IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 9, 2020, accepted September 18, 2020, date of publication September 23, 2020,
date of current version October 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026009

Online Sequential Model for Multivariate Time Series Prediction With Adaptive Forgetting Factor

JINLING DAI[1], AIQIANG XU[1], XING LIU[1], CHAO YU[2], AND YANGYONG WU[1]
[1]Naval Aviation University, Yantai 264001, China
[2]92313 Troops, Jiyuan 454650, China

Corresponding author: Jinling Dai (904227878@qq.com)

This work was supported in part by the Military Pre-Research Project under Grant 3020202090302.

ABSTRACT In the process of online prediction of multivariable non-stationary time series by kernel extreme learning machine (KELM), the dynamic characteristics of the system which are difficult to determine have always posed a big problem. We propose an online sequential prediction model with an adaptive forgetting factor (AFF) for multivariable time series to solve this problem. The multivariable time series instead of variable itself is reconstructed firstly. AFF is introduced into the objective function and can be adjusted iteratively and adaptively with the system changes. As a result, higher weight can be allocated for the fresh and more important samples while the old failure samples can be quickly forgotten. The model sparsification uses a fast leave-one-out cross-validation (FLOO-CV) method to set a prediction error threshold so that samples can be selected conditionally to form a dictionary. Besides, the dictionary parameters, including AFF and kernel parameters, are recursively updated simultaneously without increasing calculation complexity. The experimental results show that, compared with four fashionable KLEM methods, the proposed AFF-OSKELM has a better dynamic tracking ability and adaptability. Moreover, compared with single variable prediction, the spatial reconstructed multivariable has higher prediction accuracy and stability.

INDEX TERMS Kernel extreme learning machine, adaptive forgetting factor, fast leave-one-out cross-validation, online prediction, multivariable time series.

I. INTRODUCTION

Online prediction of non-stationary chaotic time series is an important research direction in the field of science and engineering. It provides an effective method for early fault detection and has wide application and significant importance in financial services, meteorological control, traffic guidance, industrial control, safety control, network monitoring, etc. [1].

Support vector machine (SVM) [2], [3], neural network [4], [5] and other machine learning methods [6] that are used in early sequence prediction. These methods are widely applied in various aspects, including real-time monitoring and control of industrial cyber-physical systems [7], [8] and freight volume forecasting [9], etc. However, they have some defects, such as slow convergence speed and ease of falling into local optimum. Huang *et al.* proposed a simple and efficient single hidden layer feedforward neural network (SLFN) learning algorithm, called extreme learning machine (ELM) to overcome these shortcomings [10]. In terms of learning efficiency, ELM is easy to implement and fast to learn [11]. In terms of the theoretical research, ELM generates random input weights and hidden layer bias values, but it still has the capability for interpolation, general approximation and classification [12]. In terms of structural risk minimization, ELM depends on the number of hidden layer neurons. ELM achieves a balance between training error and model complexity by adjusting the number of hidden layer neurons to achieve optimal generalization performance.

To determine the number of hidden layers, Huang *et al.* [13] suggests implicit mapping in ELM mapping mode, i.e. using kernel method to obtain the inner product between eigenvectors, instead of defining the eigenspace and mapping function explicitly. Additionally, a kernel-based extreme learning machine (KELM) model was proposed. Compared with ELM, KELM transforms the data in the low dimensional input space into the inner product of the high-dimensional feature space, avoiding the problem

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

of dimension disaster [14]. The problem of generalization and stability degradation caused by random assignment of hidden layer neurons is effectively improved by replacing random mapping with kernel mapping, and time is saved for optimizing the number of hidden layer neurons. It shows good application value and development potential in fault identification and sequence prediction of nonlinear systems.

In the practical application of online sequence prediction, samples arrive sequentially in the form of a data stream. The distribution and change trend differ over time which requires the learning model to learn new samples and update [15], [16]. Online sequential ELM (OS-ELM) [17] can perform accurate and efficient online incremental learning of data samples [18]–[21] due to the advantages of ELM. Given that OS-ELM is not a kernel method, its efficiency is not outstanding in dealing with nonlinear and non-stationary data [22]. Developing an online learning method based on KELM (OKELM) is a better choice.

Guo *et al.* [23] proposes an incremental learning method of KELM based on the inverse formula of block matrix termed KB-IELM, which was the first to extend KELM to online learning. In the case of a large amount of training data, KB-IELM has the best performance and fastest speed compared with OS-ELM [24], [25]. As KB-IELM has learned all the samples, its model order is equal to the number of training sample. Therefore, two problems arise: the risk of overfitting, and the linear increase in computational cost. As a result, giving up redundant information and selecting more valuable samples to construct and update the model [26], known as model sparseness [27], becomes a new direction of research.

The sparse KELM-based method has recently yielded a lot of research results. Zhang *et al.* [28] combines the fast leave-one-out cross-validation (FLOO-CV) method with OKELM. FLOO-CV adapts the error threshold and absorbs more valuable samples according to the real-time dictionary. Zhou and Wang [29] proposes a KELM online learning method based on the traditional sliding time window. The method uses the samples in the time window to construct a dictionary, with the model complexity and the generalization performance dependent on the width of the time window. Scardapane *et al.* [30] set up approximate linear independence (ALD) criterion, which improved the generalization performance and reduced the computational complexity by setting the error threshold subjectively. Zhang *et al.* [31], [32] proposed a two-step sparse method including ''constructing'' and ''pruning'' dictionary, respectively based on instantaneous learnable information measurement and cumulative coherence measurement, to realize the different choice of samples.

However, over time the distribution and changing trend of online data change [33], which put forward new model requirements. Zhang *et al.* [34] introduced an adaptive regularization factor to resolve the structural risk of the model in different nonlinear regions. Updating the kernel weight coefficient [35] is carried out additionally to improve the identification accuracy. However, both strategies increase the complexity of the model and affect the computational efficiency. Guo *et al.* [23] puts forward the concept of forgetting factor (FF), making the samples with a closer time in the dictionary have a higher weight in modeling. To obtain better predictive results, Liu *et al.* [36] combined the forgetting factor and the adaptive regularization factor, but it also costs more time.

Multivariate chaotic time series widely exist in natural, economic, social, industrial and other fields. When the prediction model contains multiple variables, the above models seem to be based on their historical time-series information for online prediction. In fact, the variables of a system often interact with each other [37], so the prediction of variables should take into account not only the historical state of variables themselves but also the state of related variables [38]. Therefore, this study aimed to reconstruct the space of variables, to transform the temporal correlation of variables into spatial correlation.

Considering that the change rate of the data stream may be irregular in a complex time-varying environment, the fixed forgetting factor cannot ensure the global adaptability to the dynamic changes of the time-varying system. Inspired by reference [39], [40], the paper introduces an element of the adaptive forgetting factor (AFF). On one hand, AFF should be reduced to accelerate the forgetting of the old failure state and to monitor the latest state of the time-varying system in time when the system is rapidly changing with an increasing prediction error. On the other hand, AFF should be increased to improve the prediction accuracy of the system in a steady state.

The existing KELM methods bring more or less the increase of calculation. In response to the problem, the FLOO-CV method is applied in the online sparseness. The samples which have larger prediction error according to the real-time dictionary are added to the dictionary. On the contrary, the samples are abandoned. At the same time, AFF will be adjusted adaptively. A learning framework based on the above is derived with KELM, it realizes the synchronous updating of kernel weight vector while dictionary sparseness.

The rest of this paper is organized as follows. The multiple variables reconstruction methods and KELM model are introduced in Sect. 2. The concrete process of online sparseness KELM with AFF (AFF-OSKELM) is derived in Sect. 3, including the construction of dictionary and parameter updating. Sect. 4 discusses the method complexity while in Sect. 5, the proposed algorithm is evaluated by both simulation and real-world data. The conclusion is presented in Sect. 6.

II. MATHEMATICAL MODEL

A. RECONSTRUCTION OF MULTIVARIATE TIME SERIES

In the modeling and prediction of multivariate time series, phase space reconstruction is usually needed. According to Takens' embedding delay theorem [41], the delay variable of unit chaotic time series can be used as the surrogate variable

of attractor of reconstructing dynamic system. The phase space reconstruction of unit chaotic time series has become a useful tool for analyzing complex nonlinear systems that widely existing in nature and human world. The results show that the reconstruction accuracy of chaotic time series can be effectively improved by adding multiple coupling time series in the reconstruction.

Suppose there is an M-dimensional multivariable time series: $X_1, X_2, \ldots, X_N$, in which $X_i = (x_{1,i}, x_{1,i}, \ldots, x_{M,i})$, $i = 1, 2, \ldots N$. After reconstruction, multivariable time series can be transformed into $\boldsymbol{v}_n$.

$$\begin{aligned}\boldsymbol{v}_n = [&x_{1,n}, x_{1,n-\tau_1}, \ldots, x_{1,n-(d_1-1)\tau_1},\\ &x_{2,n}, x_{2,n-\tau_1}, \ldots, x_{2,n-(d_1-1)\tau_1},\\ &\ldots\\ &x_{M,n}, x_{M,n-\tau_1}, \ldots, x_{M,n-(d_1-1)\tau_1}]\end{aligned} \tag{1}$$

where τ_i, d_i, $i = 1, \ldots, M$ represent the delay time and embedding dimension of multivariate time series, respectively. According to Takens' embedding theorem [41], if d_i is large enough, there is a mapping:

$$\begin{aligned}F : \boldsymbol{R}^d &\rightarrow \boldsymbol{R}^d(\boldsymbol{d} = \sum\nolimits_{i=1}^{M} d_i)\\ \boldsymbol{v}(n+1) &= F(\boldsymbol{v}(n))\end{aligned} \tag{2}$$

Eq. (2) can also be expressed as:

$$\begin{aligned}&x_1(n+1) = F_1(\boldsymbol{v}(n)),\\ &x_2(n+1) = F_2(\boldsymbol{v}(n)),\\ &\vdots\\ &x_M(n+1) = F_M(\boldsymbol{v}(n)),\end{aligned} \tag{3}$$

The output of the prediction model is defined as $\boldsymbol{y}_n = [y_{1n}, y_{2n}, \ldots, y_{Mn}] = [x_1(n+1), x_2(n+1), \ldots, x_M(n+1)]$, where y_{in} represents the output of a dimension, and $y_{in} = x_i(n+1) = F_i(\boldsymbol{v}(n)), i = 1, \ldots, M$. After determining τ_i and d_i in Eq. (3), the reconstructed vector of a multivariate sequence can be used for modeling and prediction.

