Unsupervised State-Space Modelling Using Reproducing Kernels

Felipe Tobar, Petar M. Djurić and Danilo P. Mandic

***Abstract*—A novel framework for the design of state-space models (SSMs) 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 their posterior density using Monte Carlo methods. We first introduce an offline version of the proposed algorithm, followed by an 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 show that the proposed algorithm outperforms kernel adaptive filters in the prediction of real-world time series, while also providing probabilistic estimates, a key advantage over standard methods.**

***Index Terms*—Support vector regression, system identification, 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 an 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 $\mathcal{M}$. This model is usually chosen from the class of state-space models (SSMs) with discrete time, continuous state and additive noise, where X_t is the Markovian state and Y_t the (noisy) observation. SSMs are expressed in the form

$$\begin{aligned} X_{t+1} &= f_t(X_t) + W_t, \\ Y_t &= h_t(X_t) + V_t \end{aligned} \tag{1}$$

where $t \in \mathbb{R}$, $X_t \in \mathbb{R}^n, Y_t \in \mathbb{R}^m$, the state-transition function $f_t : \mathbb{R}^n \to \mathbb{R}^n$, the sensor function $h_t : \mathbb{R}^n \to \mathbb{R}^m$, 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] to population models [3] and mathematical finance [4].

Felipe Tobar is with the Department of Engineering, University of Cambridge, UK (e-mail: fat25@cam.ac.uk), and acknowledges financial support to EPSRC grant number EP/L000776/1. Part of this work was carried out when he was a PhD student at the Department of Electrical and Electronic Engineering, Imperial College London, UK.

Petar M. Djurić is with the Department of Electrical and Computer Engineering, Stony Brook University, USA. His work was supported by the National Science Foundation under Award CCF - 1320626.

Danilo P. Mandic is with the Department of Electrical and Electronic Engineering, Imperial College London, UK.

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 [5], [6], 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 [7]) or those satisfying the Beneš condition (Beneš filter [8]). 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 $\mathcal{M}$ [9], [10].

The flexibility of PFs allows for approximating the posterior density $p(X_t|y_{1:t})$ with no rigid constrains 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 [11], SMC methods for training neural networks [12], and model order determination using reversible jump Markov chain Monte Carlo (MCMC) [13]. Low-complexity parameter identification in SSMs, suitable for online implementation, can be achieved by means of *artificial evolution* [14], that is, by considering the unknown parameters as states evolving according to a random walk. This concept provides accurate state estimates and has even been used in kernel-based SSMs [15]; however, it is well-known to result in noisy estimates due to the artificial noise injected into the system.

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 [16] have recently found application in system identification by performing nonparametric estimation of the underlying SSM for filtering and smoothing [17], [18]; however, these are not able to operate online. Kernel methods [19], [20] are particularly suited for this estimation task, as they are universal function approximators by their very nature [21] and, akin to neural networks [22], their centres and weights that can be trained to learn data relationships in regression settings. Kernel adaptive filters [23] employ the so-called *kernel trick* [24] to provide nonlinear extensions of linear adaptive filters, allowing for real-time nonlinear estimation at a linear increase in computational cost. However, existing kernel adaptive filters based on least mean square [25], [26], recursive least squares [27] and ridge regression [28], 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) [29]. 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) [30], [31], 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 [32], where [33], [34] 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; and (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 [35] 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 training set comprising available input-output data given by

$$S_N = \{(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^m \text{ s.t. } y_i = f(x_i)\}_{i=1:N}\,. \tag{2}$$

We approximate the function $f(x)$ by a function in the the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ [36] with reproducing kernel K. According to the representer theorem [37], 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* $\mathbf{a} = [a_i, \dots, a_N]$ in the form

$$f_{\mathbf{a}}(\cdot) = \sum_{i=1}^{N} a_i K(x_i, \cdot). \tag{3}$$

The centres of the kernel evaluations are referred to as *support vectors* and the approach to regression is known as support vector regression (SVR) [28].

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

When a large set of data is available to train the kernel estimator in (3), 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 [38] .

In kernel adaptive filtering [23] (the approach to nonlinear adaptive filtering using SVR), sparsification criteria include the approximate linear dependence (ALD) [39], which operates on the feature space and aims to avoid redundancies of feature samples, and the novelty criterion [40], [41], which only admits samples that (i) are distant enough from the current dictionary and (ii) improve the current estimate. Adaptive strategies include presence-based sparsification [26], 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) in 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

$$K(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{\sigma^2}\right) \tag{4}$$

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 [21]. In particular, the Gaussian kernel is proven to be an *universal kernel* [42, 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 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 observations with the knowledge of the sensor function (likelihood) and known physical constraints on the latent state. This assumption allows for a straightforward design of the dictionary.

Our aim is to estimate 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. (4). 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 $\{f_i \in \mathcal{H}\}_{i\in\mathbb{N}}$ that converges to f (see also [21], [42], [43]). Furthermore, the justification to express the optimal estimate (within $\mathcal{H}$) as a mixture of kernel evaluations follows 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 [44], kernel principal component analysis [45], and unsupervised multiple kernel learning [46].

