energies

MDPI

Article

Adaptive Online Sequential Extreme Learning Machine with Kernels for Online Ship Power Prediction

Xiuyan Peng [1,†], Bo Wang [1,*,†], Lanyong Zhang [1] and Peng Su [2]

[1] College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China; pengxiuyan@hrbeu.edu.cn (X.P.); zlyalf@sina.com (L.Z.)
[2] China Ship Development and Design Center, Wuhan 430064, China; supeng@hrbeu.edu.cn
* Correspondence: excalibur@hrbeu.edu.cn
† These authors contributed equally to this work.

Abstract: With the in-depth penetration of renewable energy in the shipboard power system, the uncertainty of its output power and the variability of sea conditions have brought severe challenges to the control of shipboard integrated power system. In order to provide additional accurate signals to the power control system to eliminate the influence of uncertain factors, this study proposed an adaptive kernel based online sequential extreme learning machine to accurately predict shipboard electric power fluctuation online. Three adaptive factors are introduced, which control the kernel function scale adaptively to ensure the accuracy and speed of the algorithm. The electric power fluctuation data of real-ship under two different sea conditions are used to verify the effectiveness of the algorithm. The simulation results clearly demonstrate that in the case of ship power fluctuation prediction, the proposed method can not only meet the rapidity demand of real-time control system, but also provide accurate prediction results.

Keywords: extreme learning machine; online sequential learning; ship power forecasting; adaptive factor; photovoltaic system

Citation: Peng, X.; Wang, B.; Zhang, L.; Su, P. Adaptive Online Sequential Extreme Learning Machine with Kernels for Online Ship Power Prediction. *Energies* **2021**, *14*, 5371. https://doi.org/10.3390/en14175371

Academic Editors: Abbas Mardani and Djaffar Ould-Abdeslam

Received: 19 May 2021
Accepted: 24 August 2021
Published: 29 August 2021

1. Introduction

1.1. Background and Motivation

With the development of ship power system electrification and the penetration of renewable energy, the integrated power system (IPS) with renewable energy power generation system such as photovoltaic (PV) power generation system has become an important development direction of ship electrification [1,2].

In order to provide stable power quality for shipboard IPS, it is of great importance to ensure the power balance and stability between supply side and demand side of shipboard power system [3,4]. In practice, however, due to the influence of uncertainty factors such as wind, temperature, waves and weather, the output power of PV system and ship load demand power are volatile, which makes it difficult to stabilize the power quality of the system. Therefore, it is essential to solve the influence of uncertainty on the power system.

At present, there are some studies to solve the impact of uncertainty on the power system. Lots of research on improving power system control focus on building more accurate non-linear models for power system [5]. In [6], a graphical user interface (GUI) is developed where the fault alarms appear on real-time status monitor whenever a fault occurs in the actual PV plant. Some other researchers are using intelligent algorithm as control approaches such as fuzzy logic [7], artificial neural networks [8,9], optimisation algorithms [10]. Due to the instability of renewable energy and the real-time change of load, the rapid and random power fluctuations bring more difficulties to signal estimation, which will significantly affect the power quality control of power system by these methods. Therefore, the method of predicting power fluctuation to provide accurate reference for ship power system controllers to eliminate uncertainty becomes significant.

1.2. Literature Review

In [11] a short-term nonparametric probabilistic method of PV powerprediction is presented. Based on long short-term memory (LSTM) recurrent neural network (RNN), Ref. [12] solves the short-term load power prediction problem for individual residential households. The back propagation neural network (BPNN) is regarded as a classical method for forecasting [13]. The accuracy of these prediction methods is high. However, it is difficult to predict the real-time fluctuation of power system due to the complex network structure. Therefore, this paper aims to design an algorithm which can predict the power fluctuation of integrated power system of ship, and lay a foundation for the control of power system.

Huang et al. proposed a machine learning algorithm for single-hidden layer feed forward neural networks, which is extreme learning machine (ELM) [14]. Due to the fast operation and applicabilities, ELM has been successfully used in many applications [15,16]. Many variants of ELM have been developed. Online sequential ELM (OS-ELM) [17] and incremental ELM (I-ELM) [18] are proposed for online sequential data. An ensemble-forecasting model based on the extreme learning machine algorithm (En-ELM) is designed in [19] to predict ultra-short-term power fluctuations, which serves as an extra signal for automatic generation control. However, the prediction algorithm is an offline prediction algorithm. Although it can avoid the randomness of ELM and improve the prediction accuracy and stability, the network structure is complex and always requires a lot of calculation. According to the support vector machine (SVM), an online sequential learning algorithm for regularized Extreme Learning Machine (OS-RELM) is presented in [20] and a regularized OS-ELM with adaptive regulation factor is proposed for time-varying nonlinear system in [21]. Based on hybrid decomposition and outlier robust based OS-ELM is developed for wind speed forecasting [22]. In [23], for online ship roll motion prediction problem, a sequential ELM is proposed using grey prediction. With the limitation of the scale for online prediction training, a forgetting mechanism is introduced into OS-ELM [24] and on the basis of [24], the researchers in [25] introduces the forgetting factor, which further increases the prediction accuracy of the algorithm. The above methods can ensure that the algorithm can realize online sequence prediction. However, it is difficult to accurately get the characteristics and trendency of online sequence data with the online prediction based on OS-ELM; the prediction error is large.

In order to reduce the prediction error, OS-ELM with kernel (KOS-ELM) [26] function is proposed and has some development. In [27], a KOS-ELM is proposed for non-stationary time series prediction. However, with the online prediction, the scale of kernel function in the algorithm is also expanding, which increases the computational complexity. Although the prediction accuracy is guaranteed, the longer prediction time can not meet the requirements of online prediction. Based on [28], an online sequential extreme learning machine with reduced kernel function (OS-RKELM) is developed, which can reduce the growth of kernel function in the process of online prediction [29]. However, the OS-RKELM can not be applied to online prediction of time series, because the processing of random data has disturbed the time correlation of data.

1.3. Main Work and Paper Organization

In order to improve the power prediction speed of the algorithm and provide enough prediction accuracy to predict online power fluctuations to achieve an accurate reference of the power system control, based on the above research, this paper proposes an adaptive KOS-ELM. The adaptive factors are introduced into KOS-ELM to ensure the prediction accuracy and speed up the prediction algorithm. By comparing the similarity and prediction error, the scale of kernel function in online prediction process is limited to speed up the operation, and the time correlation between data is kept to ensure the prediction accuracy. In order to verify the effectiveness of the proposed algorithm, this paper uses the power fluctuation data recorded by ships sailing in different sea conditions to test the algorithm. The prediction performance is compared with OS-ELM, KOS-ELM, En-ELM and LSTM.

The remaining of this paper is organised as follows: In Section 2, ELM, KELM, OS-ELM, KOS-ELM and the derivation process of the foundation are introduced. Section 3 presents the proposed AKOS-ELM algorithm. The introduced adaptive factors and the way of their calculation are also explained. The simulation of online sequence prediction based on AKOS-ELM is demonstrated in Section 4, and the performance of the algorithm is discussed. Conclusion and future work are finally derived in Section 5.

2. Brief Review of ELM Methods

In this section, an overview of the ELM, KELM and OS-ELM extension is presented. This serves to provide the necessary background for the development of the adaptive improvement in the next section.

