Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 246964, 11 pages
http://dx.doi.org/10.1155/2014/246964

Research Article

Adaptive Control Using Fully Online Sequential-Extreme Learning Machine and a Case Study on Engine Air-Fuel Ratio Regulation

Pak Kin Wong,[1] Chi Man Vong,[2] Xiang Hui Gao,[1] and Ka In Wong[1]

[1] *Department of Electromechanical Engineering, University of Macau, Macau*
[2] *Department of Computer and Information Science, University of Macau, Macau*

Correspondence should be addressed to Chi Man Vong; cmvong@umac.mo

Received 21 January 2014; Accepted 6 March 2014; Published 7 April 2014

Academic Editor: Qingsong Xu

Most adaptive neural control schemes are based on stochastic gradient-descent backpropagation (SGBP), which suffers from local minima problem. Although the recently proposed regularized online sequential-extreme learning machine (ReOS-ELM) can overcome this issue, it requires a batch of representative initial training data to construct a base model before online learning. The initial data is usually difficult to collect in adaptive control applications. Therefore, this paper proposes an improved version of ReOS-ELM, entitled fully online sequential-extreme learning machine (FOS-ELM). While retaining the advantages of ReOS-ELM, FOS-ELM discards the initial training phase, and hence becomes suitable for adaptive control applications. To demonstrate its effectiveness, FOS-ELM was applied to the adaptive control of engine air-fuel ratio based on a simulated engine model. Besides, controller parameters were also analyzed, in which it is found that large hidden node number with small regularization parameter leads to the best performance. A comparison among FOS-ELM and SGBP was also conducted. The result indicates that FOS-ELM achieves better tracking and convergence performance than SGBP, since FOS-ELM tends to learn the unknown engine model globally whereas SGBP tends to “forget” what it has learnt. This implies that FOS-ELM is more preferable for adaptive control applications.

1. Introduction

Adaptive control is a powerful control scheme for dynamic system with high uncertainty. Its principle is to, based on the output feedback of the system, self-adjust the characteristics of the controller online in a way that the tracking error is reduced while stability is maintained. One remarkable development in adaptive control is the application of neural networks to the adaptive mechanism [1–3], which is often referred to as adaptive neural control. It is well known that neural networks can approximate any nonlinear relationship by means of different network parameters and activation functions. Therefore, by expressing the system uncertainty in terms of neural networks, an adaptive neural controller is able to handle arbitrary nonlinearities through the tuning of its unknown network parameters. With this attractive feature, adaptive neural control has been extensively used in many controller design problems and practical applications [4–8].

Nevertheless, in most typical neural controllers, the parameter adjustment method, or so-called the adaptive law, is based on the backpropagation (BP) algorithm [9]. The critical drawback of this algorithm is that it is a gradient-decent based learning method which may easily converge to local minima [10, 11]. Therefore, it usually takes “more than required” steps for the controller to achieve satisfactory performance. For instance, the simulation results in an earlier work of adaptive neural control [1] showed that thousands of updating steps were needed before the controller could finally achieve the desired convergence. Moreover, in some recent studies such as [5, 6] the neural controllers were shown to perform better than traditional proportional-integral-derivative controllers. However, the controllers still take many time steps to settle every time when the desired output is changed. These results indicate that the system dynamics cannot be globally approximated. Another disadvantageous property of BP is that it updates the parameters in all the layers of the neural network, leading to a long processing time and hence a slow convergence speed.

In order to address the issues of BP, Huang et al. [12, 13] proposed a simple and fast algorithm entitled extreme learning machine (ELM). This algorithm trains only a single hidden layer feedforward neural network. Unlike BP where all the parameters need to be tuned, ELM learns the unknown nonlinear relation by updating only the output weights (parameters between the hidden layer and the output layer of the neural network); the parameters in the hidden layer are randomly initialized and remain unchanged. Due to its simple structure and learning mechanism, ELM runs much faster (up to thousands of times [11–13]) than traditional BP. Meanwhile, ELM is also superior to BP in terms of generalization performance and accuracy, which has been verified in many latest works [11, 14–18]. In this sense, employing ELM into adaptive neural control should lead to a better control performance. Yet the original ELM algorithm is only suitable for batch learning. To learn the model online, online sequential ELM (OS-ELM) was proposed in [19]. While achieving the same performance as batch ELM, OS-ELM could update the network parameters sequentially no matter whether the data comes one by one or chunk by chunk. Therefore, by replacing BP with OS-ELM, a better and faster adaptive neural controller should be obtained.

However, there are some factors limiting the direct application of OS-ELM to adaptive control. Firstly, OS-ELM is not robust for noisy data. Secondly, the initial parameters of OS-ELM, which are randomly generated, can easily lead to singular and ill-posed problems [20]. These problems significantly affect the model, so that the generalization performance could degrade to an unacceptable level. Furthermore, theoretically speaking, OS-ELM is not a fully online sequential learning algorithm; it requires a chunk of representative initial data to train a base ELM model in advance to the online sequential learning. This chunk of representative initial data is usually difficult to obtain for adaptive control problems. The number of the initial data could not be less than the number of hidden nodes either. All these together highly restrict the use of OS-ELM.

In order to deal with the aforesaid problems, regularized OS-ELM (ReOS-ELM), which was proposed by Huynh and Won [20], could be used. In ReOS-ELM, the norm of output weights is added to the objective function to avoid singular and ill-posed problems. At the same time, a regularization parameter is included for the trade-off between the optimization of output weight norm and the training error. With the introduction of the regularization parameter, the number of training data could also be less than the number of hidden nodes. A base model, however, is still required in ReOS-ELM. To overcome this limitation, this paper proposes a fully online version of ELM, entitled fully online sequential-extreme learning machine (FOS-ELM). This proposed FOS-ELM is derived from ReOS-ELM, so it retains all the advantages and properties of ReOS-ELM, with the only difference that the batch training phase is discarded. Due to the removal of batch training phase, FOS-ELM can easily be applied to any adaptive control problems. For demonstration purpose, this paper presents the application of FOS-ELM to the adaptive engine air-fuel ratio (AFR) control based on a simulated engine model. The influence of the parameters (regularization parameter and hidden node number) is analyzed in the simulation. To verify the effectiveness of FOS-ELM, stochastic gradient-descent BP (SGBP), as a sequential learning variant of BP, is also applied to the same adaptive AFR control problem for comparison.

The organization of this paper is as follows. A brief review of ELM and its variants is provided in Section 2. The details of the proposed FOS-ELM are presented in Section 3. The application of FOS-ELM to adaptive engine AFR control and the related discussions are given in Section 4. Finally, conclusions are drawn in Section 5.

