ON-LINE IDENTIFICATION OF NON-LINEAR SYSTEMS USING ADAPTIVE RBF-BASED NEURAL NETWORKS

K. Salahshoor, Ph.D.
Department of Automation
and Instrumentation, Petroleum University
of Technology, Tehran, I. R. of Iran
email: salahshoor@put.ac.ir

M. R. Jafari, M.S.
Department of Automation
and Instrumentation, Petroleum University
of Technology, Tehran, I. R. of Iran
Corresponding Author:
email:mrjafari@gmail.com

Abstract- This paper extends the sequential learning algorithm strategy of two different types of adaptive radial basis function-based (RBF) neural networks, i.e. growing and pruning radial basis function (GAP-RBF) and minimal resource allocation network (MRAN) to cater for on-line identification of non-linear systems. The original sequential learning algorithm is based on the repetitive utilization of sequential input-output data in order to accomplish the training phase.

Some interesting modifications have been proposed in the growing and pruning neurons criteria of the original GAP-RBF neural network to make the resulting modified GAP-RBF (MGAP-RBF) neural network suitable for on-line system identification applications. The Unscented Kalman Filter (UKF) has been proposed as a new learning algorithm to update the parameters of MRAN, GAP-RBF and MGAP-RBF neural networks. Moreover, to keep the resulting parameter estimation routines more sensitive to track any possible time-varying system dynamics, a variable forgetting factor strategy has been included in the UKF learning algorithm. The proposed identification algorithms have been tested on a nonisothermal continuous stirred tank reactor (CSTR) and the chaotic Mackey Glass time-series as two different benchmark problems. The resulting performances of the MRAN, GAP-RBF and the proposed MGAP-RBF neural networks being estimated with the extended Kalman filter (EKF) or the UKF learning algorithm have been evaluated for comparison purposes. Simulation results show the superiority of the proposed MGAP-RBF neural network estimated with the UKF learning algorithm.

Keywords: Adaptive On-Line Identification, GAP-RBF, MRAN, EKF, UKF.

INTRODUCTION

Many engineering applications need a suitable model structure, within which a good model is to be found, for a compact and accurate description of the dynamic behavior of the system under consideration. These models are used for simulations, predictions, analysis of the system's behavior, design of model-based controllers, and so forth.

Radial Basis Function (RBF) networks have been popularly used in many applications in recent times due to their ability to approximate complex non-linear mappings directly from the input-output data with a simple topological structure and ease of implementation of dynamic and adaptive network architectures.

Several learning algorithms, such as resource allocation networks (RAN), RAN-EKF, and MRAN, have been proposed in the literature for training RBF networks. In practical on-line applications, sequential learning algorithms are generally preferred over batch learning algorithms as they do not require retraining whenever new data is received. However, selection of a learning algorithm for an on-line application is critically dependent on its accuracy and speed. In recent years, different methods [1,3,8] have been proposed in which sequential learning algorithms are used.

Sequential learning algorithms have the following distinguishing features:

- The input-output data are used sequentially (*one-by-one*) to train the learning system;
- At any sample time instant, *only one* input-output data can be observed and used by the learning system;
- The learning system *discards* any observed input-output data as soon as the learning procedure is completed;
- The learning system has no *prior* knowledge as to how many total training observations will be presented.

A significant contribution to sequential learning was made by Platt [10] by proposing RAN in which hidden neurons were added sequentially based on the novelty of the new observed data. Enhancement of RAN, known as RAN-EKF, was proposed by [5] in which EKF rather than least mean squares (LMS) algorithm was used for updating the network parameters (centers, widths, and weights of Gaussian neurons) to improve the accuracy and obtain a more compact network. RAN and RAN-EKF can only add neurons to the network and can not prune insignificant neurons from the network. As a consequence, the networks obtained by RAN and RAN-EKF could become extremely large in some large-scale complex applications.

A significant improvement to RAN and RAN-EKF was made by [18,19] by introducing a pruning strategy based on the relative contribution of each hidden neuron to the overall network output. The resulting network referred to as MRAN has been used in a number of applications [13]. Other pruning schemes in the RBF networks have been proposed in [11,12].

Similar to the other sequential learning algorithms, MRAN accepts input-output data one-by-one. However to decide on the growing and pruning neurons mechanisms at each sample time instant, it utilizes a sliding data window to identify the neurons that contribute relatively little to the network output based on the input data distribution

received in the window.

One important point to be noted in all of these sequential algorithms is that they do not link the required learning accuracy directly to the algorithm. Instead, they all have various thresholds which have to be selected using exhaustive trial and error studies. The issue of specifying various thresholds based on the needed accuracy has been raised in MRAN for determining the inactive neurons.

Huang, et al. [1] proposed a simple sequential growing and pruning algorithm based on the relationship between the significance of a neuron and the required learning accuracy for RBF networks, referred to as GAP-RBF. Conceptually speaking, the significance of a neuron is defined as the overall contribution of that neuron to the network output averaged over all the input data received so far.

This paper focuses on the GAP-RBF and MRAN neural network to cater for on-line identification of non-linear systems. The usual EKF learning algorithm has been upgraded to the UKF learning algorithm. An exponential forgetting factor strategy has been included in the UKF learning algorithm in order to enable it to track any possible time-varying dynamic changes. To make the neurons growing and pruning procedure more adapted to the on-line identification applications, the growing and pruning criteria have been modified.

The rest of this paper is organized as follows: First MRAN and GAP-RBF neural networks will be described. After that the EKF and UKF learning algorithms will be presented. This will be followed by a description of the problems in the original GAP-RBF neural network and presentation of a modified GAP-RBF neural network. Later, the performance comparison results for the original GAP-RBF and its modified version together with the MRAN using the EKF and UKF learning algorithms will be presented on two benchmark problems. The paper will close with a summary of results and conclusions.

MRAN AND GAP-RBF NEURAL NETWORKS

In this section, the main ideas behind the MRAN and GAP-RBF neural networks are described.

- MINIMAL RESOURCE ALLOCATION NEURAL NETWORK (MRAN):

MRAN is based on the Gaussian RBF neural networks. The output of a Gaussian RBF network with K hidden neurons can be described as follows:

$$f(x_n) = \sum_{k=1}^{K} \alpha_k \phi_k(x_n) \tag{1}$$

where x_n is the input vector of the network, α_k is the connecting weight of the kth hidden neuron to the output neuron, and $\phi_k(x_n)$ denote a response of the kth hidden unit to the input vector x_n, defined by the following Gaussian function:

$$\phi_k(x_n) = \exp\left(-\frac{\|x_n - \mu_k\|^2}{\sigma_k^2}\right) \tag{2}$$

where μ_k and σ_k refer to the center and width of the kth hidden neuron respectively, and $\|.\|$ indicates the Euclidean norm.

In sequential learning, a series of training samples (x_n,y_n) , n = 1, 2, . . . are randomly drawn and presented one-by-one to the network.