B. KELM

At present, most of the neural network models used in time series prediction are based on gradient descent learning algorithm, which has some defects such as slow convergence speed and easy to fall into local optimum. For this reason, ELM is proposed to determine the output weight by linear regression. The ELM optimization problem with equality constraints is defined as follows:

$$\begin{aligned}&\min\ L_{\text{ELM}} = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\frac{1}{2}\sum_{i=1}^{n}\xi_i^2,\\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{x}_i)\cdot\boldsymbol{\beta} = y_i - \xi_i, \quad i = 1, \ldots, n\end{aligned} \tag{4}$$

In Eq. (4), $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_n]^{\mathrm{T}}$ is the output weight connecting the hidden layer and the output layer. $\boldsymbol{h}(\boldsymbol{x}_i) = [h_1(x_i), h_2(x_i), \cdots, h_L(x_i)]$ is the mapping relationship between hidden layer neurons and input samples. ξ_i and y_i represent the error and the target output value corresponding to the input sample. C is the regularization parameter.

Applying KKT optimization conditions into Eq. (4), the calculation formula of output weight $\boldsymbol{\beta}$ [22] can be expressed as Eq. (5). The mapping matrix of the input sample is $\boldsymbol{H} = [\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}_1), \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}_2), \cdots, \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}_n)]^{\mathrm{T}}$.

$$\boldsymbol{\beta} = \boldsymbol{H}^{\mathrm{T}}(C^{-1}\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{y}_n \tag{5}$$

However, the number of hidden layer nodes, which is an important parameter of ELM crucial to the performance of prediction model, usually should be selected by some time-consuming methods according to the learning tasks. By replacing the hidden layer mapping $\boldsymbol{h}(\boldsymbol{x}_i)$ in ELM by the kernel function mapping, KELM is developed. KELM does not to determine the number of hidden layers, avoiding the hidden nodes selection problem. It is proved to have better generalization performance. The kernel matrix is defined as $\boldsymbol{\Omega} = \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$ according to Mercer's conditions, where $\boldsymbol{\Omega}(i,j) = \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x}_j) = \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Suppose $\boldsymbol{A}_n = C^{-1}\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$, then the kernel weight vector is $\boldsymbol{\theta}_n = \boldsymbol{A}_n^{-1}\boldsymbol{y}_n$.

The output of kernel extreme learning machine can be expressed as Eq. (6), where $\boldsymbol{k}_n = [k(\cdot, x_1), k(\cdot, x_2), \cdots, k(\cdot, x_n)]$ represents the kernel estimation vector of the time n.

$$\begin{aligned}f(\cdot) &= \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{h}(\cdot)\boldsymbol{H}^{\mathrm{T}}(C^{-1}\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}y_n\\ &= [k(\cdot, \boldsymbol{x}_1), k(\cdot, \boldsymbol{x}_2), \ldots, k(\cdot, \boldsymbol{x}_n)]\boldsymbol{A}_n^{-1}\boldsymbol{y}_n\\ &= \boldsymbol{k}_n\boldsymbol{\theta}_n\end{aligned} \tag{6}$$

III. ONLINE SPARSENESS KELM WITH AFF

A. KELM WITH AFF

The adaptive forgetting factor is defined as a parameter λ_n of time n, and $0 \ll \lambda_n < 1$. λ_n is adaptively adjusted over time. KELM with AFF can be defined as:

$$\begin{aligned}&\min_{\boldsymbol{\beta},\xi_i}\ L_{\text{KELM}} = \frac{1}{2}\lambda_i||\boldsymbol{\beta}||^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2,\\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{x}_i)\cdot\boldsymbol{\beta} = \boldsymbol{y}_i - \xi_i, \quad i = 1, \ldots, n\end{aligned} \tag{7}$$

To solve Eq. (7) according to the KKT condition, the output weight is shown in Eq. (8), where $\boldsymbol{B} = \text{diag}\{\lambda_{n-1}, \lambda_{n-2}, \cdots, \lambda_1\}$.

$$\boldsymbol{\beta} = (C^{-1}\lambda_n\boldsymbol{I} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{y}_n \tag{8}$$

According to the matrix inversion formula: $(\boldsymbol{A}+\boldsymbol{E}\boldsymbol{C}\boldsymbol{D})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{E}(\boldsymbol{C}^{-1} + \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{E})^{-1}\boldsymbol{D}\boldsymbol{A}$, we can obtain: $(C^{-1}\lambda_n\boldsymbol{I} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{B} = \boldsymbol{H}^{\mathrm{T}}(C^{-1}\lambda_n\boldsymbol{B}^{-1} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}$. Eq. (8) can be transformed into:

$$\boldsymbol{\beta} = \boldsymbol{H}^{\mathrm{T}}(C^{-1}\lambda_n\boldsymbol{B}^{-1} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{y}_n \tag{9}$$

Suppose $\boldsymbol{A}_n = C^{-1}\lambda_n\boldsymbol{B}^{-1} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$, the kernel weight vector can be obtained as follows:

$$\boldsymbol{\theta}_n = \boldsymbol{A}_n^{-1}\boldsymbol{y}_n \tag{10}$$

Furthermore, the output form of KELM integrated with AFF can be expressed as;

$$f(\boldsymbol{x}) = \boldsymbol{h}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{H}^{\mathrm{T}}\boldsymbol{A}_n^{-1}\boldsymbol{y}_n = \boldsymbol{k}_n\boldsymbol{\theta}_n \tag{11}$$

B. DICTIONARY SPARSENESS BASED ON FLOO-CV

Suppose that the reconstruction vector of multivariable time series $\boldsymbol{v}(n)$ in Eq. (1) is represented by $\boldsymbol{x}_n$, and y_n is the predicted target variable value corresponding to $\boldsymbol{x}_n$, the sparse dictionary of n time is defined as $\boldsymbol{D}_n = \{k(\cdot, \boldsymbol{x}_i^n)\}_{i=1}^m$, where $\{\boldsymbol{x}_1^n, \boldsymbol{x}_2^n, \ldots, \boldsymbol{x}_m^n\} \subset \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and m is the scale of dictionary, $m \leq n$. By multiplying both ends of Eq. (10) by the matrix A_n of order m at the same time, we can obtain;

$$\begin{bmatrix} \bar{A}_n & \boldsymbol{V}_n \\ \boldsymbol{V}_n^{\mathrm{T}} & v_n \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{\theta}}_n \\ \theta_m^n \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{y}}_n \\ y_m^n \end{bmatrix} \tag{12}$$

where $\overline{\boldsymbol{\theta}}_n$ is the vector after $\boldsymbol{\theta}_n$ excluding the m-th element θ_m^n; $\bar{\boldsymbol{y}}_n$ is the vector after $\boldsymbol{y}_n$ excluding the m-th element y_m^n; $\overline{\boldsymbol{A}}_n$ is the matrix after $\boldsymbol{A}_n$ excluding the vector of row m and column m; $\boldsymbol{V}_n$ is the vector after $\boldsymbol{A}_n$ excluding the m-th element of column m. And $v_n = C^{-1}\lambda_n + k(\boldsymbol{x}_m^n, \boldsymbol{x}_m^n)$.

Using the sparse matrix from the first $m-1$ sample in the dictionary to predict $\boldsymbol{x}_m^n$, the output weight is $\overline{\boldsymbol{\theta}_n} = \bar{\boldsymbol{A}}_n^{-1}\bar{\boldsymbol{y}}_n$. Therefore, the prediction value of $\boldsymbol{x}_m^n$ can be written as:

$$\hat{\boldsymbol{y}}_m^n = \boldsymbol{V}_n^{\mathrm{T}}\overline{\boldsymbol{\theta}_n} = \boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1}\bar{\boldsymbol{y}}_n \tag{13}$$

From Eq. (12), it can be inferred that:

$$\begin{cases} \bar{\boldsymbol{y}}_n = \overline{\boldsymbol{A}}_n\overline{\boldsymbol{\theta}}_n + \boldsymbol{V}_n\theta_m^n \\ y_m^n = \boldsymbol{V}_n^{\mathrm{T}}\overline{\boldsymbol{\theta}}_n + v_n\theta_m^n \end{cases} \tag{14}$$

By substituting equation (13) into equation (12), we can get:

$$\hat{\boldsymbol{y}}_m^n = \boldsymbol{V}_n^{\mathrm{T}}\overline{\boldsymbol{\theta}_n} + \boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1}\boldsymbol{V}_n\theta_m^n \tag{15}$$

According to Eq. (14) and (15), the estimated FLOO-CV error of $(\boldsymbol{x}_m^n, y_m^n)$ is as follows:

$$\xi_{loo}^{(-m)}(n) = y_m^n - \hat{y}_m^n = (v_n - \boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1}\boldsymbol{V}_n)\theta_m^n \tag{16}$$

Referring to the block form of $\overline{\boldsymbol{A}}_n$ in Eq. (12), we obtain the following equation by the inverse formula of block matrix:

$$\boldsymbol{A}_n^{-1} = \rho_n^{-1}\begin{bmatrix} \rho_n\bar{\boldsymbol{A}}_n^{-1} + \bar{\boldsymbol{A}}_n^{-1}\boldsymbol{V}_n\boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1} & -\bar{\boldsymbol{A}}_n^{-1}\boldsymbol{V}_n \\ -\boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1} & 1 \end{bmatrix} \tag{17}$$

where $\rho_n = C^{-1}\lambda_n + k(\boldsymbol{x}_m^n, \boldsymbol{x}_m^n) - \boldsymbol{V}_n^{\mathrm{T}}\bar{\boldsymbol{A}}_n^{-1}\boldsymbol{V}_n$.

The (i, j) element of $\boldsymbol{A}_n^{-1}$ is represented by $\mathrm{diag}(\boldsymbol{A}_n^{-1})_i$, and Eq. (16) can be simplified into:

$$\xi_{loo}^{(-m)}(n) = \frac{(\boldsymbol{A}_n^{-1}y_n)_m}{\mathrm{diag}(\boldsymbol{A}_n^{-1})_m} \tag{18}$$

Combined with Eq. (17) and (18), the impact of AFF on $\xi_{loo}^{(-m)}(n)$ is only reflected in ρ_n. According to reference [22], it is known that the exchange of element order in the Eq. (12) makes no difference to the solution of the equation. In consequence, the FLOO-CV error of every element in dictionary can be represented as $\xi_{loo}^{(-k)}(n) = \frac{(\boldsymbol{A}_n^{-1}y_n)_k}{\mathrm{diag}(\boldsymbol{A}_n^{-1})_k}$, $k = 1, 2, \ldots, m$. The generalized error vector at time n can be expressed as $E_n = [\xi_{loo}^{(-1)}(n), \xi_{loo}^{(-2)}(n), \ldots, \xi_{loo}^{(-m)}(n)]$. Then the average generalization error of the dictionary can be obtained:

$$\varepsilon_n = \frac{1}{m}\sum_{k=1}^{m}|\xi_{loo}^{(-k)}(n)| \tag{19}$$

When the new sample $(\boldsymbol{x}_{n+1}, y_{n+1})$ of time $n+1$ arrives, the estimated value of y_{n+1} predicted by Eq. (11) is $\hat{y}_{n+1} = \boldsymbol{k}_n\boldsymbol{\theta}_n$. We first determine whether $|y_{n+1} - \hat{y}_{n+1}| > \varepsilon_n$ is true or not. If true, dictionary pruning and adaptive forgetting factor updating are carried out according to Sect. 3.4. Otherwise, it remains unchanged.