By parametrising the transition function f as $f_{\mathbf{a}} = \sum_{i=1}^N a_i K(s^i, \cdot)$, the approximated SSM takes the form

$$X_{t+1} = \sum_{i=1}^N a_i K(s^i, X_t) + W_t \tag{5}$$

$$Y_t = h(X_t) + V_t. \tag{6}$$

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

The posterior density $p(f_{\mathbf{a}}|y_{1:t})$ is then uniquely determined by the posterior density of the weights $p(\mathbf{a}|y_{1:t})$; this allows us to find the transition-function posterior and, in particular, the conditional expectation $f_* = \mathbb{E}[f_{\mathbf{a}}|y_{1:t}]$ given by

$$f_* = \int_{\mathbb{R}^N} f_{\mathbf{a}} p(\mathbf{a}|y_{1:t}) \mathrm{d}\mathbf{a} = \sum_{i=1}^N \mathbb{E}[a_i|y_{1:t}] K(s^i, \cdot). \tag{7}$$

This means that f_* can be found by only computing $\mathbb{E}[\mathbf{a}|y_{1:t}]$, since $f_* = f_{\mathbb{E}[\mathbf{a}|y_{1:t}]}$.

We now investigate how to sample from the weights posterior $p(\mathbf{a}|y_{1:t})$. By virtue of the Bayes theorem, this density can be expressed as

$$p(\mathbf{a}|y_{1:t}) = p(y_{1:t}|\mathbf{a}) \frac{p(\mathbf{a})}{p(y_{1:t})} \tag{8}$$

and can be approximated up to the normalising constant $p(y_{1:t})$; thus, we propose to sample from the posterior of $\mathbf{a}$ using the Metropolis-Hastings algorithm [47], where candidate samples $\mathbf{a}^{(c)}$ are drawn from a known distribution and then accepted on the basis of an *acceptance ratio* which involves evaluating the target density $p(\mathbf{a}|y_{1:t})$.

The evaluation of (8) requires to assume a prior $p(\mathbf{a})$, which can be, e.g., Gaussian or uniform, and to compute the likelihood $p(y_{1:t}|\mathbf{a}) = \prod_{k=0}^{t-1} p(y_{k+1}|y_{1:k}, \mathbf{a})$, where

$$p(y_{k+1}|y_{1:k}, \mathbf{a}) = \int_X p(y_{k+1}|x_{k+1}, \mathbf{a}) p(x_{k+1}|y_{1:k}, \mathbf{a}) \mathrm{d}x_{k+1}. \tag{9}$$

Recall that if the approximation of the filtering density $p(x_k|y_{1:k}, \mathbf{a})$ is available in the (particle) form $\{x_k^{(j)}, w_k^{(j)}\}$, the predictive density can be approximated by

$$p(x_{k+1}|y_{1:k}, \mathbf{a}) \approx \sum_{j=1}^{N_p} w_k^{(j)} \delta_{x_{k+1}^{(j)}}(x_{k+1}) \tag{10}$$

where the samples $x_{k+1}^{(j)} \sim p(x_{k+1}|x_k^{(j)}, \mathbf{a})$, since the state-transition density of the KSSM is known (given $\mathbf{a}$).

The particle approximation for the density in eq. (9) is given by

$$p(y_{k+1}|y_{1:k}, \mathbf{a}) \approx \sum_{j=1}^{N_p} w_k^{(j)} p(y_{k+1}|x_{k+1}^{(j)}) \tag{11}$$

where we have used $p(y_{k+1}|x_{k+1}, \mathbf{a}) = p(y_{k+1}|x_{k+1})$, since the observation function h is independent of the parameter $\mathbf{a}$, see eq. (6).

As within this sampling strategy the evaluation of $p(\mathbf{a}|y_{1:t})$ is approximated, the presented algorithm is an MCMC with a PF-approximated acceptance ratio. In this context, observe that convergence to the target density $p(\mathbf{a}|y_{1:t})$ is guaranteed and follows from the *pseudo-marginal* MCMC approach [35], or more specifically, the particle MCMC (PMCMC) algorithm [35]. 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 *marginal* MCMC algorithm (which uses the true acceptance ratio) as long as the PF approximation converges to the true likelihood. This holds for the PF estimate, as the square error between expectations taken under the PF-approximated posterior and the true posterior decreases inversely proportional to the number of particles N_p.

As a result, the posterior of the mixing parameters can be approximated by the empirical density

$$\hat{p}\left(\mathbf{a}|y_{1:t}\right) \propto \sum_{l=1}^{N_{\mathbf{a}}} \delta_{\mathbf{a}^{(l)}}(\mathbf{a}) \tag{12}$$

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

$$f_* = \sum_{i=1}^{N}\left(\frac{1}{N_{\mathbf{a}}}\sum_{l=1}^{N_{\mathbf{a}}} a_i^{(l)}\right)K(s^i,\cdot). \tag{13}$$

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

Algorithm 1 Draw S samples from the posterior density $p(\mathbf{a}|y_{1:t})$ using Metropolis-Hastings MCMC

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

B. Choice of Support Vectors and Kernel Width

The support vectors can be chosen based on the observations $y_{t\in\mathbb{N}}$ and the sensor function $h(\cdot)$. For each sample y_t, the (known) observation equation of the KSSM in eq. (6) allows us to compute a maximum likelihood estimate (MLE) of x_t (conditional to y_t) by $\widehat{x}_t = \text{argmax } p(X_t|y_t)$. For a sequence of observations $y_{1:T}$, this procedure provides a collection of estimates for the state, given by $\widehat{x}_{1:T}$, which, albeit not reliable as a filtering estimate,[1] 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 $\widehat{x}_{1:T}$ 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 become 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 $\widehat{x}_{1:T}$, that is, $\{\|\widehat{x}_{t_1} - \widehat{x}_{t_2}\|, t_1, t_2 = 1 : T\}$, 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 [21].

