A Fast Adaptive Tunable RBF Network For Nonstationary Systems

Hao Chen, Yu Gong, Xia Hong, *Senior Member, IEEE*, and Sheng Chen, *Fellow, IEEE*

***Abstract*—This paper describes a novel on-line learning approach for radial basis function (RBF) neural network. Based on an RBF network with individually tunable nodes and a fixed small model size, the weight vector is adjusted using the multi-innovation recursive least square algorithm on-line. When the residual error of the RBF network becomes large despite of the weight adaptation, an insignificant node with little contribution to the overall system is replaced by a new node. Structural parameters of the new node are optimized by proposed fast algorithms in order to significantly improve the modeling performance. The proposed scheme describes a novel, flexible, and fast way for on-line system identification problems. Simulation results show that the proposed approach can significantly outperform existing ones for nonstationary systems in particular.**

***Index Terms*—Multi-innovation recursive least square (MRLS), nonlinear, nonstationary, on-line identification, radial basis function (RBF).**

I. Introduction

ADAPTIVE system identification whereby the system model is computed recursively in time is central for obtaining mathematical models for changing systems, e.g., system control based on most recent models or electroencephalogram time series prediction for time varying signals [1]–[3]. These models can also be used in conditioning monitoring, e.g., fault detection. The adaptive modeling of nonstationary systems is particularly challenging since the conventional parameter updating may be insufficient to track the data change; or the reestimating model structurally can be computationally expensive for real time applications. This paper is aimed at compromising both issues.

Radial basis function (RBF) neural network has been widely used as an on-line approach in nonlinear system identification [4]–[7]. With a cluster of nonlinear "kernels" (or nodes) imposed on the input data, the RBF network can approximate any continuous function to an arbitrary degree of accuracy. The RBF tracks the dynamic of the systems by constantly updating the output layer weights with linear adaptive algorithms such as the least mean square and recursive least square (RLS). More recently a variant RLS, referred to as the multi-innovation RLS (MRLS) has been proposed [8]. Unlike the classic RLS algorithm which only considers the current residual error, the MRLS adaptation is based on a number of recent errors so that its performance is more robust to noise variations.

The overall performance of the RBF network depends on both the output layer weights and the structure of the network. The RBF structure is determined by multiple parameters including the number of nodes (or the model size), and the centers and variances of each node (or the position and shape of each node, respectively). In a traditional RBF network, the structure of the network is usually fixed at some compromise settings or based on experience, and only the output layer weights are updated with time. While this describes a simple realization, it is often not sufficient especially in nonstationary systems. In the resource allocating network (RAN), the RBF model starts from empty and grows with the input data based on the nearest neighbor method [9]. While the RAN algorithm can only grow the model size, a number of variants have since been proposed. Particularly the growing-and-pruning RBF (GAP-RBF) algorithm describes a computational efficient approach to adjust the RBF model size [10]. Alternatively, the on-line structure adjustment may be avoided by using a large RBF model size covering a wide dynamic range of data input [11]. Typical examples are the extreme learning machine (ELM) and its variant (see [12] and the reference therein), in which a large number of kernel nodes are randomly generated at initialization stage and fixed during the learning process. The ELM approach can achieve high accuracy with fast learning speed in some applications, but the model size may have to be very large especially in nonstationary systems and the model generalization is not guaranteed.

A common drawback in both the RAN-like and the ELM-like approaches is that none of them optimizes the structure of the RBF nodes. The node centers are either determined by the input data such as in the k-means approach (see [13]–[15]) or simply set as the input data (see [16]), and

Manuscript received May 15, 2015; revised July 27, 2015; accepted September 20, 2015. Date of publication October 28, 2015; date of current version November 15, 2016. This paper was recommended by Associate Editor L. Rutkowski.

H. Chen is with the Quanzhou Institute of Equipment Manufacturing, Haixi Institutes, Chinese Academy of Sciences, Jinjiang 362200, China (e-mail: aries4chen@hotmail.com).

Y. Gong is with the School of Mechanical, Manufacturing, and Electrical Engineering, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: y.gong@lboro.ac.uk).

X. Hong is with the School of Systems Engineering, University of Reading, Reading RG6 6AY, U.K. (e-mail: x.hong@reading.ac.uk).

S. Chen is with the Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K., and also with King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: sqc@ecs.soton.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2484378

often a common variance is used for all nodes which is set by the trial-and-error or cross-validation method [17]. As a result, the constructed network structure only fits into the data rather than the underlying model. This usually makes the model size increase with the number of the sample data, ending up with a very large model with high complexity and poor tracking performance especially in nonstationary environment. Therefore, it is highly necessary to optimize the node structure which is in general a difficult NP problem. A tunable RBF model identification algorithm was proposed in [18], where each RBF node has a tunable center vector and an adjustable diagonal covariance matrix which are optimized using the particle swarm optimization (PSO) algorithm at each forward regression stage. While this provides an exceptionally flexible RBF model with significantly reduced model size, the proposed approach is for off-line (batch) learning which is inadequate for on-line system identification. It is inspired by the tunable RBF network in [18] that, if the structure (i.e., the centers and variances) of the nodes is optimized, an efficient RBF network with much smaller model size than a conventional RBF network can be constructed. In [19], an on-line tunable RBF network was proposed, in which the model size is fixed to a small number, but the worst fit node is replaced with a new node when necessary, and the structure of the new node is optimized by the PSO algorithm to “fit” for the recent input data. Although this PSO-based tunable network has excellent performance in on-line modeling, it demands high complexity due to the complicated PSO algorithm, deterring its use in many practical on-line applications. This motivates us to investigate fast algorithm for on-line node structure optimization in this paper.

In this paper, the RBF model size is also fixed to a small number, and the worst performing node is replaced with a new node when the current RBF model does not fit the current data. Unlike our previous work in [19], the structure of the new node in this paper is not optimized by the PSO algorithm, or any other evolutionary algorithms [20]. Instead we propose a fast structure optimization approach based on iterative adaptation, which can achieve excellent modeling performance with a sparse model, yet keeps low complexity. This makes the proposed RBF network particularly suitable for on-line applications where the computation cost is often a key issue. While the proposed approach can be used in a wide range of nonlinear and nonstationary applications (where the conventional RBF networks usually apply), in the simulation part, the proposed approach is verified on two benchmark applications: 1) adaptive noise cancellation (ANC) and 2) Lorenz time series prediction. In summary, the main contribution of this paper is to propose a fast node structure optimization approach in a sparse RBF network, which includes the following aspects specifically.

1) Proposing an instantaneous error gradient descent approach to fast optimize the node structural parameters. Compared with our previous approach which uses PSO approach for the structure adaptation, the gradient descent approach not only has low complexity but also is well known for its good stability in implementation.

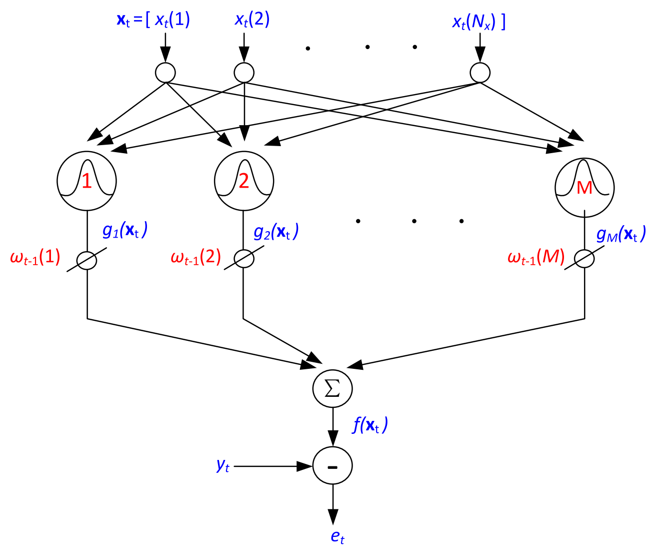


Fig. 1. RBF neural network.