2. Review of ELM and Its Variants

This section briefly reviews the related work of ELM, including basic ELM, regularized ELM (ReELM), OS-ELM, and ReOS-ELM, in order to provide necessary background.

2.1. ELM and Regularized ELM. ELM [13] is an emerging technique for training feedforward neural networks without iterations. It consists of only one hidden layer, in which the input weights are randomly generated and need not be tuned. The output weights are optimized using a Moore-Penrose pseudoinverse instead of gradient-decent method. Apart from the number of hidden nodes, no other parameters have to be manually chosen [13, 19]. For a network with one hidden layer and L hidden nodes, the output function is

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

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is the output vector of the hidden layer feature mapping with respect to the input $\mathbf{x}$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^T$ is the vector of output weights between the hidden layer and the output nodes.

For a training dataset S with N samples, matrix $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \mathbf{h}(\mathbf{x}_2), \ldots, \mathbf{h}(\mathbf{x}_N)]^T$ can be used to present the hidden layer output. The size of $\mathbf{H}$ is $N \times L$ and each row of $\mathbf{H}$ is a training sample after feature mapping.

The goal of basic ELM is to minimize the training error; that is,

$$\text{Minimize: } \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2, \tag{2}$$

where $\mathbf{T}$ is the vector of real target t_i with respect to a sample x_i from S. Mathematically, it is a multiple linear regression problem. The solution of $\boldsymbol{\beta}$ to (2) is

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. If $\mathbf{H}^T\mathbf{H}$ is nonsingular, the orthogonal projection method can be used to calculate the pseudoinverse of $\mathbf{H}$

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T. \tag{4}$$

Thus, $\boldsymbol{\beta}$ can be rewritten as

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}. \tag{5}$$

Since basic ELM is based on empirical risk minimization principle (please refer to (2)), the trained model tends to be overfitting [20, 21]. Therefore, ReELM was proposed in [21] as an improved version of ELM. A similar work has also been introduced by the authors of ELM in [22], and a more detailed explanation can be found in [23]. The optimal goal of ReELM is to minimize not only the training error, but also the norm of the output weights; that is,

$$\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2, \quad \|\boldsymbol{\beta}\|^2. \tag{6}$$

The optimization problem of ReELM for a single-output node can then be formulated as follows:

$$\begin{aligned} &\text{Minimize: } \frac{1}{2}\lambda\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{N}\xi_i^2, \\ &\text{Subject to: } h(\mathbf{x}_i)\,\boldsymbol{\beta} = t_i - \xi_i, \quad i = 1, \ldots, N, \end{aligned} \tag{7}$$

where λ is the user-specified parameter that provides a trade-off between the training error and the norm of the output weights, N is the number of training data, and ξ_i is the error for ith training data (also known as slack variable). The solution of $\boldsymbol{\beta}$ can be calculated by

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{T}. \tag{8}$$

According to Bartlett's theory [24], this resulting solution tends to have better and more stable generalization performance, as verified in [21–23].

2.2. OS-ELM and ReOS-ELM. OS-ELM, originated from basic ELM, is an online sequential learning algorithm that can learn data not only one-by-one but also chunk-by-chunk with fixed or varying chunk size [19]. It consists of two phases: initialization phase and sequential learning phase. In the initialization phase, a base ELM model is trained using a small chunk of initial training data. For instance, the output weight for an initial training dataset S_0 with N_0 training samples is obtained as

$$\boldsymbol{\beta}_0 = \mathbf{P}_0\mathbf{H}_0\mathbf{T}_0, \tag{9}$$

$$\mathbf{P}_0 = (\mathbf{H}_0^T\mathbf{H}_0)^{-1}. \tag{10}$$

Then, in the sequential learning phase, when a new chunk of training data arrives, the output weights are updated by

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^T\left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}^{(k)}\right), \tag{11}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^T\left(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T\right)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k, \tag{12}$$

where $k + 1$ indicates the $(k + 1)$th arriving training data with k starting from zero and $\mathbf{H}_{k+1}$ is the hidden layer output for the $(k + 1)$th arriving training data.

One major problem in OS-ELM is that, if the term $\mathbf{H}_0^T\mathbf{H}_0$ is singular, then (10) is unsolvable. Therefore, to avoid the singular problem, OS-ELM restricts that the initial training dataset S_0 should have at least L (hidden node number) distinct samples. To improve this situation, ReOS-ELM [20] adds a regularization term to (10); that is,

$$\mathbf{P}_0 = (\mathbf{H}_0^T\mathbf{H}_0 + \lambda\mathbf{I})^{-1}. \tag{13}$$

According to the ridge regression theory, adding a small positive value into the diagonal of $\mathbf{H}_0^T\mathbf{H}_0$ can also avoid singular problem when the number of initial training data is less than the hidden nodes number. Therefore, ReOS-ELM can resolve the constraint suffered in OS-ELM, making it suitable for case when initial number of data is small (e.g., adaptive control problems). In addition, similar to ReELM, the term λ in (13) of ReOS-ELM mainly controls the relative importance between the training error and the norm of output weights. The theory behind the improvement of ReOS-ELM over OS-ELM can be explained using the same reason of ReELM over basic ELM.

3. Proposed FOS-ELM

In this section, an improved ReOS-ELM, namely, fully online sequential-extreme learning machine (FOS-ELM), is proposed. It does not need a small chunk of initial training data to construct a base model but can achieve the same performance with ReOS-ELM.

Considering an initial training dataset $S_0 = \{(x_i, t_i) \mid i = 1, \ldots, N_0\}$ with a corresponding hidden layer output matrix $\mathbf{H}_0$, using (9) and (13), the output weights $\boldsymbol{\beta}^0$ are calculated as

$$\begin{aligned} \boldsymbol{\beta}^0 &= (\mathbf{H}_0^T\mathbf{H}_0 + \lambda\mathbf{I})^{-1}\mathbf{H}_0^T\mathbf{T}_0 = \mathbf{K}_0^{-1}\mathbf{H}_0^T\mathbf{T}_0, \\ \mathbf{K}_0 &= \mathbf{H}_0^T\mathbf{H}_0 + \lambda\mathbf{I}, \end{aligned} \tag{14}$$

where $\mathbf{T}_0 = [t_1 \cdots t_{N_0}]^T$ and $\mathbf{K}_0^{-1} = \mathbf{P}_0$.