2.1. The ELM

ELM is proposed as feedfoward neural networks with single hidden-layer [14]. In ELM, only the number of the hidden neurons ispredefined, while the configurations of the hidden neurons need not be fine-tuned, but randomly assigned. Given training samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)|\boldsymbol{x}_i \in \boldsymbol{R}^d, \boldsymbol{t}_i \in \boldsymbol{R}^m\}_{i=1}^N$, where N is the number of data, d is the dimension, m is the number of output nodes. The output function of ELM is given as

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i h(\boldsymbol{x}, \boldsymbol{\Omega}_i, b_i) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} \tag{1}$$

where $h(\cdot)$ is the activation function, L is the number of hidden nodes, $\boldsymbol{\Omega_i}$ is the input weights of additive nodes, b_i is the bias of additive nodes and $\boldsymbol{\beta} = [\beta_i, \ldots, \beta_L]^T$ is the output weights, which can be obtained through the Least Squares Estimate (LSE) method as given in Equation (2).

$$\begin{aligned} &min\ \|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\| \\ \boldsymbol{\beta} =& \boldsymbol{H}^{\dagger}\boldsymbol{T} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{T} \end{aligned} \tag{2}$$

where $\boldsymbol{T}$ is the target matrix and $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\boldsymbol{H}$.

The origin ELM can be solved as a constrained L2-regularized optimization problem [23].

$$min \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\|^2 \tag{3}$$

where $\boldsymbol{H}$ denotes the output matrix of hidden layers which maps the data from the d-dimensional input space to the L-dimensional hidden layer feature space.

Then the solution of $\boldsymbol{\beta}$ can be solved through the Karush–Kuhn–Tucker (KKT) condition and given as the following equations

$$\boldsymbol{\beta} = \boldsymbol{H}^T(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{T},\ (N \leq L) \quad or \quad \boldsymbol{\beta} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{T},\ (N \geq L) \tag{4}$$

2.2. The KELM

In [29], connections with SVM and other kernel methods become apparent when, instead of an explicit feature vector $\boldsymbol{h}(\boldsymbol{x})$, which need not be known, a kernel function (5) is considered by applying the Mercer's conditions on ELM [15,16].

$$\boldsymbol{\Omega} = \boldsymbol{H}\boldsymbol{H}^T \quad \Omega(i,j) = \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{h}(\boldsymbol{x}_j) = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5}$$

and the express in Equation (1) can be represented in kernel-based form

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_N) \end{bmatrix}^T (\boldsymbol{\Omega} + \frac{\boldsymbol{I}}{C})^{-1}\boldsymbol{T} \tag{6}$$

In [28], a RKELM method is presented using a reduced kernel matrix instead of the full kernel matrix to build the model. The SLFN with kernel function $\mathcal{K}(\cdot,\cdot)$ and L support vectors $\boldsymbol{X}_L = \{\boldsymbol{x}_i | \boldsymbol{x}_i \in \boldsymbol{R}^d\}_{i=1}^{L}$ can be written as

$$\begin{aligned} \boldsymbol{\Omega}_{N\times L}\boldsymbol{\beta} &= \boldsymbol{T} \\ \sum_{s=1}^{L} \boldsymbol{\beta}_s \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_s) &= \boldsymbol{t}_i, \quad i = 1, 2, \ldots, N \end{aligned} \tag{7}$$

where $\boldsymbol{\Omega}_{N\times L} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}_L)$ is the reduced kernel matrix, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L]$ is the output weight matrix.

2.3. The OS-ELM and KOS-ELM

The online sequential update version of the ELM, OS-ELM, has already been developed and is widely used [17]. The OS-ELM aims to sequentially update the inverse matrix $(\boldsymbol{H}^T\boldsymbol{H})^{-1}$ and the output weights $\boldsymbol{\beta}$ in Equation (2) using the Woodbury formula.

Suppose that the dataset is presented in successive chunks S_j, and given a chunk of initial training set $S_0 = \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}^{N_0}$, then the initial output weights $\boldsymbol{\beta}^{(0)}$ is computed

$$\boldsymbol{\beta}^{(0)} = (\boldsymbol{H}_0^T\boldsymbol{H}_0)^{-1}\boldsymbol{H}_0^T\boldsymbol{T}_0 = \boldsymbol{K}_0^{-1}\boldsymbol{H}_0^T\boldsymbol{T}_0 \tag{8}$$

then, suppose that another chunk of data $S_1 = \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}^{N_0+N_1}$ is given, where N_1 denotes the number of instances in this chunk, therefore, the minimizing problem becomes as

$$min \left\| \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \boldsymbol{T}_0 \\ \boldsymbol{T}_1 \end{bmatrix} \right\| \tag{9}$$

Considering both chunks of the data sets S_0 and S_1, the updated output weight $\boldsymbol{\beta}^{(1)}$ is obtained as

$$\boldsymbol{\beta}^{(1)} = \boldsymbol{K}_1^{-1} \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{T}_0 \\ \boldsymbol{T}_1 \end{bmatrix} \qquad \boldsymbol{K}_1 = \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix} \tag{10}$$

then, the $\boldsymbol{\beta}^{(1)}$ can be expressed as a function of $\boldsymbol{\beta}^{(0)}$.

$$\boldsymbol{K}_1 = \begin{bmatrix} \boldsymbol{H}_0^T & \boldsymbol{H}_1^T \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix} = \boldsymbol{K}_0 + \boldsymbol{H}_1^T\boldsymbol{H}_1 \tag{11}$$

and

$$\begin{aligned} \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{T}_0 \\ \boldsymbol{T}_1 \end{bmatrix} &= \boldsymbol{H}_0^T\boldsymbol{T}_0 + \boldsymbol{H}_1^T\boldsymbol{T}_1 = \boldsymbol{K}_0\boldsymbol{K}_0^{-1}\boldsymbol{H}_0^T\boldsymbol{T}_0 + \boldsymbol{H}_1^T\boldsymbol{T}_1 \\ &= \boldsymbol{K}_0\boldsymbol{\beta}^{(0)} + \boldsymbol{H}_1^T\boldsymbol{T}_1 = (\boldsymbol{K}_1 - \boldsymbol{H}_1^T\boldsymbol{H}_1)\boldsymbol{\beta}^{(0)} + \boldsymbol{H}_1^T\boldsymbol{T}_1 \\ &= \boldsymbol{K}_1\boldsymbol{\beta}^{(0)} - \boldsymbol{H}_1^T\boldsymbol{H}_1\boldsymbol{\beta}^{(0)} + \boldsymbol{H}_1^T\boldsymbol{T}_1 \end{aligned} \tag{12}$$

substituting Equations (11) and (12) into Equation (10), $\boldsymbol{\beta}^{(1)}$ is given by

$$\begin{aligned} \boldsymbol{\beta}^{(1)} &= \boldsymbol{K}_1^{-1} \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{H}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{T}_0 \\ \boldsymbol{T}_1 \end{bmatrix} = \boldsymbol{K}_1^{-1}(\boldsymbol{K}_1\boldsymbol{\beta}^{(0)} - \boldsymbol{H}_1^T\boldsymbol{H}_1\boldsymbol{\beta}^{(0)} + \boldsymbol{H}_1^T\boldsymbol{T}_1) \\ &= \boldsymbol{\beta}^{(0)} + \boldsymbol{K}_1^{-1}\boldsymbol{H}_1^T(\boldsymbol{T}_1 - \boldsymbol{H}_1\boldsymbol{\beta}^{(0)}) \end{aligned} \tag{13}$$