2) Proposing a fast weight adaptation based on inverse matrix lemma, so that the weight vector can be efficiently updated with the newly optimized node structure.
3) Describing several ways to initialize the node structure adaptation to ensure fast convergence.

The rest of this paper is organized as follows. Section II describes the RBF network with tunable nodes. Section III proposes fast approaches for the node optimization. Section IV summarizes the proposed algorithm. Section V verifies the proposed approach via numerical simulations. Finally, Section VI concludes the paper.

II. RBF Network With Tunable Nodes

A. Adaptive RBF Network

The adaptive RBF neural network is shown in Fig. 1, where there are M number of nodes. At time t, the input vector of the RBF network is given by

$$\mathbf{x}_t = [x_t(1), x_t(2), \ldots, x_t(N_x)]^{\mathrm{T}} \tag{1}$$

where N_x is the model input dimension or the number of input channels, and $x_t(i)$ is the input data from the ith input channel at time t. The RBF network output is given by

$$f(\mathbf{x}_t) = \sum_{i=1}^{M} w_{t-1}(i) g_i(\mathbf{x}_t) \tag{2}$$

where $g_i(\mathbf{x}_t)$ is the output of the ith node, $w_{t-1}(i)$ is the weight coefficient for the ith node at time $t-1$. Letting $\mathbf{w}_{t-1} = [\omega_{t-1}(1), \ldots, \omega_{t-1}(M)]^{\mathrm{T}}$ be the weight vector and $\boldsymbol{\phi}_t = [g_1(\mathbf{x}_t), \ldots, g_M(\mathbf{x}_t)]^{\mathrm{T}}$ be the information vector, we can rewrite (2) as

$$f(\mathbf{x}_t) = \boldsymbol{\phi}_t^{\mathrm{T}} \mathbf{w}_{t-1}. \tag{3}$$

The RBF residual error at time t is given by

$$e_t = y_t - \boldsymbol{\phi}_t^{\mathrm{T}} \mathbf{w}_{t-1} \tag{4}$$

where y_t is the observed system output at time t. In this paper, we assume without losing generality that the RBF basis function is Gaussian so that

$$g_i(\mathbf{x}_t) = \exp\bigl(-(\mathbf{x}_t - \mathbf{c}_i)^{\mathrm{T}}\mathbf{H}_i(\mathbf{x}_t - \mathbf{c}_i)\bigr) \tag{5}$$

where $\mathbf{c}_i = [c_i(1), \ldots c_i(N_x)]^{\mathrm{T}}$ which is the center vector of the ith node, $\mathbf{H}_i = \mathrm{diag}\{\sigma_i^2(1), \ldots, \sigma_i^2(N_x)\}$ which is the diagonal covariance matrix of the ith node, $c_i(j)$ and $\sigma_i^2(j)$ are the jth center and variance of the ith RBF node, respectively.

For the Gaussian RBF network, the node structure parameters include N_x pairs of centers and variances of each node, or $c_i(j)$ and $\sigma_i^2(j)$ in (5), respectively. Because the input statistic keeps varying in a nonstationary system, the "local characteristic" of the input data is often more important than the "global characteristic" [21]. This implies that the model size needs not to be large since the modeling focuses more on the recent data than the older ones. Based on this observation, we propose to fix the RBF model size at a small number, where each node has a tunable center vector and an adjustable diagonal covariance matrix which can be optimized on-line one at a time. It will be shown later that fixing the model size enables not only the MRLS to be seamlessly integrated with the node structure adjustment, but also fast approaches for the node optimization.

B. Weight Adaptation With the MRLS Algorithm

The weight vector $\mathbf{w}$ can be updated by the MRLS algorithm which is based on both current and past residual errors [8]. Putting p number of input vectors into an input matrix gives

$$\mathbf{X}_t = \bigl[\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-p+1}\bigr]^{\mathrm{T}} \in \mathbb{R}^{p\times N_x} \tag{6}$$

where p is the innovation length which determines the number of past errors used for weight adaptation. In general, a larger p has more potential in noise rejection with higher complexity and slower convergence [22]. In the noise-free case, we can have $p = 1$ so that the MRLS reduces to the classic RLS algorithm.

Passing $\mathbf{X}_t$ through the RBF nodes gives the information matrix as

$$\mathbf{\Phi}_t = \bigl[\mathbf{g}_1,\ \mathbf{g}_2, \ldots, \mathbf{g}_M\bigr] \in \mathbb{R}^{p\times M} \tag{7}$$

where $\mathbf{g}_i = [g_i(\mathbf{x}_t), g_i(\mathbf{x}_{t-1}), \ldots, g_i(\mathbf{x}_{t-p+1})]^{\mathrm{T}}$ which is the ith RBF regressor. Letting $\mathbf{e}_t = [e_t, e_{t-1}, \ldots, e_{t-p+1}]^{\mathrm{T}}$ and $\mathbf{y}_t = [y_t, y_{t-1}, \ldots, y_{t-p+1}]^{\mathrm{T}}$, we have the vector/matrix expression of (4) as

$$\mathbf{e}_t = \mathbf{y}_t - \mathbf{\Phi}_t\mathbf{w}_{t-1}. \tag{8}$$

With $\mathbf{\Phi}_t$ and $\mathbf{e}_t$, the MRLS adaption rules are given by the following steps:

$$\mathbf{\Psi}_t = \mathbf{P}_{t-1}\mathbf{\Phi}_t^{\mathrm{T}}\bigl[\lambda\mathbf{I}_p + \mathbf{\Phi}_t\P_{t-1}\mathbf{\Phi}_t^{\mathrm{T}}\bigr]^{-1} \tag{9}$$

$$\mathbf{P}_t = (\mathbf{P}_{t-1} - \mathbf{\Psi}_t\mathbf{\Phi}_t\mathbf{P}_{t-1})\lambda^{-1} \tag{10}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{\Psi}_t\mathbf{e}_t \tag{11}$$

where $\mathbf{\Psi}_t \in \mathbb{R}^{M\times p}$ is the Kalman gain matrix, $\mathbf{P} \in \mathbb{R}^{M\times M}$ is the inverse of the covariance matrix, $\mathbf{I}_p$ is the $p \times p$ identity matrix, and λ is the forgetting factor. $\mathbf{P}_t$ is usually initialized as $\mathbf{P}_0 = \delta\mathbf{I}_M$, where δ is a large constant.

C. Node Replacement

When the residual error becomes large no matter how we adapt the weight vector using MRLS above, this indicates that the current RBF structure is no longer suitable for the current data and needs updating. An insignificant node with little contribution to the overall system is replaced with a new node. In order to prevent the node replacement from occurring too frequently, the normalized "average" residual error is used to measure the overall RBF performance as

$$\bar{e}_t^2 = \frac{1}{p}\cdot\frac{\|\mathbf{e}_t\|^2}{\|\mathbf{y}_t\|^2} \tag{12}$$

where the multi-innovation error vector $\mathbf{e}_t(n)$ in (8) consists p number of most recent errors. We have the following criterion:

$$\begin{cases} \text{if } \bar{e}_t^2 < \Delta_1, & \text{the RBF structure remains unchanged} \\ \text{if } \bar{e}_t^2 \geq \Delta_1, & \text{an insignificant node is replaced with} \\ & \text{a new node} \end{cases} \tag{13}$$

where Δ_1 is a constant threshold which is set according to the performance requirement. In general, a small Δ_1 leads to small residual error and frequent node replacements.