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

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

$$\begin{aligned} X_t &= 10\text{sinc}\left(\frac{X_{t-1}}{7}\right) + W_t, \\ Y_t &= X_t + V_t \end{aligned} \tag{14}$$

where the state and observation noise variances are $\sigma_x^2 = 4$ and $\sigma_y^2 = 4$, and $X_0 \sim \mathcal{N}(0, 10)$. 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[2] about -1.7 and 8.7. We then

[1]Due to not incorporating the state dynamics in eq. (5) but the observation eq. (6) only.

[2]Recall that $\bar{x}$ is an accumulation point of the sequence $\{X_t\}_{t\in\mathbb{N}}$ if and only if any neighbourhood of $\bar{x}$ contains infinite elements of $\{X_t\}_{t\in\mathbb{N}}$.

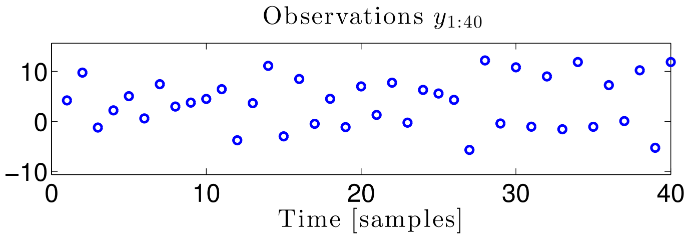

Fig. 1: Observed process for the system in eq. (14).

parametrised the system (14) using a KSSM and estimated the mixing parameters of the kernel-based transition function $f_{\mathbf{a}} = \sum_{i=1}^{N} a_i K(\cdot, s^i)$ as explained in Section III-A.

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, where $N = 7$ support vectors were chosen. Furthermore, the kernel width was $\sigma^2 = 10$, and we assumed a uniform prior $p(\mathbf{a})$, and a proposal density for the MCMC moves given by $p(\mathbf{a}^{(c)}|\mathbf{a}^{(l)}) = \mathcal{N}\left(\mathbf{a}^{(l)}, L\right)$, with the square-exponential[3] covariance matrix

$$L\left(a, a'\right) = 0.2^2 \exp\left(-0.2\|a - a'\|^2\right). \tag{15}$$

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 $p(\mathbf{a}|y_{1:40})$ 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 $p(\mathbf{a}|y_{1:40})$ for both the samples drawn using MCMC and the supervised least squares solution using the hidden samples (i.e., kernel least squares). Observe that the Markov chain converges in about 100 iterations to a zone of a probability similar to that of the supervised solution to then continue to draw samples of similar probability. Additionally, since a uniform prior $p(\mathbf{a})$ was assumed, the posterior is equal to the likelihood up to a normalising constant—see eq. (8). Fig. 3 illustrates that the supervised solution is different from the maximum likelihood estimate, as some of the samples drawn using the proposed method have a higher likelihood. This discrepancy arises from the fact that the supervised solution uses hidden-state samples whereas the proposed method does not.

[3]We refer to square exponential covariance functions and reserve the term Gaussian for reproducing kernels only.

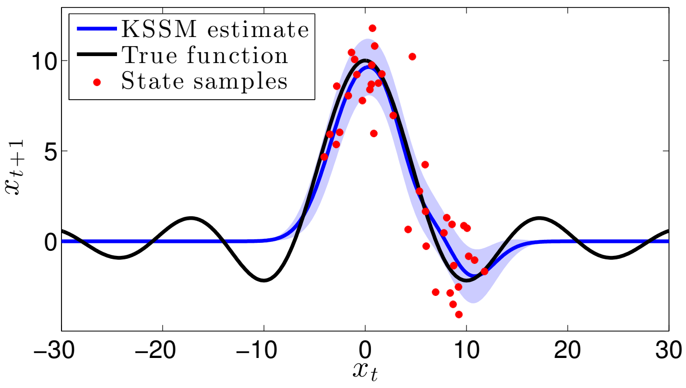

Fig. 2: Original state-transition function, state samples and kernel-based approximation.

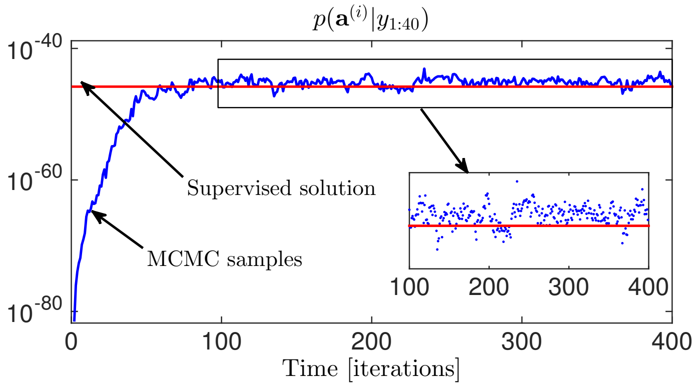

Fig. 3: Value of the posterior $p(\mathbf{a}|y_{1:40})$ for the supervised solution and the samples generated by the proposed method.

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 $f_{\mathbf{a}}$) 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

$$X_{t+1} = \sum_{i=1}^{N} a_{i,t} K(s^i, X_t) + W_t, \tag{16}$$

$$Y_t = h(X_t) + V_t \tag{17}$$

where $\mathbf{a}_t = [a_{1,t}, \dots, a_{N,t}]^T$ 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.* (16)-(17).

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 $p(\mathbf{a}_{t-1}|y_{1:t})$—such as eq. (12). This leads to the following recursion

$$\begin{aligned} p(\mathbf{a}_t|y_{1:t+1}) &= \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{p(y_{t+1}|y_{1:t})} p(\mathbf{a}_t|y_{1:t}) \\ &= \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{p(y_{t+1}|y_{1:t})} \int p(\mathbf{a}_t|\mathbf{a}_{t-1},y_{1:t}) p(\mathbf{a}_{t-1}|y_{1:t}) \mathrm{d}\mathbf{a}_{t-1} \end{aligned} \tag{18}$$

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, conditioned on $\mathbf{a}_{t-1}$, the sequence $y_{1:t}$ does not provide *posterior* evidence for $\mathbf{a}_t$.