When $(k+1)$th chunk of data set arrives as $S_{k+1} = \{(\boldsymbol{x}_i, \boldsymbol{t}_i)^{\sum_{j=0}^{k+1} N_j}\}$, $k \geq 0$ and N_{k+1} denotes the number of the observations in the $(k+1)$th chunk. By generalizing the previous arguments, the $\boldsymbol{P}_{k+1}$ and $\boldsymbol{\beta}^{(k+1)}$ can be written as

$$\boldsymbol{K}_{k+1} = \boldsymbol{K}_k + \boldsymbol{H}_{k+1}^T\boldsymbol{H}_{k+1} \quad \boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \boldsymbol{K}_{k+1}^{-1}\boldsymbol{H}_{k+1}^T(\boldsymbol{T}_{k+1} - \boldsymbol{H}_{k+1}\boldsymbol{\beta}^{(k)}) \tag{14}$$

Using the Woodbury formula, the $\boldsymbol{K}_{k+1}^{-1}$ can be expressed as a function of $\boldsymbol{K}_{k}^{-1}$, let $\boldsymbol{P}_{k+1} = \boldsymbol{K}_{k+1}^{-1}$ then the Equation (14) for updating $\boldsymbol{\beta}^{(k+1)}$ can be rewritten as

$$\begin{aligned} \boldsymbol{P}_{k+1} &= \boldsymbol{P}_k - \boldsymbol{P}_k \boldsymbol{H}_{k+1}^T (\boldsymbol{I} + \boldsymbol{H}_{k+1} \boldsymbol{P}_k \boldsymbol{H}_{k+1}^T)^{-1} \boldsymbol{H}_{k+1} \boldsymbol{P}_k \\ \boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(0)} + \boldsymbol{P}_{k+1} \boldsymbol{H}_{k+1}^T (\boldsymbol{T}_{k+1} - \boldsymbol{H}_{k+1} \boldsymbol{\beta}^{(k)}) \end{aligned} \tag{15}$$

By substituting the kernel function expression, the recursive formula of $\boldsymbol{\beta}^{(k)}$ can be obtained as the following kernel-based form, hence the KOS-ELM algorithm [26].

$$\begin{aligned} \boldsymbol{\beta}^{(1)} &= \boldsymbol{Q}_1 \boldsymbol{T}_1 = (\frac{\boldsymbol{I}}{C} + \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1))^{-1} \boldsymbol{T}_1 \\ \boldsymbol{\beta}^{(k+1)} &= \begin{bmatrix} \boldsymbol{\beta}_k - \boldsymbol{z}_{k+1} \boldsymbol{r}_{k+1}^{-1} (\boldsymbol{T}_{k+1} - \boldsymbol{\Omega}_{k+1} \boldsymbol{\beta}_k) \\ \boldsymbol{r}_{k+1}^{-1} (\boldsymbol{T}_{k+1} - \boldsymbol{\Omega}_{k+1} \boldsymbol{\beta}_k) \end{bmatrix}, \quad k \geq 2 \end{aligned} \tag{16}$$

where

$$\begin{aligned} \boldsymbol{\Omega}_{k+1} &= \left[\mathcal{K}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_1), \ldots, \mathcal{K}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \right] \\ \boldsymbol{z}_{k+1} = \boldsymbol{Q}_k \boldsymbol{\Omega}_{k+1}, \quad \boldsymbol{r}_{k+1} &= \frac{\boldsymbol{I}}{C} + \mathcal{K}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_{k+1}) - \boldsymbol{z}_{k+1}^T \boldsymbol{\Omega}_{k+1} \\ \boldsymbol{Q}_{k+1} &= \boldsymbol{r}_k^{-1} \begin{bmatrix} \boldsymbol{Q}_k \boldsymbol{r}_{k+1} + \boldsymbol{z}_{k+1} \boldsymbol{z}_{k+1}^T & -\boldsymbol{z}_{k+1} \\ -\boldsymbol{z}_{k+1}^T & \boldsymbol{I} \end{bmatrix} \end{aligned} \tag{17}$$

In [26,27], the experimental study and analysis reflected that the KOS-ELM givesbetter testperformance than SVM/LS-SVM/PSVM and OS-ELM, however, with the new chunks of data received, the scale of kernel matrix size is increasing, which leads to the exponential growth of training time complexity during the whole online learning process. To cope with the online data well, in [29], a reduced KOS-ELM is proposed, which uses the method of randomly selecting a certain amount of data from the chunk to replace the original data set for training, and provides a reduced scale OS-ELM with kernel.

3. The Proposed AKOS-ELM Algorithm

During the online learning process, all samples are equally treated by the KOS-ELM and the OS-RKELM. However, for online time series, the ability to reflect the overall characteristics and trend is different between the new and old data. Considering the differences and timeliness of the samples, on the basis of the KOS-ELM, an adaptive KOS-ELM is proposed to maintain the size of the kernel matrix, ensure the prediction accuracy and calculation speed.

3.1. The AKOS-ELM

Learning the inital sample training chunk $S_0 = \{(\boldsymbol{x}_i, \boldsymbol{t}_i) \| \boldsymbol{x}_i \in \boldsymbol{R}^d, \boldsymbol{t}_i \in \boldsymbol{R}^m\}_{i=1}^{N_0}$, based on the Equation (7), the inital output weight matrix $\boldsymbol{\beta}^{(0)}$ can be obtained as

$$\boldsymbol{\beta}^{(0)} = \boldsymbol{Z}_0^{-1} \boldsymbol{\Omega}_0^T \boldsymbol{T}_0, \quad \boldsymbol{Z}_0 = \frac{\boldsymbol{I}}{C} + \boldsymbol{\Omega}_0^T \boldsymbol{\Omega}_0, \quad \boldsymbol{\Omega}_0 = \mathcal{K}(\boldsymbol{X}_0, \boldsymbol{X}_L) \tag{18}$$

When another chunk of data $S_1 = \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=N_0+1}^{N_0+N_1}$ arrives, where N_1 is the number of the data samples in the new chunk of arrival, the output matrix of hidden layer is obtained and the corresponding output weight $\boldsymbol{\beta}^{(1)}$ then becomes

$$\boldsymbol{\beta}^{(1)} = (\frac{\boldsymbol{I}}{C} + \begin{bmatrix} \boldsymbol{\Omega}_0 \\ \boldsymbol{\Omega}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Omega}_0 \\ \boldsymbol{\Omega}_1 \end{bmatrix})^{-1} \begin{bmatrix} \boldsymbol{\Omega}_0 \\ \boldsymbol{\Omega}_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{T}_0 \\ \boldsymbol{T}_1 \end{bmatrix}, \quad \boldsymbol{\Omega}_1 = \mathcal{K}(\boldsymbol{X}_1, \boldsymbol{X}_L) \tag{19}$$

