# Adaptive Predictive Controllers Using a Growing and Pruning RBF Neural Network

***Jafari, Mohammad Reza; Salahshoor, Karim****+

*Department of Automation and Instrumentation, Petroleum University of Technology, Tehran, I.R. IRAN*

**ABSTRACT:** *An adaptive version of growing and pruning RBF neural network has been used to predict the system output and implement Linear Model-Based Predictive Controller (LMPC) and Non-linear Model-based Predictive Controller (NMPC) strategies. A radial-basis neural network with growing and pruning capabilities is introduced to carry out on-line model identification. An Unscented Kalman Filter (UKF) algorithm with an exponential time-varying forgetting factor has been presented to enable the neural network model to track any time-varying process dynamic changes. An adaptive NMPC has been designed based on the sequential quadratic programming technique. The paper makes use of a dynamic linearization approach to extract a linear model at each sampling time instant so as to develop an adaptive LMPC. The servo and regulating performances of the proposed adaptive control schemes have been illustrated on a non-linear Continuous Stirred Tank Reactor (CSTR) as a benchmark problem. The simulation results demonstrate the capability of the proposed identification strategy to effectively identify compact, accurate and transparent model for the CSTR process. It is shown that the proposed adaptive NMPC controller presents better improvement with faster response time for both servo and regulatory control objectives in comparison with the proposed adaptive LMPC, an adaptive generalized predictive controller based on Recursive Least Squares (RLS) algorithm and well-tuned PID controllers.*

**KEY WORDS:** *Neural networks, On-line identification, Adaptive control, Model-based predictive control, CSTR.*

## INTRODUCTION

Model Predictive Control (MPC) approaches have been recognized as the accepted standard to cope with some of the difficult control problems in process industry [1,2]. Their ability to handle input and output constraints, time delays, non-minimum phase behaviour and multivariable systems have made them very attractive to the industrial users. The core of all MPC algorithms is the moving horizon strategy. An identified process model is used to predict the future response and then, the control action is optimally determined so as to obtain the desired performance over a finite time horizon. Thus, the choice of process model representation is a crucial and important issue in MPC. Most of the predictive controllers still use an explicit linear model of the process to be controlled. In practice, however, most of the industrial processes posses severe non-linear dynamics. This makes the linear

* *To whom correspondence should be addressed.*
+ *E-mail: salahshoor@put.ac.ir*
*1021-9986/11/2/125 14/$/3.40*

controller to be less effective or even detrimental when the process operates over a wide range of operating conditions leading to time-varying model structures and parameters in an unknown manner. Adaptive control strategy is an interesting idea to cope with such difficult model uncertainties by providing the ability to track variations in process dynamics. Most adaptive control techniques, however, are based upon a fixed linear process model structure whose parameters are only free to be tuned to any possible process dynamic changes. This approach may have limited success because any structural dynamic changes should be adapted via the model parameters tuning.

Neural Networks (NN) have shown to have good approximation capability for modelling non-linear systems. A large number of predictive control schemes have been developed based on multi-layer neural network models since 1990. Classical neural networks have been used for the identification tasks [3-5], often as a part of adaptive predictive control schemes [6, 7]. However, there is no general procedure to choose the required number of layers and neurons to achieve an accurate approximation in a given control problem. The usual practice is to use enough neurons to capture the complexity of the underlying process dynamic without having the NN overfit the training data. For nonlinear black box identification, however, there is no guarantee that the fixed number of assumed neurons can cover the process operating range. Radial Basis Function (RBF) neural networks have been popularly used in many control applications in recent years. This is due to their ability to approximate complex non-linear mappings directly from the input-output data with a simple topological dynamic structure. Combining this network with self-generating network algorithms offers an attractive approach to make efficient adaptive neural network which can adjust its dynamic structure complexity to varying non-linear process dynamic without requiring a prior knowledge.

Several self-generating network algorithms, such as Resource Allocation Network (RAN) and minimum RAN have been proposed in the literature for training RBF neural networks [8,9]. In recent years, difficult methods [10-12] have been proposed in which sequential learning algorithms are used. *Huang et. al.* [13] proposed a simple sequential learning algorithm with network Growing and Pruning (GAP) capabilities based on the relationship between the significance of a neuron and the required model accuracy for RBF networks, referred to as GAP-RBF.

