# Functional Bayesian Filter

Kan Li, *Member, IEEE* and José C. Príncipe, *Life Fellow, IEEE*

***Abstract*—We present a general nonlinear Bayesian filter for high-dimensional state estimation using the theory of reproducing kernel Hilbert space (RKHS). By applying the kernel method and the representer theorem to perform linear quadratic estimation in a functional space, we derive a Bayesian recursive state estimator for a general nonlinear dynamical system in the original input space. Unlike existing nonlinear extensions of the Kalman filter where the system dynamics are assumed known, the state-space representation for the Functional Bayesian Filter (FBF) is completely learned online from measurement data in the form of an infinite impulse response (IIR) filter or recurrent network in the RKHS, with universal approximation property. Using a positive definite kernel function satisfying Mercer's conditions to compute and evolve information quantities, the FBF exploits both the statistical and time-domain information about the signal, extracts higher-order moments, and preserves the properties of covariances without the ill effects due to conventional arithmetic operations. We apply this novel kernel adaptive filtering (KAF) to recurrent network training, chaotic time-series estimation and cooperative filtering using Gaussian and non-Gaussian noises, and inverse kinematics modeling. Simulation results show FBF outperforms existing Kalman-based algorithms.**

## I. Introduction

The famed Kalman filter [1] is the optimal estimator in the minimum mean-square error (MMSE) sense when the model is linear and all latent and observed variables are jointly Gaussian. Kalman filters, predictors, and smoothers are enormously successful with a rich array of applications. However, its optimality depends on the linear structure of the underlying system, accurate knowledge of the system parameters, and the exact statistics of the noise terms.

Real-world applications in science and engineering involve nonlinear transformations. The notions of optimality and analytical or general close-form solution become intractable when dealing with such systems [2]. To overcome this limitation, several suboptimal solutions to the Bayesian filter or nonlinear extensions of the Kalman filter were developed that linearize or perform moment matching to approximate the nonlinear update, such as the extended Kalman filter (EKF) [3], [4], unscented Kalman filter (UKF) [5], [6], and the cubature Kalman filter (CKF) [7]. The EKF approximates a nonlinear system using first-order linearization. The UKF uses unscented transform for approximation. The CKF uses a third-degree spherical-radial cubature rule to compute the second-order statistics of a nonlinearly transformed Gaussian random variable [7]. Variants of particle filtering using sequential Monte Carlo (MC) approach have also been used for recursive Bayesian filtering, where the posterior density function is represented by a set of weighted random samples [8], [9]. Novel modeling and inference frameworks for dynamical systems is an active, ongoing research area [10]–[12].

Recently, several formulations were proposed, using the theory of RKHS, including the kernel Kalman filter (KKF) [13], the dynamical system model with a conditional embedding (DSMCE) operator [14], and the kernel Bayes' rule (KBR) [15]. The KKF is implemented in a high dimensional subspace obtained by the Kernel principal component analysis (KPCA) algorithm [16]. The DSMCE and KBR algorithms are both developed based on the embedding of conditional distributions in RKHS. A closely related concept, Gaussian process (GP) [17] regression models have also been used to learn prediction and observation models for dynamical systems [18], [19]. All of these generative approaches treat the time series or their feature-space mappings as the hidden states and describe the dynamics by the assumed state-space model (SSM) or the given hidden-state training data. This brings us to the second major shortcoming of the Kalman filter and its aforementioned nonlinear extensions.

Conventional Kalman-based methods require accurate knowledge of the system dynamics, while most real-world systems have unknown transformations. Therefore, accurate estimation and prediction cannot be obtained by these algorithms. For linear systems, various methods have been proposed to make state estimation robust against modeling errors such as parametric uncertainties, e.g., using minimization of an $\epsilon$-contaminated criterion [20], and stochastic measurement droppings by applying a sensitivity-penalization based recursive procedure with intermittent observations [21]. For dual estimation (estimating the state of a dynamic system and the model giving rise to the dynamics), a separate construct such as a neural network can be used as the functional form of the unknown model. This typically requires separate state-space representations for the signal and the weights, i.e., two dynamical systems, one for the evolution of the states, and the other, the evolution of the network parameters. The kernel Kalman filter based on the conditional embedding operator (KKF-CEO) [22] was developed to combat this issue. It constructs a state space model in an RKHS using the estimated conditional embedding operator and implements the Kalman filtering in this space. However, similar to the extended KRLS (Ex-KRLS) algorithm [23], the KKF-CEO is formulated using a simple additive noise model and does not utilize a fully developed state-space representation.

More recently, the kernel adaptive autoregressive-moving-average (ARMA) or KAARMA algorithm [24] was introduced to bridge the theories of adaptive signal processing and recurrent neural networks (RNNs), extending the current theory of feedforward kernel adaptive filtering (KAF) to include feedback. It is a true infinite impulse response (IIR) system

This work was supported, in part, by DARPA Contracts N66001-10-C-2008, N66001-15-1-4054, FA9453-18-1-0039, and ARO grant W911NF-21-1-0254.

The authors are with the Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL 32611 USA (e-mail: likan@ufl.edu; principe@cnel.ufl.edu).

in the RKHS, formulated with a full SSM. Unlike its finite-impulse response (FIR) counterpart, the memory depth of an IIR filter is independent of the filter order and the number of adaptive parameters, thus ideally suited for modeling dynamics characterized by a deep memory structure yet with a small number of free parameters, which ultimately yields a simpler, parsimonious network. KAARMA learns unknown general nonlinear continuous-time state-transition and measurement equations, using only an input sequence and the observed outputs. KAARMA is trained using stochastic gradient-descent and can operate on incomplete or deferred outputs for sequence learning. We have successfully applied KAARMA to model flight dynamics of insects [25] and speech using biologically-inspired spike trains [26].

In this paper, we derive a Kalman-like filter in the RKHS on the full state-space representation used in KAARMA. This is similar to training RNNs using the extended Kalman filter [27], except that the network is implemented in a functional space where classical linear methods are used (computed in the input space using the representer theorem and the *kernel trick*), and the universal approximation property of kernel method provides a general nonlinear solution in the input space. To distinguish our work from previous attempts, we name our novel algorithm the Functional Bayesian Filter (FBF). FBF is inherently an ARMA model, using joint estimation (state and model parameters are concatenated within a combined state vector, resulting in a single dynamical model to estimate both quantities simultaneously). There is no *a priori* requirement on system knowledge, as it can be trained from scratch using only observations, e.g., time series analysis, where time-delayed observations are used as inputs. To take advantage of FBF, the nonlinear SSM has to be expressed in terms of a linear filter in the RKHS, and the simplest conversion is through training. If accurate knowledge of the system is available (i.e., clean states), this will facilitate convergence during training. We will show that the FBF generalizes the KAARMA algorithm.

Lastly, the Kalman filter is MMSE-optimal only for a linear system and under the assumption that the model uncertainty (innovation) is fully described by first and second order statistics. This assumption results in degraded performances when data contains outliers or nonzero higher-order cumulants. The FBF, on the other hand, makes no assumption on the state transition model or the noise profile, by learning directly from measurement data. The fact that reproducing kernels are covariance functions explains their early role in inference problems [28], [29]. Because of the nonlinearity of the Gaussian kernel, all even moments of the random variable contribute to the estimation of a similarity measure [30]. The FBF propagates and updates the full statistics of the measurement distribution in the RKHS, not just the mean and covariance, which results in enhanced estimation in non-Gaussian noise environments. From an information theoretic learning (ITL) [30] perspective, the FBF natively computes information quantities such as correntropy and information potential (IP) to evolve the probability density function (pdf) of the data, using adaptive Parzen estimation or kernel density estimation (KDE) [31]. The versatility of the proposed FBF, under the powerful unifying framework of kernel methods [32], will be useful for a diverse set of applications in automatic control, machine learning, and signal processing. The major attributes of different well-known Kalman-based Bayesian filters are summarized in Table I.

TABLE I: Comparison of well-known Kalman-based Bayesian filters and FBF.

| Filter \ Property | Nonlinear Dynamics | Unknown Dynamics | Higher-Order Statistics |
|---|---|---|---|
| Kalman | × | × | × |
| EKF | limited | × | × |
| UKF | ✓ | × | × |
| CKF | ✓ | × | × |
| FBF | ✓ | ✓ | ✓ |

The remainder is organized as follows. In Section II, Bayesian filtering is reviewed. We introduce kernel adaptive filtering in Section III and present the proposed Functional Bayesian Filter algorithm along with its relationship to ITL. To improved the flow and readability, we leave the detailed derivation in the Appendix. Section IV presents the experimental results, comparing our novel method with several existing algorithms. Finally, Section V concludes this paper.

## II. Bayesian Filtering

For a nonlinear dynamic system, we are interested in estimating the hidden states recursively via the sequence of noisy observations or measurements dependent on the state. Bayesian filtering provides a unifying framework for solving this problem. Kalman filter is a special case with an optimal solution, assuming the system is linear and that the second-order statistics of the dynamic and observation noises are known.

Let a dynamical system (Fig. 1) be defined in terms of a general continuous nonlinear state-transition and observation functions, $\mathbf{f}(\cdot,\cdot)$ and $\mathbf{h}(\cdot)$, respectively,

$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) + \boldsymbol{w}_{i-1} \tag{1}$$

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \boldsymbol{v}_i \tag{2}$$

where

$$\mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \triangleq \left[f^{(1)}(\mathbf{x}_{i-1}, \mathbf{u}_i), \cdots, f^{(n_x)}(\mathbf{x}_{i-1}, \mathbf{u}_i)\right]^T \tag{3}$$

$$\mathbf{h}(\mathbf{x}_i) \triangleq \left[h^{(1)}(\mathbf{x}_i), \cdots, h^{(n_y)}(\mathbf{x}_i)\right]^T \tag{4}$$

with input $\mathbf{u}_i \in \mathbb{R}^{n_u}$, state $\mathbf{x}_i \in \mathbb{R}^{n_x}$, output $\mathbf{y}_i \in \mathbb{R}^{n_y}$, additive dynamic noise $\boldsymbol{w}_i$ and observation noise $\boldsymbol{v}$ are statistically independent processes of zero mean and known covariance matrices, and the parenthesized superscript $^{(k)}$ indicates the $k$-th column of a matrix or the $k$-th component of a vector. Note that the input, state, and output vectors have independent degrees of freedom or dimensionality.

Let $\hat{\mathbf{x}}_{i|i-1}$ be the *a priori* state estimate at step $i$, given knowledge of the process prior to step $i$, and $\hat{\mathbf{x}}_{i|i}$ be the *a posteriori* state estimate at step $i$, given new measurement $\mathbf{y}_i$, we can define the *a priori* and *a posteriori* estimate errors as

$$\mathbf{e}_{i|i-1} = \mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1} \tag{5}$$

$$\mathbf{e}_{i|i} = \mathbf{x}_i - \hat{\mathbf{x}}_{i|i}. \tag{6}$$

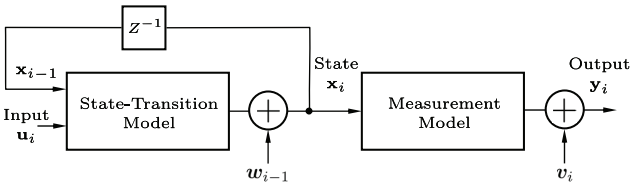



Fig. 1: General state-space model for dynamical system.

The goal is to minimize the expectations of the squared state errors, which, for unbiased estimators, is equivalent to the *a posteriori* error covariance

$$\mathbf{P}_{i|i} = \mathrm{E}[\mathbf{e}_{i|i}\mathbf{e}_{i|i}^{\mathsf{T}}] \tag{7}$$

where $\mathrm{E}[\cdot]$ denotes the expectation.

