On-line Modeling With Tunable RBF Network

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

***Abstract*— In this paper, we propose a novel on-line modeling algorithm for non-linear and non-stationary systems using a radial basis function (RBF) neural network with a fixed number of hidden nodes. Each of the RBF basis functions has a tunable centre vector and an adjustable diagonal covariance matrix. A multi-innovation recursive least square (MRLS) algorithm is applied to update the weights of RBF on-line while the modeling performance is monitored. When the modeling residual of the RBF network becomes large in spite of the weight adaptation, a node identified as insignificant is replaced with a new node, for which the tunable centre vector and diagonal covariance matrix are optimized using the quantum particle swarm optimization (QPSO) algorithm. The major contribution is to combine the MRLS weight adaptation and QPSO node structure optimization in an innovative way so that it can well track the local characteristic in the non-stationary system with a very sparse model. Simulation results show that the proposed algorithm has significantly better performance than existing approaches.**

***Index Terms*— radial basis function (RBF), on-line modeling, non-stationary, non-linear, multi-innovation recursive least square (MRLS), quantum particle swarm optimization (QPSO).**

I. Introduction

Conventional dynamical modeling is based on assumptions on linearity and stationarity of the underlying systems [1], [2]. In practice many systems exhibit non-linear and non-stationary behaviors, for which adaptive non-linear model is often needed. Unlike off-line modeling methods utilizing the whole batch of data, the on-line approach keeps adjusting the model using the incoming data so that the changing behavior of the non-stationary system is captured by the model. On-line modeling for non-stationary and non-linear systems is usually a difficult task. A common approach is to use adaptive algorithms to track the temporal variation of the system. Both linear and non-linear adaptive approaches have been proposed, with typical examples including the time-varying autoregressive-moving average with exogenous terms (TV-ARMAX) [3] and time-varying autoregressive with exogenous terms (TV-ARX) [4] for the linear approaches, and time-varying neural network (TV-NN) [5] 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 [6]. However, such approaches are for specific model structures based on some a prior knowledge of the systems, which is clearly not suitable for all non-stationary systems in practice.

A large class of non-linear systems can be modeled using the linear-in-the-parameter model. A popular choice of such models is the radial basis function (RBF) neural network due to it simplicity and the ability to approximate any continuous function to an arbitrary degree of accuracy [7]. In the on-line modeling, the weights of the RBF network are adapted by linear learning algorithms such as the least mean square (LMS) [8], givens least square (GLS) [9] and recursive least square (RLS) algorithms [10]. Recently a variant RLS algorithm, namely the multi-innovation RLS (MRLS) [11]–[13], 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, making it particularly robust against noise.

The on-line modeling RBF approaches have been well researched. This includes the resource allocating network (RAN) [14], where the network model starts from empty and grows with the input data based on the *nearest neighbor* 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 [15]. Both 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. Hence in [16], an improved approach of the RAN algorithm was proposed by limiting the size of the RBF network (L-RAN). Further in [17], [18], a more compact model can be achieved by using the minimal RAN (M-RAN) algorithm which prunes the inactive kernel nodes based on relative contribution. All of these algorithms need to carefully pre-determine many controlling parameters in order to achieve satisfactory performance. More computationally efficient growing-and-pruning RBF (GAP-RBF) algorithm [19] and the generalized GAP-RBF (GGAP-RBF) algorithm [20] were then proposed, in which only the nearest RBF node is considered for the model growing and pruning. The growing and pruning are based on the "significance" of the nodes which has a direct link to the learning accuracy, and require some a prior information such as the input data range and distribution. In order to

Manuscript received December 15, 2011; revised June 5, 2012 ; accepted September 1, 2012. This research is sponsored by the UK Engineering and Physical Sciences Research Council and DSTL under the grant number EP/H012516/1.

Hao Chen and Xia Hong are with the School of Systems Engineering, University of Reading, UK (E-mail: hao.chen@pgr.reading.ac.uk, x.hong@reading.ac.uk).

Yu Gong is with the School of Electronic, Electrical and Systems Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK, E-mail: alex.yugong@gmail.com.

guarantee the model generalization, the kernel least mean square (KLMS) algorithm was proposed in [21], in which the size of the model can grow based on the *well-posedness* analysis in reproducing kernel Hilbert spaces (RKHS). While all of these RAN-based approaches can identify a system model on-line with adjustable number of nodes (or the so-called model size), a common problem is that the structure of each individual node is not optimized. Rather, the node structure is simply set based on the incoming data (or the data itself). This makes the model size often increase with the number of the sample data, ending up with a very large model having poor model generalization and high computational expense, particularly so for non-stationary systems [22]. The RBF node selection and structure optimization are thus of particular importance.

In fact, a popular approach for structuring the RBF network model is to consider the training input data points as candidate RBF centres and to employ a common variance for every RBF node. In this approach, the node selection can be performed using the orthogonal least squares (OLS) algorithm and its variants including the Regularized-OLS (ROLS), Locally-ROLS (LROLS), LROLS-leave-one-out (LROLS-LOO) [23]–[27]. Recently a tunable RBF model identification algorithm was proposed [28], [29], where each RBF node has a tunable centre vector and an adjustable diagonal covariance matrix, and at each forward regression stage, the assocated centre vector and diagonal covariance matrix are optimized using particle swarm optimization (PSO) algorithm. This provides an exceptionally flexible RBF model in which the model size can be significantly reduced. The PSO is a population based stochastic optimisation technique inspired by social behaviour of bird flocking or fish schooling [30]. The PSO method is becoming very popular due to its simplicity in implementation, ability to quickly converge to a reasonably good solution and its ability to escape from local minima. It has been applied to a wide range of optimisation problems successfully [31], [32]. While the aforementioned approaches provide an powerful way to automatically determine the model structure of the RBF network, they are however off-line (batch) learning methods which are inadequate for on-line applications. This motivates us to propose a novel on-line RBF network in which the node structure can be adaptively optimized.

An interesting alternative to the above approaches is the recently proposed extreme learning machine (ELM) and its variant [33]–[39], where the node structure optimization are avoided by using a very large number of nodes. Both off-line and on-line ELM approaches have been proposed. In the off-line ELM [33], [34], [39], a large number of nodes are randomly generated at the beginning and fixed during the learning process. While in the on-line approach [35]–[38], a relatively smaller number of nodes are randomly applied at the *training* stage, but the node number can be adjusted during the learning stage depending on the incoming data. The on-line version can achieve similar performance as the off-line approach, but has less complexity as it does not deal with the whole batch of data [38]. While the ELM can achieve high accuracy with fast learning speed in many applications, the model size may have to be very large especially for non-stationary systems so that the model generalization is not guaranteed. In this paper, we focus on RBF networks whose structure can be adaptively optimized, as it is particularly suitable in non-stationary environments.

In this paper, we propose a novel on-line RBF network with tunable nodes. First we understand that, in the non-stationary system, because the input statistics keeps varying, the "local characteristic" of the input data is more relevant than the "global characteristic" [40], [41]. This implies that the model size needs not to be large since the modeling needs to focus on the recent data, but not the older ones. Therefore, we propose to fix the model size at a small number, and each RBF 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 RBF weights are adapted using the multi-innovation RLS (MRLS) [13], and the modeling performance is monitored. If the RBF network performs poorly despite the weight adaptation, an insignificant node with little contribution to the overall system is identified and replaced by a new node without changing the model size. The structural parameters of the new node including the centre vector and diagonal covariance matrix are optimized by using the quantum swarm optimization (QPSO) algorithm. Unlike the original PSO algorithm [30], the QPSO does not pre-specify a searching boundary and can ensure convergence to the global minimum [42], making it particularly suitable for the node structure optimization. Because the RBF network has tunable nodes, the model size can be much smaller than that of a conventional RBF network due to its structural flexibility. The main contributions of this paper are summarized as follows:

- Propose a novel on-line RBF network which is fundamentally different from existing approaches. It has a fixed model size but tunable node structure. Simulation results show that the proposed algorithm with a very sparse model has significantly better performance than existing approaches especially in non-stationary environments.
- Propose to use the QPSO algorithm for the node structure optimization on-line.
- Integrate seamlessly the MRLS weight adaptation and QPSO node optimization into one approach.

The rest of this paper is organized as follows. Section II briefly introduces the RBF neural network and describes the multi-innovation RLS algorithm for the weight adaptation; Section III proposes a novel approach to optimize the node structure on-line; Section IV summarizes the proposed algorithm; Section V compares the proposed approach with some typical existing on-line methods via numerical simulations; finally, Section VI concludes the paper.

