Echo State Kernel Recursive Least Squares Algorithm for Machine Condition Prediction

Haowen Zhou[a] , Jinquan Huang[a] , Feng Lu[a] , Jeyarajan Thiyagalingam[b] , Thia Kirubarajan[c]
[a]Jiangsu Province Key Laboratory of Aerospace Power Systems, College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China;
[b]Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, United Kingdom;
[c]Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada.

Abstract

Kernel adaptive filter (KAF) has been widely utilized for time series prediction due to its online adaptation scheme, universal approximation capability and convexity. Nevertheless, KAF's ability to handle temporal tasks is limited, because it is essentially a feed-forward neural network that lacks dynamic characteristics. Traditionally, a sliding widow that contains consecutive data points is constructed to deal with the temporal dependency between data points at neighboring time steps, but the restricted widow length may be incapable of capturing temporal patterns on a larger time scale. To manage this issue, a novel sequential learning approach called echo state KRLS (ES-KRLS) algorithm is proposed by incorporating a dynamic reservoir into kernel recursive least squares (KRLS) algorithm. The reservoir, consisting of a large number of sparsely interconnected hidden units, is treated as a temporal function that transforms the history of the time series into a high-dimensional reservoir state space. Subsequently, the spatial relationship between the reservoir state and the target output is effectively approximated by KRLS algorithm. With the utilization of the fixed reservoir, our novel method not only maintains the simplicity of the learning process but also leads to a significant improvement in the capability of modeling dynamic systems. Numerical results on benchmark tasks demonstrate the excellent performance of the novel method with respect to long-term prediction. Finally, an online prognostic method that combines ES-KRLS and a Bayesian technique is developed for tracking the health status of a degraded system and predicting remaining useful life (RUL). This prognostic method is applied to a turbofan engine degradation dataset to demonstrate its effectiveness.

Keywords: kernel adaptive filter, reservoir computing, long-term prediction, remaining useful life prediction, prognostics

1. Introduction

Kernel method provides a unified framework for pattern analysis and nonlinear signal processing, and as such they appear in numerous successful applications, including the support vector machine [1], kernel principal component analysis [2], and kernel regularization network [3]. The main idea of the kernel method is that a nonlinear mapping associated with a Mercer kernel is utilized to transform the data from the input space to a high-dimensional feature space with rich representations. However, the above-mentioned methods are formulated in a batch form. If the training samples arrive sequentially, these offline algorithms have to retrain the approximation model from scratch once a new training data is available. This may impose restriction on the applications of these algorithms to online scenarios, especially when real-time performance is emphasized. Therefore, a sequential learning method that updates the existing model incrementally would be a better choice for handling the data that arrives in a flow mode.

Online kernel-based learning (OKL) [4, 5] provides an alternative to train the model recursively. As a subfield of OKL, kernel adaptive filters (KAFs) have gained widespread use because of their simple structure, universal approximation capability, and convexity. KAFs are the generalization of the well-established linear adaptive filtering algorithms, approximating a nonlinear function by reformulating the linear structure in a reproducing kernel Hilbert space (RKHS). The KAF family includes the kernel least mean square (KLMS) [6], kernel affine projection algorithm [7], KRLS [8], and extended kernel recursive least squares [9] algorithms, to mention a few.

Although KAF has been widely used for time series prediction, two drawbacks that remain to be solved. The first is the lack of sparseness. At each iteration, KAFs allocate a kernel unit for the new data point. Consequently, the network size grows linearly with the number of training samples, leading to a continuous increase in computational burden and memory requirements. To control the growth of the network, many sparsification techniques have been adopted. The basic idea behind these methods is to select only the informative training samples to train the network, according to some criteria, such as the approximation linear dependency (ALD) criterion [8], surprise criterion [10], novelty criterion [11], or minimum description length criterion [12].

The second drawback is that KAF is incapable of capturing temporal characteristics of nonlinear dynamic systems [13], because KAF is essentially a single-layer feed-forward neural network (FNN). An FNN aims to learn a static mapping where the outputs of the network depend solely on the current inputs, thereby neglecting the dependency of data points at neighboring time steps. As a remedy, often a fixed-size sliding widow storing consecutive values is constructed to transform the temporal correlation into the spatial correlation. Nevertheless, prior knowledge is required in order to select the appropriate time embedding dimension such that the latent dynamic characteristics of the systems unfold [14]. Apart from FNN, recurrent neural network (RNN) is another artificial neural network architecture where hidden units are interconnected. The existence of the recurrence connections between units makes RNN a dynamic system. The dynamic behaviors of RNN rely on the history of the inputs, and thereby achieve the necessary memory capability. RNN has been proven to be able to model any dynamic system with arbitrary precision [15]. The connection weights of RNN are mainly trained by different gradient-based algorithms, such as back propagation through time [16], extended Kalman filtering [17] and real-time recurrent learning [18]. Unfortunately, these algorithms often suffer from slow convergence and create a high computational burden, because the gradients tend to vanish or explode rapidly through propagation.

Aimed at avoiding the difficulty of adapting the connection weights of RNN, echo state network (ESN) and liquid state machine provide a new paradigm for RNN, called reservoir computing (RC) [19-21]. The RC framework contains a dynamic reservoir, where a large number of hidden units are sparsely connected, as well as a linear readout layer. The reservoir is randomly generated and stays fixed during the learning process. The reservoir can be considered as a spatiotemporal mapping that transforms the history of the inputs into a high-dimensional reservoir state space where the readout layer is learned by simple linear regression methods. As a result of its simple and effective learning procedure, ESN has become an appealing tool for time series prediction [22-26].

One fascinating property of ESN is that a temporal task is converted into a non-temporal task of learning a static mapping, because the dynamic reservoir has a fixed recurrent topology and thus only the output weights need to be solved. This facilitates the considerable flexibility of designing the training criterion. The direct method of training the readout layer is calculating the Moore-Penrose pseudo-inverse, which may suffer from ill-posed and over-fitting problems. Hence, regularized regression methods such as ridge regression and the lasso method are utilized to obtain the output weights of ESNs [27]. To alleviate the detrimental effect of noises and outliers on time series prediction, Gaussian process regression is combined with ESN to provide a robust prediction by deriving a posterior distribution of the network output [28, 29]. In [22], the readout layer is treated as a linear support vector machine that is trained based on the epsilon-sensitive or Huber cost functions, such that the robustness to outliers is obtained. In [30], the learning procedure is formulated in a Bayesian framework, whereas the commonly used Gaussian distribution is substituted by a Laplace distribution, which is more insensitive to outliers. In order to make ESN suitable for online applications, the readout layer is traditionally constructed as linear adaptive filters, including the recursive least squares (RLS) and least mean square (LMS) [31] algorithms. The online adjustments to the output weight can also be conducted by various variants of the Kalman filter [32]. Furthermore, the readout layer is extended to a Volterra filter by using the nonlinear structure [33], but the computational complexity expands exponentially with the number of hidden units in the reservoir.

In this paper, a novel ES-KRLS algorithm that incorporates a dynamic reservoir into the KRLS algorithm is proposed. With the aid of the reservoir, the time series up to the current time step is mapped into a high-dimensional reservoir state space, rather than selected data points. Subsequently, another nonlinear mapping associated with a particular Mercer kernel is performed to transform the obtained reservoir state into the potentially infinite-dimensional RKHS, where the readout layer is constructed to rebuild the desired outputs. As a result of this two-step transformation, both the spatial and temporal dependencies between different data points are fully exploited for model approximation. As a matter of fact, the ES-KRLS algorithm is considered as an integration of ESN and KAF, created by reformulating the readout layer in RKHS. To avoid performing direct calculation in RKHS, the readout layer is expressed in the form of inner products, leading to a radial basis function network with more powerful generalization capability. Moreover, the novel method maintains the simplicity of the learning procedure and computational efficiency. In view of its excellent performance of dealing with temporal problems, ES-KRLS algorithm is implemented for prognostics, which aims to predict the fault growth trend and is considered as a long-term prediction task.

The reminder of this paper is organized as follows. Section 2 introduces online ESN. Section 3 provides a review of KRLS. Section 4 elaborates on the derivation of ES-KRLS algorithm. Section 5 presents the experimental results on chaotic time series prediction to demonstrate the feasibility of ES-KRLS algorithm. A novel prognostic method based on ES-KRLS is developed in Section 6. Section 7 provides concluding remarks about this research work.

2. Online echo state network

2.1 Echo state network

ESN consists of two basic modules, a dynamic reservoir and a liner readout layer. The reservoir is essentially a randomly-initialized discrete-time RNN, and the internal reservoir state at each time step is updated by

$$\mathbf{x}_i = h(\mathbf{W}\mathbf{x}_{i-1} + \nu \mathbf{W}_{in}\mathbf{u}_i) \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^{N\times 1}$ is the internal state, N is the hidden units in the reservoir, $\mathbf{u}_i \in \mathbb{R}^{N_\mathbf{u}\times 1}$ is the input vector fed to the network, $h(\cdot)$ is the sigmoid activation function that can be applied element-wise, $\mathbf{W} \in \mathbb{R}^{N\times N}$ are the weights of the connections between hidden units in the reservoir, $\mathbf{W}_{in} \in \mathbb{R}^{N\times N_\mathbf{u}}$ are the weights of the connections between the hidden units and the input vector, ν denotes the input scaling parameter that is introduced to adjust the operating region of the reservoir for different tasks. A small value for ν indicates that the reservoir operates around the linear region of the sigmoid function and thereby makes the reservoir suitable for a linear regression task. Otherwise, if ν is large, the reservoir exhibits strong nonlinearity and is highly qualified to solve a nonlinear problem.

The readout layer is constructed as a linear combination of the internal state and the input vector to mimic the target output with the following form

$$d_i = \mathbf{W}_{out}^{state}\mathbf{x}_i + \mathbf{W}_{out}^{input}\mathbf{u}_i = \mathbf{W}_{out}\boldsymbol{\psi}_i \tag{2}$$

where d_i is the network output, $\mathbf{W}_{out}^{state} \in \mathbb{R}^{1\times N}$, $\mathbf{W}_{out}^{input} \in \mathbb{R}^{1\times N_\mathbf{u}}$, $\mathbf{W}_{out} = [\mathbf{W}_{out}^{state}\ \ \mathbf{W}_{out}^{input}]$ are the output weights, and $\boldsymbol{\psi}_i = [\mathbf{x}_i; \mathbf{u}_i]$ is the augmented input that is the concatenation of the input vector and the internal state.