The original GAP-RBF algorithm has been modified in this paper to enhance its performance for on-line identification of non-linear systems. The new modified GAP-RBF (MGAP-RBF) neural network is used as a generic model for on-line identification of non-linear systems. The Unscented Kalman Filter (UKF) estimation algorithm has been introduced as a new learning algorithm to recursively updates the free parameters of the MGAP-RBF neural network. An exponential forgetting factor scheme has been included in the UKF algorithm to enable its tracking feature against any possible time-varying system dynamic change.

This paper proposes two indirect adaptive predictive controllers based on the MGAP-RBF neural network. An adaptive Non-linear MPC (NMPC) has been developed without restricting the process model to linear dynamics. The second proposed predictive controller is based on the popular Generalized Predictive Control (GPC) strategy. Finally, the performances of the proposed adaptive predictive controllers are illustrated on a simulated non-linear Piovoso CSTR.

The paper is organized as follows. First, the on-line non-linear system identification using the MGAP-RBF neural network is presented. Second, the proposed NMPC and GPC controllers are developed by employing the identified MGAP-RBF model. The remainder of the paper is devoted to demonstrating the servo and regulatory performances of the proposed adaptive predictive controllers on CSTR benchmark simulation problem.

## *Dynamic Non-linear System Identification*

### *GAP-RBF Algorithm*

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

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

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

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

where $\mu_k$ and $\sigma_k$ refer to the centre and width of the kth hidden neuron respectively, and $\|.\|$ indicates the Euclidean norm.

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 samples 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 [9] in which all the neurons will be checked for adding, pruning and adjusting purposes. This results in a reduction in the overall computations and thereby increasing the learning speed. The significance of the kth hidden neuron is defined as [13]:

$$E_{sig}(k) = \left|\frac{(1.8\sigma_k)^l \alpha_k}{S(X)}\right| \quad (3)$$

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.

For both growing and pruning, it is shown that [13] one needs to check only the nearest neuron based on the following Euclidean distance to the current input data $x_n$ for its significance:

$$\|x_n - \mu_{nr}\| = \min_k(\|x_n - \mu_k\|), k = 1,...,K \quad (4)$$

where $\mu_{nr}$ is the centre 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:

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

where $x_n$ is the latest input received, $\mu_{nr}$ is the centre of the hidden neuron nearest (in the Euclidean distance ) to $x_n$. $e_{min}$ is the desired approximation accuracy and $\varepsilon_n$ is a threshold to be selected appropriately. If the growing criteria (5) 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:

$$\begin{cases} \alpha_{K+1} = e_n \\ \mu_{K+1} = x_n \\ \sigma_{K+1} = \kappa\|x_n - \mu_{nr}\| \end{cases} \quad (6)$$

where $e_n = y_n - f(x_n)$.

In this case, all the other present neurons (k =1,…, 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 (5) 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 algorithm. Then, the significance of the nearest (i.e., most recently adjusted) neuron is checked via the following pruning criterion:

$$E_{sig}(nr) = \left|\frac{(1.8\sigma_{nr})^l \alpha_{nr}}{S(X)}\right| < e_{min} \quad (7)$$

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 [13] 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:

**Compute** the overall network output:

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

where K is the number of hidden neurons.

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

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

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

**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:

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

Else

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

**Check** the pruning criterion for the nearest (*nr*th) hidden neuron:

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

Endif

Endif

## MGAP-RBF ALGORITHM

The original GAP-RBF algorithm has been modified as follows to enhance its capabilities for on-line system identification applications:

- Enhancing the smooth creation of the neurons.
- Enhancing the pruning criterion to prevent probable oscillation in the number of created neurons.
- Utilization of the UKF estimation algorithm to adjust free network parameters.
- Utilization of a time-varying forgetting factor scheme to maintain a desired parameter tracking capability.

