Unsupervised State-Space Modelling Using Reproducing Kernels
Felipe Tobar, Petar M. Djurič and Danilo P. Mandic

Abstract—A novel framework for the design of state-space models (SSM) is proposed whereby the state-transition function of the model is parametrised using reproducing kernels. The nature of SSMs requires learning a latent function that resides in the state space and for which input-output sample pairs are not available, thus prohibiting the use of gradient-based supervised kernel learning. To this end, we then propose to learn the mixing weights of the kernel estimate by sampling from the posterior density of the weights with respect to the observations, using Monte Carlo methods. We first introduce the offline version of the proposed algorithm, followed by the online version which performs inference on both the parameters and the hidden state through particle filtering. The accuracy of the estimation of the state-transition function is first validated on synthetic data. Next, we illustrate that the proposed (unsupervised) algorithm outperforms (supervised) deterministic kernel adaptive filters in learning and prediction of real-world electric-grid time series, while also providing probabilistic estimates, a key advantage over standard methods.

Index Terms—Support vector regression, nonlinear filtering, Monte Carlo methods, state-space models.

I. INTRODUCTION

The filtering problem [1] refers to the estimation of a latent stochastic process $X_{1:t}$ based on the observed sequence $Y_{1:t} = y_{1:t}$, where the processes $X_{1:t}$ and $Y_{1:t}$ are related according to a mathematical model $M$. This model is usually chosen from the class of state-space models (SSM) with discrete time, continuous state, and additive noise, where $X_t$ is the Markovian state and $Y_t$ the (noisy) observation, and has the form

$$
X_{t+1} = f_t(X_t) + W_t, \quad t \in \mathbb{R}
$$

where $f_t : \mathbb{R}^n \rightarrow \mathbb{R}^n$, the sensor function $h_t : \mathbb{R}^m \rightarrow \mathbb{R}^m$, and the noise processes $W_t \in \mathbb{R}^n$ and $V_t \in \mathbb{R}^m$. The model in (1) is general enough to explain a broad class of systems arising in a variety of applications from control theory [2], [3] to population models [4] and mathematical finance [5].

The solution to the filtering problem is given by the posterior density of the latent process conditional to the observations, that is, $p(X_{1:t}|y_{1:t})$. This conditional distribution is the solution of the Kushner-Stratonovich equation [6], [7], a nonlinear measure-valued differential equation that admits a closed-form solution only for a restricted class of systems, such as linear and Gaussian ones (Kalman filter [8]) or those satisfying the Beneš condition (Beneš filter [9]). In the general case, the posterior distribution is mathematically intractable, and numerical algorithms are usually employed to find approximate solutions. In particular, sequential Monte Carlo methods (SMC), or particle filters (PF), provide accurate estimates of the posterior density in the form of a discrete set of weights and particles; these are recursively updated based on the observed signal $y_{1:t}$ and the Bayesian filtering equations defined by the dynamic model $M$ [10], [11], [12].

The flexibility of PFs allows for approximating the posterior density $p(X_{1:t}|y_{1:t})$ with no rigid constraints on the functions $f_t, h_t$ (such as linearity) or the distributions of the noise processes $V_t, W_t$ (such as Gaussianity). This makes it possible to design SSMs comprising nonlinear functions and non-Gaussian noise, for which a posterior cannot be necessarily found in a closed-form. By virtue of the nonlinear filtering capability of PF methods, the model design can therefore freely focus on empirical evidence and prior knowledge (if any) of the nature of the state and observations, rather than adopting a simpler model to fulfil the stringent requirements of the filter. More specifically, our aim is to find a model that is general enough to account for all possible observations of the process $Y_t$, while at the same time not being too uninformative, as this would result in meaningless estimates of the hidden process $X_t$. Practical model design involves choosing a state-transition function $f_t$ and an observation function $h_t$, the latter being usually known and given by the data-collection framework, whereas the former reflects the dynamical properties of the hidden process and is unknown when the understanding of the signal-generating mechanism is vague. This, so-called design of the prior, is a fundamental component of filtering applications, yet it remains an open challenge.

Existing sampling-based algorithms for system identification include maximum a posteriori model selection [13], SMC methods for training neural networks [14], and model order determination using reversible jump Markov chain Monte Carlo (MCMC) [15]. Low-complexity parameter identification in SSMs, suitable for online implementation, can be achieved by means of artificial evolution [16], [17], that is, by considering the unknown parameters as states evolving according to a random walk.

Notice that the design of the prior can be cast into a function approximation problem, thus admitting the use of (data-
driven) machine learning algorithms. For instance, Gaussian processes methods [18] have recently found application in system identification by performing nonparametric estimation of the underlying SSM for filtering and smoothing [19], [20]; however, these are not able to operate online. Kernel methods [21], [22] are particularly suited for this estimation task, as they are universal function approximators by their very nature [23] and, akin to neural networks [24], their centres and weights that can be trained to learn data relationships in regression settings [25], [26]. Kernel adaptive filters [27] employ the so-called kernel trick [28] to provide nonlinear extensions of linear adaptive filters [29], allowing for real-time nonlinear estimation at a linear increase in computational cost. However, existing kernel adaptive filters based on least mean square [30], [31], recursive least squares [32], and ridge regression [25], [33] can only perform supervised function approximation to provide point estimates, this is inadequate when the process of interest is latent and inherently stochastic, thus requiring probabilistic estimates.

Kernel-based system identification has been traditionally performed by modelling the system nonlinearities using radial basis functions (RBF) [34], this involves an a priori choice of the RBF centres in conjunction with Expectation-Maximisation (EM) to find the kernel parameters. However, this approach does not provide the complete posterior of the mixing parameters nor does it admit an online mode of operation. A nonparametric alternative is to model the state space as a reproducing kernel Hilbert space (RKHS) [35], [36], whereby the infinite-dimensional state allows for accurate filtering and smoothing, but does not provide physically meaningful estimation of the dynamics of the true (original) state. A more recent approach makes use of the kernel embeddings of distributions [37], where [38], [39] model the conditional density of the state as an element in an RKHS; this is also achieved in an offline and supervised fashion, and the model needs to be trained using state samples. There is therefore a void in the open literature on nonlinear filtering when it comes to the design of state-space models that are: (i) flexible, in order to approximate the system nonlinearities with an arbitrary degree of accuracy, (ii) Bayesian, to equip the model with the ability to approximate the full posterior of the kernel mixing weights, (iii) unsupervised, not requiring pre-training using state samples, (iv) online, whereby both the model parameters and support vectors are recursively updated.

