# On-line Identification of Non-Linear Systems Using an Adaptive RBF-Based Neural Network

Mohammad Reza Jafari, Tohid Alizadeh, Mehdi Gholami, Abdollah Alizadeh, Karim Salahshoor

***Abstract*— This paper extends the sequential growing and pruning radial basis function (GAP-RBF) to cater for on-line identification of non-linear systems. Some desired modifications on the growing and pruning neurons criteria have been proposed for the original GAP-RBF to make it more suitable for on-line identification. The unscented kalman filter (UKF) has been proposed as a new learning algorithm for GAP-RBF neural network. Moreover, a variable forgetting factor strategy has been included in the UKF algorithm to keep the parameter estimation routine more active in time-varying dynamics. Simulation results show the superiority of the modified GAP-RBF (MGAP-RBF) neural network being learned with the UKF algorithm.**

***Index Terms*— Adaptive on-line identification, GAP-RBF, MRAN, EKF, UKF.**

## I. 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 nonlinear mappings directly from the input-output data with a simple topological structure and ease of implementation of dynamic and adaptive network architecture. In practical on-line applications, sequential learning algorithms are generally preferred over batch learning algorithms as they do not require retraining whenever a 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 like [1], [2] and [3] have been proposed in which sequential learning algorithms are used. A significant contribution to sequential learning was made by Platt [4] by proposing RAN in which hidden neurons were added sequentially based on the novelty of the new data.

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. This issue of specifying the various thresholds based on the needed accuracy has been raised in MRAN for determining the inactive neurons, but the required learning accuracy was implemented in GAP-RBF [5] networks more clearly. The enhanced growing criterion proposed in the GAP-RBF algorithm will ensure that a neuron will not be added if it is insignificant to the overall performance of the network, even though it may make contribution to the single latest input data. GAP-RBF has much less computational complexity and requires less memory. Also GAP-RBF in comparison with RAN, RAN-EKF and MRAN has very few parameters for which one needs to select values for. The self-generating structure and low computational complexity of GAP-RBF make it one of the best candidates for on-line identification of complex systems with poor prior dynamic's knowledge.

This paper focuses on the GAP-RBF and MRAN algorithms. 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 for tracking any possible time-varying dynamic changes. To make the neurons growing and pruning procedure more adapted to the on-line identification applications, the corresponding criteria have been modified.

The rest of this paper is organized as follows. Section 2 describes GAP-RBF neural network briefly. Section 3 presents UKF learning algorithm. Section 4 describes the problems in the original GAP-RBF neural network and presents a modified GAP-RBF (MGAP-RBF) neural network. Section 5 presents performance comparison for the original GAP-RBF and its modified version on benchmark problems. Section 6 summarizes the resulting conclusions.

Manuscript received July 22, 2007.

M. R. Jafari is with the Automation and Instrumentation Department, Petroleum University of Technology, Tehran, Iran (e-mail: mrjafari@gmail.com).

T. Alizadeh is with the Automation and Instrumentation Department, Petroleum University of Technology, Tehran, Iran (e-mail: alizadeh_tohid@yahoo.com).

M. Gholami is with the Automation and Instrumentation Department, Petroleum University of Technology, Tehran, Iran (e-mail: meti.gholami@gmail.com).

A. Alizadeh is with Bonab Faculty of Engineering, University of Tabriz, Tabriz, Iran (e-mail: a_alizadeh@tabrizu.ac.ir).

K. Sahalshoor is with the Automation and Instrumentation Department, Petroleum University of Technology, Tehran, Iran (e-mail: selahshour@yahoo.com).

## II. GAP-RBF Neural Network

GAP-RBF uses the Gaussian RBF network (GRBF) model. The network output vector $f(x_n)$, which approximates the desired output vector $y_n$, are described as follows:

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

where $\alpha_k$ is it's connecting weight to the output neuron, and $\phi_k(x_n)$ denotes a response of the *k*th hidden unit to the inputs $x_n$ ; $\phi_k(x_n)$ is a Gaussian function given by:

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

where $\mu_k$ and $\sigma_k$ are the center and width of the Gaussian function respectively, and $\|.\|$ denotes the Euclidean norm.