II. RBF NETWORK WITH MULTI-INNOVATION RLS ADAPTATION

The adaptive RBF network is shown in Fig. 1, where there are M hidden nodes, or the model size is M. 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.

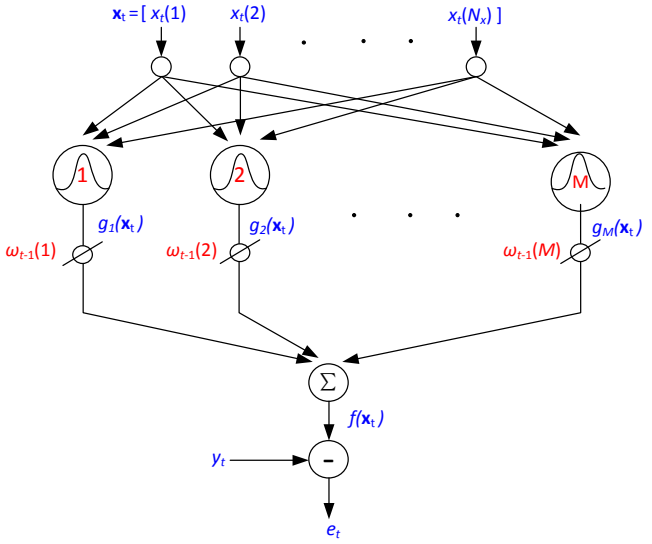


Fig. 1. RBF model structure

The RBF network output is given by

$$f(\mathbf{x}_t) = \sum_{i=1}^{M} w_{t-1}(i) g_i(\mathbf{x}_t) = \boldsymbol{\phi}_t^{\mathrm{T}} \mathbf{w}_{t-1} \tag{2}$$

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 $t-1$, $\mathbf{w}_{t-1} = [\omega_{t-1}(1), \cdots, \omega_{t-1}(M)]^{\mathrm{T}}$ and $\boldsymbol{\phi}_t = [g_1(\mathbf{x}_t), \cdots, g_M(\mathbf{x}_t)]^{\mathrm{T}}$. In this paper, the Gaussian RBF given by

$$g_i(\mathbf{x}_t) = \exp\left(-\frac{1}{2}(\mathbf{x}_t - \mathbf{c}_i)^{\mathrm{T}} \mathbf{H}_i (\mathbf{x}_t - \mathbf{c}_i)\right) \tag{3}$$

is adopted, where $\mathbf{c}_i = [c_i(1), \cdots c_i(N_x)]^{\mathrm{T}}$ and $\mathbf{H}_i = \mathrm{diag}\{\sigma_i^2(1), \cdots, \sigma_i^2(N_x)\}$ which are the centre vector and diagonal covariance matrix of the ith node respectively, with $c_i(j)$ and $\sigma_i(j)$ being the centre and standard-deviation coefficients for the jth input channel respectively. The residual error of RBF network at time t is given by

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

where y_t the observed system output at time t.

Originally described for the ARX model adaptation [11], 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

$$\mathbf{X}_t = [\mathbf{x}_t, \mathbf{x}_{t-1}, \cdots, \mathbf{x}_{t-p+1}]^{\mathrm{T}} \in \mathbb{R}^{p \times N_x} \tag{5}$$

where p is the innovation length which determines the number of past errors used for weight adaptation. Passing $\mathbf{X}_t$ through the RBF nodes gives the information matrix as

$$\begin{aligned} \boldsymbol{\Phi}_t &= \begin{bmatrix} g_1(\mathbf{x}_t) & g_2(\mathbf{x}_t) & \cdots & g_M(\mathbf{x}_t) \\ g_1(\mathbf{x}_{t-1}) & g_2(\mathbf{x}_{t-1}) & \cdots & g_M(\mathbf{x}_{t-1}) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\mathbf{x}_{t-p+1}) & g_2(\mathbf{x}_{t-p+1}) & \cdots & g_M(\mathbf{x}_{t-p+1}) \end{bmatrix} \\ &= [\boldsymbol{\phi}_t, \boldsymbol{\phi}_{t-1}, \cdots, \boldsymbol{\phi}_{t-p+1}]^{\mathrm{T}} \\ &= [\mathbf{g}_1, \ \mathbf{g}_2, \cdots, \mathbf{g}_M] \in \mathbb{R}^{p \times M} \end{aligned} \tag{6}$$

where $\mathbf{g}_i$ is the ith column of $\boldsymbol{\Phi}_t$ 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

$$\mathbf{e}_t = \mathbf{y}_t - \boldsymbol{\Phi}_t \mathbf{w}_{t-1} \tag{7}$$

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

$$\boldsymbol{\Psi}_t = \mathbf{P}_{t-1} \boldsymbol{\Phi}_t^{\mathrm{T}} \left[\lambda \mathbf{I}_p + \boldsymbol{\Phi}_t \mathbf{P}_{t-1} \boldsymbol{\Phi}_t^{\mathrm{T}}\right]^{-1} \tag{8}$$

$$\mathbf{P}_t = \left(\mathbf{P}_{t-1} - \boldsymbol{\Psi}_t \boldsymbol{\Phi}_t \mathbf{P}_{t-1}\right) \lambda^{-1} \tag{9}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \boldsymbol{\Psi}_t \mathbf{e}_t \tag{10}$$

where $\boldsymbol{\Psi}_t \in \mathbb{R}^{M \times p}$ is the Kalman gain matrix, $\mathbf{P} \in \mathbb{R}^{M \times M}$ is 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.

In this paper, the above MRLS algorithm is used to update the node weights. Unfortunately, the MRLS by itself is not adequate enough for non-stationary systems.

III. ON-LINE NODE STRUCTURE OPTIMIZATION

The modeling performance of a RBF network is determined by both the weight vector and node structure. For the Gaussian RBF network, the node structure parameters include the centre vector and covariance matrix of each node, or $\mathbf{c}_i$ and $\mathbf{H}_i$ in (3) respectively. In many RBF network learning methods, the centres are either determined by the input data such as k-means cluctering approach ([43]) or simply by being set as the input data (e.g. [21]). A common standard-deviation is often used for all nodes and set by the trial-and-error or cross-validation method [26]. Such choice of the RBF structure often leads to over large model size and poor performance in tracking the system variation for the modeling of non-stationary systems.

In this work, we propose to fix the model size at a small number. The advantage of fixing the model size is to enable the MRLS to be seamlessly integrated with the node structure optimization. Because the changeable "local characteristic" is of primary importance in a non-stationary system, the node structure parameters needs to be adaptive accordingly in case the MRLS becomes inadequate. While the joint structure

optimization for all nodes can be computationally prohibitive, we propose to only replace one "insignificant" node with a new node whose structural parameters are then optimized. To be specific, if the current RBF network performs poorly despite the weight adaptation with the MRLS, one "insignificant" node with little contribution to the overall performance is identified and replaced by a new node. The centre vector and covariance matrix of the new node are optimized by the quantum PSO (QPSO) algorithm based on the recent data, but the other nodes remain unchanged. Consequently, the RBF structure can be self-tuned to keep tracking the local characteristic of the non-stationary system, and at the same time maintain the model complexity at a moderate level in order to minimize computational cost and achieve fast tracking capability. In order to realize this strategy, it is essential to determine when the node replacement takes place and how the structure of the new node is optimized, which is discussed in detail below.

A. Node replacement

When the RBF structure is not suitable for the current data, the network residual error becomes large and one insignificant node with poor performance is replaced with a new node. In order to prevent the node replacement from occurring too frequently, the "average" residual error is used to measure the performance of the RBF network. Noting that multi-innovation error vector $\mathbf{e}_t$ in (7) consists p number of most recent errors, the normalized "average" residual error is given by

$$\bar{e}_t^2 = \frac{1}{p} \cdot \frac{\|\mathbf{e}_t\|^2}{\|\mathbf{y}_t\|^2} \tag{11}$$

Then we have the following criterion

$$\begin{cases} \text{if } \bar{e}_t^2 < \Delta_1, & \text{the RBF structure remains unchanged} \\ \text{if } \bar{e}_t^2 \geq \Delta_1, & \text{an insignificant node is replaced with} \\ & \text{a new node,} \end{cases} \tag{12}$$

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.

