the embedding of the true state is given as

$$\begin{aligned}\varepsilon_t^+ &= \varphi(\boldsymbol{x}_t) - \mu^+_{X,t} &(57)\\ &= \varphi(\boldsymbol{x}_t) - \mu^-_{X,t} - \mathcal{Q}_t(\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\mu^-_{X,t}), &(58)\end{aligned}$$

where we use Eq. 55 to substitute the embedding of the posterior belief. By substituting $\phi(\boldsymbol{y}_t)$ with Eq. 56 and defining the error of the a-priori mean embedding analogously as $\varepsilon_t^- = \varphi(\boldsymbol{x}_t) - \mu^-_{X,t}$, we arrive at

$$\begin{aligned}\varepsilon_t^+ &= \varphi(\boldsymbol{x}_t) - \mu^-_{X,t} - \mathcal{Q}_t(\mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t) + \zeta_t - \mathcal{C}_{Y|X}\mu^-_{X,t}) &(59)\\ &= \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)(\varphi(\boldsymbol{x}_t) - \mu^-_{X,t}) - \mathcal{Q}_t\zeta_t &(60)\\ &= \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\varepsilon_t^- - \mathcal{Q}_t\zeta_t &(61)\end{aligned}$$

with identity operator $\mathcal{I}$. We can now apply the expectation operator and exploit its linearity to obtain

$$\begin{aligned}\mathbb{E}\left[\varepsilon_t^+\right] &= \mathbb{E}\left[\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\varepsilon_t^- - \mathcal{Q}_t\zeta_t\right]\\ &= \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\mathbb{E}\left[\varepsilon_t^-\right] - \mathcal{Q}_t\mathbb{E}\left[\zeta_t\right]. &(62)\end{aligned}$$

Since the residual of the observation operator is zero mean ($\mathbb{E}[\zeta_t] = 0$), we see that, given an unbiased a-priori mean embedding ($\mathbb{E}[\varepsilon_t^-] = 0$), the a-posteriori mean embedding obtained from the kernel Kalman update is unbiased ($\mathbb{E}[\varepsilon_t^+] = 0$) independent of the choice of $\mathcal{Q}$. Thus, we cannot use the expected error as an optimality criterion for the Kalman gain operator.

4.2.2 Finding the optimal kernel Kalman gain operator

If the expected error—or the first moment of the error distribution—is already zero, taking the covariance of the error—or the second moment—is a consequent choice. Hence, we chose the kernel Kalman gain operator $\mathcal{Q}_t$ which minimizes the expected squared loss $\mathbb{E}\left[\left(\varepsilon_t^+\right)^\intercal \varepsilon_t^+\right]$ or equivalently the variance of the estimator. The objective for minimizing the variance can also be reformulated as minimizing the trace of the a-posteriori covariance operator $\mathcal{C}^+_{XX,t}$ of the state $\boldsymbol{x}_t$ at time t, i.e., $\min_{\mathcal{Q}_t} \mathbb{E}\left[\left(\varepsilon_t^+\right)^\intercal \varepsilon_t^+\right] = \min_{\mathcal{Q}_t} \operatorname{Tr} \mathcal{C}^+_{XX,t}$. Using the formulation of the posterior error from Eq. 61 and the independence assumption of ζ_t and ε_t allows us to reformulate the a-posteriori covariance operator as

$$\mathcal{C}^{+}_{XX,t} = \left(\mathcal{I} - \mathcal{Q}_t \mathcal{C}_{Y|X}\right) \mathcal{C}^{-}_{XX,t} \left(\mathcal{I} - \mathcal{Q}_t \mathcal{C}_{Y|X}\right)^{\intercal} + \mathcal{Q}_t \mathcal{R} \mathcal{Q}_t^{\intercal}, \tag{63}$$

where $\mathcal{R} = \mathbb{E}[\zeta_t \zeta_t^{\intercal}]$ is the covariance of the residual of the observation operator. Taking the derivative of the trace of the covariance operator and setting it to zero leads to the solution for the kernel Kalman gain operator

$$\mathcal{Q}_t = \mathcal{C}^{-}_{XX,t} \mathcal{C}^{\intercal}_{Y|X} \left(\mathcal{C}_{Y|X} \mathcal{C}^{-}_{XX,t} \mathcal{C}^{\intercal}_{Y|X} + \mathcal{R}\right)^{-1}. \tag{64}$$

We provide a detailed derivation of the optimal Kalman gain operator in “Appendix B.2”. From Eqs. 63 and 64, we can also see that it is possible to recursively estimate the covariance embedding operator independently of the mean map and of the observations. This property will allow us later to precompute the covariance embedding operator as well as the kernel Kalman gain operator to further improve the computational complexity of our algorithm. Following Simon (2006), the update of the covariance operator can be further simplified to

$$\mathcal{C}^{+}_{XX,t} = \mathcal{C}^{-}_{XX,t} - \mathcal{Q}_t \mathcal{C}_{Y|X} \mathcal{C}^{-}_{XX,t}. \tag{65}$$

The derivations of this simplification can be found in “Appendix B.3”. In the following section we will show how to obtain the empirical Kalman update rule from a finite data set.

4.3 Empirical kernel Kalman rule

The equations for the kernel Kalman rule that we derived in the previous section are based on embeddings in infinite dimensional Hilbert spaces and operators that map between these spaces. In practice, these embeddings and operators are estimated from a finite set of samples $\mathcal{D}_{XY} = \left\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\right\}$. In this section we will show how we can reformulate the kernel Kalman rule to manipulations of finite matrices by applying the kernel trick (i.e., matrix identities). Based on the data set $\mathcal{D}_{XY}$ and the corresponding feature matrices $\boldsymbol{\Upsilon}_x$ and $\boldsymbol{\Phi}$, the finite sample estimators of the prior mean embedding and the prior covariance operator are given as

$$\hat{\mu}^{-}_{X,t} = \boldsymbol{\Upsilon}_x \boldsymbol{m}^{-}_t \quad \text{and} \quad \hat{\mathcal{C}}^{-}_{XX,t} = \boldsymbol{\Upsilon}_x \boldsymbol{S}^{-}_t \boldsymbol{\Upsilon}^{\intercal}_x, \tag{66}$$

respectively, with weight vector $\boldsymbol{m}^{-}_t$ and positive definite weight matrix $\boldsymbol{S}^{-}_t$. Using this finite sample estimator of the covariance operator, the finite sample estimator of the conditional operator from Eq. 6, and by approximating the covariance of the residual of the observation operator with a diagonal $\mathcal{R} = \kappa\mathcal{I}$, we can rewrite the kernel Kalman gain operator as

$$\hat{\mathcal{Q}}_t = \boldsymbol{\Upsilon}_x \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal} \boldsymbol{\Phi}^{\intercal} \left(\boldsymbol{\Phi} \boldsymbol{O} \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal} \boldsymbol{\Phi}^{\intercal} + \kappa\mathcal{I}\right)^{-1}, \tag{67}$$

with the observation matrix $\boldsymbol{O} = (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I})^{-1} \boldsymbol{K}_{xx}$. Here, the approximation of $\mathcal{R} = \kappa\mathcal{I}$ also acts as a small regularization in the inverse to ensure its positive definiteness and to improve the numerical stability of the kernel Kalman rule. However, $\hat{\mathcal{Q}}_t$ still requires the inversion of an infinite dimensional matrix. Using matrix identities, we can solve this problem and arrive at

$$\hat{\mathcal{Q}}_t = \boldsymbol{\Upsilon}_x \underbrace{\boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal} (\boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal} + \kappa \boldsymbol{I}_n)^{-1}}_{\boldsymbol{Q}_t} \boldsymbol{\Phi}^{\intercal}, \tag{68}$$

where we defined $\boldsymbol{Q}_t = \boldsymbol{S}_t^- \boldsymbol{O}^\intercal (\boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^- \boldsymbol{O}^\intercal + \kappa \boldsymbol{I})^{-1} \in \mathbb{R}^{n \times n}$ where $\boldsymbol{G}_{yy} = \boldsymbol{\Phi}^\intercal \boldsymbol{\Phi}$ is the Gram matrix of the observations. Based on this reformulation of the kernel Kalman gain operator, we can obtain finite vector/matrix representations of the update equations for the estimator of the mean embedding (Eq. 55) and the estimator of the covariance operator (Eq. 65). For the weight vector $\boldsymbol{m}_t$, we arrive at

$$\boldsymbol{m}_t^+ = \boldsymbol{m}_t^- + \boldsymbol{Q}_t \left(\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{m}_t^- \right), \tag{69}$$

where $\boldsymbol{g}_{\boldsymbol{y}_t} = [g(\boldsymbol{y}_1, \boldsymbol{y}_t), \ldots, g(\boldsymbol{y}_n, \boldsymbol{y}_t)]^\intercal$ is the kernel vector of the measurement at time t. Similarly, we can also obtain the update equation for the weight matrix $\boldsymbol{S}_t$ as

$$\boldsymbol{S}_t = \boldsymbol{S}_t^- - \boldsymbol{Q}_t \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^-. \tag{70}$$

The algorithm requires the inversion of a $m \times m$ matrix in every iteration for computing the kernel Kalman gain matrix $\boldsymbol{Q}_t$. Hence, similar to the kernel Bayes' rule, the computational complexity of a straightforward implementation would scale cubically with the number of data points m. However, in contrast to the KBR, the inverse in $\boldsymbol{Q}_t$ is only dependent on time and not on the estimate of the mean map. Thus, the kernel Kalman gain matrix can be precomputed since it is identical for multiple parallel runs of the algorithm. Furthermore, if the stream of incoming measurements is reliable (no time steps without incoming measurement), $\boldsymbol{S}_t$ will converge to a stationary matrix and by that $\boldsymbol{Q}_t$ will become stationary as well. While many applications do not require to perform state estimations in parallel, it is a huge advantage for hyper-parameter optimization as we can evaluate multiple trajectories from a validation set simultaneously. As for most kernel-based methods, hyper-parameter optimization is crucial for scaling the approach to complex tasks. So far, the hyper-parameters of the kernels for the KBF have typically been set by heuristics as optimization would be too expensive.

Besides the hyper-parameters in the kernel functions (e.g. bandwidths) and the regularization constants of the conditional operators, we also treat the approximation of the covariance operator $\mathcal{R} \approx \kappa \mathcal{I}$ as a hyper-parameter which we optimize. Note that the selection of the minimization objective, e.g., mean squared error (MSE) or negative log-likelihood (NLL), has a substantial effect on the selection of this parameter. For the MSE objective, the parameters are chosen to only optimize the expectation of the filter output. Consequently the parameter κ acts more as a regularizer and is chosen as small as possible. In contrast, using the NLL objective also respects the variance of the filter output and thus the role as approximation to the variance $\mathcal{R}$ is more important for choosing the parameter value.

4.4 The subspace kernel Kalman rule

In Sect. 3 we have already shown how we can apply the subspace conditional embedding operator to leverage from large data sets but at the same time maintain the computational tractability of the learned models. In this section, we will now show how we can apply this technique to the kernel Kalman rule to obtain the *subspace kernel Kalman rule* (subKKR).

A core difference between the KKR and the subKKR is the representation of the embedded distributions. While we represent the embeddings for the kernel Kalman rule as weight vector $\boldsymbol{m}_t^-$ and weight matrix $\boldsymbol{S}_t^-$, we use the projections into a subspace

$$\boldsymbol{n}_t = \boldsymbol{\Gamma}^\intercal \mu_t = \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{m}_t = \boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{m}_t, \tag{71}$$

$$\boldsymbol{P}_t = \boldsymbol{\Gamma}^\intercal \mathcal{C}_{XX,t} \boldsymbol{\Gamma} = \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{S}_t \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} = \boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{S}_t \boldsymbol{K}_{x\bar{x}}. \tag{72}$$

to represent the distribution for the subspace kernel Kalman rule. These projections will later allow us to express all operations in the lower dimensional subspace instead of the space spanned by the full data set. Matrix manipulations with the full data set are then only necessary during the learning phase of the KKR not while performing inference.

We use a slightly modified version of the kernel Kalman gain from Eq. 64 where we approximate the covariance operator $\mathcal{R}$ with a diagonal operator $\kappa\mathcal{I}$. With the subspace conditional embedding operator $\hat{\mathcal{C}}^S_{Y|X}$ of the distribution $P(Y|X)$, as derived in Sect. 2.1.2 we obtain the subspace kernel Kalman gain operator as

$$\hat{\mathcal{Q}}^S_t = \hat{\mathcal{C}}^-_{XX,t}\left(\hat{\mathcal{C}}^S_{Y|X}\right)^\intercal\left(\hat{\mathcal{C}}^S_{Y|X}\hat{\mathcal{C}}^-_{XX,t}\left(\hat{\mathcal{C}}^S_{Y|X}\right)^\intercal + \kappa\mathcal{I}\right)^{-1} \tag{73}$$

We can further derive a finite matrix representation of the operator using matrix identities and the projection into the subspace spanned by the features $\boldsymbol{\Gamma}^\intercal$ as

$$\boldsymbol{\Gamma}^\intercal\hat{\mathcal{Q}}^S_t = \underbrace{\boldsymbol{P}^-_t(\boldsymbol{O}^S)^\intercal\left(\boldsymbol{K}^\intercal_{x\bar{x}}\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}^-_t(\boldsymbol{O}^S)^\intercal + \kappa\boldsymbol{I}_m\right)^{-1}\boldsymbol{K}^\intercal_{x\bar{x}}}_{\boldsymbol{Q}^S_t}\boldsymbol{\Phi}^\intercal, \tag{74}$$

where we define the subspace kernel Kalman gain matrix $\boldsymbol{Q}^S_t$ using the short hand $\boldsymbol{O}^S := (\boldsymbol{K}^\intercal_{x\bar{x}}\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m)^{-1}$. A detailed derivation can be found in "Appendix B.5". Note that $\boldsymbol{Q}^S_t \in \mathbb{R}^{m\times n}$ and not $\mathbb{R}^{m\times m}$, however when applying the subspace KKR in an inference algorithm, we can use the matrix $\boldsymbol{K}^\intercal_{x\bar{x}}$ on the right side as a projection for the high-dimensional embedding of the distribution over the variable Y to which the gain is applied (see Algorithm 2 for an example). From here, the update equation for the projection of the mean map becomes

$$\boldsymbol{n}^+_{X,t} = \boldsymbol{n}^-_{X,t} + \boldsymbol{Q}^S_t\left(\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{n}^-_t\right), \tag{75}$$

And similarly, we can derive the update equation for the covariance embedding as

$$\boldsymbol{P}^+_t = \boldsymbol{P}^-_t - \boldsymbol{Q}^S_t\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}^-_t \tag{76}$$

In contrast to the kernel Kalman gain presented in the previous section, but also in contrast to the variants of the kernel Bayes rule discussed in Sect. 2.1.5, the subspace kernel Kalman gain requires only the inversion of an $m \times m$ matrix instead of an $n \times n$ matrix, where $m \ll n$. Still, the full data set of n samples can be used to learn the Kalman gain operator.

4.5 Experimental comparison of (sub)KKR and (sub)KBR

We compare the performance of the (subspace) kernel Kalman rule to the performance of the (subspace) kernel Bayes rule on a simple stationary filtering task for estimating the expectation of a Gaussian distribution. The graphical model that we assume for this task is depicted in Fig. 2. We sample $N = 500$ latent context variables c_i uniformly from the interval $[-5, 5]$ as the mean of the Gaussian distributions. Afterwards we draw one single ($M = 1$) observed sample s_i for each context from $\mathcal{N}(c_i, \frac{1}{3})$ and learn the kernel Kalman rule and the different versions of the kernel Bayes rule with the context variables as states and the samples as observations. For the performance comparison (Fig. 3), the KKR and the KBR are learned with a kernel size of 200 samples, subKKR and subKBR are both learned with 200 samples to span the subspace and the full set of 500 samples to learn the operators. The comparison of time efficiency is summarized in Table 1 where the respective kernel size and subspace size is denoted as column header. The subKKR and subKBR have always been learned with the full data set of 500 samples. The data points for the subspace have been drawn uniformly without replacement from the full data set.

