Elsevier Editorial System(tm) for Neurocomputing
Manuscript Draft

Manuscript Number:

Title: Adaptive Structure Radial Basis Function Network Model for Processes with operating region migration

Article Type: Full Length Article (FLA)

Keywords: RBF networks, neural network model, adaptive structure network, ROLS algorithm, RBF structure adaptation.

Corresponding Author: Prof. Dingli Yu,

Corresponding Author's Institution: Liverpool John Moores University

First Author: Siong D Tok, MSc

Order of Authors: Siong D Tok, MSc; Dingli Yu; Christian Matthews, PhD; Dong-Ya Zhao, PhD; quanmin Zhu, PhD

Abstract: An adaptive structure radial basis function (RBF) network model is proposed in this paper to model nonlinear processes with operating region migration. The recursive orthogonal least squares algorithm is adopted to select new centers on-line, as well as to train the network weights. Based on the R matrix in the orthogonal decomposition, an initial center bank is formed and updated in each sample period. A new learning strategy is proposed to gain information from the new data for network structure adaptation. A center grouping algorithm is also developed to divide the centers into active and non-active groups, so that a structure with a smaller size is maintained in the final network model. The proposed RBF model is evaluated and compared to the two fixed-structure RBF networks by modeling a nonlinear time-varying numerical example. The results demonstrate that the proposed adaptive structure model is capable of adapting its structure to fit the operating region of the process effectively with a more compact structure and it significantly outperforms the two fixed structure RBF models.

Adaptive Structure Radial Basis Function Network Model for Processes with operating region migration

D. K. Siong Tok*, Ding-Li Yu*[$], Christian Mathews*, Dong-Ya Zhao[+] and Quan-Min Zhu[#]

* Process Control Group, Liverpool John Moores University, Liverpool, U.K.

[+] Department of Chemical Industrial Equipment and Control Engineering, College of Chemical Engineering, China University of Petroleum, Qingdao, China.

Deprtment of Engineering, University of West of England, Bristol, U.K.

[$] Corresponding author: D.Yu@ljmu.ac.uk

Abstract. An adaptive structure radial basis function (RBF) network model is proposed in this paper to model nonlinear processes with operating region migration. The recursive orthogonal least squares algorithm is adopted to select new centers on-line, as well as to train the network weights. Based on the R matrix in the orthogonal decomposition, an initial center bank is formed and updated in each sample period. A new learning strategy is proposed to gain information from the new data for network structure adaptation. A center grouping algorithm is also developed to divide the centers into active and non-active groups, so that a structure with a smaller size is maintained in the final network model. The proposed RBF model is evaluated and compared to the two fixed-structure RBF networks by modeling a nonlinear time-varying numerical example. The results demonstrate that the proposed adaptive structure model is capable of adapting its structure to fit the operating region of the process effectively with a more compact structure and it significantly outperforms the two fixed structure RBF models.

Keywords: RBF networks, neural network model, adaptive structure network, ROLS algorithm, RBF structure adaptation.

I. INTRODUCTION

The radial basis function network (RBFN) has been successfully applied as a nonlinear function estimator for dynamical system modelling due to its simple architecture and online training ability [1, 2]. The RBFN's structures can be classified into two categories: fixed-

structure and adaptive structure. For a fixed-structure RBFN (FS-RBFN), the number and location of centers are fixed during the modelling and operation process and the model parameters (weights) may be adapted. While, an adaptive structure RBFN has the number and location of its hidden layer neurons adapted to better fit the dynamics of the process to be modeled, in addition to the adaptation of the network parameters. In general, it produces a comparatively satisfactory performance. Thus, the performance of an RBFN is heavily dependent on its structure and it is imperative to optimize the RBFN's structure to achieve a satisfactory performance, especially in modeling a highly time-varying process. In order to achieve a satisfactory network performance, a sufficient number of centers is required and there is no prior knowledge to find the exact number of centers that needed [3]. Thus, an unnecessary large RBFN is usually used, which causes numerical ill-conditioning in the training of the network and the worsen generalization of the trained model [4].

In the past decades, the adaptation of RBFN's structures has been intensively investigated. First of all, Platt [5] made a great contribution to the dynamic RBFN's structure by introducing an algorithm called resource allocating network (RAN). For an RAN, the hidden units are gradually inserted into the hidden layer based on the novelty of new data. In a latter attempt, Karayiannis and Min [6] developed a framework for growing RBFNs which merged supervised and unsupervised learning with network growth techniques. They proposed that the structure of network could be gradually constructed by splitting and increasing the prototypes which represented the network centers. However, the insignificant hidden neurons in [5, 6] were not pruned which led to a final network with a huge structure. To solve the oversized problem, Lu et al. [7, 8] proposed a sequential learning scheme for function approximation using a minimal RBFN which was referred to as minimal RAN (M-RAN). Their pruning strategy was to prune the hidden units that had insignificant contributions to the network performance. However, the optimal network structure achieved in [7, 8] is only for a certain data sets, while the performance would be degraded if it is used to predict future behavior in other regions. In recent years, a few methods have been proposed for self-organizing RBFNs [9, 10]. Although it was claimed that these methods [9, 10] outperformed M-RAN [7] and GGAP-RBF [11], the convergence of their algorithms needed to be investigated carefully for successful applications, which complicates the entire training algorithms. Moreover, there are many unknown parameters in [9, 10] which needs preliminary runs to find optimal values for the parameters before the adaptation of network take places.