When ESN is employed to model a nonlinear dynamic system, the reservoir and the readout layer serve as distinct roles. Specifically, the reservoir with a sufficient number of hidden units is passively driven by the input sequence. Although the initial reservoir state $\mathbf{x}_i$ is randomly generated, the input signal will gradually dominate the evolution of the reservoir state. This property is called echo state property that holds if the spectral radius of $\mathbf{W}$ is less than 1 [34]. Therefore, the reservoir is considered as a temporal function that transform the history of the time series into a high-dimensional reservoir state space. By contrast, the readout layer is fundamentally a non-temporal function that is trained by a simple linear regression method.

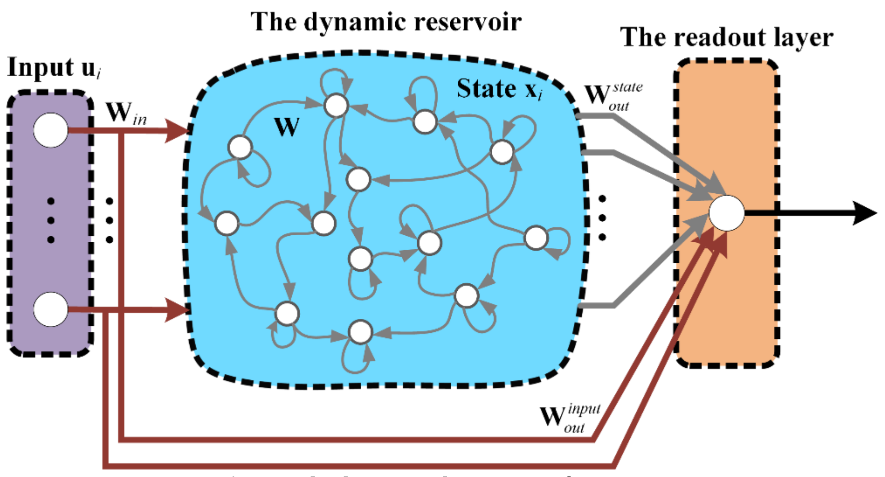


Fig. 1. The basic architecture of ESN

2.2 Online training procedure

When the training samples arrive sequentially, the existing approximation model needs to be updated recursively. Given that the reservoir topology is fixed during the learning process, only the output weights $\mathbf{W}_{out}$ require online adaption, which indicates that the readout layer acts as a linear adaptive filter. The LMS algorithm incrementally adjusts the output weights $\mathbf{W}_{out}$ by minimizing the cost function J_i as follows

$$J_i = \frac{1}{2} e_i^2 \tag{3}$$

where e_i is the estimation error. Accordingly, the optimal $\mathbf{W}_{out,i}$ at the *i-th* iteration is obtained by computing

$$\begin{aligned} e_i &= d_i - \mathbf{W}_{out,i-1} \boldsymbol{\psi}_i \\ \mathbf{W}_{out,i} &= \mathbf{W}_{out,i-1} + \eta e_i \boldsymbol{\psi}_i \end{aligned} \tag{4}$$

where η is the learning rate. However, the convergence performance of LMS algorithm deteriorates quickly due to the eigenvalue spread of the correlation matrix of the reservoir state [20]. In order to alleviate the downside effects of this eigenvalue spread, RLS algorithm provides another choice to train the output weights. Basically, RLS algorithm aims to minimize the sum of squared estimation errors up to the current time, and the corresponding cost function is expressed as

$$J_i = \sum_{j=1}^{i} [d_j - \mathbf{W}_{out} \boldsymbol{\psi}_j]^2 \tag{5}$$

Hence, the RLS algorithm operates according to the following procedures:

$$\begin{aligned} r_i &= 1 + \boldsymbol{\psi}_i^T \mathbf{P}_{i-1} \boldsymbol{\psi}_i \\ \mathbf{k}_i &= \mathbf{P}_{i-1} \boldsymbol{\psi}_i / r_i \\ e_i &= d_i - \mathbf{W}_{out,i-1} \boldsymbol{\psi}_i \\ \mathbf{W}_{out,i} &= \mathbf{W}_{out,i-1} + \mathbf{k}_i e_i \\ \mathbf{P}_i &= \mathbf{P}_{i-1} - \mathbf{k}_i \mathbf{k}_i^T r_i \end{aligned} \tag{6}$$

Compared with LMS algorithm, RLS algorithm achieves a better convergence performance at cost of more expensive computation. Despite its linear structure, the readout layer is able to mimic the target system with the desired accuracy in many cases. This can be ascribed to the fact that a nonlinear mapping is performed by the reservoir to convert the data into a high-dimensional space. Nevertheless, there is a tradeoff between the nonlinearity of the mapping and the memory capability with respect to the reservoir [35]. Specifically, the more nonlinearity the reservoir exhibits, the less memory capability the reservoir achieves. To overcome this drawback, we reconstruct the linear output layer in RKHS. In this approach, the reservoir is tuned to perform a temporal mapping, while a kernel-induced mapping focus on the nonlinear and static transformation of the reservoir state.

3. A review of Kernel recursive least squares algorithm

Consider the task of learning a mapping $f:\mathbb{U}\to\mathbb{R}$ based on a sequence of input-output pairs $\{\mathbf{u}_j, d_j\}_{j=1}^{i}$, where $\mathbb{U}\subseteq\mathbb{R}^d$ is the input space, $\mathbf{u}_j\in\mathbb{U}$ is the input vector at the *j-th* time step, and $d_j\in\mathbb{R}$ is the corresponding target output.

To begin with, a nonlinear mapping $\varphi(\cdot)$ associated with a Mercer kernel is utilized to covert the input vector $\mathbf{u}$ into a potentially infinite-dimensional feature space $\mathbb{H}$, i.e., RKHS. A Mercer kernel is a continuous, symmetric, and positive definite function $\kappa:\mathbb{U}\times\mathbb{U}\to\mathbb{R}$. Mercer's theorem ensures that any kernel can be expressed as

$$\kappa(\mathbf{u}_1,\mathbf{u}_2)=\sum_{n=1}^{\infty}\varsigma_n\phi_n(\mathbf{u}_1)\phi_n(\mathbf{u}_2) \tag{7}$$

where ς_n and ϕ_n are the eigenvalues and the eigenfunctions, respectively. Accordingly, the kernel-induced mapping $\varphi:\mathbb{U}\to\mathbb{H}$ is established as

$$\varphi(\mathbf{u})=[\sqrt{\varsigma_1}\phi_1(\mathbf{u}),\sqrt{\varsigma_2}\phi_2(\mathbf{u}),\cdots]^T \tag{8}$$

With the utilization of the kernel-induced mapping $\varphi(\cdot)$, a linear model in the feature space can be constructed as

$$f(\cdot)=\boldsymbol{\omega}_i^T\varphi(\cdot) \tag{9}$$

where $\boldsymbol{\omega}_i$ denotes the weight vector in RKHS. Then, the cost function is formulated as

$$\min_{\boldsymbol{\omega}_i} J=\sum_{j=1}^{i}\left|d_j-\boldsymbol{\omega}_i^T\varphi(\mathbf{u}_j)\right|^2+\lambda\left\|\boldsymbol{\omega}_i\right\|^2 \tag{10}$$

where λ is the regularization parameter that handles the trade-off between the training error and the model complexity. By setting the gradient of J with respect to $\boldsymbol{\omega}_i$ to zero, the optimal solution of Eq.(10) is obtained as

$$\boldsymbol{\omega}_i=\boldsymbol{\Phi}_i\left[\lambda\mathbf{I}_i+\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i\right]^{-1}\mathbf{d}_i \tag{11}$$

where $\mathbf{I}_i$ is the identity matrix, $\mathbf{d}_i=[d_1,\cdots,d_i]^T$, and $\boldsymbol{\Phi}_i=[\varphi(\mathbf{u}_1),\cdots,\varphi(\mathbf{u}_i)]$. $\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i$ in Eq.(11) can be calculated by the kernel trick, yielding

$$\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i=\begin{bmatrix}\kappa(\mathbf{u}_1,\mathbf{u}_1) & \cdots & \kappa(\mathbf{u}_1,\mathbf{u}_i)\\ \kappa(\mathbf{u}_2,\mathbf{u}_1) & \cdots & \kappa(\mathbf{u}_2,\mathbf{u}_i)\\ \vdots & \ddots & \vdots\\ \kappa(\mathbf{u}_i,\mathbf{u}_1) & \cdots & \kappa(\mathbf{u}_i,\mathbf{u}_i)\end{bmatrix} \tag{12}$$

Furthermore, the weight vector $\boldsymbol{\omega}_i$ can be also expressed in the form of a linear combination of the transformed augmented inputs as

$$\boldsymbol{\omega}_i=\boldsymbol{\Phi}_i\boldsymbol{\alpha}_i \tag{13}$$

where $\boldsymbol{\alpha}_i=[\lambda\mathbf{I}+\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i]^{-1}\mathbf{d}_i$. Let $\mathbf{Q}_i=[\lambda\mathbf{I}+\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i]^{-1}$ and we can rewrite $\mathbf{Q}_i$ in a recursive form as

$$\mathbf{Q}_i^{-1}=\begin{bmatrix}\mathbf{Q}_{i-1}^{-1} & \mathbf{h}_i\\ \mathbf{h}_i^T & \lambda+\kappa(\mathbf{u}_i,\mathbf{u}_i)\end{bmatrix} \tag{14}$$

where $\mathbf{h}_i=\boldsymbol{\Phi}_{i-1}^T\varphi(\mathbf{u}_i)=[\kappa(\mathbf{u}_1,\mathbf{u}_i),\cdots,\kappa(\mathbf{u}_{i-1},\mathbf{u}_i)]^T$. Given that $\mathbf{Q}_{i-1}$ has already been calculated in the previous iteration, the matrix inversion lemma is utilized to update $\mathbf{Q}_i$ by

$$\mathbf{Q}_i=\begin{bmatrix}\mathbf{Q}_{i-1} & \mathbf{0}\\ \mathbf{0}^T & 0\end{bmatrix}+r_i^{-1}\begin{bmatrix}\mathbf{z}_i\\ -1\end{bmatrix}\begin{bmatrix}\mathbf{z}_i^T & -1\end{bmatrix} \tag{15}$$

where

$$\mathbf{z}_i=\mathbf{Q}_{i-1}\mathbf{h}_i \tag{16}$$

$$r_i=\lambda+\kappa(\mathbf{u}_i,\mathbf{u}_i)-\mathbf{z}_i^T\mathbf{h}_i \tag{17}$$

Therefore, the expansion coefficient of the weight vector is gained by

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i \mathbf{d}_i \tag{18}$$

Finally, the approximated function $f_i(\cdot)$ at the *i-th* iteration can be expressed as

$$f_i(\mathbf{u}) = \boldsymbol{\omega}_i^T \varphi(\mathbf{u}) = \boldsymbol{\alpha}_i^T \boldsymbol{\Phi}_i^T \varphi(\mathbf{u}) = \sum_{j=1}^{i} \alpha_i^j \kappa(\mathbf{u}_j, \mathbf{u}) \tag{19}$$

where $\boldsymbol{\alpha}_i = [\alpha_i^1, \alpha_i^2, \cdots, \alpha_i^i]^T$. As is depicted in Fig. 2, KAF yields a single-layer feed-forward neural network. Despite its universal approximation capability, KAF are essentially a static function without any dynamic characteristics.