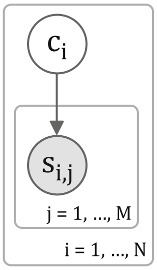


Fig. 2 Graphical model for comparing KKR to KBR

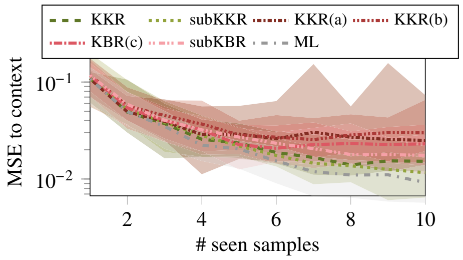


Fig. 3 Performance of the KKR updates versus the KBR updates for estimating the mean of a Gaussian distribution with 1–10 seen samples. The ML estimate serves as a baseline. Depicted are the median and the (0.15, 0.85)-quantiles of the MSE to the true mean over 20 runs

For the optimization of the hyper-parameters and for the evaluation, we have respectively generated a data set with $N = 10$ latent context variables from the same uniform distribution. These context variables are not be observed by the the filter methods. Next, we draw $M = 10$ samples from the Gaussian distribution around each context and update each method iteratively with these ten samples. For each update we compute the squared error to the true context and take the mean over all ten context variables. We use squared exponentials as kernel functions and optimize their bandwidths as well as the regularization parameters using CMA-ES (Hansen 2006). Figure 3 shows the median and the (0.15, 0.85)-quantiles of the MSE to the true context over the number of seen samples. As a baseline we depict the maximum-likelihood (ML) estimate of the expectation. We see that while in the beginning all methods perform similar to the ML estimate, with more seen samples KKR and subKKR outperform all variants of the KBR. Moreover the choice to depict the median and (0.15, 0.85)-quantiles over mean and standard deviation is due to the instable optimization behavior of the KBR which produced a lot of outliers. In Table 1 we state the time consumed to perform ten KKR/KBR updates on 10 estimation tasks for different kernel sizes. Here, the KKR/subKKR methods benefit from their ability to process the updates for all 10 estimation tasks in parallel. Yet, the ability to precompute $\boldsymbol{Q}_t/\boldsymbol{Q}_t^S$ and $\boldsymbol{S}_t/\boldsymbol{P}_t$ has not even been exploited.

In a second experiment, we have investigated how sensible the KKR is to non-constant noise in comparison to the KBR. We have sampled data similar to the previous experiment with a context variable c_i in the range $[-5, 5]$ and observations $s_{i,j}$ from the distribution $\mathcal{N}(c_i, \sigma(c_i))$. The variance of the Gaussian distribution is dependent on the context variable by $\sigma(c_i) = \exp(c_i)$. Again, we sample $N = 500$ context and one observation ($M = 1$) for each context for learning the models. For the optimization of the hyper-parameters, we have sampled a data set of $N = 10$ context variables with $M = 10$ observations for each context. And for the evaluation of the methods, we have chosen the context variables at the integers $[-5, -4, \ldots, 5]$ and have sampled again $M = 10$ observations per context. For each sampled context, we perform updates with all ten observations. The plots in Fig. 4 show mean and min/max of the estimated mean relative to the true mean (context) from which the observations have been sampled for both cases, constant noise $\sigma = \frac{1}{3}$ and variable noise

Table 1 Time consumptions of the KKR and KBR update methods for different kernel sizes

#	200	300	400	500
KKR	0.1110	0.3360	0.7155	1.3965
subKKR	0.1820	0.4655	0.9130	1.6075
KBR(a)	2.9395	9.2395	21.5035	41.6955
KBR(b)	0.8695	2.5840	5.7580	10.9900
KBR(c)	0.5455	1.5575	3.3695	6.4465
subKBR	2.3530	4.7625	7.6540	12.7960

The update was performed on 10 samples from 10 different distributions. The subspace KKR and KBR updates are trained with 500 samples in the full data set. Both KKR methods outperform the KBR methods clearly as they are able to process the update on the 10 different distributions in parallel

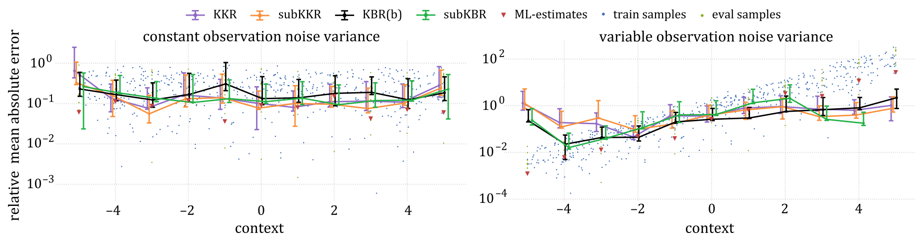


Fig. 4 Comparison of the performances of the KKR, subKKR, KBR(b), and subKBR on the estimation of the mean c_i of a Gaussian random variable, left with constant variance and right with variable variance $\exp(c_i)$. The y-axis depicts the mean absolute error relative to the context using a log-scale. Because of the exponential relation between the observation noise and the context variable, we get the linear slope in the distribution of the samples in the right plot (Color figure online)

$\sigma = \exp(c_i)$. As expected from the previous experiment, it can be clearly seen that all models perform similarly well for the case of a constant noise variance. In the case of variable noise variance, all models perform worse for smaller variances where the impact is larger in the performances of the KKR and the subKKR. Note, however, that the KBR methods suffered from numerical instabilities for large noise variances. For instance, KBR(b) was the only KBR method that yielded results for the largest variance $\sigma = \exp(5)$.

5 Applications of the kernel Kalman rule

In Sect. 4, we have shown how we can derive the KKR as an operator for approximate Bayesian updates in the framework for nonparametric inference. In this section we will present two applications of the kernel Kalman rule. In Sect. 5.1 we will first present the *kernel Kalman filter* (KKF) and discuss details about the implementation. A subspace variate of the KKF is presented in Sect. 5.2 and experimental results of both are shown in Sect. 5.3. The *kernel forward backward smoother* (KFBS) is presented as another application of the KKR in Sect. 5.4 and a subspace variate is discussed in Sect. 5.5. We finally show experimental evaluations of the KFBS and the subKFBS in Sect. 5.6.

5.1 The kernel Kalman filter

Similar to the kernel Bayes' filter (Fukumizu et al. 2013; Song et al. 2013), we can combine the kernel Kalman rule with the kernel sum rule to formulate the *kernel Kalman filter* (KKF). To learn the models of the KKF, we assume a data set $\mathcal{D}_{X\tilde{X}Y} = \left\{(\tilde{\boldsymbol{x}}_1, \boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\tilde{\boldsymbol{x}}_n, \boldsymbol{x}_n, \boldsymbol{y}_n)\right\}$ consisting of triples with preceding state $\tilde{x}_i$, state x_i, and measurement y_i as given. We further assume the states to be Markov, i.e., the state x_i is only dependent on its predecessor $\tilde{x}_i$. Based on this data set we define the feature matrices $\boldsymbol{\Upsilon}_x := [\varphi(\boldsymbol{x}_1), \ldots, \varphi(\boldsymbol{x}_n)]$, $\boldsymbol{\Upsilon}_{\tilde{x}} := [\varphi(\tilde{\boldsymbol{x}}_1), \ldots, \varphi(\tilde{\boldsymbol{x}}_n)]$, and $\boldsymbol{\Phi} := [\phi(\boldsymbol{y}_1), \ldots, \phi(\boldsymbol{y}_n)]$. In contrast to the KBF, we represent the belief state as mean map $\hat{\mu}_{X,t} = \boldsymbol{\Upsilon}_x \boldsymbol{m}_t$ and as covariance operator $\hat{\mathcal{C}}_{XX,t} = \boldsymbol{\Upsilon}_x \boldsymbol{S}_t \boldsymbol{\Upsilon}_x^\intercal$.

The forward model $P(X|\tilde{X})$ that propagates the posterior belief state at time t to the prior belief state at time $t+1$ can then be learned as conditional embedding operator

$$\hat{\mathcal{C}}_{X|\tilde{X}} = \boldsymbol{\Upsilon}_x \left(\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_n\right)^{-1} \boldsymbol{\Upsilon}_{\tilde{x}}^\intercal, \tag{77}$$

which we also call *transition operator*. Here, $\boldsymbol{K}_{\tilde{x}\tilde{x}}$ is the Gram matrix of the features of the preceding states $\boldsymbol{\Upsilon}_{\tilde{x}}$. The posterior belief state at time t is then propagated to the prior belief state at time $t+1$ time by applying the kernel sum rule. That is, we apply the transition operator to the posterior mean map and the posterior covariance embedding at time t and obtain prior mean map and prior covariance embedding at time $t+1$, i.e.,

$$\hat{\mu}^-_{X,t+1} = \hat{\mathcal{C}}_{X|\tilde{X}} \hat{\mu}^+_{X,t} = \boldsymbol{\Upsilon}_x \boldsymbol{T} \boldsymbol{m}_t^+, \qquad \Leftrightarrow \qquad \boldsymbol{m}^-_{t+1} = \boldsymbol{T}\boldsymbol{m}_t^+ \tag{78}$$

$$\hat{\mathcal{C}}^-_{XX,t+1} = \hat{\mathcal{C}}_{X|\tilde{X}} \hat{\mathcal{C}}^+_{XX,t} \hat{\mathcal{C}}^\intercal_{X|\tilde{X}} + \mathcal{V} \qquad \Leftrightarrow \qquad \boldsymbol{S}^-_{t+1} = \boldsymbol{T}\boldsymbol{S}_t^+ \boldsymbol{T}^\intercal + \boldsymbol{V}. \tag{79}$$

Note that the propagation of the covariance embedding is slightly different to the kernel sum rule by Song et al. (2013), however this formulation follows directly from the kernel chain rule (c.f. Eqs. 32 and 11). Analog to the observation matrix $\boldsymbol{O}$ (c.f. Sect. 4.3), we denote the transition matrix $\boldsymbol{T} = (\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda_{\boldsymbol{T}} \boldsymbol{I})^{-1} \boldsymbol{K}_{\tilde{x}x}$, where $\boldsymbol{K}_{\tilde{x}x} = \boldsymbol{\Upsilon}_{\tilde{x}}^\intercal \boldsymbol{\Upsilon}_x$ is the kernel matrix of the preceding states and the current states. The covariance of the transition residual $\mathcal{V}$ and its finite matrix representation $\boldsymbol{V}$ can be obtained as

$$\begin{aligned}
\mathcal{V} &= \frac{1}{m}\left(\hat{\mathcal{C}}_{X|\tilde{X}} \boldsymbol{\Upsilon}_{\tilde{x}} - \boldsymbol{\Upsilon}_x\right)\left(\hat{\mathcal{C}}_{X|\tilde{X}} \boldsymbol{\Upsilon}_{\tilde{x}} - \boldsymbol{\Upsilon}_x\right)^\intercal && (80)\\
&= \frac{1}{m}\left(\boldsymbol{\Upsilon}_x \left(\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_n\right)^{-1} \boldsymbol{\Upsilon}_{\tilde{x}}^\intercal \boldsymbol{\Upsilon}_{\tilde{x}} - \boldsymbol{\Upsilon}_x\right)\left(\boldsymbol{\Upsilon}_x \left(\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_n\right)^{-1} \boldsymbol{\Upsilon}_{\tilde{x}}^\intercal \boldsymbol{\Upsilon}_{\tilde{x}} - \boldsymbol{\Upsilon}_x\right)^\intercal && (81)\\
&= \boldsymbol{\Upsilon}_x \underbrace{\frac{1}{m}\left(\left(\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_n\right)^{-1} \boldsymbol{K}_{\tilde{x}\tilde{x}} - \boldsymbol{I}_n\right)\left(\left(\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_n\right)^{-1} \boldsymbol{K}_{\tilde{x}\tilde{x}} - \boldsymbol{I}_n\right)^\intercal}_{\boldsymbol{V}} \boldsymbol{\Upsilon}_x^\intercal. && (82)
\end{aligned}$$

On the new prior belief state that we obtain from the transition update, we can afterwards apply the kernel Kalman rule as observation update. Before we give a condensed summary of the kernel Kalman filter in Algorithm 1, we will discuss how we obtain the embedding of the distribution over the initial states in the next section. To extract some meaningful information from the RKHS-embedded distributions, we furthermore need to find a mapping of the embedded distribution back into the state space. In Sect. 5.1.2, show how we approached the so-called *preimage problem* and shortly discuss other solutions.

5.1.1 Embedding the initial state distribution

Before running the filter on incoming measurements y_t, we need to initialize the belief state with an initial mean map $\mu_{X,0}$ and an initial covariance operator $\mathcal{C}_{XX,0}$. We can obtain these initial embeddings from a data set $\mathcal{D}_0 = \{\boldsymbol{x}_1^0, \ldots, \boldsymbol{x}_N^0\}$ which consists in general of samples from the initial distribution of the system. Practically, we can obtain this data set by taking the initial states from multiple training episodes or—if we assume a stationary distribution—we can also take all training samples for the initialization. We can obtain the initial mean map by first embedding a uniform distribution into the RKHS spanned by the features of the initial states $\boldsymbol{\Upsilon}_{x,0}$. Afterwards, we apply a conditional operator to map this distribution into the Hilbert space spanned by the features $\boldsymbol{\Upsilon}_x$ as

$$\hat{\mu}_{X,0} = \boldsymbol{\Upsilon}_x \boldsymbol{m}_0 = \boldsymbol{\Upsilon}_x (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Upsilon}_{x,0} \mathbf{1}_N \frac{1}{N} \tag{83}$$

$$\Leftrightarrow \quad \boldsymbol{m}_0 = (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{K}_{x0} \mathbf{1}_N \frac{1}{N}. \tag{84}$$

where $\boldsymbol{K}_{x0} = \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Upsilon}_{x,0}$ is the kernel matrix of the training samples and the samples in $\mathcal{D}_0$, and $\mathbf{1}_N$ denotes the N-dimensional all-ones vector. Similarly, we can obtain the initial covariance embedding operator as

$$\hat{\mathcal{C}}_{XX,0} = \boldsymbol{\Upsilon}_x \boldsymbol{S}_0 \boldsymbol{\Upsilon}_x^\intercal = \frac{1}{N} \boldsymbol{\Upsilon}_x (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{K}_{x0} \boldsymbol{K}_{x0}^\intercal (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{\Upsilon}_x^\intercal - \boldsymbol{\Upsilon}_x \boldsymbol{m}_0 \boldsymbol{m}_0^\intercal \boldsymbol{\Upsilon}_x^\intercal. \tag{85}$$

Hence, we can obtain the initial weight vector $\boldsymbol{m}_0$ and the initial weight matrix $\boldsymbol{S}_0$ by computing the mean and the covariance over the columns of the matrix $\boldsymbol{C}_0 = (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{K}_{x0}$.

