A Fast Adaptive Tunable RBF Network For Non-stationary Systems

Hao Chen, *Student Member, IEEE,* Yu Gong, *Member, IEEE*, 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 aiming at non-stationary systems. Based on a RBF network with individually tunable nodes and a fixed small model size, the weight vector is adjusted using the multi-innovation recursive least square (MRLS) 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 modelling 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 non-stationary systems in particular.

Index Terms

Radial basis function (RBF), on-line identification, non-stationary, non-linear, mulit-innovation RLS

I. Introduction

On-line system identification in non-stationary environment is usually a difficult task. A common approach is to use adaptive algorithms to track the time variation of the systems, where typical examples can be found in the time-varying autoregressive-moving average with exogenous terms (TV-ARMAX) [1] and time-varying autoregressive with exogenous terms (TV-ARX) [2] for the linear approaches, and time-varying neural network (TV-NN) [3] for the non-linear approaches. In some cases, the associated time-varying parameters of a non-stationary system can be expanded by a series of basis functions, and the non-stationary modeling is simplified to the time-invariant parameter estimation. For instance, Legendre and Walsh basis functions are used for smooth and abrupt changing

non-stationary signals respectively [4]. However, such approaches are for specific model structures based on some prior knowledge of the systems, which is clearly not suitable for all non-stationary systems.

It is well understood that a large class of non-linear systems can be modeled with a linear-in-the-parameter model such as the popular radial basis function (RBF) neural network. With a cluster of non-linear “kernels” (or nodes) imposed on the input data, the RBF network can approximate any continuous function to an arbitrary degree of accuracy [5]. The RBF tracks the dynamic of the systems by constantly updating the output layer weights with adaptive algorithms such as the least mean square (LMS), givens least square (GLS) and recursive least square (RLS) [6]–[8]. Recently a variant RLS, referred to as the multi-innovation RLS (MRLS) [9]–[11] has been proposed. 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 structure of the network. The RBF structure is determined by multiple parameters including the number of nodes (or the model size), and the centres 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 implementation, it is however not sufficient to track non-stationary systems. This can be seen that, in a simple example, the nodes of a RBF network are centered around some observed data inputs. If the input data are so dynamic that there are no nodes with centres near the coming input data, the RBF is not able to cover (or track) the input data variation no matter how the weights are adjusted. This clearly suggests that the structure of the RBF network must be carefully chosen especially in non-stationary systems.

A popular approach is to apply the on-line system identification where the size of the RBF model is able to grow or prune based on the incoming data. An early development of this kind is the resource allocating network (RAN) [12], where the network model starts from empty and grows with the input data based on the *nearest neighbour* method. The RAN extended Kalman filter (RAN-EKF) algorithm improves the RAN by replacing the LMS with the extended Kalman filter to adjust the network parameters [13]. Both the RAN and RAN-EKF algorithms only grow the network size without a pruning strategy to remove the obsolete RBF nodes, so the model size can be too large in some applications. In [14], an improved approach of the RAN algorithm was proposed by limiting the size of the RBF network (L-RAN). Further in [15], [16], a more compact model can be achieved by the minimal RAN (M-RAN) algorithm which can prune the inactive kernel nodes based on relative contribution. All of these algorithms, however, need to pre-determine many controlling parameters which have significant influence on the modeling accuracy and speed. More computational efficient growing-and-pruning RBF (GAP-RBF) algorithm [17]

and the generalized GAP-RBF (GGAP-RBF) algorithm [18] were then proposed, in which only the nearest RBF node is considered for model growing and pruning. The growing and pruning are based on the “significance” of the nodes which has direct link to the learning accuracy, and require some a prior information such as the input data range and distribution. In order to guarantee the model generalization, the kernel least mean square (KLMS) algorithm was proposed in [19], in which the size of the model can grow based on the *well-posedness* analysis in reproducing kernel Hilbert spaces (RKHS).

Alternatively, the on-line structure adjustment may be avoided by using a large RBF model size covering a wide dynamic range of data input. Typical examples are the extreme learning machine (ELM) and its variant [20]–[24], 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 non-stationary systems and the model generalization is not guaranteed.

A common drawback in both the RAN-like and ELM-like approaches is that none of them optimizes the structure of the RBF nodes. In fact, in most existing RBF networks, the node centres are either determined by the input data such as in the *k*-means approach ([25]) or simply set as the input data (e.g. [19]), and often a common variance is used for all nodes which is set by the trial-and-error or cross-validation method [26]. 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 non-stationary environment [27]. Therefore, it is highly necessary to optimize the node structure which is in general a hard NP problem, where complicated optimization approaches such as the generic algorithms are often used. For example, a tunable RBF model identification algorithm was recently proposed in [28], [29], where each RBF node has a tunable centre vector and an adjustable diagonal covariance matrix. At each forward regression stage, the associated centre vector and diagonal covariance matrix are optimized using a particle swarm optimization (PSO) algorithm. 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 applications in non-stationary systems. Moreover, the PSO algorithm itself has high complexity, further deterring its use in many practical on-line applications where the computation cost is a key issue.

It is inspired by the tunable RBF model identification algorithm in [28], [29] that, if the structure (i.e. the centres and variances) of the nodes are optimized, an efficient RBF network with much smaller model size than a conventional RBF network can be constructed. A main contribution of this paper is, therefore, to propose a fast algorithm (other than the PSO or generic searching algorithms) to optimize the node structure on-line. To be specific, we propose to fix the RBF model size at a small number, where each node has a tunable centre vector

and an adjustable diagonal covariance matrix which can be optimized on-line one at a time. At each time step, the node weights are adapted by the MRLS algorithm [11], and the modeling performance is monitored. If the RBF network performs poorly despite of the weight adaptation, an insignificant node with little contribution to the overall performance is replaced with a new node without changing the model size. The structure of the new node is optimized to "fit" for the recent input data. Fast algorithms for the node optimization are also proposed in this paper to keep the computation cost low, making it particularly suitable for on-line applications.

Fundamentally different from existing approaches, the proposed solution in this paper describes a fast tunable RBF network for on-line system identification. In the simulation part, the proposed approach is verified on two benchmark applications: adaptive noise cancellation and on-line Lorenze time series prediction. Experimental results show that the proposed algorithm has significantly better performance with a very sparse model than existing approaches, yet keeps low complexity. Finally we point out that, while the proposed approach is for the RBF network, it can be easily extended to other kernel-based approaches.