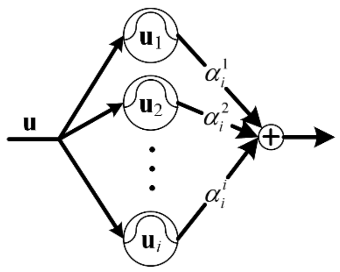


Fig. 2. The basic architecture of KAF.

4. Echo state kernel recursive least squares algorithm

4.1 Model formulation

The computational separation between the reservoir and the readout layer makes it possible to incorporate a reservoir into the KRLS algorithm.

The readout layer is reformulated as a linear model in RKHS with the following form

$$d_i = g_i(\mathbf{x}_i, \mathbf{u}_i) = g_i(\boldsymbol{\psi}_i) = \boldsymbol{\omega}_i^T \varphi(\boldsymbol{\psi}_i) \tag{20}$$

At each iteration, the readout layer is updated recursively by KRLS algorithm. From Eq.(19), we can notice that the readout layer is essentially a growing radial basis function network. As a result, the computational and memory requirements will increase linearly with the number of training samples. In this work, a novel online sparsification technique is employed to make the online temporal learning procedure more efficient. Specifically, a subset of training samples are selected to update the structure of the network, i.e., add a new kernel unit into the existing network, and the rest of samples are utilized to modify the coefficients of the network. By doing so, the growth of the network size can be curbed effectively without sacrificing the approximation performance.

For sake of clarity, we define $\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^m$, where $\mathbf{c}_k$ is the center of the *k-th* kernel unit and m is the number of kernel units. $\mathcal{C}$ is termed the dictionary and we gradually adds the augmented inputs into the dictionary according to a certain rule.

Based on the idea mentioned above, the cost function can be expressed as

$$\min_{\boldsymbol{\omega}_i} J = \sum_{j=1}^{i} \left| d_j - \boldsymbol{\omega}_i^T \varphi(\boldsymbol{\psi}_j) \right|^2 + \lambda \left\| \boldsymbol{\omega}_i \right\|^2 \tag{21}$$

where $\boldsymbol{\omega}_i = \boldsymbol{\Phi}_i \boldsymbol{\alpha}_i$ and $\boldsymbol{\Phi}_i = [\varphi(\mathbf{c}_1), \varphi(\mathbf{c}_2), \cdots, \varphi(\mathbf{c}_m)]$. Eq.(21) can be further written as

$$\min_{\boldsymbol{\alpha}_i} J = \sum_{j=1}^{i} \left| d_j - \sum_{k=1}^{m} \alpha_i^k \kappa(\mathbf{c}_k, \boldsymbol{\psi}_j) \right|^2 + \lambda \boldsymbol{\alpha}_i^T \mathbf{k}_{B,i} \boldsymbol{\alpha}_i \tag{22}$$

where

$$\mathbf{k}_{B,i} = \begin{bmatrix} \kappa(\mathbf{c}_1, \mathbf{c}_1) & \cdots & \kappa(\mathbf{c}_1, \mathbf{c}_m) \\ \kappa(\mathbf{c}_2, \mathbf{c}_1) & \cdots & \kappa(\mathbf{c}_2, \mathbf{c}_m) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{c}_m, \mathbf{c}_1) & \cdots & \kappa(\mathbf{c}_m, \mathbf{c}_m) \end{bmatrix}$$

Let $\dfrac{\partial J}{\partial \boldsymbol{\alpha}_i} = 0$, and the optimal solution of Eq.(22) is obtained as

$$\boldsymbol{\alpha}_i = \left(\lambda \mathbf{k}_{B,i} + \mathbf{k}_{P,i}\mathbf{k}_{P,i}^T\right)^{-1} \mathbf{k}_{P,i}\mathbf{d}_i \tag{23}$$

where

$$\mathbf{k}_{P,i} = \begin{bmatrix} \kappa(\mathbf{c}_1, \boldsymbol{\psi}_1) & \cdots & \kappa(\mathbf{c}_1, \boldsymbol{\psi}_i) \\ \kappa(\mathbf{c}_2, \boldsymbol{\psi}_1) & \cdots & \kappa(\mathbf{c}_2, \boldsymbol{\psi}_i) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{c}_m, \boldsymbol{\psi}_1) & \cdots & \kappa(\mathbf{c}_m, \boldsymbol{\psi}_i) \end{bmatrix}$$

Let $\mathbf{Q}_i = \left(\lambda \mathbf{k}_{B,i} + \mathbf{k}_{P,i}\mathbf{k}_{P,i}^T\right)^{-1}$, and we rewrite $\boldsymbol{\alpha}_i$ as

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i \mathbf{k}_{P,i}\mathbf{d}_i \tag{24}$$

when a new training data $\{\boldsymbol{\psi}_i, d_i\}$ arrives at the *i-th* iteration, it is used to update the coefficients of the network at the first step. The inverse matrix $\bar{\mathbf{Q}}_i$ is updated by

$$\bar{\mathbf{Q}}_i = \left[\lambda \mathbf{k}_{B,i-1} + \begin{bmatrix} \mathbf{k}_{P,i-1} & \mathbf{k}_{iB} \end{bmatrix} \begin{bmatrix} \mathbf{k}_{P,i-1}^T \\ \mathbf{k}_{iB}^T \end{bmatrix}\right]^{-1} = (\mathbf{Q}_{i-1}^{-1} + \mathbf{k}_{iB}\mathbf{k}_{iB}^T)^{-1} \tag{25}$$

where $\mathbf{k}_{iB} = \left[\kappa(\mathbf{c}_1, \boldsymbol{\psi}_i), \kappa(\mathbf{c}_2, \boldsymbol{\psi}_i), \cdots, \kappa(\mathbf{c}_m, \boldsymbol{\psi}_i)\right]^T$. Eq.(25) is further expressed as

$$\bar{\mathbf{Q}}_i = \mathbf{Q}_{i-1} - \frac{\mathbf{Q}_{i-1}\mathbf{k}_{iB}\mathbf{k}_{iB}^T\mathbf{Q}_{i-1}}{1 + \mathbf{k}_{iB}^T\mathbf{Q}_{i-1}\mathbf{k}_{iB}} \tag{26}$$

Hence, the coefficient vector is recalculated as

$$\bar{\boldsymbol{\alpha}}_i = \bar{\mathbf{Q}}_i \bar{\mathbf{k}}_{P,i}\mathbf{d}_i \tag{27}$$

where $\bar{\mathbf{k}}_{P,i} = \left[\mathbf{k}_{P,i-1}, \mathbf{k}_{iB}\right]$, and $\mathbf{d}_i = \begin{bmatrix} \mathbf{d}_{i-1} \\ d_i \end{bmatrix}$.

Following by the modification of the coefficients of the network, the ALD criterion is adopted to determine whether a new kernel unit is allocated to this training sample, which is expressed as

$$\Omega_i = \kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_i) - \mathbf{k}_{iB}^T\mathbf{k}_{B,i}^{-1}\mathbf{k}_{iB} \tag{28}$$

If Ω_i is larger than a preset constant threshold ε, the augmented input $\boldsymbol{\psi}_i$ needs to be added into the dictionary $\mathcal{C}$. The inverse matrix $\mathbf{Q}_i$ is augmented as

$$\mathbf{Q}_i = \left[\begin{bmatrix} \lambda \mathbf{k}_{B,i-1} & \lambda \mathbf{k}_{iB} \\ \lambda \mathbf{k}_{iB}^T & \lambda \kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_i) \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{k}}_{P,i}\bar{\mathbf{k}}_{P,i}^T & \bar{\mathbf{k}}_{P,i}\mathbf{k}_{iP} \\ \mathbf{k}_{iP}^T\bar{\mathbf{k}}_{P,i}^T & \mathbf{k}_{iP}^T\mathbf{k}_{iP} \end{bmatrix}\right]^{-1} \tag{29}$$

where $\mathbf{k}_{iP} = \left[\kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_1), \kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_2), \cdots, \kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_i)\right]^T$.

Since $\bar{\mathbf{Q}}_i = \left(\lambda \mathbf{k}_{B,i-1} + \bar{\mathbf{k}}_{P,i}\bar{\mathbf{k}}_{P,i}^T\right)^{-1}$ is available, $\mathbf{Q}_i$ is updated by

$$\mathbf{Q}_i = \begin{bmatrix} \bar{\mathbf{Q}}_i & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + r_i^{-1} \begin{bmatrix} \mathbf{z}_i \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{z}_i^T & -1 \end{bmatrix} \tag{30}$$

where

$$\begin{aligned} \mathbf{z}_i &= \bar{\mathbf{Q}}_i(\lambda \mathbf{k}_{iB}^T + \bar{\mathbf{k}}_{P,i}\mathbf{k}_{iP}) \\ r_i &= \lambda \kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_i) + \mathbf{k}_{iP}^T\mathbf{k}_{iP} - \mathbf{z}_i(\lambda \mathbf{k}_{iB}^T + \bar{\mathbf{k}}_{P,i}\mathbf{k}_{iP}) \end{aligned} \tag{31}$$

Hence, the coefficient vector $\boldsymbol{\alpha}_i$ is expressed as

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i\mathbf{k}_{P,i}\mathbf{d}_i = \begin{bmatrix} \bar{\boldsymbol{\alpha}}_i \\ 0 \end{bmatrix} + r_i^{-1} \begin{bmatrix} \mathbf{z}_i \\ -1 \end{bmatrix} \left[\mathbf{z}_i^T\bar{\mathbf{k}}_{P,i}\mathbf{d}_i - \mathbf{k}_{iP}\mathbf{d}_i\right] \tag{32}$$

where $\mathbf{k}_{P,i} = \begin{bmatrix} \bar{\mathbf{k}}_{P,i} \\ \mathbf{k}_{iP} \end{bmatrix}$. Accordingly, the dictionary $\mathcal{C}$ is updated by $\mathcal{C} = \mathcal{C} \cup \{\boldsymbol{\psi}_i\}$

If Ω_i is less than ε, the dictionary $\mathcal{C}$ maintains unchanged and we let

$$\begin{aligned} \mathbf{Q}_i &= \bar{\mathbf{Q}}_i \\ \boldsymbol{\alpha}_i &= \bar{\boldsymbol{\alpha}}_i \end{aligned} \tag{33}$$

To sum up, the proposed ES-KRLS algorithm is summarized in Algorithm 1.