5.1.2 The pre-image problem/recovering the state-space distribution

Recovering a distribution in the state space that is a pre-image of a given mean map is still a topic of ongoing research. There are several approaches to this problem, such as fitting a Gaussian mixture model (Mccalman et al. 2013), or sampling from the embedded distribution by optimization (Chen et al. 2010). In the experiments conducted for this paper, we approach the pre-image problem by matching a Gaussian distribution, which is a reasonable choice if the recovered distribution is unimodal. Since we embed the belief state for the kernel Kalman rule as a mean map and as a covariance operator, we can obtain the mean and covariance of a Gaussian approximations by simple matrix manipulations. The space of the samples $\mathbb{R}^d$ together with the linear kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = \boldsymbol{x}_1^\intercal \boldsymbol{x}_2$ forms an RKHS as well. Therefore, we can simply define a conditional embedding operator that maps from the Hilbert space of the feature vectors to the Hilbert space of the samples as

$$\hat{\mathcal{C}}_{\text{pre}} = \boldsymbol{X} (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{\Upsilon}_x^\intercal. \tag{86}$$

By applying this conditional operator now to the belief state, we obtain the mean of the embedded distribution in the sample space

$$\boldsymbol{\eta}_t = \hat{\mathcal{C}}_{\text{pre}} \mu_{X,t} = \hat{\mathcal{C}}_{\text{pre}} \mathbb{E}_{b_t}[\varphi(X)] = \mathbb{E}_{b_t}[\hat{\mathcal{C}}_{\text{pre}} \varphi(X)] = \mathbb{E}_{b_t}[X]. \tag{87}$$

Similarly, we can also apply this operator to the covariance embedding to obtain the covariance of the belief state in the sample space

$$\begin{aligned} \boldsymbol{\Sigma}_t &= \hat{\mathcal{C}}_{\text{pre}} \hat{\mathcal{C}}_{XX,t} \hat{\mathcal{C}}_{\text{pre}}^\intercal \\ &= \hat{\mathcal{C}}_{\text{pre}} \left(\mathbb{E}_{b_t} \left[\varphi(X) \otimes \varphi(X) \right] - \mu_{X,t} \otimes \mu_{X,t} \right) \hat{\mathcal{C}}_{\text{pre}}^\intercal \\ &= \mathbb{E}_{b_t} \left[\hat{\mathcal{C}}_{\text{pre}} \varphi(X) \otimes \varphi(X) \hat{\mathcal{C}}_{\text{pre}}^\intercal \right] - \hat{\mathcal{C}}_{\text{pre}} \mu_{X,t} \otimes \mu_{X,t} \mathcal{C}_{\text{pre}}^\intercal \\ &= \mathbb{E}_{b_t} \left[X \otimes X \right] - \boldsymbol{\eta}_t \otimes \boldsymbol{\eta}_t^\intercal \end{aligned} \tag{88}$$

However, also any other approach from the literature can be used in the kernel Kalman filter algorithm.

```
Algorithm 1: The Kernel Kalman Filter
input: triples $\{(\tilde{\boldsymbol{x}}_1, \boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\tilde{\boldsymbol{x}}_m, \boldsymbol{x}_m, \boldsymbol{y}_m)\}$,
       kernel functions $k$ and $g$, regularization parameters $\lambda$ and $\kappa$,
       let $\boldsymbol{X}$ be the matrix of all $x_i$ as columns, $\tilde{\boldsymbol{X}}$ and $\boldsymbol{Y}$ analogously,
       let $\boldsymbol{X}_0$ be the matrix of all data points used to compute the initial embeddings
compute kernel matrices
$\boldsymbol{K}_{xx} = k(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}_{\tilde{x}\tilde{x}} = k(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}})$, $\boldsymbol{K}_{\tilde{x}x} = k(\tilde{\boldsymbol{X}}, \boldsymbol{X})$, and $\boldsymbol{G}_{yy} = g(\boldsymbol{Y}, \boldsymbol{Y})$
compute model matrices
$\boldsymbol{T} = (\boldsymbol{K}_{\tilde{x}\tilde{x}} + \lambda \boldsymbol{I}_m)^{-1} \boldsymbol{K}_{\tilde{x}x}$ and $\boldsymbol{O} = (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m)^{-1} \boldsymbol{K}_{xx}$
compute initial embeddings
kernel matrix with samples of the initial distribution: $\boldsymbol{K}_0 = k(\boldsymbol{X}, \boldsymbol{X}_0)$, $\boldsymbol{C}_0 = (\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m)^{-1} \boldsymbol{K}_0$,
compute mean and variance over the columns: $\boldsymbol{m}_0 = \mathrm{mean}(\boldsymbol{C}_0)$, $\boldsymbol{S}_0 = \mathrm{var}(\boldsymbol{C}_0)$
loop
    if new observation $\boldsymbol{y}_t$ available then
        compute kernel Kalman gain
        $\boldsymbol{Q}_t = \boldsymbol{S}_t^- \boldsymbol{O}^\intercal (\boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^- \boldsymbol{O}^\intercal + \kappa \boldsymbol{I}_m)^{-1}$
        innovation update
        $\boldsymbol{m}_t^+ = \boldsymbol{m}_t^- + \boldsymbol{Q}_t (\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{m}_t^-)$
        $\boldsymbol{S}_t^+ = \boldsymbol{S}_t^- - \boldsymbol{Q}_t \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^-$
    transition update
    $\boldsymbol{m}_{t+1}^- = \boldsymbol{T} \boldsymbol{m}_t^+, \quad \boldsymbol{S}_{t+1}^- = \boldsymbol{T} \boldsymbol{S}_t^+ \boldsymbol{T}^\intercal + \boldsymbol{V}$
    project into state space
    $\boldsymbol{\eta}_t = \boldsymbol{X} \boldsymbol{O} \boldsymbol{m}_t, \quad \boldsymbol{\Sigma}_t = \boldsymbol{X} \boldsymbol{O} \boldsymbol{S}_t \boldsymbol{O}^\intercal \boldsymbol{X}^\intercal$
```

5.1.3 Embedding observation windows

So far, we assumed that we have access to the latent states $\boldsymbol{x}_i$ in our training set. However, in many setups we only have access to the partial observations $\boldsymbol{y}_i$ which do not have the Markov property. Yet, we can still learn a KKR model from the provided data by embedding time windows $\boldsymbol{y}_{t-k+1:t}$ of size k as internal state representation. Similar approaches have been used by auto-regressive HMMs (Shannon et al. 2013). With longer data windows, the transitions become more and more Markov. How many observation each data window has to contain depends on two factors: on the dimensionality of the underlying system and on the signal-to-noise ratio of the measurements $\boldsymbol{y}_i$.

5.2 The subspace kernel Kalman filter

The subspace kernel Kalman filter (subKKF) is an extension of the KKF that applies the subspace conditional embedding operator presented in Eq. 26 as well as the subspace formulation of the kernel Kalman rule derived in Sect. 4.4. In contrast to the KKF, we assume for the subKKF a data set of triples $\{(\boldsymbol{x}_1, \boldsymbol{x}'_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{x}'_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}'_i$ is the successor state to $\boldsymbol{x}_i$. The representation of the belief state changes from weight vector $\boldsymbol{m}_t$ and weight matrix $\boldsymbol{S}_t$ to the subspace projections of the embeddings $\boldsymbol{n}_t = \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{m}_t = \boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{m}_t$ and $\boldsymbol{P}_t = \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{S}_t \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} = \boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{S}_t \boldsymbol{K}_{x\bar{x}}$, respectively. Additionally, both update procedures of the kernel Kalman filter, the transition update and the innovation update, have to be substituted by their subspace counterparts. The transition update is realized by the subspace kernel sum rule and the innovation update by the subspace kernel Kalman rule. The equations are depicted in Algorithm 2.

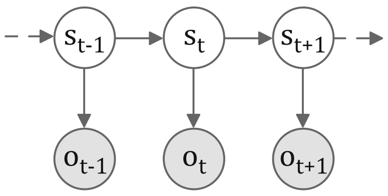


Fig. 5 Graphical model that we assume for the pendulum experiment

Since we represent the belief state as a projection into the subspace defined by $\mathbf{\Gamma}$, we can directly obtain the initial belief state by projecting the uniform embedding in the RKHS spanned by the samples from the initial distribution as

$$\boldsymbol{n}_0 = \mathbf{\Gamma}\mathbf{\Upsilon}_{x,0}\mathbf{1}_N\frac{1}{N} = \boldsymbol{K}_{\bar{x}0}\mathbf{1}_N\frac{1}{N}, \tag{89}$$

$$\boldsymbol{P}_0 = \frac{1}{N}\mathbf{\Gamma}\mathbf{\Upsilon}_{x,0}\mathbf{\Upsilon}_{x,0}^{\intercal}\mathbf{\Gamma}^{\intercal} = \frac{1}{N}\boldsymbol{K}_{\bar{x}0}\left(\boldsymbol{K}_{\bar{x}0}\right)^{\intercal}. \tag{90}$$

Here, $\boldsymbol{K}_{\bar{x}0}$ is the feature matrix of the subset and the samples from the initial state distribution. For the mapping back into the state space, we can similarly to the KKF define a subspace conditional operator as

$$\hat{\mathcal{C}}^S_{\text{pre}} := \boldsymbol{X}\boldsymbol{K}_{x\bar{x}}\left(\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m\right)^{-1}\mathbf{\Gamma}^{\intercal}. \tag{91}$$

By applying this operator to mean map and covariance embedding, we obtain the mean and variance in state space from the subspace projections as

$$\boldsymbol{\eta}_t = \boldsymbol{X}\boldsymbol{K}_{x\bar{x}}\left(\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m\right)^{-1}\boldsymbol{n}_t, \tag{92}$$

$$\mathbf{\Sigma}_t = \boldsymbol{X}\boldsymbol{K}_{x\bar{x}}\left(\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m\right)^{-1}\boldsymbol{P}_t\left(\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{K}_{x\bar{x}} + \lambda\boldsymbol{I}_m\right)^{-1}\boldsymbol{K}_{x\bar{x}}^{\intercal}\boldsymbol{X}^{\intercal}. \tag{93}$$

A concise description of the subspace kernel Kalman filter can be found in Algorithm 2.

5.3 Experimental evaluation of the kernel Kalman filter

We evaluate the performance of the KKF and the subKKF on two experiments on simulated environments, a pendulum and a quad-link, and one experiment on real-world data from a human motion tracking data set (Wojtusch and von Stryk 2015). For all kernel based methods, we use the squared exponential kernel, where we choose the kernel bandwidths according to the *median trick* (Jaakkola et al. 1999) and scale the median distances with a single optimized parameter.

5.3.1 Pendulum

In this experiment, we use a simulated pendulum as system dynamics. The state $s_0 = (q_0, \dot{q}_0)$ of the pendulum is initialized uniformly in the range $[0.1\pi, 0.4\pi]$ for the angle q_0 and in the range $[-0.5\frac{\pi}{s}, 0.5\frac{\pi}{s}]$ for the angular velocity $\dot{q}_0$. We simulate the pendulum with a frequency of 10,000 Hz and add normally distributed process noise with $\sigma = 0.1$. The filter methods observe the joint positions with additive Gaussian noise, i.e., $o_t \sim \mathcal{N}(q_t, 0.01)$ at a rate of 10 Hz. A graphical model of the pendulum is depicted in Fig. 5.

We compare the KKF, the subspace KKF (subKKF) and the KKF learned with the full data set (fullKKF) to version (a) of the kernel Bayes filter (KBF(a)) (Song et al. 2013) (the

Algorithm 2: The Subspace Kernel Kalman Filter

```
input: triples $\{(\boldsymbol{x}_1, \boldsymbol{x}'_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{x}'_n, \boldsymbol{y}_n)\}$,
       kernel functions $k$ and $g$, regularization parameters $\lambda$ and $\kappa$,
       let $\boldsymbol{X}$ be the matrix of all $x_i$ as columns, $\boldsymbol{X'}$ and $\boldsymbol{Y}$ analogously,
       let $\boldsymbol{X}_0$ be the matrix of all data points used to compute the initial embeddings
select subset of n samples
$\boldsymbol{X}_S \sim$ random strategy, or $\boldsymbol{X}_S \sim$ kernel activation heuristic
compute kernel matrices
$\boldsymbol{K}_{x\bar{x}} = k(\boldsymbol{X}, \boldsymbol{X}_S)$, $\boldsymbol{K}'_{x\bar{x}} = k(\boldsymbol{X'}, \boldsymbol{X}_S)$, and $\boldsymbol{G}_{yy} = g(\boldsymbol{Y}, \boldsymbol{Y})$
compute model matrices
$\boldsymbol{T}^S = \boldsymbol{K}'^{\intercal}_{x\bar{x}} \boldsymbol{K}_{x\bar{x}} \left(\boldsymbol{K}^{\intercal}_{x\bar{x}} \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}_n\right)^{-1}$ and $\boldsymbol{O}^S := \left(\boldsymbol{K}^{\intercal}_{x\bar{x}} \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}_n\right)^{-1}$
compute initial embeddings
kernel matrix with samples of the initial distribution: $\boldsymbol{K}_0^S = k(\boldsymbol{X}_S, \boldsymbol{X}_0)$
compute mean and variance over the columns: $\boldsymbol{m}_0 = \mathrm{mean}\left(\boldsymbol{K}_0^S\right)$, $\boldsymbol{S}_0 = \mathrm{var}\left(\boldsymbol{K}_0^S\right)$
loop
    if new observation $\boldsymbol{y}_t$ available then
        compute subspace kernel Kalman gain
        $\boldsymbol{Q}_t^S = \boldsymbol{P}_t^- \left(\boldsymbol{O}^S\right)^{\intercal} \left(\boldsymbol{K}^{\intercal}_{x\bar{x}} \boldsymbol{G}_{yy} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{P}_t^- \left(\boldsymbol{O}^S\right)^{\intercal} + \kappa \boldsymbol{I}_n\right)^{-1} \boldsymbol{K}^{\intercal}_{x\bar{x}}$
        note that you can apply the matrix $\boldsymbol{K}^{\intercal}_{x\bar{x}}$ to the kernel matrix $\boldsymbol{G}$ in the innovation update already
        at learning time to increase computational efficiency.
        innovation update
        $\boldsymbol{n}_t^+ = \boldsymbol{n}_t^- + \boldsymbol{Q}_t^S \left(\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{n}_t^-\right)$
        $\boldsymbol{P}_t^+ = \boldsymbol{P}_t^- - \boldsymbol{Q}_t^S \boldsymbol{G}_{yy} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{P}_t^-$
    transition update
    $\boldsymbol{n}_{t+1}^- = \boldsymbol{T}^S \boldsymbol{n}_t^+, \quad \boldsymbol{P}_{t+1}^- = \boldsymbol{T}^S \boldsymbol{P}_t^+ \left(\boldsymbol{T}^S\right)^{\intercal} + \boldsymbol{V}_S$
    project into state space
    $\boldsymbol{\eta}_t = \boldsymbol{X} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{n}_t, \quad \boldsymbol{\Sigma}_t = \boldsymbol{X} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{P}_t \boldsymbol{O}^S \boldsymbol{K}^{\intercal}_{x\bar{x}} \boldsymbol{X}^{\intercal}$
```

other versions, KBF(b) and KBF(c), have yielded worse results in this experiment) and the kernel Kalman filter with covariance embedding operator (KKF-CEO) (Zhu et al. 2014), as well as to standard filtering approaches such as the EKF (Julier and Uhlmann 1997) and the UKF (Wan and Van Der Merwe 2000) (which require a model of the system dynamics). To learn the models, we simulate 10 episodes with a length of 30 steps (3 s), i.e., 300 samples in total. Instead of the true state s_t of the pendulum, we use a window of 4 samples to represent the latent state. For the KKF and all KBF models, we use a kernel size of 100 samples, for the fullKKF, we use all available training samples and for the subKKF we use a set of 100 samples to span the subspace and the full data set to learn the operators. The samples for the subspace are selected from the full data set using the kernel activation heuristic. The results are shown in Fig. 6. The KKF and subKKF show clearly better results than all other non-parametric filtering methods and reach a performance level close to the EKF and UKF.