Orthogonal decomposition is a numerically stable method for solving the least squares problems. Chen and Billings [12, 13] proposed a forward regression learning approach based on the batch orthogonal least squares (OLS) algorithm to determine an RBFN's structure. In their approach, the OLS algorithm was employed to determine an appropriate set of centers from a large set of candidate centers. The center was chosen, one by one, until an adequate RBFN's structure was achieved. Chen and Grant [4] further extended this method [12, 13] to train a multi-input multi-output (MIMO) RBFN. In addition, Chng et al. [14] extended the work of Chen and Billings [12, 13] by introducing a local adaptation process for an RBFN's structure. In the work of Chng et al. [14], the subset models with higher accuracy were achieved compared to [12, 13]. The advantage in [12-14] is that the structure and parameters of the RBFN are decided simultaneously by evaluating the contributions of centers to network performance. However, one major drawback is that the optimization of network's weights is of off-line training mode as their methods [12-14] are based on batch OLS algorithm, which means that no new data can be considered during the training process. For online application in training the weights, Yu at al. [15] showed that ROLS training algorithm was capable of maintaining the same accuracy of the RBFN model as the off-line training while requiring less computation. Gomm and Yu [3] developed a forward and a backward center selection algorithms using ROLS training algorithm. For the backward selection algorithm, the structure of network is simplified by removing the centers which had smallest contribution to the network performance. On the other hand, for the forward selection algorithm the technique is to build a network by adding centers which will maximally enhance the network performance. Their method [3] resulted in an acceptable level of efficiency and accuracy with a smaller network's size. The use of the backward center selection method was extended in [16] to develop an adaptive RBFN model. However, the developed RBFN models in [3, 16] were not 'fully' adaptive as the centers can only be selected from a pre-specified candidate center set. In their further work, Yu and Yu [17] proposed an adaptive algorithm that incorporated the pruning strategy in [3] to 'fully' adapt an RBFN model using the ROLS training algorithm. The adding and pruning of centers was based on the error index between the desired and measured modeling performances. New data was added as new center if the desired modeling performance was not achieved. Results showed that a compact RBFN was achieved while the desired modeling performance was maintained. However, in this method the added new centers did not play a role immediately as the performance was degraded for a few sample periods before the positive role is observed during the migration of the process's operating point.

This paper proposes a new algorithm for the adaptation of an RBFN structure for modelling process with operating point migration using ROLS training algorithm. The advantage of this proposed algorithm is that the RBFN is able to be adapted effectively to fit the new dynamics in the new operating region of the process with a compact structure while achieving a satisfactory performance. In this developed algorithm, the RBFN's structure, the number and location of centers, and parameter (weight) are adapted based on the novelty of new data. An initial center bank with a pre-specified number of centers is formed which involves the actions of adding, pruning and grouping of centers. In adding new centers, a new strategy is designed to spread more significant centers in the current operating regions to maximize the network performance. The pruning method in [17] is extended to prune insignificant centers from the center bank. Then, the centers in the center bank are divided into two groups – active center and redundant center groups. Active centers are used to predict the process output, while redundant centers are preserved for next sample time. When the process operating point migrates largely, the original centers will not be effective to act for output prediction and the new centers in the region where the operating point moves to will be added. The developed algorithm is evaluated using a nonlinear operating point-migrating numerical example. The effective ness of the developed algorithm is proved by comparing it with two fixed structure models. The paper is organized as follows. Section II explains the ROLS training algorithm. The adaptation algorithm is presented in Section III which includes the adding, pruning and grouping of centers. The evaluation of the developed ADS-RBFN and comparison studies is demonstrated in Section IV.

II. ROLS TRAINING ALGORITHM OF AN RBFN

A standard RBFN, as shown in Fig.1, has three layers: the input layer, hidden layer and the output layer. The hidden layer consists of hidden neurons and each hidden neuron has a vector called its center. In Fig. 1, $[x_1, \dots, x_m]$ and $\left[\hat{Y}_1, \dots, \hat{Y}_p\right]$ are the input and output vectors with their entries being network m inputs and p outputs, respectively.

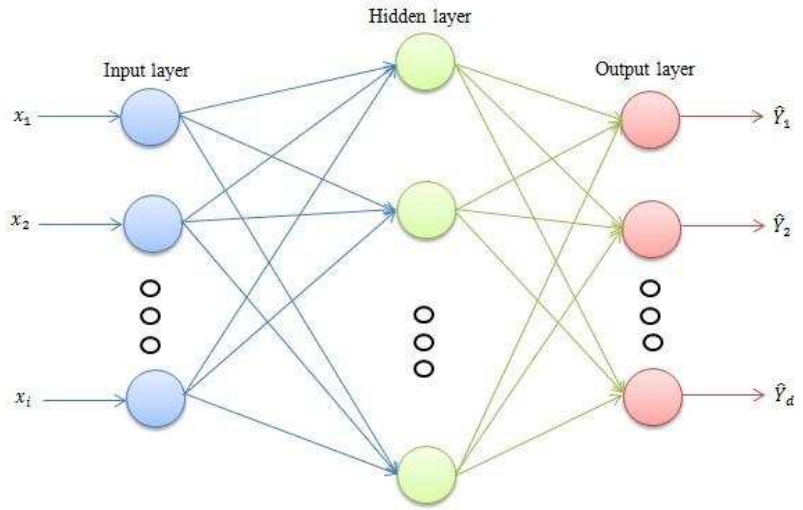

Fig. 1 The structure of an RBFN

A non-linear dynamic systems is presented by an NARX model in (1).

$$y(k) = f[(y(k-1), \dots, y(k-n_y), u(k), \dots, u(k-n_u)] + e(k) \quad (1)$$

where $u \in \Re^m$ and $y \in \Re^p$ are system input and output, and n_u and n_y are input and output orders, respectively. $e \in \Re^p$ is measurement noise. An RBFN is used as an approximate for the nonlinear function in (1), where the RBFN performs a nonlinear static mapping via the linear output transformation [3]. The input vector x of the RBFN includes all variables in function f(*) in (1), while the network output is $\hat{y}$. Here, the Gaussian function is used in the RBFN as the nonlinear basis function in (1).

$$\phi_i(k) = \exp\left(-\frac{\|x(k) - c_i\|^2}{\sigma_i^2}\right), i = 1, \dots, n_h \quad (2)$$

where $\phi(k)$ is the hidden layer output, n_h is the number of hidden layer nodes (center); $x(k)$ is the network input vector and c_i is the ith center with $i = 1, \dots, n_h$. The network output is the weighted sum of the hidden layer output and is given by,

$$y(k) = W\phi$$

where $W \in \Re^{n_h \times p}$ is the weighting matrix connecting the hidden layer nodes and network output;

The multi-output ROLS training algorithm developed in [3] is used here. By considering a set of N input-output training data,

$$Y = \hat{Y} + E = \Phi W + E \quad (3)$$

where $Y \in \Re^{N\times p}$ is the desired output matrix of the system to be modelled; $\hat{Y} \in \Re^{N\times p}$ is the output matrix of neural network.; $\Phi \in \Re^{N\times n_h}$ is the hidden layer output matrix and $E \in \Re^{N\times p}$ is the error modeling matrix.

$$Y^T = [y(1), \dots, y(N)], \qquad \hat{Y}^T = [\hat{y}(1), \dots, \hat{y}(N)],$$

$$\Phi^T = [\phi(1), \dots, \phi(N)], \qquad E^T = [e(1), \dots, e(N)],$$

The least squares problem to solve W becomes

$$J(W) = \|E\|_F = \|Y - \Phi W\|_F \tag{4}$$

where $\|*\|_F$ is the F-norm of a matrix defined as $\|A\|_F^2 = trace(A^T A)$.