We here provide a solution to general state-space modelling by parametrising the state-transition function using kernels, thus, casting the function approximation problem into that of finding a finite set of parameters. The posterior density of these mixing parameters is then approximated using a pseudo-marginal MCMC approach [40] which combines Metropolis-Hastings and particle filter stages, and in offline and online fashions. The choice of the support vectors and the prior density of the mixing weights is also discussed based on the empirical knowledge of the observed process. We provide illustrative examples on nonlinear system identification and then validate the ability of the proposed method to learn nonlinear state-transition functions through the prediction of real-world signals, and over a performance evaluation against the normalised kernel least mean square algorithm (KLMS).

II. SUPERVISED FUNCTION APPROXIMATION USING REPRODUCING KERNELS

Consider the sets $X$, $D \subseteq \mathbb{R}$, the unknown mapping

$$f : X \rightarrow D$$

and the training set comprising available input-output data given by

$$S_N = \{(x_i, y_i) \in \mathbb{R}^m \text{ s.t. } y_i = f(x_i)\}_{i=1:N}. \quad (3)$$

We approximate the function $f(x)$ by a function in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with reproducing kernel $K$. According to the representer theorem [42], the optimal estimate with respect to an arbitrary loss function can be expressed in terms of the kernel function $K$, input training samples $x_i$, and a vector of mixing parameters $a = [a_1, \ldots, a_N]$ in the form

$$f_a(\cdot) = \sum_{i=1}^{N} a_i K(x_i, \cdot). \quad (4)$$

The centres of the kernel evaluations are referred to as support vectors and the approach to regression is known as support vector regression (SVR) [23].

A. Choice of Support Vectors and Mixing Parameters in Kernel Adaptive Filtering

When a large set of data is available for the training of the kernel estimator in (4), a subset of these samples needs to be chosen to deal with the trade-off between estimation accuracy and model complexity. This is because the computational cost of SVR increases with the number of training samples without necessarily improving the estimate. This procedure is known as sparsification or dictionary learning.

In kernel adaptive filtering [27] (the approach to nonlinear adaptive filtering using SVR), sparsification criteria include the approximate linear dependence (ALD) [43], which operates on the feature space and aims to avoid redundancies of feature samples, and the novelty criterion, which only admits samples that are distant enough from the current dictionary and at the same time improve the estimate [44], [45]. Adaptive strategies include presence-based sparsification [51], which eliminates support vectors not contributing to the estimation.

For a fixed set of support vectors, referred to as dictionary, the problem of identifying the optimal (in the least mean square sense) mixing parameters is straightforward, since the SVR estimate is linear in the parameters. Therefore, by treating the kernel evaluations as regressors and the observed signal as an output, the mixing weights can be found using gradient-based methods. These include ridge regression (RR) in the offline case, and least mean square (LMS) and recursive least squares (RLS) online cases. These linear estimation algorithms are the basis of kernel adaptive filters.
B. Example: The Gaussian Kernel

An extensively-used kernel in SVR is the Gaussian kernel, an infinite-support radial basis function given by
\[ K(x_i, x_j) = \exp \left( -\frac{||x_i - x_j||^2}{\sigma^2} \right) \]
where the parameter \( \sigma > 0 \) is referred to as the kernel width.

The properties of the RKHS induced by the Gaussian kernel have been studied in [23]. In particular, the Gaussian kernel is proven to be an universal kernel [46, Example 1], meaning that its induced RKHS is dense in the space of continuous functions. This property is crucial for the use of the Gaussian kernel in function approximation.

III. KERNEL STATE-SPACE MODELS

We consider the SSMs of the form (1) and address the task of system identification for the case when the state-transition function \( f \) is unknown and the sensor function \( h \) is known. The analysis of this class of systems is motivated by the fact that, in real-world applications, the function \( h \) is usually available and given by the sensor chosen for a specific experiment; the sensor function is often even assumed linear, such as in the case when the observation is one of the states. We also assume that a plausible region exists where the state can be found, this is supported by combining the knowledge of the kernel function (likelihood) and known physical constraints on the latent state. This assumption allows for a straightforward design of the dictionary.

Our aim is then to find a meaningful estimate of the state-transition function \( f \) in eq. (1), this is achieved by searching for such an estimate in \( \mathcal{H} \), the RKHS of the Gaussian kernel in eq. (5). The assumption that \( \mathcal{H} \) contains functions that approximate \( f \) arbitrarily well stems from the fact that this RKHS is dense in the space of continuous functions \( C^0(\mathbb{R}) \), meaning that if \( f \in C^0(\mathbb{R}) \), then there exists a sequence of functions \( \{ f_i \} \in \mathcal{H} \) that converges to \( f \) (see also [23], [46], [47]). Furthermore, the justification to express the optimal estimate (within \( \mathcal{H} \)) as a mixture of kernel evaluations comes from the representer theorem, since in the unsupervised case we can still assume that there exists a set of input-output samples (albeit unknown) with respect to which the estimated function will be found. Indeed, kernels are proven to be useful within unsupervised learning, see e.g. kernel density estimation [48], kernel principal component analysis [49], and unsupervised multiple kernel learning [50].

By parametrising the transition function \( f \) as \( f_a = \sum_{i=1}^{N} a_i K(s^i, \cdot) \), the approximated SSM takes the form
\[ X_{t+1} = \sum_{i=1}^{N} a_i K(s^i, X_t) + W_t \]
\[ Y_t = h(X_t) + V_t. \]

We refer to this class of models as kernel state-space models (KSSM). Within this formulation, the system identification problem boils down to finding a set of fixed support vectors that is representative of the region where the state currently lies, and their corresponding mixing parameters \( a \). We now propose a procedure to find the posterior density of the mixing parameters conditional to the observed process in the offline case, and proceed to choosing the support vectors based on standard sparsification criteria employed by kernel adaptive filters.