Suppose now a new training dataset $S_1 = \{(x_j, t_j) \mid j = N_0 + 1, \ldots, N_0 + N_1\}$ arrives with a corresponding hidden layer output matrix $\mathbf{H}_1$. By considering both training datasets S_0 and S_1, using (9) and (13) again, the output weights $\boldsymbol{\beta}^1$ should be obtained as

$$\begin{aligned} \boldsymbol{\beta}^1 &= \left(\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix} + \lambda\mathbf{I}\right)^{-1}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} \\ &= \mathbf{K}_1^{-1}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix}, \end{aligned} \tag{15}$$

$$\begin{aligned} \mathbf{K}_1 &= \begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix} + \lambda\mathbf{I} \\ &= \mathbf{H}_0^T\mathbf{H}_0 + \mathbf{H}_1^T\mathbf{H}_1 + \lambda\mathbf{I} = \mathbf{K}_0 + \mathbf{H}_1^T\mathbf{H}_1, \end{aligned} \tag{16}$$

where $\mathbf{T}_1 = [t_{N_0+1} \cdots t_{N_0+N_1}]^T$. Now expanding the last two terms on the right-hand side of (15)

$$\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T \begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} = \mathbf{H}_0^T\mathbf{T}_0 + \mathbf{H}_1^T\mathbf{T}_1 = \mathbf{K}_0\boldsymbol{\beta}^0 + \mathbf{H}_1^T\mathbf{T}_1. \quad (17)$$

Then, combining (15), (16), and (17), $\boldsymbol{\beta}^1$ is obtained as

$$\begin{aligned}\boldsymbol{\beta}^1 &= \mathbf{K}_1^{-1}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T \begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix}\\ &= \boldsymbol{\beta}^0 + \left(\mathbf{K}_0 + \mathbf{H}_1^T\mathbf{H}_1\right)^{-1}\mathbf{H}_1^T\left(\mathbf{T}_1 - \mathbf{H}_1\boldsymbol{\beta}^0\right).\end{aligned} \quad (18)$$

Now, considering only S_1, $\boldsymbol{\beta}^1$ can be obtained as

$$\boldsymbol{\beta}^1 = (\mathbf{H}_1^T\mathbf{H}_1 + \lambda\mathbf{I})^{-1}\mathbf{H}_1^T\mathbf{T}_1. \quad (19)$$

Comparing (18) and (19), it is obvious that (19) can be obtained from (18) if and only if $\boldsymbol{\beta}^0 = \mathbf{0}$ and $\mathbf{K}_0 = \lambda\mathbf{I}$. Therefore, by initializing $\boldsymbol{\beta}^0 = \mathbf{0}$ and $\mathbf{K}_0 = \lambda\mathbf{I}$, the initial training datasets S_0 can be omitted, while a model for S_1 can still be constructed. In other words, the batch training in the initialization phase of ReOS-ELM is automatically integrated in FOS-ELM. Thereby, FOS-ELM becomes a fully online sequential learning algorithm and still can achieve the same learning performance with ReOS-ELM.

To make it clear, the proposed FOS-ELM algorithm is rewritten as below.

Step 1. Assign random values for input weights, and set $\boldsymbol{\beta}^0 = \mathbf{0}$ and $\mathbf{P}_0 = (\lambda\mathbf{I})^{-1}$.

Step 2. For the $(k+1)$th arriving training data,

(a) calculate the hidden layer output matrix $\mathbf{H}_{k+1}$;

(b) update the output weights $\boldsymbol{\beta}^{(k+1)}$ using (11) and (12).

In short, FOS-ELM is a fully online sequential learning algorithm. It is simpler than ReOS-ELM and easier to implement. Compared with OS-ELM and ReOS-ELM, FOS-ELM is more suitable for learning problems in which the training data is difficult to collect in advance. To emphasize the advantages of FOS-ELM, a detailed comparison among OS-ELM, ReOS-ELM, and FOS-ELM is summarized in Table 1.

As declared in [19], the sequential learning algorithm (11) and (12) of OS-ELM is similar to recursive least-squares (RLS) algorithm, so that all the convergence results of RLS can be applied. It has to be noted that, in fact, ReOS-ELM and FOS-ELM also share the same sequential learning update algorithm with OS-ELM, so the convergence results of RLS can also be applied to all the three algorithms, OS-ELM, ReOS-ELM, and FOS-ELM. In other words, if the three algorithms are applied to the adaptive controller, the controller stability can be guaranteed.

4. Case Study on Adaptive Engine AFR Control

To demonstrate the usefulness of the proposed algorithm, FOS-ELM is applied to the adaptive control of engine AFR based on a simulated engine model. The effectiveness of FOS-ELM in this application is discussed and a comparison with SGBP is provided in this section.

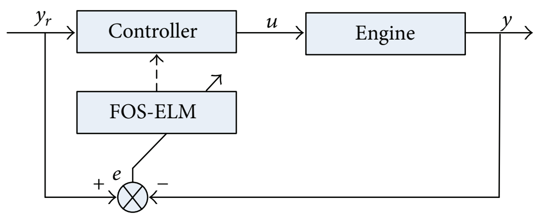

Figure 1: Adaptive engine AFR controller.

4.1. Adaptive Engine AFR Control. Engine AFR refers to the mass ratio of air to fuel present in the engine. It is a parameter that critically affects the engine emissions, brake-specific fuel consumption, and power [25]. In general, the AFR can be set to different values for different purpose. For example, using gasoline as the fuel, the AFR should be controlled to the stoichiometric AFR of gasoline, 14.7 : 1, in order to keep maximum conversion efficiency of the three-way catalytic converter [25]. In case higher engine torque is demanded, the AFR should be controlled to 12.5 : 1 in order to achieve the best engine power. For the best brake-specific fuel consumption, the AFR should be set to 16 : 1. Consequently, controlling the AFR is essential for maintaining the desired engine performance. However, the combustion process of an engine is a complex dynamic system that involves many uncertainties [11, 15]. Therefore, for illustrative purpose, this paper applies the adaptive control scheme, based on the proposed FOS-ELM, to the AFR control.

Theoretically, the dynamics of AFR can be described by a discrete approximated model in which the control appears linearly [1]:

$$\begin{aligned}y_{k+1} &= g\left(y_k, \ldots, y_{k-n+1}, u_{k-1}, \ldots, u_{k-n+1}\right)\\ &\quad + \varphi\left(y_k, \ldots, y_{k-n+1}, u_{k-1}, \ldots, u_{k-n+1}\right)u_k,\end{aligned} \quad (20)$$

where y is the AFR, u is the control input, k is the time step, n is the system order, and $\varphi(\cdot)$ must be a nonzero function. If both $g(\cdot)$ and $\varphi(\cdot)$ are known, the following control law can be used to exactly track the desired AFR, y_r:

$$u_k = \frac{y_{r_{k+1}}}{\varphi(\cdot)} - \frac{g(\cdot)}{\varphi(\cdot)}. \quad (21)$$

Therefore, assuming that FOS-ELM consists of two functions $\widehat{g}(\cdot)$ and $\widehat{\varphi}(\cdot)$, the purpose of FOS-ELM is to adaptively learn $g(\cdot)$ and $\varphi(\cdot)$ by self-tuning the parameters of $\widehat{g}(\cdot)$ and $\widehat{\varphi}(\cdot)$, based on the error from the system output feedback (i.e., $\widehat{g}(\cdot) \to g(\cdot)$ and $\widehat{\varphi}(\cdot) \to \varphi(\cdot)$). The engine AFR control scheme is illustrated in Figure 1.