Eq. (18) gives a recursive expression for the posterior in integral form; therefore, using the particle approximation $p(\mathbf{a}_{t-1}|y_{1:t}) \approx \frac{1}{N_\mathbf{a}} \sum_{l=1}^{N_\mathbf{a}} \delta_{\mathbf{a}_{t-1}^{(l)}}(\mathbf{a}_{t-1})$, yields the estimate

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) = \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{N_\mathbf{a} p(y_{t+1}|y_{1:t})} \sum_{l=1}^{N_\mathbf{a}} p(\mathbf{a}_t|\mathbf{a}_{t-1}^{(l)}). \tag{19}$$

With the $N_\mathbf{a}$ samples $\left\{\mathbf{a}_{t-1}^{(l)}\right\} \sim p(\mathbf{a}_{t-1}|y_{1:t})$ available from the previous step, the evaluation of (19) is straightforward up to a normalising constant, since $p(y_{t+1}|y_{1:t},\mathbf{a}_t)$ can be approximated by eq. (11) and the prior $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ is known. Therefore, we can also sample from (19) using a pseudo-marginal MCMC approach. The proposed recursive method is outlined in a pseudocode form in Algorithm 2.

Algorithm 2 Draw S samples from the sequence of posteriors $p(\mathbf{a}_t|y_{1:t+1}), t = 1, 2, \ldots$

```
 1: INPUT: Kernel width $\sigma$ and first observation $y_1$.
 2: Initialise: Dictionary $\{\widehat{x}_1\}$, kernel width $\sigma$, and particles
    $\{x_1^{(j)}\} \sim p(x_1|y_1)$ and $\{\mathbf{a}_1^{(l)}\} \sim p(\mathbf{a}_1)$.
 3: for all $y_t, t = 1, 2, \ldots$ do
 4:     Calculate $\widehat{x}_t$ = argmax $p(X_t|y_t)$
 5:     if $\widehat{x}_t$ deviates form dictionary then
 6:         Set $N = N + 1$, add $s^N = \widehat{x}_t$ into the dictionary
 7:         Draw $N_p$ samples from eq. (25) using MCMC and
            the proposal in (24)
 8:     else
 9:         Draw $N_p$ samples from eq. (20) using MCMC and
            the proposal in (21)
10:     end if
11: end for
```

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 $\widehat{x}_t$ to assess whether the state is in a region covered by the current dictionary; we then include $\widehat{x}_t$ 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 III-B.

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 $s = \widehat{x}_t$ is added to the dictionary at time t, can be employed as follows.

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

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) \propto p(y_{t+1}|y_{1:t},\mathbf{a}_t) \sum_{l=1}^{N_\mathbf{a}} \mathcal{N}(\mathbf{a}_t; \mathbf{a}_{t-1}^{(l)}, \mathbf{\Sigma}) \tag{20}$$

where $p(y_{t+1}|y_{1:t},\mathbf{a}_t)$ is given by eq. (9).

The MCMC candidate can then be drawn from a Gaussian density

$$\mathbf{a}_t^{(c)} \sim \mathcal{N}(\boldsymbol{\mu}, L_t) \tag{21}$$

where the mean $\boldsymbol{\mu}$ can be a randomly-selected particle of the previous estimate, i.e., $\boldsymbol{\mu} = \mathbf{a}_{t-1}^{(r)}, r \in [1, N_\mathbf{a}]$, 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

$$L_t = \text{diag}\left([p(X_t = s^1|y_t), \ldots, p(X_t = s^m|y_t)]\right). \tag{22}$$

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 $s = \widehat{x}_t$ 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 $\left\{\mathbf{a}_{t-1}^{(l)}\right\} \sim p(\mathbf{a}_{t-1}|y_{1:t})$, 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,

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) \propto p(y_{t+1}|y_{1:t},\mathbf{a}_t) \sum_{l=1}^{N_\mathbf{a}} \delta_{\mathbf{a}_{t-1}^{(l)}} \left(\begin{bmatrix} a_{1,t} \\ \vdots \\ a_{N,t} \end{bmatrix} \right). \tag{23}$$

We then consider the MCMC move

$$\begin{bmatrix} a_{1,t}^{(c)} \\ \vdots \\ a_{N,t}^{(c)} \end{bmatrix} \sim \sum_{l=1}^{N_\mathbf{a}} \delta_{\mathbf{a}_{t-1}^{(l)}} \left(\begin{bmatrix} a_{1,t} \\ \vdots \\ a_{N,t} \end{bmatrix} \right) \text{ and } a_{N+1,t}^{(c)} \sim \mathcal{N}(\mu, v) \tag{24}$$

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 μ and variance v^2.

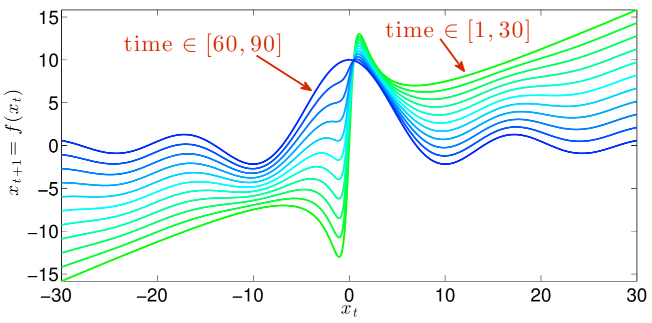

Fig. 4: The state-transition function of the system in (26) is time-varying between $t = 30$ and $t = 60$, and constant for $t < 30$ (system stable) and for $t > 60$ (system unstable).