The rest of this paper is organized as follows: Section II describes a novel 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. The 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

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

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 $\mathbf{w}_{t-1} = [\omega_{t-1}(1), \cdots, \omega_{t-1}(M)]^{\mathrm{T}}$ be the weight vector and $\boldsymbol{\phi}_t = [g_1(\mathbf{x}_t), \cdots, g_M(\mathbf{x}_t)]^{\mathrm{T}}$ be the information vector, we can rewrite (2) as

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

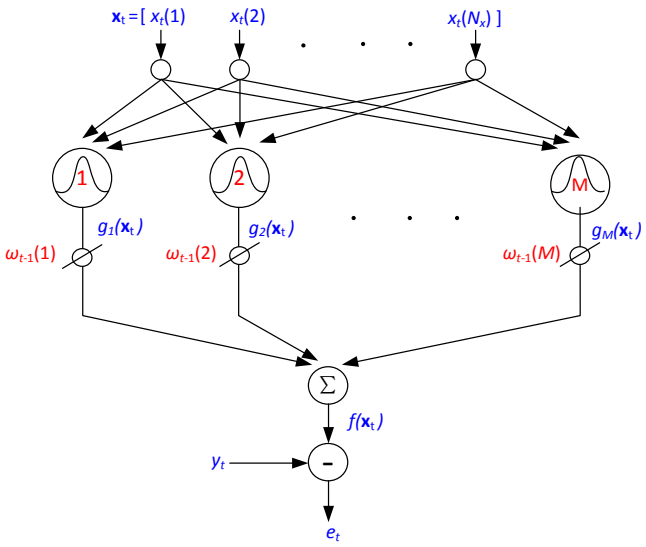


Fig. 1. The RBF neural network

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

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

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

In a non-stationary system, both the weight vector and the structure of the RBF network should be updated on-line. For the Gaussian RBF network, the node structure parameters include N_x pairs of centres and variances of each node, or $c_i(j)$ and $\sigma_i^2(j)$ in (5) respectively. Because the input statistics keeps varying in a non-stationary system, the "local characteristic" of the input data is often more important than the "global characteristic" [30], [31]. 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 centre 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

In this paper, the MRLS algorithm is used to update the weight vector $\mathbf{w}$. Originally described for the ARX model adaptation [9], the MRLS adaptation is based on both current and past residual errors. To be specific, putting p number of input vectors into an input matrix gives

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

where p is the innovation length which determines the number of past errors used for weight adaptation. If $p = 1$, the MRLS reduces to the classic RLS algorithm.

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

$$\begin{aligned} \mathbf{\Phi}_t &= \begin{bmatrix} g_1(\mathbf{x}_t) & g_2(\mathbf{x}_t) & \dots & g_M(\mathbf{x}_t) \\ g_1(\mathbf{x}_{t-1}) & g_2(\mathbf{x}_{t-1}) & \dots & g_M(\mathbf{x}_{t-1}) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\mathbf{x}_{t-p+1}) & g_2(\mathbf{x}_{t-p+1}) & \dots & g_M(\mathbf{x}_{t-p+1}) \end{bmatrix} \\ &= [\boldsymbol{\phi}_t, \boldsymbol{\phi}_{t-1}, \cdots, \boldsymbol{\phi}_{t-p+1}]^{\mathrm{T}} \\ &= [\mathbf{g}_1,\ \mathbf{g}_2, \cdots, \mathbf{g}_M] \in \mathbb{R}^{p \times M} \end{aligned} \tag{7}$$

where $\mathbf{g}_i = [g_i(\mathbf{x}_t), g_i(\mathbf{x}_{t-1}), \cdots, g_i(\mathbf{x}_{t-p+1})]^{\mathrm{T}}$ which is the ith RBF regressor. Letting $\mathbf{e}_t = [e_t, e_{t-1}, \cdots, e_{t-p+1}]^{\mathrm{T}}$ and $\mathbf{y}_t = [y_t, y_{t-1}, \cdots, y_{t-p+1}]^{\mathrm{T}}$, 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.

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

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

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

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 vecto 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

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

where Δ_1 is a constant threshold which is set according to the performance requirement. In general, the smaller the Δ_1 is, the smaller the residual error can achieve, but the more frequently the node replacement may occur.

When $\bar{e}_t \geq \Delta_1$, ideally, the "most" insignificant node with least contribution to the overall system performance is replaced with a new node. While the individual contribution of each node to the overall system can be revealed by its corresponding increment of error variance (IEV) [32], [33], the calculation of IEV suffers from high complexity. Alternatively, we can use the weighted node-output variance (WNV) to determine which nodes are "insignificant". The WNV for the ith node is defined as

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

Note $\mathbf{g}_i$, which is defined in (7), contains p number of recent output 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, despite of the weight adaptation, an insignificant node with little contribution to the overall system 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 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 modelling 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 centre and the inverse of the variance for the new node are tuned by minimizing the 2-norm of the instantaneous error. The instantaneous error at time t can be expressed as

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

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

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

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

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

The objective is to find the optimum Γ which minimizes e_t^2. With the weight vector being fixed, the optimum Γ can be found by the gradient descent search as

$$\Gamma_l = \Gamma_{l-1} - \varepsilon \frac{\boldsymbol{\nabla}}{\|\boldsymbol{\nabla}\|} e_t, \tag{19}$$

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

$$\boldsymbol{\nabla} = \frac{\partial(e_t)}{\partial \Gamma} = \left[\frac{\partial e_t}{\partial c_M(1)}, \cdots, \frac{\partial e_t}{\partial c_M(N_x)}, \frac{\partial e_t}{\partial \eta_M(1)}, \cdots \frac{\partial e_t}{\partial \eta_M(N_x)} \right]^{\mathrm{T}} \tag{20}$$

From (17), we can easily obtain that

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

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

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

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

$$\mathbf{w}_t = \mathbf{P}_t\boldsymbol{\Phi}_t^{\mathrm{T}}\mathbf{y}_t, \tag{23}$$

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 centres 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 re-write $\mathbf{\Phi}_t$ and $\mathbf{P}_t$ in the forms of

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

and

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

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

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

where

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

where $\kappa = \mathbf{g}_M^{\mathrm{T}}\mathbf{g}_M - \mathbf{g}_M^{\mathrm{T}}\mathbf{\Phi}_{t,-M}\mathbf{P}_{t,-M}\mathbf{\Phi}_{t,-M}^{\mathrm{T}}\mathbf{g}_M$. Because now only the structure of the Mth node is adapted and the structures for all others nodes remain unchanged, $\mathbf{P}_{t,-M}$ needs only be calculated once for every node replacement. Note that $\mathbf{g}_M$ 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. There is also an one off computational cost to calculate $\mathbf{P}_{t,-M}$ which is in the order of $O((M-1)^3)$.