5.3.2 Quad-link

In this experiment, we use a simulated 4-link pendulum where we observe the 2-D end-effector positions. The state $\boldsymbol{s}_t$ of the pendulum consists of the four joint angles $\boldsymbol{q}_t$ and joint velocities $\dot{\boldsymbol{q}}_t$. The first and the last joints $q_{0,t=0}, q_{3,t=0}$ are initialized uniformly in the range

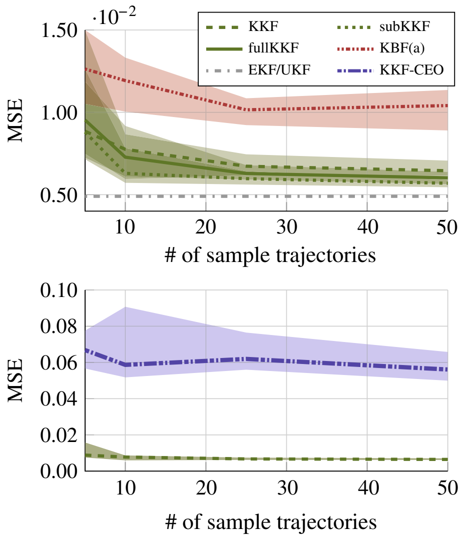


Fig. 6 Comparison of KKF to KBF(b), KKF-CEO, EKF and UKF. All kernel methods (except fullKKF) use kernel matrices of 100 samples. The subKKF method uses a subset of 100 samples and the whole data set to learn the conditional operators. Depicted is the median MSE to the ground-truth of 20 trials with the [0.25, 0.75] quantiles

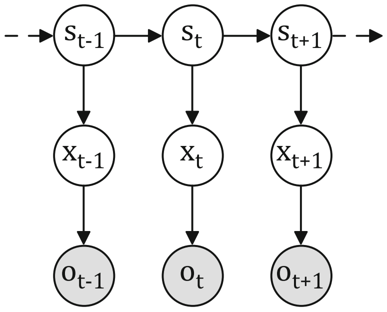

$$\begin{tikzcd}
\arrow[r, dashed] & s_{t-1} \arrow[r] \arrow[d] & s_t \arrow[r] \arrow[d] & s_{t+1} \arrow[r, dashed] \arrow[d] & {} \\
& x_{t-1} \arrow[d] & x_t \arrow[d] & x_{t+1} \arrow[d] & \\
& o_{t-1} & o_t & o_{t+1} &
\end{tikzcd}$$

Fig. 7 Graphical model that we assume for the quad-link experiment. The states s_t contain the joint angles and velocities, x_t is the position of the endeffector, and o_t the noise observation thereof

$[-0.55\pi, -0.45\pi]$ for $q_{0,t=0}$, and $[-0.5\pi, 0.5\pi]$ for $q_{3,t=0}$. The remaining joints and the joint velocities have all been initialized at 0.0. We simulate with Gaussian process noise with $\sigma = 0.01$. The filter methods observe the end-effector positions x_t with Gaussian observation noise as $o_t \sim \mathcal{N}(x_t, 0.001)$ at a rate of 10 Hz. As we assume, that we have no access to the true states, we use data windows of size 4 as representation of the latent state to learn the models. A graphical model of the system is depicted in Fig. 7.

We evaluate the prediction performance of the subKKF in comparison to the KKF-CEO, the EKF and the UKF. All other non-parametric filtering methods could not achieve a good performance or are not feasible due to the very high computation times. As the subKKF outperformed the KKF in the previous experiments and is also computationally much cheaper, we skip the comparison to the standard KKF in this and also the following experiments. We use a subspace of 500 samples, which is selected according to the kernel activation heuristic, and learn the subKKF with the full data set of 3000 samples.

In a first qualitative evaluation, we compare the long-term prediction performance of the subKKF in comparison to the UKF, the EKF and the Monte–Carlo filter (MCF) as a baseline. This evaluation is shown in Fig. 8. The first five steps of of the end-effector trajectories are observed by the filters, the following 30 steps are predicted. The UKF is not able to predict

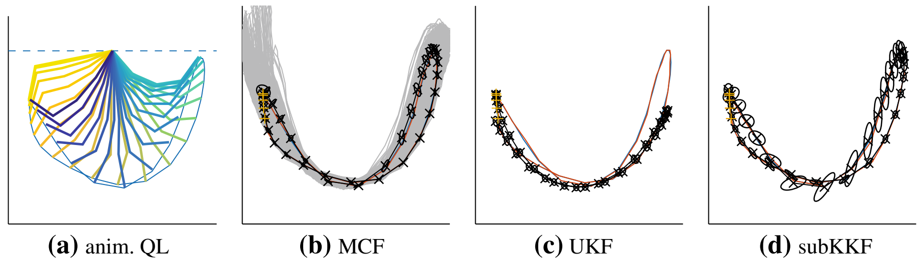
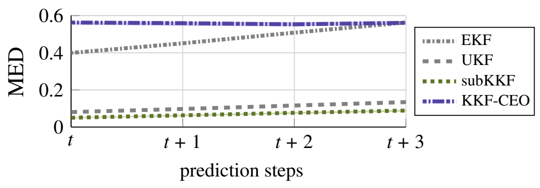


Fig. 8 Example trajectory of the quad-link end-effector. The filter outputs in black, where the ellipses enclose 90% of the probability mass. All filters were updated with the first five measurements (yellow marks) and predicted the following 30 steps. **a** Animation of the trajectory, **b-d** depict the outputs of Monte-Carlo filter, unscented Kalman filter, and subspace kernel Kalman filter, respectively (Color figure online)

Fig. 9 1, 2 and 3 step prediction performances in mean euclidean distances (MED) to the true end-effector positions of the quad-link



the movements of the quad-link end-effector due to the high non-linearity, while the subKKF is able to predict the whole trajectory.

We also compared the 1, 2 and 3-step prediction performance of the subKKF to the KKF-CEO, EKF and UKF (Fig. 9). The KKF-CEO provides poor results already for the filtering task. The EKF performs equally bad, since the observation model is highly non-linear. The UKF already yields a much better performance as it does not suffer from the linearizion of the system dynamics. The subKKF performs slightly better than the UKF.

5.3.3 Human motion data

The human motion dynamics (HuMoD) database by Wojtusch and von Stryk (2015) consists of the data sets of several motions executed by two subjects. All data sets contain the recordings from a motion capture system with 36 markers as well as the recordings of the electrical activity of 14 muscles in the legs. Additionally, data from the treadmill such as ground reaction forces and velocities are available. The x-, y-, and z-locations of the markers were recorded at 500 Hz, the muscle activities at 2000 Hz, and the data from the treadmill at 1000 Hz. Furthermore, the database contains joint positions and joint trajectories derived from the marker positions via kinematic models of the human body. In our experiments, we use the marker locations, the derived locations of the joints and the muscle activities. We subsample all data to a common frame rate of 50 Hz and transpose the x- and z-position of all markers such that the T12-marker (marker at the 12th thoracic vertebra) has $(x = 0, z = 0)$ in all frames. Note that in the HuMoD database, the x-axis points in the motion direction (i.e., along the treadmill), the y-axis points upwards and the z-axis forms a right-hand coordinate system towards the right side of the treadmill. We used walking motions at 1.0 m/s, 1.5 m/s, 2.0 m/s, and running motions at 2.0 m/s, 3.0 m/s, and 4 m/s, captured from one subject. For evaluating the trained model, we used a test data-set in which the subject transitions linearly from 0 m/s up to 4 m/s and back to 0 m/s.

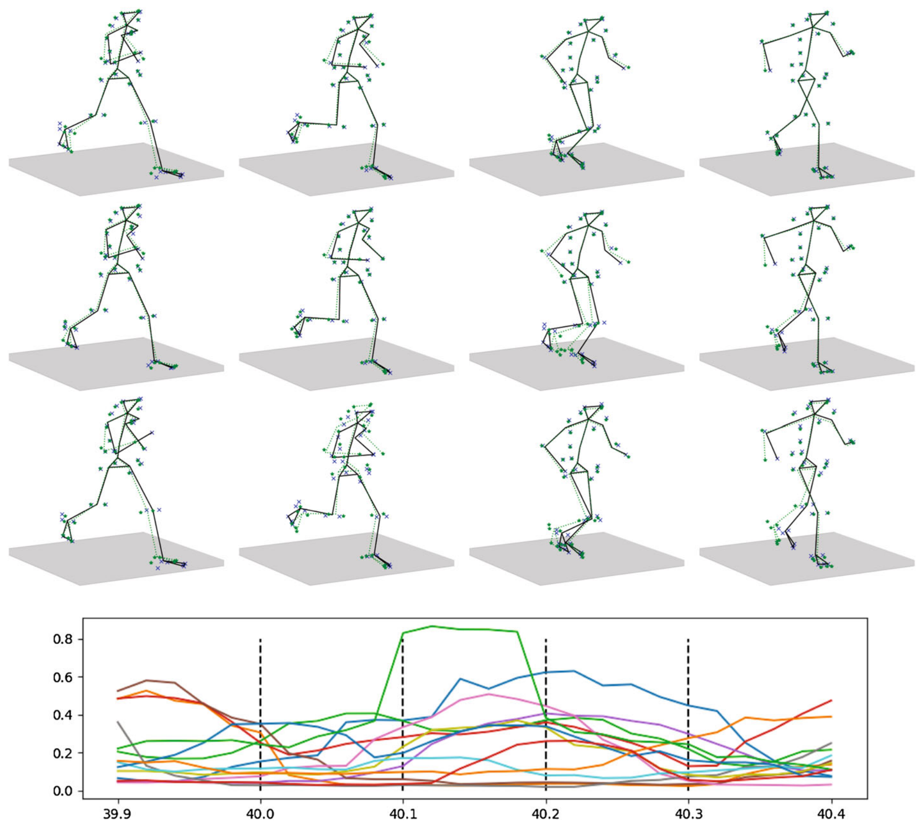


Fig. 10 Example sequence of 4 postures and the measured muscle activities. The marker and skeleton in green depict the ground-truth, the estimates from the models are depicted in black/blue. The learned models estimate the marker and joint positions from the muscle activities. The first row shows the estimated positions from the subKKF, the second row shows the estimated positions from the subKBF and the third row shows the estimated positions from a sparse GP. For all three models, we use a sample set of 2000 samples and a sparse subset of 500 samples (Color figure online)

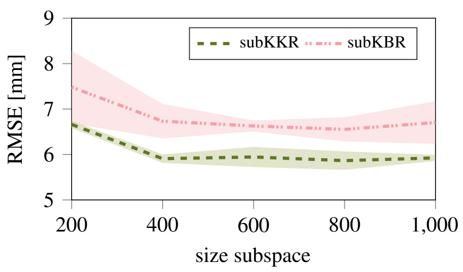


Fig. 11 Performance of the subKKF and the subKBF on the HuMoD transition data for different sizes of the subspace

In the experiment, we compare the performance of subKKF, subKBF, and sparse Gaussian process (SGP) in restoring the marker and joint positions from the muscle activities. We learn all three models using the marker and joint positions as state variables (or outputs) $\boldsymbol{x}_i$ and the muscle activities as observations (or inputs) $\boldsymbol{y}_i$. We use a set of 2000 samples to learn the kernel matrices and a subset of 500 samples to define the subspace (or as inducing inputs).

Table 2 Top: performance of subKKF, subKBF, and SGP for the HuMoD transition data for different sizes of the subspace. Bottom: time consumptions of subKKF and subKBF for filtering 100 test sequences of length 50 from the HuMoD data set

#	200	400	600	800	1000
subKKF	6.66	5.91	5.94	5.87	5.92
subKBF	7.48	6.73	6.63	6.55	6.70
SGP	76.33	70.15	68.46	63.13	58.97
subKKF	0.61 ± 0.10 s	1.71 ± 0.17 s	3.84 ± 0.40 s	7.21 ± 0.85 s	11.72 ± 0.41 s
subKBF	59 ± 1.3 s	170 ± 11.3 s	352 ± 15 s	620 ± 15 s	927 ± 174 s

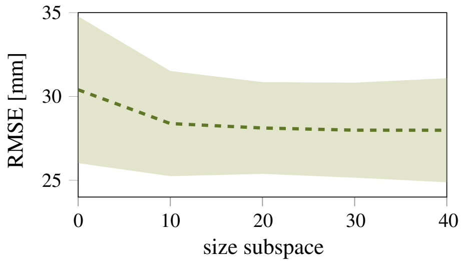


Fig. 12 Performance of the subKKF on HuMoD test sequences after 0, 10, 20, 30 and 40 iterations of the CMA-ES optimizer

For subKKF and subKBF, we use a window size of 2. While we could easily carry out the optimization of the parameters for the subKKF and for the SGP, the optimization of the parameters for the subKBF was not feasible in a considerable amount of time.

Figure 10 depicts marker and joint positions of four exemplary postures together with the muscle activities during that period of time. The locations of the exemplary postures in the time line are depicted by vertical lines in the plot of the muscle activities. While this is only a qualitative example, it depicts how the subKKF outperforms the subKBF and the SGP in restoring the positions of the markers and the joints.

We compare the performance of subKKF, subKBF and SGP for different sizes of the subspace (inducing inputs). Figure 11 depicts the performance of subKKF and subKBF. The results of the SGP can be seen in Table 2 which are clearly worse in comparison to the filtering approaches which take the temporal correlation of the data into account. Furthermore, Table 2 also depicts the time consumption of subKKF and subKBF for filtering 100 test sequences of length 50. The gain in efficiency of the subKKF over the subKBF which is around the factor 100 can be seen clearly. In Fig. 12, we depict the performance gain of the subKKF over the number of iterations of the CMA-ES optimizer. We see that in this case, the first 10 iterations yield a bigger jump in performance than the following 40 steps. However, from our experience, this is very specific to the problem and the initial setting of the parameters, which were in this experiment already very close to the optimal parameters.

5.4 The kernel forward–backward smoother

Smoothing is in contrast to filtering a post-processing routine. While filtering refers to a routine where the current state is estimated recursively from all past observations, smoothing computes the best state estimates given all available observations from the past and the future. Hence, for a given time series of observations $[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T]$, we want to obtain the belief $p(\boldsymbol{x}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$ for all $1 \leq t \leq T$.

One well-known and simple approach to smoothing is the forward–backward smoother. During a forward pass the standard filtering algorithm is applied to the observations. Afterwards, during the backward pass, an inverse filter is applied to the same time series of observations. The filter estimates of forward and backward pass are finally combined into the smoothed estimates. Since the information from the observation should be incorporated only once into the smoothed estimates, we need to combine the posterior estimates of the forward pass with the prior estimates of the backward pass (or vice versa). For the case of ordinary Kalman filters, the backward pass is hard to realize because of two problems. First, it requires an inverse model of the underlying system for the backward pass, and second, an initialization of the belief at the final state is necessary. These issues are however not applicable for the KKF as we can learn both, the inverse models and the embedding of initial distribution of the final states from data.