The proposed modifications can be described in the following two sections:

### *a) The Modified Growing and Pruning Criteria*

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 $\varepsilon_n$. 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 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 [13] and its application reported in [10-14] 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 learning algorithm 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 $\varepsilon_n$ (Eq. (9)), the following exponential time-varying pattern is proposed to make a gradual evolution of $\varepsilon_n$ from an initial higher sensitivity $\varepsilon_{min}$ to a final lower sensitivity $\varepsilon_{max}$:

$$\varepsilon_n = \varepsilon_{min} + (\varepsilon_{max} - \varepsilon_{min})(1 - e^{-n/\tau}) \quad (11)$$

where $\tau$ is the time-constant parameter that can be used to control the time rate evolution of $\varepsilon_n$ .

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 number 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. (7)) as follows:

$$\left|(1.8\sigma_{nr})^{1}\alpha_{nr}/S(X)\right| < \beta e_{min} \tag{12}$$

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

***b) The UKF Learning Algorithm***

The original GAP-RBF algorithm uses the Extended Kalman Filter (EKF) as its parameters adjusting algorithm. In practice, however, the use of the EKF has two well-known drawbacks:

- Linearization can produce highly unstable filters if the assumptions of local linearity are violated.
- The derivation of the Jacobian matrices is nontrivial in most applications and often lead to significant implementation difficulties.

To address these limitations, *Julier & Uhlmann* [11,15] developed the UKF algorithm.

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

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ y_k &= h(x_k) + v_k \end{aligned} \tag{13}$$

Where $x_k$ represents the hidden states, $u_k$ is the vector of known exogenous inputs, and $y_k$ represents the vector of noisy measured outputs. The random variables $w_k$ and $v_k$ represent process and measurement noises, respectively.

Instead of linearizing these non-linear model equations using Jacobian matrices in the EKF, the UKF uses a "deterministic sampling" approach to calculate the mean and covariance estimate of Gaussian random state variables ($x_k$) with a minimal set of 2L+1 sample points (L is the state dimension), called as sigma points. The results are accurate to the third-order (Taylor series expansion) for Gaussian inputs for all non-linearities.

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. Initialize with some initial guesses for the state estimate ($x_0$) and the error covarince matrix ($P_0$), defined as:

$$\begin{aligned} \hat{x}_0 &= E[x_0] \\ P_0 &= E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \end{aligned} \tag{14}$$

For $k \in \{1,...,\infty\}$ , (E[.] denotes the expected value).

2. Calculate the sigma points:

$$\chi_{k-1} = \left[\hat{x}_{k-1} \; \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} \; \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}}\right] \tag{15}$$

where $\lambda = \alpha^2(L+k) - L$ and $\gamma = \sqrt{L+\lambda}$ are scaling parameters.

3. Time update equations:

$$\chi^*_{k|k-1} = f\left[\chi_{k-1}, u_{k-1}\right] \tag{16}$$

$$\hat{x}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \chi^*_{i,k|k-1} \tag{17}$$

$$P^-_k = \sum_{i=0}^{2L} W_i^{(c)} \left[\chi^*_{i,k|k-1} - \hat{x}^-_k\right]\left[\chi^*_{i,k|k-1} - \hat{x}^-_k\right]^T + R^w \tag{18}$$

where $\left\{W_i^{(m)}\right\}$ and $W_i^{(c)}$ are sets and scalar weights, $R^w$ is process noise covariance, and

$$\chi_{k|k-1} = \left[\hat{x}^-_k \; \hat{x}^-_k + \gamma\sqrt{P^-_k} \; \hat{x}^-_k - \gamma\sqrt{P^-_k}\right] \tag{19}$$

$$Y_{k|k-1} = h[\chi_{k|k-1}] \tag{20}$$

$$\hat{y}^-_k = \sum_{i=0}^{2L} W_i^{(m)} Y_{i,k|k-1} \tag{21}$$

4. Measurements update equations:

$$P_{\bar{y}_k\bar{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[Y_{i,k|k-1} - \hat{y}^-_k\right]\left[Y_{i,k|k-1} - \hat{y}^-_k\right]^T + R^v \tag{22}$$

$$P_{x_k y_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[\chi_{i,k|k-1} - \hat{x}^-_k\right]\left[Y_{i,k|k-1} - \hat{y}^-_k\right]^T \tag{23}$$

$$K_k = P_{x_k y_k} P^{-1}_{\bar{y}_k\bar{y}_k} \tag{24}$$

$$\hat{x}_k = \hat{x}^-_k + K_k(y_k - \hat{y}^-_k) \tag{25}$$

$$P_k = P^-_k - K_k P_{\bar{y}_k\bar{y}_k} K_k^T \tag{26}$$

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. (27)) as follows:

$$P_k = (P_k^- - K_k P_{\bar{y}_k \bar{y}_k} K_k^T)/\eta_k \quad (27)$$

Where $\eta_k$ behaves as the forgetting factor concept in the usual recursive least squares (RLS) algorithm which undergoes the following time-varying evolution:

$$\eta_k = \eta_{k-1} + (1-\eta_{k-1})(1-e^{-t/\delta}) \ , \ 0 < \eta_k \le 1 \quad (28)$$