The learning process of MRAN involves the allocation of new hidden neurons as well as adaptation of network parameters. Its learning begins with no hidden neurons. As an input-output data (x_n,y_n) is received, MRAN learning algorithm conducts the following two basic growing and pruning steps:

1. Growing Step: The following three error criteria are first checked to decide on allocation of a new hidden neuron:

$$\|e_n\| = \|y_n - f(x_n)\| > e_{\min} \tag{3}$$

$$\varepsilon_n < d_n = \begin{cases} \min_k \|x_n - \mu_{nr}\| & (k \neq 0) \\ \varepsilon_{\max} & (k = 0) \end{cases} \tag{4}$$

$$e_{rms_n} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n} \|e_i\|^2}{M}} > e'_{\min} \tag{5}$$

where μ_{nr} is the center of the hidden neuron which is closest to current input x_n. $e_{\min}$, ε_n and $e'_{\min}$ are thresholds to be selected appropriately. $e_{\min}$ is used to determine if the existing nodes are insufficient to obtain a network output and is an instantaneous error check. ε_n ensures that the new neuron is sufficiently far from all the existing neurons and is given by $\varepsilon_n = \max\{\varepsilon_{\max}\gamma^n, \varepsilon_{\min}\}$ where γ is a decay constant ($0<\gamma<1$). Equation (5) is an additional MRAN condition, compared with RAN and RAN-EKF, which is introduced to ensure the neuron growth smooth. e_i is the network output error at ith instant and $e'_{\min}$ is the threshold selected to check whether the network meets the required sum squared error specification over a sliding window size (M) in the past.

If all three error criteria in the growing step are satisfied, a new hidden neuron is allocated with the following initial parameters to remove error:

$$\begin{aligned} \alpha_{K+1} &= e_n \\ \mu_{K+1} &= x_n \\ \sigma_{K+1} &= \kappa d_n \end{aligned} \tag{6}$$

where κ is an overlap factor which determines the overlap of the hidden neurons responses with the neighbor hidden units in the input space. The dimensionality of the covariance matrix (P_n) in the EKF or UKF learning algorithm is also adjusted by adding new rows and columns related to the number of new parameters introduced by the new hidden neuron.

Else

The network parameters ($\alpha_k, \mu_k, \sigma_k$; $k = 1,..., K$) are updated using the EKF or UKF learning algorithms. (EKF and UKF algorithms are described in the Section entitled, "Extended Kalman Filter and Unscented Kalman Filter").

Endif

2. Pruning step: The updated network is finally checked to prune the redundant hidden units that have minimal contribution to the network outputs for M consecutive observations in the sliding data window. To accomplish this objective, the contribution of all the available hidden units (k=1,..., K) are first calculated on the network outputs (j=1,...,J) as follows:

$$o_{kj}(x_n) = \alpha_{kj}\phi_k(x_n) \tag{7}$$

The highest absolute values of these contribution values are:

$$o_{\max}(x_n) = \left[\max_k |o_{k_1}|, \quad ..., \quad \max_k |o_{k_J}| \right]^T \tag{8}$$

Now, the contribution values on each network output are normalized to the highest absolute values as follows:

$$r_{kj} = \left| o_{kj} \Big/ o_{\max_j} \right| \quad ; \qquad k = 1,..., K; j = 1, \quad ..., \quad J \tag{9}$$

If the normalized contribution value of the kth hidden unit on the jth network output (r_{kj}) falls below a threshold for M consecutive observations, the kth hidden unit is regarded as a redundant unit and is pruned. Then the dimensionality of the covariance matrix (P_n) in the EKF or UKF learning algorithm is adjusted by removing rows and columns which are related to the pruned unit.

- GAP-RBF NEURAL NETWORK

GAP-RBF neural network uses the same Gaussian RBF network structure as discussed for MRAN. Consequently, its network performance can be described by Eqs. (1) and (2). However, its growing and pruning algorithm is derived based on a notion of significance for the hidden neurons (instead of the root mean square criterion in MRAN). During the sequential learning process of GAP-RBF, a series of training samples (x_n, y_n), n =1,2,... are randomly drawn from a range X with a sampling density function of $P(X)$ and presented one-by-one to the network. Each training sample would

trigger the action of adding a new hidden neuron, pruning the nearest hidden neuron, or adjusting the parameters of the nearest hidden neuron, based on only the significance of the nearest hidden neuron to the training sample. This is in contrast with the MRAN learning algorithm in which all the neurons will be checked for adding, pruning and adjusting purposes. This results in reducing the overall computations and thereby increasing the learning speed. The significance of the kth hidden neuron is defined as [2]:

$$E_{sig}(k) = \left| \frac{(1.8\sigma_k)^l \alpha_k}{S(X)} \right| \tag{10}$$

where l is the dimension of the input space ($x \in \Re^l$), and $S(X)$ denotes the estimated size of the range X where the training samples are drawn from.

In [2] for both growing and pruning, it is shown that one needs to check only the nearest neuron based on the following Euclidean distance to the current input data x_n for its significance:

$$\|x_n - \mu_{nr}\| = \min_k(\|x_n - \mu_k\|) \qquad ,k = 1,...,K \tag{11}$$

where μ_{nr} is the center of the hidden neuron which is nearest to x_n.

The learning process of GAP-RBF begins with no initial hidden neurons similar to MRAN. As new observation data (x_n, y_n) are received during the training, some of them may initiate new hidden neurons based on the growing criteria. However, the newly added neuron may have insignificant contribution to the overall performance of the whole network, and hence this neuron should not be added at all.

Therefore, GAP-RBF uses the following enhanced growing criterion for each new observation data (x_n, y_n) to prevent adding insignificant neuron leading to a smooth growing process:

$$\begin{cases} \|x_n - \mu_{nr}\| > \varepsilon_n \\ |e_n| > e_{\min} \\ \dfrac{(1.8\kappa)^l \|x_n - \mu_{nr}\|^l |e_n|)}{S(X)} > e_{\min} \end{cases} \tag{12}$$

where x_n is the latest input received, μ_{nr} is the center of the hidden neuron nearest (in the Euclidean distance) to x_n. $e_{\min}$ is the desired approximation accuracy and ε_n is a threshold to be selected appropriately. If the growing criteria (12) are satisfied for a new observation, a new significant neuron K+1 will be added and the parameters associated with the new hidden neurons are taken as follows:

$$\begin{cases} \alpha_{K+1} = e_n \\ \mu_{K+1} = x_n \\ \sigma_{K+1} = \kappa \|x_n - \mu_{nr}\| \end{cases} \tag{13}$$

where $e_n = y_n - f(x_n)$. In this case, all the other present neurons ($k = 1, \ldots, K$) will remain as significant and their parameters will be unchanged. Thus, pruning checking need not be done after a new neuron is added.

However, if a new observation (x_n, y_n) arrives and the growing criteria (12) is not satisfied, no new neuron will be added and only the parameters of the nearest neuron $(\alpha_{nr}, \mu_{nr}, \sigma_{nr})$ will be adjusted using the EKF or UKF learning algorithms. Then, the significance of the nearest (most recently adjusted) neuron is checked via the following pruning criterion:

$$E_{sig}(nr) = \left| \frac{(1.8\sigma_{nr})^l \alpha_{nr}}{S(X)} \right| < e_{\min} \tag{14}$$

If the average contribution made by the nearest neuron in the whole range X is less than the expected accuracy $e_{\min}$, it is taken as insignificant and should be removed. As discussed, at any time instant, only the single nearest neuron needs to be adjusted or needs to be checked for growing and pruning.

The complete description of the GAP-RBF learning algorithm [2] can be summarized as follows:

Given an expected desired accuracy $e_{\min}$ for each observation data (x_n, y_n), where $x_n \in \Re^l$, do the following steps:

1. **Compute** the overall network output:

$$f(x_n) = \sum_{k=1}^{K} \alpha_k \exp\left(-\frac{1}{\sigma_k^2} \|x_n - \mu_k\|^2 \right) \tag{15}$$

where K is the number of hidden neurons.

2. **Calculate** the parameters required in the growth criterion:

$$\begin{aligned} &\varepsilon_n = \max\{\varepsilon_{\max}\gamma^n, \varepsilon_{\min}\}, \qquad (0 < \gamma < 1) \\ &e_n = y_n - f(x_n) \end{aligned} \tag{16}$$

where $\varepsilon_{\min}$ and $\varepsilon_{\max}$ are minimum and maximum distance thresholds, respectively.