If $\bar{e}_t \geq \Delta_1$, a node with little contribution to the overall system performance is replaced with a new node. It is known that the increment of error variance (IEV) can be used to measure the individual contribution from each node [44], [45]. In order to calculate the IEV for the ith node, we rewrite (7) as

$$\mathbf{y}_t = \mathbf{\Phi}_t \mathbf{w}_{t-1} + \mathbf{e}_t = \mathbf{\Phi}_{t,-i} \mathbf{w}_{t-1,-i} + \omega_{t-1}(i)\mathbf{g}_i + \mathbf{e}_t \tag{13}$$

where $\mathbf{\Phi}_{t,-i}$ is the new information matrix by removing the i^{th} column $\mathbf{g}_i$ from $\mathbf{\Phi}_t$, $\mathbf{w}_{t-1,-i}$ is $\mathbf{w}_{t-1}$ with i^{th} element being removed, and $\omega_{t-1}(i)$ is the weight coefficient for the ith node. Orthogonally projecting $\mathbf{g}_i$ onto the space spanned by column vectors of $\mathbf{\Phi}_{t,-i}$ gives

$$\mathbf{q}_i = \left\{ \mathbf{I} - \mathbf{\Phi}_{t,-i} \left[\mathbf{\Phi}_{t,-i}^{\mathrm{T}} \mathbf{\Phi}_{t,-i} \right]^{-1} \mathbf{\Phi}_{t,-i}^{\mathrm{T}} \right\} \mathbf{g}_i, \tag{14}$$

where $\mathbf{I}$ is the identity matrix with appropriate dimension. Then (13) can be rewritten as

$$\mathbf{y}_t = \mathbf{\Phi}_{t,-i} \mathbf{v}_{t-1,-i} + w_{t-1}(i)\mathbf{q}_i + \mathbf{e}_t \tag{15}$$

where $\mathbf{v}_{t-1,-i} = \left[\mathbf{\Phi}_{t,-i}^{\mathrm{T}} \mathbf{\Phi}_{t,-i} \right]^{-1} \mathbf{\Phi}_{t,-i}^{\mathrm{T}} \mathbf{g}_i$ which is the least squares estimation for the new information matrix $\mathbf{\Phi}_{t,-i}$. Then we have $w_{t-1}(i) = \mathbf{q}_i^{\mathrm{T}} \mathbf{y}_t / \mathbf{q}_i^{\mathrm{T}} \mathbf{q}_i$. Because $\mathbf{q}_i$ is orthogonal to the space spanned by column vectors of $\mathbf{\Phi}_{t,-i}$, the IEV for the ith node is given by

$$\mathrm{IEV}_i = \| \, w_{t-1}(i)\mathbf{q}_i \, \|^2 = w_{t-1}^2(i) \mathbf{q}_i^{\mathrm{T}} \mathbf{q}_i. \tag{16}$$

While a node with smaller IEV has less contribution to the overall performance, we order the IEV-s for all nodes as

$$\mathrm{IEV}_{1'} \leq \mathrm{IEV}_{2'} \leq \cdots \leq \mathrm{IEV}_{M'} \tag{17}$$

where $\mathrm{IEV}_{i'}$ is for node i' with the ith smallest IEV. Therefore, node $1'$ has the least contribution to the overall performance and can be replaced with a new node.

While the IEV comparison in (17) gives the most "insignificant" node, or the node with least contribution to the overall performance, its calculation suffers from high complexity and numerical instability. This is because, as is shown in (14), the IEV calculation requires the matrix inversion of $\left[\mathbf{\Phi}_{t,-i}^{\mathrm{T}} \mathbf{\Phi}_{t,-i} \right]^{-1}$. In a highly non-stationary system, it is possible that some nodes are so badly structured sometimes that the node outputs become very small. This makes $\mathbf{\Phi}_{t,-i}^{\mathrm{T}} \mathbf{\Phi}_{t,-i}$ be ill-conditioned, resulting in numerical instability.

Alternatively, we can use the weighted node-output variance (WNV) to determine which node should be replaced. The WNV for the ith node is defined as

$$\mathrm{WNV}_i = \| \, \omega_{t-1}(i)\mathbf{g}_i) \, \|^2 = \omega_{t-1}^2(i) \mathbf{g}_i^{\mathrm{T}} \mathbf{g}_i \tag{18}$$

where $\omega_{t-1}(i)\mathbf{g}_i$ is the weighted output for the ith node. The WNV for all nodes is ordered as

$$\mathrm{WNV}_{1'} \leq \mathrm{WNV}_{2'} \leq \cdots \leq \mathrm{WNV}_{M'} \tag{19}$$

where $\mathrm{WNV}_{i'}$ is for node i' with the ith smallest WNV. Then from (19), the node $1'$ with the smallest WNV is replaced with a new node.

The relation between the IEV and WNV is shown in Fig. 2, where there are 3 nodes for illustration. In practice, since the nodes are often well "separated" to have good data coverage, the correlation between node outputs is limited. This implies that, if $\mathrm{IEV}_i \ll \mathrm{IEV}_j$, we usually have $\mathrm{WNV}_i \ll \mathrm{WNV}_j$. Therefore, if there exist a group of L ($1 \leq L \leq M$) nodes with very small WNV, they also have significantly smaller IEV than the other nodes so that they should all be replaced. In our proposed on-line scheme where one node is replaced at a time, it is not important to determine which of these nodes with small WNV is replaced first, because other nodes with small WNV are to be replaced at a later time. In fact, the IEV criterion in (17) replaces the most "insignificant" node, and the WNV criterion in (19) chooses one of the "insignificant" nodes. While they lead to similar performance

in on-line approach applications, the WNV criterion not only is more robust but also requires much less complexity than its IEV counterpart. We have done extensive simulations which all show that node replacement based on (17) and (19) lead to similar results.

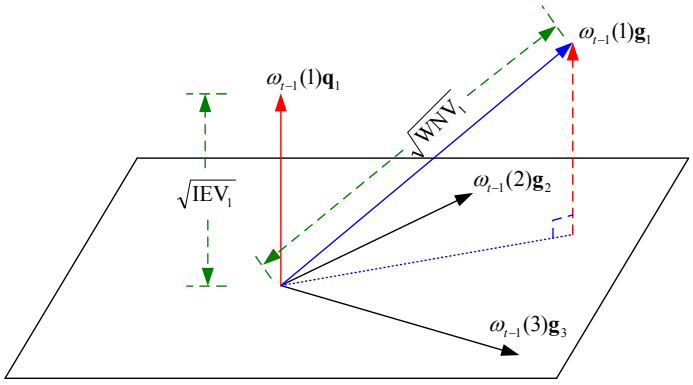


Fig. 2. The IEV and WNV

B. Iterative node structure optimization and weight adaptation

When an insignificant node is replaced with a new node, the structure (or the centre vector and covariance matrix for the Gaussian RBF node) of the new node needs to be optimized based on the recent data. Without losing generality, we assume the ith node is replaced. The structure optimization is to find the best structural parameters of the new node, $g_i(.)$, which minimizes the cost function as

$$J_t = \sum_{i=t}^{t-p+1} e_t^2(i) = \mathbf{e}_t^{\mathrm{T}}\mathbf{e}_t = \left(\mathbf{y}_t - \mathbf{\Phi}_t\mathbf{w}_{t-1}\right)^{\mathrm{T}}\left(\mathbf{y}_t - \mathbf{\Phi}_t\mathbf{w}_{t-1}\right), \tag{20}$$

with the structure for other nodes, $g_{j\neq i}(.)$, being unchanged, where the node structure determines how the information matrix $\mathbf{\Phi}_t$ is formed. Since J_t is the square error summation for the recent p inputs, the structure of the new node is optimized for the recent p data rather than the current data, making the modeling have better performance in generalization.

Because J_t depends on both the node structure and weight vector, when the structure of the new node is adjusted, the weight vector which is based on the previous structure should also be modified. The joint optimization of the structure of the new node and weight vector can be complicated. We understand that, for a particular node structure, the weight vector can be adapted by the MRLS algorithm. On the other hand, when the weight vector is given, the structure of the new node can be optimized by the the quantum-PSO (QPSO) searching algorithm [42] (which will be discussed in detail later). This clearly suggests an iterative approach for the structure optimization and weight adaptation. To be specific, when an insignificant node is replaced by a new one, we have the following iterative steps:

1) Initialize the structure of the new node, and initialize the weight coefficient of the new node as zero, and the weights of other nodes as the those before the node replacement. Use the MRLS to update the weight vector by one step.
2) Fix the weight vector, and update the structure of the new node with the QPSO algorithm by one step.
3) Fix the structure of the new node, and adapt the weight vector with the MRLS by one step.
4) Repeat the steps 2) and 3) until the normalized average error variance is small enough that

$$\frac{J_t}{\| \mathbf{y}_t \|^2} < \Delta_2, \tag{21}$$

or the maximum iteration number is reached, where Δ_2 is a pre-set constant depending on system requirement.