C. ADAPTIVE FORGETTING FACTOR

We incorporate an adaptive forgetting factor based on relative error into the KELM model to better monitor the dynamic changes of time-varying systems. First, an intermediate variable is defined as relative error:

$$\phi_n = \mu_1\phi_{n-1} + \mu_2\left|\frac{y_n - \hat{y}_n}{y_n}\right| \tag{20}$$

y_n is real value, while $\hat{y}_n$ is the estimated value predicted by Eq. (11). $\left|\frac{y_n - \hat{y}_n}{y_n}\right|$ is the absolute value of relative error. The intermediate variable of Eq. (20) represents the buffered average value of relative error in the dictionary before n time. μ_1 is the error balance coefficient used to control the weight of the preceding value, and $0 \ll \mu_1 < 1$. μ_2 is the error sensitive coefficient used primarily to control the rate approaching 0, and $0 < \mu_2 \ll 1$. The larger the system convergence error, the smaller the μ_2 and vice versa. Meanwhile, to ensure the monotone descent property in the late convergence stage, $\mu_1 + \mu_2 < 1$.

AFF can be updated according to Eq. (21) on the base of ϕ_n. Among Eq. (21), $0.9 \ll \lambda_+ \leq 1$ represents the upper limit of AFF while $0 \ll \lambda_- < 1$ represents the upper limit. Although the value is related to the specific problem, it should not be too small to influence the instability of the system. AFF is obtained as:

$$\lambda_n = \left[\frac{1}{1+\phi_n}\right]_{\lambda_-}^{\lambda_+} \tag{21}$$

From Eq. (20) and (21), we get that the current prediction error would increase sharply if the system condition changes significantly. Accordingly, ϕ_n increases rapidly in a short time as λ_n decreases. As a result, the old failure samples are quickly forgotten, and by using the latest samples, the new learning model is established to quickly track the latest state of the system. The new predictive model would gradually converge with the continuous learning of new samples. In this process, ϕ_n gradually decreases and approaches to 0, while λ_n correspondingly increases and approaches to 1 to increase AFF of effective data. Theoretically, AFF has both the ability

to track rapidly in a mutation environment and the ability to learn continuously in a steady-state environment. It is active and effective in tracking the time-varying system in real-time, and its implementation process is also very simple and effective.

D. DICTIONARY CONSTRUCTION AND PRUNING

Suppose the current dictionary is $\boldsymbol{D}_n = \{k(\cdot, \boldsymbol{x}_i^n)\}_{i=1}^m$, and $\boldsymbol{A}_n = C^{-1}\lambda_n \boldsymbol{B}^{-1} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}$. When the new samples $(\boldsymbol{x}_{n+1}, y_{n+1})$ arrive, if $m_n < m$, it is the construction stage, otherwise it is a pruning process. In the above, m is the current size of the dictionary while m is the default size.

1) $m_n < m$

For the new sample $(\boldsymbol{x}_{n+1}, y_{n+1})$, the dictionary is updated as $\boldsymbol{D}_{n+1} = \boldsymbol{D}_n \cup k(\cdot, \boldsymbol{x}_{n+1})$. And $\boldsymbol{A}_n$ can be updated as:

$$\boldsymbol{A}_{n+1} = \begin{bmatrix} \boldsymbol{A}_n & \boldsymbol{V}_{n+1} \\ \boldsymbol{V}_{n+1}^{\mathrm{T}} & v_{n+1} \end{bmatrix} \tag{22}$$

where $\boldsymbol{V}_n = [k(\boldsymbol{x}_1, \boldsymbol{x}_{n+1}), k(\boldsymbol{x}_2, \boldsymbol{x}_{n+1}), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}_{n+1})]^{\mathrm{T}}$ is the column $n+1$ of $\boldsymbol{A}_{n+1}$ excluding the $n+1$-th element v_{n+1}. And $v_{n+1} = C^{-1}\lambda_{n+1} + k(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})$.

The inverse matrix of $\boldsymbol{A}_{n+1}$ is expressed in Eq. (23), where $\rho_{n+1} = v_{n+1} - \boldsymbol{V}_{n+1}^{\mathrm{T}} \boldsymbol{A}_n^{-1} \boldsymbol{V}_{n+1}$.

$$\boldsymbol{A}_{n+1}^{-1} = \begin{bmatrix} \boldsymbol{A}_n^{-1} + \boldsymbol{A}_n^{-1}\boldsymbol{V}_{n+1}\rho_{n+1}^{-1}\boldsymbol{V}_{n+1}^{\mathrm{T}}\boldsymbol{A}_n^{-1} & -\boldsymbol{A}_n^{-1}\boldsymbol{V}_{n+1}\rho_{n+1}^{-1} \\ -\rho_{n+1}^{-1}\boldsymbol{V}_{n+1}^{\mathrm{T}}\boldsymbol{A}_n^{-1} & \rho_{n+1}^{-1} \end{bmatrix} \tag{23}$$

According to Eq. (10), the updating of kernel weight vector $\boldsymbol{\theta}_{n+1}$ is as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{A}_{n+1}^{-1}\boldsymbol{y}_{n+1} \tag{24}$$

where $\boldsymbol{y}_{n+1} = [y_1, y_2, \ldots, y_n, y_{n+1}]^{\mathrm{T}}$, with y_i representing the target value of time n.

2) $m_n = m$

When the dictionary reaches its intended size m, Sect. 3.2 will determine whether the new sample is absorbed into the dictionary. If the new sample meets the dictionary pruning condition, it replaces a key point in the current dictionary. The dictionary is updated as $\boldsymbol{D}_{n+1} = \boldsymbol{D}_n^{(-i)} \cup k(\cdot, \boldsymbol{x}_{n+1})$, where $\boldsymbol{D}_n^{(-i)}$ is the dictionary after deleting the i-th key point. The index of i is:

$$i = \arg \min_{i \in \{1,2,\ldots,m\}} \frac{(\boldsymbol{A}_n^{-1} y_n)_i}{\mathrm{diag}(\boldsymbol{A}_n^{-1})_i} \tag{25}$$

To simplify the expression, $k(\boldsymbol{x}_i^n, \boldsymbol{x}_j^n)$ is written as $k_{i,j}$ and $\boldsymbol{A}_n$ is shown in Fig. 1. As shown in Fig. 2, $\tilde{\boldsymbol{A}}_n$ is the matrix after moving the i-th row of $\boldsymbol{A}_n$ to the first row and moving the i-th column to the first column.

$$\boldsymbol{A}_n = \begin{bmatrix} C^{-1}\lambda_{n_1} + k_{1,1} & k_{1,2} & \cdots & k_{1,i} & \cdots & k_{1,m-1} & k_{1,m} \\ k_{2,1} & C^{-1}\lambda_{n_2} + k_{2,2} & \cdots & k_{2,i} & \cdots & k_{2,m-1} & k_{2,m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ k_{i,1} & k_{i,2} & \cdots & C^{-1}\lambda_{n_i} + k_{i,i} & \cdots & k_{i,m-1} & k_{i,m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ k_{m,1} & k_{m,2} & \cdots & k_{m,i} & \cdots & k_{m,m-1} & C^{-1}\lambda_{n_m} + k_{m,m} \end{bmatrix}$$

FIGURE 1. $\boldsymbol{A}_n$ before matrix transformation.

$$\tilde{\boldsymbol{A}}_n = \begin{bmatrix} C^{-1}\lambda_{n_1} + k_{i,i} & k_{i,1} & \cdots & k_{i,i-1} & k_{i,i+1} & \cdots & k_{i,m} \\ k_{1,i} & C^{-1}\lambda_{n_2} + k_{1,1} & \cdots & k_{1,i-1} & k_{1,i+1} & \cdots & k_{1,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k_{i-1,i} & k_{i-1,1} & \cdots & C^{-1}\lambda_{n_{i-1}} + k_{i-1,i-1} & k_{i-1,i+1} & \cdots & k_{i-1,m} \\ k_{i+1,i} & k_{i+1,1} & \cdots & k_{i+1,i-1} & C^{-1}\lambda_{n_{i+1}} + k_{i+1,i+1} & \cdots & k_{i+1,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k_{m,i} & k_{m,1} & \cdots & k_{m,i-1} & k_{m,i+1} & \cdots & C^{-1}\lambda_{n_m} + k_{m,m} \end{bmatrix}$$

FIGURE 2. $\boldsymbol{A}_n$ after matrix transformation.

As shown in Fig. 1 and Fig. 2, $\tilde{\boldsymbol{A}}_n =, \boldsymbol{P}\boldsymbol{A}_n, \boldsymbol{P}^{\mathrm{T}}$, where , $\boldsymbol{P}$ is the Elementary matrix of order m as follows:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \leftarrow i\text{-th rank} \tag{26}$$

There is , $\boldsymbol{P}^{-1} =, \boldsymbol{P}^{\mathrm{T}}$.