Since the reservoir stays fixed during the training phase, the temporal task of modelling dynamic systems boils down to a non-temporal task of solving a least-square convex problem. As a result of the convexity, the proposed method can avoid suffering from local minima and ensures the global optimal solution. Conversely, learning with a traditional RNN can be non-convex and the convergence behavior cannot be guaranteed. When the gradient-based algorithms are applied to train an RNN, the quality of the gradient information deteriorates quickly as time evolves. Hence, the effective propagation depth is limited and the long-term memory capability appears to be unachievable. Moreover, traditional RNNs belong to batch algorithms and are computationally expensive. This means that a traditional RNN is inherently not suitable for sequential learning. In contrast, with the ES-KRLS algorithm, the readout layer can be modified recursively once a new sample comes in.

4.2 Relations to traditional KAFs

First, when the spectral radius of $\mathbf{W}$ is set to zero, the connections between each unit that make the reservoir a dynamic system no longer exist. Therefore, the model will degenerate into a feed-forward network. Second, both the reservoir and a Mercer kernel are able to transform the data from the input space into a high-dimensional space with more abundant representations. In ES-KRLS, the fusion of a dynamic reservoir and a Mercer kernel can be interpreted as a multistage kernel, whereby both the spatial and temporal information about the original time series is effectively captured.

However, the major difference between traditional KAFs and ES-KRLS lies in the way they each achieve the memory capability for temporal tasks. When applied to modelling a dynamic system, a traditional KAF is designed as a time-delay neural network (TDNN). Specifically, a sliding window that consists of a finite number of historical inputs is constructed to convert the temporal correlation into a static input pattern. In the proposed ES-KRLS, the reservoir is essentially a temporal function that maps the time series (up to current time step) into the internal state, without the utilization of the time-embedded inputs.

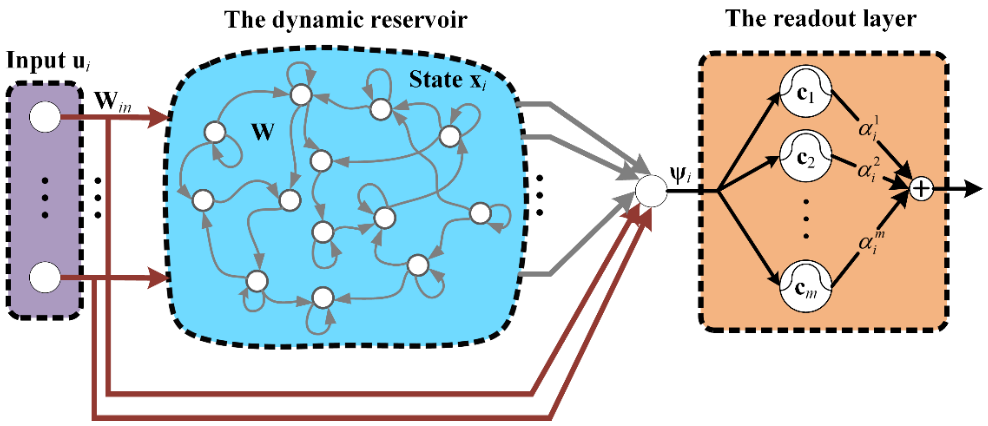


Fig. 3. The basic architecture of ES-KRLS algorithm.

Algorithm 1: Echo State Kernel Recursive Least Squares algorithm

```
Input:
  Generate a random reservoir $(\mathbf{W}_{in}, \mathbf{W}, \nu)$;
Computation:
  For time $i = 1, 2, \cdots$ do
```

```
    A new data point $\{\mathbf{u}_i, d_i\}$ arrives;
    Update the reservoir state based on $\mathbf{x}_{i+1} = h(\mathbf{W}\mathbf{x}_i + \nu\mathbf{W}_{in}\mathbf{u}_{i+1})$;
    If $i = 1$
       Let $\mathcal{C} = \mathcal{C} \cup \{\boldsymbol{\psi}_1\}$;
       Compute $\mathbf{Q}_1 = (\lambda + \kappa(\boldsymbol{\psi}_1, \boldsymbol{\psi}_1))^{-1}$;
       Compute $\boldsymbol{\alpha}_1 = \mathbf{Q}_1 d_1$;
    else
       Update $\bar{\mathbf{Q}}_i$ and $\bar{\boldsymbol{\alpha}}_i$ based on Eq.(26) and Eq.(27), respectively;
       Compute $\Omega_i$ based on Eq.(28);
       If $\Omega_i > \varepsilon$
          Update $\mathbf{Q}_i$ and $\boldsymbol{\alpha}_i$ based on Eq.(30) and Eq.(32), respectively;
          Let $\mathcal{C} = \mathcal{C} \cup \{\boldsymbol{\psi}_i\}$;
       else
          Let $\mathbf{Q}_i = \bar{\mathbf{Q}}_i$ and $\boldsymbol{\alpha}_i = \bar{\boldsymbol{\alpha}}_i$;
       End if
    End if
 End for
```

5. Performance evaluation

In order to verify the feasibility of the proposed ES-KRLS algorithm, we will carry out a series of experiments on two classical benchmark tasks. To illustrate the advantages of our methods, we also evaluate RLS-ESN [31], support vector echo state machine (SVESM) [22], KRLS, KLMS, extreme learning machine (ELM) [36]. For sake of comparison, all experiments were carried out using Matlab 2013b on a personal desktop with Intel® Core™ i5-3230M (2.6GHz) processor, 8.00GB RAM in Windows 7 operation system environment. The entries of the internal weight matrix $\mathbf{W}$ and the input weight matrix $\mathbf{W}_{in}$ are sampled from a uniform distribution over $[-0.5, 0.5]$. The reservoir parameters, including the spectral radius, the number of the hidden units, and the input scaling parameter, are tuned such that the performance of RLS-ESN is maximized. In order to demonstrate the stability of the generalization performance of the ESN-based methods, the experimental results are the averages of 50 different random reservoir initializations. For the ES-KRLS, KRLS, and KLMS algorithms, the Gaussian kernel expressed by $\kappa(\mathbf{u}_1, \mathbf{u}_2) = \exp\{-\|\mathbf{u}_1 - \mathbf{u}_2\|^2 / 2\sigma^2\}$, where σ is the kernel width is adopted. For ES-KRLS and KRLS, the kernel width and the regularization factor are chosen from $\{2^{-3}, \cdots, 2^5\} \times \{10^{-9}, \cdots, 10^{12}\}$ using the tenfold cross validation strategy.

5.1 Mackey-Glass time series

The Mackey-Glass time series has been widely utilized in the literature as a standard benchmark task for nonlinear dynamic system modeling, which is generated by a nonlinear time delay differential equation with the form

$$\frac{dx}{dt} = bx(t) + \frac{ax(t-\delta)}{1 + x(t-\delta)^c} \tag{34}$$

where $x(t)$ denotes the data point at time step *t*. The system exhibits the chaotic characteristic when the delay time $\delta > 16.8$. The parameter values in Eq.(34) are typically selected as $a = 0.2$, $b = -0.1$, $c = 10$, $\delta = 17$ and $x(0) = 1.2$. A segment of 1000 samples are employed to train the considered models and another segment of 500 samples are employed to evaluate the generalization ability of the trained models. With regard to the ESN-based algorithms including RLS-ESN, SVESM and ES-KRLS, a network warm-up time of 100 steps is utilized to wash out the initial transient caused by the randomly initialized reservoir state.

The one-step prediction task is described as using the historical data points $\mathbf{u}_i = \left[x(i), \cdots, x(i-l-1)\right]$, where l denotes the embedding dimension, to predict the next point $x(i+1)$. According to Taken's theorem [14], the input vector $\mathbf{u}_i$ is capable of revealing the dynamic characteristics of a chaotic system without ambiguity, if the embedding dimension l is sufficiently large. As for ELM, KRLS and KLMS, l is set to 10 so as to obtain a sufficient memory capacity. Due to the utilization of the reservoir, the ESN-based methods have the capability to remember the history of the time series and hence we just set l to 1.

In order to make a comprehensive comparison, we are also interested in the multi-step prediction that relies on the iteration of the one-step prediction. The estimated outputs are fed back to the trained model as the inputs to replace the missing values. The iteration process will persist until the predicted value of $x(i+h)$ is obtained, where h represents the prediction horizon. The prediction horizon for Mackey-Glass time series is commonly chosen as 84 and 120 [22, 31]. In practical scenarios, the time series is possibly corrupted by noises and outliers. Hence, we will also test the performance of different models in the presence of noises and outliers.

The root mean square error (RMSE) are employed to measure the accuracy performance of the *h*-step prediction using different models, which is given by

$$\mathrm{RMSE}_h = \sqrt{\frac{1}{L}\sum\nolimits_{j=1}^{L}\left[\hat{x}(j+h) - x(j+h)\right]^2} \tag{35}$$

where L is the length of the testing sequence and $\hat{x}(j+h)$ represents the *h*-step prediction value of $x(j+h)$.

In the first simulation, the time series used for training and testing are noiseless. The obtained results are listed in Table 1. The accuracy performance of the trained models deteriorate considerably with the increase of prediction horizon, because the prediction error is accumulated during the iteration process. The considered models except KLMS share a similar performance in the term of one-step prediction. However, ESN-based approaches exhibit the superiority over the other methods in long-term prediction. It is noteworthy that the hidden units of ELM are randomly generated and only the output weights are trained to optimally fit the target output, which is similar to ESN. Nevertheless, the absence of the connections between different units turns ELM into a static reservoir without dynamic properties. This is the reason why ELM is unable to obtain the same prediction accuracy as ESN-based algorithms. The proposed ES-KRLS achieves a much lower RMSE_{84} and RMSE_{120} values than RLS-ESN and SVESM, because the kernelized version of the output layer improves the nonlinear dynamic system modelling ability significantly. In order to demonstrate the convergence behaviors of online methods including RLS-ESN, ES-KRLS, KRLS and KLMS, we calculate RMSE in the term of one-step prediction on the testing set using the models derived from the training set at each iteration and the corresponding learning curves are depicted in Fig. 4(a). ES-KRLS converges at a faster rate and obtains a lower RMSE_1 value than KLMS, KRLS, and ES-KRLS, which implies that the introduction of the reservoir facilitates the learning process. From Fig. 4(b), we can notice that compared with ES-KRLS, the computation time consumed by KRLS at each time step increases at much faster speed. This is due to the fact that KRLS generates a continuously growing network, while ES-KRLS curbs the growth of the network size effectively (shown in Fig. 4(c)). Instead of allocating a new kernel unit to every training sample, the sparisification technique utilized by ES-KRLS selects a subset of training data to update the network structure and takes advantage of the remaining useful samples to adjust the coefficients of the existing network. In such a case, a tradeoff between the accuracy performance and the computational complexity is achieved.