We highlight that in the above iterative process, at every iteration, the MRLS weight adaption uses the same batch of input data but with different structure of the new node obtained from the QPSO optimization. In contrast, when no node is replaced, the RBF structure remains unchanged and the weight vector is adapted by the MRLS algorithm with new batch of input data once at a time.

We particularly note that it is not appropriate to use the QPSO algorithm to search for the weight vector and the new node's structure altogether. Unlike the node structural parameters as will be shown later, the weight coefficients usually have large dynamic range with little direct link to the input data. It is thus hard to properly initialize the QPSO algorithm for fast convergence. As a result, a large number of "particles" and iterations may have to be used in the QPSO searching, leading to very slow convergence and high complexity. This is especially serious in the non-stationary system where the weight coefficients keep varying with time.

C. Structure optimization with the QPSO

In this section, we first describe the QPSO algorithm for the node structure optimization, and then propose a data-driven method to initialize the QPSO to achieve fast convergence.

1) The Quantum-PSO: The QPSO is an evolutional searching method including K number of particles. When the QPSO is used for the structure optimization, every particle consists of N_x pairs of centres and standard-deviations parameters of the new node, where N_x is the number of input channels defined in (1). To be specific, at the lth iteration, the kth particle $(k = 1, \cdots, K)$ is expressed as

$$\gamma_{t,\,l}^{(k)} = \left\{ c_{t,\,l}^{(k)}(1),\ \cdots,\ c_{t,\,l}^{(k)}(N_x),\ \sigma_{t,\,l}^{(k)}(1),\ \cdots,\ \sigma_{t,\,l}^{(k)}(N_x) \right\} \tag{22}$$

where $c_{t,\,l}^{(k)}(j)$ and $\sigma_{t,\,l}^{(k)}(j)$ are the centre and standard-deviation coefficients for the jth input channel $(j = 1, \cdots, N_x)$ respectively, and the subscript t represents that the node replacement occurs at time t. We particularly note that, unlike many existing RBF approaches, we do not assume $\sigma_i(1) = \cdots = \sigma_i(N_x)$. As a result, different nodes may have different "width" of Gaussian functions for different input

channels, which enables the nodes with more flexibility in covering the data.

At every iteration, the particles move from one position to another, where every particle position corresponds to one possible structure of the new node. With the weight vector being fixed from the MRLS adaption at the previous iteration, the cost function value for the particle position $\gamma_{t,\ l}^{(k)}$ can be obtained as $J_t(\gamma_{t,\ l}^{(k)})$, $k = 1, \cdots, K$, where $J_t(.)$ is defined in (20).

At the lth iteration, the *best position* for the kth particle ($k = 1, \cdots, K$) has the minimum J_t among all of its previous positions as

$$pbest_{t,\ l}^{(k)} = \arg\min\{J_t(\gamma_{t,\ i}^{(k)}) \mid i = 0, \cdots, l\} \tag{23}$$

The global best structure of the new node at the lth iteration is the particle position with the minimum J_t as

$$gbest_{t,\ l} = \arg\min\{J_t(pbest_{t,\ l}^{(k)}) \mid k = 0, \cdots, K\} \tag{24}$$

The local attractor for the kth particle ($k = 1, \cdots, K$) for its next iteration is given by

$$\alpha_{t,\ l}^{(k)} = \varphi \cdot pbest_{t,\ l}^{(k)} + (1 - \varphi) \cdot gbest_{t,\ l} \tag{25}$$

where φ is a randomly generated number between $[0, 1]$. Then the iteration rule for the kth particle ($k = 1, \cdots, K$) is given by

$$\gamma_{t,\ l+1}^{(k)} = \alpha_{t,\ l}^{(k)} + \varsigma \cdot \beta \cdot |mbest_{t,\ l} - \gamma_{t,\ l}^{(k)}| \cdot \ln \frac{1}{\mu} \tag{26}$$

where ς is randomly generated $+1$ or -1 with equal probability, μ is a randomly generated number between $[0, 1]$, β is the only controlling parameter in the QPSO which is determined according to specific applications, and $mbest_{t,\ l}$ is the gravity centre of all particles' best position at the lth iteration which is given by

$$mbest_{t,\ l} = \frac{1}{K} \sum_{k=1}^{K} pbest_{t,\ l}^{(k)} \tag{27}$$

2) The QPSO initialization: While the QPSO can achieve global convergence, the converging time depends greatly on the particles initialization. If the particles are initialized far from the optimum solutions, a large number of particles and iterations may have to be used, leading to slow convergence and high complexity. In the original QPSO, the particles are randomly initialized. This is not desirable for the structure optimization especially in non-stationary systems where the structure parameters keep varying with time. Below we propose a data-driven method for the particle initialization which can follow the "local" statistics of the input data.

In many applications, it is likely that the optimum centres are around the input data. Some approaches even choose the node centres as the input data (e.g [14], [15], [17]–[20]), though this is not optimum. While the structure optimization is based on the recent p input data, the centre for the first particle can be initialized as the average of the recent p inputs as

$$c_{t,0}^{(1)}(j) = \frac{1}{p} \sum_{i=t-p+1}^{t} x_i(j), \qquad j = 1, \cdots, N_x \tag{28}$$

where we recall that $x_i(j)$ is the jth channel input at time i.

The centres for the remaining particles ($k = 2, \cdots, K$) are randomly initialized based on the Gaussian process as

$$c_{t,0}^{(k)}(j) = N(m_t(j), s_t(j)), \qquad j = 1, \cdots, N_x, \tag{29}$$

where $N(m_t(j), s_t(j))$ represents one realization of the Gaussian process with mean $m_t(j)$ and standard-deviation $s_t(j)$. We let $m_t(j) = c_{t,0}^{(1)}(j)$ which is centre for the first particle, and $s_t(j)$ be the standard-deviation of the input samples as

$$s_t(j) = \sqrt{\frac{1}{p} \sum_{i=t-p+1}^{t} |x_i(j) - m_t(j)|^2}, \qquad j = 1, \cdots, N_x \tag{30}$$

We highlight that the centre $c_{t,0}^{(k)}(j)$ is different for every particles because each is one random realization of the Gaussian process $N(m_t(j), s_t(j))$, though $m_t(j)$ and $s_t(j)$ are the same for all particles (except the first particle). By doing so, we have not only many initial particles with centres around the input data for fast convergence, but also some particles far from the data to achieve good coverage.

Once the centre parameters for all particles are initialized as in (28) and (29), the initial standard-deviation parameters for the particles are determined by how far the corresponding centres are away from the nearest centre of other nodes. To be specific, if the centre coefficient for the jth input ($j = 1, \cdots, N_x$) of the kth particle ($k = 1, \cdots, K$) for the new node is $c_{t,0}^{(k)}(j)$, the corresponding standard-deviation is initialized as

$$\sigma_{t,0}^{(k)}(j) = \rho \cdot |c_{t,0}^{(k)}(j) - c_{\text{nearest}}^{(k)}(j)| \tag{31}$$

where ρ is a constant scaling factor, and $c_{\text{nearest}}^{(k)}(j)$ is the centre coefficient for the jth input of the non-replaced node with nearest distance to $c_{t,0}^{(k)}(j)$.

At the beginning, the initial best structure of the new node is set as the first particle as

$$gbest_{t,0} = [c_{t,0}^{(1)}(1), \cdots, c_{t,0}^{(1)}(N_x), \quad \sigma_{t,0}^{(1)}(1), \cdots, \sigma_{t,0}^{(1)}(N_x)] \tag{32}$$

IV. The MRLS-QPSO Algorithm

The proposed approach is named as the MRLS-QPSO algorithm in this paper, as it integrates the MRLS weight adaptation and QPSO node optimization into one process. The MRLS-QPSO algorithm is fundamentally different from existing on-line RBF approaches with growing/pruning model size (e.g. [14]–[21]). In existing approaches, the centres of the newly added node are simply set as the current input data and the standard-deviations are determined from a priori information. As a result, the constructed network structure only fits into the data rather than the underlying model, which

often makes the model size grow with the data. In comparison, the proposed MRLS-QPSO algorithm can adaptively optimize both weight coefficients and node structure on-line, so that it can track the system variation with a sparse model.