This recursive process consists of two distinct phases. The *time update* computes the predictive density

$$\begin{aligned} p(\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i) &= \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_i, \mathbf{x}_{i-1}|\mathbf{D}_{i-1}, \mathbf{u}_i)\, \mathrm{d}\mathbf{x}_{i-1} \\ &= \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) p(\mathbf{x}_{i-1}|\mathbf{D}_{i-1})\, \mathrm{d}\mathbf{x}_{i-1} \end{aligned} \tag{8}$$

where $\mathbf{D}_{i-1} = \{\mathbf{u}_j, \mathbf{y}_j\}_{j=1}^{(i-1)}$ denotes the history or sequence of input-observation pairs up to time index $(i-1)$. The *measurement update* involves computing the posterior density of the current state, which is proportional to the product of the measurement likelihood and the predicted state (Bayes' rule) as

$$\begin{aligned} p(\mathbf{x}_i|\mathbf{D}_i) &= p(\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i, \mathbf{y}_i) \\ &= \frac{p(\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i) p(\mathbf{y}_i|\mathbf{x}_i)}{p(\mathbf{y}_i|\mathbf{D}_{i-1}, \mathbf{u}_i)} \end{aligned} \tag{9}$$

where the denominator or normalizing constant is

$$p(\mathbf{y}_i|\mathbf{D}_{i-1}, \mathbf{u}_i) = \int_{\mathbb{R}^{n_x}} p(\mathbf{y}_i|\mathbf{x}_i) p(\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i)\, \mathrm{d}\mathbf{x}_i. \tag{10}$$

Under the Gaussian approximation, the predictive density $p(\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i)$, the filter likelihood density $p(\mathbf{y}_i|\mathbf{x}_i)$, and the resulting posterior density $p(\mathbf{x}_i|\mathbf{D}_i)$ are all assumed to be Gaussian, effectively reducing the recursive Bayesian estimation to operations only on the means and covariances of the conditional densities involved. The time update computes the conditional mean $\hat{\mathbf{x}}_{i|i-1}$ and the covariance $\mathbf{P}_{i|i-1}$ of the Gaussian predictive density as

$$\hat{\mathbf{x}}_{i|i-1} = \mathrm{E}[\mathbf{x}_i|\mathbf{D}_{i-1}, \mathbf{u}_i] \tag{11}$$

$$\mathbf{P}_{i|i-1} = \mathrm{E}[(\mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1})(\mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1})^{\mathsf{T}}|\mathbf{y}_{1:i-1}]. \tag{12}$$

The errors in predicted measurements are zero-mean white sequences with filter likelihood density approximated by the Gaussian as

$$p(\mathbf{y}_i|\mathbf{D}_{i-1}) = \mathcal{N}(\mathbf{y}_i; \hat{\mathbf{y}}_{i|i-1}, \mathbf{P}_{yy,i|i-1}) \tag{13}$$

where $\hat{\mathbf{y}}_{i|i-1}$ is the predicted measurement with the associated covariance $\mathbf{P}_{yy,i|i-1}$. The conditional Gaussian density of the joint state and measurement is

$$p\left(\begin{bmatrix}\mathbf{x}_i \\ \mathbf{y}_i\end{bmatrix}|\mathbf{D}_{i-1}\right) = \mathcal{N}\left(\begin{bmatrix}\hat{\mathbf{x}}_{i|i-1} \\ \hat{\mathbf{y}}_{i|i-1}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{i|i-1} & \mathbf{P}_{xy,i|i-1} \\ \mathbf{P}_{xy,i|i-1}^{\mathsf{T}} & \mathbf{P}_{yy,i|i-1}\end{bmatrix}\right) \tag{14}$$

where $\mathbf{P}_{xy,i|i-1}$ is the cross-covariance.

Upon receiving a new measurement $\mathbf{y}_i$, the Bayesian filter computes the posterior density $p(\mathbf{x}_i|\mathbf{D}_i)$ yielding

$$p(\mathbf{x}_i|\mathbf{D}_i) = \mathcal{N}(\mathbf{x}_i; \hat{\mathbf{x}}_{i|i}, \mathbf{P}_{i|i}) \tag{15}$$

where

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + \mathbf{G}_i(\mathbf{y}_i - \hat{\mathbf{y}}_{i|i-1}) \tag{16}$$

$$\mathbf{P}_{i|i} = \mathbf{P}_{i|i-1} - \mathbf{G}_i\mathbf{P}_{yy,i|i-1}\mathbf{G}_i^{\mathsf{T}} \tag{17}$$

$$\mathbf{G}_i = \mathbf{P}_{xy,i|i-1}\mathbf{P}_{yy,i|i-1}^{-1} \tag{18}$$

with $\mathbf{G}_i$ being the gain or fusion factor that minimizes the *a posteriori* error covariance $\mathbf{P}_{i|i}$. While the recursive estimation is linear, no assumption has been made on the linearity of the model. If the state-transition and observation functions are linear, under the Gaussian assumption, Bayesian filtering reduces to the Kalman filter. The Kalman filter is optimal in the sense that it minimizes the estimated error covariance. The Gaussian pdf is widely used due to its convenient mathematical properties: closed under linear transformation and conditioning, and uncorrelated jointly Gaussian random variables are independent. Although rarely do all the conditions necessary for optimality exist, the Kalman filter performs well in practice, due to its simplicity, robustness, and versatility, as the Gaussian approximates many natural random processes by the central limit theorem [7].

## III. Functional Bayesian Filtering (FBF)

In this section, we present the main contribution of the paper. First, we briefly introduce the basic concept of linear filtering in the RKHS or kernel adaptive filtering [23]. For a set of $N$ data points $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^{n_x}$ is the input and $y_i \in \mathbb{R}$ is the corresponding output signal or observation, we would like to infer the underlying function $y = \mathsf{f}(\mathbf{x})$ and predict its value or the value of a new observation $y'$, for a new input vector $\mathbf{x}'$.

From a parametric approach or weight-space view to regression, the estimated latent function $\hat{\mathsf{f}}(\mathbf{x})$ is expressed in terms of a set of parameters or weight vector $\mathbf{W} \in \mathbb{R}^{n_x}$. In the standard linear form

$$\hat{\mathsf{f}}(\mathbf{x}) = \mathbf{W}^{\mathsf{T}}\mathbf{x}. \tag{19}$$

To overcome the limited expressiveness of this model, we can project the $n_x$-dimensional input vector $\mathbf{x} \in \mathbb{U} \subseteq \mathbb{R}^{n_x}$ (where $\mathbb{U}$ is a compact input domain in $\mathbb{R}^{n_x}$) into a potentially infinite-dimensional feature space $\mathbb{F}$. Define a $\mathbb{U} \to \mathbb{F}$ mapping $\phi(\cdot)$, the linear model (19) becomes

$$\hat{\mathsf{f}}(\mathbf{x}) = \langle \boldsymbol{\Omega}, \phi(\mathbf{x}) \rangle_{\mathbb{F}} = \boldsymbol{\Omega}^{\mathsf{T}}\phi(\mathbf{x}) \tag{20}$$

where $\langle \cdot, \cdot \rangle_{\mathbb{F}} : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ is the inner product, and $\boldsymbol{\Omega}$ is a potentially infinite dimensional weight vector in the feature space, denoted by Greek letter.

Using the representer theorem [33] and the *kernel trick*, (20) can be expressed as a weighted sum of basis functions

$$\hat{\mathsf{f}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) \tag{21}$$

where $\alpha_n$ are the coefficients, $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ is a Mercer kernel corresponding to the inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathbb{F}}$, and $N$ is the number of basis functions or training samples. Note that $\mathbb{F}$ is equivalent to the RKHS $\mathcal{H}$ induced by the kernel if we identify $\phi(\mathbf{x}) = \mathcal{K}(\mathbf{x}, \cdot)$, i.e., $\mathbb{F} = \mathcal{H}$ [33]. The most commonly

used kernel for KAF is the Gaussian radial basis function (RBF)

$$\mathcal{K}_a(\mathbf{x}, \mathbf{x}') = \exp\left(-a\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{22}$$

where the kernel parameter is equivalent to $a = 1/(2\sigma^2)$ in (74). Without loss of generality, we will only consider the Gaussian RBF kernel for the rest of this paper. Mercer kernels approximate uniformly an arbitrary continuous target function to any degree of accuracy over any compact subset of the input space, i.e., the universal approximation property. The theory of RKHS allows us to represent a general nonlinear function as a set of linear weights $\mathbf{\Omega}$ in the feature space.

### *III.A. State Space Representation in the RKHS*

We can construct a nonlinear state-space model or dynamical system with linear weights in the RKHS. For simplicity, we rewrite the dynamical system equations (1-2) in terms of a new hidden state vector

$$\mathbf{s}_i \triangleq \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \\ \mathbf{h} \circ \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i) \end{bmatrix} \tag{23}$$

$$\mathbf{y}_i = \mathbf{s}_i^{(n_s - n_y + 1:n_s)} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix}}_{\mathbb{I}} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{bmatrix} \tag{24}$$

where $\mathbf{I}_{n_y}$ is an $n_y \times n_y$ identity matrix, $\mathbf{0}$ is an $n_y \times n_x$ zero matrix, and $\circ$ is the function composition operator. This augmented state vector $\mathbf{s}_i \in \mathbb{R}^{n_s}$ is formed by concatenating the output $\mathbf{y}_i$ with the original state vector $\mathbf{x}_i$, i.e., augmented state dimension $n_s = n_x + n_y$. With this rewriting, the measurement equation simplifies to a fixed selector matrix $\mathbb{I} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix}$. Note, despite the parsimonious structure of (24), there is no restriction on the measurement equation, as $\mathbf{h} \circ \mathbf{f}$ in (23) is its own set of general nonlinear equations, i.e., this is functionally equivalent to the generative model shown in Fig. 1.

Next, we define an equivalent transition function $\mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_i)$ taking as argument the new state variable $\mathbf{s}$. Using this notation, the dynamic system can be expressed in terms of the augmented state vector and input

$$\mathbf{x}_i = \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) \tag{25}$$

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) = \mathbf{h} \circ \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i). \tag{26}$$

To learn the general continuous nonlinear transition and observation functions, $\mathbf{g}(\cdot, \cdot)$ and $\mathbf{h} \circ \mathbf{g}(\cdot, \cdot)$, respectively, we apply the theory of RKHS. First, we map the augmented state vector $\mathbf{s}_i$ and the input vector $\mathbf{u}_i$ into two separate RKHSs as $\varphi(\mathbf{s}_i) \in \mathcal{H}_s$ and $\phi(\mathbf{u}_i) \in \mathcal{H}_u$, respectively. By the representer theorem, the state-space model defined by (25-26) can be expressed as the following set of weights (functions in the input space) in the joint RKHS $\mathcal{H}_{su} \triangleq \mathcal{H}_s \otimes \mathcal{H}_u$:

$$\mathbf{\Omega} \triangleq \mathbf{\Omega}_{\mathcal{H}_{su}} \triangleq \begin{bmatrix} \mathbf{g}(\cdot, \cdot) \\ \mathbf{h} \circ \mathbf{g}(\cdot, \cdot) \end{bmatrix} \tag{27}$$

where $\otimes$ is the tensor-product operator. This formulation preserves the functionalities of the separate state-transition and observation equations while consolidating them into a single set of weights in the RKHS (i.e., a single, parsimonious network), which greatly facilitates the construction of an empirical model and the adaptation of parameters.

We define the new features in the tensor-product RKHS as

$$\psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \triangleq \varphi(\mathbf{s}_{i-1}) \otimes \phi(\mathbf{u}_i) \in \mathcal{H}_{su}. \tag{28}$$

It follows that the tensor-product kernel is defined by

$$\begin{aligned} \langle \psi(\mathbf{s}, \mathbf{u}), \psi(\mathbf{s}', \mathbf{u}') \rangle_{\mathcal{H}_{su}} &\triangleq \mathcal{K}_{a_{su}}(\mathbf{s}, \mathbf{u}, \mathbf{s}', \mathbf{u}') \\ &= (\mathcal{K}_{a_s} \otimes \mathcal{K}_{a_u})(\mathbf{s}, \mathbf{u}, \mathbf{s}', \mathbf{u}') \\ &= \mathcal{K}_{a_s}(\mathbf{s}, \mathbf{s}') \cdot \mathcal{K}_{a_u}(\mathbf{u}, \mathbf{u}'). \end{aligned} \tag{29}$$

This construction has several advantages over the simple concatenation of the input $\mathbf{u}$ and the state $\mathbf{s}$. First, the tensor product of two positive definite kernels is also a positive definite kernel [32]. Second, since the adaptive filtering is performed in an RKHS using features, there is no constraint on the original input signals or the number of signals, as long as we use the appropriate reproducing kernel for each signal. Last but not least, this formulation imposes no restriction on the relationship between the signals in the original input space. This is important for signals with different representations and spatio-temporal scales, such as biological systems [24].

Finally, the kernel state-space model becomes

$$\mathbf{s}_i = \mathbf{\Omega}^T \psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \tag{30}$$

$$\mathbf{y}_i = \mathbb{I}\mathbf{s}_i. \tag{31}$$

Fig. 2 shows a simple kernel ARMA model corresponding to the SSM in (30 and 31). In general, the states $\mathbf{s}_i$ are assumed hidden, and the desired output does not need to be available at every time step, e.g., a deferred desired output value for $\mathbf{y}_i$ may only be observed at the final indexed step $i = f$ or $\mathbf{d}_f$. It has been proven that fully connected RNNs and nonlinear autoregressive exogenous (NARX) networks with a finite number of parameters are as powerful as Turing machines, i.e., universal computation devices [34], [35].