C. Structure initialization of the new node

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

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

- The simplest way is to initialize the centres as the current data:

$$c_M^{(0)}(j) = x_t(j), \quad j = 1, \cdots, N_x \tag{28}$$

where superscript $^{(0)}$ is used to represent the initial iteration.

- Since the new node optimization depends on the recent q input data, the centres can also be initialized as the centre of the previous data input which is given by

$$c_M^{(0)}(j) = \frac{1}{q} \sum_{i=t-q+1}^{t} x_i(j), \quad j = 1, \cdots, N_x \tag{29}$$

- Alternatively a more robust way is to randomly initialize the centres based on the Gaussian process as

$$c_M^{(0)}(j) = N(m_t(j), s_t(j)), \quad j = 1, \cdots, N_x, \tag{30}$$

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 centre of the previous data input which is given by (29), and $s_t(j)$ is set as the standard-derivation of the samples as

$$s_t(j) = \sqrt{\frac{1}{q} \sum_{i=t-q+1}^{t} |x_i(j) - m_t(j)|^2}, \qquad j = 1, \cdots, N_x \tag{31}$$

2) Variance initialization: Once the centres of the new nodes are initialized as above, the initial standard-derivations are determined by how far the corresponding centres are away from the nearest centre of other nodes. To be specific, if the the jth centre of the new node is initialized as $c_M^{(0)}(j)$, the corresponding standard-derivation is initialized as

$$\sigma_M^{(0)}(j) = \rho \cdot |c_M^{(0)}(j) - c_{\text{nearest}}(j)|, \quad j = 1, \cdots, N_x \tag{32}$$

where ρ is a constant scaling factor, and $c_{\text{nearest}}(j)$ is the jth centre of the remaining nodes with nearest distance to $c_M^{(0)}(j)$. Finally we have

$$\eta_M^{(0)}(j) = \frac{1}{\left(\sigma_M^{(0)}(j)\right)^2}, \quad j = 1, \cdots, N_x \tag{33}$$

We note that, while there exist other ways of initialization, it is not the focus of this paper.

IV. The algorithm summary

The proposed approach operates at two modes: *weight adaptation mode* and *node optimization mode*. When the underling 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 now both the weight coefficients and node structure of the RBF structure are adapted on-line, the proposed scheme can well track the non-stationary processes with a very sparse model and yet maintains a low level of computation. The proposed algorithm is summarized as below.

Initialization

Initialize the structure of the RBF nodes

Initialize the weight vector as $\mathbf{w}_0 = [0, \cdots, 0]^{\mathrm{T}}$
Initialize $\mathbf{P}_0 = \delta \mathbf{I}$ for the MRLS, where δ is a large number.

For every observation pair $\{\mathbf{x}_t, y_t\}$, $t = 1, 2, 3, \cdots$

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 centres and inverse of variances of the new node

With the structure of other nodes unchanged, calculate $\mathbf{P}_{t,-M} = \left(\mathbf{\Phi}_{t,-M}^{\mathrm{T}} \mathbf{\Phi}_{t,-M}\right)^{-1}$

For $l = 1, \cdots, L$, 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 $\mathbf{w}_t = \mathbf{P}_t \mathbf{\Phi}_t^{\mathrm{T}} \mathbf{y}_t$

Given the current weight vector $\mathbf{w}_t$, adapt the structure of the new node

Calculate the gradient vector $\boldsymbol{\nabla}$ as in (21).
Adapt the structural vector as $\Gamma_l = \Gamma_{l-1} - \varepsilon \frac{\boldsymbol{\nabla}}{\|\boldsymbol{\nabla}\|} e_t$
If a variance inverse $\eta(j) < 0$, then it is reset to a small positive value.

If the network residual is small enough that $J_t / \parallel \mathbf{y}_t \parallel^2 \leq \Delta_2$, 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

At the beginning, the structure of the RBF needs to be initialized as, for example, simply around the input data. While the initial structure is not usually optimum, it will be optimized on-line at a later time.

We 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 should be copied into the MRLS adaptation.

V. Simulations

In this section, computer simulations are given to compare the proposed algorithm with some 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 [12], GAP-RBF [17] and ELM algorithms [20], the controlling parameters are carefully chosen based on trial-and-error to achieve best performance.

We consider two benchmark applications below: *adaptive noise cancellation (ANC)* and *on-line time series prediction*. The performance for different approaches are compared based on the mean square error (MSE) which, at time t, is defined as

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

We note that, unlike another commonly used definition of the MSE which is obtained by averaging over many independent runs, the MSE defined in (34) is the average in time based on one typical run.

A. Adaptive noise cancellation

As a benchmark application, the adaptive noise cancellation (ANC) is used to extract signals buried in noise [34]. The ANC model is shown in Fig. 2, where the desired signal $s(t)$ is corrupted by a uncorrelated noise $n(t)$ to give the measured signal $d(t)$ in the primary channel, and the reference noise $x(t)$ is generated by feeding $n(t)$ through a secondary channel with unknown transfer function $T(.)$. The task of ANC is to adaptively adjust $F(.)$, based on the system residual $\varepsilon(t)$, to cancel the noise embedded in $d(t)$.

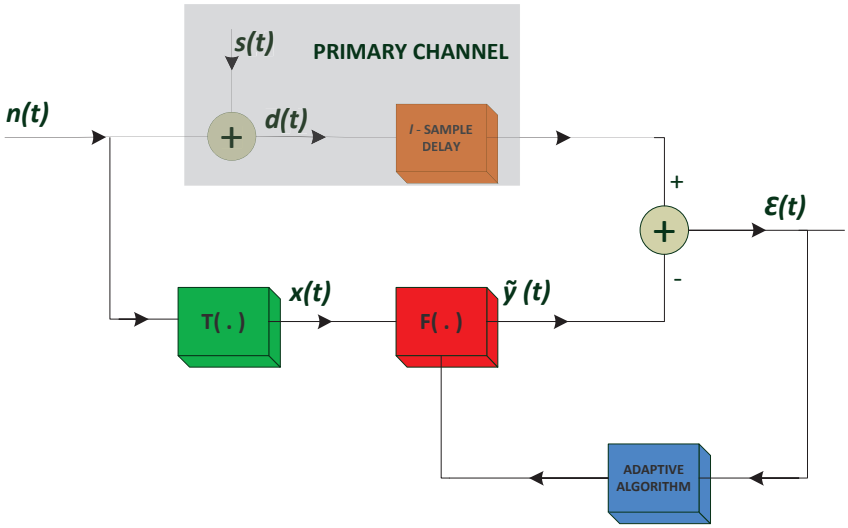


Fig. 2. Typical model of the ANC.