Fig. 3 shows the flowchart of the proposed MRLS-QPSO algorithm.

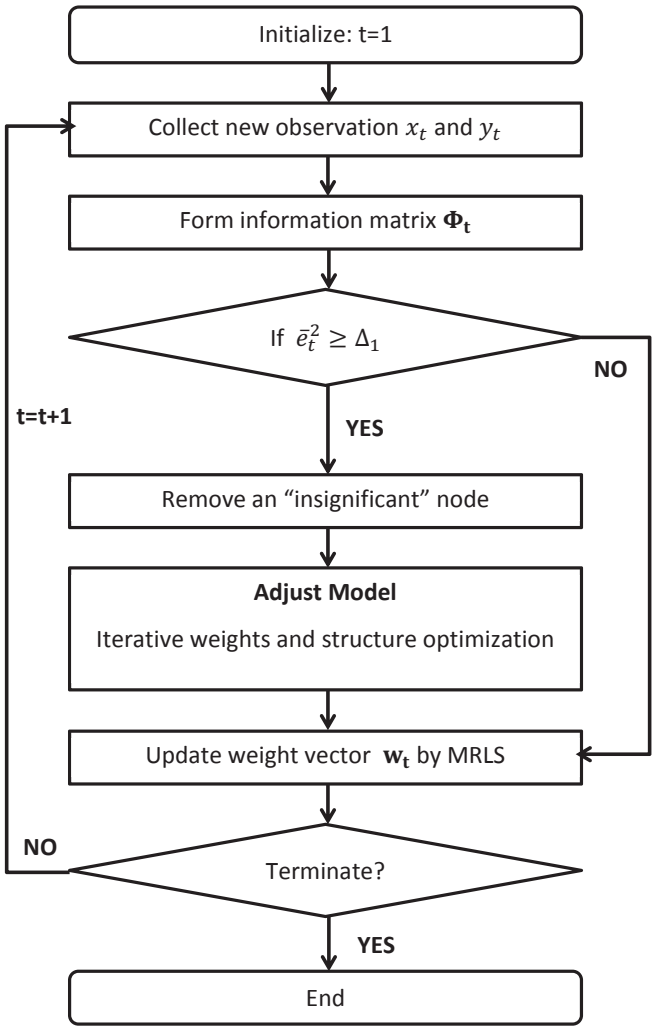


Fig. 3. Flowchart of the proposed MRLS-QPSO algorithm.

The MRLS-QPSO algorithm is summarized as below.

Initialization

Initialize the structure of the RBF nodes

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

Set the error threshold Δ_1 for the node replacement in (12).
Set the error threshold Δ_2 for the iterative structure and weight optimization in (21).

Set the QPSO controlling parameters β in (26).

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 (5) and (6) respectively.

Obtain the error vector $\mathbf{e}_t$ and the average error power $\bar{e}_t^2$ in (7) and (11) respectively.

If $\bar{e}_t^2 \geq \Delta_1$, do **Structure adaptation**

Calculate the WNV for each node as in (18) and order them as $\mathrm{WNV}_{1'} \leq \cdots \leq \mathrm{WNV}_{M'}$
Replace the node $1'$ with a new node.

Initialization for the iterative weights and structure optimization

Initialize particle centres and standard-deviations as in (28), (29) (31) respectively.
Choose the initial best structure for the new node as the first particle as

$$gbest_{t,0} = [c_{t,0}^{(1)}(1), \cdots, c_{t,0}^{(1)}(N_x), \quad \sigma_{t,0}^{(1)}(1), \cdots, \sigma_{t,0}^{(1)}(N_x)]$$

Initialize the weight of the new node to 0.
Maintain the weights of other nodes unchanged.
Initialize $\mathbf{P}_{t,l=0} = \delta \mathbf{I}$ for the MRLS, where δ is a large number.

For $l = 1, \cdots, L$, do iterative weights and structure optimization,

Weight vector adaptation with the MRLS

With the structures of the new node $gbest_{t,l-1}$ and the same input data $\mathbf{X}_t$, do

Obtain the new information matrix $\mathbf{\Phi}_{t,l}$ as in (6)
Obtain the new error vector $\mathbf{e}_{t,l}$ as in (7).

Update the weights with the MRLS as

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

$$\mathbf{P}_{t,l} = \left(\mathbf{P}_{t,l-1} - \mathbf{\Psi}_{t,l}\mathbf{\Phi}_{t,l}\mathbf{P}_{t,l-1}\right)\lambda^{-1}$$

$$\mathbf{w}_{t,l} = \mathbf{w}_{t,l-1} + \mathbf{\Psi}_{t,l}\mathbf{e}_{t,l}$$

Structure optimization for the new node with the QPSO

Fix the weight vector at $\mathbf{w}_{t,l}$

For every particle, $k = 1, \cdots, K$, do
Randomly generate φ and μ within $[0, 1]$.
Randomly generate ς to be $+1$ or -1.

Update $pbest_{t,\,l}^{(k)}$ and $gbest_{t,\,l}$ as in (23) and (24) respectively.
Update the local attractor as

$$\alpha_{t,\,l}^{(k)} = \varphi \cdot pbest_{t,\,l}^{(k)} + (1 - \varphi) \cdot gbest_{t,\,l}$$

Update the gravity centre of all particles as

$$mbest_{t,\,l} = \frac{1}{K} \sum_{k=1}^{K} pbest_{t,\,l}^{(k)}$$

Update the particles as

$$\gamma_{t,\,l+1}^{(k)} = \alpha_{t,\,l}^{(k)} + \varsigma \cdot \beta \cdot |mbest_{t,\,l} - \gamma_{t,\,l}^{(k)}| \cdot \ln \frac{1}{\mu}$$

If $J_t / \parallel \mathbf{y}_t \parallel^2 \leq \Delta_2$, iterative optimization stops.
Else go to the next iteration.

Otherwise, if $\bar{e}_t^2 < \Delta_1$, do **weight adaptation** only

Adapt the weight with the MRLS as in (7), (8),(9) and (10)

End

Initially the centres of all nodes can be randomly placed around the data or simply the data itself, and the variances can be set as a common value. While such initial structure setting is not optimum, when the on-line adaptation goes on, each of the nodes is to be replaced by a new one with a better structure once at a time. It is expected that the node replacement occurs frequently at the initial stage. At the beginning, the node structure and weight vector adaptation start simultaneously. This is verified by extensive simulations including all of those in Section V.

Another issue is the model size selection which depends on specific applications. While there are many model size selection algorithms (e.g. [46], [47]), most are for off-line approaches or stationary systems so that they are not applicable in on-line applications. The model size selection for the proposed algorithm will be left as an interesting open topic for future research. In many cases, the model size of the proposed approach can be easily determined empirically, e.g. by trial-and-error of the size until there is no significant gain in model performance. Extensive simulation results are given later in this paper to compare the performance of the proposed algorithm with different model sizes.

Finally, we point out that, although the proposed approach is for the RBF network with Gaussian kernels, it can be easily adapted to many other associative networks with linear-in-the-parameters structure, e. g. thin-plate-spline, B-spline networks.

V. Simulation and discussions

In this section, computer simulations are given to compare the proposed MRLS-QPSO algorithm with some typical on-line modeling approaches including the linear MRLS, RAN, GAP-RBF and ELM algorithms. Except for the linear MRLS, all approaches apply Gaussian nodes.

In the proposed MRLS-QPSO approach, the error threshold for the node replacement in (12) is set as $\Delta_1 = 10^{-3}$, 10 swarm particles are used in the QPSO, and the iterative weight and structure adaption process stops if $\Delta_2 < 10^{-6}$ or the iteration number reaches 5. In all other approaches, the controlling parameters are carefully chosen based on trial-and-error to achieve best performance. We note that, while both off-line and on-line ELM approaches are available [33]–[39], the off-line ELM with a fixed model size based on the whole batch of data is used in this section for comparison, because it is easier to set up the simulation for the best performance. This is reasonable because the off-line and on-line ELM approaches have similar performance [38]. Thus the comparison is logically general for all versions of the ELM.