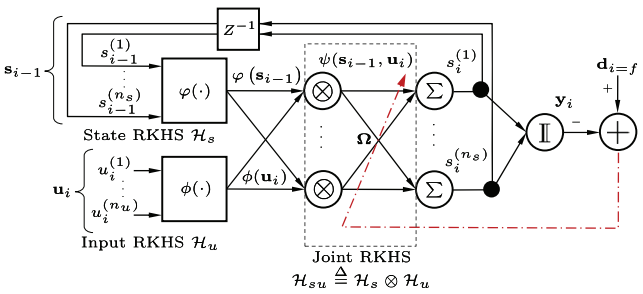



Fig. 2: ARMA model in the RKHS.

### *III.B. Bayesian filtering in the RKHS*

We can learn $\mathbf{\Omega}$, the dynamical system weights in the RKHS, using stochastic gradient descent [24]. Since the theory of RKHS allows us to use a linear form of the state-space representation to obtain nonlinear solution, we can also use Bayesian filtering to train $\mathbf{\Omega}$ and model noise statistics.

For a discrete-time linearized dynamical system described by the following state transition and measurement equations:

$$\mathbb{s}_i = \mathbf{F}_{i-1}\mathbb{s}_{i-1} + \boldsymbol{w}_{i-1} \tag{32}$$

$$\mathbf{y}_i = \mathbf{H}_i\mathbb{s}_i + \boldsymbol{v}_i. \tag{33}$$

where the super-augmented state vector is defined as

$$\mathbb{s}_i \triangleq \begin{bmatrix} \mathbf{s}_i \\ \mathbf{\Omega}_i \end{bmatrix} \tag{34}$$

with $\mathbf{s}_i = [\mathbf{x}_i, \mathbf{y}_i]^\intercal$, same as in (23), and we treat the weight matrix $\mathbf{\Omega}_i$ in the RKHS at time $i$ as an $n_{\mathbf{\Omega}}$-dimensional (potentially infinite) vector rather than a matrix, via an orderly arrangement of the weight parameters, the state transition matrix can be expressed in block form as

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{F}_1(i) & \mathbf{F}_2(i) \\ \mathbf{0} & \mathbf{I}_{n_{\mathbf{\Omega}}} \end{bmatrix} \tag{35}$$

where $\mathbf{F}_1(i)$ is an $n_s \times n_s$ matrix, $\mathbf{F}_2(i)$ is an $n_s \times n_{\mathbf{\Omega}}$ matrix, and $\mathbf{I}_{n_{\mathbf{\Omega}}}$ is an $n_{\mathbf{\Omega}} \times n_{\mathbf{\Omega}}$ identity matrix. State transition equation (32) becomes

$$\begin{bmatrix} \mathbf{s}_i \\ \mathbf{\Omega}_i \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1(i) & \mathbf{F}_2(i) \\ \mathbf{0} & \mathbf{I}_{n_{\mathbf{\Omega}}} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{i-1} \\ \mathbf{\Omega}_i \end{bmatrix} + \boldsymbol{w}_{i-1}. \tag{36}$$

Since the network output $\mathbf{y}_i$ is a subvector of the hidden state $\mathbf{s}_i$, the measurement function $\mathbf{H}_i$ in (33) becomes a simple projection onto the last $n_y$ components of $\mathbf{s}_i$

$$\mathbf{y}_i = \mathbf{H} \begin{bmatrix} \mathbf{s}_i \\ \mathbf{\Omega}_i \end{bmatrix} + \boldsymbol{v}_i \tag{37}$$

where

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbb{I} & \mathbf{0} \end{bmatrix} \tag{38}$$

with $\mathbb{I} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix} \in \mathbb{R}^{n_y \times n_s}$ being a fixed selector matrix.

From the state-space model in (36), the state transition matrix block in (35) consists of the following Jacobian matrices (similar to EKF linearization)

$$\mathbf{F}_1(i) = \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \tag{39}$$

and

$$\mathbf{F}_2(i) = \frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}_i}. \tag{40}$$

Using the representer theorem, RKHS weights $\mathbf{\Omega}_i$ at time $i$ (where $i \geq 1$) can be written as a linear combination of prior features

$$\mathbf{\Omega}_i = \mathbf{\Psi}_i \mathbf{A}_i \tag{41}$$

where $\mathbf{\Psi}_i \triangleq [\psi(\mathbf{s}_{-1}, \mathbf{u}_0), \cdots, \psi(\mathbf{s}_{i-2}, \mathbf{u}_{i-1})] \in \mathbb{R}^{n_{\mathbf{\Omega}} \times i}$ is a collection of the $i$ past tensor-product features (the first feature $\mathbf{\Psi}_1 = [\psi(\mathbf{s}_{-1}, \mathbf{u}_0)]$ is typically seeded randomly), and $\mathbf{A}_i \triangleq [\boldsymbol{\alpha}_{i,1}, \cdots, \boldsymbol{\alpha}_{i,n_s}] \in \mathbb{R}^{i \times n_s}$ is the set of coefficients with column vector $\boldsymbol{\alpha}_{i,k} \in \mathbb{R}^i$ corresponding to the $k$-th state dimension ($1 \leq k \leq n_s$). Thus, each of the $k$-th state component of the filter weights at time step $i$ becomes

$$\mathbf{\Omega}_i^{(k)} = \mathbf{\Psi}_i \mathbf{A}_i^{(k)} = \mathbf{\Psi}_i \boldsymbol{\alpha}_{i,k} \tag{42}$$

which corresponds to a general nonlinear function in the input space.

Since the hidden states are propagated using linear operator in the RKHS, i.e., $\mathbf{s}_i = \mathbf{\Omega}_i^\intercal \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)$, using the representer theorem (41), we can compute $\mathbf{F}_1(i)$ in the input space as

$$\begin{aligned} \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} &= \frac{\partial \mathbf{\Omega}_i^\intercal \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{s}_{i-1}} \\ &= \mathbf{A}_i^T \frac{\partial \mathbf{\Psi}_i^\intercal \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{s}_{i-1}} \\ &= \underbrace{2a_s \mathbf{A}_i^\intercal \mathbf{K}_i \mathbf{D}_i^\intercal}_{\boldsymbol{\Lambda}_i} \end{aligned} \tag{43}$$

where the partial derivative is evaluated using Gaussian RBF tensor-product kernel, and

$$\mathbf{K}_i \triangleq \operatorname{diag}(\mathbf{\Psi}_i^\intercal \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)) \tag{44}$$

is an $i \times i$ diagonal matrix with eigenvalues $\mathbf{K}_i^{(j,j)} = \mathcal{K}_{a_s}(\mathbf{s}_{j-2}, \mathbf{s}_{i-1}) \cdot \mathcal{K}_{a_u}(\mathbf{u}_{j-1}, \mathbf{u}_i)$, where $1 \leq j \leq i$, and $\mathbf{D}_i \triangleq [(\mathbf{s}_{-1} - \mathbf{s}_{i-1}), \cdots, (\mathbf{s}_{i-2} - \mathbf{s}_{i-1})] \in \mathbb{R}^{n_s \times i}$ is the difference matrix between state centers of the filter and the current input state variable $\mathbf{s}_{i-1}$, as a result of applying the chain rule to the state-variable Gaussian RBF kernel. We collect the terms into an $n_s \times n_s$ matrix

$$\boldsymbol{\Lambda}_i \triangleq \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} = 2a_s \mathbf{A}_i^\intercal \mathbf{K}_i \mathbf{D}_i^\intercal \tag{45}$$

which we call the state-transition gradient, where each entry $(1 \leq l, m \leq n_s)$ can be expressed as

$$\begin{aligned} \boldsymbol{\Lambda}_i^{(l,m)} &= 2a_s \sum_{j=1}^{i} \underbrace{\boldsymbol{\alpha}_{i,l}^{(j)} \left( \mathbf{s}_{j-2}^{(m)} - \mathbf{s}_{i-1}^{(m)} \right)}_{\mathbb{a}_{i,l}^{(j,m)}} \mathbf{K}_i^{(j,j)} \\ &= 2a_s \sum_{j=1}^{i} \mathbb{a}_{i,l}^{(j,m)} \mathcal{K}_{a_s}(\mathbf{s}_{j-2}, \mathbf{s}_{i-1}) \cdot \mathcal{K}_{a_u}(\mathbf{u}_{j-1}, \mathbf{u}_i) \end{aligned} \tag{46}$$

which has an information theoretic interpretation as a weighted Parzen estimate of the joint input data $(\mathbf{s}_i, \mathbf{u}_i)$ pdf.

The second block is computed as

$$\mathbf{F}_2(i) = \mathbf{1}_{n_s} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \tag{47}$$

where $\mathbf{1}_{n_s} \in \mathbf{R}^{n_s \times 1}$ is a vector of all ones. The estimated state covariance matrix is given by

$$\mathbf{P}_{i|i-1} = \mathbf{F}_i \mathbf{P}_{i-1|i-1} \mathbf{F}_i^\intercal + \mathbf{Q}_{i-1} \tag{48}$$

where

$$\mathbf{Q}_i = \mathbf{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^\intercal\right] \tag{49}$$

is the process noise covariance matrix. In general, $\mathbf{Q}$ will contain non-zero off-diagonal entries (correlated state variables), however, for simplicity, we will assume diagonal matrices for the noise covariance matrices (unless provided) and, similarly, initialize an unknown state covariance $\mathbf{P}$ as a diagonal matrix for simplicity.

The block structure of the state transition matrix $\mathbf{F}$ yields the following estimated state covariance matrix decomposition

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 \\ \mathbf{P}_3 & \mathbf{P}_4 \end{bmatrix} \tag{50}$$

where $\mathbf{P} \in \mathbb{R}^{(n_s + n_{\mathbf{\Omega}}) \times (n_s + n_{\mathbf{\Omega}})}$, $\mathbf{P}_1 \in \mathbb{R}^{n_s \times n_s}$, and $\mathbf{P}_4 \in \mathbb{R}^{n_{\mathbf{\Omega}} \times n_{\mathbf{\Omega}}}$ are symmetric, with $\mathbf{P}_2 \in \mathbb{R}^{n_s \times n_{\mathbf{\Omega}}} = \mathbf{P}_3^\intercal$. Substituting (50) and (35) into (48) yields

$$\begin{bmatrix} \mathbf{P}_1(i|i-1) & \mathbf{P}_2(i|i-1) \\ \mathbf{P}_3(i|i-1) & \mathbf{P}_4(i|i-1) \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1(i) & \mathbf{F}_2(i) \\ \mathbf{0} & \mathbf{I}_{n_{\mathbf{\Omega}}} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1(i-1|i-1) & \mathbf{P}_2(i-1|i-1) \\ \mathbf{P}_3(i-1|i-1) & \mathbf{P}_4(i-1|i-1) \end{bmatrix} \begin{bmatrix} \mathbf{F}_1^\intercal(i) & \mathbf{0} \\ \mathbf{F}_2^\intercal(i) & \mathbf{I}_{n_{\mathbf{\Omega}}} \end{bmatrix} + \begin{bmatrix} \sigma_s^2 \mathbf{I}_{n_s} & \mathbf{0} \\ \mathbf{0} & \sigma_{\mathbf{\Omega}}^2 \mathbf{I}_{n_{\mathbf{\Omega}}} \end{bmatrix}. \tag{51}$$

Using the superscripts $^-$ and $^+$ as shorthands for the *a priori* estimate $(i|i-1)$ and *a posteriori* estimates $(i-1|i-1)$ or $(i|i)$, where appropriate, we obtain the following update rules:

$$\mathbf{P}_1^- = \left[\mathbf{F}_1\mathbf{P}_1^+ + \mathbf{F}_2(\mathbf{P}_2^+)^\intercal\right]\mathbf{F}_1^\intercal + \underbrace{\left[\mathbf{F}_1\mathbf{P}_2^+ + \mathbf{F}_2\mathbf{P}_4^+\right]}_{\mathbf{P}_2^-}\mathbf{F}_2^\intercal + \sigma_s^2\mathbf{I}_{n_s} \quad (52)$$

$$\mathbf{P}_2^- = \mathbf{F}_1\mathbf{P}_2^+ + \mathbf{F}_2\mathbf{P}_4^+ \quad (53)$$

$$\mathbf{P}_3^- = \mathbf{P}_2^{-\intercal} \quad (54)$$

$$\mathbf{P}_4^- = \mathbf{P}_4^+ + \sigma_{\mathbf{\Omega}}^2\mathbf{I}_{n_{\mathbf{\Omega}}}. \quad (55)$$