The sequential learning algorithm for GAP-RBF networks, defines a notion of significance for the hidden neurons, and a growing and pruning criterions for on-line adjusting of the network size.

### A. GAP-RBF Learning Algorithm

During the sequential learning process, a series of training samples ( $x_i, y(x_i)$ ); (i = 1,2,...) are randomly drawn 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. The significance of the *k*th hidden neuron k is defined as [5]:

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

(3)

Given the set of all hidden neurons in A, the center of the nearest hidden neuron is $c_r$, the input is x and the nearest hidden neurons are defined as:

$$\left\{ \; c_r \mid (c_r \in A) \vee (\|x - c_r\| = \min_{i=1,\dots,K} (\|x - \mu_i\|)) \; \right\} \tag{4}$$

Like MRAN and EMRAN, the GAP-RBF network begins with no hidden neurons. As new observations are received during the training, some of them may initiate new hidden neurons according to the growing criterion. An enhanced growing criterion for new observation $(x_n, y_n)$, to make the growing process smooth is:

$$\begin{cases} \|x_n - c_r\| > \varepsilon_n \\ |e_n| > e_{\min} \\ \dfrac{(1.8\kappa\|x_n - c_r\|^l |e_n|)}{S(X)} > e_{\min} \end{cases} \tag{5}$$

If these conditions are satisfied, then a new K+1 neuron will be added and the parameters associated with the new hidden neuron are taken as follows:

$$\begin{cases} \alpha_{K+1} = e_n \\ c_{K+1} = x_n \\ \sigma_{K+1} = \kappa\|x_n - c_r\| \end{cases}$$

(6)

where $e_n = y_n - f(x_n)$ and $c_r$ is the centre of the nearest hidden neuron to $x_n$. As the training carries on, some of the new observations may trigger the learning algorithm to prune some hidden neurons.

The pruning criterion based on the new observation $(x_n, y_n)$ is:

$$E_{sig}(k) = \left| \frac{(1.8\sigma_k)^l \alpha_k}{S(X)} \right| < e_{\min} \tag{7}$$

where $S(X)$ is the estimated size of the input range.

If the average contribution made by the *k*th neuron in the whole range *X* is less than the expected accuracy $e_{min}$ and the *k*th neuron is insignificant; then the *k*th neuron can be removed. If no neuron adds or prunes, the nearest neuron will be adjusted with the EKF or UKF algorithm.

The complete description of the GAP-RBF learning algorithm [5] can be summarized as follows:

Given an approximation error $e_{min}$, for each observation $(x_n, y_n)$, where $x_n \in \Re^l$ , do

1. compute the overall network output

$$f(x_n) = \sum_{k=1}^{K} \alpha_k \exp\left( -\frac{1}{\sigma_k^2} \|x_n - \mu_k\|^2 \right) \tag{8}$$

where K is the number of hidden neurons.

2. calculate the parameters required in the growth criterion

$$\varepsilon_n = \max\{\varepsilon_{\max}\gamma^n, \varepsilon_{\min}\}, \qquad (0 < \gamma < 1)$$
$$e_n = y_n - f(x_n) \tag{9}$$

apply the 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

$$\alpha_{K+1} = e_n$$
$$\mu_{K+1} = x_n \tag{10}$$
$$\sigma_{K+1} = \kappa\|x_n - \mu_{nr}\|$$

Else

adjust the network parameters $\alpha_{nr}, \mu_{nr}, \sigma_{nr}$ for the nearest neuron only, using the EKF algorithm.
Check the criterion for pruning the hidden neuron:
*If* $\left|(1.8\sigma_{nr})^l \alpha_{nr} / S(X)\right| < e_{\min}$ , remove the nearest (*nr*th) hidden neuron and do the necessary changes in the EKF algorithm.

## III. Unscented Kalman Filter (UKF)

The inherent flaws of the EKF are due to its linearization approach for calculating the mean and covariance of random variables which undergoes a non-linear transformation. As shown in [2], [6] and [7], the UKF addresses these flaws by utilizing a deterministic "sampling" approach to calculate mean and covariance terms. This approach results in approximations that are accurate to the third order (Taylor series expansion) for Gaussian inputs for all nonlinearities.