These algorithms are compared in the application of on-line time series prediction. To be specific, in this section, the T-step ahead prediction is to use the past four samples

$$\mathbf{x}_t = [y_t,\ y_{t-6},\ y_{t-12},\ y_{t-18}]^{\mathrm{T}} \tag{33}$$

to estimate the future sample y_{t+T}.

The prediction performance is measured by the root mean square error (RMSE) and mean absolute error (MAE). At time t, the RMSE and MAE are defined as

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

$$\mathrm{MAE}(t) = \frac{1}{t} \sum_{i=1}^{t} \|y_i - f(\mathbf{x}_i)\| \tag{35}$$

respectively.

Two benchmark chaotic time series are considered in this section: Mackey-Glass and Lorenz time series. In order to fully verify the proposed approach, both stationary and non-stationary cases are considered. In the stationary case, the parameters controlling the chaotic time series behavior are fixed. While in the non-stationary case, these controlling parameters are changing either abruptly (piecewise function) or continuously.

A. Mackey-Glass time series

The Mackey-Glass time series is generated from the differential delay equation as

$$\frac{dx(t)}{dt} = \frac{ax(t-c)}{1 + x^{10}(t-c)}) - bx(t) \tag{36}$$

where a, b and c are controlling parameters, and especially when $c \geq 17$ the equation shows typical chaotic behavior. In this simulation, 5000 samples are generated by the 4th order Runge-Kutta method with the step size of 1, and the last 3000 samples are used for the prediction. The forward prediction step is set as $T = 60$.

1) Mackey-Glass time series with fixed parameters: First, we let $a = 0.2$, $b = 0.1$ and $c = 30$. Table I and Fig. 4 compare the final prediction performance and RMSE learning

TABLE I

Mackey-Glass (fixed parameters): FINAL PREDICTION PERFORMANCE

Algorithm	RMSE	MAE	RBF nodes
MRLS	0.3693	0.2937	–
RAN	0.1145	0.0768	726
GAP-RBF	0.1278	0.0957	22
ELM (100 nodes)	0.0761	0.0560	100
ELM (300 nodes)	0.0476	0.0339	300
ELM (500 nodes)	0.0376	0.0253	500
MRLS-QPSO (5 nodes)	0.0297	0.0182	5
MRLS-QPSO (15 nodes)	0.0180	0.0070	15
MRLS-QPSO (25 nodes)	0.0168	0.0053	25

curves for different approaches respectively. It is clear that the linear MRLS has the worst performance. The RAN and GAP-RBF have comparable performance, but the GAP-RBF has far fewer number of nodes than the RAN. This is because the GAP-RBF can prune the model size and the RAN can not. Both the ELM and proposed approaches have significantly better performance than the others, and the proposed approach with 5 nodes has similar performance to the ELM approach with 500 nodes.

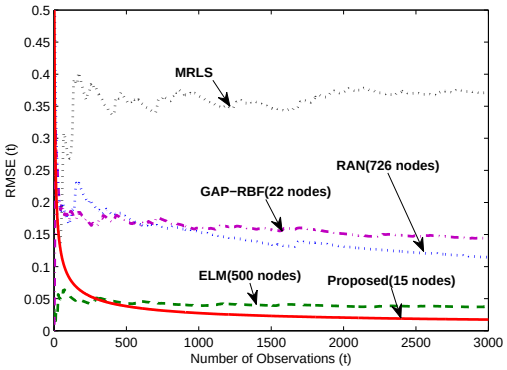
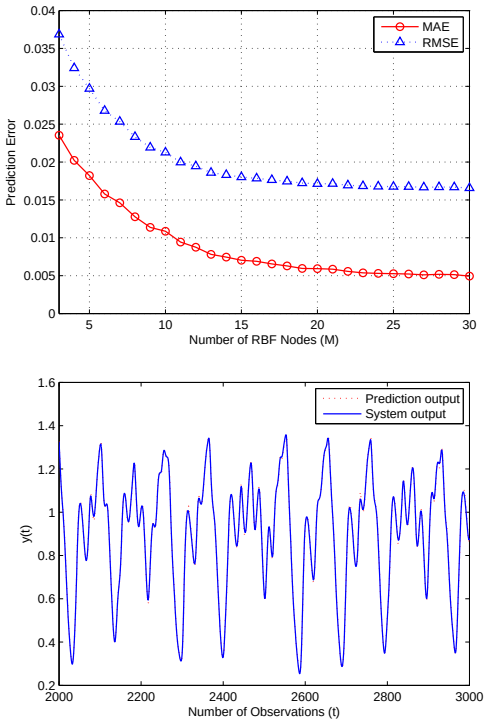


Fig. 4. **Mackey-Glass (fixed parameters)**: RMSE learning curves.

Fig. 5 (a) shows the final prediction performance of the proposed algorithm with different number of RBF nodes. It is clear that the prediction performance improves with larger number of nodes M, but the improvement becomes insignificant when $M \geq 15$. We recall that with $M = 5$, the proposed approach has similar performance to the ELM approach with $M = 500$. Fig. 5 (b) shows the proposed algorithm with $M = 15$ can very well predict the time series on-line.

2) Mackey-Glass time series with piecewise function: In this simulation, the above Mackey-Glass time series are weighted by the piecewise function as

$$y_1(t) = y(t) \cdot \Psi(t), \tag{37}$$



(a) Prediction performance *vs* No. of nodes (M)
(b) On-line prediction with $M = 15$

Fig. 5. **Mackey-Glass (fixed parameters)**: prediction performance of the proposed MRLS-QPSO algorithm.

where

$$\Psi(t) = \begin{cases} 0.0005 \cdot t^{\frac{1}{0.98}} & 0 \leq t \leq 999 \\ 0.0006 \cdot t^{\frac{1}{0.95}} & 1,000 \leq t \leq 1,999 \\ 0.0003 \cdot t^{\frac{1}{0.97}} & 2,000 \leq t \leq 2,999 \end{cases} \tag{38}$$

$y_1(t)$ is used for the time series prediction. As is shown in Fig. 7 (b), $y_1(t)$ is clearly non-stationary as it has different dynamic range at different time intervals.

Table II and Fig. 6 compare the final prediction performance and RMSE learning curves in this non-stationary case for different approaches respectively. Comparing these results with those in the previous simulation for the stationary case, we can observe that, while all approaches have worse prediction performance than those in the stationary case, the comparison between different approaches are similar, and particularly the proposed MRLS-QPSO still has the best performance. Except for the proposed approaches, all other RBF approaches have significantly larger number of nodes.

Fig. 7 (a) shows the final prediction performance of the proposed algorithm with different number of RBF nodes. Unlike that in the previous stationary case, the prediction performance changes little with more nodes. In fact, the RMSE may even become slightly worse when the model size becomes larger. This actually matches our previous statement that in the non-stationary scenario, "local" characteristic is

TABLE II

Mackey-Glass (piecewise function): FINAL PREDICTION PERFORMANCE

Algorithm	RMSE	MAE	RBF nodes
MRLS	0.9238	0.6837	–
RAN	0.6673	0.4848	2378
GAP-RBF	0.5323	0.3597	195
ELM (300 nodes)	0.3853	0.2744	300
ELM (500 nodes)	0.3227	0.2226	500
ELM (1,000 nodes)	0.2215	0.1470	1,000
MRLS-QPSO (5 nodes)	0.1158	0.0384	5
MRLS-QPSO (10 nodes)	0.1897	0.0314	10

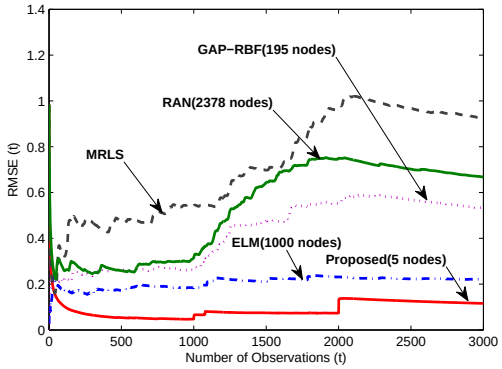
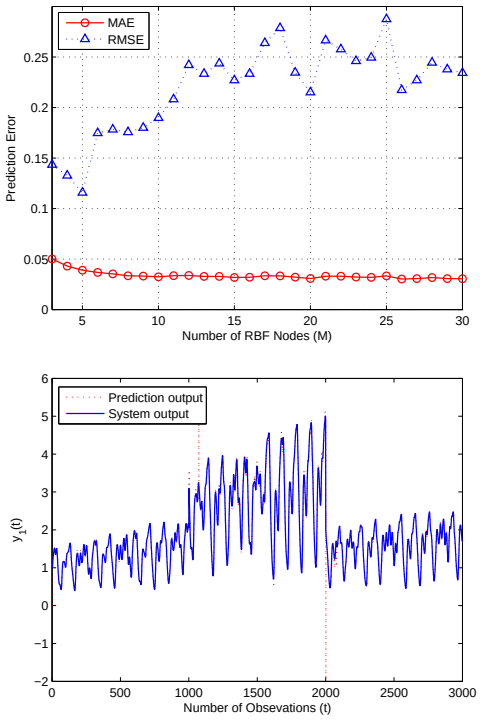


Fig. 6. **Mackey-Glass (piecewise function)**: RMSE learning curves.