The Kalman gain $\mathbb{K}$, using the innovation covariance, is

$$\mathbb{S}_i = \mathbf{H}\mathbf{P}^-\mathbf{H}^\intercal + \mathbf{R} = \mathbf{H}\mathbf{P}^-\mathbf{H}^\intercal + \sigma_y^2\mathbf{I}_{n_y} \quad (56)$$

where $\mathbf{R}$ and $\sigma_y$ are the measurement covariance matrix and output standard deviation, respectively, yielding

$$\begin{aligned}
\mathbb{K}_i &= \mathbf{P}^-\mathbf{H}_i^\intercal\mathbb{S}_i^{-1} \\
&= \begin{bmatrix}\mathbf{P}_1^- & \mathbf{P}_2^-\\ \mathbf{P}_3^- & \mathbf{P}_4^-\end{bmatrix}\begin{bmatrix}\mathbb{I}^\intercal\\ \mathbf{0}\end{bmatrix}\left(\begin{bmatrix}\mathbb{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{P}_1^- & \mathbf{P}_2^-\\ \mathbf{P}_3^- & \mathbf{P}_4^-\end{bmatrix}\begin{bmatrix}\mathbb{I}^\intercal\\ \mathbf{0}\end{bmatrix} + \sigma_y^2\mathbf{I}_{n_y}\right)^{-1} \\
&= \begin{bmatrix}\mathbf{P}_1^-\mathbb{I}^\intercal\\ \mathbf{P}_3^-\mathbb{I}^\intercal\end{bmatrix}\left(\mathbb{I}\mathbf{P}_1^-\mathbb{I}^\intercal + \sigma_y^2\mathbf{I}_{n_y}\right)^{-1} \\
&= \begin{bmatrix}\mathbf{P}_1^-\mathbb{I}^\intercal\\ (\mathbf{P}_2^-)^\intercal\mathbb{I}^\intercal\end{bmatrix}\left(\mathbb{I}\mathbf{P}_1^-\mathbb{I}^\intercal + \sigma_y^2\mathbf{I}_{n_y}\right)^{-1} \\
&\triangleq \begin{bmatrix}\mathbf{L}_1\\ \mathbf{L}_2\end{bmatrix}\left(\mathbf{M}_i + \sigma_y^2\mathbf{I}_{n_y}\right)^{-1} \\
&\triangleq \begin{bmatrix}\mathbf{L}_1\\ \mathbf{L}_2\end{bmatrix}\mathbf{N}_i
\end{aligned} \quad (57)$$

where, for clarity, we defined the following matrices

$$\mathbf{L}_1 \triangleq \mathbf{P}_1^-\mathbb{I}^\intercal \quad (58)$$

$$\mathbf{L}_2 \triangleq (\mathbf{P}_2^-)^\intercal\mathbb{I}^\intercal \quad (59)$$

$$\mathbf{M} \triangleq \mathbb{I}\mathbf{L}_1 \quad (60)$$

$$\mathbf{N} \triangleq (\mathbf{M} + \sigma_y^2\mathbf{I}_{n_y})^{-1} \quad (61)$$

with $\mathbf{L}_1 \in \mathbb{R}^{n_s\times n_y}$ (last $n_y$ columns of $\mathbf{P}_1^-$), $\mathbf{L}_2 \in \mathbb{R}^{n_{\mathbf{\Omega}}\times n_y}$ (last $n_y$ rows of $\mathbf{P}_2^-$, transposed), and $\mathbf{M}_i \in \mathbb{R}^{n_y\times n_y}$ (the $n_y \times n_y$ lower-right corner of $\mathbf{P}_1^-$) being submatrices of the decomposed state-covariance matrix $\mathbf{P}$, since the linear mappings $\mathbf{H}$ and $\mathbb{I}$ are defined in (38) as simple projections onto the last $n_y$ coordinates of state $\mathbf{s}_i$, and $\mathbf{N}$ is the inverse of the innovation covariance matrix.

Clearly, the Kalman gain matrix consists of two parts

$$\mathbb{K}_i = \begin{bmatrix}\mathbb{K}_1\\ \mathbb{K}_2\end{bmatrix} = \begin{bmatrix}\mathbf{L}_1\mathbf{N}_i\\ \mathbf{L}_2\mathbf{N}_i\end{bmatrix} \quad (62)$$

where $\mathbb{K}_1 \in \mathbb{R}^{n_s\times n_y}$ describes the changes in network activity (states $s_i$), and $\mathbb{K}_2 \in \mathbb{R}^{n_{\mathbf{\Omega}}\times n_y}$ corresponds to weight $\mathbf{\Omega}_i$ changes in response to errors.

Updating the *a posteriori* state estimate gives

$$\begin{aligned}
\mathbb{s}^+ &= \mathbb{s}^- + \mathbb{K}_i\mathbf{e}_i \\
\begin{bmatrix}\mathbf{s}\\ \mathbf{\Omega}\end{bmatrix}^+ &= \begin{bmatrix}\mathbf{s}\\ \mathbf{\Omega}\end{bmatrix}^- + \begin{bmatrix}\mathbb{K}_1\\ \mathbb{K}_2\end{bmatrix}\mathbf{e}_i.
\end{aligned} \quad (63)$$

Updating the *a posteriori* covariance estimate gives

$$\begin{aligned}
\mathbf{P}^+ &= (\mathbf{I} - \mathbb{K}\mathbf{H})\mathbf{P}^- \\
\begin{bmatrix}\mathbf{P}_1^+ & \mathbf{P}_2^+\\ \mathbf{P}_3^+ & \mathbf{P}_4^+\end{bmatrix} &= \left(\mathbf{I} - \begin{bmatrix}\mathbb{K}_1\\ \mathbb{K}_2\end{bmatrix}\begin{bmatrix}\mathbb{I} & \mathbf{0}\end{bmatrix}\right)\begin{bmatrix}\mathbf{P}_1^- & \mathbf{P}_2^-\\ \mathbf{P}_3^- & \mathbf{P}_4^-\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{P}_1^- & \mathbf{P}_2^-\\ \mathbf{P}_3^- & \mathbf{P}_4^-\end{bmatrix} - \begin{bmatrix}\mathbb{K}_1\\ \mathbb{K}_2\end{bmatrix}\begin{bmatrix}\mathbb{I}\mathbf{P}_1^- & \mathbb{I}\mathbf{P}_2^-\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{P}_1^- & \mathbf{P}_2^-\\ \mathbf{P}_3^- & \mathbf{P}_4^-\end{bmatrix} - \begin{bmatrix}\mathbb{K}_1\mathbf{L}_1^\intercal & \mathbb{K}_1\mathbf{L}_2^\intercal\\ \mathbb{K}_2\mathbf{L}_1^\intercal & \mathbb{K}_2\mathbf{L}_2^\intercal\end{bmatrix}.
\end{aligned} \quad (64)$$

Specifically, measurement updates are given by

$$\mathbf{s}^+ = \mathbf{s}^- + \mathbb{K}_1\mathbf{e} \quad (65)$$

$$\mathbf{\Omega}^+ = \mathbf{\Omega}^- + \mathbb{K}_2\mathbf{e} \quad (66)$$

$$\mathbf{P}_1^+ = \mathbf{P}_1^- - \mathbb{K}_1\mathbf{L}_1^\intercal \quad (67)$$

$$\mathbf{P}_2^+ = \mathbf{P}_2^- - \mathbb{K}_1\mathbf{L}_2^\intercal \quad (68)$$

$$\mathbf{P}_4^+ = \mathbf{P}_4^- - \mathbb{K}_2\mathbf{L}_2^\intercal \quad (69)$$

The covariance blocks are initialized as follows

$$\mathbf{P}_1(0) = \sigma_s^2\mathbf{I}_{n_s} \quad (70)$$

$$\mathbf{P}_4(0) = \sigma_{\mathbf{\Omega}}^2\mathbf{I}_{n_{\mathbf{\Omega}}} \quad (71)$$

$$\mathbf{P}_2(0) = \mathbf{0} \quad (72)$$

where diagonal matrices are used for simplicity, and the initial state is assumed to be independent of the filter weights. The functional Bayesian filtering algorithm is summarized in Algorithm 1. Detailed derivation is presented in the Appendix.

Functional Bayesian filtering requires the state-space model to be expressed as a linear model in the RKHS (using the representer theorem, as a finite linear combination of kernel products evaluated on the training data). This can be learned online directly from observations, from either known or unknown system dynamics. Once the generative model, in terms of the kernel filter weights $\mathbf{\Omega}$, is obtained or fixed (after training), Algorithm 1 reduces to just the state update, with state $\mathbb{s}_i = \mathbf{s}_i$ in (34), state transition equation $\mathbf{F} = \mathbf{F}_1$ in (32), and covariance $\mathbf{P} = \mathbf{P}_1$ in (50).

### *III.C. Complexity*

For conventional online kernel adaptive filters, the number of basis functions grows linearly with each new sample. The FBF memory and computational complexities for each recursive update are $O(N)$ and $O(N^2)$. Unlike gradient descent learning in KAARMA [24], only a single state vector is produced per update, in the current trajectory. Nonetheless, quadratic complexity can be quite prohibitive, especially for continuous tracking. We can reduce the resource requirements by limiting or sparsifying $N$ the number of bases used. Any online kernel method lends itself naturally to sparsification techniques as data samples are evaluated on an individual basis. We have developed several techniques to curb the growth of similar networks, including novelty and surprise-based criteria or a combination of both [24], [36]. This is an active research area and beyond the scope of this paper.

**Algorithm 1** Functional Bayesian Filter

**Initialization:**

$n_u$: input dimension
$n_y$: output dimension
$n_s$: state dimension
$a_u$: input kernel parameter
$a_s$: state kernel parameter
$\sigma_s^2$: state variance
$\sigma_{\mathbf{\Omega}}^2$: weight variance
$\sigma_y^2$: output variance
$\mathbf{P}_1(0) = \sigma_s^2 \mathbf{I}_{n_s}$
$\mathbf{P}_4(0) = \sigma_{\mathbf{\Omega}}^2 \mathbf{I}_{n\mathbf{\Omega}}$
$\mathbf{P}_2(0) = \mathbf{0}$
Randomly initialize input $\mathbf{u}_0 \in \mathbb{R}^{n_u}$
Randomly initialize states $\mathbf{s}_{-1}$ and $\mathbf{s}_0 \in \mathbb{R}^{n_s}$
Randomly initialize coefficient matrix $\mathbf{A} \in \mathbb{R}^{1 \times n_s}$
$\mathbf{\Psi} = [\psi(\mathbf{s}_{-1}, \mathbf{u}_0)]$: initial feature matrix
$\mathbf{S} = [\mathbf{s}_{-1}]$: initial state dictionary
$\mathbb{I} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix} \in \mathbb{R}^{n_y \times n_s}$: measurement matrix

**for** $i = 1, \cdots$ **do**

**Predict:**

Get current input $\mathbf{u}_i$ and past state $\mathbf{s}_{i-1}$
Propagate *a priori* state estimate

$$\mathbf{s}_i^- = \mathbf{\Omega}_i^\intercal \psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \quad \text{see (30)}$$

Compute state transition dynamics:

$$\mathbf{F}_1(i) = \underbrace{2a_s \mathbf{A}_i^\intercal \mathbf{K}_i \mathbf{D}_i^\intercal}_{\mathbf{\Lambda}_i} \quad (43)$$

$$\mathbf{F}_2(i) = \mathbf{1}_{n_s} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \quad (47)$$

Propagate *a priori* covariance estimate:

$$\mathbf{P}_1^- = \left[\mathbf{F}_1 \mathbf{P}_1^+ + \mathbf{F}_2 (\mathbf{P}_2^+)^\intercal\right] \mathbf{F}_1^\intercal + \underbrace{\left[\mathbf{F}_1 \mathbf{P}_2^+ + \mathbf{F}_2 \mathbf{P}_4^+\right]}_{\mathbf{P}_2^-} \mathbf{F}_2^\intercal + \sigma_s^2 \mathbf{I}_{n_s} \quad (52)$$

$$\mathbf{P}_4^- = \mathbf{P}_4^+ + \sigma_{\mathbf{\Omega}}^2 \mathbf{I}_{n\mathbf{\Omega}} \quad (55)$$

**Update:**

Obtain innovation or measurement residual

$$\mathbf{e}_i = \mathbf{d}_i - \mathbf{y}_i$$

Compute Kalman gain:

$$\mathbf{L}_1 \leftarrow \text{last } n_y \text{ columns of } \mathbf{P}_1^- \quad (58)$$

$$\mathbf{L}_2 \leftarrow \text{last } n_y \text{ columns of } (\mathbf{P}_2^-)^\intercal \quad (59)$$

$$\mathbf{M} \leftarrow \text{the } n_y \times n_y \text{ lower-right corner of } \mathbf{P}_1^- \quad (60)$$

$$\mathbf{N} \leftarrow (\mathbf{M} + \sigma_y^2 \mathbf{I}_{n_y})^{-1} \quad (61)$$

$$\mathbb{K}_1 \leftarrow \mathbf{L}_1 \mathbf{N} \quad (62)$$

$$\mathbb{K}_2 \leftarrow \mathbf{L}_2 \mathbf{N} \quad (62)$$

Update *a posteriori* estimates:

$$\mathbf{s}^+ = \mathbf{s}^- + \mathbb{K}_1 \mathbf{e} \quad (65)$$

$$\mathbf{\Omega}^+ = \mathbf{\Omega}^- + \mathbb{K}_2 \mathbf{e} \quad (66)$$

$$\mathbf{P}_1^+ = \mathbf{P}_1^- - \mathbb{K}_1 \mathbf{L}_1^\intercal \quad (67)$$

$$\mathbf{P}_2^+ = \mathbf{P}_2^- - \mathbb{K}_1 \mathbf{L}_2^\intercal \quad (68)$$

$$\mathbf{P}_4^+ = \mathbf{P}_4^- - \mathbb{K}_2 \mathbf{L}_2^\intercal \quad (69)$$

**end for**

### *III.D. The Kernel Advantage*

This paper's major contribution is a novel formulation under a unifying framework that tackles all three major shortcomings of classic Kalman filter: linearity, prior knowledge of accurate system parameters, and lower-order innovation statistics.