For non-Gaussian inputs, approximations are accurate to at least the second-order [2]. Whereas, the linearization approach of the EKF results only in first order accuracy.

Consider the non-linear system, described by (11). The UKF algorithm can be derived by the following steps:

$$\begin{aligned} x_{k+1} &= f_k(x_k, u) + w_k \\ y_k &= h_k(x_k) + v_k \end{aligned} \tag{11}$$

1. Initialize with:

$$\begin{aligned} \hat{x}_0 &= \mathrm{E}[x_0] \\ P_0 &= \mathrm{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \end{aligned} \tag{12}$$

for $k \in \{1, \ldots, \infty\}$ , (E[.] denote the expected value).

2. Calculate the sigma points:

$$\chi_{k-1} = \left[ \hat{x}_{k-1} \quad \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} \quad \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}} \right] \tag{13}$$

3. Time update:

$$\overset{*}{\chi}_{k|k-1} = F\left[\chi_{k-1}, u_{k-1}\right] \tag{14}$$

$$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \overset{*}{\chi}_{i,k|k-1} \tag{15}$$

$$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} \left[\overset{*}{\chi}_{i,k|k-1} - \hat{x}_k^-\right] \left[\overset{*}{\chi}_{i,k|k-1} - \hat{x}_k^-\right]^T + R^w \tag{16}$$

where $R^w$ is process noise covariance, and

$$\chi_{k|k-1} = \left[ \hat{x}_k^- \quad \hat{x}_k^- + \gamma\sqrt{P_k^-} \quad \hat{x}_k^- - \gamma\sqrt{P_k^-} \right] \tag{17}$$

$$\mathrm{Y}_{k|k-1} = h[\chi_{k|k-1}] \tag{18}$$

$$\hat{y}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathrm{Y}_{i,k|k-1} \tag{19}$$

4. Measurements update equations:

$$P_{\bar{y}_k \bar{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right] \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T + R^v \tag{20}$$

where $R^v$ is process noise covariance, and

$$\chi_{k|k-1} = \left[ \hat{x}_k^- \quad \hat{x}_k^- + \gamma\sqrt{P_k^-} \quad \hat{x}_k^- - \gamma\sqrt{P_k^-} \right] \tag{21}$$

$$P_{x_k y_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\chi_{i,k|k-1} - \hat{x}_k^-\right] \left[\mathrm{Y}_{i,k|k-1} - \hat{y}_k^-\right]^T \tag{22}$$

$$\mathrm{K}_k = P_{x_k y_k} P_{\bar{y}_k \bar{y}_k}^{-1} \tag{23}$$

$$\hat{x}_k = \hat{x}_k^- + \mathrm{K}_k (y_k - \hat{y}_k^-) \tag{24}$$

$$P_k = P_k^- - \mathrm{K}_k P_{\bar{y}_k \bar{y}_k} \mathrm{K}_k^T \tag{25}$$

## IV. The Modified GAP-RBF Neural Network and the Modified UKF Learning Algorithm

The proposed modifications can be described in two sections. First, changes in the GAP-RBF algorithm and second on the UKF learning method.

### A. *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 en and $\varepsilon_n$ . 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 exciting (PE) of the input data. If the input data have enough degree of PE, smooth and accurate output can be obtained with suitable adjusting of the threshold values, but if the inputs do not be PE, which may happen for instance in the case of closed-loop identification, then the threshold values can not help and hence modification on adding and pruning rules will be necessary.

In on-line identification, it should be better to change the adding strategy in such a way that, the rate of adding neurons increased in the beginning of the identification process, and as the identification process continues on, the rate of adding neurons decreased. In the original algorithm [5], and its applications [1], [5] and [8], neurons are added hardly in the start of the modeling process. This causes large errors in the start of the algorithm and hence the learning EKF or UKF techniques can not estimate the relevant parameters. As the rate of neuron creation can be controlled with $\varepsilon_n$ , an exponential function is proposed to be used in order to increase $\varepsilon_n$ as follows:

$$\varepsilon_n = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min})(1 - e^{-n/\tau}) \tag{26}$$

where $\tau$ is the parameter that can be used to control the rate of increasing $\varepsilon_n$ .