The purpose of the controller is to control the amount of fuel injected to the engine so that the corresponding AFR can match the target AFR. The control signal u in (21) is the fuel injection time of the injectors. The longer the fuel injection time is, the larger the amount of the fuel injected is. To simplify the problem and focus on performance of FOS-ELM, a simulated engine model (465Q gasoline engine at

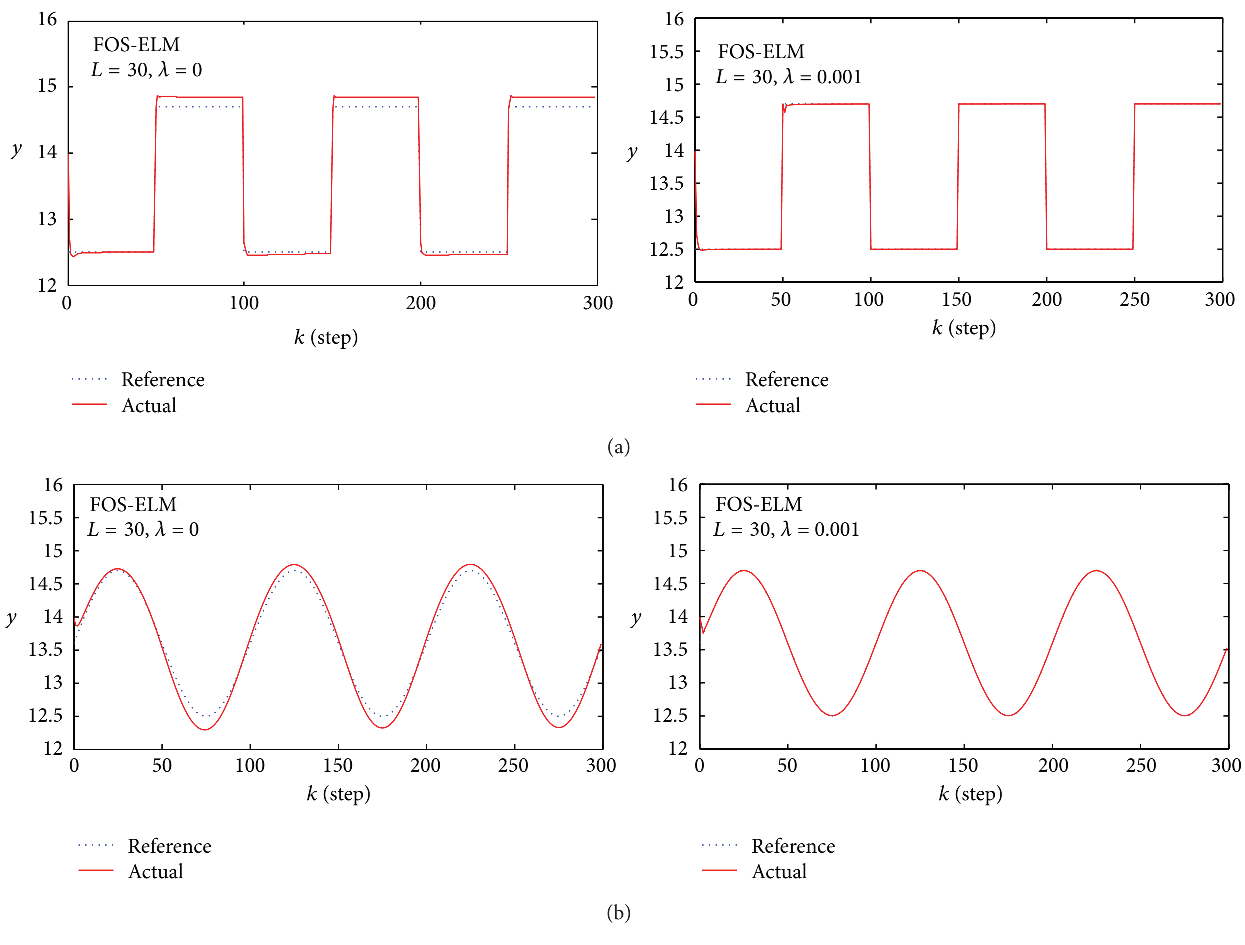

Figure 2: Effectiveness of λ for (a) square wave reference; (b) sine wave reference.

Table 1: Comparison among OS-ELM, ReOS-ELM, and FOS-ELM.

	OS-ELM	ReOS-ELM	FOS-ELM
Objective function	$\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$	$\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$ and $\|\boldsymbol{\beta}\|^2$	$\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$ and $\|\boldsymbol{\beta}\|^2$
Parameters	Hidden node number L	Hidden node number L Regularization parameter λ	Hidden node number L Regularization parameter λ
Training method	Initial offline batch training and followed by online sequential training	Initial offline batch training and followed by online sequential training	**Online sequential training**
Initial sample number N_0	$N_0 \geq L$	$1 \leq N_0 \leq L$	$\mathbf{N_0 = 0}$
Overfitting risk	Yes	No	No
Learning performance	Same as ELM	Same as ReELM	Same as ReELM

engine speed of 3500 rpm and manifold pressure of 85 kPa) [26] is used in this paper, given as

$$y_{k+1} = 0.2 \sin(y_k) + 3.5(9 - u_k). \tag{22}$$

Two reference AFR outputs (y_r) are used to evaluate the performance of FOS-ELM. The first one is a square wave, of which the amplitude changes between 12.5 and 14.7 every 50 steps. This can test the step response of the adaptive controller. The other reference command is a sine wave, of which the amplitude varies between 12.5 and 14.7 with a period of 100 steps. This, on the other hand, can test the continuous tracking performance of the controller. In addition, all the simulations in the following sections were implemented in MATLAB and executed on a PC with Intel Core i7 CPU and 4 GB RAM onboard.