Generally speaking, the new data can better describe the trend of the current data. When the old and new data are similar, or the contribution to reflect the characteristics of the data is similar, we can maintain the size of the kernel matrix and ensure the calculation speed by introducing an adaptive factor μ to adjust the weights of the new and old chunk of data. The validity period of any sample chunk is determined by memory factor n. Therefore, the $\boldsymbol{\beta}^{(1)}$ can be represented with the factor as

$$\begin{aligned}\boldsymbol{\beta}^{(1)} =&(\mu_0\frac{\boldsymbol{I}}{C}+\begin{bmatrix}\mu_0\boldsymbol{\Omega}_0\\ \boldsymbol{\Omega}_1\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\Omega}_0\\ \boldsymbol{\Omega}_1\end{bmatrix})^{-1}\begin{bmatrix}\mu_0\boldsymbol{\Omega}_0\\ \boldsymbol{\Omega}_1\end{bmatrix}^T\begin{bmatrix}\boldsymbol{T}_0\\ \boldsymbol{T}_1\end{bmatrix}\\ =&(\mu_0\frac{\boldsymbol{I}}{C}+\mu_0\boldsymbol{\Omega}_0^T\boldsymbol{\Omega}_0+\boldsymbol{\Omega}_1^T\boldsymbol{\Omega}_1)^{-1}(\mu_0\boldsymbol{\Omega}_0^T\boldsymbol{T}_0+\boldsymbol{\Omega}_1^T\boldsymbol{T}_1)\end{aligned} \tag{20}$$

Let $\boldsymbol{Z}_1=\mu_0\frac{\boldsymbol{I}}{C}+\mu_0\boldsymbol{\Omega}_0^T\boldsymbol{\Omega}_0+\boldsymbol{\Omega}_1^T\boldsymbol{\Omega}_1$, then Equation (19) can be transformed into

$$\boldsymbol{\beta}^{(1)}=\boldsymbol{Z}_1^{-1}(\mu_0\boldsymbol{\Omega}_0^T\boldsymbol{T}_0+\boldsymbol{\Omega}_1^T\boldsymbol{T}_1)=\boldsymbol{\beta}^{(0)}+\boldsymbol{Z}_1^{-1}\boldsymbol{H}_1^T(\boldsymbol{T}_1-\boldsymbol{H}_1\boldsymbol{\beta}^{(0)}) \tag{21}$$

In general, when the $(k+1)$th chunk of dataset $S_{k+1}=\{(\boldsymbol{x}_i,\boldsymbol{t}_i)\}_{i=(\sum_{j=0}^{k}N_j)+1}^{\sum_{j=0}^{k+1}N_j}$ arrives, where $k\geq 0$ and N_{k+1} denotes the number of the observations in the $(k+1)$th chunk. When the number of datasets does not reach the maximum memory factor $k\leq n$, $\boldsymbol{Z}_{k+1}$ and $\boldsymbol{\beta}^{(k+1)}$ can be written as

$$\begin{aligned}&\boldsymbol{Z}_{k+1}=\mu_k\boldsymbol{Z}_k+\boldsymbol{\Omega}_{k+1}^T\boldsymbol{\Omega}_{k+1},\quad \boldsymbol{\beta}^{(k+1)}=\boldsymbol{\beta}^{(k)}+\boldsymbol{Z}_{k+1}^{-1}\boldsymbol{\Omega}_{k+1}^T(\boldsymbol{T}_{k+1}-\boldsymbol{\Omega}_{k+1}\boldsymbol{\beta}^{(k)})\\ &\boldsymbol{\Omega}_{k+1}=\mathcal{K}(\boldsymbol{X}_{k+1},\boldsymbol{X}_L),\quad \boldsymbol{X}_{k+1}=\{\boldsymbol{x}_i\}_{i=(\sum_{j=0}^{k}N_j)+1}^{\sum_{j=0}^{k+1}N_j},\quad \boldsymbol{T}_{k+1}=\begin{bmatrix}\boldsymbol{t}_{\sum_{j=0}^{k}N_{j+1}}^T\\ \vdots\\ \boldsymbol{t}_{\sum_{j=0}^{k+1}N_j}^T\end{bmatrix}\end{aligned} \tag{22}$$

With the Wood-bury formula employed, the update formula of $\boldsymbol{Z}_{k+1}^{-1}$ is derived

$$\begin{aligned}\boldsymbol{Z}_{k+1}^{-1}=&(\mu_k\boldsymbol{Z}_k+\boldsymbol{\Omega}_{k+1}^T\boldsymbol{\Omega}_{k+1})^{-1}\\ =&(\mu_k\boldsymbol{\Omega}_k)^{-1}-(\mu_k\boldsymbol{\Omega}_k)^{-1}\boldsymbol{H}_{k+1}^T(\boldsymbol{I}+\boldsymbol{\Omega}_{k+1}(\mu_k\boldsymbol{\Omega}_k)^{-1}\boldsymbol{\Omega}_{k+1}^T)^{-1}\\ &\times\boldsymbol{\Omega}_{k+1}(\mu_k\boldsymbol{Z}_k)^{-1}\\ =&\frac{1}{\mu_k}\boldsymbol{Z}_k^{-1}-(\frac{1}{\mu_k})^2\boldsymbol{Z}_k^{-1}\boldsymbol{\Omega}_{k+1}^T(\boldsymbol{I}+\frac{1}{\mu_k}\boldsymbol{\Omega}_{k+1}\boldsymbol{Z}_k^{-1}\boldsymbol{\Omega}_{k+1}^T)^{-1}\boldsymbol{\Omega}_{k+1}\boldsymbol{Z}_k^{-1}\\ =&\frac{1}{\mu_k}(\boldsymbol{Z}_k^{-1}-\boldsymbol{Z}_k^{-1}\boldsymbol{\Omega}_{k+1}^T(\mu_k\boldsymbol{I}+\boldsymbol{\Omega}_{k+1}\boldsymbol{Z}_k^{-1}\boldsymbol{\Omega}_{k+1}^T)^{-1}\boldsymbol{\Omega}_{k+1}\boldsymbol{Z}_k^{-1})\end{aligned} \tag{23}$$

Let $\boldsymbol{G}_{k+1}=\boldsymbol{Z}_{k+1}^{-1}$, then the equations for updating $\boldsymbol{\beta}^{(k+1)}$ can be derived as

$$\begin{aligned}\boldsymbol{G}_{k+1}=&\frac{1}{\mu_k}(\boldsymbol{G}_k-\boldsymbol{G}_k\boldsymbol{\Omega}_{k+1}^T(\mu_k\boldsymbol{I}+\boldsymbol{\Omega}_{k+1}\boldsymbol{G}_k\boldsymbol{\Omega}_{k+1}^T)^{-1}\boldsymbol{\Omega}_{k+1}\boldsymbol{G}_k)\\ \boldsymbol{\beta}^{(k+1)}=&\boldsymbol{\beta}^{(k)}+\boldsymbol{G}_{k+1}\boldsymbol{\Omega}_{k+1}^T(\boldsymbol{T}_{k+1}-\boldsymbol{\Omega}_{k+1}\boldsymbol{\beta}^{(k)})\end{aligned} \tag{24}$$

When the data set size reaches the limitation $k\geq n$, the next arriving data set needs to be decided whether to keep the data set by the factor ε and discard the oldest data set.