Kernel methods have additional properties that make them especially appealing for real-world applications. For example, we can exploit the structure of the RKHS to our advantage as we already demonstrated with the nearest instance centroid estimation (NICE) approach [37]. In fact, the representer theorem yields an input-output function that is a sum of terms centered at the data samples. We can partition this large sum into quasi-orthogonal sub-sums, simplifying the process and allowing the design of novel input-output maps by simple concatenation, which opens the door for fast search procedure to compose on-the-fly new filters using ideas from transfer learning, as required for nonstationary environments.

For Bayesian filtering, it is crucial that the two basic properties of an error covariance are preserved in each iteration, namely symmetry and positive definiteness. In practice, due to numerical errors by arithmetic operations, these two properties are often lost or destroyed and a square-root formulation is required [38]. Kernel methods, on the other hand, are immune to these problems due to the use of positive definite kernel function satisfying Mercer's conditions.

Furthermore, as hinted in Sec. III.A., kernel methods have the added advantage of preserving the learning algorithm regardless of input type (numerical or nonnumerical), by selecting the appropriate reproducing kernel. This formulation imposes no restriction on the representation or relationship between the input signals, e.g., we can model a biological system, taking spike trains, continuous amplitude local field potentials (LFPs), and vectorized state variables as inputs. As a proof of concept, we implemented KAARMA for biologically-inspired spike input in [26].

### *III.E. Relationship to Information Theoretic Learning (ITL)*

The aesthetics of Kalman filtering lies in its parsimonious form, by considering only the uncertainty represented in the covariance. It is the optimal linear MSE filter. In developing the functional Bayesian filter, we see that higher-order moments are automatically preserved such that the estimation of the state uses all statistical information contained in the data. Here, we elucidate the properties of ITL, its connection to FBF, and the performance boost.

ITL is a framework to adapt nonparametric systems using information quantities such as entropy and divergence [30]. ITL criterion is still directly estimated from the data via Parzen kernel estimator, but it extracts more information from the data for adaptation, and therefore yields solutions that are more accurate than MSE in non-Gaussian and nonlinear signal processing.

A more general form of correntropy (cross-correntropy) between two random variables is defined as

$$V_\sigma(X, Y) \triangleq \mathrm{E}[G_\sigma(X - Y)] \quad (73)$$

where $G_\sigma$ is the normalized Gaussian kernel

$$G_\sigma(x - x_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right) \tag{74}$$

with kernel size or bandwidth parameter $\sigma > 0$.

The sample estimate of correntropy for a finite number of data $\{x_i, y_i\}_{i=1}^N$ is

$$\hat{V}_{N,\sigma}(X, Y) = \frac{1}{N} \sum_{i=1}^{N} G_\sigma(x_i - y_i). \tag{75}$$

Using Taylor series expansion for the Gaussian kernel, correntropy can be expressed as

$$V_\sigma(X, Y) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^n \sigma^{2n} n!} \mathrm{E}[(X - Y)^{2n}] \tag{76}$$

which involves all the even-order moments of the random variable $X - Y$ (where the kernel choice dictates the expansion, e.g., the Laplacian kernel contains all the moments) [30].

In fact, all learning algorithms that use nonparametric pdf estimates in the input space admit an alternative formulation as kernel methods expressed in terms of inner products. As shown above, the kernel techniques are able to extract higher-order statistics of the data that should lead to performance improvements for non-Gaussian environments. A major limitation of conventional statistical measures is the i.i.d. assumption, because most practical problems involve some correlation or temporal structure. Therefore, most statistical methods are not using all the available information in the case of temporally correlated (non-white) input signals. Unlike conventional measures, the generalized correlation function effectively exploits both the statistical and the temporal information about the input signal. The maximum correntropy Kalman filter (MCKF) was proposed to improve the robustness of Kalman filter against impulsive noises in [39], which adopts the robust maximum correntropy criterion (MCC) as the optimality criterion, instead of the minimum mean-square-error (MMSE). This useful feature is intrinsic in the FBF formulation, even without explicitly defining ITL cost functions (please refer to the detailed derivation of $\mathbf{P}_2^-$ in the Appendix).

### *III.F. Shortcomings of FBF*

As already mentioned earlier, unlike its input-space counterpart, the computational cost of kernel methods is a function of the number of training instances, not the dimensionality of the input. Although the Gaussian RBF kernel yields reasonably good results under the general assumption of smoothness (especially in the absence of expert knowledge on the data or task domain), performance will depend on the choice of kernel and its parameters for individual applications. In the RKHS, the metric of similarity is defined by both the kernel and its parameters, e.g., the Gaussian bandwidth. The same input data can be mapped to vastly different functionals depending on the kernel bandwidth selected. Some well-known techniques for bandwidth selection include cross-validation, nearest neighbors, penalizing functions, and plug-in methods [40]. For unknown dynamical systems, the dimensionality of the hidden state is another hyperparameter. However, since the state space is continuous-valued, we have found empirically that a small dimension of 3 or 4 provides sufficient diversity to learn an equivalent attractor for many synthetic and real dynamics [24]–[26]. These and many others, such as the inverse mapping from the RKHS back to the input space (the pre-image problem [41]) which would further simplify the formulation and improve estimation accuracy, are all open problems that require further theoretical analysis.

## IV. Experiments and Results

Here, we illustrate and evaluate the proposed Functional Bayesian Filter using numerical examples. As a proof-of-concept, we consider the following tasks: recurrent network training, chaotic time-series estimation and cooperative filtering using Gaussian and non-Gaussian noises, and inverse kinematics modeling.

### *IV.A. Cooperative Filtering for Signal Enhancement*

First, we consider the scenario of an unknown nonlinear system with only noisy observations available, and compare the denoising performance (signal enhancement) of the functional Bayesian filter with that of the cubature Bayesian filter in training a dynamical system in the form of a recurrent neural network on the Mackey-Glass (MG) chaotic time series $y_t$ [42], defined by the following delay differential equation

$$\frac{dy_t}{dt} = \frac{\beta y_{(t-\tau)}}{1 + y_{(t-\tau)}^n} - \gamma y_t$$

where $\beta = 0.2$, $\gamma = 0.1$, $\tau = 30$, $n = 10$, discretized at a sampling period of 6 seconds using the fourth-order Runge-Kutta method, with initial condition $y_0 = 0.9$. Chaotic dynamics are extremely sensitive to initial conditions: in general, small perturbations in the present state yield widely diverging outcomes, rendering long-term prediction intractable, popularized as the butterfly effect [43]. Since the only data available is the time series, we use time-delay embedding to model the dynamics of the chaotic system, i.e., to predict the next data sample $y_{i+1}$, the input is a properly chosen sliding window $\mathbf{u}_i = [y_i, y_{i-1}, \cdots, y_{i-(\ell-1)}]$, where $\ell$ is the embedding dimension. Here, we follow the experimental setup outlined in [38], with embedding length $\ell = 7$, and, instead of using the clean data $y_t$, we construct a nonlinear empirical model of the dynamical system from only the noisy measurements (desired output $d_i$).

Cooperative filtering aims to construct an empirical model using (pseudo-) clean data extracted from the noisy measurements. The signal estimator is coupled with the weight parameter estimator. For the cubature Kalman approach (specifically, the square-root version or SCKF), an RNN is used to model the dynamics with its weights, the weights of the RNN are estimated from the latest signal estimate (and *vice versa*). The SSM for the RNN architecture, trained using square-root cubature Kalman filter (SCKF), is defined as

$$\begin{aligned} \mathbf{W}_i &= \mathbf{W}_{i-1} + \boldsymbol{q}_{i-1} \\ d_i &= \mathbf{W}^{(\mathrm{o})} \mathrm{h}(\mathbf{W}^{(\mathrm{r})} \mathbf{x}_{i-1} + \mathbf{W}^{(\mathrm{i})} \mathbf{u}_i) + r_i \end{aligned}$$

where the input weight $\mathbf{W}^{(\mathrm{i})}$, the recurrent weight $\mathbf{W}^{(\mathrm{r})}$, and the output weight $\mathbf{W}^{(\mathrm{o})}$ are matrices of appropriate dimensions (collectively, they can be arranged into an orderly weight vector $\mathbf{W}_i$), the process noise is additive Gaussian with zero-mean and covariance $Q_i$, i.e., $\boldsymbol{q}_i \sim \mathcal{N}(0, Q_i)$, the measurement

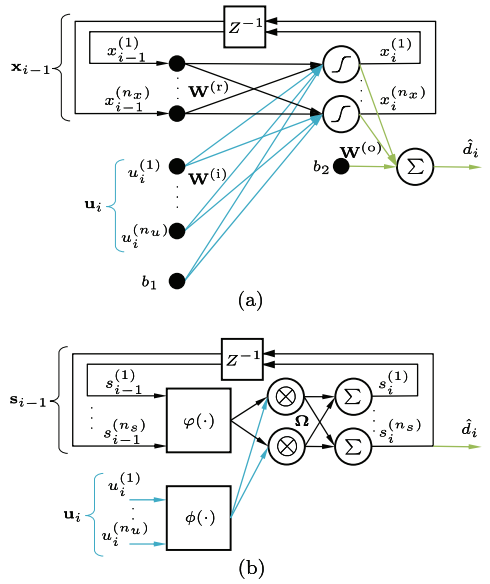



Fig. 3: Recurrent network trained using (a) Square-Root Cubature Kalman Filter (b) Functional Bayesian Filter.

noise is $r_i \sim \mathcal{N}(0, R_i)$, internal state or output of the hidden layer at time $(i-1)$ is $\mathbf{x}_{i-1}$, the input is denoted $\mathbf{u}_i$, the desired output $d_i$ is the measurement, and $\mathrm{h}(\cdot)$ denotes the activation function. The input embedding dimension is set at $n_u = 7$, with one self-recurrent hidden layer with 5 neurons ($n_x = 5$), a single output, and bias at each node, as shown in Fig. 3(a). The hidden neuron activations use hyperbolic tangent function, and the output neuron is linear. The dynamic state-space model for the signal estimator is written as

$$\mathbf{u}_i = \mathbf{f}(\mathbf{u}_{i-1}, \hat{\mathbf{W}}_{i-1|i-1}, \mathbf{x}_{i-2}) + [1 \quad 0 \cdots 0]^\intercal w_{i+1}$$
$$d_i = [1 \quad 0 \cdots 0]\mathbf{u}_i + v_i$$

where $\mathbf{f}$ denotes the 7-5R-1 RNN state transition function concatenated with the delayed output, the measurement noise $v_i \sim \mathcal{N}(0, \sigma_v^2)$ corresponds to the signal-to-noise ratio (SNR), the process noise $w_{i-1} \sim \mathcal{N}(0, \sigma_w^2)$ was fixed to be $10\%$ of $\sigma_v^2$, and the initial estimate is assumed to be zero with unity covariance.

A noisy (10 dB SNR) MG chaotic time sequence of 1000 samples is used to train the RNN. For each training run, ten batches were made. Each batch consists of 100 time-step updates, from a randomly selected starting point in the training sequence. The state of the RNN at time step $i = 0$ was assumed to be zero, i.e., $\mathbf{x}_0 = \mathbf{0}$. During the test phase, SCKF is performed on an independent sequence of 100 noisy samples using the state-transition equation obtained from the fixed weights $\mathbf{W}$. The ensemble-averaged mean square error (MSE) is computed for 50 independent training runs.