A. Offline Learning of the State-Transition Function

The estimate \( f_a \) can be regarded as a mapping from \( \mathbb{R}^N \) to \( \mathcal{H} \) according to \( a \to f_a = \sum_{i=1}^{N} a_i K(s^i, \cdot) \). As a consequence, by considering \( a \) as a random vector, \( f_a \) becomes a random function, the posterior density of which can be found using Bayesian inference.

The posterior density \( p(f_a | y_{1:t}) \) is then uniquely determined by the posterior density of the weights \( p(a | y_{1:t}) \): this allows us to find the transition-function posterior and, in particular, the conditional expectation \( f_\ast = \mathbb{E}[f_a | y_{1:t}] \) given by
\[ f_\ast = \int_{\mathbb{R}^N} f_a p(a | y_{1:t}) da = \sum_{i=1}^{N} \mathbb{E}[a_i | y_{1:t}] K(s^i, \cdot). \]

This means that \( f_\ast \) can be found by only computing \( \mathbb{E}[a_i | y_{1:t}] \), since \( f_\ast = f_\mathbb{E}[a_i | y_{1:t}] \).

We now investigate how to sample from the weights posterior \( p(a | y_{1:t}) \). By virtue of the Bayes theorem, this density can be expressed as
\[ p(a | y_{1:t}) = \frac{p(a) p(y_{1:t} | a)}{p(y_{1:t})} \]
and can be evaluated up to the normalising constant \( p(y_{1:t}) \); thus, we propose to sample from the posterior of \( a \) using MCMC.

The evaluation of \( p(y_{1:t} | a) \) requires a prior \( p(a) \), which can be e.g. Gaussian or uniform, and to compute the likelihood \( p(y_{1:t} | a) = \prod_{k=0}^{t-1} p(y_{k+1} | y_{1:k}, a) \), where
\[ p(y_{k+1} | y_{1:k}, a) = \int_{X} p(y_{k+1} | x_{k+1}, a) p(x_{k+1} | y_{1:k}, a) dx_{k+1}. \]

Recall that if the approximation of the filtering density \( p(x_k | y_{1:k}, a) \) is available in the (particle) form \( \{ x_k^{(j)}, w_k^{(j)} \} \), the predictive density can be approximated by
\[ p(x_{k+1} | y_{1:k}, a) \approx \sum_{j=1}^{N_p} w_k^{(j)} \delta_{x_{k+1}^{(j)}}(x_{k+1}) \]
where the samples \( x_{k+1}^{(j)} \sim p(x_{k+1} | x_k^{(j)}, a) \), since the state-transition density can be approximated using particle filters for the considered KSSM (conditional to \( a \)).

The particle approximation for the predictive density in eq. (10) is given by
\[ p(y_{k+1} | y_{1:k}, a) \approx \sum_{j=1}^{N_p} w_k^{(j)} p(y_{k+1} | x_{k+1}^{(j)}) \]
where we have used \( p(y_{k+1} | x_{k+1}, a) = p(y_{k+1} | x_{k+1}) \), since the observation function \( h \) is independent of the parameter \( a \), see eq. (7).

Although the resulting algorithm is an MCMC with a PF-approximated acceptance ratio, its convergence properties...
are guaranteed and follow from the pseudo-marginal MCMC approach \cite{20}, or more specifically, the particle MCMC (PMCMC) algorithm \cite{20}. These have shown that when the evaluation of the acceptance ratio requires sampling of latent variables (i.e., the state in our case) the sampling algorithm can be seen as an MCMC operating on the joint space of the parameters and the state with an exact acceptance ratio. To find the marginal posterior of the parameters, the state is then integrated out. The effect of the number of particles \(N_p\) is also very clear in the algorithm: the MCMC converges to the true posterior as long as the PF approximation converges to the true prior. PMCMC algorithm \cite{40} have shown that when the approximation converges to the true prior and the true posterior decreases inversely proportional to \(N_p\). As a result, the posterior of the mixing parameters can be obtained through MCMC \cite{20}.

\begin{equation}
\hat{p}(a|y_{1:t}) \propto \sum_{i=1}^{N_p} \delta_{a(i)}(a) \tag{13}
\end{equation}

where the samples \(\{a(i)\}_{i=1:N_p}\) are obtained through MCMC sampling as described above. Consequently, the posterior mean of the KSSM transition function is given by

\begin{equation}
f_s = \sum_{i=1}^{N_p} \left( \frac{1}{N_p} \sum_{i=1}^{N_p} a(i) \right) K(s^t, \cdot). \tag{14}
\end{equation}

The pseudocode for the proposed method using Metropolis-Hastings MCMC is given in Algorithm 1.