4.2. Performance of FOS-ELM.

The performance of FOS-ELM on the adaptive AFR controller is evaluated by three cases. As compared to OS-ELM, there is a regularization parameter λ introduced in FOS-ELM. Therefore, the first case is to test the effectiveness of λ. Moreover, as

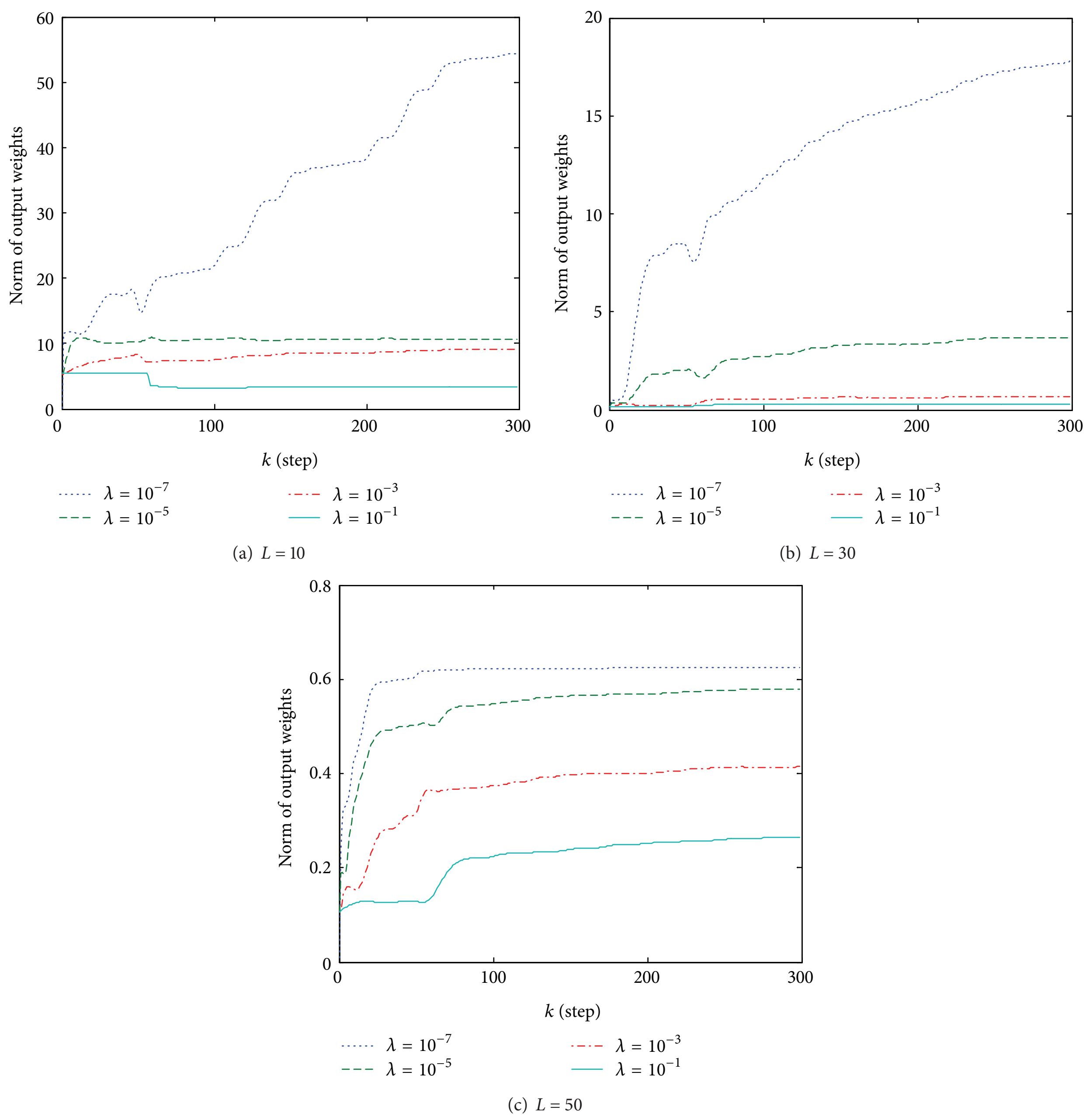

FIGURE 3: Norm of output weights with varying λ at (a) $L = 10$; (b) $L = 30$; (c) $L = 50$.

there is another important parameter in FOS-ELM, namely, the hidden nodes number L, the influence of these two parameters on the performance of FOS-ELM should also be analyzed. Thus, in the second case, simulations under various λ and L values are presented. In the last case, disturbances are introduced to the reference command in order to test the robustness of FOS-ELM.

4.2.1. Effectiveness of λ. In this case, the FOS-ELM used in the adaptive AFR controller was run under two λ values: $\lambda = 0$ and $\lambda = 0.001$. For both situations, L was set to 30. The simulation results are presented in Figure 2.

It can be seen that the tracking performance of FOS-ELM without using λ is quite poor as compared to that with λ. This is mainly due to the singular problem. It should be noted that when $\lambda = 0$, it becomes a special case of OS-ELM which trains the base model using the first arriving training sample. As in this adaptive problem, the training sample arrives in a one-by-one manner; the number of initial training data in the first step is also one. This is against the restriction of OS-ELM that the initial training dataset should have at least L distinct samples in the initial phase. Thus, the term $\mathbf{H}_0^T\mathbf{H}_0$ is mainly determined by the first arriving training sample and the singular $\mathbf{H}_0^T\mathbf{H}_0$ is inevitable.