When $\bar{e}_t^2 \geq \Delta_1$, the most insignificant node with least contribution to the overall system performance is replaced with a new node. The "significance" of a node is revealed by the weighted node-output variance (WNV) as

$$\mathrm{WNV}_i = \|\ \omega_{t-1}(i)\mathbf{g}_i\ \|^2 = \omega_{t-1}^2(i)\mathbf{g}_i^{\mathrm{T}}\mathbf{g}_i \tag{14}$$

where $\mathbf{g}_i$ is defined in (7) containing the recent p outputs from node i. The WNV-s for all nodes are ordered as

$$\mathrm{WNV}_{1'} \leq \mathrm{WNV}_{2'} \leq \cdots \leq \mathrm{WNV}_{M'} \tag{15}$$

where $\mathrm{WNV}_{i'}$ is for node i' with the ith smallest WNV. Then from (15), the node $1'$ with the smallest WNV is replaced with a new node.

III. Node Optimization With Iterative Adaptation

At each time step, the weight vector is adapted by the MRLS algorithm. The residual error of the network output is monitored. If the RBF network performs poorly, or the residual error is large, an insignificant node with the smallest WNV is replaced with a new node without changing the model size. The structure of the new node is then optimized to fit for the current input data. At the same time, the weight vector should also be updated as otherwise it is only suitable for the old network before the node replacement.

In this section, we describe fast algorithms for the node structure and the weight vector adaptation. We assume without losing generality that, at time t, the Mth node is replaced with a new node. While the joint optimization of the structure of the new node and the weight vector is too complicated, we propose an iterative adaptation approach. At every iteration, either the structure of the new node or the weight vector is fixed first, and the other is updated, and vice versa. The iteration continues until the modeling residual error is sufficiently small or the maximum iteration number is reached.

A. Fast Structural Parameters Update Using Instantaneous Error Gradient Descent

At every iteration, when the weight vector is fixed, the center and the inverse of the variance for the new node are tuned by minimizing the two-norm of the instantaneous error. The instantaneous error at time t can be expressed as

$$e_t = y_t - \sum_{i=1}^{M-1} w_{t-1}(i) g_i(\mathbf{x}_t) - w_{t-1}(M) g_M(\mathbf{x}_t). \quad (16)$$

From (5), the kernel function for the new node $g_M(\mathbf{x}(t))$ is obtained as

$$g_M(\mathbf{x}_t) = \exp\left(- \sum_{j=1}^{N_x} \eta_M(j) \cdot (x_t(j) - c_M(j))^2 \right) \quad (17)$$

where $\eta_M(j) = 1/\sigma_M^2(j)$ which is the inverse of the variance for the jth input of the Mth node. While the model size and the structure of the remaining nodes remain unchanged, the structural vector for adaptation is defined as

$$\boldsymbol{\Gamma} = [c_M(1), \ldots, c_M(N_x), \eta_M(1), \ldots, \eta_M(N_x)]^{\mathrm{T}}. \quad (18)$$

The objective is to find the optimum $\boldsymbol{\Gamma}$ which minimizes e_t^2. With the weight vector being fixed, the optimum $\boldsymbol{\Gamma}$ can be found by the gradient descent search as

$$\boldsymbol{\Gamma}_l = \boldsymbol{\Gamma}_{l-1} - \varepsilon \frac{\boldsymbol{\nabla}}{\|\boldsymbol{\nabla}\|} e_t \quad (19)$$

where subscript l represents the iteration step l, ε is a small positive step size, $\|.\|$ denotes Euclidian norm, and $\boldsymbol{\nabla}$ is the gradient vector which is given by

$$\begin{aligned} \boldsymbol{\nabla} &= \frac{\partial(e_t)}{\partial \boldsymbol{\Gamma}} \\ &= \left[\frac{\partial e_t}{\partial c_M(1)}, \ldots, \frac{\partial e_t}{\partial c_M(N_x)}, \frac{\partial e_t}{\partial \eta_M(1)}, \ldots, \frac{\partial e_t}{\partial \eta_M(N_x)} \right]^{\mathrm{T}}. \end{aligned} \quad (20)$$

From (17), we can easily obtain that

$$\boldsymbol{\nabla} = \begin{bmatrix} -2\eta_M(1) w_{t-1}(M) g_M(\mathbf{x}_t)[(x_t(1) - c_M(1))] \\ \vdots \\ -2\eta_M(N_x) w_{t-1}(M) g_M(\mathbf{x}_t)[x_t(N_x) - c_M(N_x)] \\ w_{t-1}(M) g_M(\mathbf{x}_t)[x_t(1) - c_M(1)]^2 \\ \vdots \\ w_{t-1}(M) g_M(\mathbf{x}_t)[x_t(N_x) - c_M(N_x)]^2 \end{bmatrix}. \quad (21)$$

We highlight that, because $\eta_M(j)$ is the inverse of the variance which must be positive, it should set to be a small positive value whenever it appears negative during the iteration. The computational cost of the above gradient descent search is in the order of $O(N_x)$ scaled by the iteration number.

B. Fast Weight Adaptation Using the Inverse of Matrix Block Decomposition Lemma

At every iteration, when the node structure is fixed, the weight vector is calculated by the least-square (LS) method based on the recent input data. It is reasonable to use only the recent input data for the LS weight estimation because, when the node replacement occurs, the underlying system varies significantly (otherwise no node is replaced), and the previous data should be "forgotten" in order to capture the "local" characteristic.

The LS estimate of $\mathbf{w}_t$ based on recent q input vectors is obtained by minimizing the least squares cost function as

$$J_t = \sum_{j=t}^{t-q+1} e_t^2(j) = \mathbf{e}_t^{\mathrm{T}} \mathbf{e}_t = (\mathbf{y}_t - \boldsymbol{\Phi}_t \mathbf{w}_t)^{\mathrm{T}} (\mathbf{y}_t - \boldsymbol{\Phi}_t \mathbf{w}_t) \quad (22)$$

where $\boldsymbol{\Phi}_t$ is a $q \times M$ matrix which is obtained similar to (7), and $\mathbf{y}_t$ consists of recent q number of observed system output. In general, a large q has good LS estimation accuracy but may loss the performance in tracking, and a small q is used in highly nonstationary systems with, for example, abrupt changes.