\begin{algorithm}
\caption{Draw \(S\) samples from the posterior density \(p(a|y_{1:T})\) using Metropolis-Hastings MCMC}
\begin{algorithmic}
\STATE 1: INPUT: Observations \(y_{1:T}\), support vectors \(\{s^t\}_{i=1:N}\), kernel width \(\sigma\) and initial particles \(\{x_0(j)\}_{j=1:N_p}\).
\STATE 2: Set: MCMC move \(q(a^{(l+1)}|a^{(l)})\), first sample: \(a^{(1)} \sim q(a|0)\), prior \(p(a)\), and sample number \(l = 1\).
\STATE 3: while \(i < S\) do
\STATE 4: Propose a candidate move: \(a^{(c)} \sim q(a|a^{(l)})\)
\STATE 5: for all \(k = 1 : t\) do
\STATE 6: Approximate \(p(x_{k-1}|y_{1:k-1}, a^{(c)})\) using \(N_p\) weighted particles \(\{x_{k-1}(j), w_{k-1}(j)\}\) obtained by a particle filter.
\STATE 7: for all \(x_{k-1}(j), j = 1 : N_p\) do
\STATE 8: Sample \(x_k(j) \sim p(x_k|x_{k-1}, a^{(c)})\)
\STATE 9: end for
\STATE 10: Compute \(\hat{p}(y_k|y_{1:k-1}, a^{(c)}) = \sum_{j=1}^{N_p} w_{k-1}(j)p(y_k|x_{k}(j))\)
\STATE 11: end for
\STATE 12: Compute \(\hat{p}(a^{(c)}|y_{1:t}) = p(a^{(c)}) \prod_{k=1}^{t} \hat{p}(y_k|y_{1:k-1}, a^{(c)})\)
\STATE 13: Set \(a^{l+1} = a^{(c)}\) and \(l = l + 1\) with probability \(A = \min\{1, \frac{p(a^{(c)}|y_{1:t})q(a^{(l)}|a^{(c)})}{p(a^{(c)}|y_{1:t})q(a^{(l)}|a^{(c)})} \}\)
\STATE 14: end while
\end{algorithmic}
\end{algorithm}

B. Choice of Support Vectors and Kernel Width

The support vectors can be chosen based on the observations \(y_{i\in\mathbb{N}}\) and the sensor function \(h(\cdot)\). For each sample \(y_t\), the (known) observation equation of the KSSM in eq. (7) allows us to compute a maximum likelihood estimate (MLE) of \(x_t\) (conditional to \(y_t\)) by \(\tilde{x}_t = \text{argmax} p(X_t|y_t)\). For a sequence of observations \(y_{1:T}\), this procedure provides a collection of estimates for the state, given by \(\tilde{x}_{1:T}\), which, albeit not reliable as a filtering estimate \cite{22} provides insight into where the state can be found. We can now apply standard sparsification criteria from kernel adaptive filtering, such as approximate linear dependence or the coherence criterion, to the sequence \(\tilde{x}_{1:T}\) so as to choose the support vectors. Notice that standard kernel adaptive filters operate directly on the MLE sequence to find the mixing weights, as they do not cater for different signal-evolution and observation stages (and noises), as a consequence, their point estimates becomes unreliable for increasingly noisy signals.

Observe that the support vectors can also be found in a Bayesian fashion together with the mixing parameters, however, this would require using reversible jump MCMC methods, thus increasing the computational complexity of the overall algorithm. On the other hand, the proposed heuristic approach for the choice of support vectors exploits knowledge of the sensor function, is straightforward to implement, and has lower computational complexity.

After the dictionary is chosen, the kernel width can be set based on the desired smoothness of the estimate. An empirical approach to find a suitable kernel width is to analyse the distribution of the norm of the differences across the sequence \(\tilde{x}_{1:T}\), that is, \(\{||\tilde{x}_{t_1} - \tilde{x}_{t_2}||, t_1, t_2 = 1 : T\}\), and then choose a kernel width according to the spread of these variates. Furthermore, observe that due to the universal property of the Gaussian kernel, the performance of the kernel regression is not restricted to a particular value of the kernel width \cite{23}.

C. Example: Estimation of a Nonlinear State-Transition Function

The following example provides an insight into the proposed algorithm. Consider the system

\begin{equation}
X_t = 10\text{sinc}\left(\frac{X_{t-1}}{7}\right) + W_t \tag{15}
\end{equation}

\begin{equation}
Y_t = X_t + V_t
\end{equation}

where the state and observation noise variances are \(\sigma_x^2 = 4\) and \(\sigma_y^2 = 4\), and \(X_1 \sim \mathcal{N}(0, 10)\). This SSM state-transition function was chosen because its state is bounded (as the norm of \(f(x) = 10\text{sinc}(x/7)\) vanishes for \(x \to \infty\)). Also, this system does not converge, as the sequence \(X_{t+1} = f(X_t)\) has two accumulation points\footnote{Due to not incorporating the state dynamics in eq. (15) but the observation eq. (17) only.} about \(-1.7\) and \(8.7\). We then parametrised the system \(X_{t+1} = f(X_t)\) using a KSSM and estimated the mixing parameters of the kernel-based transition function \(f_\nu = \sum_{i=1}^{N} a_i K(\cdot, s^t)\) as explained in Section III-A.

\footnote{Recall that \(\tilde{x}\) is an accumulation point of the sequence \(\{X_t\}_{t\in\mathbb{N}}\) if and only if any neighbourhood of \(\tilde{x}\) contains infinite elements of \(\{X_t\}_{t\in\mathbb{N}}\).}
We considered 40 observations of the process $Y_t$ denoted by $y_{1:40}$, shown in Fig. 1 and set the support vectors according to Section III-B. Furthermore, the kernel width was $\sigma^2 = 10$, and we assumed a uniform prior $p(\alpha)$, and a proposal density for the MCMC moves given by $p(\alpha^{(i)}|\alpha^{(r)}) = N(\alpha^{(r)}, L)$, with the square-exponential covariance matrix

$$L(a_i, a_j) = 0.2^2 \exp \left(-0.2||a_i - a_j||^2\right).$$  \hspace{1cm} (16)

This candidate proposal allows the MCMC moves to be smooth.

The kernel estimate was then computed according to Algorithm 1, 400 samples were drawn from $p(\alpha|y_{1:40})$ and only the last 200 were considered to compute the estimates (thus allowing the chain to converge). Fig. 2 shows the true SSM transition function, the hidden state samples corresponding to the considered time period, and the posterior mean of the KSSM transition function ($f_*$) with its one-standard-deviation confidence interval. Observe that the mean estimate $f_*$ matches the true underlying transition function for the regions of the state space where the hidden state resides. The posterior variance of the kernel estimate is also consistent with the spread of the unobserved samples, as for the regions where the state samples were more disperse, the estimate of the posterior variance was larger.

Recall that MCMC sampling generates a sequence of samples that move to areas of high probability, and then explore such areas. Fig. 3 shows the value of the posterior $p(\alpha|y_{1:40})$ for the supervised solution and the samples generated by the proposed method.