Suppose $\tilde{\boldsymbol{A}}_n^{-1} =, \boldsymbol{W}_n$, then we can get that , $\boldsymbol{W}_n = \tilde{\boldsymbol{A}}_n^{-1} = (\boldsymbol{P}\boldsymbol{A}_n, \boldsymbol{P}^{\mathrm{T}})^{-1} = \boldsymbol{P}\boldsymbol{A}_n, \boldsymbol{P}^{\mathrm{T}}$ from the characteristic of P. $\tilde{\boldsymbol{A}}_n^{-1}$ can be transformed into:

$$\tilde{\boldsymbol{A}}_n^{-1} = \begin{bmatrix} \boldsymbol{W}_n^{(1,1)} & \boldsymbol{W}_n^{(1,2:\mathrm{end})} \\ \boldsymbol{W}_n^{(2:\mathrm{end},1)} & \boldsymbol{W}_n^{(2:\mathrm{end},2:\mathrm{end})} \end{bmatrix} \tag{27}$$

$\tilde{\boldsymbol{A}}_n$ can be expressed in block matrix as $\tilde{\boldsymbol{A}}_n = \begin{bmatrix} \bar{v}_n & \overline{\boldsymbol{V}}_n \\ \overline{\boldsymbol{V}}_n^{\boldsymbol{T}} & \boldsymbol{A}_n^{(-i)} \end{bmatrix}$, where $\bar{v}_n = C^{-1}\lambda_{n_i} + k(\boldsymbol{x}_i, \boldsymbol{x}_i)$ and $A_n^{(-i)}$ is the matrix after removing the key point that need to be removed from $\boldsymbol{A}_n$. According to the inverse formula of block matrix, we can get (28), as shown at the bottom of the page, where $\overline{\rho}_n = \overline{v_n} - \overline{\boldsymbol{V}_n}(\boldsymbol{A}_n^{(-i)})^{-1}\overline{\boldsymbol{V}_n^{\mathrm{T}}}$. From Eq. (28) and (29), it can be inferred that that:

$$\begin{aligned} \boldsymbol{W}_n^{(2:\mathrm{end},2:\mathrm{end})} &= (\boldsymbol{A}_n^{(-i)})^{-1}\overline{\boldsymbol{V}}_n^{\mathrm{T}}\overline{\rho}_n^{-1}\overline{\boldsymbol{V}_n}(\boldsymbol{A}_n^{(-i)})^{-1} + (\boldsymbol{A}_n^{(-i)})^{-1} \\ &= \frac{[-(\boldsymbol{A}_n^{(-i)})^{-1}\overline{\boldsymbol{V}}_n^{\mathrm{T}}\overline{\rho}_n^{-1}][-\overline{\rho}_n^{-1}\overline{\boldsymbol{V}_n}(\boldsymbol{A}_n^{(-i)})^{-1}]}{\overline{\rho}_n^{-1}} \\ &\quad + (\boldsymbol{A}_n^{(-i)})^{-1} \\ &= \frac{\boldsymbol{W}_n^{(1,2:\mathrm{end})}\boldsymbol{W}_n^{(2:\mathrm{end},1)}}{\boldsymbol{W}_n^{(1,1)}} + +(\boldsymbol{A}_n^{(-i)})^{-1} \end{aligned} \tag{29}$$

$$\tilde{\boldsymbol{A}}_n^{-1} = \begin{bmatrix} \overline{\rho}_n^{-1} & \overline{\rho}_n^{-1}\overline{\boldsymbol{V}_n}(\boldsymbol{A}_n^{(-i)})^{-1} \\ -(\boldsymbol{A}_n^{(-i)})^{-1}\overline{\boldsymbol{V}_n^{\boldsymbol{T}}}\overline{\rho}_n^{-1} & (\boldsymbol{A}_n^{(-i)})^{-1}\overline{\boldsymbol{V}_n^{\boldsymbol{T}}}\overline{\rho}_n^{-1}\overline{\boldsymbol{V}_n}(\boldsymbol{A}_n^{(-i)})^{-1} + (\boldsymbol{A}_n^{(-i)})^{-1} \end{bmatrix} \tag{28}$$

We can obtain that $(\boldsymbol{A}_n^{(-i)})^{-1} = \boldsymbol{W}_n^{(2:\text{end},2:\text{end})} - \frac{\boldsymbol{W}_n^{(1,2:\text{end})}\boldsymbol{W}_n^{(2:\text{end},1)}}{\boldsymbol{W}_n^{(1,1)}}$ from Eq. (28) and (29). To add the new sample to the dictionary as a key point, then $\boldsymbol{A}_{n+1}$ can be expressed as Eq. (30), where $\boldsymbol{V}_{n+1} = [k_{1,n+1}, \ldots, k_{i-1,n+1}, k_{i+1,n+1} \ldots k_{m,n+1}]^{\mathrm{T}}$ and $v_{n+1} = C^{-1} + k_{n+1,n+1}$.

$$\boldsymbol{A}_{n+1} = \begin{bmatrix} \boldsymbol{A}_n^{(-i)} & \boldsymbol{V}_{n+1} \\ \boldsymbol{V}_{n+1}^{\mathrm{T}} & v_{n+1} \end{bmatrix} \tag{30}$$

According to inverse formula of block matrix, we can obtain (31), as shown at the bottom of the page.

Therefore, the recursion and updating relation of $\boldsymbol{A}_{n+1}^{-1}$ and $\boldsymbol{A}_n^{-1}$ can be obtained from Eq. (27), (29) and (31), and the incremental kernel sparseness of the dictionary is achieved. When the new sample arrives, the updating equation of kernel weight can be obtained as Eq. (32), where $\boldsymbol{y}_{n+1} = [y_1^n, \ldots, y_{i-1}^n, y_{i+1}^n, \ldots, y_m^n, y_{n+1}^n]$.

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{A}_{n+1}^{-1}\boldsymbol{y}_{n+1} \tag{32}$$

IV. ALGORITHM FLOW AND COMPLEXITY ANALYSIS

A. ALGORITHM AND FLOW CHART

Fig. 3 is the schematic of the model structure, and the details can be summarized as Algorithm1:

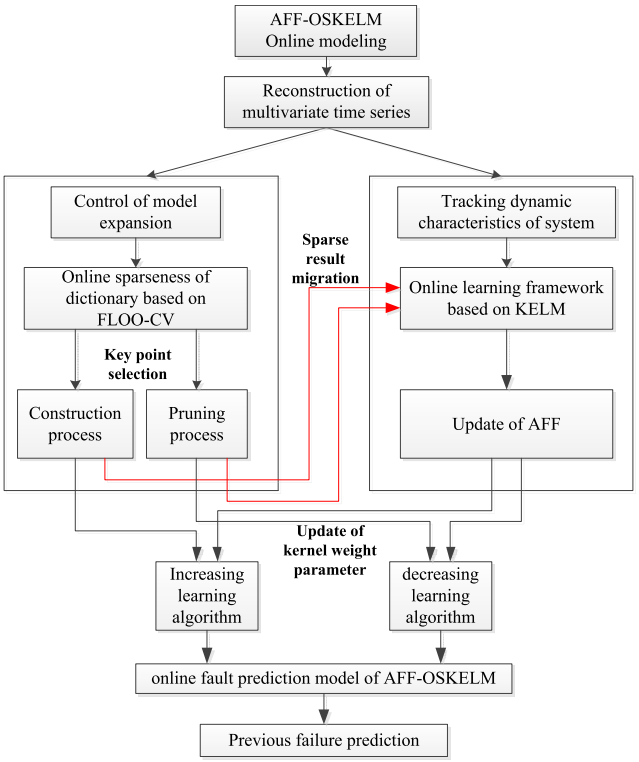


FIGURE 3. Flow chart of AFF-OSKELM.

B. THE ALGORITHM COMPLEXITY

During the dictionary construction process, the algorithm complexity of $\boldsymbol{k}_n$, $\boldsymbol{A}_{n+1}^{-1}$ and are $O(m_n)$, $O(m_n^2)$ and $O(m)$ respectively. During the pruning process, the computational complexity of $\tilde{\boldsymbol{A}}_n^{-1}$, $(\boldsymbol{A}_n^{(-i)})^{-1}$, $\bar{\boldsymbol{k}}_n$, $\boldsymbol{A}_{n+1}^{-1}$ and λ_n are $O(m^2)$, $O((m-1)^2)$, $O(m-1)$, $O(m)$ and $O(m^2)$ respectively. In conclusion, the algorithm complexity of dictionary updating is $O(m)$. The FLOO-CV method only adds the adaptive threshold estimation, of which the computational complexity is $O(m)$. However, the size of the dictionary should not be set too large in the initialization process, so the complexity of the algorithm in this paper is $O(m^2)$, which can fully meet the online prediction needs.

V. EXPERIMENTAL ANALYSIS

To verify the performance, the proposed AFF-OSKELM is compared with four latest KELM methods in this section: (1)KB-IELM [23], (2)ALD-KOS-ELM [20], (3)NOS-KELM [34], (4)FF-OSKELM [22] (fixed forgetting factor which $0 \ll \lambda < 1$). Here OS-ELM methods are not considered for that the randomness of the initial weight setting of ELM input layer can lead to the randomness of the experimental results [19]. And the accuracy of ReOS-ELM is far worse than KELM methods. Simultaneously, the outcomes of single-variable and multi-variable predictions are analyzed. Three examples are given to demonstrate the effectiveness of the proposed method. The examples are classified into an artificial sequence in Sect. 5.1 and two actual sequences in Sect. 5.2 and Sect. 5.3. Multivariable Lorenz chaotic time series is the sequence generated by the differential equation. The actual sequences include annual sunspot numbers time series [34] and time series of Beijing PM2.5.

The computational complexity was measured by training time and test time. The root mean square error (RMSE) is used to measure the performance of predicting accuracy, while maximum absolute prediction error (MAPE) and mean relative error rate (MRPE) are used to measure the prediction stability. They are defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}(i) - y(i)\right|^2}$$

$$\text{MAPE} = \max_{i=1,\cdots,n}\left|\hat{y}(i) - y(i)\right|$$

$$\text{MRPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|\hat{y}(i) - y(i)\right|}{y(i)}$$

$$\boldsymbol{A}_{n+1}^{-1} = \begin{bmatrix} \rho_{\mathrm{n+1}}^{-1}(\boldsymbol{A}_n^{(-i)})^{-1}\boldsymbol{V}_{n+1}\boldsymbol{V}_{n+1}^{\mathrm{T}}(\boldsymbol{A}_n^{(-i)})^{-1} + (\boldsymbol{A}_n^{(-i)})^{-1} & -\rho_{\mathrm{n+1}}^{-1}(\boldsymbol{A}_n^{(-i)})^{-1}\boldsymbol{V}_{n+1} \\ -\rho_{\mathrm{n+1}}^{-1}\boldsymbol{V}_{n+1}^{\mathrm{T}}(\boldsymbol{A}_t^{(-i)})^{-1} & \rho_{\mathrm{n+1}}^{-1} \end{bmatrix} \tag{31}$$

TABLE 1. Parameter setting of example 1.

Methods	Regularization factor	kernel parameter (σ)	other parameters
KB-IELM	2×10^3	10^2	
NOS-KELM	2×10^4	$3*10^3$	$Dic=50$, $\delta=10^{-2}$, $\eta=0.8$
ALD-KOS-ELM	2×10^4	10^5	$\delta=10^{-7}$
FF-OSKELM	2×10^4	10^3	$\lambda=0.998$, $Dic=50$
AFF-OSKELM	2×10^4	$4*10^3$	$\mu_1=0.9$, $\mu_2=0.008$, $\phi_1=0.002$, $Dic=50$

Algorithm 1

Initialization: Set σ, τ_i, d_i, m, λ_+, λ_-, μ_1, μ_2, λ_1

Let $m_n = 1$, $n = 1$, compute $\boldsymbol{A}_1 = C^{-1}\lambda_1 B^{-1} + HH^{\mathrm{T}}$, $\boldsymbol{\theta}_1 = \boldsymbol{A}_1^{-1} y_1$, $D_n = \{k(\cdot, x_1)\}$.
1 $n = n + 1$, new sample $(\boldsymbol{x}_{n+1}, y_{n+1})$ is arrived;
2 If $m_n < m$
3 $m_n = m_n + 1$, $\boldsymbol{D}_{n+1} = \boldsymbol{D}_n \cup k(\cdot, \boldsymbol{x}_{n+1})$;
4 Using (11) to compute $\hat{y}_{n+1} = k_n\theta_n$;
5 Using (20), (21) to update ϕ_{n+1} and λ_{n+1};
6 Using (22) to compute $\boldsymbol{A}_{n+1}$ and (23), (24) to compute $\boldsymbol{\theta}_{n+1}$;
7 else
8 Using (11) to compute $\hat{y}_{n+1} = k_n\theta_n$, and (18), (19) to compute ε_n;
9 If $|y_{n+1} - \hat{y}_{n+1}| \le \varepsilon_n$
10 $\boldsymbol{D}_{n+1} = \boldsymbol{D}_n$, $\lambda_{n+1} = \lambda_n$;
11 else
12 Using (25) to find the index i;
13 $\boldsymbol{D}_{n+1} = \boldsymbol{D}_n^{(-i)} \cup k(\cdot, \boldsymbol{x}_{n+1})$;
14 Using (20), (21) to update ϕ_{n+1} and λ_{n+1};
15 Using (28), (30) to compute $\boldsymbol{A}_{n+1}$ and (31), (32) to compute $\boldsymbol{\theta}_{n+1}$;
16 End if
17 End if
18 Output $\hat{y}_{n+1}$, return to step 1.