With orthogonal transformation, (4) becomes

$$J(W) = \left\| \begin{bmatrix} \hat{Y} \\ \tilde{Y} \end{bmatrix} - \begin{bmatrix} R \\ 0 \end{bmatrix} W \right\|_F = \left\| \begin{bmatrix} \hat{Y} - RW \\ \tilde{Y} \end{bmatrix} \right\|_F \tag{5}$$

where $\hat{Y}$ is an $n_h \times p$ matrix and $\tilde{Y}$ is an $(N - n_h) \times p$ matrix.

From (5), the optimal W can be solved from backward substitution,

$$RW = \hat{Y} \tag{6}$$

and leaves $\left\|\tilde{Y}\right\|_F$ as the residual. This is the batch algorithm.

For recursive ROLS training algorithm, the cost function becomes

$$J(k) = \|E(k)\|_F = \left\| \begin{bmatrix} Y(k-1) \\ y^T(k) \end{bmatrix} - \begin{bmatrix} \Phi(k-1) \\ \phi^T(k) \end{bmatrix} W(k) \right\|_F. \tag{7}$$

Applying QR decomposition to ϕ(k-1) in (7), and multiply the inverse of Q(k-1) to Y(k-1), we have

$$\Phi(k-1) = Q(k-1) \begin{bmatrix} R(k-1) \\ 0 \end{bmatrix}$$

$$Q^T(k-1)Y(k-1) = \begin{bmatrix} \hat{Y}(k-1) \\ \tilde{Y}(k-1) \end{bmatrix}$$

$$J(k) = \left\| \begin{bmatrix} \hat{Y}(k-1) \\ y^T(k) \\ \tilde{Y}(k-1) \end{bmatrix} - \begin{bmatrix} R(k-1) \\ \phi^T(k) \\ 0 \end{bmatrix} W(k) \right\|_F. \tag{8}$$

With the arrival of new data, the update is described as follows,

$$\begin{bmatrix} R(k-1) \\ \dots \\ \phi^T(k) \end{bmatrix} = Q_1 \begin{bmatrix} R(k) \\ \dots \\ 0 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \hat{Y}(k) \\ \dots \\ \tilde{y}^T(k) \end{bmatrix} = Q_1^T(k) \begin{bmatrix} \hat{Y}(k-1) \\ \dots \\ y^T(k) \end{bmatrix} \quad (10)$$

The final cost function is

$$J(k) = \left\| \begin{bmatrix} \hat{Y}(k) - R(k)W(k) \\ \tilde{y}^T(k) \\ \tilde{Y}(k-1) \end{bmatrix} \right\|_F . \quad (11)$$

The optimal weight $W(k)$ is then solved as,

$$R(k)W(k) = \hat{Y}(k) \quad (12)$$

and leaves the residual as

$$y\left\|\tilde{Y}(k)\right\|_F^2 = \left\| \begin{bmatrix} \tilde{y}(k) \\ \tilde{Y}(k-1) \end{bmatrix} \right\|_F^2 = \|\tilde{y}^T(k)\|_F^2 + \left\|\tilde{Y}(k-1)\right\|_F^2 \quad (13)$$

The procedure of the ROLS training algorithm is therefore as follows.

1) Set the initial value for R, $\hat{Y}$ and $\left\|\tilde{Y}(k)\right\|_F^2$ as below,
   a. $R(0) = \alpha I$ where α is a small positive value.
   b. $\hat{Y}(0)$ and $\left\|\tilde{Y}(0)\right\|_F^2 = 0$.
2) At iteration k, with the arrival of new data $y^T(k)$, compute $\phi(k)$. Then, calculate $R(k)$ and $\hat{Y}(k)$ using (9) and (10), respectively.

III. ADS-RBFN ADAPTATION

Model structure adaptation for RBF network is this work is mainly achieved by updating the number and locations of the centers according to the current operating region. More centers will enable the network to have more accurate mapping but result in a big network size, whilst fewer centers will reduce the mapping accuracy but result in a smaller network, which consequently enhance the model generalization and reduce computing load.

The adaptation of the ADS-RBFN is implemented by evaluating the contribution of each center to the model prediction performance, and then according to the contribution to decide which center will be added or pruned. Also, the location of the added center needs to be determined to reflect migration of the system operating point. Firstly, an initial center bank with a pre-specified number of centers is formed by arbitrarily selecting some input data points as initial centers. Secondly, at each sample time, the network learns the information of

the center with the most contribution and the information of the new data. Then, determine the location of the added center according to the information.

The third step is to prune a center, which has the least contribution among the centers in the center bank at each sample time. This is to maintain the size of the center bank, which also maintains the computational demand that have been increased from the addition of new centers. The last step is that, after updating the center bank with the added and pruned centers, the centers are classified into two groups, active center group and redundant center group, based on their contributions to the network performance. The aim of the strategy to group these centers is to achieve a compact optimal network structure without degrading the network performance. Active centers will have bigger weight in contribution to the network output compared to redundant centers. Active centers are used for network prediction, while redundant centers are preserved for the later selection at the next sample time.