Observe that for samples generated according to (24) the evaluation of the target distribution in (23) is straightforward and is given by

$$\hat{p}(\mathbf{a}_t | y_{1:t+1}) \propto p(y_{t+1} | y_{1:t}, \mathbf{a}_t). \tag{25}$$

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 a Gaussian prior on the new coefficient.

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

$$\begin{aligned} X_{t+1} &= f_t(X_t) + W_t \\ Y_t &= 5 + X_t/2 + V_t \end{aligned} \tag{26}$$

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

$$f_t(x) = \begin{cases} \frac{x}{2} + \frac{25x}{1+x^2}, & t < 30 \\ \frac{60-t}{30}\left(\frac{x}{2} + \frac{25x}{1+x^2}\right) + \frac{t-30}{30} 10\text{sinc}\left(\frac{x}{7}\right), & 30 \leq t \leq 60 \\ 10\text{sinc}\left(\frac{x}{7}\right), & t > 60. \end{cases} \tag{27}$$

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 μ in (24) was chosen using optimal least squares fit of the MLE samples $\widehat{x}_t$.

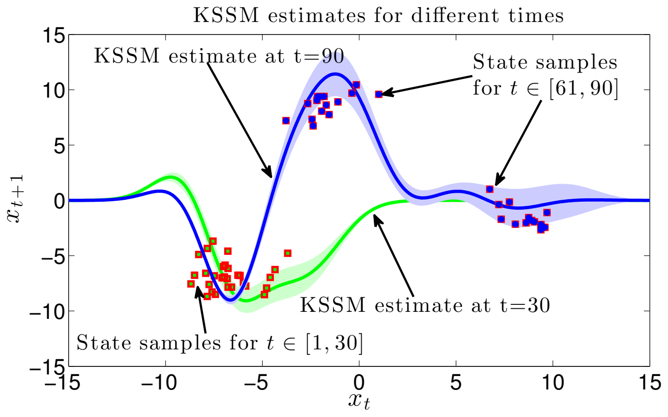

Fig. 5: KSSM estimate (mean and standard deviation) of the time-varying state-transition function in (27) for $t = 30$ and $t = 90$. The true state samples are plotted with red borders for the regions $[1, 30]$ (green fill) and $[61, 90]$ (blue fill).

Fig. 5 shows the so-learnt state-transition function at two different time instants, $t = 30$ and $t = 90$. Observe that the function learnt until $t = 30$ is very localised; this is due to the system state being stable and therefore residing within a limited region of the state space. As t grows larger, the transition function changes according to eq. (27), thus becoming time varying for $t \in [30, 60]$. In the last third of the analysed period, $t \in [60, 90]$, the transition function is again time invariant and the state lies on a larger region of the state space. Fig. 5 also shows the estimate at $t = 90$, where the KSSM successfully learnt the dependency of the state samples through the observation sequence. Notice that the estimate at time $t = 90$ resembles the data samples corresponding to the region and successfully updated the value of the function learnt with previous measurements (i.e., those for $t < 30$); furthermore, this estimate also shows the ability to retain useful information by not relying exclusively on new samples only, as the estimate at $t = 90$ still resembles the samples in $t \in [1, 30]$.

Similarly to the offline learning case in the previous section, the recursive estimate provided by the proposed KSSM is localised, since kernel adaptive filtering sparisifcation criteria only allow for learning the transition function in regions *visited* by the hidden state—see eq. (11). In this sense, attempting to find a full-support estimate of the true transition function in fig. 4 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. (11). The filtered state signal resulting from the function estimation implementation

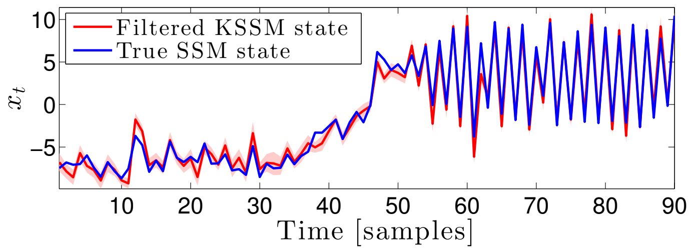

Fig. 6: Filtered state signal using the KSSM and SIR particle filter. The original state is shown in blue and the posterior mean in red, with a one-standard-deviation confidence interval in light red.

is shown in fig. 6 (mean and standard deviation), together with the true state.

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 prediction of the UK national grid.

A. Multivariate KSSM for Autoregressive Modelling

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

$$\bar{X}_{t+1} = F_t(\bar{X}_t) + G_t \tag{28}$$

where $\bar{X}_t = [X_t, \dots, X_{t-\tau+1}]^T$,

$$F_t(\bar{X}_t) = \begin{bmatrix} f_t(X_t, \dots, X_{t-\tau+1}) \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix}, \text{ and } G_t = \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{29}$$

The process $\bar{X}_t$ 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

$$\begin{bmatrix} X_{t+1} \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} a_{i,t} K(s^i, \bar{X}_t) \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix} + \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{30}$$

$$Y_t = h(\bar{X}_t) + V_t \tag{31}$$

where Y_t is the observed process, h_t is the observation function and V_t observation noise.