3. **Apply** the growth criterion for adding neurons:

If $|e_n| > e_{\min}$ and $\|x_n - \mu_{nr}\| > \varepsilon_n$ and $(1.8\kappa \|x_n - \mu_{nr}\|^l)|e_n| / S(X) > e_{\min}$

Allocate a new hidden neuron $K + 1$ with:

$$\begin{aligned} &\alpha_{K+1} = e_n \\ &\mu_{K+1} = x_n \\ &\sigma_{K+1} = \kappa \|x_n - \mu_{nr}\|. \end{aligned} \tag{17}$$

Else

Adjust the network parameters $\alpha_{nr}, \mu_{nr}, \sigma_{nr}$ for the nearest neuron only, using the

EKF or UKF algorithm.

4. Check the pruning criterion for the nearest (nrth) hidden neuron:

If $\left|(1.8\sigma_{nr})^{l}\alpha_{nr}/S(X)\right| < e_{\min}$, **remove** the nearest (nrth) hidden neuron and **do** the necessary changes in the EKF or UKF algorithm.

Endif

Endif

EXTENDED KALMAN FILTER AND UNSCENTED KALMAN FILTER

- EXTENDED KALMAN FILTER (EKF)

The Kalman filter (KF) in its standard form is a popular choice for any linear stochastic estimation problem. However, in most real applications of interest, the system dynamics and observation equations are non-linear and hence suitable modifications to the standard kalman filter are required. The extended kalman filter (EKF) provides these modifications by linearizing the non-linear process and observation models around the last state estimate. In this way, the EKF gives an approximation of the optimal estimate and hence can be considered as the most common and popular approach for both non-linear state estimation and parameter estimation (e.g. learning the weights of a neural network).

Let the process be estimated and the associated observation relationship be described by the following non-linear state-space model:

$$\begin{aligned} x_{k+1} &= f\ (x_k, u_k) + w_k \\ y_k &= h\ (x_k) + v_k \end{aligned} \tag{18}$$

where x_k represents the unobserved state of the system, u_k is a known exogenous input, and y_k is the observed measurement output. The process noise w_k drives the dynamic system, and the observation noise is given by v_k.

The EKF involves the following recursive estimation of the mean ($\hat{x}$) and error covariance (P) of the state estimate under a Gaussian assumption:

1. Consider the last estimated state $\hat{x}(k \mid k)$.
2. Linearize the non-linear system dynamics $x_{k+1} = f\ (x_k, u_k) + w_k$ around $\hat{x}(k \mid k)$.
3. Apply the prediction cycle of the kalman filter algorithm to the linearized system dynamics in order to calculate a prior state estimate $\hat{x}(k+1 \mid k)$ and a prior estimate of the error covariance matrix $P\ (k+1|k)$.
4. Linearize the non-linear measurement dynamics $y_k = h\ (x_k) + v_k$ around $\hat{x}(k+1 \mid k)$.
5. Apply filtering or update cycle of the kalman filter algorithm to the linearized measurement dynamics in order to calculate the posteriori state estimate $\hat{x}(k+1 \mid k)$ and the posteriori estimate of error covariance matrix $P(k+1|k)$.

Let $F(k)$ and $H(k)$ be the Jacobian matrices of the non-linear process $f(.)$ and observation $h(.)$ models around the estimated state, denoted by:

$$F(k) = \frac{\partial f(x,u)}{\partial x}\Big|_{\hat{x}(k|k),u_k}$$
$$H(k+1) = \frac{\partial h(x)}{\partial x}\Big|_{\hat{x}(k+1|k)} \quad (19)$$

The EKF works almost like a standard KF, except for F and H, which vary in time based on the estimated state $\hat{x}$. Its actual algorithm can be stated in terms of the following two distinct cycles:

Predict Cycle

Predict next state, before measurements are taken:

$$\hat{x}(k+1|k) = f\ (\hat{x}(k|k), u_k)$$
$$P(k+1|k) = F(k)P(k|k)F^T(k) + Q(k) \quad (20)$$

Update Cycle

Update state, after measurements are taken:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)\left[y_{k+1} - h\ (\hat{x}(k+1|k))\right]$$
$$K(k+1) = P(k+1|k)H^T(k+1)\left[H(k+1)P(k+1|k)H(k+1)^T + R(k+1)\right]^{-1} \quad (21)$$
$$P(k+1|k+1) = \left[I - K(k+1)H(k+1)\right]P(k+1|k)$$

where Q and R are covariance matrices, describing the second order properties of the state and measurement noise (i.e., $Q = E(ww^T)$ and $R = E(vv^T)$ where $E(.)$ denotes the expected operator). K denotes the kalman gain matrix, used in the update observer.

- UNSCENTED KALMAN FILTER (UKF)

The EKF has two important potential drawbacks. First, the derivation of the Jacobian matrices, that is the linear approximators to the non-linear functions, can be complex causing implementation difficulties. Second, this linearization can lead to filter instability if the estimation time-step intervals are not sufficiently small [3].

To address these limitations, Julier and Uhlmann [3,4] developed the UKF. Therefore, it was extended to parameter estimation applications by Wan and Van der Merwe [14,16].

Instead of linearizing these non-linear functions using Jacobian matrices (Eq. (19)), the UKF uses a "deterministic sampling" approach to calculate the mean and covariance estimates of Gaussian random state variables (i.e. x) with a minimal set of $2L+1$ sample points (L is the state dimension), called as sigma points [3,15,16], through the actual non-linear functions. This approach yields more accurate results compared to ordinary functions linearization in the EKF.

The results are accurate to the third order (Taylor series expansion) for Gaussian inputs for all non-linearities. For non-Gaussian inputs, the results are accurate to at least the second order [3]. Whereas, the linearization approach of the EKF results only in the first order accuracy.

The UKF algorithm can be implemented by the following steps:

1- The algorithm is started with some initial guesses for the state estimate (x_0) and the error covariance matrix (P_0), defined as:

$$\hat{x}_0 = \mathrm{E}[x_0]$$
$$P_0 = \mathrm{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \tag{22}$$

where E[.] denotes the expected value.

The UKF algorithm is followed the recursive application of the "deterministic sampling" to the non-linear system equations (Eq. (18)) for other time instants $k \in \{1,...,\infty\}$ as follows:

2- It calculates a collection of sigma points, stored in the columns of the $L \times (2L+1)$ sigma point matrix χ_{k-1} where L is the dimension of the state vector as:

$$\chi_{k-1} = \left[\hat{x}_{k-1} \quad \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} \quad \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}} \right] \tag{23}$$

where $\lambda = \alpha^2(L+\kappa) - L$ and $\gamma = \sqrt{L+\lambda}$ are scaling parameters. The constant α determines the spread of the sigma points around $\hat{x}$ and κ is a secondary scaling parameter.

3- Propagate each column of χ_{k-1} ($\chi_{i,k-1}$) through the non-linear system state equation $f(x_k, u_k)$ to perform the prediction or time-update step as:

$$\chi^*_{i,k|k-1} = f\left[\chi_{i,k-1}, u_{k-1}\right] \qquad ;i = 0, \ldots, 2L \tag{24}$$

Then, the a priori estimate values for state and error covariance are calculated as:

$$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \chi^*_{i,k|k-1} \tag{25}$$

$$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} \left[\chi^*_{i,k|k-1} - \hat{x}_k^-\right] \left[\chi^*_{i,k|k-1} - \hat{x}_k^-\right]^T + R^w \tag{26}$$

where $\{W_i\}$ is a set of scalar weights as follows: $(W_0^{(m)} = \lambda/(L+\lambda))$, $W_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta), W_i^{(m)} = W_i^{(c)} = 1/2(L+\lambda)$; $i = 1,\ldots,2L$, β is a parameter used to incorporate any prior knowledge about the distribution of X and R^w is the process noise covariance matrix.