A. Add New Centers

For the structure of the RBFN, adding a new center means adding a hidden neuron. A new strategy of adding new centers is designed based on the information combining the center giving the most contribution to the network performance and the new data. At sample time k, the matrix $R(k-1)$ is updated with new data $x(k)$ using ROLS training algorithm. From the updated matrix $R(k)$ that contains the information of new data, the contribution of each center to the network performance is evaluated. Consider the evaluation index for contribution of each center proposed in [3],

$$\left\|\hat{Y}\right\|_F = \sum_{i=1}^{n_h} \left\|\hat{y}_i \hat{y}_i^T\right\|_F \qquad (14)$$

where $\hat{y}_i^T$ is the ith row of $\hat{Y}$. This shows that ith center has a separable contribution of $\left\|\hat{y}_i \hat{y}_i^T\right\|_F$ to $\left\|\hat{Y}\right\|_F$. Thus, the center with the most contribution C_{mc} can be found by computing $\left\|\hat{y}_i \hat{y}_i^T\right\|_F$ for each center and then compare them. The location of the added center should consider both the center with C_{mc} and the new data $x(k)$. The former represents the location for more effective center, while the latter represents the current operating region of the process. Ideally the best location for the added center should be found by the line search along the connection line of the most effective center and the new data, which is the optimal location in terms of maximal contribution to the prediction of current system output. In this research, the location of the new center is determined by the equation in (15) with a proper ε,

$$C_{new} = \varepsilon C_{mc} + (1 - \varepsilon)x(k) \quad (15)$$

where $0 < \varepsilon < 1$ is a parameter to be selected using the trial and error method for specific process. Smaller ε tends to use the current effective center location, while the bigger ε tends to move the new center to the new operating region. A compromise between the two can generate a smoother move to the new operating region which will benefit the future predictions. After adding a new center, new matrix R_{new} with previous N samples is retrained using [17],

$$\begin{bmatrix} \phi(k-N+1) \\ \phi(k-N+2) \\ \vdots \\ \phi(k) \end{bmatrix} = Q_{new} \begin{bmatrix} R_{new} \\ 0 \end{bmatrix} \quad (16)$$

$$\hat{Y}_{new} = Q_{new}^{T} \begin{bmatrix} y(k-N+1) \\ y(k-N+2) \\ \vdots \\ y(k) \end{bmatrix} \quad (17)$$

where R_{new} and $\hat{Y}_{new}$ are the updated matrices with newly added centers.

B. Prune Centers

In order to maintain the size of the center bank, an insignificant center is pruned from the center bank. In other words, a center which has the least contribution to the network performance is removed. For an RBFN's structure, pruning a center implies removing a hidden layer neuron which is associated to a column vector in matrix R. To calculate the modeling residual, each column of matrix R is removed, sequentially, and the matrix R is re-triangularized [3, 18]. The pruning algorithm using orthogonal decomposition developed in [3, 18] is as follows. If jth center is removed, the corresponding jth column vector of matrix R, r_j is removed as well, which results in matrix R',

$$R' = \left[r_1, \ldots, r_{j-1}, \ldots, r_{j+1}, \ldots, r_{n_h}\right]. \quad (18)$$

After the removal of the column r_j, the matrix R' is no longer an upper triangular matrix. Thus, it is necessary to re-triangularize the matrix R',

$$R' = Q' \begin{bmatrix} R_j \\ \ldots \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \hat{Y}_j \\ \ldots \\ \tilde{y}_j^T \end{bmatrix} = Q'^T \hat{Y} \quad (19)$$

and the cost function becomes

$$J_j^2 = \left\| Q' \begin{bmatrix} \hat{Y}_j \\ \cdots \\ \tilde{y}_j^T \end{bmatrix} - Q' \begin{bmatrix} R_j \\ \cdots \\ 0 \end{bmatrix} W_j \right\|_F^2 = \left\| \begin{matrix} \hat{Y}_j - R_j W_j \\ \tilde{y}_j^T \\ \tilde{Y} \end{matrix} \right\|_F^2$$

$$= \left\| \hat{Y}_j - R_j W_j \right\|_F^2 + \left\| \tilde{y}_j^T \right\|_F^2 + \left\| \tilde{Y} \right\|_F^2. \quad (20)$$

The weight, W_j can be solved from

$$R_j W_j = \hat{Y}_j \quad (21)$$

The residual is given as

$$\left\| \tilde{Y}_j \right\|_F^2 = \left\| \tilde{y}_j^T \right\|_F^2 + \left\| \tilde{Y} \right\|_F^2. \quad (22)$$

From (22), it can be seen that the increment in residual caused by removing the jth column of matrix R, jth center, is $\left\| \tilde{y}_j^T \right\|_F^2$. Thus, the procedure is summarized as: use (18) to remove the column of matrix R in turn and compute the residual $\left\| \tilde{y}_j^T \right\|_F^2$ using (22). Then, the jth column of matrix R with least residual $\left\| \tilde{y}_j^T \right\|_F^2$ is removed, and matrix R is re-triangularized using (19).

C. Group Centers

After the adding and pruning centers, the centers in the center bank are classified into two groups which are active centers and redundant centers. The centers in the active group will be used to predict the process output, while the centers in the redundant group will not be included in the network for process output prediction at this sampling period, but will be preserved for later use in the consequent sampling instants. So, the relation between the hidden neurons and the output neurons for active and inactive centers are illustrated in Fig.2. While, the redundant centers, which may contain the information for next sample time, are preserved in the center bank.

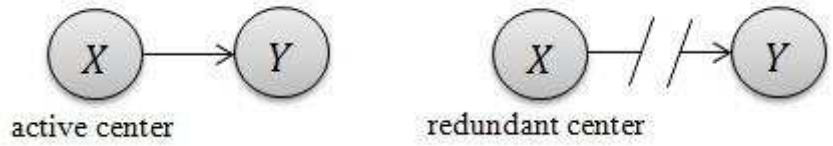

Fig. 2 The connections between hidden neuron X and output neuron Y for active and redundant centers

The center pruning algorithm provides a good foundation for center grouping. This is implemented by evaluating the modeling residual when each center is grouped as a redundant center, sequentially. When the grouping procedure stops, the remaining centers would be active centers. In other words, it is the contribution of each center to the network performance that decides which group the center belongs to. Akaike's final prediction error (FPE) criterion in is used to stop the grouping procedure,

$$FPE = \frac{1+\beta(n_p/N)}{1-\beta(n_p/N)}V, \quad V = \left\|\tilde{Y}(N)\right\|_F^2/(N) \tag{23}$$

where V is the loss function, n_p is the number of weights and β is a weighting factor. The value of $\beta = 2$ is suggested in [3]. However, due to that the sample data N is a fixed parameter in (23) for every sample time k, the value of β can be manipulated to decide the number of active centers. In order to stop the grouping procedure, FPE has to be larger than the past FPE FPE_i. Thus, the equation to calculate number of active centers is derived as

$$1-\beta(n_p/N) = 0$$

or

$$n_p = \frac{N}{\beta} \qquad c_a = \frac{N}{\beta} \tag{24}$$

where c_a is the number of active centers.