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 (τ) 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 [48], 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

$$\text{Scaled frequency} = 50(\text{Raw frequency} - 50) \tag{32}$$

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. (30), 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.[5] Notice that support vectors were included for all six time steps considered, where the learnt mapping corresponded to a smooth mapping form $\mathbb{R}^2$ to $\mathbb{R}$ that represented the hidden state 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 was set to be uniform, thus accepting samples with high likelihood which capture the instantaneous dynamics of the signal—see eq. (19).

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

[4]Data available from 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.

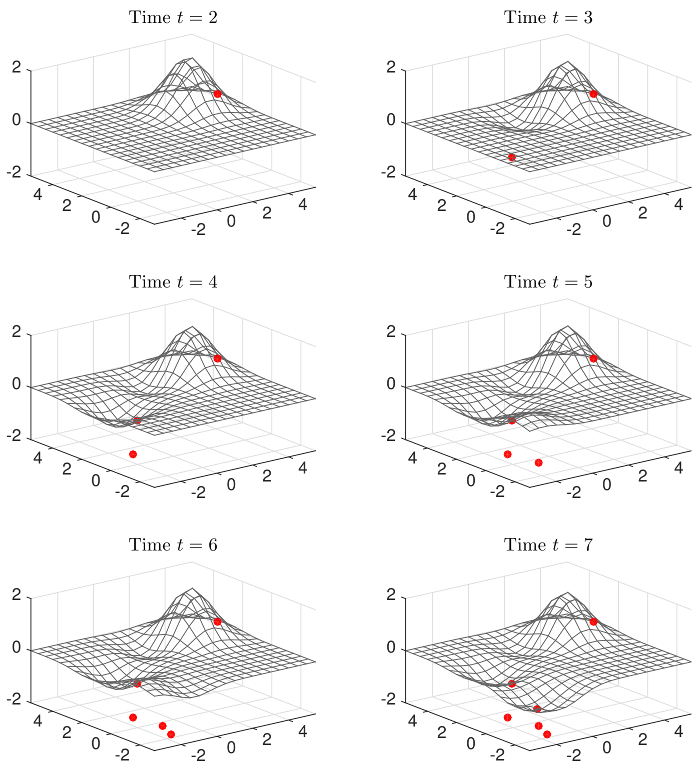

Fig. 7: Online estimation of the state-transition function for $t \in [2, 7]$. The hidden state samples are shown in red.

the observation signal y_t using kernels and an LMS-based update rule; see [41].

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) [49] particle filter with stratified sampling was considered.

For a meaningful comparison, both KLMS and KSSM used the same dictionary based on the coherence criterion [41], the same sequence of maximum likelihood estimates $\{\max_{x_t} p(x_t|y_t)\}_{t=1:200}$, 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 that 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 $t \in$

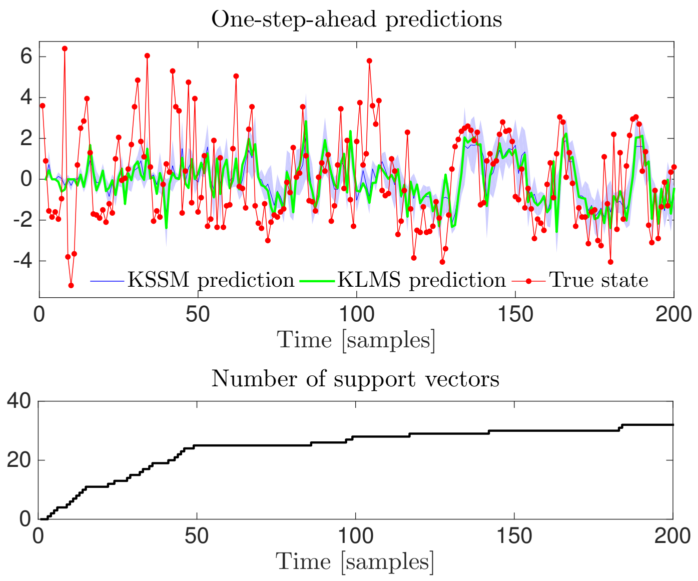

Fig. 8: Kernel predictions of the UK National Grid frequency. **[Top]** The original signal is shown in red, the KLMS prediction in green, and the KSSM prediction in blue with its one-standard-deviation confidence interval in light blue. **[Bottom]** Number of support vectors for kernel algorithms using the coherence sparsification criterion.

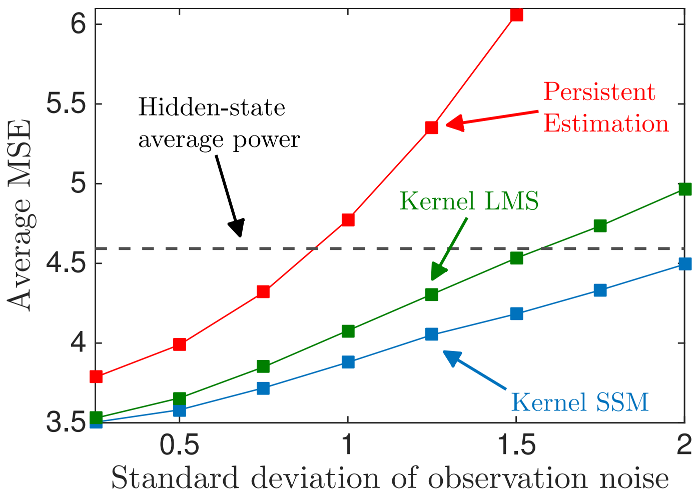

Fig. 9: Average prediction performance of the considered algorithms over 100 trials. The MSE was calculated in steady state, that is, for the last half of the estimation period (100 samples).