5.4.1 Computing the smoothed belief state as a weighted average

Assuming that we have the a-posteriori belief states from the forward pass and the a-priori belief states from the backward pass as

$$\left\{\left(\mu^+_{f,1}, \mathcal{C}^+_{f,1}\right), \dots, \left(\mu^+_{f,T}, \mathcal{C}^+_{f,T}\right)\right\} \quad \text{and} \quad \left\{\left(\mu^-_{b,1}, \mathcal{C}^-_{b,1}\right), \dots, \left(\mu^-_{b,T}, \mathcal{C}^-_{b,T}\right)\right\}, \tag{94}$$

respectively, we can combine the mean maps into a smoothed belief state as the weighted average

$$\mu_{s,t} = \mathcal{Z}_{f,t}\mu^+_{f,t} + \mathcal{Z}_{b,t}\mu^-_{b,t}. \tag{95}$$

Since both, the estimator from the forward pass and the estimator from the backward pass, are unbiased, the weighting operators $\mathcal{Z}_{f,t}$ and $\mathcal{Z}_{b,t}$ need to satisfy $\mathcal{I} = \mathcal{Z}_{f,t} + \mathcal{Z}_{b,t}$ in order to get an unbiased estimator of the smoothed mean map, i.e.,

$$\mathbb{E}\left[\varphi(X_t) - \mu^s_t\right] \overset{!}{=} 0 \tag{96}$$

$$\mathbb{E}\left[\varphi(X_t) - \mathcal{Z}_{f,t}\mu^{f+}_t + \mathcal{Z}_{b,t}\mu^{b-}_t\right] \overset{!}{=} 0 \tag{97}$$

$$\mathbb{E}\left[\varphi(X_t)\right] - \mathcal{Z}_{f,t}\mathbb{E}\left[\mu^{f+}_t\right] + \mathcal{Z}_{b,t}\mathbb{E}\left[\mu^{b-}_t\right] \overset{!}{=} 0 \tag{98}$$

$$\mu_t - \left(\mathcal{Z}_{f,t} + \mathcal{Z}_{b,t}\right)\mu_t \overset{!}{=} 0 \tag{99}$$

$$\Rightarrow \quad \mathcal{Z}_{f,t} + \mathcal{Z}_{b,t} = \mathcal{I} \tag{100}$$

Thus, the weighting operators can be expressed by each other as $\mathcal{Z}_{b,t} = \mathcal{I} - \mathcal{Z}_{f,t}$ and vice versa. We substitute this representation back into the smoothing update in Eq. 95 to obtain

$$\mu_{s,t} = \mathcal{Z}_{f,t}\mu^+_{f,t} + \left(\mathcal{I} - \mathcal{Z}_{f,t}\right)\mu^-_{b,t}. \tag{101}$$

5.4.2 Finding the optimal weighting operators

We obtain the optimal weighting operators by minimizing the squared error of the smoothing mean map which is equivalent to minimizing the trace of the smoothed covariance embedding operator $\mathcal{C}_{s,t}$, i.e.,

$$\min \mathbb{E}\left[\left(\varphi(X_t) - \mu^s_t\right)^\intercal \left(\varphi(X_t) - \mu^s_t\right)\right] = \min \mathbb{E}\left[\operatorname{Tr}\left(\varphi(X_t) - \mu^s_t\right)\left(\varphi(X_t) - \mu^s_t\right)^\intercal\right] \tag{102}$$

$$= \min \operatorname{Tr} \mathcal{C}_{s,t}. \tag{103}$$

We can then use Eq. 101 to rewrite the covariance operator as

$$\mathcal{C}_{s,t} = \mathbb{E}\left[\left(\varphi(X_t) - \mathcal{Z}_{f,t}\mu^+_{f,t} + \left(\mathcal{I} - \mathcal{Z}_{f,t}\right)\mu^-_{b,t}\right)\left(\dots\right)^\intercal\right] \tag{104}$$

$$= \mathbb{E}\left[\left(\mathcal{Z}_{f,t}(\epsilon_f - \epsilon_b) + \epsilon_b\right)\left(\dots\right)^\intercal\right], \tag{105}$$

where $\epsilon_f = \varphi(x_t) - \mu^+_{f,t}$ is the error of the a-posteriori estimate of the forward pass and $\epsilon_b = \varphi(x_t) - \mu^-_{b,t}$ is the error of the a-priori estimate of the backward pass and where we used the relation $\varphi(X_t) = \mathcal{I}\varphi(X_t) = (\mathcal{Z}_{f,t} + \mathcal{Z}_{b,t})\varphi(X_t) = (\mathcal{Z}_{f,t} + (\mathcal{I} - \mathcal{Z}_{f,t}))\varphi(X_t)$. By expanding the square and since the cross-covariances of the errors from forward and the backward pass are zero (i.e., $\mathbb{E}[\epsilon_f\epsilon_b^\intercal] = 0$), we arrive at

$$\mathcal{C}_{s,t} = \mathbb{E}\left[\mathcal{Z}_{f,t}\left(\epsilon_f\epsilon_f^\intercal + \epsilon_b\epsilon_b^\intercal\right)\mathcal{Z}_{f,t}^\intercal - \mathcal{Z}_{f,t}\epsilon_b\epsilon_b^\intercal - \epsilon_b\epsilon_b^\intercal\mathcal{Z}_{f,t}^\intercal + \epsilon_b\epsilon_b^\intercal\right]. \tag{106}$$

Lastly, we can take the derivative and set it to zero to obtain the optimal $\mathcal{Z}_{f,t}$ as

$$0 \overset{!}{=} \frac{\partial \operatorname{Tr}\mathcal{C}_{s,t}}{\partial \mathcal{Z}_{f,t}} = 2\mathbb{E}\left[\mathcal{Z}_{f,t}\left(\epsilon_f\epsilon_f^\intercal + \epsilon_b\epsilon_b^\intercal\right) - \epsilon_b\epsilon_b^\intercal\right] \tag{107}$$

$$0 \overset{!}{=} \mathcal{Z}_{f,t}\left(\mathbb{E}\left[\epsilon_f\epsilon_f^\intercal\right] + \mathbb{E}\left[\epsilon_b\epsilon_b^\intercal\right]\right) - \mathbb{E}\left[\epsilon_b\epsilon_b^\intercal\right] \tag{108}$$

$$0 \overset{!}{=} \mathcal{Z}_{f,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right) - \mathcal{C}^-_{b,t} \tag{109}$$

$$\mathcal{Z}_{f,t} = \mathcal{C}^-_{b,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right)^{-1}. \tag{110}$$

From the condition on the weighting operators stated in Eq. 100, it furthermore follows that $\mathcal{Z}_{b,t} = \mathcal{C}^-_{f,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right)^{-1}$.

5.4.3 Smoothing the covariance embedding operator

Taking the representation of the smoothed covariance in Eq. 106 and substituting the covariance operators and the optimal weighting operator $\mathcal{Z}_{f,t}$ gives us the following smoothed covariance operator

$$\mathcal{C}_{s,t} = \mathcal{C}^-_{b,t} - \mathcal{C}^-_{b,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right)^{-1}\mathcal{C}^-_{b,t} \tag{111}$$

$$= \mathcal{C}^-_{b,t} - \left(\mathcal{I} - \mathcal{C}^+_{f,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right)^{-1}\right)\mathcal{C}^-_{b,t} \tag{112}$$

$$= \mathcal{C}^+_{f,t}\left(\mathcal{C}^+_{f,t} + \mathcal{C}^-_{b,t}\right)^{-1}\mathcal{C}^-_{b,t} \tag{113}$$

From the optimal solution of the weighting operator, we can now see that the smoothing update of the covariance embedding operator can be expressed as

$$\mathcal{C}_{s,t} = \mathcal{Z}_{b,t}\mathcal{C}^-_{b,t} = \mathcal{Z}_{f,t}\mathcal{C}^+_{f,t} \tag{114}$$

In the following section, we will show how the smoothing update can be expressed with finite samples using vector/matrix operations.

5.4.4 The empirical kernel forward–backward smoother

We assume that we are given the weight vectors and weight matrices from the forward and the backward pass as $\{(\boldsymbol{m}^+_{f,1}, \boldsymbol{S}^+_{f,1}), \dots, (\boldsymbol{m}^+_{f,T}, \boldsymbol{S}^+_{f,T})\}$ and $\{(\boldsymbol{m}^-_{b,1}, \boldsymbol{S}^-_{b,1}), \dots, (\boldsymbol{m}^-_{b,T}, \boldsymbol{S}^-_{b,T})\}$, respectively. Since the weighting operator $\hat{\mathcal{Z}}_{b,t}$ can be expressed by $\hat{\mathcal{Z}}_{f,t}$ and vice versa, we only need to compute one of the weighting operators which we choose to be $\hat{\mathcal{Z}}_{f,t}$. We add the identity operator with a small scalar γ in the inverse to improve the numerical stability and obtain

$$\begin{aligned}
\hat{\mathcal{Z}}_{f,t} &= \hat{\mathcal{C}}^-_{b,t}\left(\hat{\mathcal{C}}^+_{f,t} + \hat{\mathcal{C}}^-_{b,t} + \gamma\mathcal{I}\right)^{-1} && (115)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{S}^-_{b,t}\boldsymbol{\Upsilon}_x^\intercal\left(\boldsymbol{\Upsilon}_x\left(\boldsymbol{S}^+_{f,t} + \boldsymbol{S}^-_{b,t}\right)\boldsymbol{\Upsilon}_x^\intercal + \gamma\mathcal{I}\right)^{-1} && (116)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{S}^-_{b,t}\left(\boldsymbol{\Upsilon}_x^\intercal\boldsymbol{\Upsilon}_x\left(\boldsymbol{S}^+_{f,t} + \boldsymbol{S}^-_{b,t}\right) + \gamma\boldsymbol{I}_n\right)^{-1}\boldsymbol{\Upsilon}_x^\intercal && (117)\\
&= \boldsymbol{\Upsilon}_x \underbrace{\boldsymbol{S}^-_{b,t}\left(\boldsymbol{K}_{xx}\left(\boldsymbol{S}^+_{f,t} + \boldsymbol{S}^-_{b,t}\right) + \gamma\boldsymbol{I}_n\right)^{-1}}_{\boldsymbol{Z}_{f,t}}\boldsymbol{\Upsilon}_x^\intercal, && (118)
\end{aligned}$$

where we use the matrix identity $\boldsymbol{A}(\boldsymbol{BA} + \boldsymbol{I})^{-1} = (\boldsymbol{AB} + \boldsymbol{I})^{-1}\boldsymbol{A}$ and defined the finite weighting matrix $\boldsymbol{Z}_{f,t}$. With this weighting matrix we can now combine the mean maps of the forward and the backward pass as

$$\begin{aligned}
\hat{\mu}_{s,t} &= \hat{\mathcal{Z}}_{f,t}\hat{\mu}^+_{f,t} + \left(\mathcal{I} - \hat{\mathcal{Z}}_{f,t}\right)\hat{\mu}^-_{b,t} && (119)\\
&= \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{\Upsilon}_x^\intercal\hat{\mu}^+_{f,t} + \left(\mathcal{I} - \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{\Upsilon}_x^\intercal\right)\hat{\mu}^-_{b,t} && (120)\\
&= \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\boldsymbol{m}^+_{f,t} + \boldsymbol{\Upsilon}_x\boldsymbol{m}^-_{b,t} - \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\boldsymbol{m}^-_{b,t} && (121)\\
\boldsymbol{\Upsilon}_x\boldsymbol{m}^s_t &= \boldsymbol{\Upsilon}_x\left(\boldsymbol{m}^-_{b,t} + \boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\left(\boldsymbol{m}^+_{f,t} - \boldsymbol{m}^-_{b,t}\right)\right). && (122)
\end{aligned}$$

And similarly we can obtain the smoothed estimate of the covariance operator as

$$\begin{aligned}
\hat{\mathcal{C}}^s_t &= \hat{\mathcal{Z}}_{f,t}\hat{\mathcal{C}}^+_{f,t} && (123)\\
&= \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{\Upsilon}_x^\intercal\boldsymbol{\Upsilon}_x\boldsymbol{S}^+_{f,t}\boldsymbol{\Upsilon}_x^\intercal && (124)\\
\boldsymbol{\Upsilon}_x\boldsymbol{S}^s_t\boldsymbol{\Upsilon}_x^\intercal &= \boldsymbol{\Upsilon}_x\boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\boldsymbol{S}^+_{f,t}\boldsymbol{\Upsilon}_x^\intercal && (125)
\end{aligned}$$

A concise description of the kernel forward–backward smoothing algorithm can be found in Algorithm 3.

5.4.5 Initialization of the backward kernel Kalman filter

A critical aspect of the classical forward–backward smoothing algorithm is the initialization of the belief state for the backward pass. Often the distribution over the initial state is well known but not a distribution over the terminal state, where it is often not even clear how a terminal state is defined. For the backward kernel Kalman filter, two approaches can be used to initialize the belief state. The first approach assumes that we have multiple episodes in the training data, where each episode terminates in a terminal state of the system. We can then compute the initialization for the backward pass analogously to the initialization for the KKF described in Sect. 5.1.1. The second approach simply assumes that the system has

Algorithm 3: The Kernel Forward–Backward Smoother

```
input: data set $\mathcal{D}_{\tilde{X}XX'Y} = \{(\tilde{\boldsymbol{x}}_1, \boldsymbol{x}_1, \boldsymbol{x}'_1, \boldsymbol{y}_1), \dots, (\tilde{\boldsymbol{x}}_n, \boldsymbol{x}_n, \boldsymbol{x}'_n, \boldsymbol{y}_n)\}$,
       kernel functions $k$ and $g$, regularization parameters $\lambda$ and $\kappa$,
       let $\boldsymbol{X}$ be the matrix of all $x_i$ as columns, $\tilde{\boldsymbol{X}}$, $\boldsymbol{X}'$ and $\boldsymbol{Y}$ analogously,
       let $\boldsymbol{X}_0$ be the matrix of all data points used to initialize the forward pass
       let $\boldsymbol{X}_T$ be the matrix of all data points used to initialize the backward pass
learn the forward filter
See Algorithm 1 with data matrices $\boldsymbol{X}_0$, $\tilde{\boldsymbol{X}}$, $\boldsymbol{X}$, and $\boldsymbol{Y}$.
learn the backward filter
See Algorithm 1 with data matrices $\boldsymbol{X}_T$, $\boldsymbol{X}$, $\boldsymbol{X}'$, and $\boldsymbol{Y}$, note that you need to learn the transition model
from $\boldsymbol{X}'$ to $\boldsymbol{X}$.
apply forward filter
Compute filtered estimates $\left\{\left(\boldsymbol{m}^+_{f,1}, \boldsymbol{S}^+_{f,1}\right), \dots, \left(\boldsymbol{m}^+_{f,T}, \boldsymbol{S}^+_{f,T}\right)\right\}$
apply backward filter and compute smoothed estimates
loop
    if observation $\boldsymbol{y}_t$ available then
        compute kernel Kalman gain
        $\boldsymbol{Q}_{b,t} = \boldsymbol{S}^-_{b,t}\boldsymbol{O}^\intercal(\boldsymbol{G}_{yy}\left(\boldsymbol{O}\boldsymbol{S}^-_{b,t}\boldsymbol{O}^\intercal + \boldsymbol{R}_b\right) + \kappa\boldsymbol{I})^{-1}$
        innovation update
        $\boldsymbol{m}^+_{b,t} = \boldsymbol{m}^-_{b,t} + \boldsymbol{Q}_{b,t}(\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy}\boldsymbol{O}\boldsymbol{m}^-_t)$
        $\boldsymbol{S}^+_{b,t} = \boldsymbol{S}^-_{b,t} - \boldsymbol{Q}_{b,t}\boldsymbol{G}_{yy}\boldsymbol{O}\boldsymbol{S}^-_{b,t}$
    transition update
    $\boldsymbol{m}^-_{b,t-1} = \boldsymbol{T}_b\boldsymbol{m}^+_{b,t}, \quad \boldsymbol{S}^-_{b,t-1} = \boldsymbol{T}_b\boldsymbol{S}^+_{b,t}\boldsymbol{T}_b^\intercal + \boldsymbol{V}_b$
    compute smoothed estimate
    $\boldsymbol{Z}_{f,t} = \boldsymbol{S}^-_{b,t}\left(\boldsymbol{K}_{xx}\left(\boldsymbol{S}^+_{f,t} + \boldsymbol{S}^-_{b,t}\right) + \gamma\boldsymbol{I}_n\right)^{-1}$
    $\boldsymbol{m}^s_t = \boldsymbol{m}^-_{b,t} + \boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\left(\boldsymbol{m}^+_{f,t} - \boldsymbol{m}^-_{b,t}\right)$
    $\boldsymbol{S}^s_t = \boldsymbol{Z}_{f,t}\boldsymbol{K}_{xx}\boldsymbol{S}^+_{f,t}$
    project into state space
    $\boldsymbol{\eta}^s_t = \boldsymbol{X}\boldsymbol{O}\boldsymbol{m}^s_t, \quad \boldsymbol{\Sigma}^s_t = \boldsymbol{X}\boldsymbol{O}\boldsymbol{S}^s_t\boldsymbol{O}^\intercal\boldsymbol{X}^\intercal$
```

a stationary distribution which is covered by the training data. The initialization is then the embedding of the distribution over all the samples in the training set.