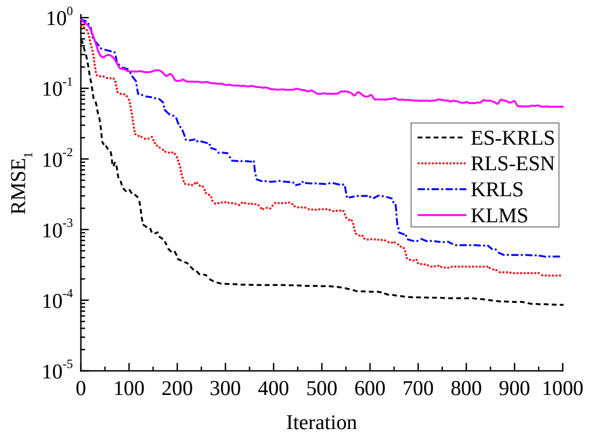
Table 1 Specifications of the employed reservoirs for time series prediction

Parameter	Mackey-Glass	Laser
Spectral radius	0.99	0.99
Number of hidden units	300	600

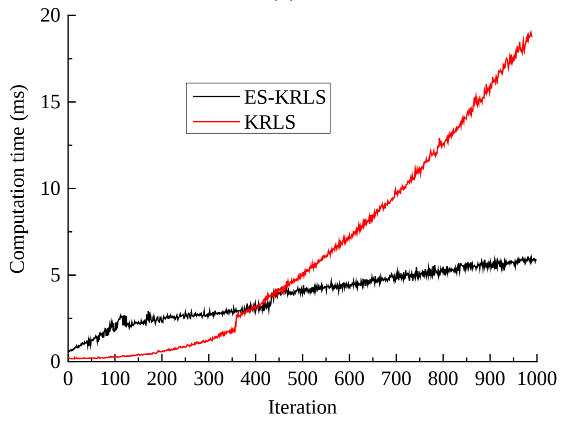
Input scaling parameter	0.8	0.60

Table 2 Experimental results on noiseless Mackey-Glass time series

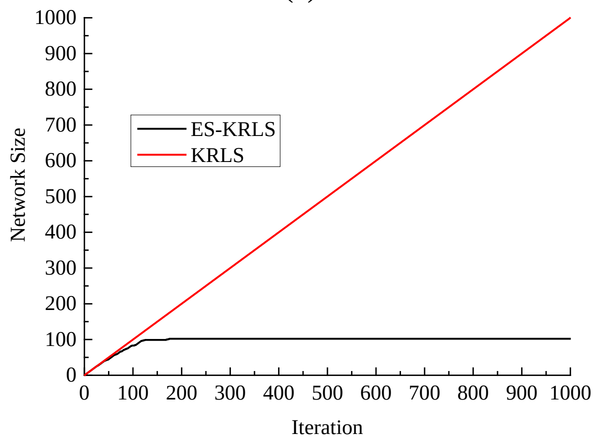
Algorithm	$RMSE_1$	$RMSE_{84}$	$RMSE_{120}$	Training time (s)
RLS-ESN	1.845E−4 ± 2.561E−5	2.733E−2 ± 1.536E−2	5.722E−2 ± 4.282E−2	1.731
SVESM	2.362E−4 ± 3.574E−5	2.238E−2 ± 1.859E−2	3.755E−2 ± 2.588E−2	0.186
ES-KRLS	8.610E−5 ± 2.520E−5	1.862E−3 ± 1.394E−3	4.207E−3 ± 2.051E−3	3.627
KRLS	4.153E−4	9.550E−2	1.624E−1	5.725
KLMS	5.499E−2	2.854E−1	3.288E−1	0.070
ELM	2.309E−4	9.684E−2	1.784E−1	0.116



(a)



(b)



(c)

Fig. 4. Performance comparison between online methods on the one-step prediction task.

(a) Learning curves; (b) Computation time curves; (c) Network growth curves.

In the second simulation, we investigate how noises and outliers affects the performance of the considered models. Both the training data and the testing data are corrupted by heavy-tailed impulsive noises, namely the α-steady noises. The tail parameter and the dispersion parameter of the α-steady distribution are set to 1.4 and 0.1, respectively [37]. Fig. 5 depicts the noisy training sequence. Given that the value of RMSE is sensitive to the large prediction error caused by outliers, RMSE is calculated according to the noiseless time series. As is demonstrated in Table 3, the α-steady noises bring about the minimal effect on the performance of ES-KRLS, which may imply the robustness of ES-KRLS to outliers and noises. Despite the influence of outliers, the time series generated by ES-KRLS in Fig. 6 fits the original testing sequence with the satisfactory accuracy.

Table 3 Experimental results on Mackey-Glass time series polluted by alpha-stable noises

Algorithm	RMSE_1	RMSE_{84}	RMSE_{120}	Training time (s)
RLS-ESN	8.816E−2 ± 1.013E−2	2.086E−1 ± 3.299E−2	2.691E−1 ± 6.002E−2	0.971
SVESM	9.130E−2 ± 9.250E−3	2.392E−1 ± 4.516E−2	3.462E−1 ± 9.786E−2	0.184
ES-KRLS	1.247E−2 ± 6.147E−4	3.632E−2 ± 3.681E−3	5.279E−2 ± 3.549E−3	3.378
KRLS	6.148E−2	2.593E−1	3.559E−1	6.328
KLMS	6.213E−2	3.012E−1	3.126E−1	0.100
ELM	6.081E−2	2.628E−1	2.891E−1	0.116

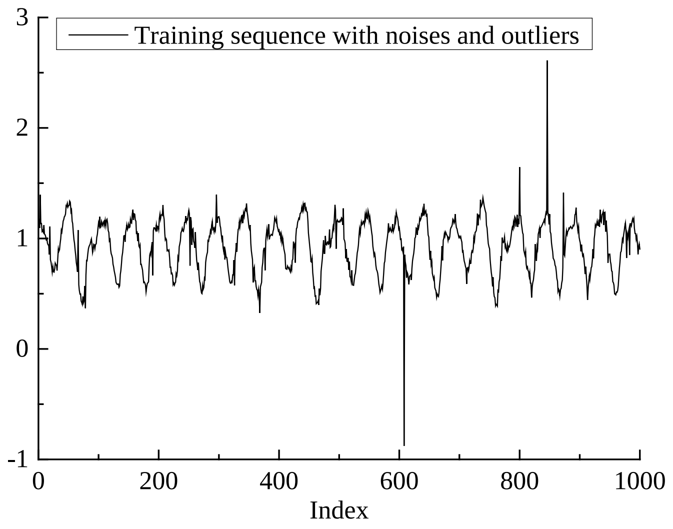
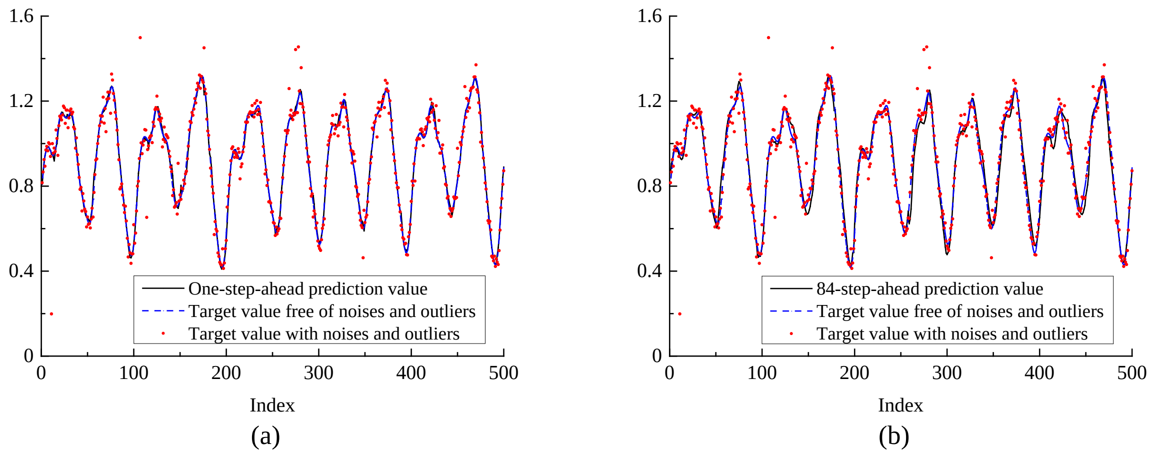
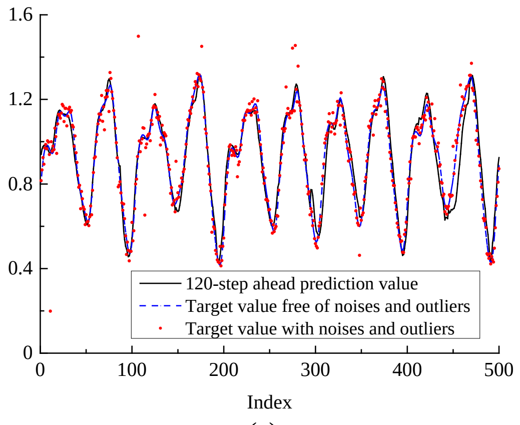


Fig. 5. The training sequence contaminated by the α-steady noises.





(c)

Fig. 6. The prediction performance of ES-KRLS on the Mackey-Glass time series with α-steady noises. (a) One-step-ahead prediction results; (b) 84-step-ahead prediction results; (c) 120-step-ahead prediction results.

5.2 Laser time series

In this example, the data is generated by a far-infrared laser in a chaotic state, which is available from the Santa Fe time-series competition [38]. The laser time series is particularly difficult to predict because of its chaotic dynamics and the fact that only three intensity collapses occur in the data set [8]. The training sequence contains 1000 data points (see Fig. 7), and the subsequent 100 data points are employed to verify the feasibility of the trained models. The data is first normalized into the closed interval $[0,1]$. The embedded dimension l for the ESN-based methods and the other ones are set to 1 and 40, respectively. After the learning process, the trained models continue to run autonomously for another 100 steps.

The successive prediction values generated by ES-KRLS are displayed in Fig. 8. It can be seen that ES-KRLS is successful in predicting a breakdown event that occurs around the 1070-*th* step. The winning entry achieved a normalized mean squared error (NMSE-the mean squared error divided by the variance of the target values) of 0.028 by utilizing a TDNN [39]. The numerical results listed in Table 4 demonstrate ES-KRLS is superior to the other models considered. It is noteworthy that KRLS and ELM outperform RLS-ESN and SVESM in this experiment. This can be explained by the fact that the task of predicting the laser time series iteratively requires both a long-term memory ability and highly nonlinear mapping, because the laser time series changes dramatically over time. However, traditional ESNs with a linear output layer fail to solve the dilemma between long-term memory ability and the strong nonlinearity.