**Remark 1.** The proposed method is capable of learning the state-transition function in a localised manner for regions containing state samples. Such an approach is unsupervised and provides a sequence of estimates (i.e., samples of the posterior of $f_*$) for which the posterior probability is similar to that of the least-squares supervised solution.

For filtering applications, the estimate of the SSM needs to be sequentially updated so that it is always representative of the region of operation. We now introduce an adaptive version of the proposed algorithm that incorporates new observations to learn the state-transition function in a recursive fashion.

**IV. ONLINE UPDATE OF THE KSSM MODEL**

To model time-varying state-transition functions when the observations $y_t \in \mathbb{N}$ arrive sequentially, we propose the kernel-based time-varying SSM of the form

$$X_{t+1} = \sum_{s=1}^{N} a_{s,t} K(s^i, X_t) + W_t,$$  \hspace{1cm} (17)

$$Y_t = h(X_t) + V_t$$  \hspace{1cm} (18)

where $\mathbf{a}_t = [a_{1,t}, \ldots, a_{N,t}]^T$ is the vector of mixing parameters at time instant $t$.

**Remark 2.** Observe that within the time-varying KSSM, posterior inference on the mixing parameters should be addressed by targeting the density $p(\mathbf{a}_t|y_{1:t+1})$, since $\mathbf{a}_t$ will only be available through $x_{t+1}$, and consequently, through $y_{t+1}$ and not $y_t$—see eqs. (17)-(18).
A. Recursive Sampling From \( p(\mathbf{a}_t | y_{1:t+1}) \)

To find a recursive expression for the posterior \( p(\mathbf{a}_t | y_{1:t+1}) \), we assume: (i) a prior density for the transition of the weights \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) \), and (ii) a particle approximation for \( p(a_{t-1} | y_{1:t}) \)—such as eq. (13). This leads to the following recursion

\[
p(\mathbf{a}_t | y_{1:t+1}) = \frac{p(y_{t+1} | y_{1:t}, \mathbf{a}_t)}{p(y_{t+1} | y_{1:t})} p(\mathbf{a}_t | y_{1:t})
\]

\[
= \frac{p(y_{t+1} | y_{1:t}, \mathbf{a}_t)}{p(y_{t+1} | y_{1:t})} \sum_{\mathbf{a}_{t-1}} p(\mathbf{a}_t | \mathbf{a}_{t-1}, y_{1:t}) p(\mathbf{a}_{t-1} | y_{1:t}) d\mathbf{a}_{t-1}
\]

where, based on Remark 2, we can write \( p(\mathbf{a}_t | \mathbf{a}_{t-1}, y_{1:t}) = p(\mathbf{a}_t | \mathbf{a}_{t-1}) \) since, conditional to \( \mathbf{a}_{t-1} \), the sequence \( y_{1:t} \) does not provide posterior evidence for \( \mathbf{a}_t \).

Eq. (19) gives a recursive expression for the posterior in integral form; therefore, using the particle approximation \( p(\mathbf{a}_{t-1} | y_{1:t}) \approx \frac{1}{N_a} \sum_{l=1}^{N_a} \delta_{\mathbf{a}_{t-1}^{(l)}}(\mathbf{a}_{t-1}) \), yields the estimate

\[
\hat{p}(\mathbf{a}_t | y_{1:t+1}) = \frac{p(y_{t+1} | y_{1:t}, \mathbf{a}_t)}{N_a p(y_{t+1} | y_{1:t})} \sum_{l=1}^{N_a} p(\mathbf{a}_t | \mathbf{a}_{t-1}^{(l)}).
\]

With the \( N_a \) samples \( \{ \mathbf{a}_{t-1}^{(l)} \} \sim p(\mathbf{a}_{t-1} | y_{1:t}) \) available from the previous step, the evaluation of (20) is straightforward up to a normalising constant, since \( p(y_{t+1} | y_{1:t}, \mathbf{a}_t) \) can be approximated by eq. (12) and the prior \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) \) is assumed to be known. Therefore, we can also sample from (20) using a pseudo-marginal MCMC approach. The proposed recursive method is outlined in a pseudo-code form in Algorithm 2.

B. Online Sparsification and Choice of the Prior \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) \)

We next consider kernel adaptive filtering sparsification, where, for every observation \( y_t \), we compute the MLE of the state given by \( \tilde{x}_t \) to assess whether the state is in a region covered by the current dictionary; we then include \( \tilde{x}_t \) as a support vector based on either the ALD or coherence criterion. Notice that this represents an adaptive version of the sparsification procedure explained in Section 11.1.

The proposed algorithm admits a flexible design of the prior transition for the mixing weights \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) \). One alternative, based on whether a new sample \( s = \tilde{x}_t \) is added to the dictionary at time \( t \), can be employed as follows.

1) Sample \( s = \tilde{x}_t \) is not added: When the dictionary is not modified, we assume \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) = \mathcal{N}(\mathbf{a}_t; \mathbf{a}_{t-1}, \Sigma) \) where \( \Sigma \) is a square-exponential covariance matrix that ensures smooth transitions moves; we can then write

\[
\hat{p}(\mathbf{a}_t | y_{1:t+1}) \propto p(y_{t+1} | y_{1:t}, \mathbf{a}_t) \sum_{l=1}^{N_a} \mathcal{N}(\mathbf{a}_t; \mathbf{a}_{t-1}^{(l)}, \Sigma)
\]

where \( p(y_{t+1} | y_{1:t}, \mathbf{a}_t) \) is given by eq. (10).

The MCMC candidate can then be drawn from a Gaussian density

\[
\mathbf{a}_t^{(c)} \sim \mathcal{N}(\mu, L_t)
\]

where the mean \( \mu \) can be a randomly-selected particle of the previous estimate, i.e., \( \mu = \mathbf{a}_{t-1}^{(r)}, r \in [1, N_a] \), or an initial guess for the new weight computed by another algorithm (such as KLMS). Furthermore, the variance of the proposal can be chosen to be proportional to the likelihood of \( X_t \) with respect to the observation \( y_t \), that is

\[
L_t = \text{diag} \left( \{ p(X_t = s^1 | y_t), \ldots, p(X_t = s^m | y_t) \} \right).
\]

By choosing the MCMC moves in this way, the mixing of the chain can be accelerated if a reliable guess for the mixing weight is available, while at the same time, the covariance of the proposal allows to explore zones from where the latest state sample may have been generated.