$[100, 200]$ defined by

$$\text{Average MSE} = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{T-T_0}\sum_{t=T_0}^{T}(x_t - x^r_{t|t-1})^2 \quad (33)$$

where $R = 100$ is the number of realisations, $T_0 = 100$ and $T = 200$ are the initial and final time steps considered for the MSE respectively, and $x^r_{t|t-1}$ is the rth 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 mixing weights given by KLMS, 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 $p(\mathbf{a}|y_{1:t})$. This allows us to impose desired properties on the solution through the prior density $p(\mathbf{a})$, such as regularisation, to then use the observations to approximate the posterior of the mixing weights. An alternative approach is to find the maximum likelihood estimate of the weights; this is achieved by maximising $p(y_{1:t}|\mathbf{a})$ and then replacing the unknown vector $\mathbf{a}$ in eq. (3) by the maximum likelihood estimate rather than *integrating out* the parameters as in eq. (7). This dilemma is common to many problems in Bayesian inference [50] 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 $p(\mathbf{a}|y_{1:t})$, 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 $\widehat{x}_{t\in\mathbb{N}}$ 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.

C. Choice of Hyperparameters: Noise Variances, Support Vectors and Kernel Width

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, which 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 $\widehat{x}_{t\in\mathbb{N}}$ 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 [21] 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 $p(\mathbf{a}|\mathbf{y}_{1:t})$ using MCMC—see eq. (11). Although the simulations in this paper considered the classic SIR approach [49], 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 [15], or risk-sensitive PF [51] so as to explore critical regions of the state-space; these are proven to be advantageous in the identification of critical state behaviour [52].

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 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 provided meaningful estimates of the transition functions and outperformed the deterministic KLMS for noisy observations, while also providing a full statistical prediction including confidence intervals for nonstationary time series in real time.

ACKNOWLEDGMENTS

The authors wish to thank Nilesh Tripuraneni and Matthew W. Hoffman for enlightening discussions on particle MCMC, and the anonymous reviewers for constructive feedback.

REFERENCES

[1] A. Bain and D. Crisan, *Fundamentals of Stochastic Filtering*. Springer, 2009.
[2] K. J. Åström, *Introduction to Stochastic Control Theory*. Courier Dover Publications, 2012.
[3] H. Iijima, T. Nagaike, and T. Honda, "Estimation of deer population dynamics using a Bayesian state-space model with multiple abundance indices," *The Journal of Wildlife Management*, vol. 77, no. 5, pp. 1038–1047, 2013.
[4] S. Kim, N. Shephard, and S. Chib, "Stochastic volatility: Likelihood inference and comparison with ARCH models," *The Review of Economic Studies*, vol. 65, no. 3, pp. 361–393, 1998.
[5] R. Stratonovich, "Conditional Markov processes," *Theory of Probability and its Applications*, vol. 5, no. 156–178, 1960.
[6] H. Kushner, "On the differential equations satisfied by conditional probability densities of Markov processes, with applications," *SIAM Control Ser. A*, vol. 21, no. 1, pp. 106–119, 1964.
[7] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[8] V. E. Beneš, "Exact finite dimensional filters for certain diffusion with nonlinear drift," *Stochastics*, vol. 5, no. 1-2, pp. 65–92, 1981.
[9] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
[10] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.
[11] P. M. Djurić, "Asymptotic MAP criteria for model selection," *IEEE Trans. on Signal Processing*, vol. 46, no. 10, pp. 2726–2735, Oct 1998.
[12] J. F. de Freitas, M. Niranjan, A. H. Gee, and A. Doucet, "Sequential Monte Carlo methods to train neural network models," *Neural computation*, vol. 12, no. 4, pp. 955–993, 2000.
[13] P. J. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.
[14] J. Liu and M. West, *Sequential Monte Carlo Methods in Practice*. Springer, 2001, ch. Combined parameter and state estimation in simulation-based filtering.
[15] F. Tobar and D. Mandic, "A particle filtering based kernel HMM predictor," in *Proc. of IEEE ICASSP*, 2014, pp. 8019–8023.
[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
[17] M. Deisenroth, R. Turner, M. Huber, U. Hanebeck, and C. Rasmussen, "Robust filtering and smoothing with Gaussian processes," *IEEE Trans. on Automatic Control*, vol. 57, no. 7, pp. 1865–1871, July 2012.
[18] R. Frigola, F. Lindsten, T. B. Schon, and C. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3156–3164.
[19] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.
[20] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
[21] I. Steinwart, D. Hush, and C. Scovel, "An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels," *IEEE Trans. on Information Theory*, vol. 52, no. 10, pp. 4635–4643, 2006.
[22] D. P. Mandic and J. A. Chambers, *Recurrent neural networks for prediction: Learning algorithms, architectures, and stability*. John Wiley & Sons, 2001.
[23] W. Liu, J. C. Principe, and S. Haykin, *Kernel adaptive filtering: A comprehensive introduction*. Wiley, 2010.
[24] A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.
[25] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.
[26] F. Tobar, S.-Y. Kung, and D. Mandic, "Multikernel least mean square algorithm," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 265–277, 2014.
[27] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
[28] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Advances in Neural Information Processing Systems 9*, 1996, pp. 155–161.
[29] Z. Ghahramani and S. T. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in *Advances in Neural Information Processing Systems 11*. MIT Press, 1999, pp. 431–437.
[30] L. Ralaivola and F. D'Alché-Buc, "Time series filtering, smoothing and learning using the kernel Kalman filter," in *Proc. of IEEE IJCNN*, vol. 3. IEEE, 2005, pp. 1449–1454.
[31] W. Liu, I. Park, Y. Wang, and J. Principe, "Extended Kernel Recursive Least Squares Algorithm," *IEEE Trans. on Signal Processing*, vol. 57, no. 10, pp. 3801–3814, Oct. 2009.
[32] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.
[33] K. Fukumizu, L. Song, and A. Gretton, "Kernel Bayes' rule: Bayesian inference with positive definite kernels," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3753–3783, 2013.
[34] M. Kanagawa, Y. Nishiyama, A. Gretton, and K. Fukumizu, "Monte Carlo filtering using kernel embedding of distributions," in *Proc. of AAAI Conference on Artificial Intelligence*, 2014, pp. 1897–1903.
[35] C. Andrieu and G. O. Roberts, "The pseudo-marginal approach for efficient Monte Carlo computations," *The Annals of Statistics*, pp. 697–725, 2009.
[36] N. Aronszajn, "Theory of reproducing kernels," *Trans. of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.
[37] B. Schölkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," in *Computational Learning Theory*, D. Helmbold and B. Williamson, Eds. Springer Berlin Heidelberg, 2001, vol. 2111, pp. 416–426.
[38] P. Honeine, "Analyzing sparse dictionaries for online learning with kernels," *IEEE Trans. on Signal Processing*, 2014, in print.
[39] Y. Engel, S. Mannor, and R. Meir, "Sparse online greedy support vector regression," in *Proc. of ECML*, 2002, pp. 84–96.
[40] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.
[41] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.
[42] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 67–93, 2001.
[43] H. Q. Minh, "Some properties of Gaussian reproducing kernel Hilbert spaces and their implications for function approximation and learning theory," *Constructive Approximation*, vol. 32, no. 2, pp. 307–338, 2010.
[44] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.
[45] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.
[46] J. Zhuang, J. Wang, S. Hoi, and X. Lan., "Unsupervised multiple kernel learning," *Journal of Machine Learning Research*, no. 20, pp. 129–144, 2011.