All kernel methods use Gaussian kernel as kernel function, i.e., $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\sigma)$. Initialize $\lambda_+ = 1$, $\lambda_- = 1$. All experiments are conducted in Matlab2018a environment running on a Windows 8 PC with Intel Core i5-4210U 2.40GHz CPU and 8GB RAM.

A. LORENZ CHAOTIC TIME SERIES

The section first verifies the effectiveness of the proposed method on Lorenz chaotic time series. Lorenz chaotic equation is a set of three variable differential equations and is expressed as follows:

$$\begin{cases} \mathrm{d}x/\mathrm{d}t = a(y - x) \\ \mathrm{d}y/\mathrm{d}t = bx - y - xz \\ \mathrm{d}z/\mathrm{d}t = xy - cz \end{cases}$$

We apply the fourth-order Runge–Kutta method and let $a = 10$, $b = 28$, $c = 8/3$, $(x(1), y(1), z(1)) = (10, 1, 0)$. Delay times are set as $\tau_1 = \tau_2 = \tau_3 = 2$ and embedding dimension $m_1 = m_2 = m_3 = 6$. As shown in Fig. 4, the samples are used for prediction after processing data of three variables between (4001, 6400). The space reconstruction generates 1200 groups of data, with the previous 1000 groups used to train and the last 200 groups to test.

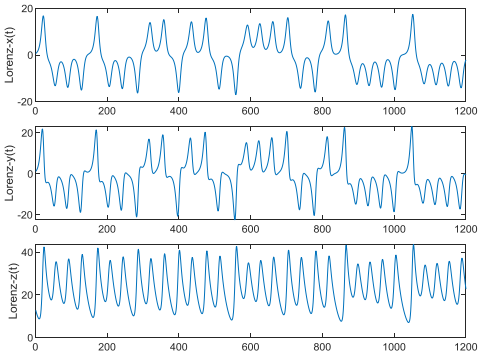


FIGURE 4. Lorenz chaotic time series.

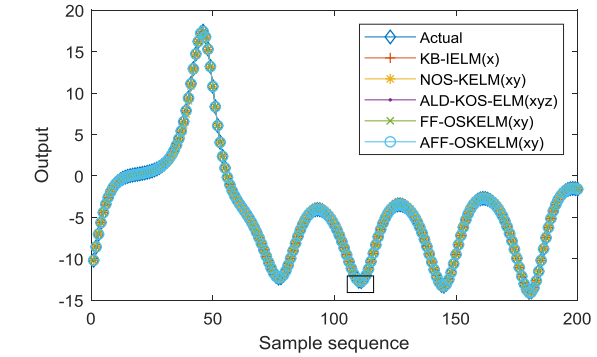
Here variable "x" is set as target prediction variable. As a result, we use variable "x" and reconstructed vector including "x" ("xy", "xz" and "xyz") as input variables. After simulation and comparison, when the parameters of Lorenz chaotic time series are set as shown in Table 1, the best prediction results obtained from each method are shown in Table 2. The second column refers to the input of spatial reconstructed variables. Depending on the specific simulation process, the simulation time has some degree of randomness. The optimal results of each method are marked in bold type. We can get from Table 2 that:

(1) From the perspective of test time, except KB-IELM, the time consumption of all methods is 10^{-4}, which is short enough to satisfy the online prediction requirements [22]. This is because KB-IELM learns all samples in the simulation process, while the other methods conduct online sparing and therefore have lower algorithm complexity.
(2) From the method perspective, the combination of FLOO-CV and KELM has a better prediction performance in all methods, including FF-OSKELM and AFF-OSKELM.
(3) From the perspective of forgetting factor, the prediction accuracy and stability of AFF-OSKELM are 12.9% and 5.8% higher than FF-OSKELM. When the input variable is "xy", AFF-OSKELM is 39.5% and 38.6%

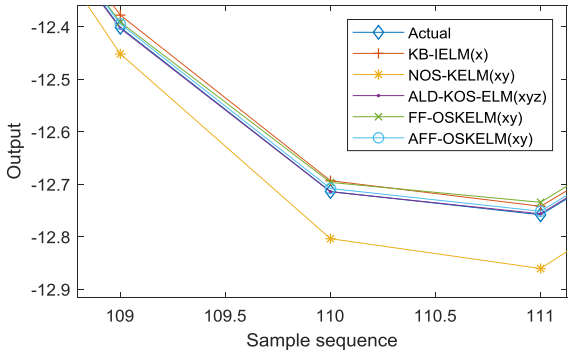
higher than FF-OSKELM in terms of accuracy and stability. AFF-OSKELM with input "xy" increases the test accuracy by a level compared with all methods. This shows the dynamic tracking ability and adaptability of AFF in a time-varying environment.

(4) From the perspective of input variables, the prediction performance of single variable input "x" is nearly the worst compared with other input combinations. The best prediction result of ALD-KOS-ELM is obtained when the reconstruction variables of "xyz" are inputted, while NOS-KELM, FF-OSKELM and AFF-OSKELM get the best results when "xy" is inputted. The consideration of multiple variables is indicated to be of great importance for the prediction of a single variable. Moreover, it can be seen that there is no obvious monotonic correlation between the prediction accuracy and the number of input variables.

The results of the best prediction accuracy of each method (bold in Table 2) are shown in Fig. 5. Among them: (1) Fig. (a) shows that the Lorenz chaotic time series can be effectively monitored by any method; (2) Fig. (b) shows that the local enlarged curve is still close to the actual curve after magnification. The prediction accuracy of AFF-OSKELM is the best compared with other methods, while the prediction effect of NOS-KELM is the worst. This further verified the results in Table 2.



(a) Overall forecast curve



(b) Local enlarged curve

FIGURE 5. The best prediction curve of different methods in example 1.

Fig. 6 shows the APE of the optimal prediction results under each method. Notably, the amplitude of NOS-KELM

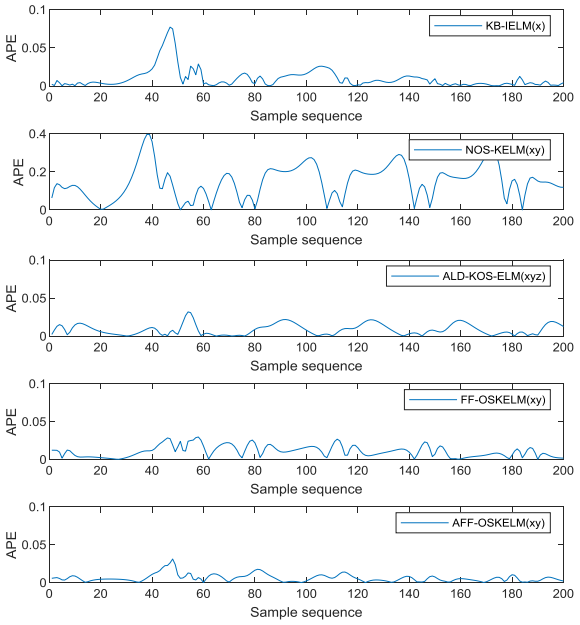


FIGURE 6. APE of different methods in example 1.

in the figure is the highest, and its prediction stability is the worst. At the same time, the dictionary sparseness method based on FLOO-CV error has good stability, and the integration of adaptive forgetting factor further improves the prediction stability.

In the prediction process, the learning curve of each system is represented by RMSE, as shown in Fig. 7. The degree of convergence of the NOS-KELM learning curve is much lower than that of other approaches, therefore it is not evaluated in this section. With the increase of prediction steps, all methods begin to converge at about 50 samples. Among them, AFF-OSKELM can converge to a more accurate stage and has a smoother and more stable learning process.

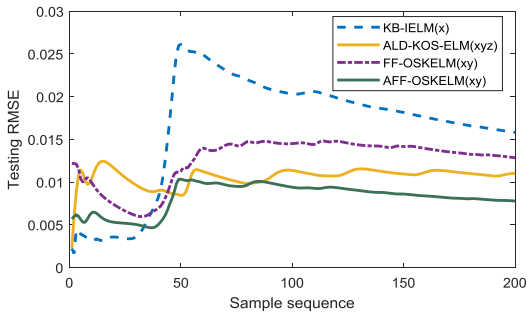


FIGURE 7. The learning curve of different methods in example 1.

Fig. 8 shows the relationship between the number of training samples and time, and Fig. 9 shows the change of AFF during the dictionary construction process. Notably: (1) KB-IELM has the highest computational complexity because it has learned all the samples, and the number of training samples increases monotonously with the input

TABLE 2. The prediction results of example 1.