For a given $g_M(.)$, the LS solution of (22) is given by

$$\mathbf{w}_t = \mathbf{P}_t \boldsymbol{\Phi}_t^{\mathrm{T}} \mathbf{y}_t \quad (23)$$

where $\mathbf{P}_t = (\boldsymbol{\Phi}_t^{\mathrm{T}} \boldsymbol{\Phi}_t)^{-1}$. For a given structure of the new node $g_M(.)$, (23) is the closed loop solution for $\mathbf{w}_t$. This further indicates that the structure (the centers and variances) of the new node and the weight vector can be optimized iteratively. We note that, in the iterative approach, (23) is performed at every iteration with the same input data but for the latest node structure. It is clear that the main computation cost in the weight adaptation comes from the matrix inversion $\mathbf{P}_t$. Fortunately, in the proposed scheme, only one insignificant node is replaced at a time without changing the model size. This means that only the last column of $\boldsymbol{\Phi}_t$ is changed at every iteration. Exploiting this property can significantly reduce the calculation of $\mathbf{P}_t$, which is accomplished by making use of the inverse of matrix block decomposition lemma to avoid the repetitive matrix inversions $\mathbf{P}_t$ at every iteration. To be specific, we can rewrite $\boldsymbol{\Phi}_t$ and $\mathbf{P}_t$ in the forms of

$$\boldsymbol{\Phi}_t = [\boldsymbol{\Phi}_{t,-M} \quad \mathbf{g}_M] \quad (24)$$

and

$$\mathbf{P}_t = \begin{bmatrix} \boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \boldsymbol{\Phi}_{t,-M} & \boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \mathbf{g}_M \\ \mathbf{g}_M^{\mathrm{T}} \boldsymbol{\Phi}_{t,-M} & \mathbf{g}_M^{\mathrm{T}} \mathbf{g}_M \end{bmatrix}^{-1} \quad (25)$$

respectively. Letting $\mathbf{P}_{t,-M} = (\boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \boldsymbol{\Phi}_{t,-M})^{-1}$, and applying the inverse of matrix block decomposition lemma, we obtain

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^{\mathrm{T}} & c \end{bmatrix} \quad (26)$$

where

$$\begin{cases} \mathbf{A} = \mathbf{P}_{t,-M} + \frac{1}{\kappa} \mathbf{P}_{t,-M} \boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \mathbf{g}_M \mathbf{g}_M^{\mathrm{T}} \boldsymbol{\Phi}_{t,-M} \mathbf{P}_{t,-M} \\ \mathbf{b} = -\frac{1}{\kappa} \mathbf{P}_{t,-M} \boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \mathbf{g}_M \\ c = \frac{1}{\kappa} \end{cases} \quad (27)$$

where $\kappa = \mathbf{g}_M^{\mathrm{T}} \mathbf{g}_M - \mathbf{g}_M^{\mathrm{T}} \boldsymbol{\Phi}_{t,-M} \mathbf{P}_{t,-M} \boldsymbol{\Phi}_{t,-M}^{\mathrm{T}} \mathbf{g}_M$. Because only the structure of the Mth node is adapted and the structures for all others nodes remain unchanged, for every node replacement,

$\mathbf{P}_{t,-M}$ needs only be calculated once with complexity in the order of $O((M-1)^3)$. On the other hand, $\mathbf{g}_M$ needs to be iteratively calculated for the node optimization, so the computation cost for the LS weight calculation is $O(p)$ scaled by the number of iterations.

C. Structure Initialization of the New Node

The convergence of the iterative node structure and the weight vector adaptation well depends on the initialization. It is important to initialize the centers and variances of the new node for the iterative adaption process.

1) Center Initialization: In the RBF structure, the nodes should well cover the input data to ensure good generalization. It is likely that the optimum centers are around the input data. There are several possible ways to initialize the centers of new node.

1) The simplest way is to initialize the centers as the current data
$$c_M^{(0)}(j) = x_t(j), \quad j = 1, \ldots, N_x \tag{28}$$
where superscript $^{(0)}$ is used to represent the initial iteration.
2) Since the new node optimization depends on the recent q input data, the centers can also be initialized as the center of the previous data input which is given by
$$c_M^{(0)}(j) = \frac{1}{q}\sum_{i=t-q+1}^{t} x_i(j), \quad j = 1, \ldots, N_x. \tag{29}$$
3) Alternatively a more robust way is to randomly initialize the centers based on the Gaussian process as
$$c_M^{(0)}(j) = N(m_t(j), s_t(j)), \quad j = 1, \ldots, N_x \tag{30}$$
where $N(m_t(j), \sigma_t(j))$ represents one realization of the Gaussian process with mean $m_t(j)$ and standard-deviation $s_t(j)$, $m_t(j)$ is center of the previous data input which is given by (29), and $s_t(j)$ is set as the standard-derivation of the samples as
$$s_t(j) = \sqrt{\frac{1}{q}\sum_{i=t-q+1}^{t} |x_i(j) - m_t(j)|^2}, \qquad j = 1, \ldots, N_x. \tag{31}$$

2) Variance Initialization: Once the centers of the new nodes are initialized as above, the initial standard-derivations are determined by how far the corresponding centers are away from the nearest center of other nodes. To be specific, if the jth center of the new node is initialized as $c_M^{(0)}(j)$, the corresponding standard-derivation is initialized as
$$\sigma_M^{(0)}(j) = \rho \cdot \left| c_M^{(0)}(j) - c_{\text{nearest}}(j) \right|, \quad j = 1, \ldots, N_x \tag{32}$$
where ρ is a constant scaling factor, and $c_{\text{nearest}}(j)$ is the jth center of the remaining nodes with nearest distance to $c_M^{(0)}(j)$. Finally we have
$$\eta_M^{(0)}(j) = \frac{1}{\left(\sigma_M^{(0)}(j)\right)^2}, \quad j = 1, \ldots, N_x. \tag{33}$$

IV. Algorithm Summary

The proposed approach operates at two modes: 1) weight adaptation mode and 2) node optimization mode. When the underlying system varies little, the RBF works in the weight adaptation mode, where the RBF structure remains fixed and the weight vector is adapted by the MRLS algorithm to track the variation of the input data. When the input data varies so large that the MRLS weight adaption fails to track, the RBF network switches to the node optimization mode, where an insignificant node is replaced by a new node without changing the model size. The structure of the new node and the weight vector are iteratively optimized by the gradient descent search and LS estimation, respectively. Because both the MRLS and the gradient descent search approaches are guaranteed to be stable, the convergence of the proposed algorithm (which switches between the two modes) is also stable.

Because now both the weight coefficients and node structure of the RBF structure are adapted on-line, the proposed scheme can well track the nonstationary processes with a very sparse model and yet maintains a low level of computation. The proposed algorithm is summarized in "Algorithm 1: fast tunable RBF networks."

The proposed approach requires fixing the model size M which depends on dynamic state of the system. In practice, some a priori knowledge of the system is usually available, and the model size is determined empirically. Or the model size can be set as a small value at the beginning and gradually increased until there is no significant performance gain observed. We point out that, the error functions e_t^2 is generally multimodel with respect to the structure vector. Rather than finding the global optimal with whole data set as in many off-line approaches, the proposed approach achieves the best possible solution within the sampling time. The proposed initialization method is also helpful for the new node structure from trapping in the same local minimum.

We also highlight that, the weight vector is updated by the MRLS based on the p recent errors in the weight adaptation mode, and by the LS estimation based on the q recent errors in the node optimization mode, where it is not necessary to have $p = q$. In both modes, we need to update P_t which is a $M \times M$ matrix, where M is the number of nodes in the RBF network. In order to achieve smooth transition from the node optimization mode to the weight adaptation mode, $\mathbf{P}_t$ from the last iteration in the node optimization is copied into the MRLS adaptation.

V. Simulations

In this section, computer simulations are given to compare the proposed algorithm with typical approaches including the RAN, GAP-RBF, and ELM algorithms. All these approaches (including the proposed one) apply Gaussian nodes. For fair comparison, in the RAN [9], GAP-RBF [10], and ELM algorithms [23], the controlling parameters are carefully chosen based on trial-and-error to achieve the best performance.