Here, assume that S_k is acquired and the $(k-u)$th chunk of data S_{k-u} is discarded. Then, the expression in Equation (18) can be represented as

$$\begin{aligned} \boldsymbol{Z}_k =& \mu_{k-1}\boldsymbol{Z}_{k-1} + \boldsymbol{\Omega}_k^T\boldsymbol{\Omega}_k = \sum_{i=k-n+1}^{k-1} \mathcal{M}_i^{k-1}\boldsymbol{\Omega}_i^T\boldsymbol{\Omega}_i + \boldsymbol{\Omega}_k^T\boldsymbol{\Omega}_k \\ \boldsymbol{\beta}^{(k)} =& \boldsymbol{Z}_k^{-1}\begin{bmatrix} \mathcal{M}_{k-n+1}^{k-1}\boldsymbol{\Omega}_{k-n+1} \\ \vdots \\ \mu_{k_1}\boldsymbol{\Omega}_{k-1} \\ \boldsymbol{\Omega}_k \end{bmatrix}^T \begin{bmatrix} \mathcal{M}_{k-n+1}^{k-1}\boldsymbol{T}_{k-n+1} \\ \vdots \\ \mu_{k_1}\boldsymbol{T}_{k-1} \\ \boldsymbol{T}_k \end{bmatrix} \end{aligned} \tag{25}$$

where $\mathcal{M}_j^{k-1} = \prod_{i=j}^{k-1}\mu_i$, $j = k-n+1,\ldots,k-1$. At the time where the $(k+1)$th data chunk S_{k+1} is received, the output weight $\boldsymbol{\beta}^{(k+1)}$ is denoted as follows

$$\begin{aligned} \boldsymbol{Z}_{k+1} =& \mu_k\boldsymbol{Z}_k + \boldsymbol{\Omega}_{k+1}^T\boldsymbol{\Omega}_{k+1} - \mathcal{M}_{k-n+1}^k\boldsymbol{\Omega}_{k-n+1}^T\boldsymbol{\Omega}_{k-n+1} = \sum_{i=k-u+2}^{k} \mathcal{M}\boldsymbol{\Omega}_i^T\boldsymbol{\Omega}_i + \boldsymbol{\Omega}_{k+1}^T\boldsymbol{\Omega}_{k+1} \\ \boldsymbol{\beta}^{(k+1)} =& \boldsymbol{Z}_{k+1}^{-1}\begin{bmatrix} \mathcal{M}_{k-n+2}^k\boldsymbol{\Omega}_{k-n+2} \\ \vdots \\ \mu_k\boldsymbol{\Omega}_k \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}^T \begin{bmatrix} \mathcal{M}_{k-n+2}^k\boldsymbol{T}_{k-n+2} \\ \vdots \\ \mu_k\boldsymbol{T}_k \\ \boldsymbol{T}_{k+1} \end{bmatrix} \end{aligned} \tag{26}$$

and the Equation (24) can be rewritten as the following equations

$$\begin{aligned} \boldsymbol{\beta}^{(k+1)} =& \mu_k^2\boldsymbol{\beta}^{(k)} + \boldsymbol{G}_{k+1}\begin{bmatrix} \mathcal{M}_{k-n+1}^k\boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}^T \left(\begin{bmatrix} \mathcal{M}_{k-n+1}^k\boldsymbol{T}_{k-n+1} \\ \boldsymbol{T}_{k+1} \end{bmatrix} - \mu_k^2\begin{bmatrix} \boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}\boldsymbol{\beta}^{(k)}\right) \\ \boldsymbol{G}_{k+1} =& \frac{1}{\mu_k}\left(\boldsymbol{G}_k - \boldsymbol{G}_k\begin{bmatrix} -\mathcal{M}_{k-n+1}^k\boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}^T\right. \\ & \left.\times\left(\mu_k\boldsymbol{I} + \begin{bmatrix} \boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}\boldsymbol{G}_k\begin{bmatrix} -\mathcal{M}_{k-n+1}^k\boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}^T\right)^{-1}\begin{bmatrix} \boldsymbol{\Omega}_{k-n+1} \\ \boldsymbol{\Omega}_{k+1} \end{bmatrix}\boldsymbol{G}_k\right) \end{aligned} \tag{27}$$

3.2. The Adaptive Factors

In the previous part, two adaptive factors n, μ are used to improve the adaptability of the algorithm. n is a factor used to maintain the size of the online training dataset and the size of the kernel function for the proposed algorithm, and to adjust the weights of the old and new chunks in the online process, the adaptive factor μ is employed. In order to make sure the proposed algorithm has better effect, these two factors are required to have the ability of updating. The function of adaptive factor in the algorithm is shown in the Figure 1.

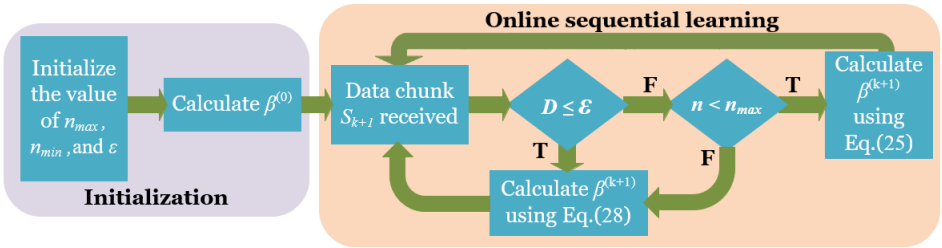

Figure 1. The diagram of the AKOS-ELM algorithm.

Here, two elements are introduced to make μ adaptive, λ and E. λ denotes the time weight in the current kernel function matrix, $(0 < \lambda < 1)$, the newer the data chunk,

the larger the weight. The other element, E, reflects the prediction errors of the training instances on the previous model. Then, the express of the factor μ can be written as an exponential function.

$$\mu_k = 1 - exp(-\frac{\lambda_k}{E_k}) \tag{28}$$

Since the value ranges of λ_k is from 0 to 1, and the $E > 0$, the $-\frac{\lambda_k}{E_k} < 0$, then, the right side of Equation (28) belongs 0 and 1, which makes sure the adaptive factor μ ranges from 0 to 1.

Similarly, the value of n is also related to two elements: the range of maximum and minimum scale of kernel function, which is presented as n_{max} and n_{min}. The value of the two elements can be obtained by the optimization function before the online prediction algorithm begins.

$$min \ \|\boldsymbol{Y}(n_{max}, n_{min}) - \boldsymbol{T}\| \tag{29}$$

where $\boldsymbol{Y}$ denotes the prediction result of the current training chunk.

To maintain the size of kernel function and update the data used for training in kernel function, an adaptive factor ε is employed to determine whether to retain or replace data sets by calculating the similarity between old and new data. Similarity based on the Euclidean distance can be calculated by

$$D_k = \sum_{i=1}^{N_k} \frac{1}{1 + \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|_2} \tag{30}$$

where $\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2$ represents the Eucliden distance between the kth sample chunk and the $k+1$th sample chunk.

When $D_k \leq \varepsilon$, the new chunk is considered to be similar to the current data. Because the new data can better reflect the current trend, the new sample chunk is retained and the oldest one is removed to maintain the current kernel size. When $D_k > \varepsilon$, it means that the new data chunk brings new data characteristics, so it can be retained directly until it reaches the maximum scale n_{max}, then, the oldest data chunk should be removed as the previous processing method. In general, ε can be given by the normalized empirical value; in this paper $\varepsilon = 0.5$.