Methods	Variable	Training time(/s)	Training RMSE	Test time(/s)	Test RMSE	MAPE	MRPE
KBIELM	x	28.1220	0.0084	0.0155	**0.0158**	**0.0771**	0.0044
	xy	28.4925	**0.0063**	**0.0151**	0.0249	0.1527	**0.0016**
	xz	**28.1195**	0.0068	0.0199	0.0569	0.3420	0.0042
	xyz	27.6192	0.0066	0.0164	0.0343	0.1991	0.0032
NOS- KELM	x	**3.8166**	0.2145	$\mathbf{7.1974*10^{-4}}$	0.2128	0.5814	0.1863
	xy	3.8735	**0.1689**	$8.1639*10^{-4}$	**0.1709**	**0.3966**	0.0574
	xz	3.9427	0.6443	$9.7890*10^{-4}$	0.9558	2.1398	0.6358
	xyz	4.2440	0.2283	$8.9808*10^{-4}$	0.2444	0.6483	**0.0573**
ALD-KOS- ELM	x	**1.5615**	0.1599	$\mathbf{5.6622*10^{-4}}$	0.1445	0.6031	0.0355
	xy	1.7709	0.0297	$6.6928*10^{-4}$	0.0183	0.0582	0.0115
	xz	2.2090	0.0783	$7.2573*10^{-4}$	0.0513	0.1308	0.0406
	xyz	1.8813	**0.0140**	$9.0021*10^{-4}$	**0.0110**	**0.0319**	**0.0087**
FF-OSKELM	x	3.6364	0.0201	$7.2445*10^{-4}$	0.0224	0.0497	0.0156
	xy	**3.5859**	**0.0102**	$\mathbf{7.0478*10^{-4}}$	**0.0129**	**0.0296**	**0.0044**
	xz	3.8756	0.0134	$9.1818*10^{-4}$	0.0220	0.0814	0.0095
	xyz	3.6119	0.0234	$9.5089*10^{-4}$	0.0201	0.0956	0.0117
AFF-OSKELM	x	3.6479	0.0236	$\mathbf{7.5918*10^{-4}}$	0.0273	0.0475	0.0140
	xy	3.9584	**0.0070**	$7.7619*10^{-4}$	**0.0078**	**0.0311**	**0.0027**
	xz	**3.5717**	0.0139	$7.7846*10^{-4}$	0.0130	0.0412	0.0112
	xyz	3.7359	0.0144	$7.9070*10^{-4}$	0.0193	0.0503	0.0162

TABLE 3. Parameter setting of example 2.

Methods	Regularization factor	Kernel parameter (σ)	Other parameters
KB-IELM	10^3	10^6	
NOS-KELM	$2*10^4$	10^6	$Dic = 50$, $\delta = 18$, $\eta = 0.8$
ALD-KOS-ELM	$2*10^4$	10^6	$\delta = 3*10^{-7}$
FF-OSKELM	2	$2.5*10^4$	$\lambda = 0.995$, $Dic = 30$
AFF-OSKELM	2	$2.5*10^4$	$\mu_1 = 0.8$, $\mu_2 = 0.008$, $\phi_1 = 0.005$, $Dic = 30$

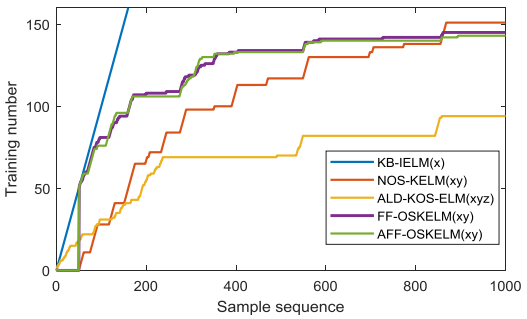


FIGURE 8. Training sample number of different methods in example 1.

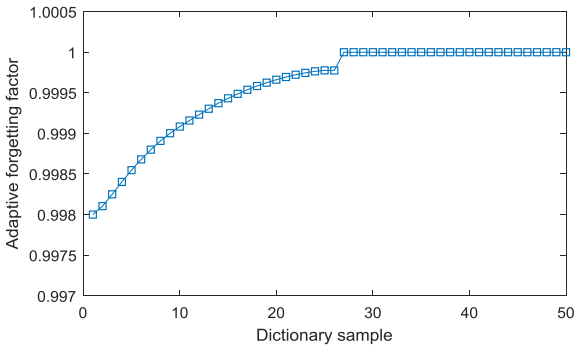


FIGURE 9. AFF changes of example 1.

sample sequence. (2) Although ALD-KOS-ELM has the least number of training samples, it is heavily dependent on the additional threshold parameter. (3) Compared with NOS-KLEM, the FLOO-CV method adopted by AFF-OSKELM and FF-OSKELM converges faster in the training process. (4) AFF-OSKELM and FF-OSKELM use the same dictionary construction method and have similar training curves. However, the integration of AFF further promotes the stability of training samples. (5) In Fig. 9, the fluctuation times of the AFF are consistent with the dictionary size, and eventually, converge to 1. This indicates that the prediction of the Lorenz chaotic time series has no relationship with the sample distance latterly.

B. ANNUAL SUNSPOT NUMBER TIME SERIES

In Example 2, a univariate prediction example is used to verify the performance of AFF. The annual sunspot number time series is a chaotic time series that is nonlinear and non-stationary. The 311 annual sunspot number series samples are reconstructed in space between 1700 and 2010. With the embedding dimension set $m = 10$, 306 samples are obtained. 50 samples are used as prediction and simulation data, while 256 samples are taken as training data.

Through simulation and comparison, when the parameters of sunspot numbers are set as shown in Table 3, the prediction results of each method can be obtained as shown

TABLE 4. The prediction results of example 2.

Methods	Training time(/s)	Training RMSE	Test time(/s)	Test RMSE	MAPE	MRPE
KB-IELM	11.0019	**14.2640**	$7.5609*10^{-4}$	17.4803	47.5091	0.4776
NOS-KELM	1.0119	20.3545	$6.8767*10^{-4}$	23.8857	71.1295	0.4684
ALD-KOS-ELM	**0.3430**	14.6806	$8.8482*10^{-4}$	18.2107	55.3053	**0.3436**
FF-OSKELM	0.7879	17.3149	$\mathbf{2.0442*10^{-4}}$	14.7049	39.5766	0.6570
AFF-OSKELM	0.7117	15.6438	$3.8275*10^{-4}$	**14.1051**	**37.1297**	0.6023

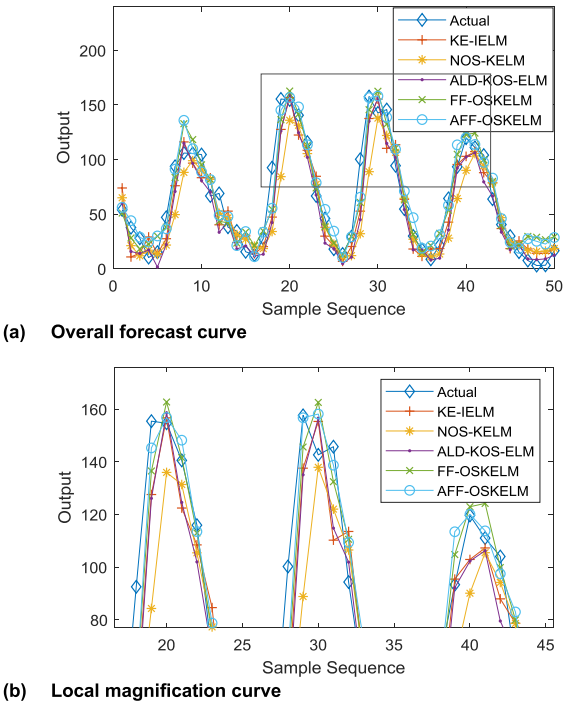


FIGURE 10. Prediction curve of different methods in example 2.

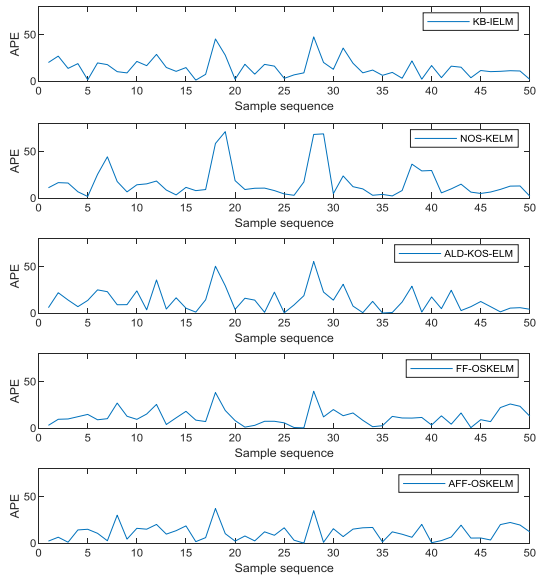


FIGURE 11. APE of different methods in example 2.

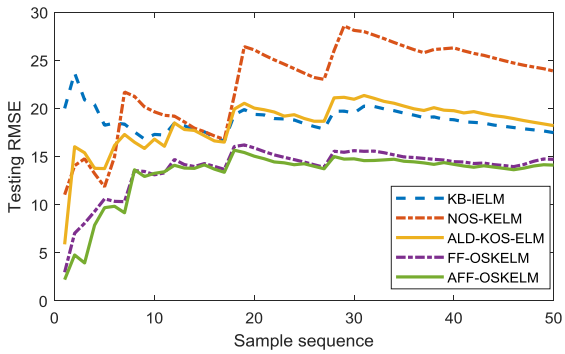


FIGURE 12. The learning curve of different methods in example 2.

in Table 4. By coarsening the optimal results of each method, it is notable that: (1) The time consumption of all methods for single variable prediction with a small sample number is at the level of 10-4, which meets the online prediction requirements. With the additional calculation process of AFF, AFF-OSKELM consumes more time than FF-OSKELM. (2) Although KB-IELM has the highest training accuracy, AFF-OSKELM has the highest predicting accuracy, which is 24.0% average higher than other methods. (3) Besides time consumption, the prediction performance of AFF-OSKELM is better than those of others, which confirms the advantage of AFF.

Although the approximate tracking and prediction of the target can be done by all the methods in Fig. 10(a), AFF-OSKELM is still the nearest to the real curve in Fig. 10(b). As shown in Fig. 11, APE of AFF-OSKELM is similar to that of FF-OSKELM. Close observation shows that the incorporation of AFF makes the amplitude of APE smaller and thus has greater stability.

Fig. 12 shows the learning curve of each method in the testing process. AFF-OSKELM is the smoothest and the outcomes of the subsequent prediction are the most stable. Despite the similar learning curve, AFF-OSKELM is smoother than FF-OSKELM due to the changes of AFF during dictionary construction in Fig. 13.

Fig. 14 shows the impact of the size of dictionary on the testing RMSE of AFF-OSKELM when the other parameters are set as Table 3. In order to keep the dictionary sparsity, only the dictionary size below 100 is compared in Fig.14. At first, the testing RMSE is smaller with the size of dictionary larger and reaches a minimum at 30. Then the accuracy tends to be stable. When the size of dictionary is larger than 50, the testing RMSE becomes unstable. Therefore, we set the size of dictionary as 30.

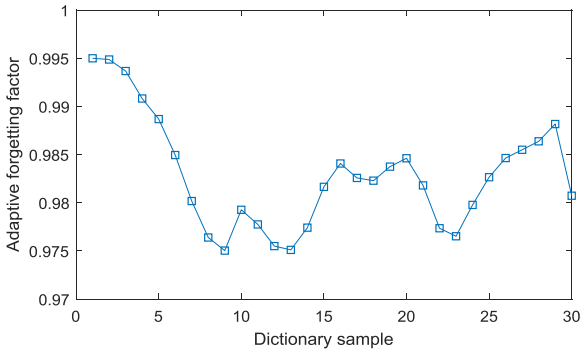


FIGURE 13. AFF changes of example 2.