(a) Prediction performance *vs* No. of nodes (M)
(b) On-line prediction with $M = 5$

Fig. 7. **Mackey-Glass (piecewise function)**: prediction performance of the proposed MRLS algorithm.

more important than the "global" one. With fewer nodes, the proposed algorithm "forgets" the previous data faster so that it focuses more on the recent data. Fig. 7 (b) clearly shows that with only 5 nodes, the proposed algorithm well predicts this non-stationary time series on-line.

B. Lorenz time series

The Lorenz chaotic time series is also often used as a benchmark in many applications [48]. As a three dimensional and highly non-linear 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. For the 5000 data samples generated for $y(t)$, the last 3000 stable samples are used for the prediction since Lorenz system is very sensitive to the initial condition.

1) Lorenz time series with fixed parameters: We first consider the Lorenz time series with fixed parameters as $a = 10$, $b = 8/3$ and $c = 28$. The trajectory of this Lorenz system is shown in Fig. 8.

Table III, IV and V compare the final prediction performance for different approaches for the prediction step as $T = 20$, 40 and 60 respectively. In all of the tables, the linear MRLS has the worst prediction performance, because a linear approach cannot model the non-linear time series. The RAN and GAP-RBF achieve comparable prediction performance but the GAP-RBF has more compact model than the RAN. The model size for the GAP-RBF is still very large compared to the proposed approach: several tens *vs* only several.

The performance of the ELM and the proposed MRLS-QPSO algorithms with different model size are also shown in the tables. It is seen that when the ELM and GAP-RBF have comparable model sizes, their performance are comparable as well. But the ELM can achieve better performance than the GAP-RBF with more nodes.

It is also clear that, while the performance of the ELM depends greatly on the model size, the proposed approach is less sensitive to the model size. The propose algorithm with 5 nodes has better performance than the ELM with 500 (or more) nodes. This clearly states the importance of the node structure optimization.

Fig. 9 compares the MAE learning curves of the proposed

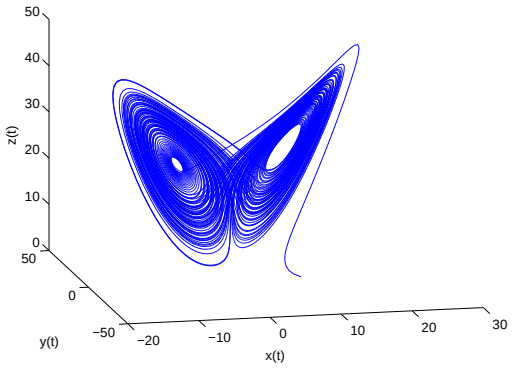


Fig. 8. Lorenz Attractor.

TABLE III
Lorenz time series (fixed parameters): FINAL PREDICTION PERFORMANCE, T=20

Algorithm	RMSE	MAE	RBF nodes
MRLS	7.4271	5.9214	–
RAN	2.7486	2.2543	150
GAP-RBF	2.3294	1.4829	70
ELM (100 nodes)	3.7038	2.8546	100
ELM (500 nodes)	0.7893	0.5389	500
MRLS-QPSO (5 nodes)	0.2076	0.1224	5
MRLS-QPSO (7 nodes)	0.1946	0.1008	7
MRLS-QPSO (9 nodes)	0.1822	0.0858	9

TABLE IV
Lorenz time series (fixed parameters): FINAL PREDICTION PERFORMANCE, T=40

Algorithm	RMSE	MAE	RBF nodes
MRLS	8.1432	6.0564	–
RAN	3.4014	2.7920	167
GAP-RBF	4.2010	2.6401	55
ELM (100 nodes)	5.2150	3.6280	100
ELM (500 nodes)	1.5095	1.0353	500
MRLS-QPSO (5 nodes)	0.3155	0.1375	5
MRLS-QPSO (7 nodes)	0.3027	0.1233	7
MRLS-QPSO (9 nodes)	0.2933	0.1197	9

TABLE V
Lorenz time series (fixed parameters): FINAL PREDICTION PERFORMANCE, T=60

Algorithm	RMSE	MAE	RBF nodes
MRLS	9.9497	5.6003	–
RAN	4.0956	3.2670	130
GAP-RBF	6.2938	3.9318	66
ELM (100 nodes)	7.8140	5.5330	100
ELM (500 nodes)	3.2723	2.1628	500
ELM (1,000 nodes)	0.6011	0.4008	1000
MRLS-QPSO (5 nodes)	0.4529	0.1436	5
MRLS-QPSO (7 nodes)	0.4225	0.1333	7
MRLS-QPSO (9 nodes)	0.4068	0.1297	9

approach with different innovation length p applied in the MRLS weight adaptation, where the Gaussian noise with zero mean and 0.02 variance is added to the Lorenz time series to highlight the noise rejection effect from p. It is clearly shown that the larger the p is, the more robust the MRLS-QPSO against the noise. But a higher p leads to higher complexity.

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

$$b = \frac{4 + 3(1 + \sin(0.1t))}{3} \qquad (39)$$
$$c = 25 + 3(1 + \cos(2^{0.001t})).$$

The prediction step is fixed at $T = 40$, and number of nodes for the proposed approach and the ELM are fixed at 5 and $1,000$ respectively. We note that the performance comparison of the proposed and ELM algorithms with different model sizes is similar to that in the above simulation so that the results are not shown.

Figs. 10 (a) and (b) compare the RMSE and model size learning curves for different approaches respectively. It is clearly shown that the linear MRLS algorithm has the worst performance. The RAN and GAP-RBF approaches have comparable prediction performance. While the GAP-RBF can produce a more compact model than the RAN, both approaches have model sizes increasing with the number of data input. The ELM approach with 1000 nodes has better performance than the RAN and GAP-RBF, and the proposed algorithm has the best prediction performance with the smallest model size among all approaches.

Fig. 11 shows that the proposed MRLS-QPSO algorithm can predict this non-stationary time series almost perfectly.

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 drift to obtain

$$y_1(t) = 1.1^{0.01t} \cdot y(t), \qquad (40)$$

and $y_1(t)$ is used for the time series prediction. The prediction step is fixed at $T = 40$, and number of nodes for the proposed approach is fixed at 5.

Fig. 12 shows that the proposed MRLS-QPSO algorithm can well track the $y_1(t)$ on-line. It is clear that $y_1(t)$ is even more non-stationary than the time series in the previous simulation with time varying controlling parameters, where the dynamic range of $y_1(t)$ goes from around [-20,20] initially to about [-2000,2000] in the end.