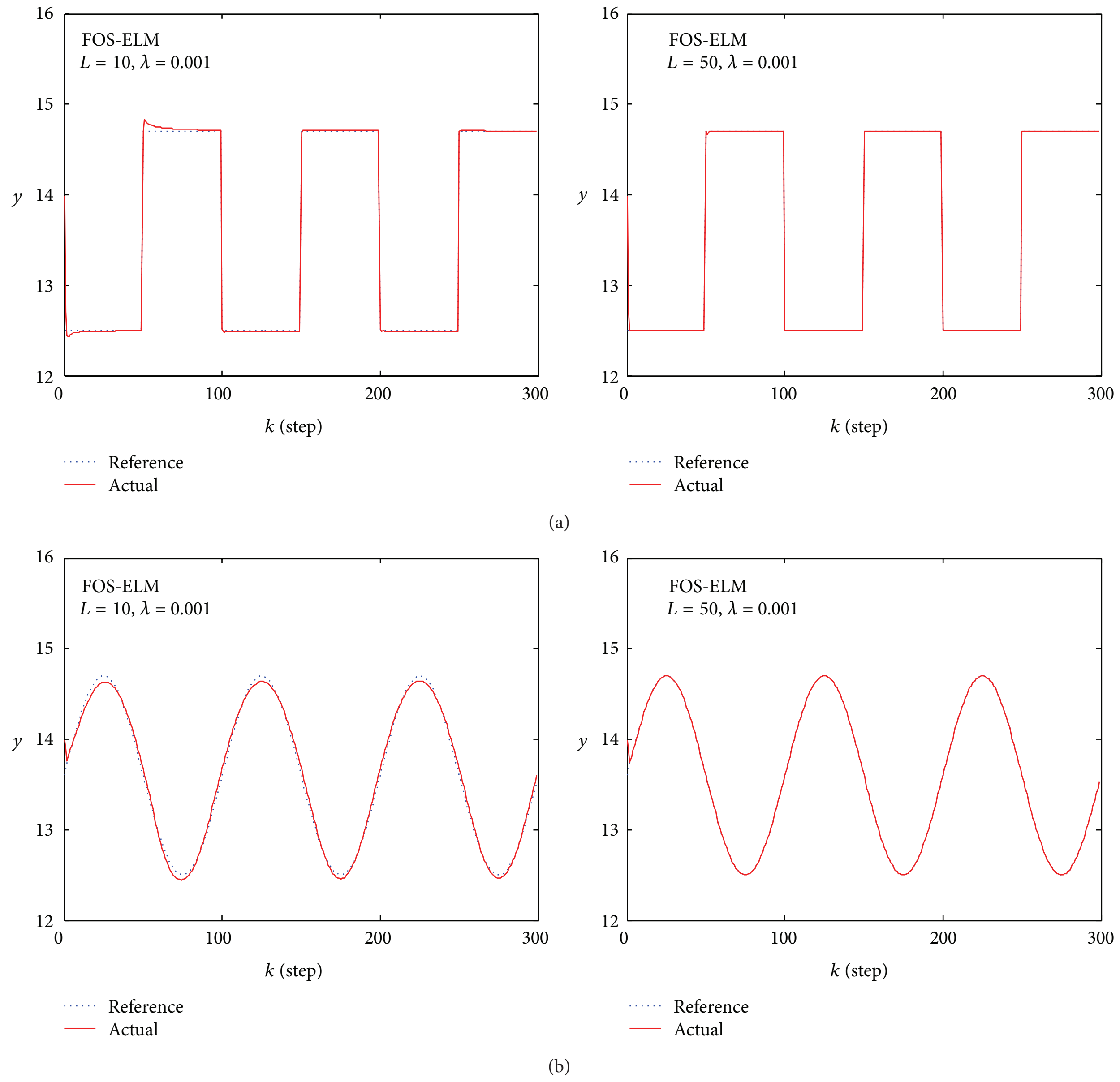

FIGURE 4: Influence of L for (a) square wave reference; (b) sine wave reference.

Furthermore, the rank of $\mathbf{P}_k$ (used in the update phase as given in (12)) basically remains at one in the sequential learning phase, so the learning performance of the FOS-ELM without λ is simply equal to an OS-ELM with only one hidden node; even a large number of hidden nodes are set. As a result, the tracking error of FOS-ELM without using λ cannot be eliminated.

4.2.2. Influence of λ and L. Referring to (7), the regularization parameter λ mainly controls the relative importance of the two terms: training error and norm of output weights. If λ is small, reducing the training error is more important. Otherwise, minimizing the norm of output weights is more important. Therefore, by setting λ to a small value, the tracking error will rapidly be reduced. However, if λ is too small (very close to 0), the matrix $(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})$ will tend to become singular, which leads to the situation suffered in the first case. Moreover, the hidden node number L is another factor that can affect the performance of FOS-ELM. It mainly controls the dimensionality of the model. Since, in all the variants of ELM, the parameters in the hidden node are randomly generated, it should follow that the larger the hidden node number is, the better the representation power is. As λ is already introduced in the FOS-ELM model, the overfitting problem due to large hidden node number is avoided. To analyze the influence of λ and L on the model, simulations were run at 4 different λ values (10^{-7}, 10^{-5}, 10^{-3}, and 10^{-1}) and 3 different L values (10, 30, and 50). The norm of the output weights of the FOS-ELM at each step was recorded. The results are shown in Figure 3.

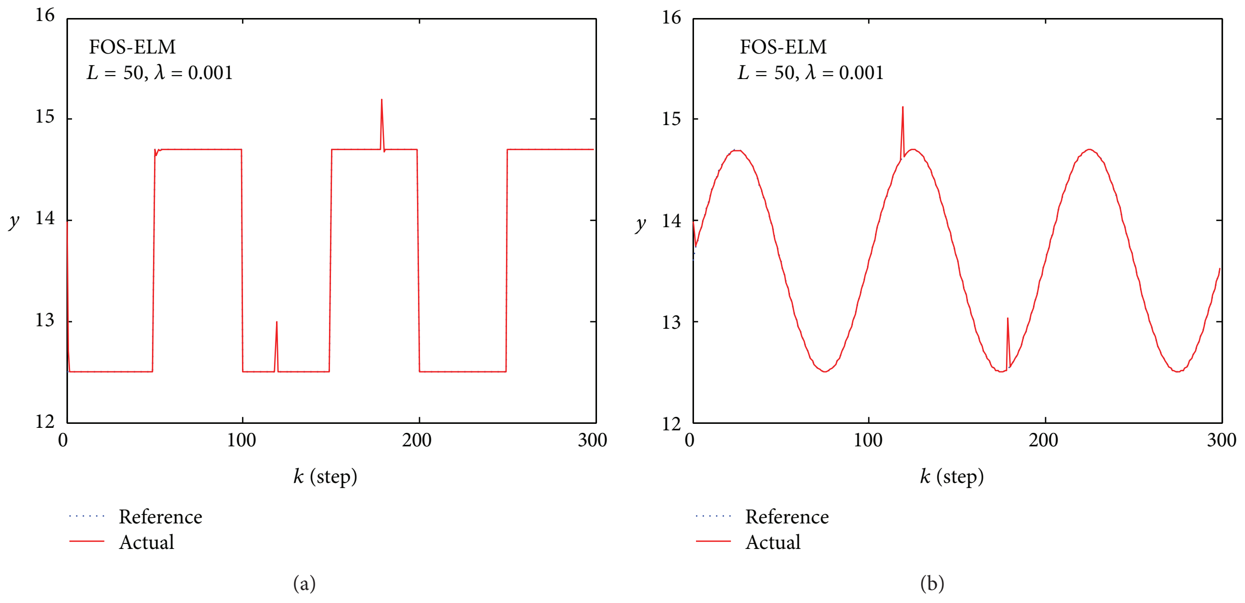

FIGURE 5: Effect of disturbance for (a) square wave reference; (b) sine wave reference.