Another problem is due to oscillation in the number of created neurons that can cause big errors in the identification results. This oscillation will be occurred in on-line identification, when the numbers of neurons that are created are small and the inputs data have small degree of PE, too. The effects of mentioned problem can be reduced with changing the pruning criteria by adding a new pruning factor ( $\beta$ ) as follows:

$$\left|(1.8\sigma_{nr})^{l}\alpha_{nr}/S(X)\right| < \beta e_{\min} \tag{27}$$

where $0 < \beta \le 1$.

### B. The Modified UKF Algorithm

In on-line identification, learning algorithms should be fast. However, to accomplish this objective, the initial value of the covariance matrix can not be set at large value because this option will produce large error and hence the neural network output will oscillate and consequently can not be converged when other samples are received. Besides small initial value of the covariance matrix has another bad effect. This slows down the identification process, which is not acceptable in the on-line applications. In order to be able to track any dynamic changes efficiently, learning algorithms should be fast enough to be converged in a few number of iterations. A factor, called as $\eta$, is proposed to be used in the UKF algorithm which acts as the forgetting factor concept in the usual recursive least squares (RLS) algorithm.

When the GAP-RBF toggles to adjust the parameters, $\eta$ is reset to the initial value, and then it changes exponentially in such a way that the final value is reached in a few number of iterations. As a consequence, the modified equation is as follows:

$$P_k = (P_k^- - \mathrm{K}_k P_{\bar{y}_k\bar{y}_k}\mathrm{K}_k^T)/\eta_k \tag{28}$$

where

$$\eta_k = \eta_{k-1} + (1-\eta_{k-1})(1-e^{-t/\delta}),\ 0 < \eta_k \le 1 \tag{29}$$

where t is the time duration that is spent during which the GAP-RBF toggles to the UKF learning algorithm. t is then reset to zero when the GAP-RBF algorithm is exited from the UKF. $\delta$ is the parameter that controls the rate of changing in $\eta$.

## V. Simulation Results

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

### A. Nonisothermal Continues Stirred Tank Reactor (nonisothermal CSTR)

The process is a nonisothermal CSTR with an irreversible reaction (A $\rightarrow$ B) which consists of two nonlinear ordinary differential equations, given by [9]:

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{Af} - C_A) - k_0 C_A \exp(-\frac{E}{RT})\phi_c(t), \tag{30}$$

$$\frac{dT}{dt} = \frac{q}{V}(T_f - T) + \frac{(-\Delta H)k_0 C_A}{\rho C_p}\exp(-\frac{E}{RT})\phi_c(t) + \frac{\rho_c C_{pc}}{\rho C_p V} q_c\left[1-\exp\left(-\frac{hA}{q_c\rho C_{pc}}\phi_h(t)\right)\right]\times(T_{cf} - T) \tag{31}$$

where

| | |
|---|---|
| $\phi_h(t)$ | fouling coefficient |
| $\phi_c(t)$ | 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.

This CSTR has a time-varying characteristic given by: $h_d = \phi_h(t)\times h = (1-\alpha_h t)h_A$ where $\alpha_h = 0.01$ is fouling constant.

The free parameters have been fixed in all the experiments as follows in order to be able to compare and analyze the experiments results at the same conditions: $\varepsilon_{\max}$ = 0.1, $\varepsilon_{\min}$ = 0.01, $\kappa$ = 0.85, $\gamma$ = 0.995 and $e_{min}$=0.001.

In order to make the picture of the performed simulation experiments more real, two distinct white noises with variance of 0.01 have been added to the original dynamics in (30) and (31) to simulate as the process noises. Since the effluent concentration (CA) has been assumed as the process output, a white noise with variance of 0.01 has also been added to it as the measurement noise.

The identification experiments have been performed using a random excitation signal with mean of 0.5 and variance of 1 superimposed on the multi-levels coolant flowrate, as the manipulated variable, in order to drive the CSTR process at different operating conditions starting with the initial nominal condition specified by Table I. As shown in Fig.1, the process is excited for a sequence of +10%, -20%, +5% and finally another +5% from the initial nominal condition.