The procedure of center grouping algorithm is summarized as follows:

Step 1 Initialize V and FPE for the network after updating the center bank.

Step 2 Compute the new loss function V_j when each center is grouped in turn using (18) and (22).

Step 3 Set i = arg min (V_j) and compute the FPE for the smallest loss function, FPE_i using (23). If $FPE_i < FPE$, group the center i as redundant center and go to step 4. If $FPE_i > FPE$, go to step 5.

Step 4 Then, set $= R_i$, $\hat{Y} = \hat{Y}_i$, $V = V_i$, FPE $=$ FPE$_i$ and $n_h = n_h - 1$. Go to step 2.

Step 5 Stop the grouping procedure. The remaining centers in the center bank are active centers and the optimal weight W_j can be computed using (21).

D. ADS-RBFN Adaption Procedure

At each sample time, the center bank will be updated with the adding, pruning and grouping of centers. The main step of the proposed adaptive algorithm is summarized as follows.

Step 1 Initialize an initial RBFN by using a set of N samples data, form a center bank by arbitrarily choosing data points and obtain an initial matrix R and W.

Step 2 At each sample time k, update the matrix R with new data $x(k)$ using (9). Evaluate the contribution of centers and add a new center into center bank using (14) and (15), respectively. Then, generate a matrix R_{new} and $\hat{Y}_{new}$ using (16) and (17), respectively.

Step 3 Prune a center that causes the least increase in modeling residual from the center bank by following the summarized pruning procedure given in Section B.

Step 4 Group the centers in the center bank into two groups: active and redundant centers, using the provided grouping procedure in Section C. Use the active center to form a network model to make prediction.

Step 5 $k = k + 1$, go to step 2.

IV. NUMERICAL EXAMPLE

To demonstrate the proposed algorithm, the ADS-RBFN is used to model a nonlinear dynamic system with a large migration of the operating point for one-step-ahead prediction. The system is chosen from [19],

$$y(k+1) = \frac{y(k)}{1 + y(k)^2} + u(k)^3 \quad (25)$$

A set of 900 input/output data samples has been generated and collected in a specific way where the system outputs fall into three obvious different regions. Region 1 represents the first 330 data, region 2 represents data samples from 331 to 660, and region 3 represents the 661 to 900 data. This is to test the effectiveness of adaptation of the proposed algorithm.

The ADS-RBFN is chosen to have two inputs, one output and an initial center bank with 20 centers. The β in FPE is selected as 4. The number of active centers is calculated using (24). Thus, there are 13 active centers and 7 redundant centers in the center bank.

In order to evaluate the performance of the ADS-RBFN, two fixed structure RBF networks (FS-RBFNs) are employed for performance comparison purpose. With K-means algorithm, first FS-RBFN has 20 centers distributed in region 1 of the system as shown in Fig. 3 (top). For second FS-RBFN, which was employed as a two-stage training for an RBFN [20] , 20 centers are used and are distributed in the whole operating space including all the three regions as shown in Fig. 3 (center). In addition, mean absolute error (MAE) is used to measure the network prediction errors. After training, another set of data with the same number of samples is acquired and used to test the three trained network. The first and the second network model with the fixed structure and preselected centers, while the last network model uses the proposed algorithm to adapt the structure on-line. The MAE obtained in the test for the three networks are listed in Table I.

TABLE I
Performance comparison of the three RBFNs

Networks	MAE
FS-RBFN with 20 centers at region 1	1.123
FS-RBFN with 20 centers at all regions	0.2638
ADS-RBFN	0.082

The performance of the two FS-RBFNs is displayed in Fig. 4 and Fig. 5, respectively. In Fig. 4, it shows that the first FS-RBFN only performs well in region 1 where the centers are distributed, as shown in Fig. 3 (top). The degradation of performance can be clearly observed at region 2 and region 3 as the model predictions are considerably deviated from the process output. For the second FS-RBFN, the performance is not satisfactory in all regions especially in region 3. This is due to that the centers do not sufficiently cover the data region of the system. In order to improve its performance, a bigger set of centers is needed but unnecessary big size of RBFN will cause poor generalization [4]. In comparison to the two FS-RBFNs, the result of the ADS-RBFN in Fig. 6 clearly shows that it is an ideal model that accurately predicts the system outputs for all three different regions, because the 13 active centers were adapted effectively to all regions of the system output, as illustrated in Fig. 3 (bottom). It can be observed that the 13 active centers emigrate from region 1 to region 2, then to region 3 following the moving of the system's operating point. Also from Table I, the values of MAE

clearly suggest that the ADS-RBFN has the best performance among three networks. Moreover, it has a more compact structure with only 13 centers.