The results in Figure 3 show that the influence of L is more significant than λ. As shown in Figure 3(a), when L is small, the norm of the output weights tends to "blow up," especially when λ is close to zero (e.g., $\lambda = 10^{-7}$). This "blow up" often leads to bad generalization performance [24]. On the contrary, as shown in Figure 3(c), if L is set to a large number, say 50, then different λ values do not affect the norm of output weights too much; the trend at different λ basically remains the same. This indicates that λ is not very sensitive to the norm of output weight, as long as it is not zero and L is sufficient. In order to investigate how the change of L affects the controller performance, simulations were run at two different L values: $L = 10$ and $L = 50$, with a fixed $\lambda = 0.001$. The results are shown in Figure 4.

From Figure 4, it can be seen that a FOS-ELM with a larger number of hidden nodes has better tracking performance. This verifies the idea that the hidden node number is in proportion to the representation power. As a remark, these simulation results are in accordance with the proof of ELM by Huang et al. [13].

4.2.3. Robustness of FOS-ELM. One powerful feature of adaptive controller is its robustness to disturbance. Therefore, to evaluate the robustness of FOS-ELM on this adaptive control application, two disturbances were introduced to the reference AFR command at $k = 120$ and $k = 180$. The simulations were run at $\lambda = 0.001$ and $L = 50$. The results are presented in Figure 5. It shows that although disturbances are introduced, the FOS-ELM adaptive controller can converge quickly back to the reference command, indicating that FOS-ELM is quite robust to disturbance.

4.3. Comparison to SGBP. SGBP, being a variant of BP for sequential learning application [9], is a typical algorithm for adaptive neural control. In order to show the benefits of FOS-ELM over SGBP on adaptive neural control, SGBP was also applied to the adaptive engine AFR control problem, and a comparison between FOS-ELM and SGBP was carried out. Similar to FOS-ELM, SGBP has two parameters, known as learning rate and the hidden node number. In the comparison, the hidden node number for SBGP is 20, and three learning rates ($\eta = 0.005, 0.01, 0.05$) were assigned. For FOS-ELM, the hidden node number is set to 50, with a regularization factor of 0.001 again. The results of the two methods are provided in Figure 6.

The comparative results from Figure 6 imply two things. The first one is that SGBP is quite sensitive to its learning rate. As shown in Figure 6(a), a small learning rate leads to a slow convergence speed, while, as shown in Figure 6(c), a large learning rate leads to an oscillation in the convergence process. It was found from some preliminary tests (not shown here) that the learning rate is also strongly associated with the hidden node number. In other words, to determine the structure and parameters of SGBP, expert experience is necessary [11]. Comparing to SGBP, FOS-ELM is less sensitive to parameters. Usually, the regularized parameter λ can be set to a small value like $\lambda = 0.001$, and the hidden node number L can be set to a large value like 50. This has already been verified in the previous section.

Another implication from Figure 6 is that FOS-ELM can achieve better global control performance as compared to SGBP. Referring to Figure 6(b), every time when the amplitude changes, several steps were required by SGBP to adapt to the desired reference. This shows that SGBP always tends to "forget" what it has learnt. The reason behind this phenomenon is that SGBP updates both input weights and output weights to achieve the desired output, which may easily suffer from local minima (i.e., optimal for the most recent arrived data). Thus, when the desired output changes, the weights need to be adjusted again for tracking the desired

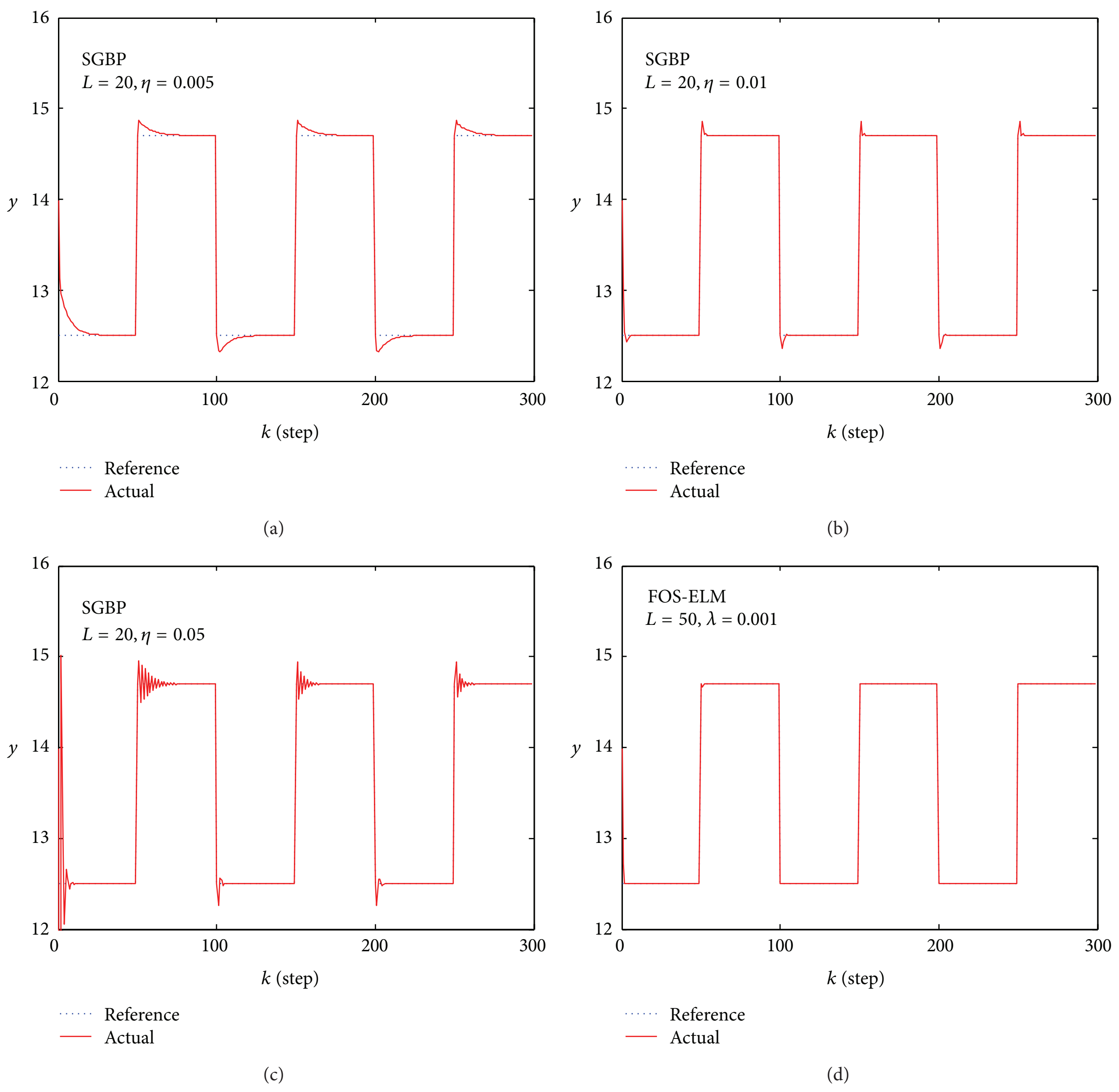

FIGURE 6: Controller performance for (a) SGBP with $L = 20$ and $\eta = 0.005$; (b) SGBP with $L = 20$ and $\eta = 0.01$; (c) SGBP with $L = 20$ and $\eta = 0.05$; (d) FOS-ELM with $L = 50$ and $\lambda = 0.001$.

output. In contrast, the theory behind FOS-ELM is to seek for a global optimal (i.e., optimal for all the seen data). Hence, as shown in Figure 6(d), once the model is learnt, the controller can directly adapt to the desired output even if it changes frequently. In fact, by referring to Figure 3, the norm of the output weights becomes stable after several learning steps, which also verifies that the model has been learnt and no further update is required. This unique feature of FOS-ELM is highly suitable for adaptive control applications.