The sigma points are updated using a priori estimate values of the state ($\hat{x}_k^-$) and error covariance (P_k^-):

$$\chi_{k|k-1} = \left[\hat{x}_k^- \quad \hat{x}_k^- + \gamma\sqrt{P_k^-} \quad \hat{x}_k^- - \gamma\sqrt{P_k^-} \right] \tag{27}$$

Then, each column of $\chi_{k|k-1}$ is propagated through non-linear system observation equation $h(x_k)$ to predict the measurement values ($\hat{y}_k^-$) as:

$$\mathrm{Y}_{k|k-1} = h[\chi_{k|k-1}] \qquad ; i = 0, \ldots, 2L$$

(28)

$$\hat{y}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathrm{Y}_{i,k|k-1} \tag{29}$$

4- after the prediction step, the correction or measurement update step is performed to calculate the posteriori estimate state as follows:

$$\hat{x}_k = \hat{x}_k^- + \mathrm{K}_k (y_k - \hat{y}_k^-) \tag{30}$$

where y_k is the actual measurement vector and K_k is kalman gain defined by:

$$\mathrm{K}_k = P_{x_k y_k} P_{y_k y_k}^{-1} \tag{31}$$

where

$$P_{x_k y_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\chi_{i,k|k-1} - \hat{x}_k^-\right] \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T \tag{32}$$

$$P_{\tilde{y}_k \tilde{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right] \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T + R^v \tag{33}$$

where R^v is the measurement noise covariance matrix. Finally, the posteriori estimate of the error covariance is computed as:

$$P_k = P_k^- - \mathrm{K}_k P_{\tilde{y}_k \tilde{y}_k} \mathrm{K}_k^T \tag{34}$$

THE MODIFIED GAP-RBF NEURAL NETWORK AND THE UKF LEARNING ALGORITHM

Some desired modifications have been proposed in the growing and pruning neurons criteria to make the resulting MGAP-RBF neural network suitable for on-line system identification applications. The UKF learning algorithm with a variable forgetting factor scheme has been proposed as a new learning algorithm to update the free parameters of the MRAN, GAP-RBF and MGAP-RBF neural networks. The proposed modifications can be described in the following two sections.

- THE MODIFIED GAP-RBF NEURAL NETWORK

In order to have smoothly output response and avoid oscillation, the mechanism of adding and pruning should be allowed to change smoothly. The rate of adding or pruning of neurons can be controlled with threshold values of e_n and ε_n. Selection of these values depends on the complexity of the system, input data for identification and the required accuracy for the model.

But the most important and effective factor is the persistent excitation (PE) property of the input data. If the input data have enough degree of PE, smooth and accurate

output can be obtained with suitable adjustment of the threshold values; but, if the inputs do not possess PE property, which may occur for instance in the case of closed-loop identification, then tuning the threshold values can not help and hence some desired modifications on the growing and pruning criteria will be necessary.

In on-line applications, identification usually starts with not exact prior knowledge about the network structure and parameters. Thus, it is a better approach to allow the identification algorithm to adapt its modeling process with an initial higher increase in its rate of neurons growth in order to improve such uncertain circumstances in the beginning as fast as possible. Then, as the identification process continues on and the input data is more prone to lose its richness property (PE), it would be logical to decrease the modeling sensitivity by lowering the rate of neurons growth.

However, evaluating the original GAP-RBF algorithm [2] and its application reported in [1,2,17] demonstrates that the neurons are added hardly in the start of the modeling process. This can cause large initial errors in the identification and the EKF or UKF learning algorithms may not be able to estimate parameters properly under such under-parameterized situation.

As the rate of neuron creation in GAP-RBF can be controlled by ε_n (Eq. (16)), the following exponential time-varying pattern is proposed to make a gradual evolution of ε_n from an initial higher sensitivity $\varepsilon_{\min}$ to a final lower sensitivity $\varepsilon_{\max}$:

$$\varepsilon_n = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min})(1 - e^{-n/\tau}) \tag{35}$$

where τ is the time-constant parameter that can be used to control the time rate evolution of ε_n.

Another problem is due to probable oscillation in the number of created neurons which can cause big errors in the identification results. This phenomenon can occur in on-line identification especially when the numbers of created neurons are small and the input data have small degree of PE, too. These detrimental effects can be improved by changing the pruning criterion (Eq. (14)) as follows:

$$\left|(1.8\sigma_{nr})^l \alpha_{nr} / S(X)\right| < \beta e_{\min} \tag{36}$$

in which a new pruning factor ($0 < \beta \leq 1$) has been added.

- THE MODIFIED UKF LEARNING ALGORITHM

In on-line identification, the estimation learning algorithm should be fast enough to adapt the identification model to any possible time-varying dynamic changes in the process.

The covariance matrix can be initialized with a large value. This option, however, causes rapid fluctuations in the initial neural network parameters estimates and hence

endangers the estimator convergence. Besides, choosing small initial covariance matrix will make the estimator adaption very slow. On the other hand, when the process dynamic changes, some of the previous estimation information will lose its accuracy as far as the new process dynamic is concerned. Thus, there should be a means of draining off old information at a controlled rate. One useful way of rationalizing the desired approach is to modify the covariance matrix update relationship (Eq. (34)) as follows:

$$P_k = (P_k^- - \mathrm{K}_k P_{\bar{y}_k \bar{y}_k} \mathrm{K}_k^T)/\eta_k \tag{37}$$

where η_k behaves as the forgetting factor concept in the usual recursive least squares (RLS) algorithm which undergoes the following time-varying evolution:

$$\eta_k = \eta_{k-1} + (1-\eta_{k-1})(1-e^{-1/\delta}) \qquad ,0 < \eta_k \le 1 \tag{38}$$

where t is the recursive time interval that is spent in the UKF learning algorithm to estimate the GAP-RBF neural network free parameters with fined structure. Thus, t is reset to zero when any network structural change, i.e., neuron creation or pruning, occurs. This scheme maintains a desired parameter adaptive capability in the UKF algorithm whenever process dynamics undergoes a time-varying change. Because, η_k starts with a lower initial value to accelerate the parameter estimation, its value is changed exponentially with a desired time-constant (δ) to a higher final value to assure the estimator convergence property.

SIMULATION CASE STUDIES

In this section, the identification capabilities of the modified GAP-RBF neural network with the modified UKF learning algorithm are tested on two benchmark problems. The resulting performances are compared with the original GAP-RBF and MRAN neural networks being estimated with both the EKF and UKF learning algorithms.

- NONISOTHERMAL CONTINUES TANK REACTOR (NONISOTHERMAL CSTR)

The process is a nonisothermal CSTR with an irreversible reaction ($\mathrm{A} \rightarrow \mathrm{B}$) which consists of two non-linear ordinary differential equations, given by [7]:

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{Af} - C_A) - k_0 C_A \exp(-\frac{E}{RT})\phi_c(t), \tag{39}$$

$$\frac{dT}{dt} = \frac{q}{V}(T_f - T) + \frac{(-\Delta H)k_0 C_A}{\rho C_p}\exp(-\frac{E}{RT})\phi_c(t) + \frac{\rho_c C_{pc}}{\rho C_p V} q_c \left[1 - \exp\left(-\frac{h_A}{q_c \rho C_{pc}}\phi_h(t)\right)\right] \times (T_{cf} - T) \tag{40}$$

where

$\phi_h(t)$ fouling coefficient $0 < \phi_h < 1$

$\phi_c(t)$ catalyst deactivation coefficient

C_A effluent concentration, *the controlled variable*

q_c coolant flow rate, *the manipulated variable*

q feed flow rate, *disturbance*

C_{Af} feed concentration

T_f feed temperature

T_{cf} coolant inlet temperature

The remaining model parameters together with the operating conditions are presented in Table 1.