The ANC is equivalent to a system identification problem as is shown in Fig. 3, where $x(t)$ is regarded as the system input, $d(t-l)$ is the measured system output with a delay l which is set as $l=1$ below. The problem can now be viewed as adjusting $F(.)$ to approximate $T^{-1}(.)$ which is normally highly non-linear. While either $T^{-1}(.)$ or $s(t)$ may vary with time, Fig. 2 describes a typical on-line system identification problem in non-linear non-stationary environment.

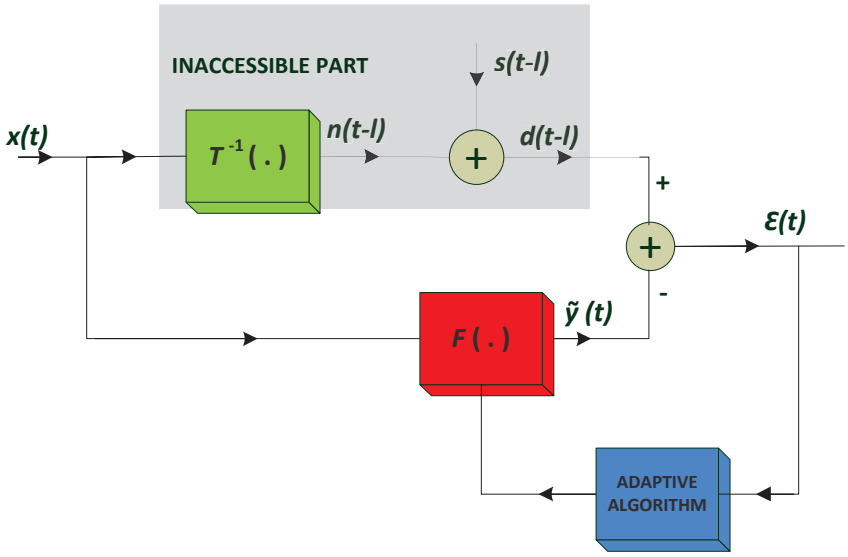


Fig. 3. System identification equivalent to the ANC.

In this simulation, the non-linear channel $T(.)$ is described as

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

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

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

In this experiment, the proposed algorithm is compared with the RAN and GAP-RBF algorithms. In the proposed approach, the number of nodes is fixed at 10, the innovation length is simply set as $p = 1$ so that the MRLS reduces to a classic RLS algorithm.

1) **Simulation A.1** - *Stationary ANC model:* First we consider a stationary ANC model, where the parameters of the non-linear 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. 4.

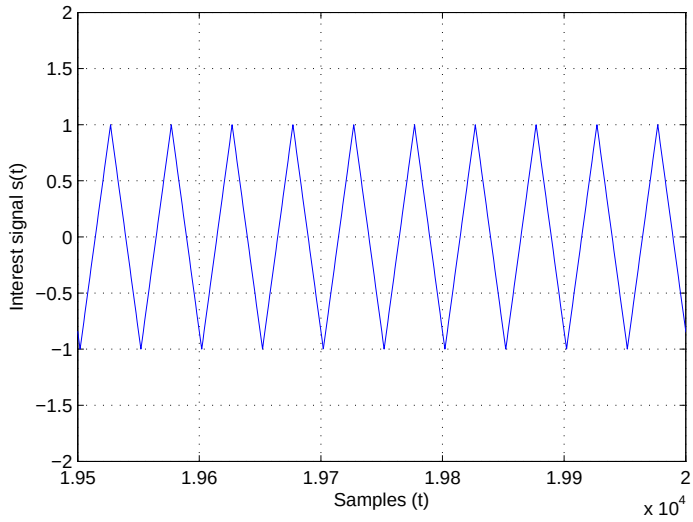


Fig. 4. **Simulation A.1**: desired signal $s(t)$.

Fig. 5 (a) and (b) show 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 cancelled.

Fig. 6 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 10 nodes.

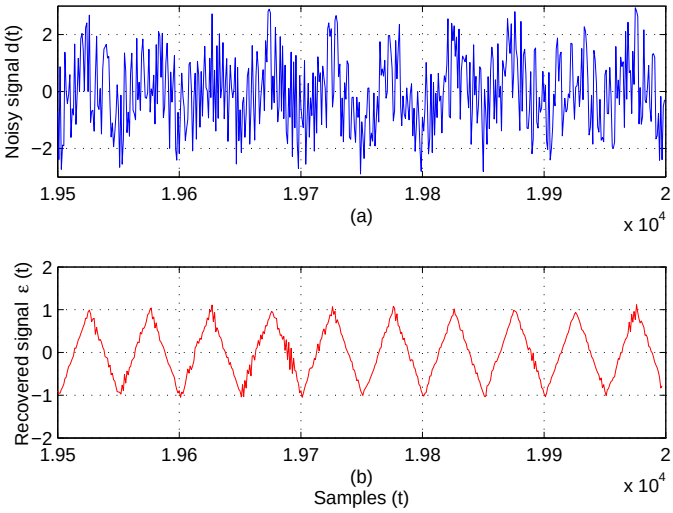


Fig. 5. **Simulation A.1**: Noisy *vs* recovered signals in the stationary ANC model.

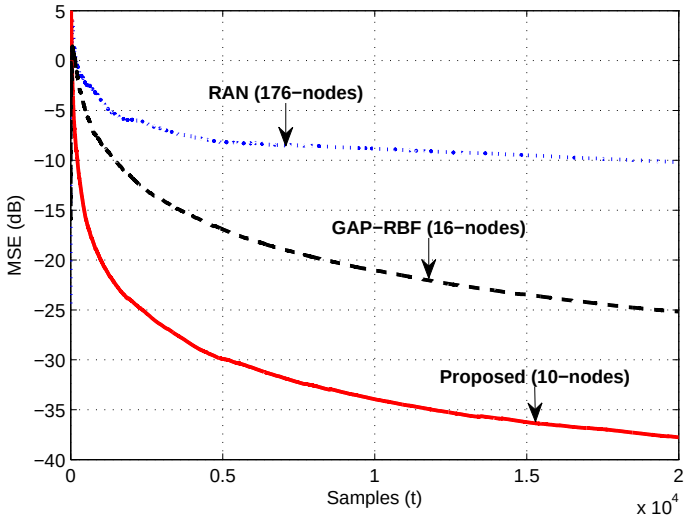


Fig. 6. **Simulation A.1**: MSE learning curves for the stationary ANC model.

2) **Simulation A.2** - *ANC with time varying* $T(.)$*:* In this simulation, we let the non-linear channel $T(.)$ be vary with time. 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. 4.