We consider two benchmark applications below: 1) ANC and 2) on-line time series prediction. The performance for different approaches are compared based on the

Algorithm 1: Fast tunable RBF networks

```
Initialization
    Initialize the structure of the RBF nodes
    Initialize the weight vector as $\mathbf{w}_0 = [0, \dots, 0]^T$
    Initialize $\mathbf{P}_0 = \delta \mathbf{I}$ for the MRLS, where $\delta$ is a large number.
For every observation pair $\{\mathbf{x}_t, y_t\}$, $t = 1, 2, 3, \dots$
    Form the input matrix $\mathbf{X}_t$ and information matrix $\mathbf{\Phi}_t$ as in (6) and (7) respectively.
    Obtain the error vector $\mathbf{e}_t$ and the average error power $\bar{e}_t^2$ in (8) and (12) respectively.
    If $\bar{e}_t^2 < \Delta_1$, **weight adaptation mode**
        Adapt the weight vector with the MRLS algorithm as in (8), (9), (10) and (11)
    Else if $\bar{e}_t^2 \geq \Delta_1$, **node optimization mode**
        Calculate the WNV for each node as in (14) and order them as $\mathrm{WNV}_{1'} \leq \cdots \leq \mathrm{WNV}_{M'}$
        Replace the node $1'$ with a new node.
        Initialize the centers and inverse of variances of the new node
        With the structure of other nodes unchanged, calculate $\mathbf{P}_{t,-M} = (\mathbf{\Phi}_{t,-M}^T \mathbf{\Phi}_{t,-M})^{-1}$
        For $l = 1, \dots, L$, iteratively adapt the node structure and weight vector
            **Given the current node structure, calculate the LS estimate of the weight vector**
                Update the information matrix as $\mathbf{\Phi}_t = [\mathbf{\Phi}_{t,-M} \quad \mathbf{g}_M]$
                Update $\mathbf{P}_t$ in a fast way as in (26).
                Obtain the LS estimation as $\mathbf{w}_t = \mathbf{P}_t \mathbf{\Phi}_t^T \mathbf{y}_t$
            **Given the current weight vector $\mathbf{w}_t$, adapt the structure of the new node**
                Calculate the gradient vector $\nabla$ as in (21).
                Adapt the structural vector as $\mathbf{\Gamma}_l = \mathbf{\Gamma}_{l-1} - \varepsilon \frac{\nabla}{\|\nabla\|} e_t$
                If a variance inverse $\eta(j) < 0$, then it is reset to a small positive value.
        If the network residual is small enough that $J_t / \| \mathbf{y}_t \|^2 \leq \Delta_2$, iteration stops.
        Else go to the next iteration.
    End of the iteration.
    **Copy $\mathbf{P}_t$ from the last iteration into the MRLS adaptation for the next input data.**
  End
```

mean square error (MSE) which, at time t, is defined as

$$\mathrm{MSE}(t) = \frac{1}{t} \sum_{i=1}^{t} (y_i - f(\mathbf{x}_i))^2. \tag{34}$$

In all simulations, the model size is set as $M = 10$, the iteration number $L = 10$, the step size in gradient descent search for the structure vector [defined in (19)] is set as $\varepsilon = 0.2$, the innovation length and forgetting factor for the MRLS weight adaptation are set as $p = 1$ and $\lambda = 0.98$, respectively. Other parameters are set individually for the ANC and time series prediction applications as will be shown later.

A. Adaptive Noise Cancellation

The ANC is used to extract signals buried in noise [24]. The ANC can be equivalent to the system identification shown in Fig. 2, where $x(t)$ is the regarded system input which is generated by feeding the uncorrelated noise $n(t)$ through the secondary channel with unknown transfer function $T(.)$, and

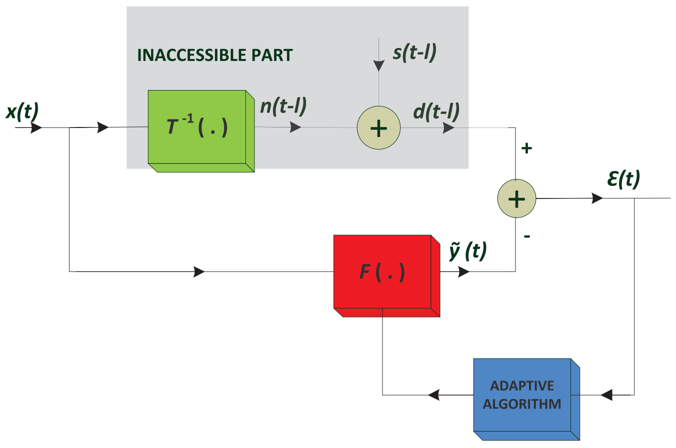


Fig. 2. System identification equivalent to the ANC.

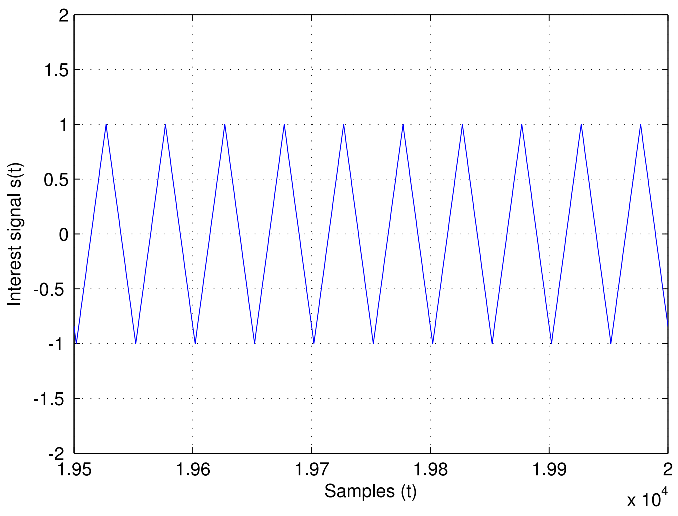


Fig. 3. Simulation A.1: desired signal $s(t)$.

$d(t-l)$ is the measured system output with a delay l which is set as $l = 1$ in this simulation. The task of ANC can be viewed as adjusting $F(.)$ to approximate the inverse of the unknown transfer function $T^{-1}(.)$. Because $T^{-1}(.)$ is normally highly nonlinear and may vary with time, Fig. 2 describes a typical on-line system identification problem in nonlinear nonstationary environment.

In this simulation, the nonlinear channel $T(.)$ is described as

$$x(t) = a_1 x(t-1) + a_2 x(t-2) + b_1 n(t-1) + b_2 n(t-2) + b_3 n(t-3) + c_1 n^2(t-2) + c_2 n(t-2) x(t-1) \tag{35}$$

where a_i, b_j, and c_k are some coefficients which will be set below. The modeling input vector to $F(.)$ is given by

$$\mathbf{u}(t) = \left[x(t), x(t-1), x(t-2), \hat{y}(t-1), \hat{y}(t-2)\right]^T. \tag{36}$$

The proposed algorithm is compared with the RAN and GAP-RBF algorithms, where the threshold to trigger node replacement is set as $\Delta_1 = 20$, the threshold to stop the iterative adaptation $\Delta_2 = 2$ and the LS length for the fast weight adaptation $q = 1000$.

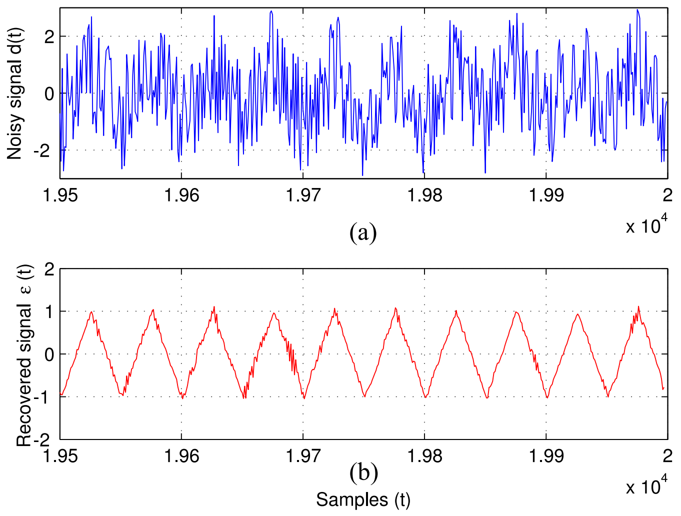


Fig. 4. Simulation A.1: noisy versus recovered signals in the stationary ANC model. (a) Corrupted signals. (b) Recovered signals.