**Table I. Nominal CSTR Operating Condition**

| Nominal CSTR Operating Condition | |
|---|---|
| $q = 100 l \min^{-1}$ | $E/R = 9.95\times10^{3} K$ |
| $C_{Af} = 1 moll l^{-1}$ | $-\Delta H = 2\times10^{5} cal.mol^{-1}$ |
| $T_f = 350K$ | $\rho, \rho_c = 1000 gl^{-1}$ |
| $T_{cf} = 350K$ | $C_p, C_{pc} = 1 ca\lg^{-1} K^{-1}$ |
| $V = 1001$ | $q_c = 103.41 l \min^{-1}$ |
| $h_A = 7\times10^{5} cal \min^{-1} K^{-1}$ | $T = 440.2K$ |
| $k_0 = 7.2\times10^{10} \min^{-1}$ | $C_A = 8.36\times10^{-2} mol.l^{-1}$ |

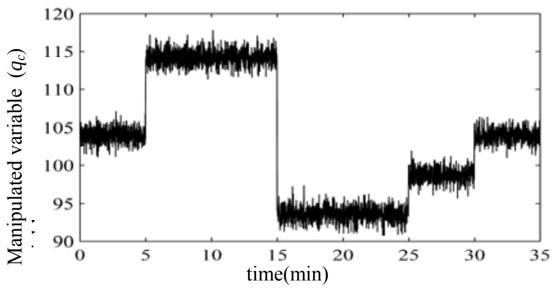


**Fig.1. Noisy input sequence.**

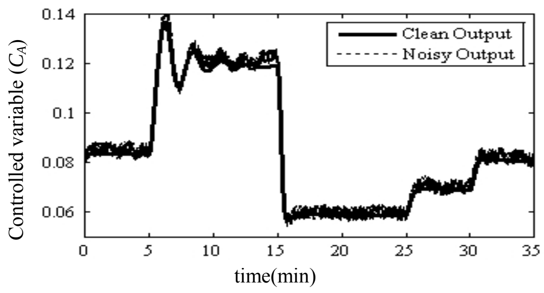
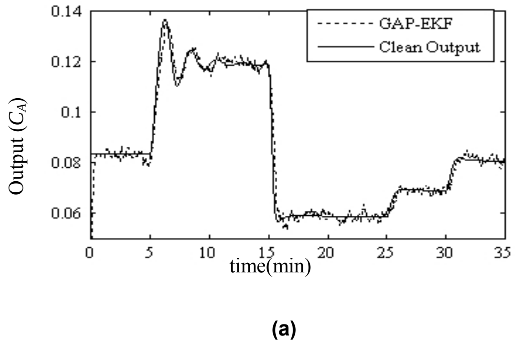


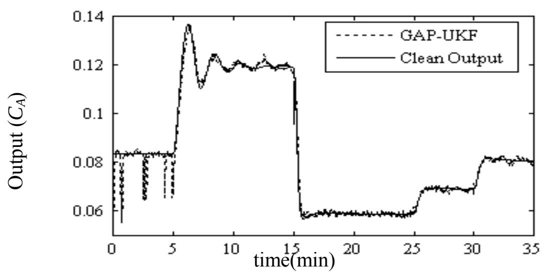
**Fig. 2. Real (noisy) and clean CSTR output**

The obtained process output response has been illustrated in Fig.2 together with a non-real noise-free process output response for the demonstration purposes.

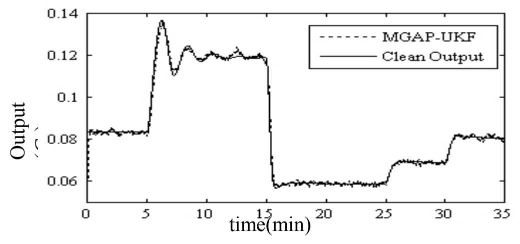
The complete identification results for different performed experiments with GAP-RBF and MGAP-RBF algorithms have been shown in Fig. 3. In Fig. 3 (a-c) vertical axis is output concentration ($C_A$) and in Fig. 3 (d-f) the vertical axis is the number of created neurons.