4. Simulation Results and Discussion

4.1. Setup

The shipboard load electric power data used in this paper is obtained during the sea trials of 146 M Multi-Purpose Offshore Carrier, Table 1 gives some important information about this ship. The electricity system of the shipboard power system is 690 V, 60 Hz, which consists of four main generating units and a set of 690 V distribution board; the 690 V distribution board is set into two sections through the bus coupler switch. In the paper, the data used in this paper are normalized or proportional transformed, which can express the change trend of the original data. With uncertainty caused by various working condition demand, random waves and wind, as stated above, PV system output power is highly volatile and the ship load electric power (mainly propulsion load power) also changes randomly to maintain dynamic performance like ship speed and bus voltage. Therefore, the real measurement data of shipboard electric power is employed to test the prediction performance of the proposed algorithm. The number of training set is selected as 3000, and the testing set is 500. The experimental studies have been conducted using MATLAB 2016b, MathWorks, MA, USA on a PC with 3.20 GHz CPU, 16G Memory and Windows 10 (64 bit).

Table 1. Ship important parameters.

Parameter	Value	Unit
Length overall	146.00	m
Length in waterline	142.28	m
Breadth in waterline	14.00	m
Moulded depth	8.00	m
Each generator capcity	2218	KW
Thruster capacity	1000/2300	KW
Converter efficiency	0.75	-

To verify the performance of the AKOS-ELM on the time series problem, we tested the algorithm in different sea conditions. The effect of the algorithm is compared from two benchmarks: root mean squad error (RMSE) and algorithm execution time. In the experimental test, the input and output of regression problem need to be normalized or proportional transformed, and therefore the input and output in this paper are normalized to the range $[-1, 1]$. In order to further demonstrate the performance of the algorithm, this paper uses LSTM, En-ELM, OS-ELM and KOS-ELM prediction methods as a comparison group. The weights and biases of OS-ELM are randomly chosen from $[-1, 1]$. The hidden layer number of LSTM is 2, neurons and epochs are set as 50 and 120, respectively. The number of sub-ELM is set as 50 for En-ELM. The kernel function is selected as Gaussian kernel and the additive activation function is chosen as sigmoid function.

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = exp(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2), \quad g(\boldsymbol{x}) = \frac{1}{1 + exp(-\boldsymbol{a}\boldsymbol{x} + b)} \tag{31}$$

4.2. Result

Two benchmark test data sets have been studied here, as shown in Figure 2, each data set used for this section training and test is 3500 data samples. In order to show the power fluctuation directly, when comparing the test results with the real value, the power value is normalized in the range of $[-1, 1]$, and Level-3 and Level-5 represent different sea condition level.

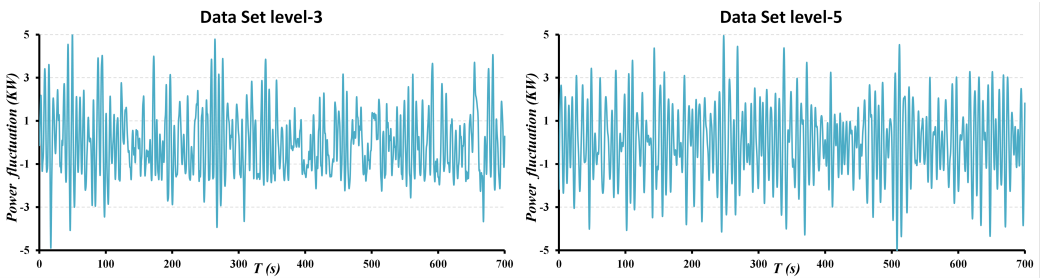

Figure 2. Two benchmark power fluctuation data sets.

As mentioned before, the first 3000 data of each data set shown in Figure 2 are used for initialization training, and the last 500 data are used to test the prediction ability of the algorithm.

4.2.1. Scenario 1

In Figures 3 and 4, using the learning mode of one by one, the comparison of the prediction results of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM and LSTM with the test data set are shown, respectively, and the comparison of the prediction errors of the three methods is given at last.

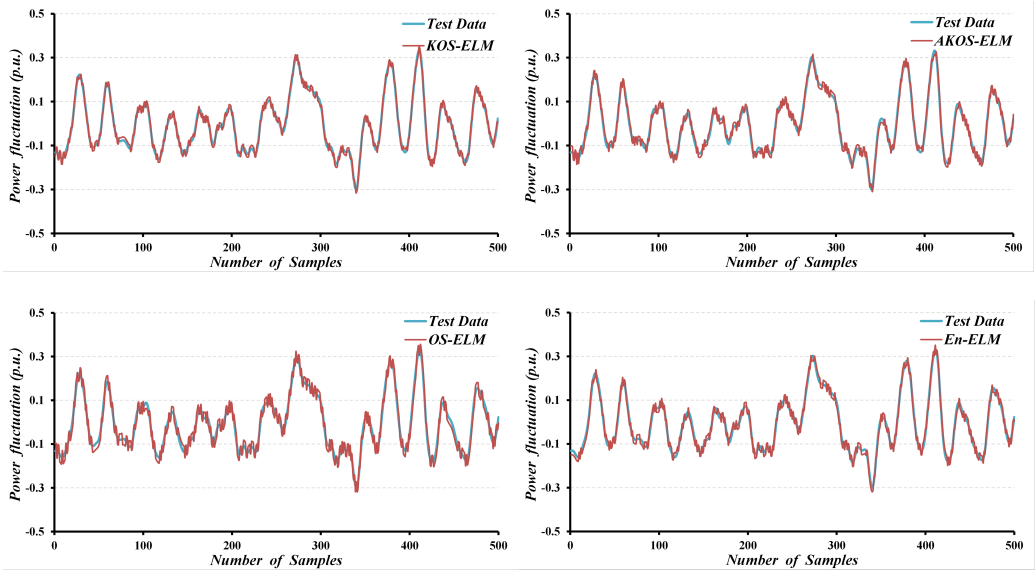

Figure 3. Prediction performance comparison of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM, LSTM using Level-3 data set (part a).

It can be seen from Figures 3 and 4 that KOS-ELM has the lowest prediction error, however, the deviation of prediction error between the KOS-ELM algorithm and AKOS-ELM is not big, which is only about 18% better than AKOS-ELM. This is because the kernel function of KOS-ELM contains the characteristics of all the data from the beginning to the end, so the prediction model obtained by this method is more able to reflect the characteristics and trend of real data. While in the online prediction process, AKOS-ELM removes part of the data to ensure the speed of the algorithm, so it can not get the same prediction accuracy as KOS-ELM. However, the KOS-ELM method is time-consuming, which is about 22 times more than AKOS-ELM as shown in Table 2, because in the online prediction process, the scale of kernel function is increasing, which leads to the increase of computational complexity.

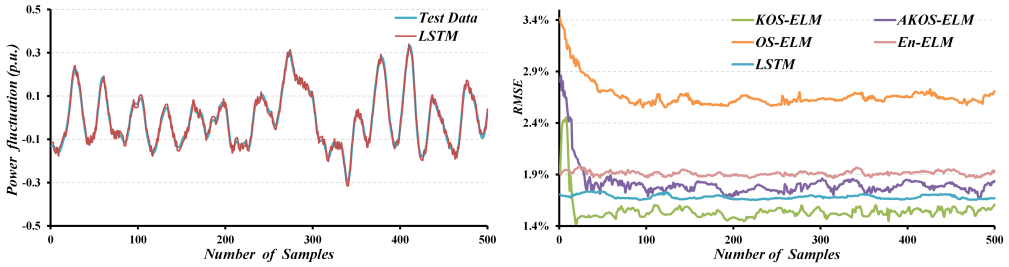

Figure 4. Prediction performance comparison of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM, LSTM using Level-3 data set (part b).