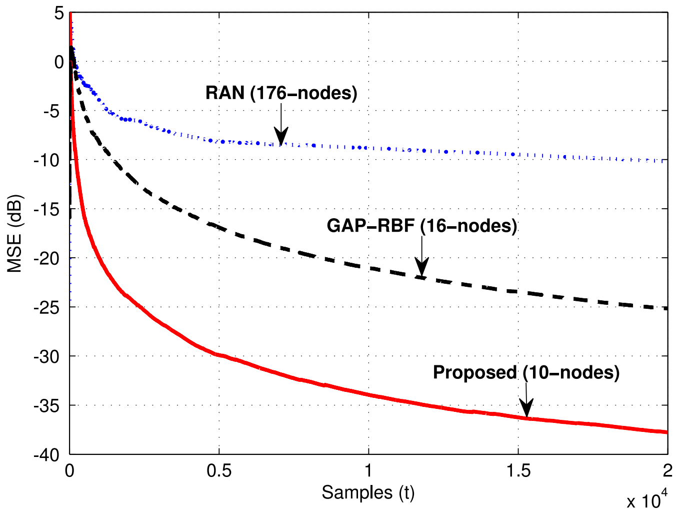


Fig. 5. Simulation A.1: MSE learning curves for the stationary ANC model.

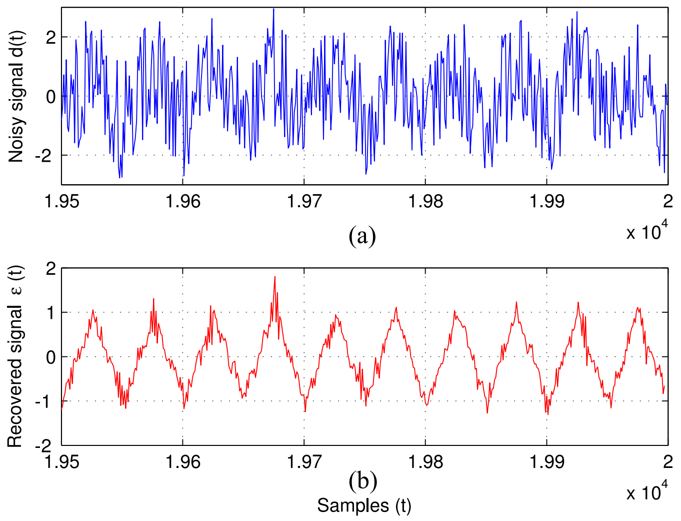


Fig. 6. Simulation A.2: noisy versus recovered signals in the ANC with time varying $T(.)$. (a) Corrupted signals. (b) Recovered signals.

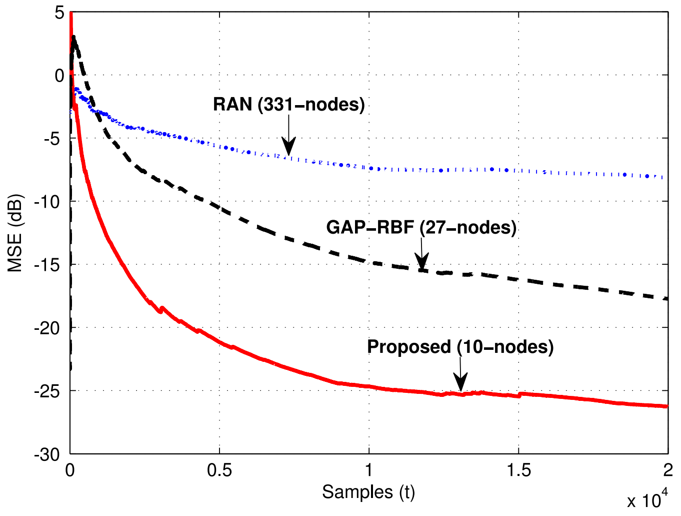


Fig. 7. Simulation A.2: MSE learning curves for the ANC with time varying $T(.)$.

1) Simulation A.1—Stationary ANC Model: First we consider a stationary ANC model, where the parameters of the nonlinear channel in (35) are fixed as $a_1 = 0.25$, $a_2 = 0.1$, $b_1 = 0.5$, $b_2 = 0.1$, $b_3 = -0.2$, $c_1 = 0.2$, and $c_2 = 0.08$.

The desired signal is a sawtooth signal of unit magnitude with period of 50 samples, as is shown in Fig. 3.

Fig. 4(a) and (b) shows the corrupted signal $d(t)$ and the recovered signal $\varepsilon(t)$ during the last 500 samples for the proposed approach, respectively, where it is clearly shown that the noise has been significantly canceled.

Fig. 5 compares the MSE learning curves for different approaches. The GAP-RBF has clearly better MSE performance than the RAN, and it also requires much fewer nodes. This is because the GAP-RBF can both grow and prune the model size, while the RAN can only increase the model. It is clear that the proposed approach has the best MSE performance with only ten nodes.

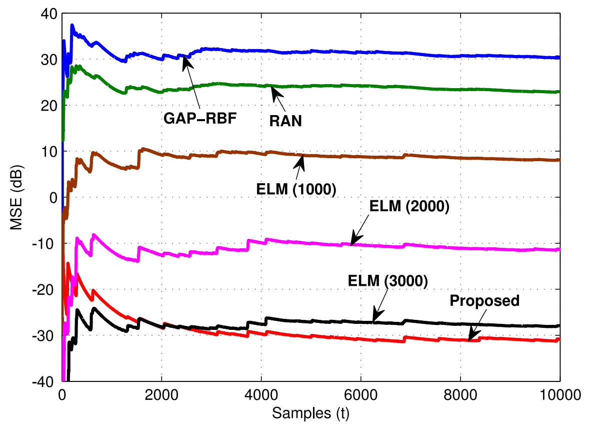
2) Simulation A.2—ANC with Time Varying T(.): In this simulation, we let the nonlinear channel $T(.)$ be vary with time so this becomes a nonstationary system modeling. To be specific, the first coefficient in (35) is set varying with time as

$$a_1 = 0.2 + \left|0.5 \cdot \sin\left(\frac{t}{1000\pi}\right)\right| \tag{37}$$

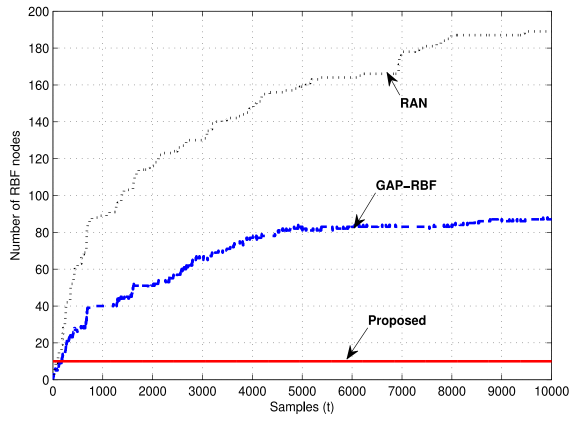
and other coefficients are the same as those in Simulation A.1. The desired signal is also the sawtooth signal as is shown in Fig. 3.