5.5 The subspace kernel forward–backward smoother

If we use the subspace kernel Kalman filter to perform the forward and the backward pass, we obtain as outcome the subspace projections $\boldsymbol{n}_t$ of the mean map and $\boldsymbol{P}_t$ covariance embedding instead of the weight vectors $\boldsymbol{m}_t$ and weight matrices $\boldsymbol{S}_t$, respectively. To perform smoothing on these subspace projections, we need to find the weighting matrices for the smoothing update analog to Eq. 101. Though, as the representation is already in a finite domain, we can directly apply the optimal solution found in Eq. 110 to the subspace projections of the covariance operator. Hence, the weighting matrix for the subspace kernel forward–backward smoother (subKFBS) becomes

$$\boldsymbol{Z}^S_{f,t} = \boldsymbol{P}^-_{b,t}\left(\boldsymbol{P}^+_{f,t} + \boldsymbol{P}^-_{b,t}\right)^{-1} \tag{126}$$

Algorithm 4: The Subspace Kernel Forward–Backward Smoother

```
input: data set $\mathcal{D}_{\tilde{X}XX'Y} = \{(\tilde{\boldsymbol{x}}_1, \boldsymbol{x}_1, \boldsymbol{x}'_1, \boldsymbol{y}_1), \ldots, (\tilde{\boldsymbol{x}}_n, \boldsymbol{x}_n, \boldsymbol{x}'_n, \boldsymbol{y}_n)\}$,
       kernel functions $k$ and $g$, regularization parameters $\lambda$ and $\kappa$,
       let $\boldsymbol{X}$ be the matrix of all $x_i$ as columns, $\tilde{\boldsymbol{X}}$, $\boldsymbol{X}'$ and $\boldsymbol{Y}$ analogously,
       let $\boldsymbol{X}_0$ be the matrix of all data points used to initialize the forward pass
       let $\boldsymbol{X}_T$ be the matrix of all data points used to initialize the backward pass
learn forward and backward filter
See Algorithm 1 with data matrices $\boldsymbol{X}_0$, $\boldsymbol{X}$, $\boldsymbol{X}'$, and $\boldsymbol{Y}$ for the forward filter and data matrices $\tilde{\boldsymbol{X}}$, $\boldsymbol{X}$, and $\boldsymbol{Y}$ for the backward filter. Note that you need to learn the transition model for the backward filter from $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}}$.
apply forward filter
Compute filtered estimates $\left\{\left(\boldsymbol{n}^+_{f,1}, \boldsymbol{P}^+_{f,1}\right), \ldots, \left(\boldsymbol{n}^+_{f,T}, \boldsymbol{P}^+_{f,T}\right)\right\}$
apply backward filter and compute smoothed estimates
loop
    if observation $\boldsymbol{y}_t$ available then
        compute subspace kernel Kalman gain
        $\boldsymbol{Q}^S_t = \boldsymbol{P}^-_{b,t}\left(\boldsymbol{O}^S\right)^\intercal\left(\boldsymbol{K}^\intercal_{x\bar{x}}\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}^-_{b,t}\left(\boldsymbol{O}^S\right)^\intercal + \kappa\boldsymbol{I}_m\right)^{-1}\boldsymbol{K}^\intercal_{x\bar{x}}$
        innovation update
        $\boldsymbol{n}^+_{b,t} = \boldsymbol{n}^-_{b,t} + \boldsymbol{Q}^S_t\left(\boldsymbol{g}_{\boldsymbol{y}_t} - \boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{n}^-_{b,t}\right)$,   $\boldsymbol{P}^+_{b,t} = \boldsymbol{P}^-_{b,t} - \boldsymbol{Q}^S_t\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}^-_{b,t}$
    transition update
    $\boldsymbol{n}^-_{b,t-1} = \boldsymbol{T}^S\boldsymbol{n}^+_{b,t}$,   $\boldsymbol{P}^-_{b,t-1} = \boldsymbol{T}^S\boldsymbol{P}^+_{b,t}\left(\boldsymbol{T}^S\right)^\intercal + \boldsymbol{V}^S$
    compute smoothed estimate
    $\boldsymbol{Z}^S_{f,t} = \boldsymbol{P}^-_{b,t}\left(\boldsymbol{P}^+_{f,t} + \boldsymbol{P}^-_{b,t}\right)^{-1}$
    $\boldsymbol{n}_{s,t} = \boldsymbol{Z}^S_{f,t}\boldsymbol{n}^+_{f,t} + \left(\boldsymbol{I}_m - \boldsymbol{Z}^S_{f,t}\right)\boldsymbol{n}^-_{b,t}$,   $\boldsymbol{P}_{s,t} = \boldsymbol{Z}^S_{f,t}\boldsymbol{P}^+_{f,t}$
    project into state space
    $\boldsymbol{\eta}_{s,t} = \boldsymbol{X}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{n}_{s,t}$,   $\boldsymbol{\Sigma}_{s,t} = \boldsymbol{X}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}_{s,t}\boldsymbol{O}^S\boldsymbol{K}^\intercal_{x\bar{x}}\boldsymbol{X}^\intercal$
```

From here, we can obtain the equations for the smoothing update of the subspace kernel forward–backward smoother easily by

$$\boldsymbol{n}_{s,t} = \boldsymbol{Z}^S_{f,t}\boldsymbol{n}^+_{f,t} + \left(\boldsymbol{I} - \boldsymbol{Z}^S_{f,t}\right)\boldsymbol{n}^-_{b,t}, \qquad \text{and} \tag{127}$$

$$\boldsymbol{P}_{s,t} = \boldsymbol{Z}^S_{f,t}\boldsymbol{P}^+_{f,t}. \tag{128}$$

Algorithm 4 gives a compact description of the subKFBS.

5.6 Experimental evaluation of the kernel forward backward smoother

We evaluate the kernel forward backward smoother with two experiments. In the first experiment on data from a simulated pendulum, we show the performance gain of the KFBS over the KKF. In the second experiment, we apply the KFBS on data of a table tennis ball recorded with a camera-based tracking system and show how the KFBS and subKFBS are able to restore the full trajectory of the ball while only having observations at the first four and at the last time step.

5.6.1 Pendulum

We simulate a pendulum similar to the one from Sect. 5.3.1, however we initialize the pendulum in the range $[-0.25\pi, 0.25\pi]$ and with a angular velocity sampled from the range $[-2\frac{\pi}{s}, 2\frac{\pi}{s}]$. During the simulation, we apply Gaussian process noise with $\sigma = 0.01$ and as observations we use the angular displacement and add Gaussian observation noise with $\sigma = 0.2$. To learn the KFBS models, we sample 100 episodes with each 30 steps where a step corresponds to 0.1 s. The training samples are 200 windows of four observations, which we select by the kernel activation heuristic explained in Sect. 3.1. To find the optimal parameters, we apply CMA-ES (Hansen 2006) where we use the negative log-likelihood of the ground-truth to the smoothed estimate as optimality criterion. During the optimization, we use a test data set of 10 episodes, where we still observe at each time step. Later, we evaluate the smoothing performance on an evaluation data set where we do not observe at each time step but only at $t = [1{-}4, 6, 11, 16, 21, 27{-}30]$. This optimization procedure yielded better results than directly optimizing with only partial observations.

In Fig. 13, we show a qualitative comparison of the forward and the backward pass to the smoother. The results are as expected: the forward pass yields better results in the first half of the episode, and the backward pass yields better results in the second half. The smoother combines both estimates and outperforms the filter results. The smoothing can also be observed in the profiles of the standard deviation. While the variance from the filters increases at each time step without observation until the next measurement, the variance of the smoother is much smaller and only rises slightly between the observations.

In Fig. 14, we compare the performance of a standard KKF to the KFBS and the subKFBS for different kernel sizes on the same state estimation task for a simulated pendulum. The subKFBS has been learned with 300 samples in the full data set. Depicted are the median and the [0.15, 0.85]-quantiles of the MSE over 20 repetitions. The KFBS and the subKFBS clearly outperform the KKF for small kernel sizes (50, 100) and also yield better results for larger kernel sizes (i.e., 150 and 200 samples). The subKFBS yields slightly better results than the KFBS. In addition, we see from the quantiles of the MSE that the KFBS and the subKFBS have a more stable behavior in the optimization process than the KKF.

5.6.2 Tabletennis

In a second experiment, we perform smoothing on observations of a table tennis ball (Gomez-Gonzalez et al. 2016). The data set contains 54 trajectories of a table tennis ball tracked with a camera system, where each trajectory contains 51 observations which are recorded with a frequency of 100 Hz. We train the subKFBS with the data of 34 trajectories and use 10 trajectories for optimizing the parameters using CMA-ES (Hansen 2006). The remaining 10 trajectories are used for evaluating the results. For the smoothing task, the ball has been observed at the first five time steps and then again at the last time step.

Figure 15 shows qualitative examples of smoothed trajectories using the subKFBS in comparison the output of the subKKF. Here, we used data windows of size 4 and learned the models with 300 samples in the training data set and 100 samples in the subset. We optimized the regularization parameters and all bandwidths with CMA-ES (Hansen 2006). The plot shows how the subKFBS can estimate accurately the path of the ball only from observations at the beginning and at the end of the trajectory, while the subKKF diverges from the actual trajectory over time. Especially the impact position of the ball on the table can be estimated much better by the subKFBS than by the subKKF.

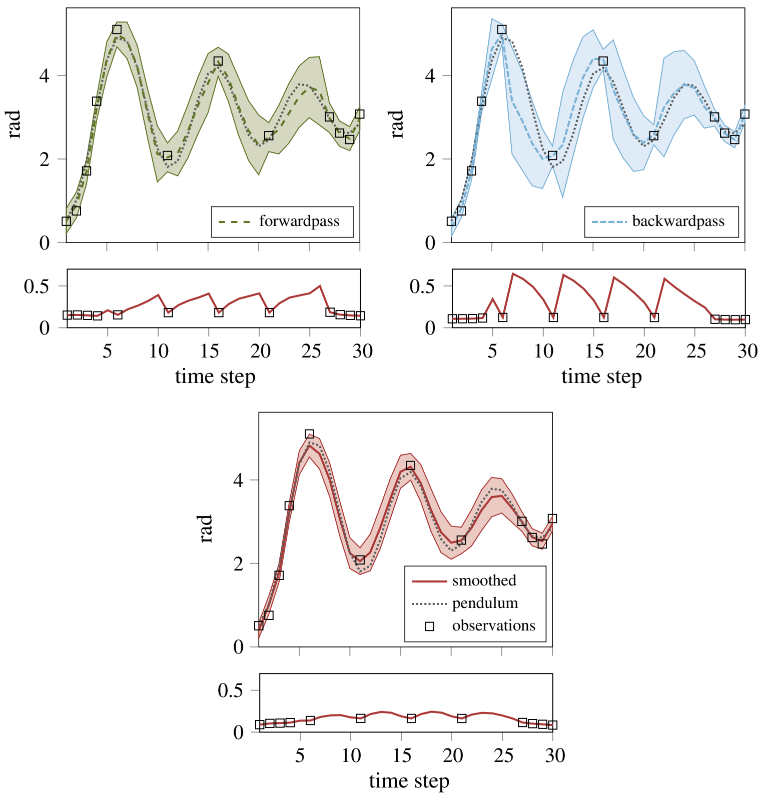


Fig. 13 Qualitative comparison of the forward and the backward pass to the smoothed estimates of the KFBS on a simulated pendulum. The upper plots show the mean and variance output of the filter/smoother, the lower plots show the profiles of the standard deviation. While the forward pass already yields good estimates in the first half of the time series, the smoother incorporates the good estimates from the backward pass in the second half and outperforms the filters. In addition, the smoother yields a more confident about its estimate

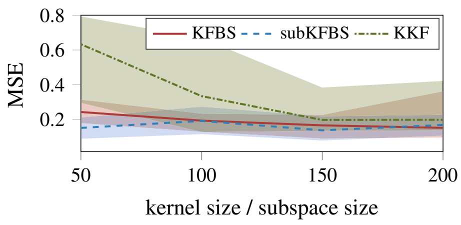


Fig. 14 The KFBS and the sub KFBS outperform the KKF clearly for small kernel sizes but also with more samples in the gram matrices. The task was to estimate the state of a pendulum from noisy partial observations. Depicted are the median and the [0.15, 0.85]-quantiles of the MSE over 20 repetitions

We also compare the KFBS to the subKFBS for different kernel sizes with the same smoothing task on recorded table tennis ball data. Figure 16 shows a comparison of the MSE, depicting the median and the [0.05, 0.95]-quantiles over 20 repetitions. The KFBS has been learned with a varying kernel size of 50, 100, 150, and 200 samples. The subKFBS uses the same number of samples to span the subspace but learns the models always with 400 samples in the full training set. While the subKFBS outperforms the KFBS for all kernel sizes, the KFBS achieves a similar performance to the KFBS when learned with 200 samples.

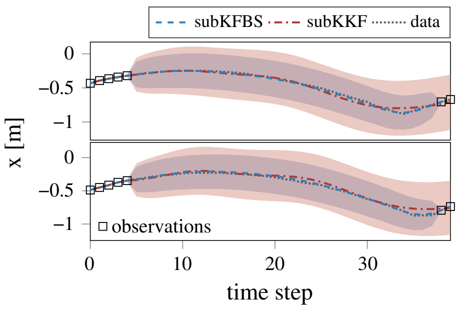


Fig. 15 In comparison to the KKF, the KFBS is able to reconstruct the trajectories of a table tennis ball from observations at the first five and at the last two time steps. The plot shows the z-coordinate of two trajectories of a table tennis ball recorded with a camera-based tracking system. We learned the subKFBS/subKKF with 100 samples in the subspace and 300 points in the training set

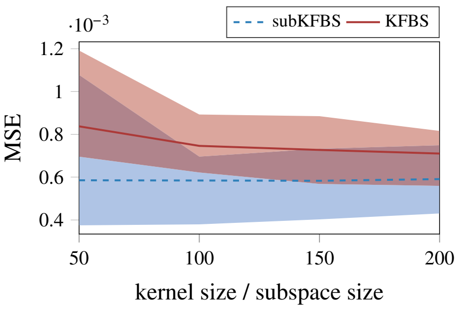


Fig. 16 The subKFBS performs better than the KFBS in the table tennis ball smoothing task. This difference in the MSE between the estimates and the noisy recorded data is more prevalent for small kernel sizes and decreases with the number of samples in the Gram matrices. Depicted is the median and the [0.05, 0.95]-quantiles of the MSE over 20 repetitions

6 Conclusion and future work

In this paper, we have presented the kernel Kalman rule (KKR) as an alternative to the kernel Bayes' rule (KBR) in the framework for nonparametric inference Song et al. (2013). In contrast to the KBR, the KKR is computationally more efficient, numerically more stable and follows from a clear optimization objective. We have further combined the KKR as Bayesian update with the kernel sum rule to formulate the kernel Kalman filter (KKF). The kernel Kalman filter can be applied to nonlinear state estimation tasks as it learns the probabilistic transition and observation dynamics as linear functions on embeddings of the belief state in high-dimensional Hilbert spaces from data. In difference to existing kernel Kalman filter formulations, the KKF also provides a more general formulation that is much closer to the original Kalman filter equations and can also be applied to partially observable systems.