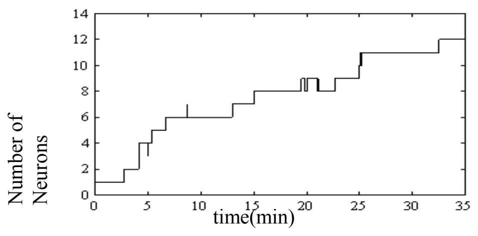
**(a)**



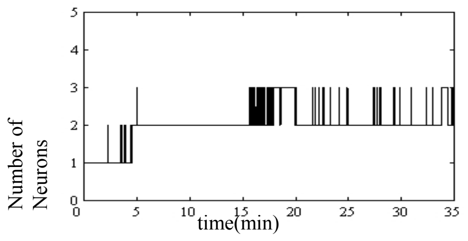
**(b)**



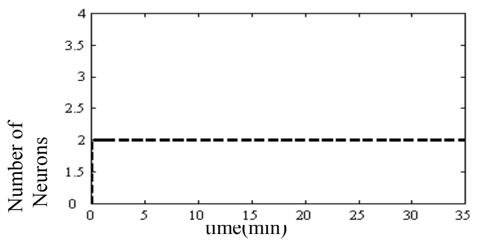
**(c)**



**(d)**



**(e)**



**(f)**

**Fig. 3. Approximated network output and neuron updating progress. (a)-(c) GAP-EKF, GAP-UKF, MGAP-UKF output networks respectively. (d)-(f) neuron updating progress of mentioned networks.**

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.

**Table II. Performance Comparison of Different Neural Networks**

| Type of NN | ISE |
|---|---|
| GAP-EKF | 0.2057 |
| GAP-UKF | 0.1646 |
| MGAP-UKF | 0.0276 |

## VI. CONCLUSIONS

Two different types of adaptive neural networks, i.e. GAP-RBF with EKF and UKF learning algorithms and MGAP-RBF have been utilized for on-line identification of non-linear systems. In this paper, this original network structure has been modified in order to enable their applications for on-line identification purposes. The basic contributions can be summarized as follows:

- Modification in neuron creation criterion.
- Modification in neuron pruning criterion.

Furthermore, the learning mechanisms MGAP-RBF have been enriched with the UKF algorithm. The proposed learning algorithm has also been included with a forgetting factor strategy in order to increase it's alertness to track any dynamic changes efficiently. The proposed on-line identification techniques have been implemented on non-linear CSTR and Mackey-Glass time-series. The analysis of the resulting outcomes indicates the following observations:

- 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.
- The GAP-RBF-based identification algorithms have a lower degree of computational complexity 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.
- 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. The UKF learning algorithm performs better in the face of contaminating noise. However, its computational complexity is higher than EKF.

As a result, 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 simplicity of the network-model structure.

## REFERENCES

[1] Huang G.B., Saratchandran P., and Sundararajan N., “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation,” IEEE Trans. Neural Networks, vol. 16, No. 1, January 2005.

[2] Julier S. J. and Uhlmann J. K., “A New Extension of the Kalman Filter to Nonlinear Systems,” in Proc. of AeroSense: The 11th Int. Symp. A.D.S.S.C., 1997.

[3] Nishida K., Yamauchi K., Omori T., “An On-line Learning Algorithm with Dimension Selection using Minimal Hyper Basis Function Networks”, SICE Annual Conference in Sapporo, August 4-6, 2004.

[4] Platt J., “A resource-allocating network for function interpolation,” Neural Computat., vol. 3, pp. 213–225, 1991.

[5] Huang G.-B., Saratchandran P., and Sundararajan N., “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, December 2004.

[6] Wan E. A. and van der Merwe R., “The Unscented Kalman Filter for Nonlinear Estimation,” in Proc. of IEEE Symposium 2000 (AS-SPCC), Lake Louise, Alberta, Canada, Oct. 2000.

[7] Wan E., van der Merwe R., and Nelson A. T., “Dual Estimation and the Unscented Transformation,” in Neural Information Processing Systems MIT Press 12, pp. 666–672, 2000.

[8] Wang Y., Huang G.-B., Saratchandran P., and Sundararajan N., “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 31 - August 4, 2005.

[9] Nikravesh M., “Dynamic neural network control”, Ph.D. Dissertation, University of South Carolina, Columbia, SC, 1994.