Fig. 6(a) and (b) shows the corrupted signal $d(t)$ and the recovered signal $\varepsilon(t)$ during the last 500 samples for the proposed approach, respectively. It is clearly shown that the signal can still be well extracted from buried noise when the noise model is time varying.

Fig. 7 compares the MSE leaning curves for different approaches. While the proposed approach has clearly the best performance, it is interesting to observe that both the RAN and GAP-RBF use significantly more nodes in this nonstationary case than in the stationary case, where particularly, the RAN algorithm ends up with a model size of 331.

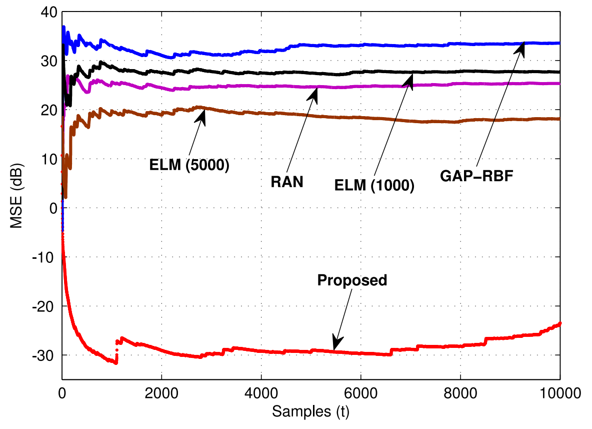


(a)

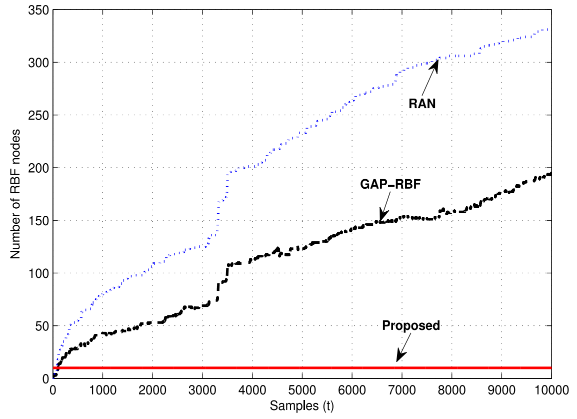


(b)

Fig. 8. Simulation B.1: Lorenz series with fixed parameters. (a) MSE learning curves. (b) Model size learning curves.



(a)



(b)

Fig. 9. Simulation B.2: Lorenz series with time varying parameters. (a) MSE learning curves. (b) Model size learning curves.

B. On-line Time Series Prediction

In the second experiment, we consider the on-line time series prediction, where the Lorenz chaotic time series is chosen as it is often used as a benchmark in many applications (see [25], [26]). As a 3-D and highly nonlinear system, the Lorenz system is governed by three differential equations as $dx(t)/dt = ay(t) - ax(t)$, $dy(t)/dt = cx(t) - x(t)z(t) - y(t)$, and $dz(t)/dt = x(t)y(t) - bz(t)$, where a, b, and c are the parameters that control the behavior of the Lorenz system. In the simulations, the fourth order Runge–Kutta approach with the step size of 0.01 is used to generate the Lorenz samples, and only the Y-dimension samples, $y(t)$, are used for the time series prediction. In all simulations, there are 15 000 data samples generated in $y(t)$. Because the Lorenz system is very sensitive to the initial condition, only the last 10 000 stable samples of $y(t)$ are used in the simulations.

We consider a 60-steps forward prediction, where the prediction task is to use the past four samples

$$\mathbf{x}_t = \left[y_t,\ y_{t-6},\ y_{t-12},\ y_{t-18}\right]^{\mathrm{T}} \tag{38}$$

to estimate the sample y_{t+60}.

In this simulation, the proposed algorithm is compared with the RAN, GAP-RBF, and ELM algorithms, where the threshold to trigger node replacement is set as $\Delta_1 = 10^{-4}$, the threshold to stop the iterative adaptation $\Delta_2 = 10^{-5}$, and the LS length for the fast weight adaptation $q = 10$.

1) Simulation B.1—Lorenz Time Series With Fixed Parameters: First we fix the parameters of the Lorenz system as $a = 10$, $b = 8/3$, and $c = 28$.

Fig. 8(a) and (b) shows the MSE and model size learning curves, respectively. It is shown that the RAN and GAP-RBF achieve comparable prediction performance. While the GAP-RBF has a more compact model than the RAN, its model size is still very large compared to the proposed approach. In the ELM approach, the model size is fixed at a large value. In Fig. 8(a), the MSE performance for the ELM with model sizes of 1000, 2000, and 3000 are all shown, which are denoted as ELM(1000), ELM(2000), and ELM(3000), respectively. It is clear that the MSE performance of the ELM improves with more nodes. While the ELM algorithm has significantly better performance than the RAN or GAP-RBF algorithm, it uses a lot more nodes. Among all of the approaches, the proposed algorithm with only ten nodes has the best performance. It is also shown in Fig. 8(a) that the ELM requires as many as 3000 nodes to achieve comparable prediction performance as the proposed approach.

2) Simulation B.2—Lorenz Time Series With Time Varying Parameters: In this simulation, we let the Lorenz parameters vary with time to obtain a nonstationary system. Specifically, we set $a = 10$ and

$$b = \frac{4 + 3(1 + \sin(0.1t))}{3} \tag{39}$$

$$c = 25 + 3\left(1 + \cos\left(2^{0.001t}\right)\right). \tag{40}$$

Fig. 9(a) and (b) compares the MSE and model size learning curves for different approaches, respectively. It is shown that the RAN performs better than the GAP-RBF, but it uses more nodes. As is shown in Fig. 9(b), for both RAN and GAP-RBF algorithms, the model sizes keep increasing with the number of data input. For the ELM approach, both the model sizes of 1000 and 5000 are tested. It is clear that the proposed algorithm is the only approach here that can well track this nonstationary Lorenz time series. Particularly, the performance of the ELM with model size as large as 5000 is still not comparable with the proposed approach which only use ten nodes.

VI. Conclusion

In this paper, we proposed a novel RBF structure with adaptive tunable nodes for on-line system identification problems in nonlinear and nonstationary environment. The model size of the RBF network is fixed at a small number in order to capture the local characteristic of the nonstationary systems. The weight vector is normally adapted by the MRLS algorithm while the modeling performance is monitored every time step. If the modeling residual becomes high, an insignificant node is replaced with a new node, of which the structural parameters and weights are optimized using proposed fast iterative algorithms including the gradient decent algorithm and LS estimator. The novelty in the proposed algorithm is that matrix algebra and model structural properties are exploited in order to achieve real time tracking capability. The proposed algorithm describes a novel on-line identification approach which is fundamentally different from existing approaches. The proposed approach is verified by numerical simulations on two benchmark applications: 1) ANC and 2) on-line Lorenz time series prediction. The results show that the proposed approach significantly outperforms existing approaches. This is because the proposed approach keeps a compact model size to capture the local statistics of the underlying system, while in existing approaches the model size is often too large for time varying system modeling. Finally, we point out that, although the proposed on-line modeling approaches use tunable Gaussian RBF network with fixed model size, it can be easily extended to many other associative networks with linear-in-the-parameter structure, e.g., thin-plate-spline and B-spline networks. In the future, we are interested in improving the flexibility of the model. This can be achieved by adding a simultaneous model size adaptation mechanism or using multiple kernel models.