Table 1: Nominal CSTR Operating Condition.

Nominal CSTR Operating Condition	
$q = 100l\,\min^{-1}$	$E/R = 9.95\times10^{3}K$
$C_{Af} = 1mol\,l^{-1}$	$-\Delta H = 2\times10^{5}cal.mol^{-1}$
$T_f = 350K$	$\rho, \rho_c = 1000gl^{-1}$
$T_{cf} = 350K$	$C_p, C_{pc} = 1cal\,g^{-1}K^{-1}$
$V = 100l$	$q_c = 103.41l\,\min^{-1}$
$h_A = 7\times10^{5}cal\,\min^{-1}K^{-1}$	$T = 440.2K$
$k_0 = 7.2\times10^{10}\,\min^{-1}$	$C_A = 8.36\times10^{-2}mol.l^{-1}$

The nonisothermal CSTR exhibits time-varying characteristics due to fouling phenomenon. Fouling occurs when a material is deposited on a heat transfer surface during the period of process operation. In practice, it is common for heat transfer surfaces to become contaminated with deposits and this causes additional resistance to the flow of heat. Linear fouling is a common approach to model the development of a fouling film over a period of time [7], which can be described by the following time-varying equation:

$$h_d = \phi_h(t)h = (1-\alpha_h t)h_A \quad (41)$$

where

t time,

h heat transfer coefficient, cleaned,

h_d heat transfer coefficient, scaled,

$\alpha_h = 0.01$ fouling constant.

Therefore, the heat transfer coefficient h defined in Eq. (40) is replaced by h_d. The open-loop step response of the CSTR process for a series of step changes in q_c is shown in Figure 1 for constant heat transfer coefficient ($\phi_h(t) = 1$). It is seen that the process is

highly non-linear. To demonstrate the effect of time-varying nature of the process due to the assumed time varying heat transfer coefficient (Eq. (41)), the previously mentioned open-loop step response tests, have been repeated; Figure 2 shows the resulting responses. As illustrated, the process is nonstationary and does not have a steady-state condition.

In order to be able to compare and analyze all the simulation outcomes, each simulation experiment was organized under the same conditions as follows:

During the first 5 time samples (minutes), the CSTR process was run at the nominal operating conditions given in Table 1. The initial free parameters were initialized at the following fined values (in all the identification algorithms):

$\varepsilon_{\max}=0.1$, $\varepsilon_{\min}=0.01$, $\kappa=0.85$, $\gamma=0.995$, $e_{min}=0.001$, $e'_{\min}=0.001$ and $M=50$.

Then, the process was deviated from the initial nominal operating condition via a sequence of multi-level step changes of +10%, -20%, +5% and finally another +5% in the coolant flowrate (q_c) as the manipulated variable to cover the process non-linearities over a wide range of operating conditions. A normally distributed random signal (mean =0.5, variance =1) was superimposed on each operating point level as shown in Figure 3 to enrich the identification data.

The process dynamics described in Eqs. (39) and (40), do not include any random process disturbances. However, in order to perform the simulation experiments under real circumstances, two independent white noises with variance of 0.01 have been added to the process dynamics as the process noises. Similarly, a white noise with variance of 0.01 has also been added to the process output (the effluent concentration (C_A)) as the measurement noise.

The complete identification simulation results for different performed experiments with various versions of MRAN and GAP-RBF algorithms, i.e.

MRAN-EKF	MRAN NN with EKF learning algorithm
GAP-RBF-EKF	GAP-RBF NN with EKF learning algorithm
MRAN-UKF	MRAN NN with UKF learning algorithm
GAP-RBF-UKF	GAP-RBF NN with UKF learning algorithm
MGAP-RBF-UKF	MGAP-RBF NN with UKF learning algorithm

have been summarized in Table 2 for concise comparison purposes. The results demonstrate the superiority of the proposed MGAP-RBF-UKF algorithm with respect to other algorithms in terms of the following two main objectives:

1. Network model accuracy in terms of Integral of Squared Error (ISE) measure.
2. Network model structure simplicity or compactness of number of neurons.

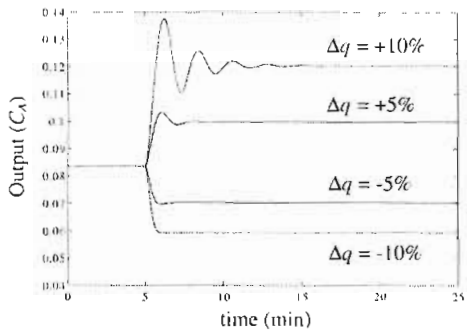

Figure 1: Open-loop step response with constant heat transfer coefficient.

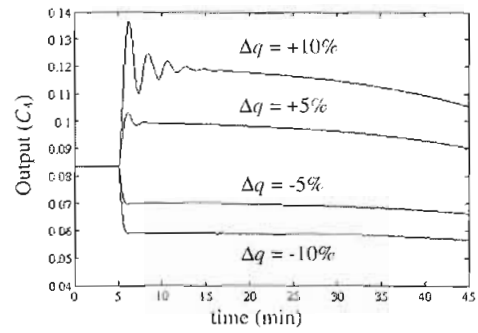

Figure 2: Open-loop step response with time-varying heat transfer coefficient (fouling effect).

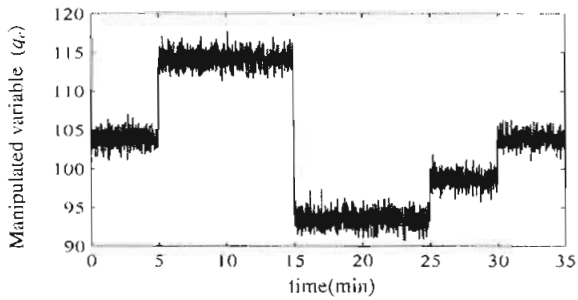

Figure 3: Noisy input sequence.

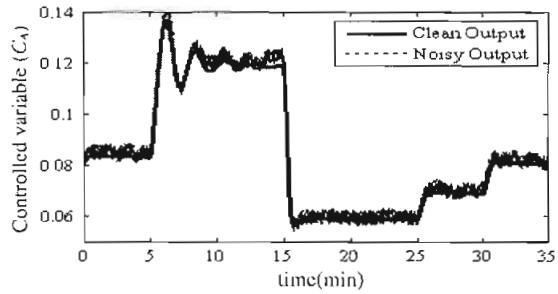

Figure 4: Real (noisy) and clean CSTR output.

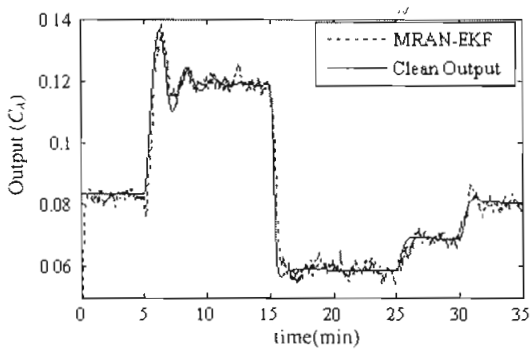

Figure 5: Output of MRAN with EKF algorithm.