2) Sample \( s = \tilde{x}_t \) is added: When the support vector \( s^{N+1} \) is added to the dictionary, the dimension of the mixing weights increases to \((N + 1)\). To facilitate learning in the region where the new support vector has been chosen, the prior \( p(\mathbf{a}_t | \mathbf{a}_{t-1}) \) is designed as follows: the first \( N \) components of the mixing weights are chosen uniformly from the set of samples generated in the previous step \( \{ \mathbf{a}_{t-1}^{(l)} \} \sim p(\mathbf{a}_{t-1} | y_{1:t}) \), and the new coefficient, \( a_{N+1,t} \), is chosen to be uniformly distributed and independent from the previous coefficients, thus \( p(a_{N+1,t} | \mathbf{a}_{t-1}) \propto 1 \). In other words,

\[
\hat{p}(\mathbf{a}_t | y_{1:t+1}) \propto p(y_{t+1} | y_{1:t}, \mathbf{a}_t) \sum_{l=1}^{N_a} \delta_{\mathbf{a}_{t-1}^{(l)}} \left( \begin{array}{c} a_{1,t} \\ \vdots \\ a_{N+1,t} \end{array} \right)
\]

We then consider the MCMC move

\[
\left[ \begin{array}{c} a_{1,t}^{(c)} \\ \vdots \\ a_{N+1,t}^{(c)} \end{array} \right] \sim \sum_{l=1}^{N_a} \delta_{\mathbf{a}_{t-1}^{(l)}} \left( \begin{array}{c} a_{1,t} \\ \vdots \\ a_{N+1,t} \end{array} \right) \quad \text{and} \quad a_{N+1,t}^{(c)} \sim \mathcal{N}(\mu, v^2)
\]

where the first \( N \) parameters are sampled from the discrete uniform distribution and the new parameter is sampled independently from the rest of the parameters and is given by a normal density of mean \( \mu \) and variance \( v^2 \).

Observe that for samples generated according to (25) the evaluation of the target distribution in (24) is straightforward and is given by

\[
\hat{p}(\mathbf{a}_t | y_{1:t+1}) \propto p(y_{t+1} | y_{1:t}, \mathbf{a}_t).
\]

This MCMC proposal move aims to retain previous knowledge by sampling from the past estimates for the first \( N \) entries of the weights, while at the same time exploring values for the new parameter by setting uniform priors on the new samples.

C. Example: Identification of a Time-Varying State-Transition Function

We now test the suitability of the proposed algorithm for online learning of a nonlinear state-transition function. Consider the time-varying SSM with an affine observation function given by

\[
X_{t+1} = f_t(X_t) + W_t
\]

\[
Y_t = 5 + X_t/2 + V_t
\]
Fig. 4: The state-transition function of the system in (27) is time-varying between $t = 30$ and $t = 60$, and constant for $t < 30$ (system stable) and for $t > 60$ (system unstable).

where the variances of the state and observation noises were set to $\sigma_x^2 = 1$ and $\sigma_y^2 = 0.5$, $x_1 \sim \mathcal{N}(0, 1)$, and 

$$f_t(x) = \begin{cases} \frac{x}{2} + \frac{25x}{1 + x^2}, & t < 30 \\ \frac{60-t}{30} \left( \frac{x}{2} + \frac{25x}{1 + x^2} \right) + \frac{t-30}{30} \cdot 10 \text{sinc} \left( \frac{x}{2} \right), & 30 \leq t \leq 60 \\ 10 \text{sinc} \left( \frac{x}{2} \right), & t > 60 \end{cases}$$

(28)

This time-varying function is motivated by real-world applications where systems switch between different operation conditions. In this example, the system is time-invariant and stable for $t \in [1, 30]$, time-varying for $t \in [31, 60]$, and time-invariant with two accumulation points for $t \in [61, 90]$ (as in Example 1). This case is typical in anomaly detection scenarios, where an early identification of the data relationship is crucial. The evolution of the state-transition function is shown in Fig. 4.

We approximated the function $f_t$ in a parametric manner by $f_{\mathbf{a}_t}(\cdot) = \sum_{i=1}^{N} a_{i,t}K(\cdot, s^i)$, where the mixing weights were computed in a recursive fashion according to Section IV-A and Algorithm 2. The procedures for choosing the support vectors and the MCMC candidates are given in Section IV-B. Furthermore, for fast convergence of the chain, the parameter $\mu$ in (25) was chosen using optimal least squares fit of the MLE samples $\hat{x}_t$.

Similarly to the offline learning case in the previous section, the recursive estimate provided by the proposed KSSM is localised, since kernel adaptive filtering sparsification criteria only allow for learning the transition function in regions visited by the hidden state—see eq. (12). In this sense, attempting to find a full-support estimate of the true transition function in Fig. 3 is unrealistic, therefore, the proposed method aims to find an estimate of the transition function that is consistent with the observed sequence, even if it only learns the state dynamics for a limited region of the state space.

Recall that the proposed algorithm performs joint parameter identification and state estimation, as the samples from the filtering density are needed to sample from the posterior of the mixing parameters—see eq. (12). The filtered state signal resulting from the function estimation implementation is shown in Fig. 6 (mean and standard deviation), together with the true state.
Fig. 6: Filtered state signal using the KSSM and SIR particle filter. The original state is shown in blue and the posterior mean in red, with a one-standard-deviation confidence interval in light red.

V. LEARNING AND PREDICTION OF REAL-WORLD TIME SERIES

Kernel adaptive filters perform prediction of time series through nonlinear autoregressive modelling based on a fixed number of delayed samples. This task can be naturally addressed using SSMs by considering the delayed samples of the time series as the hidden state of the SSM. We now introduce a multivariate version of the proposed KSSM that performs autoregressive modelling and prediction in the context of frequency estimation of a power system.