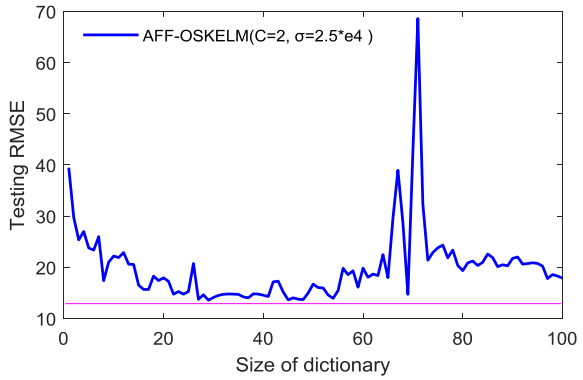


FIGURE 14. The impact of dictionary size on AFF-OSKELM prediction.

The initial AFF value can be obtained by Eq. (21) with parameters in parameter setting table, which is usually the same as FF of FF-OSKELM. For example, we can get $\lambda_1 = 0.995$ of AFF-OSKELM in example 2, equal to FF of FF-OSKELM. As shown in Table 5 is the impact of initial AFF value change on the testing RMSE of AFF-OSKELM. Since $0 \ll \lambda_n < 1$, the initial AFF is set ranging from 0.990 to 0.999 as Table 5. Except for $\lambda_1 = 0.990$, the testing RMSE changes little with λ_1 larger. This is because there are other parameters, e.g. μ_1 and μ_2, that influence the prediction result. As FF is 0.995 in the experiment of FF-OSKELM method, the initial AFF is set 0.955 as well.

TABLE 5. The impact of initial AFF value on prediction.

Initial AFF Value	0.990	0.991	0.992	0.993	0.994
Testing RMSE	14.3434	14.1056	14.1055	14.1053	14.1052
Initial AFF Value	0.995	0.996	0.997	0.998	0.999
Testing RMSE	14.1051	14.1050	14.1049	14.1048	14.1047

C. PM2.5 TIME SERIES OF BEIJING

PM2.5 refers to the particles in the atmosphere with an aerodynamic equivalent diameter of less than 2.5 mm. The higher the concentration of PM2.5 in the air, the more serious the air pollution is, which has a significant impact on human health, air quality and meteorological visibility. After long-term observation, the concentration of PM2.5 was found to

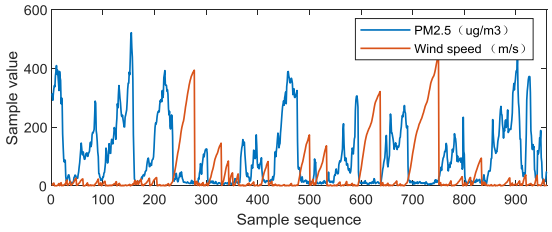


FIGURE 15. PM2.5 time series of Beijing.

be closely related to the change of wind speed. The higher wind speed is, the smaller concentration of PM2.5 is, and vice versa. Hence this segment forecasts PM2.5 according to Beijing's historic PM2.5 time series and wind speed.[1]

As shown in Fig. 15, 960 samples of PM2.5 (μg/m^3) and wind speed (m/s) of 40 days in Beijing from November 22 to December 31, 2014, are used in the experiment. Set time delay as $\tau_1 = \tau_2 = 1$h and embedding dimension as $m_1 = m_2 = 5$. The first 720 groups were used as training samples and the last 240 groups as test samples.

Through simulation and comparison, the experimental results can be obtained in Table 7 when the experimental parameters are set as shown in Table 6. The second column input "1" refers to the reconstruction vector with input variables of PM2.5. "12" refers to the reconstruction vector with input variables of PM2.5 and wind speed. The bold part represents the optimal index for all methods. Table 7 indicates that:

(1) From the perspective of time consumption, the test time of AFF-OSKELM and FF-OSKELM is the best, which verifies the low computational complexity of this method.
(2) From the perspective of test accuracy, compared with other methods, the input "1" and "12" of AFF-OSKELM can improve the test accuracy by 2.88% and 3.94% respectively. Besides, the addition of AFF is confirmed to have a positive effect on the predictive effect.
(3) From the perspective of stability, since KB-IELM learns all the samples, its MRPE performance is the best but also has the highest time cost. The worst MRPE performance notwithstanding, the MAPE index of AFF-OSKELM and FF-OSKELM is the best, which indicates certain advantages in stability.
(4) From input variables, multi-variable input predictions are 2.87% higher than single-variable input predictions. This proves that the consideration of multivariable is of great significance.

The predicted overall curve, local enlarged curve and root mean square error of AFF-OSKELM with the input "12" in Table 7 are shown in Fig. 16. The prediction curve can accurately track the change trend of the real curve, and its prediction error remains at a low level.

[1] https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

TABLE 6. Parameter setting of example 3.

Methods	Regularization factor	Kernel parameter(σ)	Other parameters
KB-IELM	200	$4*10^6$	
NOS-KELM	4	$4*10^6$	$Dic=100$, $\delta=10^{-2}$, $\eta=0.8$
ALD-KOS-LM	2	$4*10^5$	$\delta=10^{-6}$
FF-OSKELM	4	$4*10^5$	$\lambda=0.9997$, $Dic=100$
AFF-OSKELM	4	$4*10^5$	$\mu_1=0.89$, $\mu_2=0.0045$, $\phi_1=0.00012$, $Dic=100$

TABLE 7. The prediction results of example 3.

Methods	Variable	Training time(/s)	Training RMSE	Test time(/s)	Test RMSE	MAPE	MRPE
KB-	1	13.3283	27.0644	0.0144	30.6049	169.4303	0.3152
IELM	12	12.7353	30.5964	0.0131	30.5964	168.3591	**0.2996**
NOS-	1	4.2357	26.2164	0.0013	31.4540	162.6895	0.3493
KELM	12	4.0697	**25.1698**	0.0015	30.9982	167.5076	0.2957
ALD-	1	**1.1443**	27.4771	0.0019	32.6621	177.2775	0.3974
KOS-ELM	12	3.6401	24.9572	0.0039	29.9865	174.1098	0.3343
FF-	1	3.3745	26.2710	0.0011	29.9974	**162.4091**	0.7129
OSKELM	12	3.2464	25.2150	0.0012	29.4089	163.2418	0.5426
AFF-	1	3.3228	26.2937	**0.0011**	29.9921	162.8217	0.7119
OSKELM	12	3.2856	25.2080	0.0013	**29.4013**	163.2847	0.5406

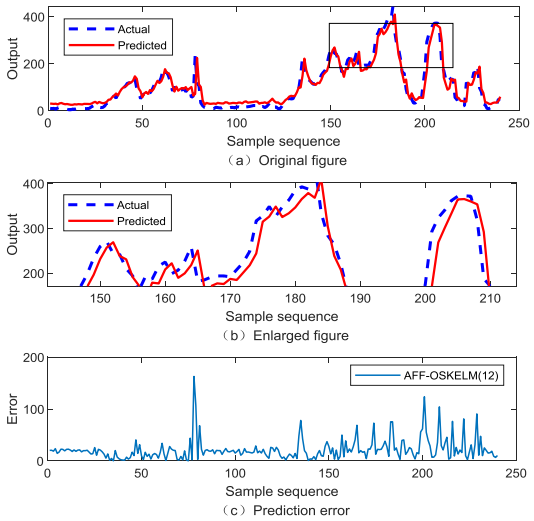


FIGURE 16. Prediction results of AFF-OSKELM under input "12" in example 3.

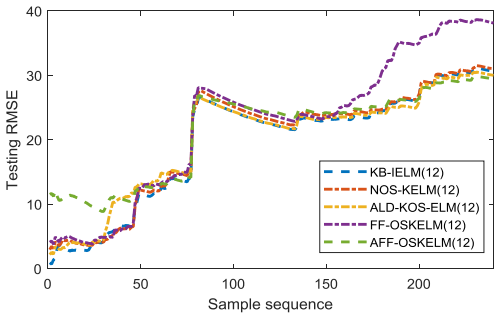


FIGURE 17. The learning curve of different methods in example 3.

Fig. 17 shows the prediction learning curve of the optimal prediction results, i.e. when the input variable is ''12''. The learning curves of all methods tend to be stable except for

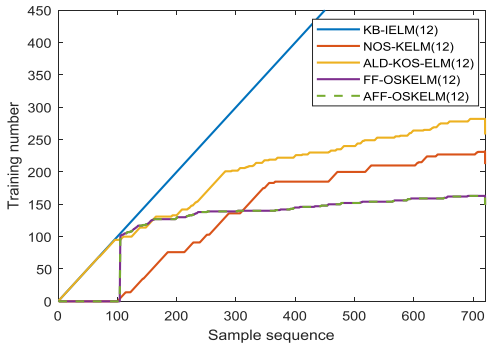


FIGURE 18. Training number of different methods in example 3.

FF-OSKELM and the AFF-OSKELM curve is the smoothest. The prediction result of AFF-OSKELM is more accurate. Accordingly, Fig. 18 shows the number of training samples for each method in Fig. 17. The AFF-OSKELM and FF-OSKELM training curves are very similar, and their training samples are less than other methods, as a result of which their time consumption is the lowest in Table 7.

VI. CONCLUSION

We propose a new multivariable time series prediction model with AFF, namely AFF-OSKELM, to enhance the dynamic tracking capability and improve prediction performance and timeliness of non-stationary time series. The experimental results show that: (1) The proposed method has low computational complexity; (2) The AFF-OSKELM method has higher prediction accuracy and stability than other KELM based methods; (3) In the case of multivariable time series, the prediction effect of input multi-variable is better than that based on a single variable.

The proposed method has the following advantages: (1) The AFF concept is introduced which can more effectively track the rate of change in the data flow in time-varying situations; (2) The outcome of the target prediction is more reliable given the changes in the related variables; (3) A general learning framework is established to realize the process of dictionary sparseness without increasing the computational complexity. Finally, there is a synchronous updating of the kernel weight coefficient and AFF.

In the future, we will also explore the effects of kernel type and kernel parameters on the prediction performance of AFF-OSKELM.

ACKNOWLEDGMENT

The author Jinling Dai thanks for the data support from Chao Yu and the help from the laboratory.

REFERENCES

[1] Z. Tian, C. Qian, B. Gu, L. Yang, and F. Liu, "Electric vehicle air conditioning system performance prediction based on artificial neural network," *Appl. Thermal Eng.*, vol. 89, pp. 101–114, Oct. 2015, doi: 10.1016/j.applthermaleng.2015.06.002.

[2] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 24–38, May 2009, doi: 10.1109/MCI.2009.932254.