Table VI and Fig. 13 compare the final prediction perfor-

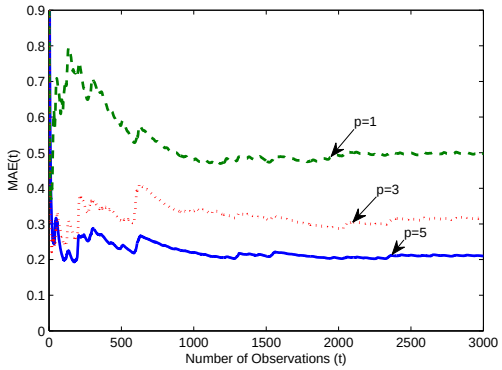
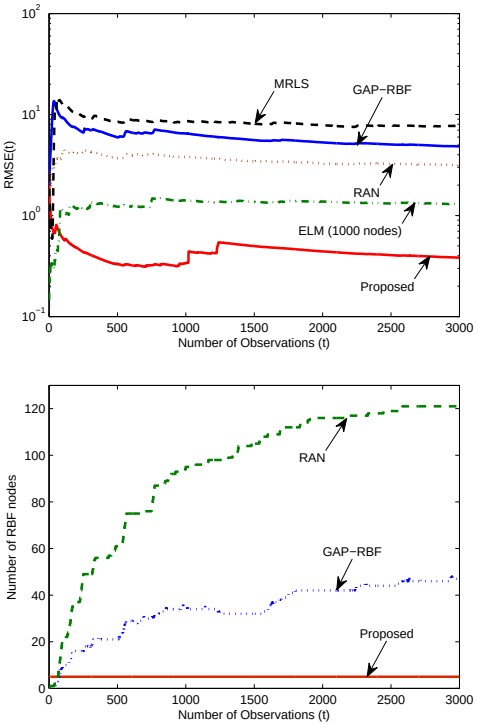


Fig. 9. **Lorenz time series (fixed parameters)**: the MRLS-QPSO with different innovation length p.

TABLE VI

Lorenz time series (time base drift): FINAL PREDICTION PERFORMANCE, T=40

Algorithm	RMSE	MAE	RBF nodes
MRLS	425.1496	246.1235	–
RAN	400.5218	270.2865	2769
GAP-RBF	381.8548	209.7623	987
ELM (1,000 nodes)	287.7531	186.0943	1000
ELM (2,000 nodes)	180.9195	115.4693	2000
ELM (3,000 nodes)	88.2288	46.2154	3000
MRLS-QPSO (5 nodes)	26.6658	11.1356	5



(a) RMSE learning curves
(b) Model size learning curves

Fig. 10. **Lorenz time series (time varying parameters)**: RMSE and model size learning curves.

mance and RMSE learning curves for different approaches respectively. It is clearly shown that the proposed MRLS-QPSO has significantly better performance than the others. In fact, the MRLS-QPSO is effectively the only approach here that can well track this highly non-stationary Lorenz time series.

VI. Conclusion

In this paper, a novel on-line RBF modeling approach is proposed for non-linear and non-stationary dynamic systems. The major contribution is to combine the MRLS weight adaptation and QPSO node optimization in an innovative way. Based on a RBF model with a fixed small number of nodes, the MRLS is used to adapt the node weights at every time step. The node optimization, of replacing the worst existent node by a new one, is however activated when the modeling performance becomes inadequate while using the MRLS alone. This is achieved by optimizing the center vector and covariance matrix of the new node using the QPSO on-line. A data-driven initialization method is proposed in order to achieve fast convergence in the QPSO. Numerical simulations have demonstrated that the proposed MRLS-QPSO algorithm can achieve significantly better performance than existing approaches with a very sparse model.

The algorithm proposed here has been specifically aimed at modeling general non-stationary, non-linear dynamical systems using real time observational data. A number of simulated time series with various non-stationarities have been used as good demonstrators. Potential applications of the proposed algorithm range from military applications (e.g. target tracking) to fundamental communications applications (e.g. channel equalization). We have shown how a very competitive performance can be achieved by integratively employing flexible model structure, recursive parameter estimation and evolutionary computational techniques.

References

[1] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[2] L. Ljung, *System Identification - Theory For the User*. PTR Prentice Hall, 1999.

[3] 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.

[4] H. Peng, T. Ozaki, Y. Toyoda, H. Shioya, K. Nakano, V. Haggan-Ozaki, and M. Mori, "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.

[5] 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.

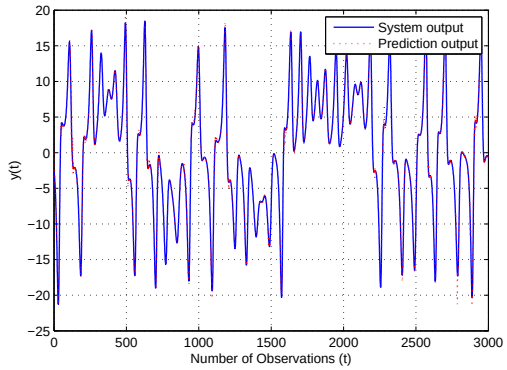


Fig. 11. **Lorenz time series (time varying parameters)**: on-line 40-steps ahead prediction by the proposed MRLS-QPSO.

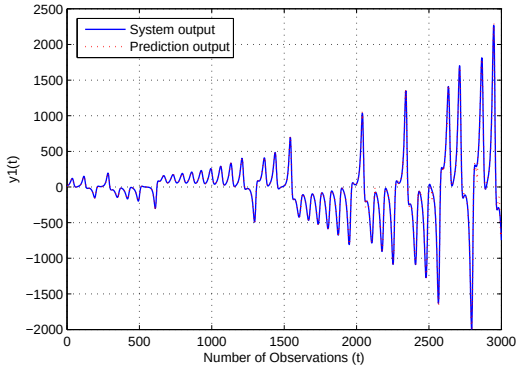


Fig. 12. **Lorenz time series (time base drift)**: on-line prediction with 40-steps ahead by the MRLS-QPSO.

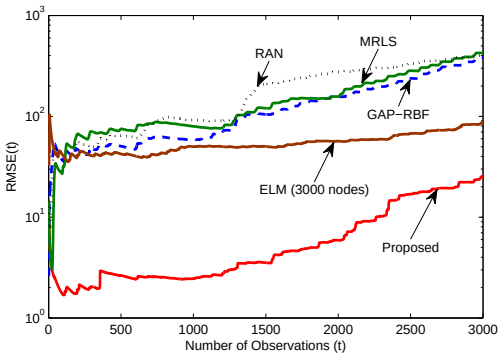


Fig. 13. **Lorenz time series (time base drift)**: RMSE learning curves.

to sequential learning with neural networks," *Neural Computation*, vol. 5, no. 6, pp. 954–975, Nov. 1993.

[16] 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.

[17] L. Yingwei, 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.

[18] ——, "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.

[19] G.-B. 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.

[20] ——, "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.

[21] 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.

[22] 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.

[23] S. Chen, E. S. Chng, and K. Alkadhimi, "Regularized orthogonal least squares algorithm for constructing radial basis function networks," *International Journal of Control*, vol. 64, no. 5, pp. 829–837, 1996.

[24] S. Chen, "Locally regularised orthogonal least squares algorithm for the construction of sparse kernel regression models," in *Signal Processing, 2002 6th International Conference on*, vol. 2, Aug. 2002, pp. 1229–1232.

[25] X. Hong, P. Sharkey, and K. Warwick, "Automatic nonlinear predictive model-construction algorithm using forward regression and the press statistic," *Control Theory and Applications, IEE Proceedings -*, vol. 150, no. 3, pp. 245 – 254, May. 2003.

[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] S. Chen, X. Hong, and C. Harris, "Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 4, pp. 1708 –1717, Aug. 2004.

[28] S. Chen, X. Hong, B. L. Luk, and C. J. 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] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural*

[6] Y. Zhong, K.-M. 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.

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

[8] 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.

[9] 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.

[10] M. Birgmeier, "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.

[11] 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.

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

[13] 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.

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

[15] V. Kadirkamanathan and M. Niranjan, "A function estimation approach

Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, Nov./Dec. 1995, pp. 1942 –1948.

[31] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363 – 371, Nov. 2002.

[32] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *Power Systems, IEEE Transactions on*, vol. 18, no. 3, pp. 1187 – 1195, Aug. 2003.

[33] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, Jul. 2004, pp. 985 – 990.

[34] ——, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489 – 501, Jul. 2006.

[35] N.-Y. Liang, G.-B. 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.

[36] G.-B. Huang, L. Chen, and C.-K. 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.

[37] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16-18, pp. 3460 – 3468, Oct. 2008.

[38] G. Feng, G.-B. 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.

[39] G.-B. 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.

[40] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. 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.

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

[42] J. Sun, W. Xu, and B. Feng, "A global search strategy of quantum-behaved particle swarm optimization," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1, Dec. 2004, pp. 111 – 116.

[43] 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.

[44] 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.

[45] 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.

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

[47] X. Hong, R. Mitchell, S. Chen, C. J. Harris, K. Li, and G. Irwin, "Model selection approaches for nonlinear system identification: a review," *International Journal of Systems Science*, vol. 39, no. 10, pp. 925–946, Oct. 2008.

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