While the KKF can be applied to state estimation and prediction based on past observations, we extend this work by introducing the kernel forward backward smoother (KFBS) which infers the belief state from current, past, and future information. We have shown in an experimental evaluation how this additional information leads to a performance gain of the KFBS over the KKF. As kernel methods typically scale poorly with the number of data points in the kernel matrices, we have introduced a sparsification technique that leverages from the full training set while representing the embeddings only with a small subset of the data. This technique leads to significant gains of the computational efficiency while yielding similar or even slightly better results than whithout the sparsification.

We have shown that it is possible to learn the kernel Kalman rule and other kernelized inference methods also from partial observations if sliding windows of the time series provide sufficient statistics. However in future work, we want to concentrate on learning the transition dynamics in the RKHS with an expectation-maximization algorithm in case of missing information about the latent state as we think that this leads to better models of the dynamics and also improves the accuracy of the estimated variance.

A Derivations for the subspace conditional operator

We define the subspace conditional operator $\mathcal{C}^S_{Y|X}$ as the mapping from an embedding $\varphi(\boldsymbol{x}) \in \mathcal{H}_X$ to the mean embedding $\mu_{Y|\boldsymbol{x}} \in \mathcal{H}_Y$ of the conditional distribution $P(Y|\boldsymbol{x})$ conditioned on a certain variate $\boldsymbol{x}$. To obtain this subspace conditional operator, we first introduce an auxiliary conditional operator $\mathcal{C}^{\text{aux}}_{Y|X}$ which maps from the subspace projection of the embedding $\boldsymbol{\Gamma}^\intercal \varphi(\boldsymbol{x})$ to the mean map of the conditional distribution, i.e.,

$$\hat{\mu}_{Y|\boldsymbol{x}} = \hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \varphi(\boldsymbol{x}). \tag{129}$$

We can then derive this auxiliary conditional operator by minimizing the squared error on the full data set

$$\hat{\mathcal{C}}^{\text{aux}}_{Y|X} = \arg\min_{\mathcal{C}} \left\| \boldsymbol{\Phi} - \mathcal{C}\boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \right\|_2 \tag{130}$$

$$0 = \frac{\partial}{\partial \hat{\mathcal{C}}^{\text{aux}}_{Y|X}} \left\| \boldsymbol{\Phi} - \hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \right\|_2 \tag{131}$$

$$0 = -2\left(\boldsymbol{\Phi} - \hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x\right) \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \tag{132}$$

$$\hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} = \boldsymbol{\Phi} \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \tag{133}$$

$$\hat{\mathcal{C}}^{\text{aux}}_{Y|X} = \boldsymbol{\Phi} \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \left(\boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} + \lambda \boldsymbol{I}\right)^{-1}. \tag{134}$$

We can then substitute this result for the auxiliary conditional operator in Eq. 23 and obtain the subspace conditional operator as

$$\hat{\mathcal{C}}^S_{Y|X} = \hat{\mathcal{C}}^{\text{aux}}_{Y|X} \boldsymbol{\Gamma}^\intercal \tag{135}$$

$$= \boldsymbol{\Phi} \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \left(\boldsymbol{\Gamma}^\intercal \boldsymbol{\Upsilon}_x \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Gamma}^\intercal \tag{136}$$

$$= \boldsymbol{\Phi} \boldsymbol{K}_{x\bar{x}} \left(\boldsymbol{K}_{x\bar{x}}^\intercal \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Gamma}^\intercal, \tag{137}$$

where $\boldsymbol{K}_{x\bar{x}} = \boldsymbol{\Upsilon}_x^\intercal \boldsymbol{\Gamma} \in \mathbb{R}^{n \times m}$ is the kernel matrix of the sample feature set $\boldsymbol{\Upsilon}_x$ and its subset $\boldsymbol{\Gamma}$.

B Derivations for the kernel Kalman rule and its applications

B.1 The residual of the observation operator is unbiased

We can easily show that the residual of the observation operator is unbiased by taking the expectation

$$\mathbb{E}_Y[\boldsymbol{\zeta}_t] = \mathbb{E}_Y[\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t)] \tag{138}$$

$$= \mathbb{E}_{Y,X}[\phi(\boldsymbol{y}_t) - \mathcal{C}_{Y|X}\varphi(\boldsymbol{x}_t)] \tag{139}$$

$$= \mathbb{E}_{Y,X}[\phi(\boldsymbol{y}_t)] - \mathcal{C}_{Y|X}\mu_{X,t} \tag{140}$$

$$= \mathbb{E}_X[\mathbb{E}_{Y|X}[\phi(\boldsymbol{y}_t)]] - \mathbb{E}_X[\mathbb{E}_{Y|X}[\phi(\boldsymbol{y}_t)]] = 0. \tag{141}$$

Here, we used the definition of the conditional embedding operator found in Song et al. (2013).

B.2 Derivation of the optimal Kalman gain operator

We want to find the kernel Kalman gain operator $\mathcal{Q}_t$ which minimizes the expected squared loss $\mathbb{E}\left[\left(\boldsymbol{\varepsilon}_t^+\right)^\intercal \boldsymbol{\varepsilon}_t^+\right]$ or equivalently the variance of the estimator. The objective for minimizing the variance can also be reformulated as minimizing the trace of the a-posteriori covariance operator $\mathcal{C}_{XX,t}^+$ of the state $\boldsymbol{x}_t$ at time t, i.e.,

$$\min_{\mathcal{Q}_t} \mathbb{E}\left[\left(\boldsymbol{\varepsilon}_t^+\right)^\intercal \boldsymbol{\varepsilon}_t^+\right] = \min_{\mathcal{Q}_t} \operatorname{Tr} \mathbb{E}\left[\boldsymbol{\varepsilon}_t^+ \left(\boldsymbol{\varepsilon}_t^+\right)^\intercal\right] \tag{142}$$

$$= \min_{\mathcal{Q}_t} \operatorname{Tr} \mathbb{E}\left[\left(\varphi(\boldsymbol{x}_t) - \mu_{X,t}^+\right)\left(\varphi(\boldsymbol{x}_t) - \mu_{X,t}^+\right)^\intercal\right] \tag{143}$$

$$= \min_{\mathcal{Q}_t} \operatorname{Tr} \mathcal{C}_{XX,t}^+. \tag{144}$$

By substituting the posterior error with $\boldsymbol{\varepsilon}_t^+ = (\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X})\boldsymbol{\varepsilon}_t^- - \mathcal{Q}_t\boldsymbol{\zeta}_t$, we can now rewrite this a-posteriori covariance operator as

$$\mathcal{C}_{XX,t}^+ = \mathbb{E}\left[\boldsymbol{\varepsilon}_t^+ \left(\boldsymbol{\varepsilon}_t^+\right)^\intercal\right] \tag{145}$$

$$= \mathbb{E}\left[\left(\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\boldsymbol{\varepsilon}_t^- - \mathcal{Q}_t\boldsymbol{\zeta}_t\right)\left(\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\boldsymbol{\varepsilon}_t^- - \mathcal{Q}_t\boldsymbol{\zeta}_t\right)^\intercal\right] \tag{146}$$

$$= \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\mathbb{E}\left[\boldsymbol{\varepsilon}_t^-(\boldsymbol{\varepsilon}_t^-)^\intercal\right]\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)^\intercal \tag{147}$$

$$- \mathcal{Q}_t\mathbb{E}\left[\boldsymbol{\zeta}_t(\boldsymbol{\varepsilon}_t^-)^\intercal\right]\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)^\intercal - \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\mathbb{E}\left[\boldsymbol{\varepsilon}_t^-\boldsymbol{\zeta}_t^\intercal\right]\mathcal{Q}_t^\intercal \tag{148}$$

$$+ \mathcal{Q}_t\mathbb{E}\left[\boldsymbol{\zeta}_t\boldsymbol{\zeta}_t^\intercal\right]\mathcal{Q}_t^\intercal \tag{149}$$

Since the residual of the observation operator $\boldsymbol{\zeta}_t$ is assumed to be independent from the estimation error and since we assume the a-priori estimate to be zero mean we get

$$\mathbb{E}\left[\boldsymbol{\zeta}_t(\boldsymbol{\varepsilon}_t^-)^\intercal\right] = \mathbb{E}\left[\boldsymbol{\zeta}_t\right]\mathbb{E}\left[(\boldsymbol{\varepsilon}_t^-)^\intercal\right] = \mathbb{E}\left[(\boldsymbol{\varepsilon}_t^-)^\intercal\right]\mathbb{E}\left[\boldsymbol{\zeta}_t\right] = \mathbb{E}\left[\boldsymbol{\varepsilon}_t^-\boldsymbol{\zeta}_t^\intercal\right] = 0. \tag{150}$$

With this insight the posterior covariance operator can be reformulated as

$$\mathcal{C}_{XX,t}^+ = \left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\mathcal{C}_{XX,t}^-\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)^\intercal + \mathcal{Q}_t\mathcal{R}\mathcal{Q}_t^\intercal, \tag{151}$$

where $\mathcal{R} = \mathbb{E}[\zeta_t\zeta_t^\intercal]$ is the covariance of the residual of the observation operator. Taking the derivative of the trace of the covariance operator and setting it to zero leads to the solution for the kernel Kalman gain operator as

$$0 = 2\left(\mathcal{I} - \mathcal{Q}_t\mathcal{C}_{Y|X}\right)\mathcal{C}_{XX,t}^-\left(-\mathcal{C}_{Y|X}^\intercal\right) + 2\mathcal{Q}_t\mathcal{R} \tag{152}$$

$$\mathcal{Q}_t\mathcal{C}_{Y|X}\mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal + \mathcal{Q}_t\mathcal{R} = \mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal \tag{153}$$

$$\mathcal{Q}_t\left(\mathcal{C}_{Y|X}\mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal + \mathcal{R}\right) = \mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal \tag{154}$$

$$\mathcal{Q}_t = \mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal\left(\mathcal{C}_{Y|X}\mathcal{C}_{XX,t}^-\mathcal{C}_{Y|X}^\intercal + \mathcal{R}\right)^{-1}. \tag{155}$$

B.3 Simplifying the update of the covariance operator

Following the derivations in Simon (2006), we can find a simpler formulation for the update of the covariance operator in Eq. 63. First, we substitute the kernel Kalman gain from Eq. 64 using the notation $\mathcal{U} = \left(\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X} + \mathcal{R}\right)$ which yields

$$\mathcal{C}^{+}_{XX,t} = \left(\mathcal{I} - \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\right)\mathcal{C}^{-}_{XX,t}\Big(\dots\Big)^{\intercal} + \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{R}\Big(\dots\Big)^{\intercal} \tag{156}$$

By expanding both terms we obtain

$$\begin{aligned}\mathcal{C}^{+}_{XX,t} &= \mathcal{C}^{-}_{XX,t} - \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} - \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(157)\\ &\quad + \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(158)\\ &\quad + \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{R}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(159)\end{aligned}$$

If we now combine the second and third as well as the last two terms in this equation, we arrive at

$$\begin{aligned}\mathcal{C}^{+}_{XX,t} &= \mathcal{C}^{-}_{XX,t} - 2\mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(160)\\ &\quad + \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\left(\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X} + \mathcal{R}\right)\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(161)\\ &= \mathcal{C}^{-}_{XX,t} - 2\mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} + \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{U}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(162)\\ &= \mathcal{C}^{-}_{XX,t} - \mathcal{C}^{-}_{XX,t}\mathcal{C}^{\intercal}_{Y|X}\mathcal{U}^{-1}\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(163)\\ &= \mathcal{C}^{-}_{XX,t} - \mathcal{Q}_t\mathcal{C}_{Y|X}\mathcal{C}^{-}_{XX,t} &(164)\end{aligned}$$

With this update equation for the estimator covariance, which is more concise and more similar to the original equations of the Kalman filter, we can summarize the kernel Kalman rule as follows: I. compute the kernel Kalman gain operator $\mathcal{Q}_t$ (Eq. 64), II. update the estimator of the mean embedding $\mu^{+}_{X,t}$ (Eq. 55), III. update the estimator of the covariance embedding $\mathcal{C}^{+}_{XX,t}$ (Eq. 65).

B.4 Derivations for the sample-based kernel Kalman rule

The Kalman gain operator can be rewritten with the sample estimators of mean embedding, covariance embedding and conditional embedding operator as

$$\hat{\mathcal{Q}}_t = \boldsymbol{\Upsilon}_x \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal}\boldsymbol{\Phi}^{\intercal}\left(\boldsymbol{\Phi}\boldsymbol{O}\boldsymbol{S}^{-}_t\boldsymbol{O}^{\intercal}\boldsymbol{\Phi}^{\intercal} + \kappa\mathcal{I}\right)^{-1}, \tag{165}$$