Fig. 7 (a) and (b) show 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

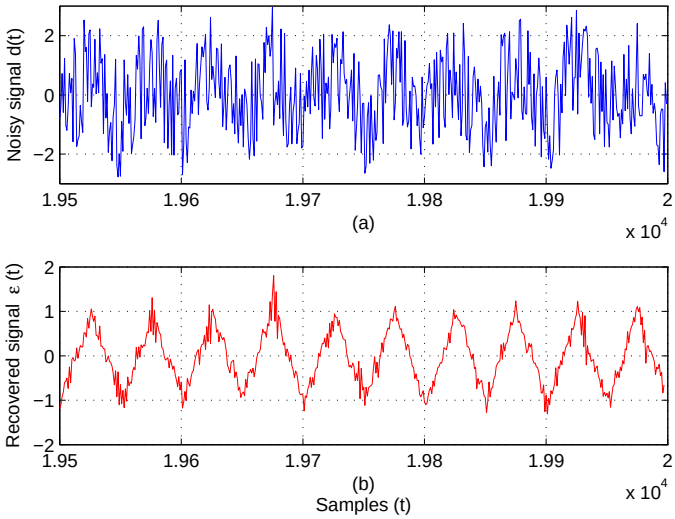


Fig. 7. **Simulation A.2**: Noisy *vs* recovered signals in the ANC with time varying $T(.)$.

when the noise model is time varying.

Fig. 8 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 non-stationary case than in the stationary case, where particularly, the RAN algorithm ends up with a model size of 331.

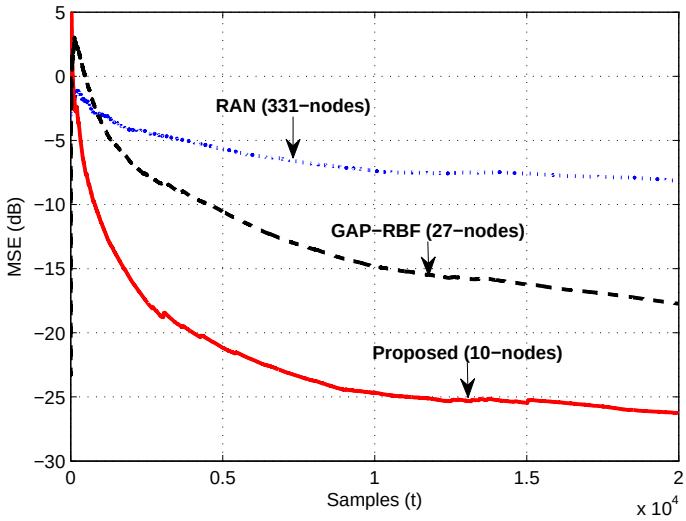


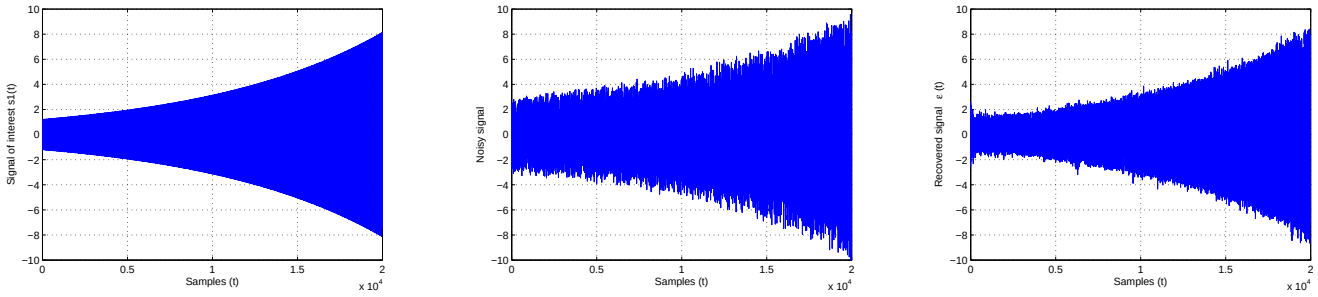
Fig. 8. **Simulation A.2**: MSE learning curves for the ANC with time varying $T(.)$.

3) **Simulation A.3** - *ANC with time varying $s(t)$:* In this simulation, we consider another non-stationary case. The desired signal is now varying with time as

$$s(t) = 1.1^{0.001t} \cdot sawtooth(t), \tag{38}$$

where $sawtooth(t)$ is the sawtooth signal shown in Fig. 4. The non-linear channel $T(.)$ is fixed as same as that in Simulation A.1.

Fig. 9 (a), (b) and (c) show the original desired signal $s(t)$, the noisy signal $d(t)$ and the recovered signal $\varepsilon(t)$ for the proposed approach respectively. While this clearly describes a highly non-stationary system, the effect of the noise cancellation can be observed by comparing the envelops for different signals.



(a) Desired signal $s(t)$ (b) Noisy signal $d(t)$ (c) Recovered signal $\varepsilon(t)$

Fig. 9. **Simulation A.3**: Signals in the ANC model with time varying $s(t)$.

Fig. 10 (a) and (b) show the noisy signal $d(t)$ and the recovered signal $\varepsilon(t)$ during the last 500 samples for the proposed approach respectively, where the effect of the noise cancellation is clearly seen. We note that, since the noise power remains unchanged, the signal-to-noise-ratio (SNR) increases with $s(t)$ as well. This is the reason that the corrupted signal looks less noisy in Fig. 10 (a) than that in Fig. 5 (a) or Fig. 7 (a).

Fig. 11 (a) and (b) compare the MSE and model size learning curves for different approaches respectively. It is shown again that the proposed approach has obviously the best MSE performance. The GAP-RBF performs significantly better than the RAN, while the RAN effectively fails to track the system. It is interesting to observe in Fig. 11 (b) that, without a scheme to take out obsolete nodes, the RAN increases its model size to as large as over 7,000. While the GAP-RBF maintains a relatively compact model, its model size is still much larger than that in the proposed approach which only uses 10 nodes.