[3] D. M. Zhou, F. Gao, and X. H. Guan, "Application of accurate online support vector regression in energy price forecast," presented at the World Congr. Intell. Control Automat., Jun. 2004.

[4] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 359–372, Mar. 2007, doi: 10.1109/TNN.2006.885113.

[5] R. Pino, J. Parreno, A. Gomez, and P. Priore, "Forecasting next-day price of electricity in the spanish energy market using artificial neural networks," *Eng. Appl. Artif. Intell.*, vol. 21, no. 1, pp. 53–62, Feb. 2008, doi: 10.1016/j.engappai.2007.02.001.

[6] Y. Bai and T. Li, "Robust fuzzy inference system for prediction of time series with outliers," in *Proc. Int. Conf. Fuzzy Theory Appl. (iFUZZY)*, Nov. 2012, pp. 16–18, doi: 10.1109/iFUZZY.2012.6409738.

[7] Y. Jiang, S. Yin, and O. Kaynak, "Data-driven monitoring and safety control of industrial cyber-physical systems: Basics and beyond," *IEEE Access*, vol. 6, pp. 47374–47384, 2018, doi: 10.1109/ACCESS.2018.2866403.

[8] S. Yin, J. J. Rodriguez-Andina, and Y. Jiang, "Real-time monitoring and control of industrial cyberphysical systems: With integrated plant-wide monitoring and control framework," *IEEE Ind. Electron. Mag.*, vol. 13, no. 4, pp. 38–47, Dec. 2019, doi: 10.1109/MIE.2019.2938025.

[9] S. Yin, Y. Jiang, Y. Tian, and O. Kaynak, "A data-driven fuzzy information granulation approach for freight volume forecasting," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 1447–1456, Feb. 2017, doi: 10.1109/TIE.2016.2613974.

[10] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, Jun. 2011, doi: 10.1007/s13042-011-0019-y.

[11] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006, doi: 10.1109/TNN.2006.875977.

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006, doi: 10.1016/j.neucom.2005.12.126.

[13] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012, doi: 10.1109/tsmcb.2011.2168604.

[14] R. Xu, X. Liang, J. S. Qi, Z. Y. Li, and S. S. Zhang, "Advances and trends in extreme learning machine," *Chin. J. Comput.*, vol. 42, no. 13, pp. 1–32, Jul. 2019.

[15] W. Guo, J. J. Yu, K. M. Tang, and T. Xu, "Survey of online sequential extreme learning algorithm for dynamic data stream analysis," *Comput. Sci.*, vol. 46, no. 4, pp. 1–7, Apr. 2019, doi: 10.11896/j.issn.1002-137X.2019.04.001.

[16] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006, doi: 10.1109/TNN.2006.880583.

[17] C, B. Lu, Y and Mei, "An accurate and robust online sequential learning algorithm for feedforward networks," *J. Shanghai Jiaotong Univ.*, vol. 49, no. 8, pp. 1137–1143, Aug. 2006, doi: 10.16183/j.cnki.jsjtu.2015.08.010.

[18] H. T. Huynh and Y. Won, "Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks," *Pattern Recognit. Lett.*, vol. 32, no. 14, pp. 1930–1935, Oct. 2011, doi: 10.1016/j.patrec.2011.07.016.

[19] S. Huang, B. Wang, J. Qiu, J. Yao, G. Wang, and G. Yu, "Parallel ensemble of online sequential extreme learning machine based on MapReduce," *Neurocomputing*, vol. 174, pp. 352–367, Jan. 2016, doi: 10.1016/j.neucom.2015.04.105.

[20] P. K. Wong, X. H. Gao, K. I. Wong, and C. M. Vong, "Online extreme learning machine based modeling and optimization for point-by-point engine calibration," *Neurocomputing*, vol. 277, pp. 187–197, Feb. 2018, doi: 10.1016/j.neucom.2017.02.104.

[21] X. Jia, R. Wang, J. Liu, and D. M. W. Powers, "A semi-supervised online sequential extreme learning machine method," *Neurocomputing*, vol. 174, pp. 168–178, Jan. 2016, doi: 10.1016/j.neucom.2015.04.102.

[22] M. Zhu, A. Q. Xu, Q. Q. Chen, and R. F. Li, "An improved KELM based online condition prediction method," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 45, no. 7, pp. 1370–1379, Jul. 2019.

[23] L. Guo, J. H. Hao, and M. Liu, "An incremental extreme learning machine for online sequential learning problems," *Neurocomputing*, vol. 128, pp. 50–58, Mar. 2014, doi: 10.1016/j.neucom.2013.03.05.5.

[24] M. Han, R. Zhang, and M. Xu, "Multivariate chaotic time series prediction based on ELM–PLSR and hybrid variable selection algorithm," *Neural Process. Lett.*, vol. 46, no. 2, pp. 705–717, Apr. 2017, doi: 10.1007/s11063-017-9616-4.

[25] J. S. Lim, S. Lee, and H. S. Pang, "Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimations," *Neural Comput. Appl.*, vol. 22, pp. 569–576, Feb. 2012, doi: 10.1007/.s00521-012-0873-x.

[26] X. YU, M. J. RASHID FENG, J. Y. Feng, and N. Hobbs, "Online glucose prediction using computationally efficient sparse kernel filtering algorithms in type-1 diabetes," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 1, pp. 3–15, Jan. 2018, doi: 10.1109/TCST.2018.2843785.

[27] X. Zhou, Z. Liu, and C. Zhu, "Online regularized and kernelized extreme learning machines with forgetting mechanism," *Math. Problems Eng.*, vol. 2014, pp. 1–11, Jul. 2014.

[28] Y. T. Zhang, C. Ma, Z. N. Li and H, B, Fan, "Online modeling of kernel extreme learning machine based on fast leave-one-out crosss-validation," *J. Shanghai Jiaotong Univ.*, vol. 48, no. 5, pp. 641–646, May 2014. [Online]. Available: https://xueshu.baidu.com/usercenter/paper/show?paperid=f8c06a978be9a67721189656e0280167&site=xueshu_se&hitarticle=1

[29] X.-R. Zhou and C.-S. Wang, "Cholesky factorization based online regularized and kernelized extreme learning machines with forgetting mechanism," *Neurocomputing*, vol. 174, pp. 1147–1155, Jan. 2016, doi: 10.1016/j.neucom.2015.10.033.

[30] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "Online sequential extreme learning machine with kernels," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2214–2220, Sep. 2015, doi: 10.1109/TNNLS.2014.2382094.

[31] W. Zhang, A. Q. Xu, and M. Z. Gao, "Online condition prediction of avionic devices based on sparse kernel incremental extreme learning machine," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 43, no. 10, pp. 1–10, Oct. 2017, doi: 10.13700/j.bh.1001-5965.2016.0802.

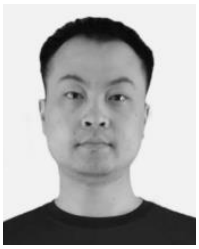
[32] W. Zhang, A. Q. Xu, and M. Z. Gao, "An online condition prediction algorithm based on cumulative coherence measurement," *J. Shanghai Jiaotong Univ.*, vol. 51, no. 11, pp. 1391–1398, Nov. 2017, doi: 10.16183/j.cnki.jsjtu.2017.11.016.

[33] J. Sun, H. Fujita, P. Chen, and H. Li, "Dynamic financial distress prediction with concept drift based on time weighting combined with AdaBoost support vector machine ensemble," *Knowl.-Based Syst.*, vol. 120, pp. 4–14, Mar. 2017, doi: 10.1016/j.knosys.2016.12.019.

[34] W. Zhang, A. Xu, D. Ping, and M. Gao, "An improved kernel-based incremental extreme learning machine with fixed budget for nonstationary time series prediction," *Neural Comput. Appl.*, vol. 31, no. 3, pp. 637–652, Mar. 2019, doi: 10.1007/s00521-017-3096-3.
[35] W. Zhang, A. Q. Xu, and D. F. Ping, "Nonlinear system online identification based on kernel sparse learning algorithm with adaptive regulation factor," *Syst. Eng. Electron.*, vol. 39, no. 1, pp. 223–230, Jan. 2017. [Online]. Available: https://xueshu.baidu.com/usercenter/paper/show?paperid=f8c06a978be9a67721189656e0280167&site=xueshu_se&hitarticle=1
[36] X. Liu, H. Q. Xiong, J. Y. Zhao, and M. Zhu, "Online based on improved sparse KELM non-stationary dynamic system state prediction," *Syst. Eng. Electron.*, to be published. [Online]. Available: http://kns.cnki.net/kcms/details/11.2422.tn.20 2003 23.2114.002html
[37] X. Wang and M. Han, "Multivariate time series prediction based on multiple Kernel extreme learning machine," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 198–201.
[38] X. Y. Wang and M. Han, "Multivariate chaotic time series prediction using multiple Kernel extreme learning machine," *Acta Phys. Sinica*, vol. 64, no. 7, pp. 129–135, Sep. 2014, doi: 10.7498/aps.64.070504.
[39] W. Cao, Z. Ming, Z. Xu, J. Zhang, and Q. Wang, "Online sequential extreme learning machine with dynamic forgetting factor," *IEEE Access*, vol. 7, pp. 179746–179757, 2019, doi: 10.1109/ACCESS.2019.2959032.
[40] B. Mirza and Z. Lin, "Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification," *Neural Netw.*, vol. 80, pp. 79–94, Aug. 2016, doi: 10.1016/j.neunet.2016.04.008.
[41] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick* (Lecture Notes in Mathematics), vol. 898, D. A. Rand and L.-S. Young, Eds. Springer-Verlag, Nov. 2006, pp. 366–381, doi: 10.1007/BFb0091924.

JINLING DAI was born in Jiangsu, China, in 1991. She received the B.S. and M.S. degrees from Naval Aviation University, Yantai, China, in 2014 and 2016, respectively, where she is currently pursuing the Ph.D. degree in information and communication engineering. Her research interests include equipment test and diagnosis technology.

AIQIANG XU was born in 1963. He received the B.S. degree from Naval Aviation University, Yantai, China, in 1983, and the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2005. He is currently a Professor and a Ph.D. Supervisor with the Department of Scientific Research, Naval Aviation University, as well as a Taishan Scholar of Shandong, China. His research interests include testing and diagnosis of the electronic information systems.



XING LIU received the B.S. and M.S. degrees from Naval Aviation University, Yantai, China, in 2008 and 2010, respectively, where he is currently pursuing the Ph.D. degree in military equipment. His research interests include equipment test and diagnosis technology.

CHAO YU received the B.S. degree from Naval Aviation University, Yantai, China, in 2015.

YANGYONG WU received the B.S. degree from Naval Aviation University, Yantai, China, in 2016, where he is currently pursuing the M.S. degree in instrument science and technology. His research interests include test and diagnosis technology of electronic equipment.