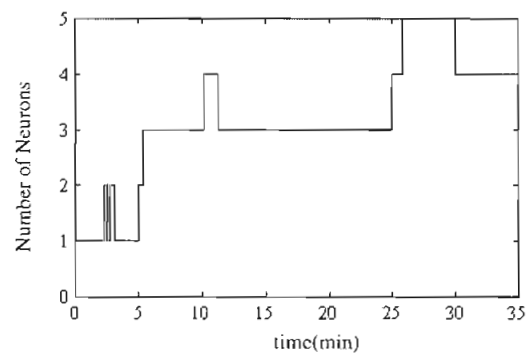

Figure 6: Number of neurons MRAN-EKF.

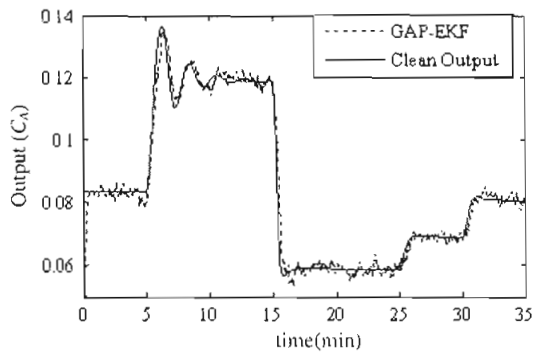

Figure 7: Output of GAP-RBF with EKF algorithm.

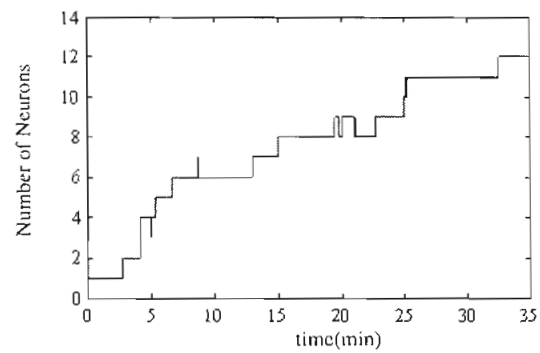

Figure 8: Number of neurons GAP-RBF-EKF.

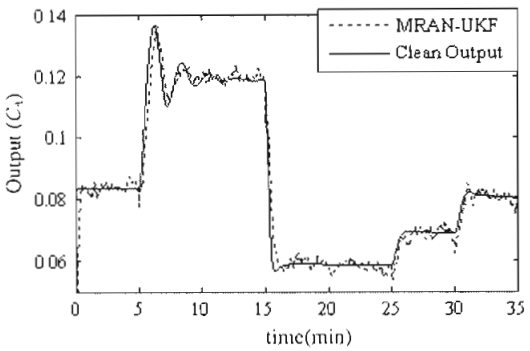

Figure 9: Output of MRAN with UKF algorithm.

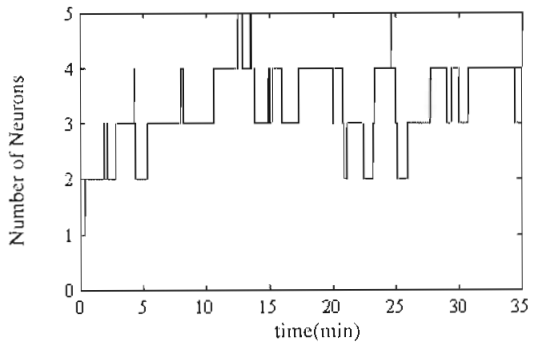

Figure 10: Number of neurons MRAN-UKF.

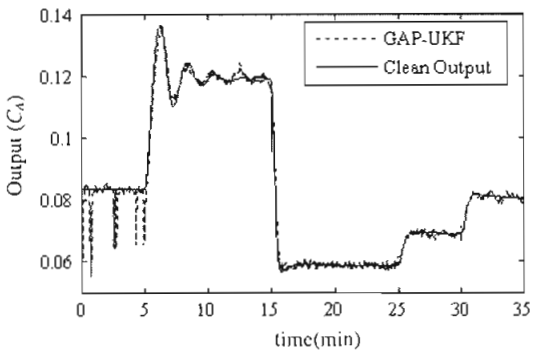

Figure 11: Output of GAP-RBF with UKF algorithm.

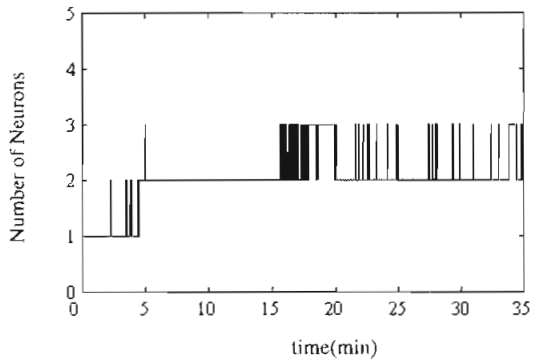

Figure 12: Number of neurons GAP-RBF-UKF.

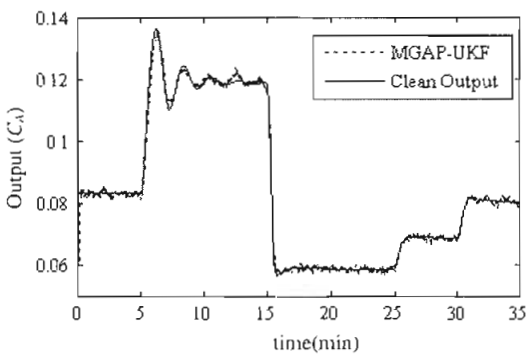

Figure 13: Output of MGAP-RBF with UKF algorithm.

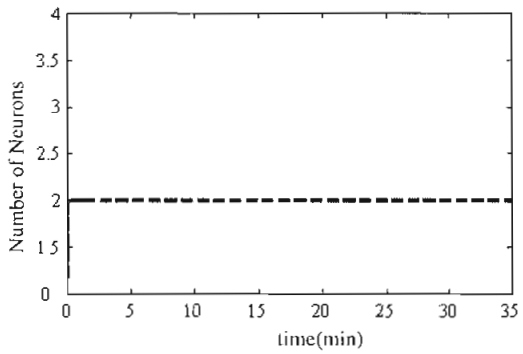

Figure 14: Number of neurons MGAP-RBF-UKF.

Table 2: Performance Comparison of Different Algorithms.

Type of algorithm	ISE	Resulting Performance	Figs.
MRAN-EKF	0.2972	starts with 1 neuron and terminates at 4, moderate fluctuations up to 5 neurons.	5,6
GAP-RBF-EKF	0.2057	starts with 1 neuron and terminates at 12, moderate fluctuations.	7,8
MRAN-UKF	0.2798	starts with 1 neuron and terminates at 3, high fluctuations up to 5 neurons.	9,10
GAP- RBF-UKF	0.1646	starts with 1 neuron and terminates at 2, very high fluctuations up to 3 neurons.	11,12
MGAP- RBF-UKF	0.0276	starts with 1 neuron and continues on with a constant 2 neurons.	13,14

- CHAOTIC MACKEY-GLASS TIME-SERIES BENCHMARK

It is very useful to examine the performance of the proposed identification algorithm (MGAP- RBF-UKF) to predict a chaotic time-series. The Mackey-Glass time-series has been introduced as a difficult benchmark problem in the literature exhibiting a very complex behavior, which may not be demonstrated by physical systems.

This time series can be generated by the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \tag{42}$$

where $a = 0.2$, $b = 0.1$, $\tau = 17$ and $x(0)=0.3$.