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 [35]. As a three dimensional and highly non-linear

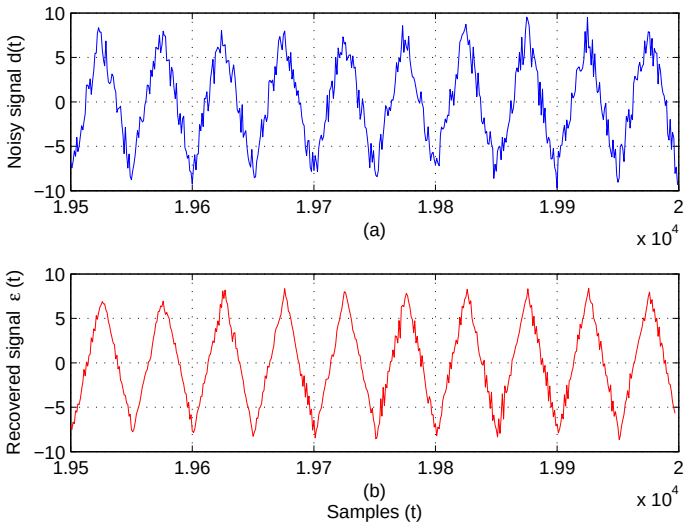
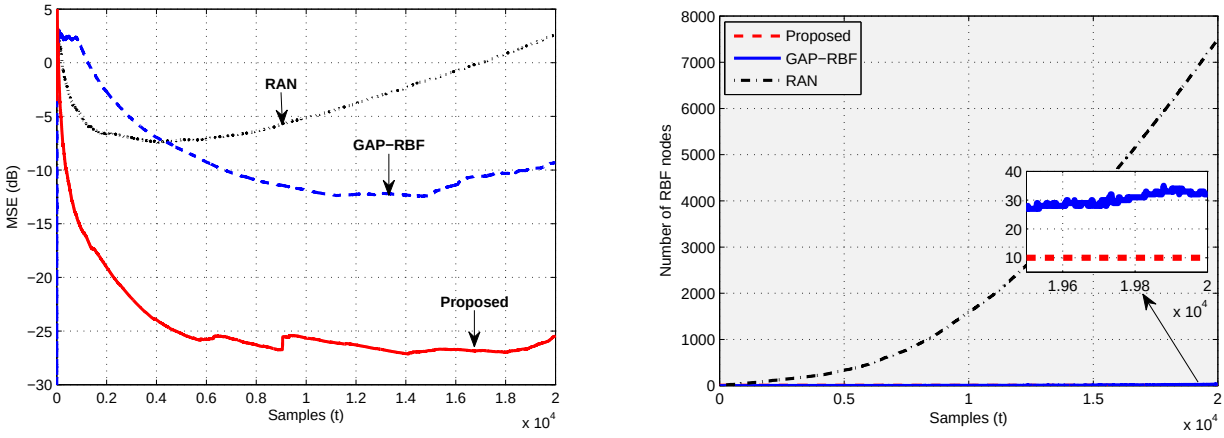


Fig. 10. **Simulation A.3**: Noisy *vs* recovered signals in the ANC with time varying $s(t)$.



(a) MSE learning curves

(b) Model size learning curves

Fig. 11. **Simulation A.3**: ANC with time varying $s(t)$.

system, the Lorenz system is governed by three differential equations as

$$\frac{dx(t)}{dt} = ay(t) - ax(t)$$
$$\frac{dy(t)}{dt} = cx(t) - x(t)z(t) - y(t)$$
$$\frac{dz(t)}{dt} = x(t)y(t) - bz(t)$$

where a, b and c are parameters that control the behavior of the Lorenz system. In the simulations, the 4th 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

$$\mathbf{x}_t = [y_t,\ y_{t-6},\ y_{t-12},\ y_{t-18}]^{\mathrm{T}} \tag{39}$$

to estimate the sample y_{t+60}.

In this experiment, the number of nodes is fixed at 5, and the innovation length is also set as $p = 1$ for the proposed approach. The proposed algorithm is compared with the RAN, the GAP-RBF and the ELM algorithms. We note that, while both off-line and on-line ELM have been proposed, it is the off-line ELM that is used below, in which the whole batch of data is taken in at the beginning. This is because that, on the one hand, it is hard to tune the on-line ELM to give satisfactory performance, and on the other hand, even the off-line ELM (which has better performance than its on-line counterpart) does not perform as well as the proposed approach. Therefore, it is reasonable to use the off-line ELM for the comparison purpose.

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 $c = 28$, where the trajectory of Lorenz system is shown in Fig. 12.

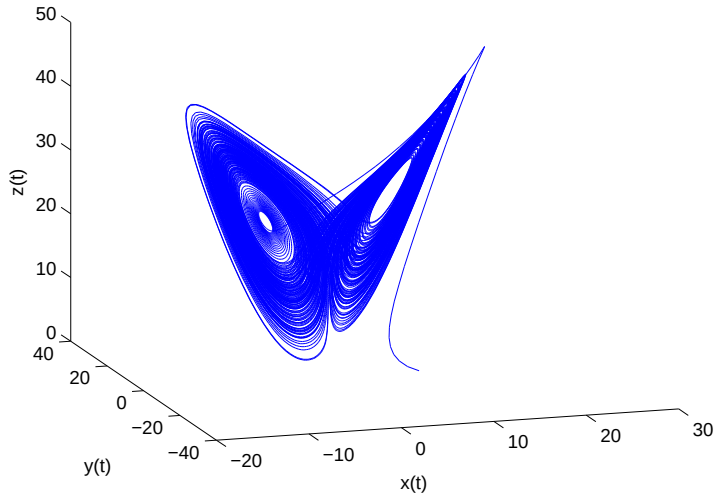
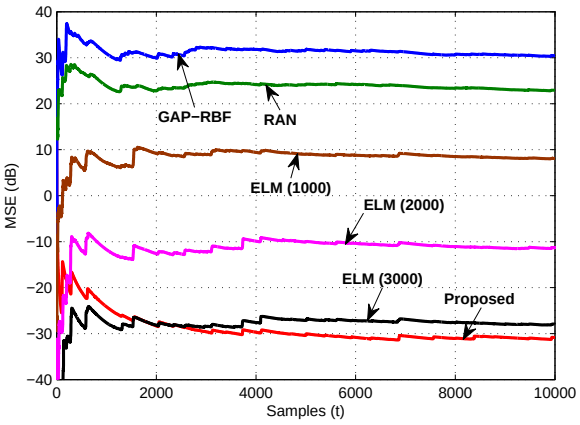
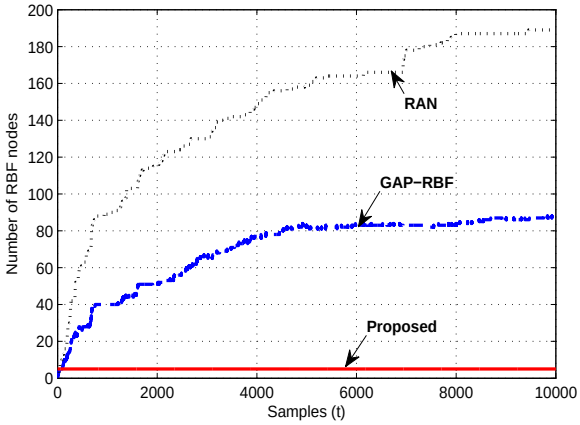


Fig. 12. **Simulation B.1**: Lorenz Attractor.