It can be seen that the algorithm time of OS-ELM and AKOS-ELM is close, which is about 40% of AKOS-ELM. This is because the AKOS-ELM algorithm has adaptive factors, which limit the scale of kernel function in the online prediction process. However, OS-ELM can not get the data features well. The prediction accuracy of AKOS-ELM is 30% higher than that of OS-ELM. AKOS-ELM has better performance than OS-ELM when meeting the execution time requirements.

When only the prediction accuracy is compared, the accuracy of LSTM is between KOS-ELM and AKOS-ELM, which is 18% better than AKOS-ELM, while the accuracy of En-ELM is not as good as AKOS-ELM. LSTM offline training network can not adapt to the changing sea conditions and the power change of ship flexible load in real time, so LSTM needs to continuously train network parameters. When there are too many hidden layers

in LSTM, the training time is too long. When the hidden layers are reduced, the prediction accuracy is low. The same is true of En-ELM. Too many sub-ELMs make the training time too long, and too few sub-ELMs will reduce the prediction accuracy. In this test, the average training time of LSTM is about 5 s and that of En-ELM is about 4 s.

Table 2. Performance comparison of KOS-ELM, AKOS-ELM, OS-ELM En-ELM and LSTM on Level-3 sea condition.

Algorithms	Execution Time (s)	RMSE (Average)		Nodes
		Training	Testing	
En-ELM	5.349	0.0191		120
LSTM	4.276	0.0168		120
KOS-ELM	19.317	0.0148	0.0155	120
AKOS-ELM	0.8837	0.0183	0.0188	120
OS-ELM	0.5136	0.0255	0.0261	120

4.2.2. Scenario 2

In order to further verify the performance of the algorithm on different data sets, we use Level-5 sea condition data set as shown in Figure 2. The following test results are obtained.

As can be seen from Figures 5 and 6 and Table 3 that the test results of the two data sets are qualitatively consistent. In this data set, because the sea condition is worse than before, the power fluctuation is larger and the characteristics and trend of the data set are more difficult to obtain, resulting in greater prediction error than before. Due to the increase of uncertainty in data sets, the kernel function of the proposed algorithm is kept at the maximum scale in the online prediction process, which makes the algorithm more time-consuming than before. However, the growth of uncertainty has little effect on KOS-ELM, because it does not control the scale of kernel function.

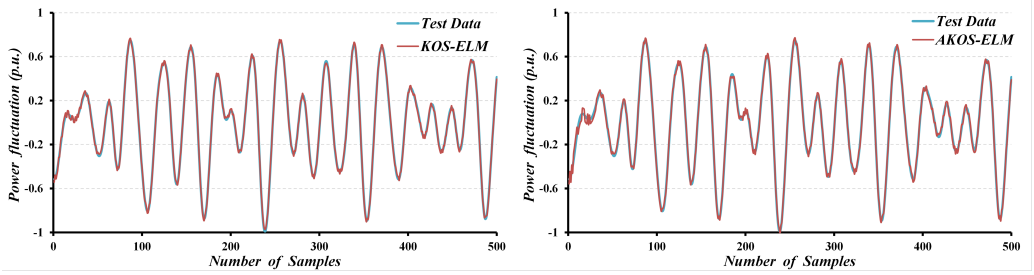

Figure 5. Prediction performance comparison of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM, LSTM using Level-5 data set (part a).

The deviation of prediction error between the KOS-ELM algorithm and AKOS-ELM is still not big, which is only about 17.7% better than AKOS-ELM. While the execution time of KOS-ELM is about 18 times more than AKOS-ELM as shown in Table 3, the OS-ELM has a lower execution time under this sea condition, which is about half of AKOS-ELM, however, the prediction accuracy of AKOS-ELM is 50% higher than that of OS-ELM. In this scenario, the average training time of LSTM is about 5.5 s and that of En-ELM is about 4.3 s. LSTM has better prediction accuracy which is 14% higher than AKOS-ELM, while AKOS-ELM is 17% better than En-ELM. It can be seen that despite the changes of sea conditions, the proposed algorithm still maintains a stable advantage compared with other prediction algorithms.

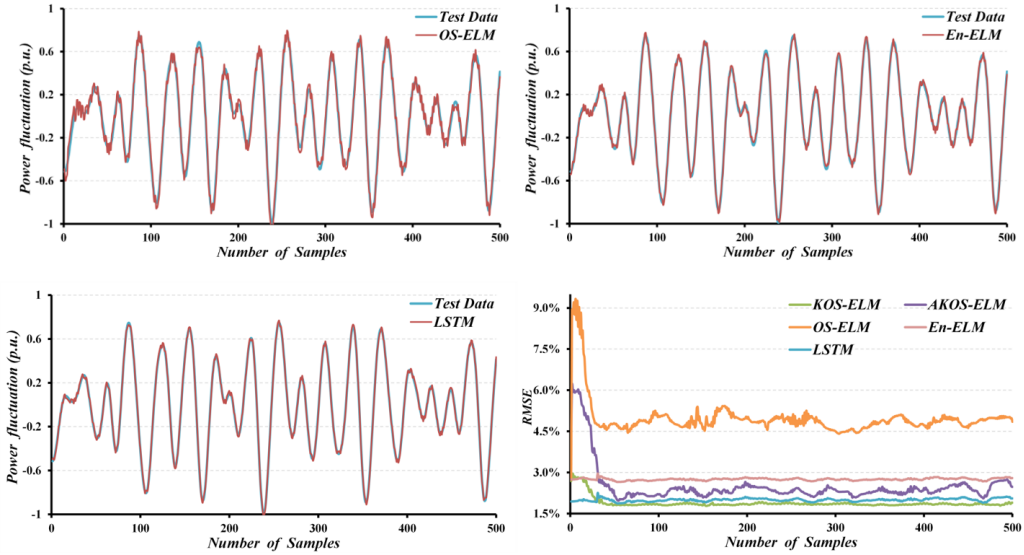

Figure 6. Prediction performance comparison of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM and LSTM using a Level-5 data set (part b).

Table 3. Performance comparison of KOS-ELM, AKOS-ELM, OS-ELM, En-ELM and LSTM on Level-5 sea condition.

Algorithms	Execution Time (s)	RMSE (Average)		Nodes
		Training	Testing	
En-ELM	5.613	0.0275		120
LSTM	4.592	0.0199		120
KOS-ELM	23.856	0.0188	0.0190	120
AKOS-ELM	1.3274	0.0227	0.0231	120
OS-ELM	0.5571	0.0483	0.0497	120

In summary, the comparison results of two data sets indicate that AKOS-ELM has less time-consuming without sacrificing much accuracy, the comprehensive performance is better than the other algorithms.

5. Conclusions and Future Work

With the deep penetration of renewable energy in ship power systems, it is of great significance to predict power fluctuation and provide accurate reference for ship power system controllers to eliminate uncertainty. In this paper, a fast and accurate online sequential kernel-based adaptive extreme learning machine, AKOS-ELM, is proposed for sequences with time-dependent characteristics. Three adaptive factors are introduced into the algorithm to ensure the rapidity and accuracy of the prediction algorithm. Multiple benchmarks are comprehensively tested using real ship power fluctuations under two sea conditions and compared with several prediction algorithms.