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 fixed structure. Thus, $t$ is reset to zero when any network structural change occurs, 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, $\eta_k$ start with a lower initial value to accelerate the parameter estimation. Then, its value is changed exponentially with a desired time-constant ($\delta$) to a higher final value to assure the estimator convergence property.

## ADAPTIVE NEURAL-BASED MODEL PREDICTIVE CONTROLLERS

In this section, two adaptive versions of neural-based MPC controllers are proposed. Both controllers utilize the MGAP-RBF neural network as the generic non-linear model of the process to be controlled.

### *a) System Dynamic Model*

The MPC methodology requires a suitable dynamic model capturing all the salient features of the system to be controlled in order to predict its response with reasonable accuracy over a finite time horizon.

One standard model structure that has been used for non-linear identification is the following general Non-linear Auto-Regressive with eXogenous (NARX) input equation, recommended by *Narendra et al.* [16]:

$$y(t) = f[y(t-1), y(t-2), ..., y(t-n), u(t), ..., u(t-m)] \quad (29)$$

where y(t) and u(t) are, respectively, the system output and input; n and m are the corresponding time lags of $\{y(t)\}$ and $\{u(t)\}$; and f(.) is an unknown non-linear function to be identified. The MGAP-RBF neural network is used in this paper to approximate the non-linear function f(.) at each sampling time, employing $X_n = [y(t-1), y(t-2), ..., y(t-n), u(t), ..., u(t-m)]$ as the network input vector. The network parameter learning is implemented in the on-line identification phase with the proposed UKF estimation algorithm.

### *b) Adaptive Neural-Based NMPC Controller*

The proposed NMPC controller utilizes the identified MGAP-RBF neural network to predict the system output over the finite prediction horizon. The output prediction is then used by a numerical optimization program to determine the desired control sequence at every sampling instant that minimizes the following general cost function to take the controlled system to a desired operating point $(u_s, x_s, y_s)$ specified by steady-state economic objectives:

$$J = \sum_{j=1}^{N} \|y(t+j) - y_s\|_R^q + \sum_{j=1}^{M-1} \|\Delta u(t+j)\|_P^q + \sum_{j=1}^{M-1} \|u(t+j) - u_s\|_Q^q \quad (30)$$

where P,Q and R are weighting matrices; N and M define prediction and control horizon, respectively.

The minimization is subject to the follofwing operational constraints:

$$\underline{y} \le y(t+j) \le \bar{y} \quad , \quad \forall j = 1, N \quad (31)$$

$$\underline{u} \le u(t+j) \le \bar{u} \quad , \quad \forall j = 1, M-1$$

$$\Delta\underline{u} \le \Delta u(t+j) \le \Delta\bar{u} \quad , \quad \forall j = 1, M-1$$

This general optimal control formulation problem can be written as the following non-linear programming

problem for the proposed NMPC controller with $w^T = [u^T x^T y^T]$ as the design parameter vector:

$$\min_{w} J(w) \qquad (32)$$

subject to: $c(w)=0, \quad g(w) \leq 0$

where the equality constraint vector **c** corresponds to the model constraints $f(\mathbf{x},\mathbf{u}) = 0$, $\mathbf{y} - h(\mathbf{x}) = 0$ and has $n_c$ components, while the inequality canstraint vector **g** relates to Eq. (31) and has $n_g$ components.