References

[1] K. F. K. Wong, A. Galka, O. Yamashita, and T. Ozaki, "Modelling nonstationary variance in EEG time series by state space GARCH model," *Comput. Biol. Med.*, vol. 36, no. 12, pp. 1327–1335, Dec. 2006.

[2] L. Rutkowski, "Generalized regression neural networks in time-varying environment," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 576–596, May 2004.

[3] L. Rutkowski, "Adaptive probabilistic neural networks for pattern classification in time-varying environment," *IEEE Trans. Neural Netw.*, vol. 15, no. 4, pp. 811–827, Jul. 2004.

[4] Q. Zhou, P. Shi, S. Xu, and H. Li, "Observer-based adaptive neural network control for nonlinear stochastic systems with time delay," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 1, pp. 71–80, Jan. 2013.

[5] Q. Meng, B. Li, H. Holstein, and Y. Liu, "Parameterization of point-cloud freeform surfaces using adaptive sequential learning RBF networks," *Pattern Recognition*, vol. 46, no. 8, pp. 2361–2375, 2013.

[6] J. Qiao, Z. Zhang, and Y. Bo, "An online self-adaptive modular neural network for time-varying systems," *Neurocomputing*, vol. 125, pp. 7–16, Feb. 2014.

[7] M. Gan, H. Li, and H. Peng, "A variable projection approach for efficient estimation of RBF-ARX model," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 476–485, Mar. 2015.

[8] F. Ding, P. X. Liu, and G. Liu, "Multi-innovation least-squares identification for system modeling," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 40, no. 3, pp. 767–778, Jun. 2010.

[9] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, Jun. 1991.

[10] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 34, no. 6, pp. 2284–2292, Dec. 2004.

[11] S. Ferrari, F. Bellocchio, V. Piuri, and N. Alberto Borghese, "A hierarchical RBF online learning algorithm for real-time 3-D scanner," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 275–285, Feb. 2010.

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

[13] C. F. Fung, S. A. Billings, and W. Luo, "On-line supervised adaptive training using radial basis function networks," *Neural Netw.*, vol. 9, no. 9, pp. 1597–1617, Dec. 1996.

[14] S. Chen, K. Labib, and L. Hanzo, "Clustering-based symmetric radial basis function beamforming," *IEEE Signal Process. Lett.*, vol. 14, no. 9, pp. 589–592, Sep. 2007.

[15] D. S. Yeung, W. W. Y. Ng, D. Wang, E. C. C. Tsang, and X.-Z. Wang, "Localized generalization error model and its application to architecture selection for radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp. 1294–1305, Sep. 2007.

[16] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[17] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modeling using orthogonal forward regression with PRESS statistic and regularization," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 898–911, Apr. 2004.

[18] S. Chen, X. Hong, and C. J. Harris, "Particle swarm optimization aided orthogonal forward regression for unified data modelling," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 477–499, Aug. 2010.

[19] H. Chen, Y. Gong, and X. Hong, "Online modeling with tunable RBF network," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 935–947, Jun. 2013.

[20] P. A. Gutiérrez, C. Hervas-Martinez, and F. J. Martínez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 246–263, Feb. 2011.

[21] N. E. Huang *et al.*, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proc. Roy. Soc. London A Math. Phys. Eng. Sci.*, vol. 454, no. 1971, pp. 903–995, Mar. 1998.

[22] F. Ding and T. Chen, "Performance analysis of multi-innovation gradient type identification methods," *Automatica*, vol. 43, no. 1, pp. 1–14, Jan. 2007.

[23] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Jul. 2006.

[24] S. Billings and C. Fling, "Recurrent radial basis function networks for adaptive noise cancellation," *Neural Netw.*, vol. 8, no. 2, pp. 273–290, 1995.

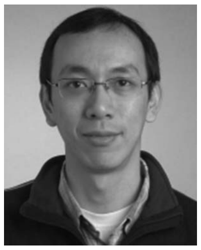
[25] E. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, Mar. 1963.

[26] A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

Hao Chen received the B.Eng. degree in automatic control from the National University of Defense Technology, Changsha, China, in 2006, the M.Sc. degree in control systems with distinction from the University of Sheffield, Sheffield, U.K., in 2009, and the Ph.D. degree in cybernetics from the School of Systems Engineering, University of Reading, Reading, U.K., in 2014, sponsored by the Engineering and Physical Sciences Research Council and Defence Science and Technology Laboratory from the British Government.

From 2014 to 2015, he was a Post-Doctoral Research Fellow with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada, and Syncrude Canada Ltd., Fort McMurray, AB, Canada. He is currently an Associate Professor with Haixi Institutes, Chinese Academy of Sciences, Jinjiang, China. His current research interest include online learning, soft sensors, system identification, neural networks, machine learning, and signal processing.

Dr. Chen was a recipient of the Chinese Government Award for Outstanding Self-financed Students Abroad in 2012.



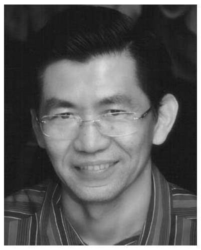
Yu Gong received the B.E. and M.E. degrees from the University of Electronics and Science Technology of China, Chengdu, China, in 1992 and 1995, respectively, and the Ph.D. degree in communications from the National University of Singapore, Singapore, in 2002, all in electronic engineering.

He took several research positions with the Institute of Infocomm Research, Singapore, and the Queens University of Belfast, Belfast, U.K., respectively. From 2006 to 2012, he was an Academic Member with the School of Systems Engineering, University of Reading, Reading, U.K. Since 2012, he has been with the School of Electronic, Electrical, and Systems Engineering, Loughborough University, Loughborough, U.K. His current research interests include signal processing and communications, such as wireless communications, cooperative networks, nonlinear and nonstationary system identification, and adaptive filters.

Xia Hong (SM'02) received the B.Sc. and M.Sc. degrees from the National University of Defense Technology, Changsha, China, in 1984 and 1987, respectively, and the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 1998, all in automatic control.

She was a Research Assistant with the Beijing Institute of Systems Engineering, Beijing, China, from 1987 to 1993. From 1997 to 2001, she was a Research Fellow with the Department of Electronics and Computer Science, University of Southampton, Southampton, U.K. She is currently a Professor with the School of Systems Engineering, University of Reading, Reading, U.K. She has published over 150 research papers, and coauthored a research book. Her current research interests include nonlinear systems identification, data modeling, estimation, and intelligent control, neural networks, pattern recognition, and learning theory and their applications.

Prof. Hong was a recipient of the Donald Julius Groen Prize by IMechE in 1999.



Sheng Chen (M'90–SM'97–F'08) received the B.E. degree in control engineering from East China Petroleum Institute, Dongying, China, in 1982, the Ph.D. degree in control engineering from City University, London, U.K., in 1986, and the higher doctoral and Doctor of Sciences degrees from the University of Southampton, Southampton, U.K., in 2005.

From 1986 to 1999, he held a research and academic appointments with the University of Sheffield, Sheffield, U.K., the University of Edinburgh, Edinburgh, U.K. and the University of Portsmouth, Portsmouth, U.K. Since 1999, he has been with the Department of Electronics and Computer Science, University of Southampton, Southampton, U.K., where he is currently a Professor of Intelligent Systems and Signal Processing. He has published over 550 research papers. He is a Distinguished Adjunct Professor with King Abdulaziz University, Jeddah, Saudi Arabia. His current research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, intelligent control system design, and evolutionary computation methods and optimization.

Dr. Chen is an ISI highly cited researcher in engineering in 2004. He is a fellow of IET. He was elected to a fellow of the U.K. Royal Academy of Engineering in 2014.