A. Multivariate KSSM for Autoregressive Modelling

Recall that the stochastic process satisfying the \( \tau \)-order difference equation \( X_{t+1} = f_t(X_t, \ldots, X_{t-\tau+1}) + W_t \) can be expressed as a first-order multivariate process of the form

\[
\tilde{X}_{t+1} = F_t(\tilde{X}_t) + G_t
\]

where \( \tilde{X}_t = [X_t, \ldots, X_{t-\tau+1}]^T \),

\[
F_t(\tilde{X}_t) = \begin{bmatrix} f_t(X_t, \ldots, X_{t-\tau+1}) & \vdots & \vdots & \vdots \\ X_t & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{t-\tau+2} & \vdots & \vdots & \vdots \end{bmatrix}, \quad \text{and} \quad G_t = \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}.
\]

The process \( \tilde{X}_t \) can then be considered as the hidden state of an SSM, thus yielding a SSM formulation of higher-order autoregressive process with noisy observations. Furthermore, if the state-transition function \( f_t \) is modelled using kernels, we obtain a multivariate version of the proposed KSSM suited for autoregressive time series given by

\[
\begin{bmatrix} X_{t+1} \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} a_{i,t} K(s^t, \tilde{X}_t) \\ W_t \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\]

where \( Y_t = h(\tilde{X}_t) + V_t \)

\[
Y_t = h(\tilde{X}_t) + V_t
\]

Remark 3. The proposed KSSM method can be used for the design of nonlinear higher-order autoregressive models. Observe that the order of the process \( \tau \) and the number of support vectors \( N \) are independent design parameters which can be chosen, respectively, according to the dynamics of the signal and the desired accuracy. As a consequence, the number of parameters to find is given by the number of support vectors and not by the order of the process.

B. Frequency Estimation using KSSM

Frequency estimation is a key paradigm in Signal Processing and has been used in Smart Grids applications [51], where it allows for the planning and control of grid operation.

We considered the frequency signal of the UK national grid\(^4\) for the day 17 July 2014, where measurements became available every five minutes. The frequency data, originally in the region [49.85 Hz, 50.15 Hz], were scaled according to

Scaled frequency = 50(Raw frequency – 50) \hfill (33)

for simplicity of presentation; the scaled frequency was then in the region [-8,8] and had an average power equal to \( \mathbb{E}[x_t^2] = 4.593 \). The scaled frequency was modelled, according to eq. (31), by the hidden process of a KSSM of order two, and the observation process was created by adding Gaussian noise of standard deviation \( \sigma_y = 1 \) to the state. For fast convergence of KSSM, the proposal density for the MCMC sampler was set to the value of the weights found by KLMS applied to the noisy observation sequence \( y_{1:T} \). The aim of this experiment was to recursively approximate the transition function of the frequency process, using the method in Section IV and to perform a one-step ahead prediction.

Fig. 7 shows the estimates of the transition function for time instants \( t \in [2,7] \), together with the state samples until the corresponding time step \( \mathbb{E}[x_t^2] \). Notice that support vectors were included for \( t \in [2,5] \), and that for these time steps the learnt mapping corresponded to a smooth mapping from \( \mathbb{R}^2 \) to \( \mathbb{R} \) that represented the hidden state samples. Furthermore, observe that for \( t \in [6,7] \) (where support vectors were not added) the learnt mapping remained robust to the contribution of new samples. As in the synthetic examples in the previous sections, the estimates were updated only locally, this is a consequence of the MCMC move densities presented in Section IV-B.

To facilitate online prediction, the transition prior of the parameter set was set to be uniform, thus accepting samples with high likelihood which capture the instantaneous dynamics of the signal—see eq. (20).

The proposed KSSM was also implemented for Bayesian prediction of the frequency signal. We considered 200 samples, corresponding to approximately 16.7 hours, and validated the KSSM in the one-step-ahead prediction setting. The prediction algorithms considered were:

**Persistent Estimation:** The prediction at time \( t \) is simply the previous value of the observed process \( y_{t-1} \).

**Kernel Normalised Least Mean Square (KLMS):** A standard in kernel adaptive filtering, where the estimate is produced by performing nonlinear regression directly on the observation signal \( y_t \) using kernels and an LMS-based update rule; see [45].

\(^4\)Data available from [http://www.gridwatch.templar.co.uk/](http://www.gridwatch.templar.co.uk/)

\(^5\)The case \( t = 1 \) is omitted since at least two observations are needed to perform the inference on the mixing parameters—see Remark 2.
Kernel State-Space Model (KSSM): The adaptive version of the proposed method, where the predictions are generated by propagating the particles of the state according to the estimated transition function. The mean of the MCMC proposal was set to the weights found by KLMS; furthermore, a sequential importance resampling (SIR) [52] particle filter with stratified sampling was considered.

For a meaningful comparison, both KLMS and KSSM used the same dictionary based on the coherence criterion [45], the same sequence of maximum likelihood estimates \( \{ \max_r p(x_t | y_t) \}_{t=1:200} \), and the same kernel width \( \sigma = 2 \).

Fig. 8 shows the one-step-ahead predictions using the considered kernel algorithms against the true frequency signal (top), and the number of support vectors (bottom). Observe that the kernel predictions were fairly inaccurate for \( t < 50 \), as not enough samples had been processed, and consequently a large volume of support vectors were added during this period. We can therefore consider this period as the transient of the kernel adaptive estimators. As \( t \) increased, the kernel predictions became progressively better, in particular, the one-standard-deviation confidence interval of the KSSM prediction also became larger to include the true frequency signal.