5. Conclusions

In this paper, a novel fully online learning algorithm entitled FOS-ELM is proposed for adaptive neural control. It keeps the same learning performance with ReOS-ELM but discards the initial batch training phase adopted in ReOS-ELM. Without the initial training phase, FOS-ELM becomes easier to be implemented and more suitable for online learning task, of which the training data is difficult to be provided in advance, for example, adaptive control problems.

To demonstrate its effectiveness, the proposed FOS-ELM is applied to the adaptive control of engine AFR based on a simulated engine model. As the performance of FOS-ELM is determined by two important parameters, namely, regularization parameter and hidden node number, the influence of these parameters was analyzed. Furthermore, a comparison between FOS-ELM and SGBP on the adaptive control application was also carried out. The results imply the following.

(1) Without a regularization parameter, FOS-ELM becomes a special OS-ELM and leads to the singular problem, but, with a regularization parameter, both singular and overfitting problems are solved simultaneously, leading to better tracking performance.

(2) The norm of output weights is less sensitive to the regularization parameter but it is sensitive to the number of hidden nodes. It is found that a large number of hidden nodes with a small regularization parameter can result in a smaller norm of output weights and better tracking performance.

(3) SGBP always tends to "forget" what it has learnt, as its algorithm easily falls into a local optimal. Conversely, FOS-ELM aims to seek for a global optimal. Thus, once the model is learnt globally, the controller can directly adapt to the desired output even if the desired output changes frequently. This is the unique feature of FOS-ELM.

As FOS-ELM can achieve good tracking and convergence performance than traditional SGBP, it is more preferable for adaptive neural control applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research is supported by the University of Macau Research Grant, Grant nos. MYRG081(Y1-L2)-FST12-WPK, MYRG077(Y1-L2)-FST13-WPK, and MYRG075(Y1-L2)-FST13-VCM, and Macau FDCT Grant FDCT/075/2013/A.

References

[1] F. C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 44–48, 1990.

[2] C. Y. Lee and J. J. Lee, "Adaptive Control for Uncertain Nonlinear Systems Based on Multiple Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 34, no. 1, pp. 325–333, 2004.

[3] Y. J. Liu, C. L. P. Chen, G. X. Wen, and S. Tong, "Adaptive neural output feedback tracking control for a class of uncertain discrete-time nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1162–1167, 2011.

[4] C. C. Tsai, H. C. Huang, and S. C. Lin, "Adaptive neural network control of a self-balancing two-wheeled scooter," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 4, pp. 1420–1428, 2010.

[5] Y. Xiaofang, W. Yaonan, S. Wei, and W. Lianghong, "RBF networks-based adaptive inverse model control system for electronic throttle," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 750–756, 2010.

[6] J. Z. Peng and R. Dubay, "Identification and adaptive neural network control of a DC motor system with dead-zone characteristics," *ISA Transactions*, vol. 50, no. 4, pp. 588–598, 2011.

[7] L. S. Guo and L. Parsa, "Model reference adaptive control of five-phase IPM motors based on neural network," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 3, pp. 1500–1508, 2012.

[8] G. Q. Xia, X. C. Shao, A. Zhao, and H. Y. Wu, "Adaptive neural network control with backstepping for surface ships with input dead-zone," *Mathematical Problems in Engineering*, vol. 2013, Article ID 530162, 9 pages, 2013.

[9] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, G. B. Orr and K. R. Müller, Eds., pp. 9–50, Springer, Berlin, Germany, 1998.

[10] H. J. Rong and G. S. Zhao, "Direct adaptive neural control of nonlinear systems with extreme learning machine," *Neural Computing and Applications*, vol. 22, pp. 577–586, 2013.

[11] K. I. Wong, P. K. Wong, C. S. Cheung, and C. M. Vong, "Modelling of diesel engine performance using advanced machine learning methods under scarce and exponential data set," in *Applied Soft Computing*, vol. 13, pp. 4428–4441, 2013.

[12] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 985–990, Budapest, Hungary, July 2004.

[13] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[14] C. J. Lu and Y. J. E. Shao, "Forecasting computer products sales by integrating ensemble empirical mode decomposition and extreme learning machine," *Mathematical Problems in Engineering*, vol. 2012, Article ID 831201, 15 pages, 2012.

[15] K. I. Wong, P. K. Wong, C. S. Cheung, and C. M. Vong, "Modeling and optimization of biodiesel engine performance using advanced machine learning methods," *Energy*, vol. 55, pp. 519–528, 2013.

[16] P. K. Wong, C. M. Vong, C. S. Cheung, and K. I. Wong, "Diesel engine modelling using extreme learning machine under scarce and exponential data sets," *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, vol. 21, supplement 2, pp. 87–98, 2013.

[17] Y. Lan, Z. J. Hu, Y. C. Soh, and G. B. Huang, "An extreme learning machine approach for speaker recognition," *Neural Computing and Applications*, vol. 22, pp. 417–425, 2013.

[18] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum class variance extreme learning machine for human action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 1968–1979, 2013.

[19] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[20] H. T. Huynh and Y. Won, "Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1930–1935, 2011.

[21] W. Deng, Q. Zheng, and C. Lin, "Regularized extreme learning machine," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pp. 389–395, Nashville, Tenn, USA, April 2009.

[22] G. B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.

[23] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[24] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[25] P. K. Wong, H. C. Wong, and C. M. Vong, "Online time-sequence incremental and decremental least squares support vector machines for engine air-ratio prediction," *International Journal of Engine Research*, vol. 13, no. 1, pp. 28–40, 2012.

[26] G. Y. Li, *Application of Intelligent Control and MATLAB to Electronically Controlled Engines*, Publishing House of Electronics Industry, Beijing, China, 1 edition, 2007 (Chinese).