The test results show that compared with other online prediction algorithms, AKOS-ELM algorithm has better comprehensive effect, that is, it provides higher prediction accuracy while ensuring rapidity. The test results also show that compared with the traditional off-line training prediction algorithm, AKOS-ELM can still provide high prediction accuracy under the same number of epochs, and the rapidity of the proposed algorithm makes it meet the needs of real-time control system and can be more easily applied to power system control. Based on the above results, for the case of time-dependent series and online prediction demand, such as ship power fluctuation prediction, the algorithm can not only meet the rapidity demand of real-time control system, but also provide accurate prediction results.

Future work will study the combination of the proposed prediction algorithm with traditional power system controllers, such as generator controllers. The prediction algorithm can replace the traditional compensator and provide additional control signals for the power coordinated control of ship power system, so as to eliminate the problems caused by the power uncertainty of power system, such as frequent power fluctuation.

Author Contributions: Conceptualization, B.W.; Data curation, P.S.; Funding acquisition, L.Z.; Investigation, X.P.; Methodology, B.W.; Project administration, L.Z.; Resources, X.P. and P.S.; Software, B.W.; Validation, P.S.; Writing – original draft, B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Industry and Information Technology of People's Republic of China (25B0-02) and supported by the Fundamental Research Funds for the Central Universities (3072020CFT2403).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Relevant data can be obtained from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Skjong, E.; Volden, R.; Rødskar, E.; Molinas, M.; Johansen, T.A.; Cunningham, J. Past, present, and future challenges of the marine vessel's electrical power system. *IEEE Trans. Transp. Electrif.* **2016**, *2*, 522–537 [CrossRef]
2. Seenumani, G. Real-Time Power Management of Hybrid Power Systems in All Electric Ship Applications. Ph.D. Thesis, Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA, 2010.
3. Doerry, N.; McCoy, K. *Next Generation Integrated Power System: NGIPS Technology Development Roadmap*; Naval Sea Systems Command: Washington, DC, USA, 2007.
4. Prousalidis, J.M.; Tsekouras, G.J.; Kanellos, F.D. New challenges emerged from the development of more efficient electric energy generation units. In Proceedings of the 2011 IEEE Electric Ship Technologies Symposium, Alexandria, VA, USA, 10–13 April 2011; pp. 374–381.
5. Chen, X.; Lin, J.; Liu, F.; Song, Y. Stochastic assessment of AGC systems under non-Gaussian uncertainty. *IEEE Trans. Power Syst.* **2019**, *34*, 705–717. [CrossRef]
6. Iqbal, M.S.; Niazi, Y.A.K.; Khan, U.A.; Lee, B.W. Real-time fault detection system for large scale grid integrated solar photovoltaic power plants. *Int. J. Electr. Power Energy Syst.* **2021**, *130*, 106902. [CrossRef]
7. Mohanty, P.K.; Sahu, B.K.; Pati, T.K.; Panda, S.; Kar, S.K. Design and analysis of fuzzy PID controller with derivative filter for AGC in multi-area interconnected power system. *IET Gener. Transm. Distrib.* **2016**, *10*, 3764–3776. [CrossRef]
8. Qian, D.; Fan, G. Neural-network-based terminal sliding mode control for frequency stabilization of renewable power systems. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 706–717. [CrossRef]
9. Xu, D.; Liu, J.; Yan, X.G.; Yan, W. A novel adaptive neural network constrained control for a multi-area interconnected power system with hybrid energy storage. *IEEE Trans. Ind. Electron.* **2018**, *65*, 6625–6634. [CrossRef]
10. Liao, K.; Xu, Y. A robust load frequency control scheme for power systems based on second-order sliding mode and extended disturbance observer. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3076–3086. [CrossRef]
11. Golestaneh, F.; Pinson, P.; Gooi, H.B. Very short-term nonparametric probabilistic forecasting of renewable energy generation–with application to solar energy. *IEEE Trans. Power Syst.* **2016**, *31*, 3850–3863. [CrossRef]
12. Kong, W.; Dong, Z.Y.; Jia, Y.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Trans. Smart Grid* **2019**, *10*, 841–851. [CrossRef]
13. Jethmalani, C.R.; Simon, S.P.; Sundareswaran, K.; Nayak, P.S.R.; Padhy, N.P. Auxiliary hybrid PSO-BPNN-based transmission system loss estimation in generation scheduling. *IEEE Trans. Ind. Inf.* **2017**, *13*, 1692–1703. [CrossRef]
14. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
15. Huang, G.-B.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2012**, *42*, 513–529. [CrossRef]
16. Liu, X.; Lin, S.; Fang, J.; Xu, Z. Is extreme learning machine feasible? A theoretical assessment (Part I). *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 7–20. [CrossRef] [PubMed]
17. Liang, N.-Y.; Huang, G.-B.; Saratchandran, P.; Sundararajan, N. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Trans. Neural Netw.* **2006**, *17*, 1411–1423. [CrossRef] [PubMed]
18. Jin, B.; Jing, Z.; Zhao, H. Incremental and decremental extreme learning machine based on generalized inverse. *IEEE Access* **2017**, *5*, 20852–20865. [CrossRef]

19. Wen, S.; Wang, Y.; Tang, Y.; Xu, Y.; Li, P. Proactive Frequency Control based on Ultra-Short-Term Power Fluctuation Forecasting for High Renewables Penetrated Power Systems. *IET Renew. Power Gener.* **2019**, *13*, 2166–2173. [CrossRef]
20. Shao, Z.; Er, M.J. An online sequential learning algorithm for regularized Extreme Learning Machine. *Neurocomputing* **2016**, *173*, 778–788. [CrossRef]
21. Lu, X.; Zhou, C.; Huang, M.; Lv, W. Regularized online sequential extreme learning machine with adaptive regulation factor for time-varying nonlinear system. *Neurocomputing* **2016**, *174*, 617–626. [CrossRef]
22. Zhang, D.; Peng, X.; Pan, K.; Liu, Y. A novel wind speed forecasting based on hybrid decomposition and online sequential outlier robust extreme learning machine. *Energy Convers. Manag.* **2019**, *180*, 338–357. [CrossRef]
23. Yin, J.C.; Zou, Z.J.; Xu, F.; Wang, N.N. Online ship roll motion prediction based on grey sequential extreme learning machine. *Neurocomputing* **2014**, *129*, 168–174. [CrossRef]
24. Zhao, J.; Wang, Z.; Park, D.S. Online sequential extreme learning machine with forgetting mechanism. *Neurocomputing* **2012**, *87*, 79–89. [CrossRef]
25. Zou, Q.Y.; Wang, X.J.; Zhou, C.J.; Zhang, Q. The memory degradation based online sequential extreme learning machine. *Neurocomputing* **2018**, *275*, 2864–2879. [CrossRef]
26. Scardapane, S.; Comminiello, D.; Scarpiniti, M.; Uncini, A. Online sequential extreme learning machine with kernels. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2214–2220. [CrossRef]
27. Wang X.; Min, H. Online sequential extreme learning machine with kernels for nonstationary time series prediction. *Neurocomputing* **2014**, *145*, 90–97. [CrossRef]
28. Deng, W.Y.; Ong, Y.S.; Zheng, Q.H. A Fast Reduced Kernel Extreme Learning Machine. *Neural Netw.* **2016**, *76*, 29–38. [CrossRef] [PubMed]
29. Deng, W.-Y.; Ong, Y.-S.; Tan, P.S.; Zheng, Q.-H. Online sequential reduced kernel extreme learning machine. *Neurocomputing* **2016**, *174*, 72–84. [CrossRef]