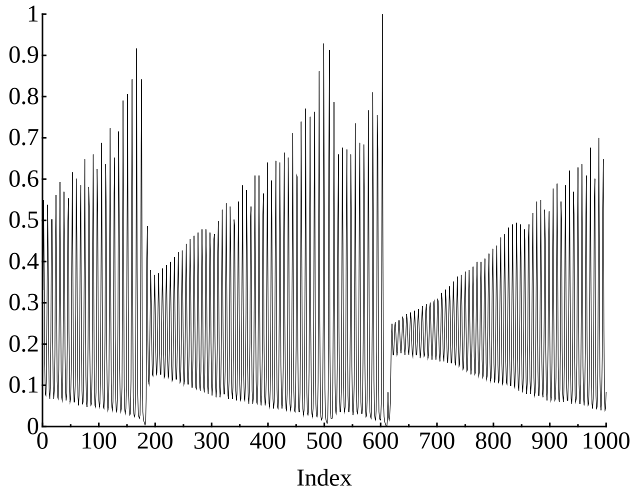


Fig. 7. The laser time series for training.

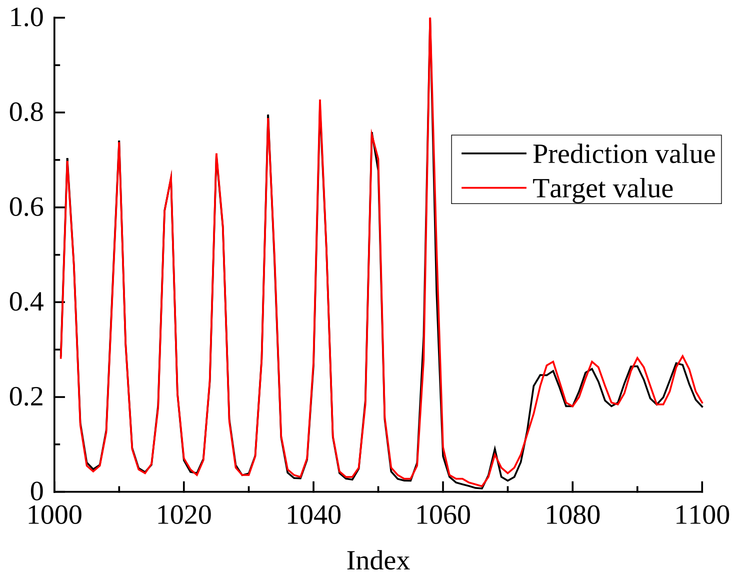


Fig. 8. The prediction performance of ES-KRLS on the laser time series.

Table 4 Experimental results on laser time series.

Algorithm	NMSE	Training time (s)
RLS-ESN	4.446E−1 ± 2.013E−1	4.323
SVESM	3.549E−1 ± 1.650E−1	0.279
ES-KRLS	9.267E−3 ± 1.019E−3	2.374
KRLS	3.731E−2	3.519
KLMS	4.144E−1	0.187
ELM	2.560E−2	0.328

6. Applications

Prognostics focus on predicting the fault growth trend of a degraded system, which plays a crucial role in mission scheduling and maintenance decision-making [40-42]. Prognostic methods are mainly classified into two categories, namely data-driven and model-based methods. Model-based approaches are rooted in the prior knowledge about the underlying failure mechanism. Considering that accurate mathematical models of the complex engineered systems can be unavailable, the practical application of model-based approaches is limited. Data-driven methods provide a viable alternative, inferring the hidden health state of a system from the sensory data directly. Artificial neural network (ANN) is one of the most widely used tools for prognostics and health management. In [43-45], ANNs are employed to build the direct mapping relationship between the degradation signals, which are directly collected from the system or extracted from raw sensory data, and RUL. ANNs can also be used to model the degradation process of an equipment and extrapolate the degradation signals until the predicted value reaches a predefined threshold, whereby RUL is achieved as the time interval between the current time and the failure time [46-48]. In addition, similarity-based approaches for RUL prediction match the degradation pattern of a testing instance to the reference degradation patterns and RUL is derived from the matched training instances [49-51]. Most of these methods require abundant run-to-failure data so as to produce accurate RUL predictions. However, run-to-failure data are possibly unavailable in many practical scenarios. Moreover, the evolution of the health state of a system is subject to system inherent uncertainties and variations in operational conditions. Hence, it is necessary to develop a fault growth model that can adapt to performance degradation over service time.

In this paper, a failure propagation model equipped with an online adaption scheme is developed for machine condition prediction. A linear regression method is first used to convert multi-dimensional sensory data into a one-dimensional health index (HI) that indicates the health state of a system. Then, a state space model that learns the evolution of the HI of the system is constructed. The state model represented by ES-KRLS can be updated recursively using the new observed information. A Bayesian technique is employed to estimate the current health state of the monitored system. Further, the future health states are predicted by keep the state transition function running iteratively.

6.1 Construction of Health index

In order to quantify the degradation level of an aging mechanical system, a HI that is defined in the closed interval $[0,1]$ is constructed. A linear model is utilized to convert the multi-dimensional sensory data $\mathbf{y}_i = [y_i^1, \cdots, y_i^k]$ collected at the *i-th* time step into a noisy HI z_i, which is expressed as

$$z_i = \mathbf{T}\mathbf{y}_i^T \tag{36}$$

where $\mathbf{T} \in \mathbb{R}^{1\times k}$ is the transformation vector. To achieve this linear model, a training dataset should be prepared in advance. The sensory data collected at the beginning and the end of an engineered system's life are stored in two matrices $\mathbf{Y}_1 \in \mathbb{R}^{D_1 \times k}$ and $\mathbf{Y}_0 \in \mathbb{R}^{D_0 \times k}$, respectively. Hence, the transformation vector $\mathbf{T}$ can be obtained by

$$\mathbf{T} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{Z} \tag{37}$$

where $\mathbf{Y} = [\mathbf{Y}_1; \mathbf{Y}_0]$, $\mathbf{S} = [\mathbf{Z}_1; \mathbf{Z}_0]$, $\mathbf{Z}_1 \in \mathbb{R}^{D_1 \times 1}$ is a unity vector, and $\mathbf{Z}_0 \in \mathbb{R}^{D_0 \times 1}$ is a zero vector.

Once the transformation vector $\mathbf{T}$ is available, the health status of the degraded system can be effectively characterized by converting the online monitoring data into a noisy HI based on Eq.(36). The HI of a system is assumed to range between 1 and 0 throughout the whole service life. Specifically, a machine starts from a healthy state with an HI value of 1, and continues to run until a failure occurs, which lead to the value for HI reaching 0.

6.2 Representation of degradation process using state-space model

Due to the utilization of the noisy measurement data, a Bayesian estimation technique is required to track the health condition of a system. For this purpose, a state-space model is used to characterize the degradation process of the system, which is expressed as follows:

$$\text{State model:} \quad s_i = \mathbf{F}(s_{i-1}) + \omega_i \tag{38}$$

$$\text{Measurement model:} \quad z_i = s_i + \upsilon_i \tag{39}$$

where s_i is the actual health state of the system, ω_i is the process noise, υ_i is the measurement noise, and $\mathbf{F}(\cdot)$ is the state transition function denoted by ES-KRLS algorithm.

Given the dynamic system defined in Eq.(38) and Eq.(39), the state estimation is achieved using particle filtering (PF), whereby an approximation to the posterior state possibility distribution function (PDF) $p(s_i \mid z_{1:i})$ is obtained. To be specific, a set of samples (or particles) $\{s_i^n\}_{n=1}^N$ with associated weights $\{w_i^n\}_{n=1}^N$ are used to approximate posterior PDF $p(s_i \mid z_{1:i})$ as follows:

$$p(s_i \mid z_{1:i}) \approx \sum_{n=1}^{N} w_i^n \delta(s_i - s_i^n) \tag{40}$$

where N is the number of particles, and $\delta(\cdot)$ is the Dirac function. The weight w_i^n can be updated iteratively by

$$w_i^n \approx w_{i-1}^n p(z_i \mid s_i^n) \tag{41}$$

6.3 RUL prediction procedure

The detailed procedure for RUL prediction is described as follows:

Step 1: The new observation data $\mathbf{y}_i$ is converted into the HI $\mathbf{z}_i$ via Eq.(36).

Step 2: The weights is updated by Eq.(41), and the actual health state of the monitored system at the *i-th* time step is estimated as

$$\tilde{s}_i \approx \sum_{n=1}^{N} w_i^n s_i^n \tag{42}$$

The resampling strategy is then adopted to avoid the degeneracy problem.

Step 3: A new input-output pair $\{\tilde{s}_{i-1}, \tilde{s}_i\}$ is used to update the state transition function in Eq.(38) according to **Algorithm 1**.

Step 4: A multistep-ahead machine condition prediction is carried out by making one-step-ahead prediction successively. Concretely, the predicted value is fed back to the state transition function $\mathbf{F}(\cdot)$ as the input for another step prediction, as shown below:

$$\hat{s}_{i+p} = \mathbf{F}(\hat{s}_{i+p-1}) \tag{43}$$

When the predicted state reaches the predefined failure threshold, the RUL is estimated as

$$\mathrm{RUL}_i = t_f - i \tag{44}$$

where t_f is the predicted failure time.

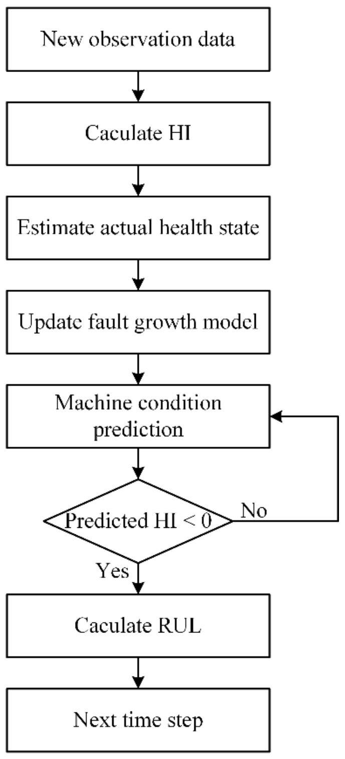

New observation data
Caculate HI
Estimate actual health state
Update fault growth model
Machine condition prediction
Predicted HI < 0
No
Yes
Caculate RUL
Next time step

Fig. 9. The schematic of the novel online prognostic method.

6.4 A Case study in a turbofan engine degradation dataset

In order to verify the feasibility of the proposed prognostic method, a turbofan engine degradation dataset generated by Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) is used [52]. C-MAPSS is a highly detailed, component-level engine model and a schematic of the turbofan engine is depicted in Fig. 10. Each engine unit is represented by a multivariate time series and the data for each flight cycle contain the unit ID, operating cycle index, 3 values that indicates the operational settings and 21 sensor measurements contaminated by high level of unknown noises. Each engine unit operates under six different operational conditions and starts with different level of initial wear caused by manufacturing and assembly variations. The failure threshold is unknown for the users.