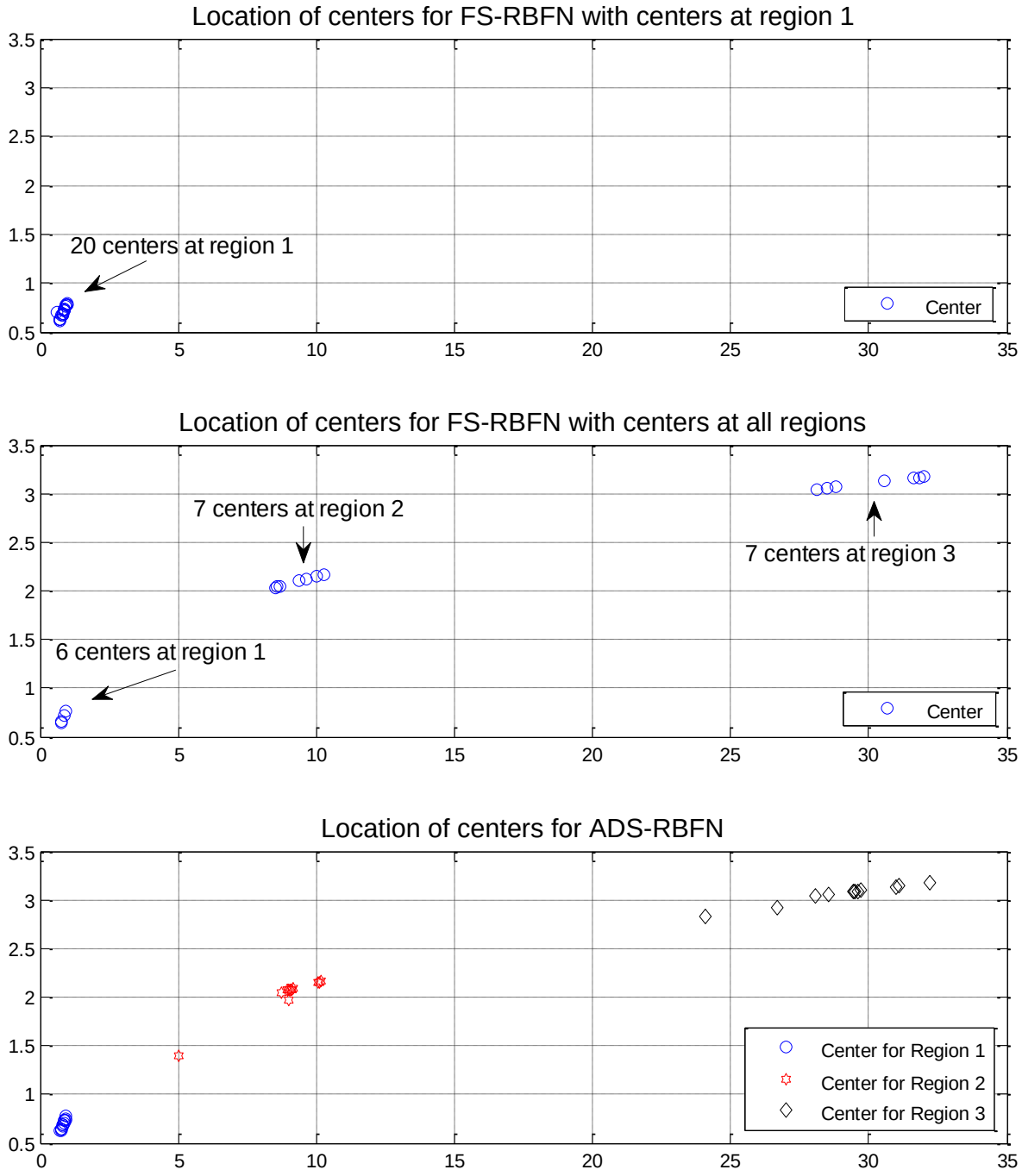

Fig. 3 The locations of centers for three networks

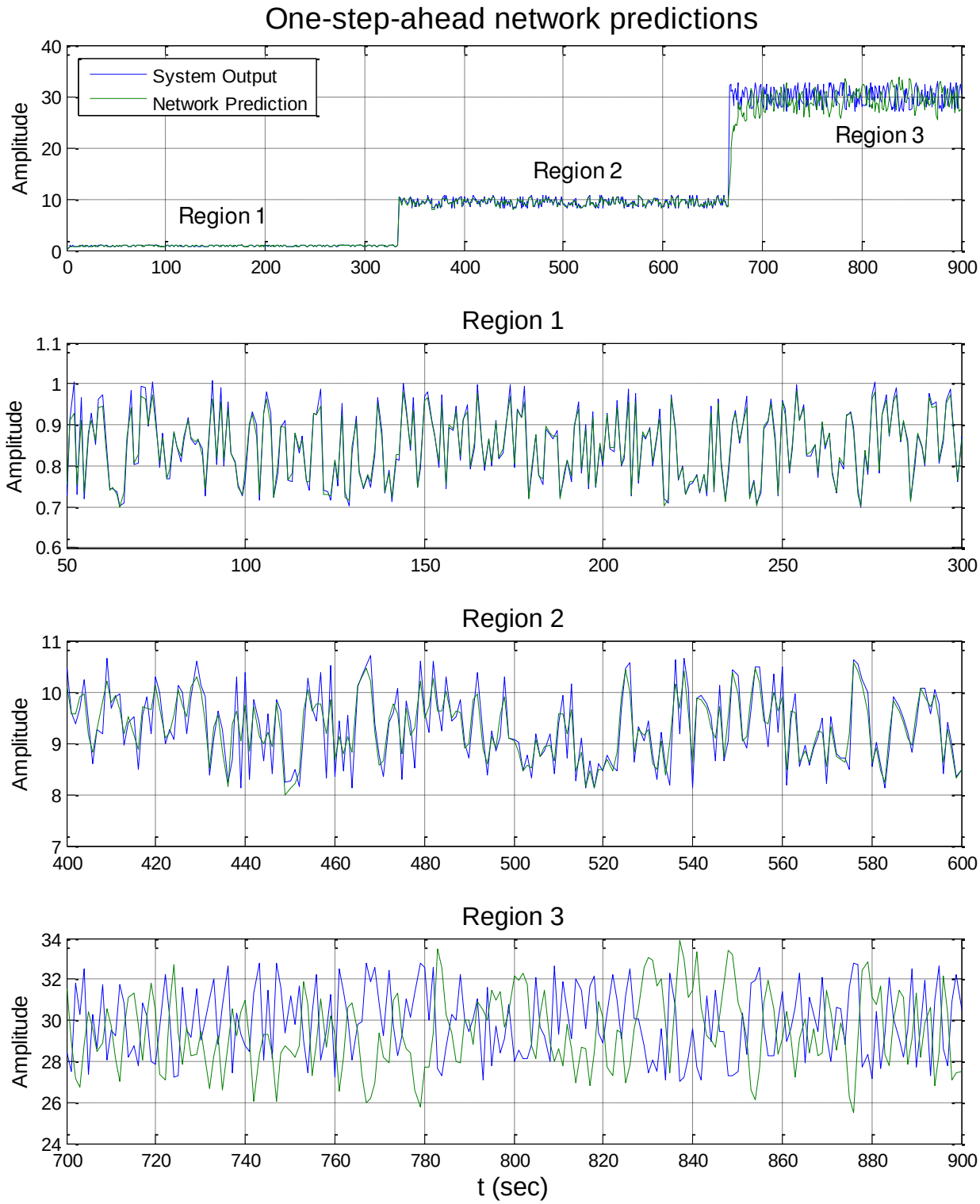

Fig. 4 Performance of FS-RBFN with centers distributed in region 1

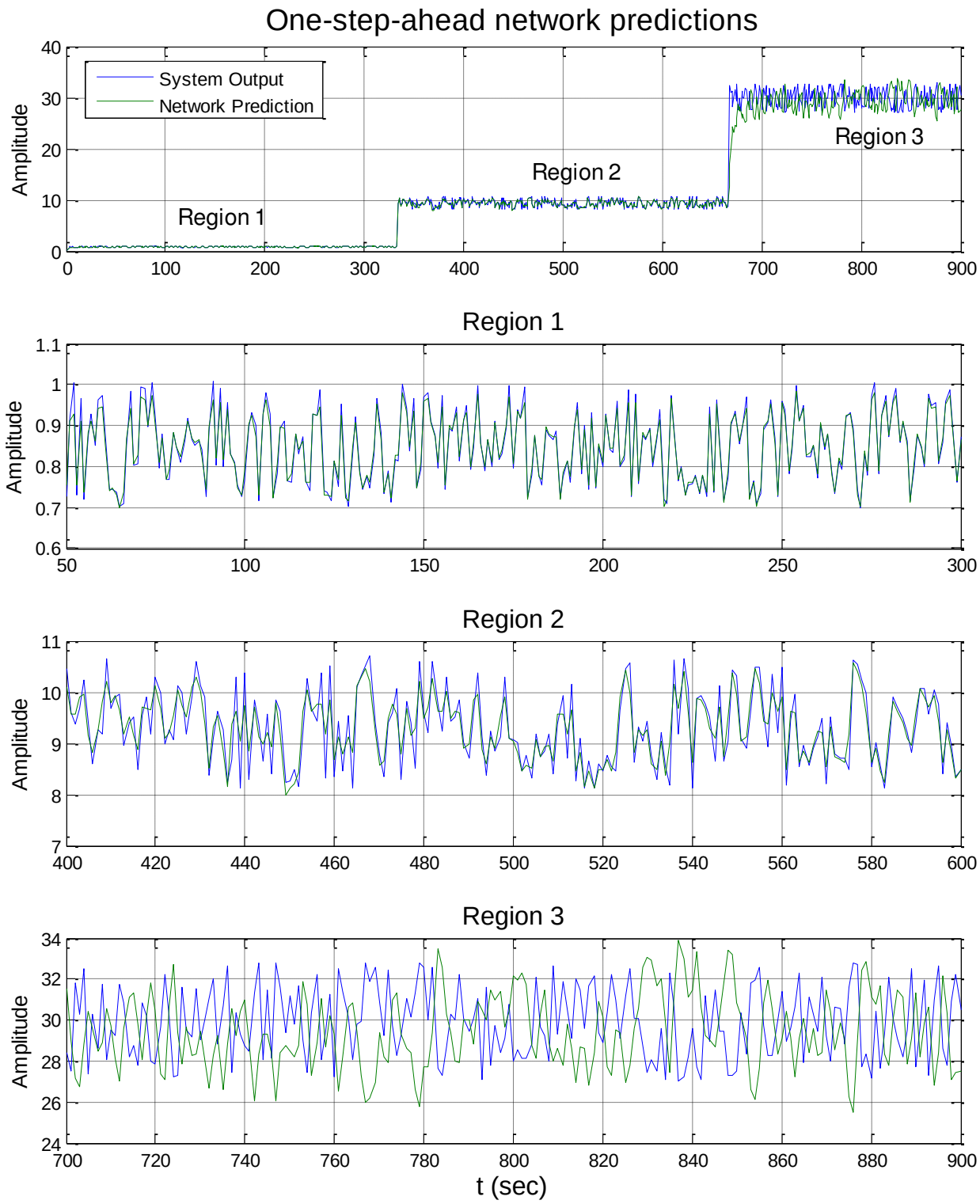

Fig. 5 Performance of FS-RBFN with centers distributed in all three regions

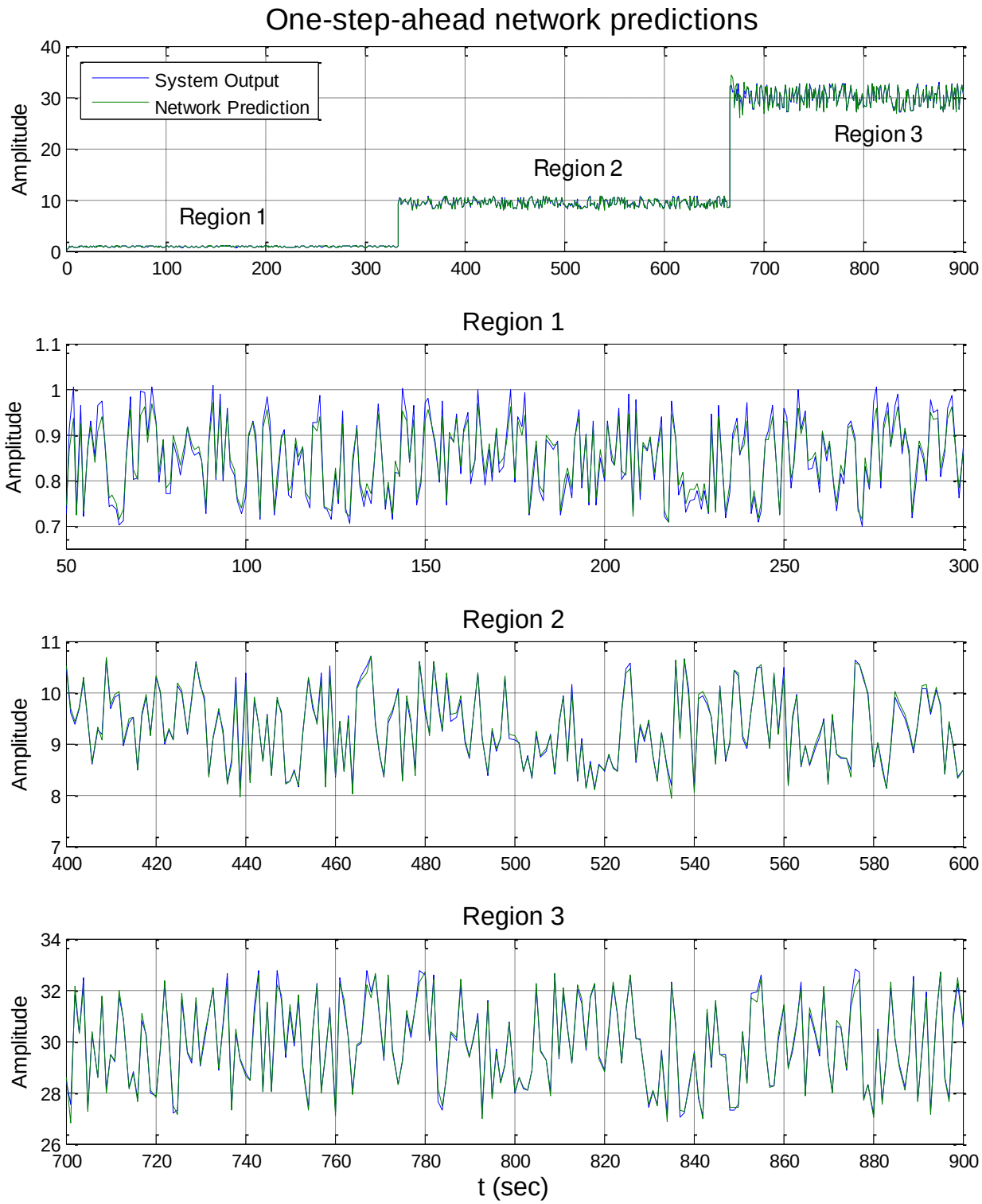

Fig. 6 Performance of ADS-RBFN in three regions

V. CONCLUSION

In this paper, a new algorithm based on the ROLS training is proposed for designing a structure adaptive RBFN model. A new strategy of adding new centers based on the information of the center with the most contribution and the new data is developed. In the meantime, the insignificant center is pruned from the center bank to maintain the minimum size of the network model. In addition, a small modification on the parameter of FPE enables the network to have a compact architecture by grouping the centers in the center bank. The effectiveness of the proposed algorithm is demonstrated by applying it in modeling a nonlinear numerical example with significant operating point emigration. The simulation results demonstrate that the developed ADS-RBFN adapts its structure dynamically following the emigration of the system operating point without degrading the prediction performance. Comparison with the two fixed structure RBFN shows that it outperforms the FS-RBFNs.

Acknowledgment

This work is partially supported by the National Nature Science Foundation of China under Grant 61004080, 61273188, Shandong Provincial Natural Science Foundation under Grant ZR2011FM003, and the Fundamental Research Funds for the Central Universities of China, Development of key technologies project of Qingdao Economic and Technological Development Zone under Grant 2011-2-52, Taishan Scholar Construction Engineering Special funding.