The identification experiments for this case study are configured to predict x at one-step ahead, based on the past values of x at 50, 56, 62 and 68 steps back. The difficulty in time-series prediction is increased by injecting a random Gaussian noise signal with a mean value of 0.1 and variance of 0.1 to the series $x(t)$. The identification procedure for all experiments done in this case study is run with an initial state of $x(0)=0.3$.

Similar to the previous case study, all the free GAP-RBF and MRAN parameters have been fixed as the following values:

$\varepsilon_{\max} = 0.1$, $\varepsilon_{\min} = 0.01$, $\kappa = 0.2$, $\gamma = 0.995$, $e_{min} = 0.001$, $e'_{\min} = 0.01$ and $M=50$.

As shown in Figures (15-24) the resulting identification time-series have been demonstrated by two separate time durations. Figures (15,17,19,21,23) illustrate the initial transient stages of the identification experiments. They have been included in order to show the well-behaved nature of the results, in the early on-line identification stages. Figures (16,18,20,22,24) show the 500 last stages of the identification experiments.

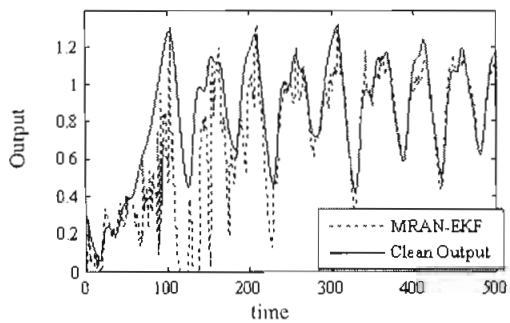

Figure 15: Initial Output of MRAN-EKF.

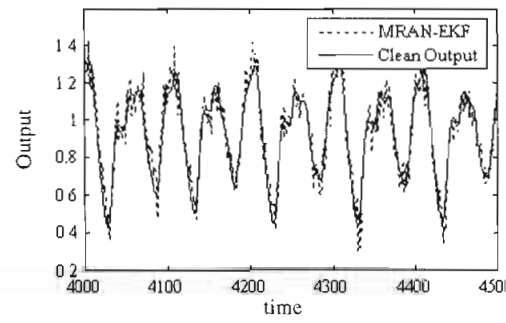

Figure 16: Final Output of MRAN-EKF.

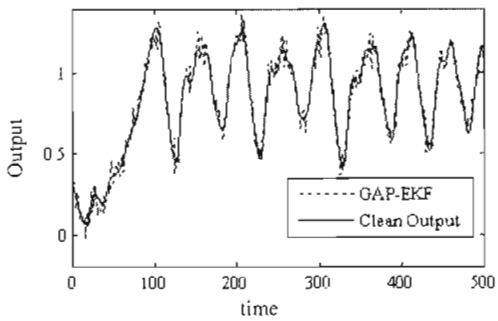

Figure 17: Initial Output of GAP-EKF.

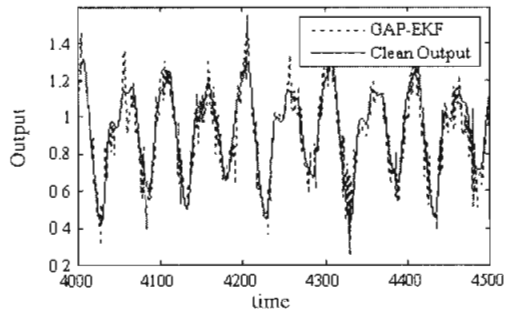

Figure 18: Final Output of GAP-EKF.

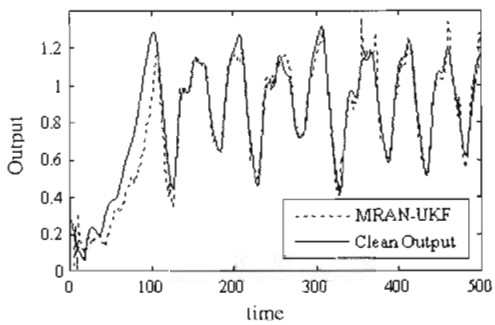

Figure 19: Initial Output of MRAN-UKF.

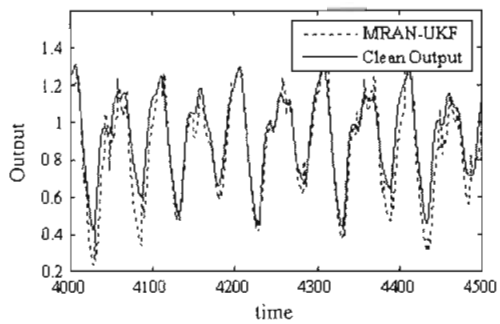

Figure 20: Final Output of MRAN-UKF.

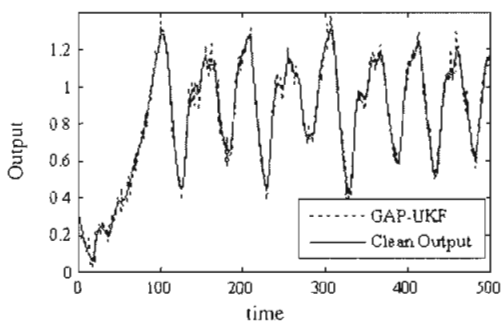

Figure 21: Initial Output of GAP-UKF.

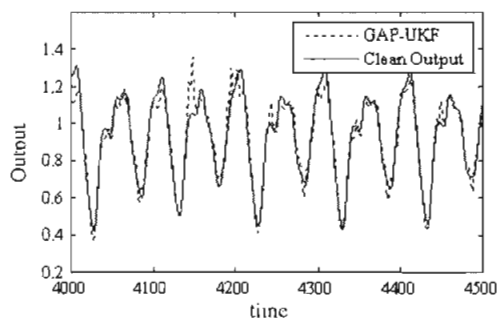

Figure 22: Final Output of GAP-UKF.

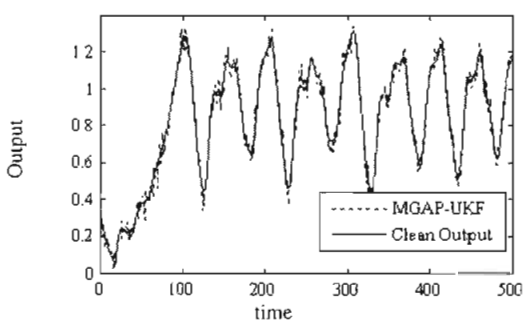

Figure 23: Initial Output of MGAP-UKF.

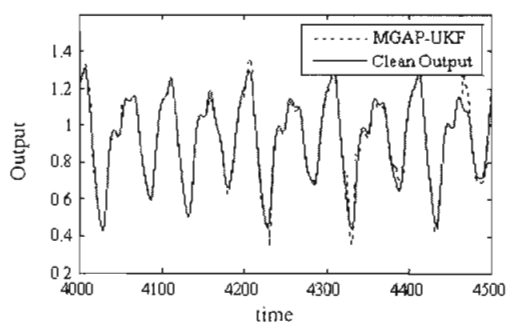

Figure 24: Final Output of MGAP-UKF.

Table 3 summarizes the prediction performance of different algorithms in terms of the ISE measure. It is seen that the proposed MGAP-RBF-UKF algorithm gives a

superior performance over other algorithms in terms of the network model accuracy. Figure 25 displays the neuron updating time history during the on-line learning identification. It can be observed that the proposed MGAP-RBF-UKF algorithm learns fast and obtains small network architecture with more stable and steady behavior.