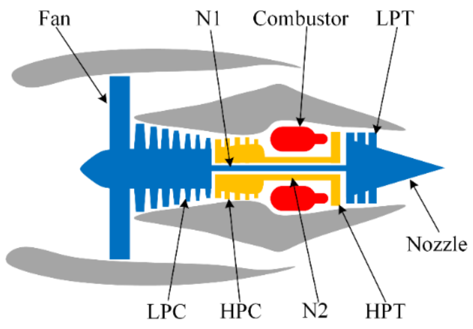


Fig. 10. A schematic of the turbofan engine.

In this work, the 7 sensors, including T24, T30, T50, P30, Ps30, phi and BPR, are selected [49]. The measurement data are normalized into the closed interval $[-1,1]$. Six different transformation vector associated with each operational condition are trained. The sensory data that locate in different condition regimes are converted into the HI using the corresponding linear model. Fig. 13 provides an illustration of the selected sensor signals and the constructed HI under multiple operating conditions. We can notice that large fluctuation in the sensor signals caused by variations in operational conditions overwhelms the influence of performance degradation. By contract, the constructed HI shows a clear trend over time and is thereby capable of providing a better characterization of the degradation process of the turbofan engine.

Fig. 14 depicts the HI estimation results produced by the proposed method. Given a state-space-based degradation model and the noisy HI, the health status is estimated by PF at each operational cycle. The estimated HI is then used to update the state transition function. It is seen that the trained model is able to track the evolution of the health status of the monitored engines. For comparison, the state transition function is learned by different algorithms and the experimental results are provided in Table 5. We can notice that the estimation performance of ES-KRLS is better than those of the other methods. The RUL prediction results generated by the proposed method are depicted in Fig. 15. When the engine units experience the end of service life, the predicted RUL gradually converges to the actual RUL. This can be attributed to the fact that the online adaption scheme keep the degradation model incorporating new observed information.

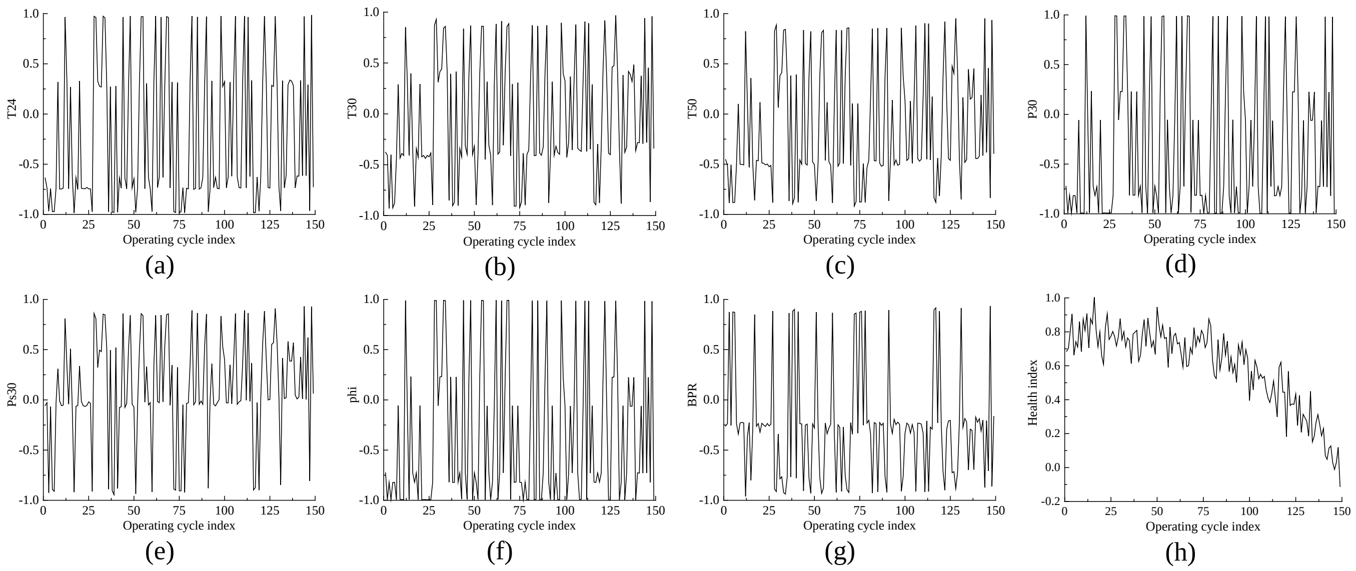


Fig. 13. The selected sensor signals and the constructed HI for engine no. 1.

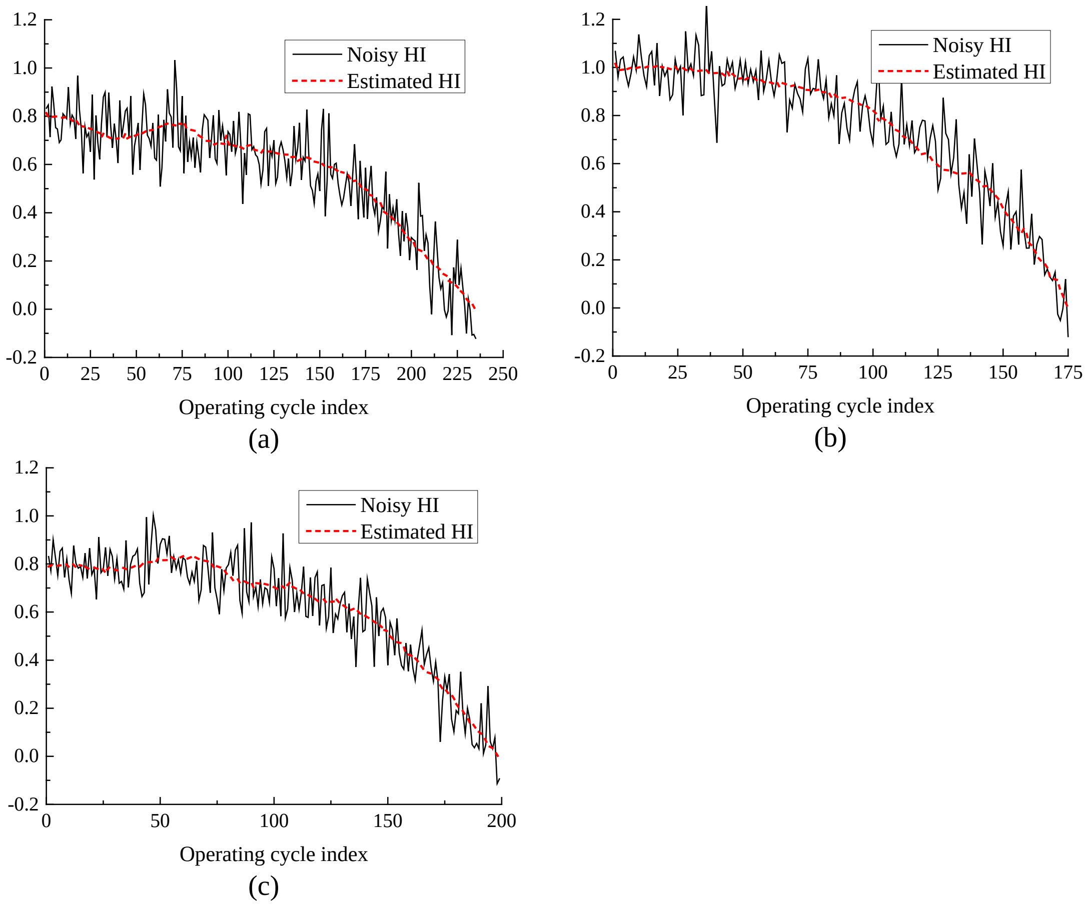


Fig. 14. The HI estimation results produced by the proposed method. (a) Engine unit no. 4; (b) Engine unit no. 6; (c) engine unit no. 9.

Table 5 Performance comparison on HI estimation

Unit number	Algorithms	RMSE
4	RLS-ESN	0.0932
	ES-KRLS	0.0859
	SVESM	0.0952
	KRLS	0.0932
	ELM	0.0887
6	RLS-ESN	0.0948
	ES-KRLS	0.0837
	SVESM	0.1082
	KRLS	0.1057
	ELM	0.0902
9	RLS-ESN	0.0872
	ES-KRLS	0.0797
	SVESM	0.0906
	KRLS	0.0860
	ELM	0.0849

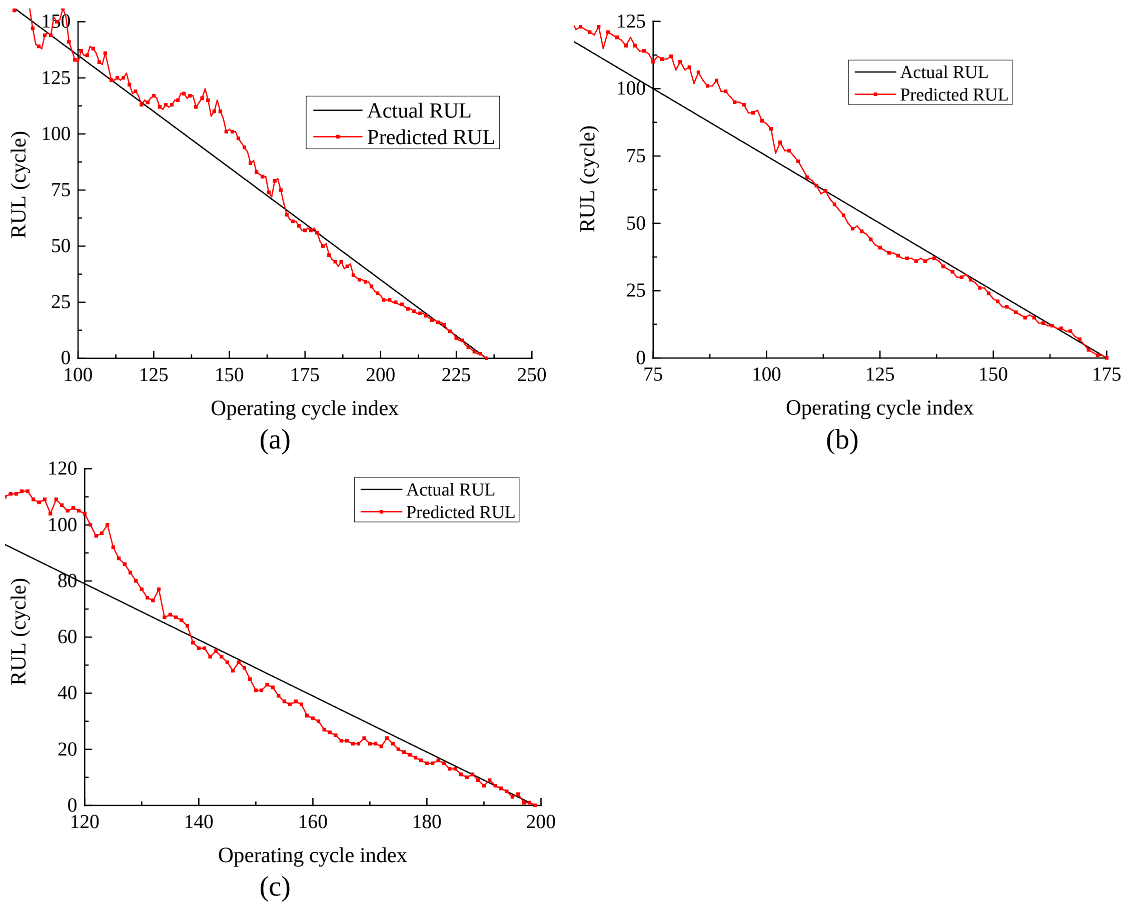


Fig. 15. The RUL prediction results produced by the proposed method. (a) Engine unit no. 4; (b) Engine unit no. 6; (c) Engine unit no. 9.