Fig. 13 (a) and (b) show 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. 13 (a), the MSE performance for the ELM with model sizes of 1000, 2000 and



(a) MSE learning curves



(b) Model size learning curves

Fig. 13. **Simulation B.1**: Lorenz series with fixed parameters.

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 5 nodes has the best performance. It is also shown in Fig. 13 (a) that the ELM requires as many as 3000 nodes to achieve comparable prediction performance as the proposed approach.

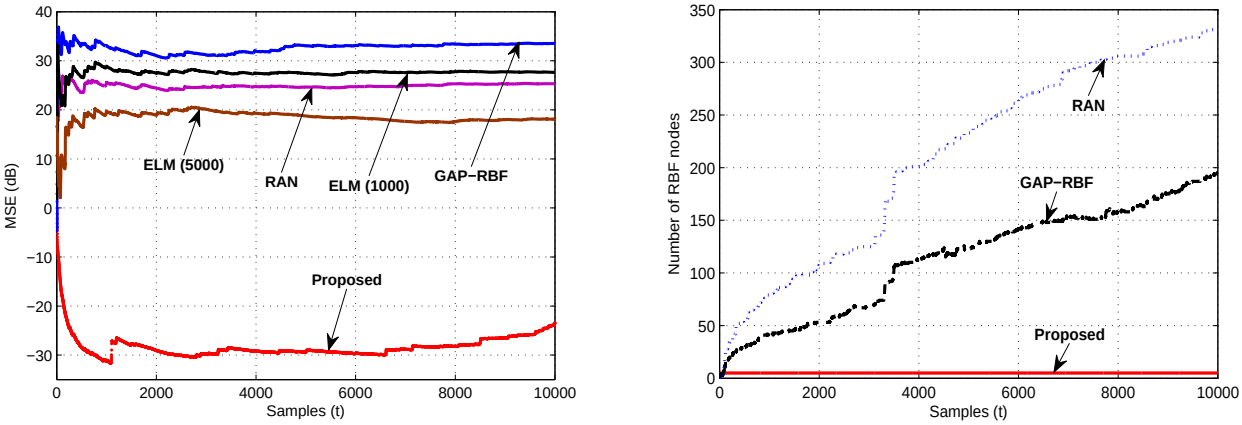
2) **Simulation B.2** - *Lorenz time series with time varying parameters:* In this simulation, we let the Lorenz parameters vary with time to obtain a non-stationary system. Specifically, we set $a = 10$ and

$$b = \frac{4 + 3(1 + \sin(0.1t))}{3} \tag{40}$$

$$c = 25 + 3(1 + \cos(2^{0.001t})) \tag{41}$$

Fig. 14 (a) and (b) compare 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. 14 (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, which are denoted as ELM(1000) and ELM(5000) respectively. It is clear that the proposed algorithm is the only approach here that can well track this non-stationary Lorenz time series. Particularly, the performance of the ELM with model size as large as $5,000$ is still not comparable with the proposed approach which only use 5 nodes.

3) **Simulation B.3** - *Lorenz time series with time base drift:* In this simulation, the parameters of the Lorenz systems are fixed as $a = 10$, $b = 8/3$, $c = 28$, but the samples of $y(t)$ are weighted by an exponential time based



(a) MSE learning curves

(b) Model size learning curves

Fig. 14. **Simulation B.2**: Lorenz series with time varying parameters.

drift to obtain

$$y_1(t) = 1.1^{0.01t} \cdot y(t), \tag{42}$$

where $y_1(t)$ is used for the time series prediction. It is clear that $y_1(t)$ is even more non-stationary than the time series in Simulation B.2.

Fig. 15 shows how the proposed algorithm predicts $y_1(t)$ on-line, where it is clear that, even with this very high non-stationary time series, the proposed approach is still able to track it very well.

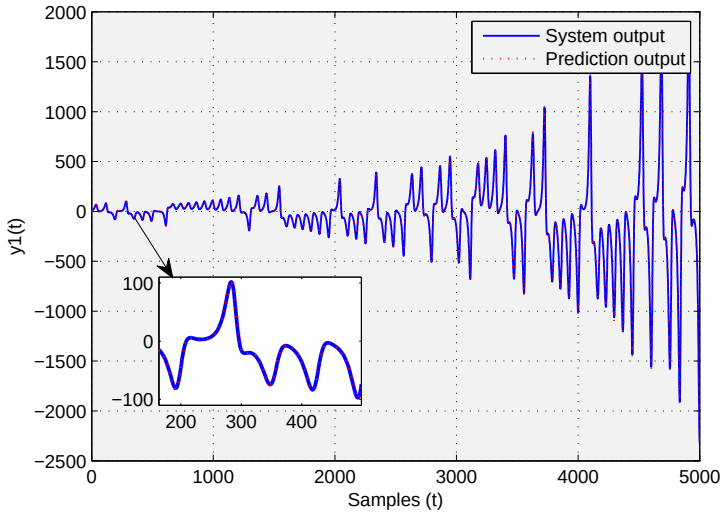
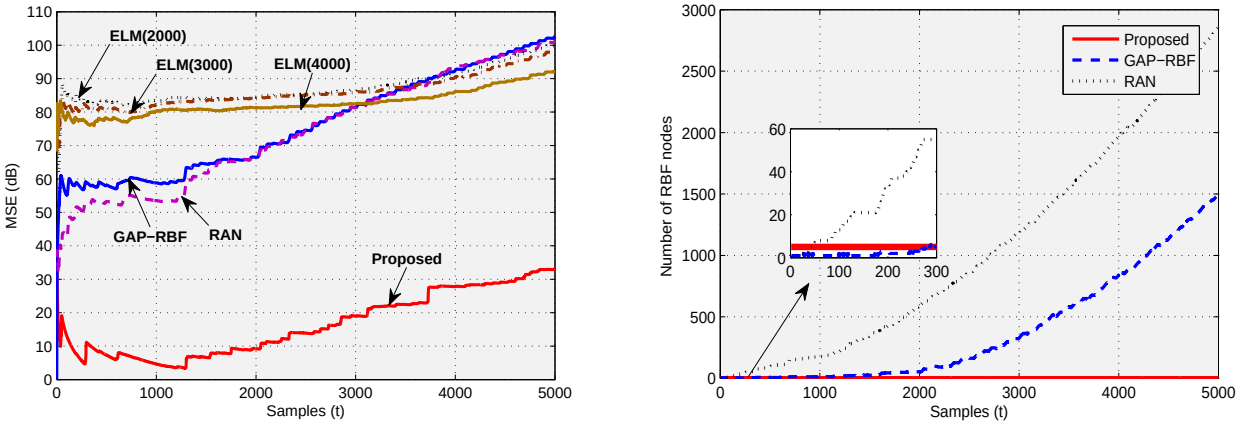


Fig. 15. **Simulation B.3**: on-line prediction by the proposed algorithm.