[47] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer, 2013.

[48] Y. Xia, S. Douglas, and D. Mandic, “Adaptive frequency estimation in smart grid applications: Exploiting noncircularity and widely linear adaptive estimators,” *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 44–54, 2012.

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

[50] D. J. C. MacKay, “Comparison of approximate methods for handling hyperparameters,” *Neural Computation*, vol. 11, no. 5, pp. 1035–1068, 1999.

[51] S. Thrun, L. J., and V. Verma, “Risk sensitive particle filters,” in *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[52] M. Orchard, P. Hevia-Koch, B. Zhang, and L. Tang, “Risk measures for particle-filtering-based state-of-charge prognosis in lithium-ion batteries,” *IEEE Trans. on Industrial Electronics*, vol. 60, no. 11, pp. 5260–5269, 2013.

Felipe Tobar (S’13-M’15) is currently a Postdoctoral Research Associate in the Computational and Biological Learning Group at the University of Cambridge, UK. He holds a PhD in Adaptive Signal Processing (2014) from Imperial College London, UK, and received the BSc (2008) and MSc (2010) degrees in Electrical Engineering from Universidad de Chile.

His research interests lie within the interface between signal processing and machine learning, and include support vector regression, Gaussian processes, Bayesian inference, Monte Carlo methods, and nonlinear adaptive filtering.

Petar M. Djurić (S’86-M’90-SM’99-F’06) received the B.S. and M.S. degrees in electrical engineering from the University of Belgrade, Belgrade, Yugoslavia, in 1981 and 1986, respectively, and the Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston, RI, USA, in 1990. He is currently a Professor with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA, where he joined in 1990. From 1981 to 1986, he was a Research Associate with the Vinča Institute of Nuclear Sciences, Belgrade. His research has been in the area of signal and information processing with primary interests in the theory of signal modeling, detection, and estimation; Monte Carlo-based methods; signal and information processing over networks; and applications in a wide range of disciplines. He has been invited to lecture at many universities in the United States and overseas. Prof. Djurić was a recipient of the IEEE Signal Processing Magazine Best Paper Award in 2007 and the EURASIP Technical Achievement Award in 2012. In 2008, he was the Chair of Excellence of Universidad Carlos III de Madrid-Banco de Santander. From 2008 to 2009, he was a Distinguished Lecturer of the IEEE Signal Processing Society. He has been on numerous committees of the IEEE Signal Processing Society and of many professional conferences and workshops. He is the Editor-in-Chief of the IEEE Transactions on Signal and Information Processing over Networks.

Danilo P. Mandic (M’99-SM’03-F’12) is a Professor in signal processing with Imperial College London, London, U.K., where he has been working in the area of nonlinear adaptive signal processing and nonlinear dynamics. He has been a Guest Professor with Katholieke Universiteit Leuven, Leuven, Belgium and a Frontier Researcher with RIKEN, Tokyo. His publication record includes two research monographs titled Recurrent Neural Networks for Prediction (West Sussex, U.K.: Wiley, August 2001) and Complex-Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models (West Sussex, U.K.: Wiley, April 2009), an edited book titled Signal Processing for Information Fusion (New York: Springer, 2008), and more than 200 publications on signal and image processing.

He has been a member of the IEEE Technical Committees on Signal Processing Theory and Methods and Machine Learning for Signal Processing, an Associate Editor for the IEEE Transactions on Circuits and Systems II, the IEEE Transactions on Signal Processing, the IEEE Transactions on Neural Networks, and IEEE Signal Processing Magazine. He has produced award winning papers and products from his collaboration with the industry, and has received President’s Award for excellence in postgraduate supervision at Imperial College. He is a member of the London Mathematical Society.