For FBF, the recurrent architecture is parsimonious, Fig. 3(b), and the signal state and network weights are estimated simultaneously by construction. The kernel parameters for the state, input, state covariance $\mathbf{P}_1$, and weight covariance $\mathbf{P}_4$ are $a_s = 0.6$, $a_u = 1.8$, $a_{P_1} = 0.4$, and $a_{P_4} = 0.2$, respectively. The state covariance, output variance, and weight covariance are initialized as $\sigma_s^2 = 10$, $\sigma_y^2 = 0.08$, $\sigma_{P_4}^2 = 10$, respectively. There are no bias terms for our FBF implementation. Using the small-step-size theory framework [44], which self-regularizes KAF, we scale the state Kalman gain $\mathbb{K}_1$ by a constant factor of 0.5, and the weight gain $\mathbb{K}_2$ by a constant factor of 0.1. Note, the FBF is an online kernel method with a dictionary constructed incrementally with each incoming sample (the initial point is randomly initialized), e.g., after 100 samples, the dictionary size becomes 101.

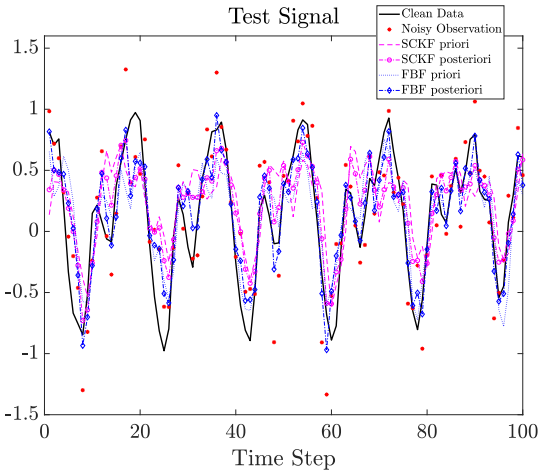



Fig. 4: Signal enhancement for noisy Mackey-Glass chaotic time series.

SCKF has been successfully validated to significantly outperform other known nonlinear filters such as EKF and central-difference Kalman filter (CDKF) and provides improved numerical stability over CKF [38]. It is important to reiterate that the two essential properties of error covariance matrix (symmetry and positive definiteness) are always preserved in FBF, since we are using positive definite kernel functions satisfying Mercer's conditions, unlike input-space arithmetic operations such as CKF, where these two properties are often lost or destroyed and a square-root version is preferred. Here we focus on the performance comparison of SCKF and FBF. Fig. 4 shows the filtering performance on the independent test signal during one of the 50 runs. The "priori" label denotes the time update using the predictive density before receiving a new measurement; "posteriori", the measurement update from the posterior density. Fig. 5 shows the ensemble-averaged MSE (error bars represent one standard deviation) over all 50 runs versus the number of batch iterations, where each training iteration consists of a 100-sample noisy sequence with random starting point in the 1000-sample training data. Clearly, the FBF significantly improves the quality of the signal as compared to the SCKF.

### *IV.B. Ikeda*

Next, we evaluate the performance of the FBF using multivariate chaotic time series and under various non-Gaussian noise conditions. The 2D Ikeda map is defined by

$$\mathbf{x}_{i+1} = \begin{bmatrix} x_{i+1}^{(1)} \\ x_{i+1}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 + c\left(x_i^{(1)} \cos t_i - x_i^{(2)} \sin t_i\right) \\ c\left(x_i^{(1)} \sin t_i + x_i^{(2)} \cos t_i\right) \end{bmatrix} \tag{77}$$

where parameter $c = 0.84$ and $t_i = 0.4 - \frac{6}{1+\left(x_i^{(1)}\right)^2+\left(x_i^{(2)}\right)^2}$ with initial condition $\mathbf{x}_0 = [1, 0]^\intercal$. Four different types of

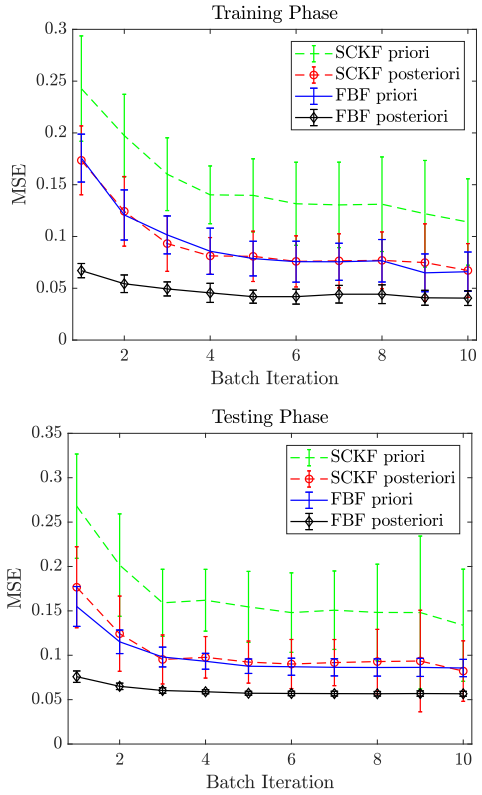



Fig. 5: Ensemble-averaged Mean-Squared Error (MSE) over 50 runs vs. number of batch iterations (each training iteration consists of a 100-sample sequence with random starting point).

additive noise (Gaussian, Laplacian, uniform, and symmetric alpha-stable) are introduced to obtain the noisy observations $\mathbf{y}_i$, with 3 dB of SNR, from which we will estimate the clean data. Since the variance of alpha-stable noise is undefined for $\alpha < 2$, SNR was estimated as the ratio of the signal power to the squared dispersion or scale parameter $\gamma$ of the noise [30]. The initial 201 data points are used for training, and the next 200 points are for testing. For each estimator, 20 independent trials were averaged to produce the MSE result (mean $\pm$ $\sigma$). The nonlinear Kalman extensions (EKF, UKF, and CKF) assumes known accurate system models, which should provide an advantage. For DSMCE, the conditional embedding operator construction assume known hidden states $\mathbf{x}_i$ and noisy measurements $\mathbf{y}_i$, which should also provide an advantage during training. For KKF-CEO and FBF, only the noisy measurements are used for training, i.e., using $\mathbf{y}_{i-1}$ as input to predict $\mathbf{y}_i$. Please refer to [22] for a detailed discussion on the experimental setup.

For FBF, the empirical hidden states $\hat{\mathbf{x}}_i$ have dimension $n_x = 2$, i.e., $n_s = 4$ for the augmented state vectors $\mathbf{s}_i$, with input state kernel parameter $a_s = 0.8$, state covariance kernel parameter $a_{\mathbf{P}_1} = 0.8$, and weight covariance kernel parameter $a_{\mathbf{P}_4} = 0.8$. Initialization was set for the following parameters: state variance $\sigma_s^2 = 1$, weight variance $\sigma_{\mathbf{\Omega}}^2 = 1$, output variance $\sigma_y^2 = 40$.

Table II summarizes the mean-squared-error (MSE) testing set performance of the proposed FBF with the performances of EKF, UKF, CKF, DSMCE, and KKF-CEO. Despite the obvious disadvantage of not having accurate system knowledge or access to hidden states during training, FBF outperforms the best in all four noisy environments tested. The next-best performances are given by KKF-CEO. Kernel methods are able to leverage high-dimensional nonlinear representation of the signal in the RKHS to better model dynamical systems. Furthermore, the FBF uses information theoretic learning to preserve the nonparametric nature of correlation learning and MSE adaptation. The cost function is still directly estimated from observation via a Parzen kernel estimator, but it extracts more information from the data for adaptation, and therefore yields solutions that are more accurate than MSE in non-Gaussian and nonlinear signal processing. The FBF outperformed the KKF-CEO method because it uses a full state-space representation constructed in the RKHS, which can scale with the complexity of the nonlinear dynamics, and not only assuming a simple additive noise system model.

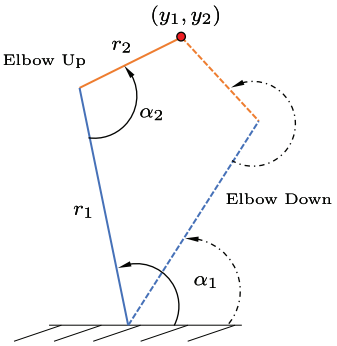



Fig. 6: Two-joint robot arm illustrating how the Cartesian coordinates $(y_1, y_2)$ of the end effector is mapped to the given angles $(\alpha_1, \alpha_2)$. The solid and dashed lines show the 'elbow up' and 'elbow down' situations, respectively.

### *IV.C. Modeling Inverse Kinematics in a Robotic Arm*

In a two-joint robotic arm, Fig. 6, given the joint angles $(\alpha_1, \alpha_2)$, the kinematics equations give the Cartesian coordinate of the robot arm end-effector position as:

$$(y_1, y_2) = \begin{cases} r_1 \cos(\alpha_1) - r_2 \cos(\alpha_1 + \alpha_2) \\ r_1 \sin(\alpha_1) - r_2 \sin(\alpha_1 + \alpha_2) \end{cases}$$

where $r_1 = 0.8$ and $r_2 = 0.2$ are the link lengths, with $\alpha_1 \in [0.3, 1.2]$ and $\alpha_2 \in [\pi/2, 3\pi/2]$ are the joint ranges. Finding the mapping from $(y_1, y_2)$ to $(\alpha_1, \alpha_2)$ is called the *inverse kinematics*, which is not a one-to-one mapping: as shown in Fig. 6, both the elbow-up and elbow-down joint angles result in the same tip-of-the-arm position.

Let the state vector be $\mathbf{x} = [\alpha_1, \alpha_2]^\intercal$, and the measurement vector be $y = [y_1, y_2]^\intercal$. The state-space representation of the inverse kinematic problem is written as

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \boldsymbol{w}_i$$
$$\mathbf{y}_i = \begin{bmatrix} \cos(\alpha_1) - \cos(\alpha_1 + \alpha_2) \\ \sin(\alpha_1) - \sin(\alpha_1 + \alpha_2) \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \boldsymbol{v}_k$$

with zero-mean Gaussian measurement and process noises, $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}[0.01^2, 0.1^2])$ and $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, 0.005\mathbf{I}_2)$ respectively, where $\mathbf{I}_2$ is the 2D identity matrix.

We compare the nonlinear filter performances of cubature Kalman filter and FBF using the root-mean square error (RMSE) of the angles over 200 Monte Carlo runs. As a

TABLE II: Test Set MSE of 2D Ikeda Map

| Alg. / Noise | EKF | UKF | CKF | DSMCE | KKF-CEO | **FBF** |
|---|---|---|---|---|---|---|
| Gaussian | $0.3630 \pm 0.0448$ | $0.2639 \pm 0.0218$ | $0.2374 \pm 0.0176$ | $0.3918 \pm 0.0502$ | $0.2253 \pm 0.0168$ | $\mathbf{0.1803 \pm 0.0129}$ |
| Laplacian | $0.3897 \pm 0.0407$ | $0.2719 \pm 0.0217$ | $0.2574 \pm 0.0199$ | $0.3555 \pm 0.0626$ | $0.2121 \pm 0.0202$ | $\mathbf{0.1687 \pm 0.0117}$ |
| Uniform | $0.3843 \pm 0.0308$ | $0.2696 \pm 0.0213$ | $0.2427 \pm 0.0193$ | $0.3945 \pm 0.0800$ | $0.2384 \pm 0.0180$ | $\mathbf{0.1848 \pm 0.0103}$ |
| Stable ($\alpha = 1.6$) | $0.3021 \pm 0.1232$ | $0.2465 \pm 0.0969$ | $0.2580 \pm 0.0991$ | $0.2319 \pm 0.1335$ | $0.1461 \pm 0.0413$ | $\mathbf{0.1224 \pm 0.0205}$ |

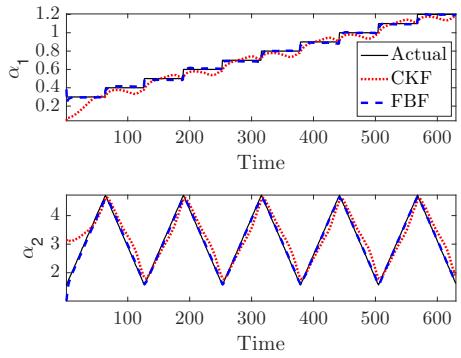



Fig. 7: Inverse Kinematics State Estimation

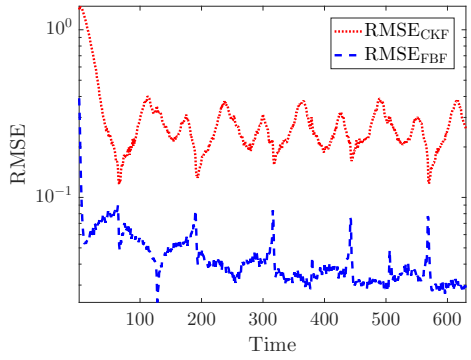



Fig. 8: Performances

self-assessment of the estimation errors, a filter produces an error covariance matrix. Thus, we consider the filter-estimated RMSE as the square-root of the appropriate diagonal entries of the covariance, averaged over the runs. The filter estimate is deemed consistent if the true and estimated RMSE are equal. Again, we see that FBF outperforms CKF.