Fig. 16 (a) and (b) show the MSE and model size learning curves for different approaches respectively. Among all of the approaches, only the proposed can well track this highly non-stationary Lorenz time series. The RAN and



(a) MSE learning curves

(b) Model size learning curves

Fig. 16. **Simulation B.3**: Lorenz series with time base drift.

GAP-RBF have similar performance, and both have model size going up with the input data to nearly 3000 and 1500 respectively. For the ELM approach, we let the model sizes be at 2000, 3000 and 4000 respectively, similarly denoted as above. It is shown that, no matter how large the model size, the performance of the ELM is always far from satisfactory.

VI. Conclusion

In this article, we proposed a novel RBF structure with adaptive tunable nodes for on-line system identification problems in non-linear and non-stationary environment. The model size of the RBF network is fixed at a small number in order to capture the local characteristic of the non-stationary systems. The weight vector is normally adapted by the MRLS algorithm while the modelling performance is monitored every time step. If the modelling 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: adaptive noise cancellation and on-line Lorenze time series prediction. The results show that the proposed approach significantly outperforms existing approaches.

References

[1] M. Tsatsanis and G. Giannakis, “Time-varying system identification and model validation using wavelets,” *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3512 –3523, Dec. 1993.

[2] H. P., T. O., Y. T., H. S., K. N., V. H.O., and M. M., "RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process," *Control Engineering Practice*, vol. 12, no. 2, pp. 191 – 203, Feb. 2004.

[3] M. Iatrou, T. Berger, and V. Marmarelis, "Modeling of nonlinear nonstationary dynamic systems with a novel class of artificial neural networks," *Neural Networks, IEEE Transactions on*, vol. 10, no. 2, pp. 327 –339, Mar. 1999.

[4] Y. Zhong, K. Jan, K. Ju, and K. Chon, "Representation of time-varying nonlinear systems with time-varying principal dynamic modes," *Biomedical Engineering, IEEE Transactions on*, vol. 54, no. 11, pp. 1983 –1992, Nov. 2007.

[5] J. Park and I. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, Summer 1991.

[6] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, Summer 1989.

[7] S. Chen, S. A. Billings, and P. M. Grant, "Recursive hybrid algorithm for non-linear system identification using radial basis function networks," *International Journal of Control*, vol. 55, no. 5, pp. 1051–1070, May. 1992.

[8] M. B., "A fully kalman-trained radial basis function network for nonlinear speech modeling," in *Neural Networks, 1995. Proceedings, IEEE International Conference on*, vol. 1, Nov./Dec. 1995, pp. 259 –264.

[9] F. Ding, Y. Shi, and T. Chen, "A new identification algorithm for multi-input ARX systems," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 2, Jul. 2005, pp. 764 – 769.

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

[11] F. Ding, P. Liu, and G. Liu, "Multiinnovation least-squares identification for system modeling," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, no. 3, pp. 767 –778, Jun. 2010.

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

[13] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Computation*, vol. 5, no. 6, pp. 954–975, Nov. 1993.

[14] C. Molina and M. Niranjan, "Pruning with replacement on limited resource allocating networks by F-projections," *Neural Computation*, vol. 8, no. 4, pp. 855–868, May. 1996.

[15] Y. Lu, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation*, vol. 9, no. 2, pp. 461–478, Feb. 1997.

[16] ——, "Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm," *Neural Networks, IEEE Transactions on*, vol. 9, no. 2, pp. 308 –318, Mar. 1998.

[17] G. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 6, pp. 2284 –2292, Dec. 2004.

[18] ——, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *Neural Networks, IEEE Transactions on*, vol. 16, no. 1, pp. 57 –67, Jan. 2005.

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

[20] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489 – 501, Jul. 2006.

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

[22] G. Huang, L. Chen, and C. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,” *Neural Networks, IEEE Transactions on*, vol. 17, no. 4, pp. 879 – 892, Jul. 2006.

[23] G. Feng, G. Huang, Q. Lin, and R. Gay, “Error minimized extreme learning machine with growth of hidden nodes and incremental learning,” *Neural Networks, IEEE Transactions on*, vol. 20, no. 8, pp. 1352 –1357, Aug. 2009.

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

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

[26] S. Chen, X. Hong, C. Harris, and P. Sharkey, “Sparse modeling using orthogonal forward regression with press statistic and regularization,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 2, pp. 898 – 911, Apr. 2004.

[27] J. Kivinen, A. Smola, and R. Williamson, “Online learning with kernels,” *Signal Processing, IEEE Transactions on*, vol. 52, no. 8, pp. 2165 – 2176, Aug. 2004.

[28] S. Chen, X. Hong, B. L. Luk, and C. Harris, “A tunable radial basis function model for nonlinear system identification using particle swarm optimisation,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, Dec. 2009, pp. 6762–6767.

[29] S. Chen, X. Hong, and C. J. Harris, “Particle swarm optimization aided orthogonal forward regression for unified data modelling,” *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 4, pp. 477–499, Aug. 2010.

[30] N. Huang, Z. Shen, S. Long, M. Wu, H. Shih, Q. Zheng, N. Yen, C. Tung, and H. Liu, “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis,” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, Mar. 1998.

[31] J. D. Farmer and J. Sidorowich, “Predicting chaotic time series,” *Phys. Rev. Lett.*, vol. 59, no. 8, pp. 845–848, Aug. 1987.

[32] X. Hong and S. Billings, “Givens rotation based fast backward elimination algorithm for RBF neural network pruning,” *Control Theory and Applications, IEE Proceedings* -, vol. 144, no. 5, pp. 381 –384, Sep. 1997.

[33] X. Hong, C. Harris, M. Brown, and S. Chen, “Backward elimination methods for associative memory network pruning,” *International Journal of Hybrid Intelligent Systems*, vol. 1, no. 1-2, pp. 90–98, Apr. 2004.

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

[35] E. Lorenz, “Deterministic nonperiodic flow,” *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130 – 141, Mar. 1963.