7. Conclusion

In this paper, we propose a novel ES-KRLS algorithm which bridges the gap between ESN and KAF. Compared with traditional ESNs, ES-KRLS reconstructs the output layer in RKHS by utilizing a kernel-induced nonlinear mapping. As a variant of KAFs, ES-KRLS is able to deal with the temporal dependencies between different data points efficiently due to the utilization of the dynamic reservoir. The fixed topology of the reservoir also facilitates the training procedure. Experiments on benchmark tasks demonstrates the superiority of ES-KRLS algorithm over traditional KAFs, especially in long-term prediction.

In addition, a state space model that captures dynamic characteristics of the degradation process of a system is constructed for health state estimation and RUL prediction. The state transition function represented by ES-KRLS can be updated iteratively using the new observed information. A case study in a degradation dataset illustrates that the online adaption scheme enables the degradation model to adapt to performance degradation of the monitored machine and variations in operational conditions.

Acknowledgments

This research was supported by Funding of Jiangsu Innovation Program for Graduate Education (under Grant KYLX16_0400), the Fundamental Research Funds for the Central Universities, and National Natural Science Foundation of China (under Grant 51276087).

References

[1] V. Vapnik, The nature of statistical learning theory, Springer, New York, 1995.
[2] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem,

Neural Computation, 10 (1998) 1299-1319.
[3] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Computation, 7 (1995) 219-269.
[4] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Transactions on Signal Processing, 52 (2004) 2165-2176.
[5] K. Slavakis, S. Theodoridis, I. Yamada, Online kernel-based classification using adaptive projection algorithms, IEEE Transactions on Signal Processing, 56 (2008) 2781-2796.
[6] W. Liu, P.P. Pokharel, J.C. Principe, The kernel least-mean-square algorithm, IEEE Transactions on Signal Processing, 56 (2008) 543-554.
[7] W. Liu, J.C. Príncipe, Kernel affine projection algorithms, EURASIP Journal on Advances in Signal Processing, 2008 (2008) 784292.
[8] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, IEEE Transactions on Signal Processing, 52 (2004) 2275-2285.
[9] W. Liu, I. Park, Y. Wang, J.C. Príncipe, Extended kernel recursive least squares algorithm, IEEE Transactions on Signal Processing, 57 (2009) 3801-3814.
[10] W. Liu, I. Park, J.C. Principe, An information theoretic approach of designing sparse kernel adaptive filters, IEEE Transactions on Neural Networks, 20 (2009) 1950-1961.
[11] C. Richard, J.C.M. Bermudez, P. Honeine, Online prediction of time series data with kernels, IEEE Transactions on Signal Processing, 57 (2009) 1058-1067.
[12] S. Zhao, B. Chen, Z. Cao, P. Zhu, J.C. Principe, Self-organizing kernel adaptive filtering, EURASIP Journal on Advances in Signal Processing, 2016 (2016) 106.
[13] J.F. Kolen, S.C. Kremer, A field guide to dynamical recurrent networks, Wiley-IEEE Press, New York, 2001.
[14] F. Takens, Detecting strange attractors in turbulence, Dynamical Systems and Turbulence, 898 (1981) 366-381.
[15] K. Funahashi, Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Networks, 6 (1993) 801-806.
[16] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proceedings of the IEEE, 78 (1990) 1550-1560.
[17] R.J. Williams, Training recurrent networks using the extended Kalman filter, Neural Networks, 1992. IJCNN., International Joint Conference on, IEEE,1992, pp. 241-246.
[18] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Computation, 1 (1989) 270-280.
[19] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science, 304 (2004) 78-80.
[20] M. LukošEvičIus, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review, 3 (2009) 127-149.
[21] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Computation, 14 (2002) 2531-2560.
[22] Z. Shi, M. Han, Support vector echo-state machine for chaotic time-series prediction, IEEE Transactions on Neural Networks, 18 (2007) 359-372.
[23] F. Wyffels, B. Schrauwen, A comparative study of reservoir computing strategies for monthly time series prediction, Neurocomputing, 73 (2010) 1958-1964.
[24] H. Cui, C. Feng, Y. Chai, R.P. Liu, Y. Liu, Effect of hybrid circle reservoir injected with wavelet-neurons on performance of echo state network, Neural Networks, 57 (2014) 141-151.
[25] M.C. Soriano, S. Ortín, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, G. Van der Sande, Delay-based reservoir computing: noise effects in a combined analog and digital implementation, IEEE Transactions on Neural Networks and Learning Systems, 26 (2015) 388-393.
[26] M. Xu, M. Han, Adaptive Elastic Echo State Network for Multivariate Time Series Prediction, IEEE Transactions on Cybernetics, 46 (2016) 2173-2183.
[27] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, M. Nuttin, Pruning and regularization in reservoir computing, Neurocomputing, 72 (2009) 1534-1546.
[28] S.P. Chatzis, Y. Demiris, Echo state Gaussian process, IEEE Transactions on Neural Networks, 22 (2011) 1435-1445.
[29] H. Soh, Y. Demiris, Spatio-temporal learning with the online finite and infinite echo-state Gaussian processes, IEEE Transactions on Neural Networks and Learning Systems, 26 (2015) 522-536.
[30] D. Li, M. Han, J. Wang, Chaotic time series prediction based on a novel robust echo state network, IEEE Transactions on Neural Networks and Learning Systems, 23 (2012) 787-799.

[31] H. Jaeger, Adaptive Nonlinear System Identification with Echo State Networks, Advances in Neural Information Processing Systems,2003, pp. 609-616.
[32] M. Han, M. Xu, X. Liu, X. Wang, Online multivariate time series prediction using SCKF- γ ESN model, Neurocomputing, 147 (2015) 315-323.
[33] L. Boccato, A. Lopes, R. Attux, F.J. Von Zuben, An extended echo state network using Volterra filtering and principal component analysis, Neural Networks, 32 (2012) 292-302.
[34] M. Buehner, P. Young, A tighter bound for the echo state property, IEEE Transactions on Neural Networks, 17 (2006) 820-824.
[35] D. Verstraeten, J. Dambre, X. Dutoit, B. Schrauwen, Memory versus non-linearity in reservoirs, Neural Networks (IJCNN), The 2010 International Joint Conference on, IEEE,2010, pp. 1-8.
[36] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing, 70 (2006) 489-501.
[37] Z. Wu, J. Shi, X. Zhang, W. Ma, B. Chen, I. Senior Member, Kernel recursive maximum correntropy, Signal Processing, 117 (2015) 11-16.
[38] A.S. Weigend, Time series prediction: forecasting the future and understanding the past, Santa Fe Institute Studies in the Sciences of Complexity, (1994).
[39] A.S. Weigend, N.A. Gershenfeld, Results of the time series prediction competition at the Santa Fe Institute, Neural Networks, 1993., IEEE International Conference on, IEEE,1993, pp. 1786-1793.
[40] A.K. Jardine, D. Lin, D. Banjevic, A review on machinery diagnostics and prognostics implementing condition-based maintenance, Mechanical Systems and Signal Processing, 20 (2006) 1483-1510.
[41] A. Heng, S. Zhang, A.C. Tan, J. Mathew, Rotating machinery prognostics: State of the art, challenges and opportunities, Mechanical Systems and Signal Processing, 23 (2009) 724-739.
[42] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, D. Siegel, Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications, Mechanical Systems and Signal Processing, 42 (2014) 314-334.
[43] F.O. Heimes, Recurrent neural networks for remaining useful life estimation, Prognostics and Health Management, 2008. PHM 2008. International Conference on, IEEE,2008, pp. 1-6.
[44] M. Rigamonti, P. Baraldi, E. Zio, echo state network for the remaining useful life prediction of a turbofan engine, annual conference of the prognostics and health management society 2015,2016, pp. 255-270.
[45] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, N. Zerhouni, Direct Remaining Useful Life Estimation Based on Support Vector Regression, IEEE Transactions on Industrial Electronics, 64 (2017) 2276-2285.
[46] A. Heng, A.C. Tan, J. Mathew, N. Montgomery, D. Banjevic, A.K. Jardine, Intelligent condition-based prediction of machinery reliability, Mechanical Systems and Signal Processing, 23 (2009) 1600-1614.
[47] E. Ramasso, M. Rombaut, N. Zerhouni, Joint prediction of continuous and discrete states in time-series based on belief functions, IEEE Transactions on Cybernetics, 43 (2013) 37-50.
[48] K. Javed, R. Gouriveau, N. Zerhouni, Novel failure prognostics approach with dynamic thresholds for machine degradation, Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, IEEE,2013, pp. 4404-4409.
[49] T. Wang, J. Yu, D. Siegel, J. Lee, A similarity-based prognostics approach for remaining useful life estimation of engineered systems, Prognostics and Health Management, 2008. PHM 2008. International Conference on, IEEE,2008, pp. 1-6.
[50] E. Zio, F. Di Maio, A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system, Reliability Engineering & System Safety, 95 (2010) 49-57.
[51] Q. Zhang, P.W. Tse, X. Wan, G. Xu, Remaining useful life estimation for mechanical systems based on similarity of phase space trajectory, Expert Systems with Applications, 42 (2015) 2353-2360.
[52] A. Saxena, K. Goebel, D. Simon, N. Eklund, Damage propagation modeling for aircraft engine run-to-failure simulation, Prognostics and Health Management, 2008. PHM 2008. International Conference on, IEEE,2008, pp. 1-9.