REFERENCE

1. Hao, Y., et al., Advantages of Radial Basis Function Networks for Dynamic System Design., IEEE Trans. on Industrial Electronics, 2011. **58**(12): p. 5438-5450.
2. Broomhead, D.S. and D. Lowe, Multivariable functional interpolation and adaptive networks. Complex Systems, 1988. **2**: p. 321-355.
3. Gomm, J.B. and D.L. Yu, Selecting radial basis function network centers with recursive orthogonal least squares training. IEEE Trans. on Neural Netwoks, 2000. **11**(2): p. 306-314.
4. Chen, S., P.M. Grant, and C.F.N. Cowan, Orthogonal least-square algorithm for training multioutput radial basis function networks., IEE Proceedings F: Radar and Signal Processing, 1992. **139**(6): p. 378-384.

5. Platt, J., A resource-allocating network for function interpolation. Neural Computing, 1991. **3**: p. 213-225.
6. Karayiannis, N.B. and G.W. Mi, Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques. IEEE Trans. on Neural Netwoks, 1997. **8**(6): p. 1492-1506.
7. Lu, Y., N. Sundararajan, and P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks. Neural Computing, 1997. **9**: p. 461-478.
8. Lu, Y., N. Sundarajan, and P. Saratchandran, Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. IEEE Trans. on Neural Netwoks, 1998. **9**(2): p. 308-318.
9. Qiao, J.-F. and H.-G. Han, Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. Automatica, 2012. **48**(8): p. 1729-1734.
10. Han, H.-G., Q.-l. Chen, and J.-F. Qiao, An efficient self-organizing RBF neural network for water quality prediction. Neural Networks, 2011. **24**(7): p. 717-725.