The steady-state prediction performance was assessed for different levels of observation noise, over 100 realisations, in terms of the average mean square error (MSE) for \( t \in [100, 200] \)

\[
\text{Average MSE} = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{T-T_0} \sum_{t=T_0}^{T} (x_t - x_t^r(t-1))^2 \tag{34}
\]

Fig. 9 shows the average steady-state MSE for all three considered algorithms with respect to observation noise and the power of the signal to give intuition as to when the estimates become inaccurate due to the large observation noise. Both

\[ R = 100 \] is the number of realisations, \( T_0 = 100 \) and \( T = 200 \) are the initial and final time steps considered for the MSE respectively, and \( x_t^r(t-1) \) is the \( r \)-th realisation of the one-step-ahead prediction of the frequency signal. For each realisation, the observed signal was recreated by adding zero mean Gaussian noise of standard deviation in the range \( \sigma_y \in [0.25, 2] \).

Fig. 9 shows the average steady-state MSE for all three considered algorithms with respect to observation noise and the power of the signal to give intuition as to when the estimates become inaccurate due to the large observation noise. Both
kernel predictors outperformed the persistent estimate in all cases, confirming that the kernel-based predictors are indeed capable of capturing the nonlinear dynamics of the signal. Observe that KSSM, while considering candidate weights close to the KLMS weights, outperformed KLMS by incorporating the prior on the existence of observation disturbances, a model property that is absent in the KLMS formulation. Furthermore, notice how the MSE of both algorithms approached the power of the hidden state as the observation noise increased, after this point any algorithm becomes irrelevant since their estimate would be worse than setting the estimate to zero. Finally, recall that the KSSM-based particle filter allows for placing error bars on the estimate (as seen in fig. 8), whereas KLMS provides point estimates only. This illustrates the ability of the proposed KSSM algorithm to design meaningful dynamic models in an unsupervised learning setting, since the estimated model not only outperforms the standard in kernel adaptive filtering (KLMS), but also give a full probabilistic description of the nonlinear dynamics.

VI. DISCUSSION

The proposed KSSM paradigm for unsupervised learning of state-space models is flexible and admits different criteria for finding the mixing parameters, designing PF and MCMC stages, and setting hyperparameters. We now further elaborate on some key steps within the proposed approach.

A. Criterion to Find the Mixing Parameters

As explained in Section III-A, we performed parameter estimation by targeting the posterior density \( p(a|y_{1:t}) \). This allows us to impose desired properties of the solution, such as e.g. regularisation, through the prior density \( p(a) \), and then to use the observations to compute the pdf of the mixing weights. An alternative approach is to find the maximum likelihood of the weights, this is achieved by maximising \( p(y_{1:t}|a) \) and then replacing the random vector \( a \) in eq. (4) by the maximum likelihood estimate rather than integrating out the parameters as in eq. (5). This dilemma is common to many problems in Bayesian inference [33] and either alternative has advantages and disadvantages. The posterior density approach gives a complete pdf of the estimated transition function at an increased computational cost, whereas the maximum likelihood approach is of lower complexity but gives only a point estimate.

B. Candidate Move for Markov Chain Monte Carlo

We considered a Metropolis-Hasting MCMC sampling from the posterior \( p(a|y_{1:t}) \), where the candidate move plays an important role in both convergence of the chain and the area it explores. To boost the convergence of the chain, we can consider the sequence of maximum likelihood estimates of the state \( \hat{x}_{t\in N} \) and perform supervised learning of this signal—we can then propose the MCMC moves in this region. This concept was applied in the above online simulations, as it provided short convergence time and explored a localised region of the parameter space.


The standard in Bayesian inference is to define prior densities for all the unknown parameters and then find their posterior with respect to the observed data, this includes model orders and variances. Although this leads to a full Bayesian model where all the quantities follow from the observations, in practice this approach can be prohibitively expensive, especially when the dimension of the parameters needs to be determined (as it is the case with the choice of support vectors). To this end, we considered concepts from kernel adaptive filtering so as to reduce the computational complexity of our method; in particular, the support vectors were chosen by sparsifying the maximum likelihood estimates of the state \( \hat{x}_{t\in N} \) and the kernel width and noise variances by trial-and-error. We have shown that this approach is well suited for real-world signals and is also in line with the KSSM formulation since (i) the performance of kernel regression is robust to the choice of the kernel width [23] and (ii) the misadjustment of the choice of the noise variance can be corrected by the (posterior) variance of the mixing weights.

D. Estimation of the Filtering Density \( p(x_t|y_{1:t}) \)

Within the proposed KSSM, the filtering density is computed in order not only to perform joint system identification and state estimation, but also because such a density is needed to sample from the posterior \( p(a|y_{1:t}) \) using MCMC—see eq. (12). Although the simulations in this paper considered the classic SIR approach [52], any variant of particle filter can be used. In the online case, we can consider uninformative priors for the state when the current estimate is still in its transient [33], or risk-sensitive PF [54] so as to explore critical regions of the state-space; these are proven to be advantageous in the identification of critical state behaviour [55].

VII. CONCLUSIONS

We have proposed a novel framework for the design of state-space models by parametrising their (nonlinear) state-transition function using reproducing kernels and then finding the mixing parameters in an unsupervised fashion using a Monte Carlo methods.

The representer theorem allows us to express the optimal estimate of the state-transition function as a weighted sum of kernels evaluated on a set of support vectors. The proposed KSSM chooses the support vectors using sparsification criteria from kernel adaptive filtering (this provides a representative dictionary at low computational cost), and then finds the posterior of the mixing weights using a combination of MCMC and particle filters (the convergence properties of which are guaranteed by the pseudo-marginal MCMC approach). The resulting estimation method then provides probabilistic estimates for both the model and the underlying state process.

Through simulation examples on synthetic time series, we have first validated the proposed approach by comparing the likelihood of the estimates with those obtained via supervised learning, and by showing that the learnt transition function
resembles the (unobserved) state samples. We have then employed the KSSM for learning and prediction of real-world power-grid frequency signals, where the proposed method has deployed the KSSM for learning and prediction of real-world nonlinear drift, resembling the (unobserved) state samples. We have then employed the KSSM for learning and prediction of real-world nonlinear drift, resembling the (unobserved) state samples.

REFERENCES