## V. Conclusion

Various aspects of the classic Kalman filter have been extended for practical applications. This paper presents a novel formulation that tackles all three major shortcomings–linearity, prior knowledge of accurate system parameters, and lower-order statistics–under a unifying framework, using the theory of reproducing kernel Hilbert Space. Applying kernel method and the representer theorem to perform linear quadratic estimation on a full state space model in a functional space, we derive the Bayesian recursive state estimation for a general nonlinear dynamical system in the original input space. Unlike existing nonlinear Kalman extensions where the system dynamics are assumed known, the state-space representation for the Functional Bayesian Filter is completely learned from observation, with universal approximation property. Using positive definite kernel functions satisfying Mercer's conditions to compute information quantities, the FBF exploits both the statistical and time-domain information about the signal, extracts higher-order moments, and preserves the properties of covariances without the ill effects due to conventional arithmetic operations. This novel kernel adaptive filtering algorithm is applied to chaotic time-series estimation and prediction using Gaussian and non-Gaussian noises and inverse kinematics modeling. FBF outperforms existing algorithms under different noise conditions in the experiments.

Kernel method is extremely versatile and comes with many appealing properties. In the future, we will examine sparsification techniques for FBF, apply FBF to nonnumerical data such as graphs and modeling biological systems using neural spike trains, and explore applications for nonstationary environments using ideas from kernel transfer learning.

## References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME, Series D., Journal of Basic Eng.*, vol. 82, pp. 35–45, 1960.

[2] H. Kushner, "Approximations to optimal nonlinear filters," *IEEE Transactions on Automatic Control*, vol. 12, no. 5, pp. 546–556, October 1967.

[3] B. Anderson and J. Moore, *Optimal Filtering*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1979.

[4] S. Haykin, *Kalman Filtering and Neural Networks*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[5] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of 11th International Symposium on Aerospace/Defense Sensing (AeroSense), Simulations and Controls*, 1997, pp. 182–193.

[6] E. A. Wan and R. V. D. Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, Oct 2000, pp. 153–158.

[7] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, June 2009.

[8] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

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

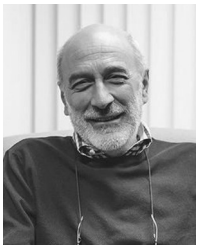
[10] A. Aravkin, J. V. Burke, L. Ljung, A. L., and G. Pillonetto, "Generalized Kalman smoothing: Modeling and algorithms," *Automatica*, vol. 86, pp. 63 – 86, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109817304326

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

[12] B. Jia, M. Xin, and Y. Cheng, "High-degree cubature kalman filter," *Automatica*, vol. 49, no. 2, pp. 510 – 518, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510981200550X

[13] L. Ralaivola and F. d'Alché-Buc, "Dynamical modeling with kernels for nonlinear time series prediction," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, ser. NIPS'03. Cambridge, MA, USA: MIT Press, 2003, pp. 129–136. [Online]. Available: http://dl.acm.org/citation.cfm?id=2981345.2981362
[14] L. Song, J. Huang, A. J. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 961–968. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553497
[15] K. Fukumizu, L. Song, and A. Gretton, "Kernel Bayes' rule: Bayesian inference with positive definite kernels," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 3753–3783, Dec. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2567709.2627677
[16] B. Schölkopf, A. J. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, July 1998.
[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press, 2005.
[18] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3471–3476.
[19] R. Turner, M. P. Deisenroth, and C. E. Rasmussen, "State-space inference and learning with Gaussian processes," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 13–15 May 2010, pp. 868–875.
[20] P. Neveux, E. Blanco, and G. Thomas, "Robust filtering for linear time-invariant continuous systems," *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 4752–4757, 2007.
[21] T. Zhou, "Robust recursive state estimation with random measurement droppings," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 156–171, 2016.
[22] P. Zhu, B. Chen, and J. C. Príncipe, "Learning nonlinear generative models of time series with a Kalman filter in RKHS," *IEEE Transactions on Signal Processing*, vol. 62, no. 1, pp. 141–155, Jan 2014.
[23] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NJ, USA: Wiley, 2010.
[24] K. Li and J. C. Príncipe, "The kernel adaptive autoregressive-moving-average algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 334–346, Feb. 2016.
[25] ——, "Automatic insect recognition using optical flight dynamics modeled by kernel adaptive ARMA network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2726–2730.
[26] ——, "Biologically-inspired spike-based automatic speech recognition of isolated digits over a reproducing kernel Hilbert space," *Frontiers in Neuroscience*, vol. 12, p. 194, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00194
[27] R. J. Williams, "Training recurrent networks using the extended Kalman filter," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, vol. 4, June 1992, pp. 241–246 vol.4.
[28] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.
[29] E. Parzen, "Statistical methods on time series by Hilbert space methods," Applied Mathematics and Statistics Laboratory, Stanford, CA, Tech. Rep. 23, 1959.
[30] J. C. Príncipe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. New York, NY, USA: Springer, 2010.
[31] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 09 1962. [Online]. Available: https://doi.org/10.1214/aoms/1177704472
[32] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.
[33] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. 14th Annual Conf. Comput. Learning Theory*, 2001, pp. 416–426.
[34] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent NARX neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 27, no. 2, pp. 208–215, dec 1997.
[35] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000085710136
[36] K. Li and J. C. Príncipe, "Surprise-novelty information processing for Gaussian online active learning (SNIP-GOAL)," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–6.
[37] ——, "Transfer learning in adaptive filters: The nearest instance centroid-estimation kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6520–6535, 2017.
[38] I. Arasaratnam, "Cubature Kalman filtering: Theory & applications," Ph.D. dissertation, McMaster University, 2009.
[39] B. Chen, X. Liu, H. Zhao, and J. C. Príncipe, "Maximum correntropy Kalman filter," *Automatica*, vol. 76, pp. 70 – 77, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510981630396X
[40] W. Härdle, *Applied Nonparametric Regression*, ser. Econometric Society Monographs. Cambridge University Press, 1990.
[41] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*. MIT Press, 1999, pp. 536–542. [Online]. Available: http://papers.nips.cc/paper/1491-kernel-pca-and-de-noising-in-feature-spaces.pdf
[42] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, Jul. 1977.
[43] E. Ott, *Chaos in dynamical systems*, 2nd ed. Cambridge University Press, 2002.
[44] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall, 2002.

**Kan Li** (S'08-M'15) received the B.A.Sc. degree in electrical engineering from the University of Toronto in 2007, the M.S. degree in electrical engineering from the University of Hawaii in 2010, and the Ph.D. degree in electrical engineering from the University of Florida in 2015. He was awarded an SBIR grant from the NSF in 2016 to develop ultra-low power, always-on voice trigger. He worked as a research scientist at the University of Florida from 2018-2020, and is currently the principal lead for new product development and senior research scientist at Aventusoft Inc., supported by SBIR grants from DARPA, NIH, and NSF. His main research interests include artificial intelligence, machine learning, and signal processing, with applications in biologically-inspired computing, computer vision, health care, lifelong learning, time series analysis, and nonlinear dynamical system modeling.



**José C. Príncipe** (M'83-SM'90-F'00-LF'17) is the BellSouth and Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, and the Founding Director of the Computational NeuroEngineering Laboratory (CNEL). His primary research interests are in advanced signal processing with information theoretic criteria and adaptive models in reproducing kernel Hilbert spaces (RKHS), with application to brain-machine interfaces (BMIs). Dr. Príncipe is a Fellow of the IEEE, ABME, and AIBME. He is the past Editor in Chief of the IEEE Transactions on Biomedical Engineering, past Chair of the Technical Committee on Neural Networks of the IEEE Signal Processing Society, Past-President of the International Neural Network Society, and a recipient of the IEEE EMBS Career Award, the IEEE Neural Network Pioneer Award, and the IEEE SPS Claude Shannon-Harry Nyquist Technical Achievement Award.

## APPENDIX

Since $n_{\mathbf{\Omega}}$ is infinite for the Gaussian kernel, we show how to compute each of the submatrices $\mathbf{P}_i^-$ of the *a priori* state covariance matrix estimate $\mathbf{P}$, except $\mathbf{P}_3^-$ since it is simply the transpose of $\mathbf{P}_2^-$, in the input space using the *kernel trick*. In order to make the actual computation tractable, we break the weight matrix vector $\mathbf{\Omega}_i$ in (34) into individual state dimension components and rewrite (36) as

$$\begin{bmatrix} \mathbf{s}_i \\ \mathbf{\Omega}_i^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 & \mathbf{F}_2^{(k)} \\ \mathbf{0} & \mathbf{I}_{n_{\mathbf{\Omega}^{(k)}}} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{i-1} \\ \mathbf{\Omega}_i^{(k)} \end{bmatrix} + \boldsymbol{w}_{i-1}^{(k)} \tag{78}$$

where $\mathbf{\Omega}_i^{(k)}$ are all the weights connected to the $k$-th output state node ($1 \leq k \leq n_s$) and

$$\mathbf{F}_2^{(k)}(i) \triangleq \frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}_i^{(k)}} = \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal. \tag{79}$$

At each time step $i$, this process is repeated for each of the $n_s$ state components.

### A.A. Computing $\boldsymbol{P}_2^-$ Recursively

Since the update for $(\mathbf{P}_2^-)^{(k)} \in \mathbb{R}^{n_s \times n_{\mathbf{\Omega}^{(k)}}}$ is a subroutine of $(\mathbf{P}_1^-)^{(k)}$, as shown in (52), we first show how to recursively update this submatrix. Substituting (43), (79), and (68) into (53), for the $k$-th state component, gives

$$\begin{aligned} \left(\mathbf{P}_2^-\right)^{(k)} &= \boldsymbol{\Lambda}_i \left(\mathbf{P}_2^+\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \\ &= \boldsymbol{\Lambda}_i \left(\mathbf{P}_2^- - \mathbb{K}_1 \mathbf{L}_2^\intercal\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \\ &= \boldsymbol{\Lambda}_i \left(\mathbf{P}_2^- - \mathbb{K}_1 \mathbb{I} \mathbf{P}_2^-\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \\ &= \underbrace{\boldsymbol{\Lambda}_i \left(\mathbf{I}_{n_s} - \mathbb{K}_1 \mathbb{I}\right)}_{\boldsymbol{\Lambda}_i'} \left(\mathbf{P}_2^-\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \\ &\triangleq \boldsymbol{\Lambda}_i' \left(\mathbf{P}_2^-\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)}. \end{aligned} \tag{80}$$

The above expression has the same form as Equation (28) in the kernel adaptive ARMA formulation [24], which we reproduce here:

$$\frac{\partial \mathbf{s}_i}{\partial \mathbf{\Omega}_i^{(k)}} = \boldsymbol{\Lambda}_i \frac{\partial \mathbf{s}_{i-1}}{\partial \mathbf{\Omega}_i^{(k)}} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal. \tag{81}$$

We see that the FBF formulation reduces to gradient descent if the state variable Kalman gain $\mathbb{K}_1$ is zero and the weights covariance matrix $\left(\mathbf{P}_4^+\right)^{(k)}$ is the identity matrix, i.e., unit variance. In other words, the FBF generalizes the kernel adaptive ARMA algorithm. Using the ITL interpretation, where the state-transition gradient $\boldsymbol{\Lambda}_i$ is a weighted Parzen estimate of the joint input data $(\mathbf{s}_i, \mathbf{u}_i)$ pdf, we are effectively evolving the density estimate recursively.