However, this formulation still has the inversion of a potentially infinite dimensional operator. To this end, we can apply the matrix identity $\boldsymbol{A}(\boldsymbol{B}\boldsymbol{A} + \boldsymbol{I})^{-1} = (\boldsymbol{A}\boldsymbol{B} + \boldsymbol{I})^{-1}\boldsymbol{A}$ to the kernel Kalman gain matrix as follows

$$\begin{aligned}\hat{\mathcal{Q}}_t &= \boldsymbol{\Upsilon}_x \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal}\underbrace{\boldsymbol{\Phi}^{\intercal}}_{A}\Big[\underbrace{\boldsymbol{\Phi}\boldsymbol{O}\boldsymbol{S}^{-}_t\boldsymbol{O}^{\intercal}}_{B}\underbrace{\boldsymbol{\Phi}^{\intercal}}_{A} + \kappa\mathcal{I}\Big]^{-1} &(166)\\ &= \boldsymbol{\Upsilon}_x \boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal}(\underbrace{\boldsymbol{\Phi}^{\intercal}}_{A}\underbrace{\boldsymbol{\Phi}\boldsymbol{O}\boldsymbol{S}^{-}_t\boldsymbol{O}^{\intercal}}_{B} + \kappa\boldsymbol{I})^{-1}\underbrace{\boldsymbol{\Phi}^{\intercal}}_{A} &(167)\\ &= \boldsymbol{\Upsilon}_x \underbrace{\boldsymbol{S}^{-}_t \boldsymbol{O}^{\intercal}(\boldsymbol{G}_{yy}\boldsymbol{O}\boldsymbol{S}^{-}_t\boldsymbol{O}^{\intercal} + \kappa\boldsymbol{I})^{-1}}_{\boldsymbol{Q}_t}\boldsymbol{\Phi}^{\intercal}, &(168)\end{aligned}$$

where we defined $\boldsymbol{Q}_t = \boldsymbol{S}_t^- \boldsymbol{O}^\top (\boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^- \boldsymbol{O}^\top + \kappa \boldsymbol{I})^{-1} \in \mathbb{R}^{n \times n}$ with the Gram matrix of the observations $\boldsymbol{G}_{yy} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$. Based on this formulation of the kernel Kalman gain operator, we can now derive finite vector/matrix formulations for the update equations of the estimator of the mean embedding (Eq. 55) and the estimator of the covariance operator (Eq. 65). For the estimator of the mean embedding, we obtain

$$\begin{aligned}
\hat{\mu}_{X,t}^+ &= \hat{\mu}_{X,t}^- + \hat{\mathcal{Q}}_t \left(\phi(\boldsymbol{y}_t) - \hat{\mathcal{C}}_{Y|X} \hat{\mu}_{X,t}^-\right) && (169)\\
\boldsymbol{\Upsilon}_x \boldsymbol{m}_t^+ &= \boldsymbol{\Upsilon}_x \boldsymbol{m}_t^- + \boldsymbol{\Upsilon}_x \boldsymbol{Q}_t \boldsymbol{\Phi}^\top \left(\phi(\boldsymbol{y}_t) - \boldsymbol{\Phi} \left(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m\right)^{-1} \boldsymbol{\Upsilon}_x^\top \boldsymbol{\Upsilon}_x \boldsymbol{m}_t^-\right), && (170)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{m}_t^- + \boldsymbol{\Upsilon}_x \boldsymbol{Q}_t \left(\boldsymbol{\Phi}^\top \phi(\boldsymbol{y}_t) - \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \left(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m\right)^{-1} \boldsymbol{\Upsilon}_x^\top \boldsymbol{\Upsilon}_x \boldsymbol{m}_t^-\right), && (171)\\
\boldsymbol{m}_t^+ &= \boldsymbol{m}_t^- + \boldsymbol{Q}_t \left(\boldsymbol{g}_{:\boldsymbol{y}_t} - \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{m}_t^-\right), && (172)
\end{aligned}$$

where $\boldsymbol{g}_{:\boldsymbol{y}_t} = \boldsymbol{\Phi}^\top \phi(\boldsymbol{y}_t)$ is the embedding of the measurement at time t. And the estimator for the covariance operator gets

$$\begin{aligned}
\hat{\mathcal{C}}_{XX,t}^+ &= \hat{\mathcal{C}}_{XX,t}^- - \boldsymbol{\Upsilon}_x \boldsymbol{Q}_t \boldsymbol{\Phi}^\top \hat{\mathcal{C}}_{Y|X} \hat{\mathcal{C}}_{XX,t}^- && (173)\\
\boldsymbol{\Upsilon}_x \boldsymbol{S}_t \boldsymbol{\Upsilon}_x^\top &= \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{\Upsilon}_x^\top - \boldsymbol{\Upsilon}_x \boldsymbol{Q}_t \boldsymbol{\Phi}^\top \boldsymbol{\Phi} \left(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m\right)^{-1} \boldsymbol{\Upsilon}_x^\top \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{\Upsilon}_x^\top && (174)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{\Upsilon}_x^\top - \boldsymbol{\Upsilon}_x \boldsymbol{Q}_t \boldsymbol{G}_{yy} \left(\boldsymbol{K}_{xx} + \lambda \boldsymbol{I}_m\right)^{-1} \boldsymbol{K}_{xx} \boldsymbol{S}_t^- \boldsymbol{\Upsilon}_x^\top && (175)\\
\boldsymbol{S}_t &= \boldsymbol{S}_t^- - \boldsymbol{Q}_t \boldsymbol{G}_{yy} \boldsymbol{O} \boldsymbol{S}_t^-. && (176)
\end{aligned}$$

B.5 Derivation of the subspace kernel Kalman gain operator/matrix

To derive the gain operator of the subspace kernel Kalman rule, we apply the subspace conditional operator

$$\hat{\mathcal{C}}_{Y|X}^S = \boldsymbol{\Phi} \boldsymbol{K}_{x\bar{x}} \left(\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Gamma}^\top, \tag{177}$$

to the kernel Kalman gain from Eq. 64 and obtain

$$\begin{aligned}
\hat{\mathcal{Q}}_t^S &= \hat{\mathcal{C}}_{XX,t}^- \left(\hat{\mathcal{C}}_{Y|X}^S\right)^\top \left(\hat{\mathcal{C}}_{Y|X}^S \hat{\mathcal{C}}_{XX,t}^- \left(\hat{\mathcal{C}}_{Y|X}^S\right)^\top + \kappa \mathcal{I}\right)^{-1} && (178)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top \underbrace{\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top}_{\boldsymbol{A}} \Big(\underbrace{\boldsymbol{\Phi} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top}_{\boldsymbol{B}} \underbrace{\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top}_{\boldsymbol{A}} + \kappa \mathcal{I} \Big)^{-1}. && (179)
\end{aligned}$$

Here, we denote $\boldsymbol{O}^S := \left(\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{K}_{x\bar{x}} + \lambda \boldsymbol{I}\right)^{-1}$. To obtain a finite dimensional matrix in the inverse, we apply again the matrix identity $\boldsymbol{A}(\boldsymbol{B}\boldsymbol{A} + \boldsymbol{I})^{-1} = (\boldsymbol{A}\boldsymbol{B} + \boldsymbol{I})^{-1}\boldsymbol{A}$ and arrive at

$$\begin{aligned}
\hat{\mathcal{Q}}_t^S &= \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top \Big(\underbrace{\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top}_{\boldsymbol{A}} \underbrace{\boldsymbol{\Phi} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top}_{\boldsymbol{B}} + \kappa \boldsymbol{I} \Big)^{-1} \underbrace{\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top}_{\boldsymbol{A}} && (180)\\
&= \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top \Big(\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{G}_{yy} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top + \kappa \boldsymbol{I} \Big)^{-1} \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top. && (181)
\end{aligned}$$

Using the projecting the subspace kernel Kalman gain into the subspace spanned by the features $\boldsymbol{\Gamma}^\top$ leads to

$$\boldsymbol{\Gamma}^\top \hat{\mathcal{Q}}_t^S = \boldsymbol{\Gamma}^\top \boldsymbol{\Upsilon}_x \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top \Big(\boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{G}_{yy} \boldsymbol{K}_{x\bar{x}} \boldsymbol{O}^S \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{S}_t^- \boldsymbol{K}_{x\bar{x}} (\boldsymbol{O}^S)^\top + \kappa \boldsymbol{I} \Big)^{-1} \boldsymbol{K}_{x\bar{x}}^\top \boldsymbol{\Phi}^\top \tag{182}$$

$$= \underbrace{\boldsymbol{P}_t^-(\boldsymbol{O}^S)^\intercal\left(\boldsymbol{K}_{x\bar{x}}^\intercal\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}_t^-(\boldsymbol{O}^S)^\intercal+\kappa\boldsymbol{I}\right)^{-1}\boldsymbol{K}_{x\bar{x}}^\intercal}_{\boldsymbol{Q}_t^S}\boldsymbol{\Phi}^\intercal, \tag{183}$$

where we define the subspace kernel Kalman gain matrix $\boldsymbol{Q}_t^S$. Using this representation, we can obtain the update equations for the subspace projections of mean embedding as

$$\boldsymbol{\Gamma}^\intercal\hat{\mu}_{X,t}^+ = \boldsymbol{\Gamma}^\intercal\left(\hat{\mu}_{X,t}^- + \hat{\mathcal{Q}}_t^S\left(\phi(\boldsymbol{y}_t) - \hat{\mathcal{C}}_{Y|X}^S\hat{\mu}_{X,t}^-\right)\right) \tag{184}$$

$$\boldsymbol{n}_{X,t}^+ = \boldsymbol{n}_{X,t}^- + \boldsymbol{Q}_t^S\left(\boldsymbol{g}_{:\boldsymbol{y}_t} - \boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{n}_t^-\right), \tag{185}$$

and similarly for the covariance operator as

$$\boldsymbol{\Gamma}^\intercal\hat{\mathcal{C}}_{XX,t}^+\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^\intercal\left(\hat{\mathcal{C}}_{XX,t}^- - \hat{\mathcal{Q}}_t^S\hat{\mathcal{C}}_{Y|X}^S\hat{\mathcal{C}}_{XX,t}^-\right)\boldsymbol{\Gamma} \tag{186}$$

$$\boldsymbol{P}_t^+ = \boldsymbol{P}_t^- - \boldsymbol{Q}_t^S\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{\Gamma}^\intercal\hat{\mathcal{C}}_{XX,t}^-\boldsymbol{\Gamma} \tag{187}$$

$$= \boldsymbol{P}_t^- - \boldsymbol{Q}_t^S\boldsymbol{G}_{yy}\boldsymbol{K}_{x\bar{x}}\boldsymbol{O}^S\boldsymbol{P}_t^- \tag{188}$$

References

Agresti, A. (2002). Deriving standard errors with the delta method. In *Categorical data analysis, Wiley series in probability and statistics* (p. 73ff). New York: Wiley.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*(3), 337–404.

Boots, B., Gretton, A., & Gordon, G. J. (2013). Hilbert space embeddings of predictive state representations. In *Proceedings of the 29th international conference on uncertainty in artificial intelligence (UAI)*.

Chen, Y., Welling, M., & Smola, A. (2010). Super-samples from kernel herding. In *Proceedings of the twenty-sixth conference on uncertainty in artificial intelligence (UAI)*. AUAI Press.

Csató, L., & Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, *14*(3), 641–668.

Drineas, P., & Mahoney, M. W. (2005). On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, *6*, 23.

Fukumizu, K., Song, L., & Gretton, A. (2013). Kernel Bayes' rule: Bayesian inference with positive definite kernels. *Journal of Machine Learning Research*, *14*(1), 3683–3719.

Gauss, C. F. (1823). *Theoria combinationis observationum erroribus minimis obnoxiae*. Göttingen: H. Dieterich.

Gebhardt, G. H. W., Kupcsik, A., & Neumann, G. (2015). Learning subspace conditional embedding operators. Workshop on large-scale kernel learning at ICML 2015.

Gebhardt, G. H. W., Kupcsik, A., & Neumann, G. (2017). The kernel kalman rule—Efficient nonparametric inference with recursive least squares. In *AAAI conference on artificial intelligence*.

Gomez-Gonzalez, S., Neumann, G., Schölkopf, B., & Peters, J. (2016). Using probabilistic movement primitives for striking movements. In *IEEE-RAS international conference on humanoid robots* (pp. 502–508).

Grünewälder, S., Lever, G., Baldassarre, L., Patterson, S., Gretton, A., & Pontil, M. (2012). Conditional mean embeddings as regressors. In *Proceedings of the 29th international conference on machine learning*.

Hansen, N. (2006). The CMA evolution strategy: A comparing review. In J. A. Lozano, P. Larrañaga, I. Inza, & E. Bengoetxea (Eds.), *Towards a new evolutionary computation. Studies in fuzziness and soft computing* (Vol. 192). Berlin, Heidelberg: Springer.

Hsu, D., Kakade, S. M., & Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, *78*(5), 1460–1480.

Jaakkola, T., Diekhans, M., & Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the international conference on intelligent systems for molecular biology*. ISMB.

Jaeger, H. (2000). Observable operator models for discrete stochastic time series. *Neural Computation*, *12*(6), 1371–1398.

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *International symposium on aerospace/defense sensing, simulation, and controls*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, *82*(1), 35–45.

Kawahara, Y., Yairi, T., & Machida, K. (2007). A kernel subspace method by stochastic realization for learning nonlinear dynamical systems. *Advances in Neural Information Processing Systems*, *19*, 665–672.

Mccalman, L., O 'callaghan, S., & Ramos, F. (2013). Multi-modal estimation with kernel embeddings for learning motion models. In *2013 IEEE international conference on robotics and automation (ICRA)* (pp. 2845–2852). Karlsruhe: IEEE.

McElhoe, B. A. (1966). An assessment of the navigation and course corrections for a manned flyby of mars or venus. In *IEEE transactions on aerospace and electronic systems AES-2*.

Nishiyama, Y., Afsharinejad, A., Naruse, S., Boots, B., & Song, L. (2016). The nonparametric kernel Bayes smoother. *International Conference on Artificial Intelligence and Statistics*, *41*, 547–555.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems* (pp. 1–8).

Ralaivola, L., & d'Alche Buc, F. (2005). Time series filtering, smoothing and learning using the kernel Kalman filter. In *Proceedings 2005 IEEE international joint conference on neural networks* (Vol. 3, pp. 1449–1454).

Schölkopf, B., Herbrich, R., Smola, A. J. (2001). A generalized representer theorem. In: D. Helmbold & B. Williamson (Eds.), *Computational learning theory (COLT), 2001*. Lecture Notes in Computer Science (Vol. 2111). Berlin, Heidelberg: Springer.

Shannon, M., Zen, H., & Byrne, W. (2013). Autoregressive models for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech and Language Processing*, *21*(3), 587–597.

Simon, D. (2006). *Optimal state estimation*. Hoboken, NJ, USA: Wiley.

Smith, G. L., Schmidt, S. F., & McGee, L. A. (1962). *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. Washington D.C.: National Aeronautics and Space Administration.

Smola, A. J., & Bartlett, P. P. (2001). Sparse greedy Gaussian process regression. *Advances in Neural Information Processing Systems*, *13*(13), 619–625.

Smola, A. J., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *Proceedings of the 18th international conference on algorithmic learning theory, lecture notes in computer science* (Vol. 4754, pp. 13–31). Berlin: Springer.

Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. Y. Weiss, B. Schölkopf, & J. C. Platt (Eds.), *Advances in neural information processing systems* (Vol. 18, pp. 1257–1264). MIT Press.

Song, L., Boots, B., Siddiqi, S. M., Gordon, G. J., & Smola, A. J. (2010). Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 991–998).

Song, L., Huang, J., Smola, A., & Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th annual international conference on machine learning—ICML'09* (pp. 1–8). New York, USA: ACM Press.

Song, L., Fukumizu, K., & Gretton, A. (2013). Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, *30*(4), 98–111.

Sorenson, H. W. (1970). Least-squares estimation: From Gauss to Kalman. *IEEE Spectrum*, *7*(7), 63–68.

Sun, W., Capobianco, R., Gordon, G. J., Bagnell, J. A., & Boots, B. (2016a). Learning to smooth with bidirectional predictive state inference machines. In *The conference on uncertainty in artificial intelligence (UAI 2016)*.

Sun, W., Venkatraman, A., Boots, B., & Bagnell, J. A. (2016b). Learning to filter with predictive state inference machines. In *International conference on machine learning* (pp. 1197–1205).

Wan, E. A. E., & Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium* (pp. 153–158). IEEE.

Williams, C. K. I., & Seeger, M. (2000). Using the Nyström method to speed up Kernel machines. In *Proceedings of the 13th international conference on neural information* (pp. 661–667).

Wojtusch, J., & von Stryk, O. (2015). Humod—A versatile and open database for the investigation, modeling and simulation of human motion dynamics on actuation level. In *IEEE-RAS international conference on humanoid robots (humanoids)*. IEEE.

Zhu, P., Chen, B., & Principe, J. C. (2014). Learning nonlinear generative models of time series with a Kalman filter in RKHS. *IEEE Transactions on Signal Processing*, *62*(1), 141–155.

Affiliations

Gregor H. W. Gebhardt[1] · **Andras Kupcsik**[2] · **Gerhard Neumann**[2,3]

Andras Kupcsik
andrasgabor.kupcsik@de.bosch.com

Gerhard Neumann
geri@robot-learning.de

1 Technische Universität Darmstadt, Computational Learning for Autonomous Systems, Hochschulstr. 10, 64285 Darmstadt, Germany

2 Bosch Center for Artificial Intelligence, Robert-Bosch-Campus 1, 71272 Renningen, Germany

3 University of Lincoln, Lincoln Centre for Autonomous Systems, Brayford Pool, LN6 7TS Lincoln, UK