Table 3: Performance Comparison of Different Algorithms.

Type of NN	ISE	Figs.
MRAN-EKF	50.2384	15,16
GAP-RBF-EKF	25.2522	17,18
MRAN-UKF	64.5485	19,20
GAP- RBF-UKF	31.3089	21,22
MGAP- RBF-UKF	14.6591	23,24

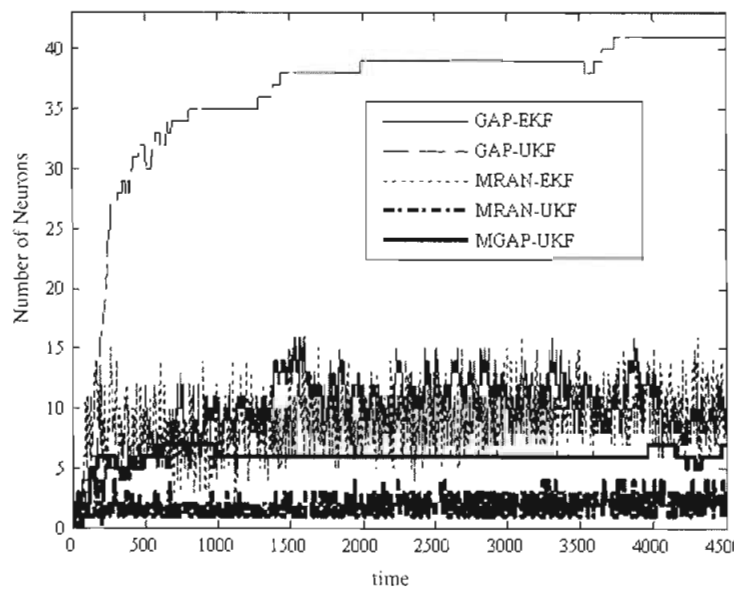

Figure 25: Time History of Neuron On-Line Updating For Different Algorithms.

CONCLUSIONS

In this paper, two different types of adaptive neural networks, i.e. GAP-RBF and MRAN, have been utilized for on-line identification of non-linear systems. The original versions of these neural networks have already been employed in off-line identification modes for function approximation and time-series prediction purposes using two distinct training and testing phases [1,2]. This paper has proposed some desired modifications in order to make the resulting MGAP-RBF neural network more suitable for on-line system identification purposes.

The basic contributions in this regard can be summarized as follows:

1. Modification in neuron creation criterion.

2. Modification in neuron pruning criterion.

The original GAP-RBF and MRAN neural networks are EKF-based in network parameter adjustment or learning algorithm. In this paper, the UKF, as an extension to the EKF, was used for updating the MRAN, GAP-RBF and MGAP-RBF network parameters.

The UKF learning algorithm was also modified to incorporate the forgetting factor concept, so as to increase its alertness to track any system dynamic changes efficiently.

The analysis of the resulting outcomes indicates the following observations:

1. The main flaw of the MRAN-based identification algorithms is the dependency of its learning accuracy and generalization performance on the growing-pruning window size, which is normally chosen on an exhaustive simulation trial and error basis.
2. The MRAN-based identification algorithms require higher computational complexity due to the fact that all the neuron parameters should be updated during each learning phase when no neuron is added or omitted.
3. The main benefit of the GAP-RBF-based identification algorithms is that any desired accuracy requirement can directly be introduced in the algorithm by defining a significance quality measure for the quality of each neuron.
4. The GAP-RBF-based identification algorithms have a lower degree of computational complexity in comparison with the MRAN-based ones due to the fact that only the nearest neuron parameters should be updated during each learning phase. However their tuning parameters are higher and more sensitive to the threshold initialization values.
5. The UKF learning algorithm leads to better accuracy of the results in comparison with the EKF one due to utilization of a deterministic sampling approach to calculate mean and covariance terms.
6. The UKF learning algorithm performs better in the face of contaminating noise. However, its computational complexity is higher.
7. The proposed MGAP-RBF neural network with the UKF learning algorithm using the developed forgetting factor strategy gives the best performance. This achievement can mainly be expressed in terms of the resulting accuracy and the network-model structure simplicity.

REFERENCES

[1] Huang G. B., et al., "A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Approximation." *IEEE Trans. Neural Networks,* Vol. 16, No. 1, 2005.

[2] Huang G. B., et al., "An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF) Networks." *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, Vol. 34, No. 6, 2004.

[3] Julier S. J. and Uhlmann J. K., "A New Extension of the Kalman Filter to Non-Linear Systems." in: *Proc. of AeroSense: The 11th Int. Symp. A.D.S.S.C.*, 1997.

[4] Julier S. J., et al., "A New Approach for Filtering Non-linear Systems." in: *Proceedings of the American Control Conference*, pp. 1628–1632, 1995.

[5] Kadirkamanathan V. and Niranjan M., "A Function Estimation Approach to Sequential Learning with Neural Networks." *Neural Computat.*, Vol. 5, pp. 954–975, 1993.

[6] Kazufumi I. and Kaiqi X., "Gaussian Filters for Non-linear Filtering Problems." IEEE Transactions on Automatic Control, Vol. 45, No. 5, pp. 910–927, 2000.

[7] Nikravesh M., "Dynamic Neural Network Control." Ph.D. Dissertation, University of South Carolina, Columbia, *SC*, 1994.

[8] Nishida K., et al., "An On-line Learning Algorithm with Dimension Selection Using Minimal Hyper Basis Function Networks." *SICE Annual Conference in Sapporo*, August, 2004.

[9] Norgaard M., et al., "Advances in Derivative-Free State Estimation for Non-linear Systems." *Tech. Rep. IMM-REP-1998-15*, Tech. Univ. of Denmark, 2000.

[10] Platt J., "A Resource-Allocating Network for Function Interpolation." *Neural Computat.*, Vol. 3, pp. 213–225, 1991.

[11] Rojas I., et al., "Time Series Analysis Using Normalized PG-RBF Network with Regression Weights." *Neurocomputing*, Vol. 42, pp. 267-285, 2002.

[12] Salmer´on M., et al., "Improved Ran Sequential Prediction Using Orthogonal Techniques." *Neurocomputing*, Vol. 41, pp. 153-172, 2001.

[13] Sundararajan N., et al., *Radial Basis Function Neural Networks with Sequential Learning: MRAN and its Applications.* River Edge: Singapore; World Scientific: NJ, 1999.

[14] Van der Merwe R., et al., "The Unscented Particel Filter." in: *Advances in Neural Information Processing Systems, 13*, 2001.

[15] Wan E. A. and van der Merwe R., "The Unscented Kalman Filter for Non-Linear Estimation." in: *Proc. of IEEE Symposium 2000 (AS-SPCC)*, Lake Louise, Alberta, Canada, October 2000.

[16] Wan E., ea al., "Dual Estimation and the Unscented Transformation." in: *Advances in Neural Information Processing Systems*, MIT Press, 12, pp. 666-672, 2000.

[17] Wang Y., et al., "Time Series Study of GGAP-RBF Network: Predictions of Nasdaq Stock and Nitrate Contamination of Drinking Water." *Proceedings of International Joint Conference on Neural Networks*, Montreal, Canada, July-

August, 2005.

[18] Yingwei L., et al., "A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function (RBF) Neural Networks." *Neural Computation*, Vol. 9, pp. 461-478, 1997.

[19] Yingwei L., et al., "Performance Evaluation of a Sequential Minimal Radial Basis Function (RBF) Neural Network Learning Algorithm." *IEEE Trans. Neural Networks*, Vol. 9, No. 2, pp. 308-318, 1998.