Given the initialization in (72), the ensuing update becomes

$$\mathbf{P}_2^{(k)}(1|0) = \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_0, \mathbf{u}_1)^\intercal \mathbf{P}_4^{(k)}(0). \tag{82}$$

By induction, we can factor out the basis functions $\psi(\cdot, \cdot)$ and express the recursion (80) as

$$\begin{aligned} \left(\mathbf{P}_2^-\right)^{(k)} &= \boldsymbol{\Lambda}_i' \boldsymbol{V}_{i-1}^{(k)} \left(\mathbf{\Psi'}_{i-1}^\intercal\right)^{(k)} + \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \\ &= \left[\boldsymbol{\Lambda}_i' \boldsymbol{V}_{i-1}^{(k)}, \mathbf{I}_{n_s}^{(k)}\right] \begin{bmatrix} \left(\mathbf{\Psi'}_{i-1}^\intercal\right)^{(k)} \\ \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^\intercal \left(\mathbf{P}_4^+\right)^{(k)} \end{bmatrix} \\ &= \boldsymbol{V}_i^{(k)} \left(\mathbf{\Psi'}_i^\intercal\right)^{(k)} \end{aligned} \tag{83, 84}$$

where $\left(\mathbf{\Psi'}_i\right)^{(k)} \triangleq \left[\left(\mathbf{\Psi'}_{i-1}\right)^{(k)}, \left(\mathbf{P}_4^+\right)^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)\right] \in \mathbb{R}^{n_{\mathbf{\Omega}} \times i}$ is the dictionary of features generated by the input sequence and corresponding forward-propagated states, normalized by $\mathbf{P}_4^{(k)}(i-1|i-1)$, and $\boldsymbol{V}_i^{(k)} \triangleq \left[\boldsymbol{\Lambda}_i' \boldsymbol{V}_{i-1}^{(k)}, \mathbf{I}_{n_s}^{(k)}\right] \in \mathbb{R}^{n_s \times i}$ is the updated state-transition gradient, with initializations $\left(\mathbf{\Psi'}_1\right)^{(k)} = [\sigma^2_{\mathbf{\Omega}^{(k)}} \psi(\mathbf{s}_0, \mathbf{u}_1)]$ and $\boldsymbol{V}_1^{(k)} = \mathbf{I}_{n_s}^{(k)}$, respectively. Note that this implementation allows a different initial variance for each state variable, however, unless explicitly given, for simplicity and without loss of generality, we assume a constant initial variance across all states, as in (71).

### A.B. Updating $\boldsymbol{P}_4^+$

Before proceeding to the computation for $(\mathbf{P}_1^-)^{(k)}$, we note that (83) can be greatly simplified if $\left(\mathbf{P}_4^+\right)^{(k)}$ is just a scalar multiplied by the identity matrix. Here we show that $\mathbf{P}_4$ maintains its diagonal form after the measurement update, given its scaled identity matrix initialization in (71):

$$\begin{aligned} \mathbf{P}_4^{(k)}(1|1) &= \mathbf{P}_4^{(k)}(1|0) - \left(\mathbb{K}_2 \mathbf{L}_2^\intercal\right)^{(k)} \\ &= \mathbf{P}_4^{(k)}(0) + \sigma^2_{\mathbf{\Omega}^{(k)}} \mathbf{I}_{n_{\mathbf{\Omega}}} - \left(\mathbf{P}_2^{-\intercal} \mathbb{I}^\intercal (\mathbf{M} + \sigma_y^2 \mathbf{I}_{n_y})^{-1} \mathbb{I} \mathbf{P}_2^-\right)^{(k)} \\ &= \underbrace{2\sigma^2_{\mathbf{\Omega}^{(k)}}}_{\sigma^{2-}_{\mathbf{\Omega}^{(k)}}(1) \triangleq \sigma^2_{\mathbf{\Omega}^{(k)}}(1|0)} \mathbf{I}_{n_{\mathbf{\Omega}}} - \mathbf{\Psi'}_1 \underbrace{\boldsymbol{V}_1^{(k)\intercal} \mathbb{I}^\intercal \mathbf{N} \mathbb{I} \boldsymbol{V}_1^{(k)}}_{\mathbf{B}_1^{(k)}} \mathbf{\Psi'}_1^\intercal \end{aligned} \tag{85}$$

$$= \sigma^{2-}_{\mathbf{\Omega}^{(k)}}(1) \mathbf{I}_{n_{\mathbf{\Omega}}} - \underbrace{[\sigma^2_{\mathbf{\Omega}^{(k)}} \psi(\mathbf{s}_0, \mathbf{u}_1)] \mathbf{B}_1^{(k)} \left[\sigma^2_{\mathbf{\Omega}^{(k)}} \psi(\mathbf{s}_0, \mathbf{u}_1)\right]}_{\varsigma^2_{\mathbf{\Omega}^{(k)}}(1) \mathbf{I}_{n_{\mathbf{\Omega}}}} \tag{86}$$

$$= \left(\sigma^{2-}_{\mathbf{\Omega}^{(k)}}(1) - \varsigma^2_{\mathbf{\Omega}^{(k)}}(1)\right) \mathbf{I}_{n_{\mathbf{\Omega}}} \tag{87}$$

$$\triangleq \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(1) \mathbf{I}_{n_{\mathbf{\Omega}}} \tag{88}$$

where the superscripts $^-$ and $^+$ are used in conjunction with a single parenthesized index $(i)$ as shorthands for specifying the *a priori* estimate $(i|i-1)$ and *a posteriori* estimates $(i|i)$ in iteration $i$, the term $\mathbf{B}_1^{(k)}$ is a scalar from the associative property of matrix multiplication, and $\varsigma^2_{\mathbf{\Omega}^{(k)}}(1)$ is a weighted scalar product using the *kernel trick*. In general, for an arbitrary iteration $i > 1$, the term $\mathbf{B}_i^{(k)} \triangleq \boldsymbol{V}_i^{(k)\intercal} \mathbb{I}^\intercal \mathbf{N} \mathbb{I} \boldsymbol{V}_i^{(k)} \in \mathbb{R}^{i \times i}$ will be a square matrix and can be computed in a straightforward manner. If we denote its $l$-th row and $m$-th column element as $\mathbf{b}_{lm}^{(k)}$ and apply the *kernel trick*, then the entire second term on the right-hand side of (86) effectively becomes a scalar (to reconcile with the dimensionality of the RKHS, we multiply it with an identity matrix $\mathbf{I}_{n_{\mathbf{\Omega}}}$):

$$\begin{aligned} &\mathbf{\Psi'}_i \underbrace{\boldsymbol{V}_i^{(k)\intercal} \mathbb{I}^\intercal \mathbf{N} \mathbb{I} \boldsymbol{V}_i^{(k)}}_{\mathbf{B}_i^{(k)}} \mathbf{\Psi'}_i^\intercal \\ &= [\mathbf{\Psi'}_{i-1}, \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(i-1) \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)] \mathbf{B}_i^{(k)} \begin{bmatrix} \mathbf{\Psi'}_{i-1}^\intercal \\ \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(i-1) \psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \end{bmatrix} \\ &= \sum_{m=1}^{i} \sum_{l=1}^{i} \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(l-1) \psi(\mathbf{s}_{l-1}, \mathbf{u}_l) \mathbf{b}_{lm}^{(k)} \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(m-1) \psi(\mathbf{s}_{m-1}, \mathbf{u}_m) \\ &= \sum_{m=1}^{i} \sum_{l=1}^{i} \mathbf{b}_{lm}^{(k)} \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(l-1) \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(m-1) \mathcal{K}_{a_s}(\mathbf{s}_{l-1}, \mathbf{s}_{m-1}) \mathcal{K}_{a_u}(\mathbf{u}_l, \mathbf{u}_m) \\ &\triangleq \varsigma^2_{\mathbf{\Omega}^{(k)}}(i). \end{aligned} \tag{89}$$

Similar to the update for $\mathbf{P}_2^-$, we see that there is an information theoretic interpretation here: the RKHS filter parameter

covariance $\mathbf{P}_4^+$ is updated using a weighted information potential of the joint input data $(\mathbf{s}_i, \mathbf{u}_i)$.

Since we only need to keep track of the scalar $\sigma^{2+}_{\mathbf{\Omega}^{(k)}}(i)$, this simplifies the weighted features in (84) to

$$\mathbf{\Psi'}_i \triangleq [\mathbf{\Psi}'_{i-1}, \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)] \tag{90}$$

$$\boldsymbol{V}_i^{(k)} \triangleq \left[\boldsymbol{\Lambda}'_i \boldsymbol{V}_{i-1}^{(k)}, \sigma^{2+}_{\mathbf{\Omega}^{(k)}}(i-1)\mathbf{I}_{n_s}^{(k)}\right] \tag{91}$$

and

$$\left(\mathbf{P}_2^-\right)^{(k)} = \boldsymbol{V}_i^{(k)} \mathbf{\Psi'}_i^{\mathsf{T}} \tag{92}$$

where we can now separate the shared feature set completely from their state-variable dependent coefficients for ease of computation and storage, i.e., dropping the superscript $^{(k)}$ for the standard representer theorem form.

### *A.C. Updating $\boldsymbol{P}_1^-$*

Finally, we have all the tools to compute $\mathbf{P}_1^-$. In order to unclutter the notation, we first show some intermediate steps. Substituting (79) and (68) for $\mathbf{F}_2^{(k)}$ and $(\mathbf{P}_2^+)^{(k)}$, respectively, the following product can be expressed as

$$\begin{aligned}\mathbf{F}_2^{(k)} \left((\mathbf{P}_2^+)^{(k)}\right)^{\mathsf{T}} &= \mathbf{I}_{n_s}^{(k)} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^{\mathsf{T}} \left((\mathbf{I}_{n_s} - \mathbb{K}_1 \mathbb{I}) \left(\mathbf{P}_2^-\right)^{(k)}\right)^{\mathsf{T}} \\ &\overset{(a)}{=} \mathbf{I}_{n_s}^{(k)} \underbrace{\psi(\mathbf{s}_{i-1}, \mathbf{u}_i)^{\mathsf{T}} \mathbf{\Psi}'_{i-1}}_{\mathbf{k}_{i-1}} \boldsymbol{V}_{i-1}^{(k)\mathsf{T}} (\mathbf{I}_{n_s} - \mathbb{K}_1 \mathbb{I})^{\mathsf{T}}\end{aligned} \tag{93}$$

where equality $(a)$ follows from (92), and $\mathbf{k}_{i-1}$ is a vector of kernel evaluations:

$$\mathbf{k}_{i-1} = \begin{bmatrix} \mathcal{K}_{a_s}(\mathbf{s}_{i-1}, \mathbf{s}_0)\mathcal{K}_{a_u}(\mathbf{u}_i, \mathbf{u}_1) \\ \vdots \\ \mathcal{K}_{a_s}(\mathbf{s}_{i-1}, \mathbf{s}_{i-2})\mathcal{K}_{a_u}(\mathbf{u}_i, \mathbf{u}_{i-1}) \end{bmatrix}. \tag{94}$$

Similarly

$$(\mathbf{P}_2^-)^{(k)} \left(\mathbf{F}_2^{(k)}\right)^{\mathsf{T}} = \boldsymbol{V}_i^{(k)} \underbrace{\mathbf{\Psi}_i'^{\mathsf{T}} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i)}_{\mathbf{k}_i^{\mathsf{T}}} \left(\mathbf{I}_{n_s}^{(k)}\right)^{\mathsf{T}}. \tag{95}$$

Finally, rewriting the *a priori* state covariance estimate in (52) for the $k$-th state component, using (93) and (95), yields

$$\begin{aligned}(\mathbf{P}_1^-)^{(k)} &= \left[\mathbf{F}_1 (\mathbf{P}_1^+)^{(k)} + (\mathbf{F}_2)^{(k)} \left((\mathbf{P}_2^+)^{(k)}\right)^{\mathsf{T}}\right] \mathbf{F}_1^{\mathsf{T}} \\ &\quad + (\mathbf{P}_2^-)^{(k)} \left(\mathbf{F}_2^{(k)}\right)^{\mathsf{T}} + \sigma_s^2 \mathbf{I}_{n_s}^{(k)} \\ &= \left[\boldsymbol{\Lambda}_i (\mathbf{P}_1^+)^{(k)} + \mathbf{I}_{n_s}^{(k)} \mathbf{k}_{i-1} \boldsymbol{V}_{i-1}^{(k)\mathsf{T}} (\mathbf{I}_{n_s} - \mathbb{K}_1 \mathbb{I})^{\mathsf{T}}\right] \boldsymbol{\Lambda}_i^{\mathsf{T}} \\ &\quad + \boldsymbol{V}_i^{(k)} \mathbf{k}_i^{\mathsf{T}} \left(\mathbf{I}_{n_s}^{(k)}\right)^{\mathsf{T}} + \sigma_s^2 \mathbf{I}_{n_s}^{(k)}.\end{aligned} \tag{96}$$

which can be calculated in a straightforward manner, since all the terms involved are now expressed using finite dimensional vectors and matrices.