11. Huang, G.-B., P. Saratchandran, and N. Sundarajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans. on Neural Netwoks, 2005. **16**(1): p. 57-67.
12. Chen, S., C. F. N. Cowan, and P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. on Neural Netwoks, 1991. **2**(2): p. 302-309.
13. Chen, S. and S.A. Billings, Neural Network for nonlinear dynamic system modeling and identification. Int. J. Control, 1992. **56**(2): p. 337-340.
14. Chng, E., H.H. Yang, and S. Bos, Orthogonal least-squares learning algorithm with local adaptation process for the radial basis function networks. IEEE Signal Processing Letters, 1996. **3**(8): p. 253-255.
15. Yu, D.L., J.B. Gomm, and D. Williams, A recursive orthogonal least squares algorithm for training RBF networks. Neural Processing Letters, 1997. **5**(3): p. 167-176.
16. Yu, D.L., et al. Adaptive RBF model for model-based control. 2004 WCICA Fifth World Congress on Intelligent Control and Automation. 2004, Hangzhou, China.
17. Yu, D.L. and D.W. Yu, A new structure adaptation algorithm for RBF networks and its application. Neural Computing and Applications, 2007. **16**(1): p. 91-100.

18. Hong, X. and S.A. Billings, Givens rotation based fast backward elimination algorithm for RBF neural network pruning., IEE Proceedings Part D: Control Theory and Applications, 1997. **144**(5): p. 381-384.
19. Narendra, K.S. and K. Parthasarathy, Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks, 1990. **1**(1): p. 4-27.
20. Wang, S.W., et al., Adaptive neural network model based predictive control for air–fuel ratio of SI engines. Engineering Applications of Artificial Intelligence, 2006. **19**(2): p. 189-200.