This finite horizon constrained optimization problem is solved using Sequential Quadratic Programming (SQP) method [17,18]. This method allows to mimic Newton's method for constrained optimization just as is done for unconstrained optimization. At each iteration, an approxiamtion is made of the Hessian of the Lagrangian function $L(w,\lambda_1,\lambda_2) = J(w) + \lambda_1^T c(w) + \lambda_2^T g(w)$ ($\lambda_1 \in \Re^{n_c}$ and $\lambda_2 \in \Re^{n_g}$ are the Lagrange multipliers ) to generate a Quadratic Programming (QP) subproblem as follows:

$$\min_{d} \nabla J(w_k)^T d + \frac{1}{2} d^T B_k d \quad , \quad (d \in \Re^n) \qquad (33)$$

Subject to the linearized constraints:

$$c(w_k) + \nabla c(w_k)^T d = 0 \qquad (34)$$

$$h(w_k) + \nabla h(w_k)^T d \leq 0 \qquad (35)$$

where $B_k = \nabla^2_{ww} L(w,\lambda_1,\lambda_2)$ is a positive definite approximation of the Hessian matrix of the Lagrangian function. This QP subproblem is then solved at each iteration by the projection method used in the MATLAB optimization toolbax to obtain a new search direction vector as $d_k^T = [d_u^T d_x^T d_y^T]$.

The vector is used in $w_{k+1} = w_k + \alpha_k d_k$ to converge to the original optimization problem solution.

***c) Adaptive Neural-Based GPC Controller***

Real-time dynamic optimization using a non-linear model may give rise to computational difficulties. Therefore, an alternative predictive control approach is presented in this section which reduces the computational burden.

The proposed LMPC controller is based on the well-known GPC strategy [19] which has received a lot of attenion from both industry and academia. A MGAP- RBF neural network is identified on-line to incorporate any system nonlinearity to considerable accuracy from empirical input-output system data. The identified model is then linearized around the actual system operating point at each sampling time instant. The resulting time-variant linear model is utilized to design the GPC controller. Fig. 1 shows the basic concept of predictive learning control strategy based on dynamic linearization of the MGAP-RBF neural network.

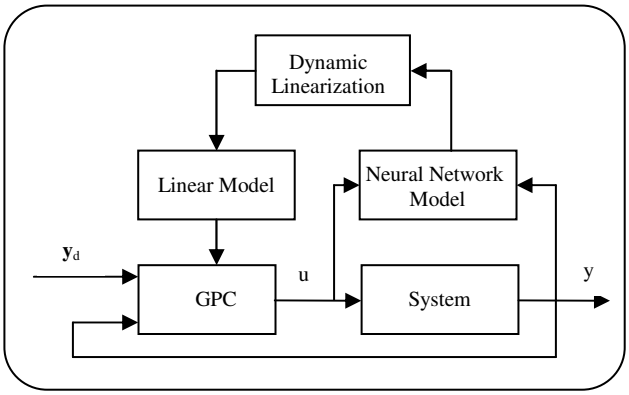


***Fig. 1: Predictive control scheme based on non-linear models.***

*Dynamic Linearization*

Base on the updated MGAP-RBF neural network, linear models can be extracted at each sampling time instant. In contrast with classic linearization which is only performed in equilibria along the static input-output mapping, the system in linearized at each time instant regardless of whether the system is in a steady or in a transient state.

Hence, actual information about the system dynamics is available in all states of operations. As a result, the objective is to calculate the time-variant parameters $a_i(t)$ and $b_i(t)$ in the following general ARX model:

$$G(z,t) = \frac{b_1 z^{-1} + b_2 z^{-2} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n}} = \frac{B(z^{-1})}{A(z^{-1})} \qquad (36)$$

for arbitrary system states.

The free model parameters can be obtained using a first-order Taylor series approximation of the non-linear MGAP-RBF model:

$$a_k(t) = -\frac{\partial y(t)}{\partial y(t-i)} \Big|_{\chi=\chi(t)} \qquad (37)$$

$$b_k(t) = -\frac{\partial y(t)}{\partial u(t-i)} \Big|_{\chi=\chi(t)} \qquad (38)$$

Knowing that the detailed MGAP-RBF network structure and parameters of the mapping $y = f(x_n)$ are available at each sampling time instant via the identification procedure. This allowes the linear model parameters $a_i(t)$ and $b_i(t)$ to be easily calculated using the chain rule of the gradient algorithm.

*GPC Control Problem Formulation*

GPC control strategy exploits a particular kind of linear system model called as Controlled Auto-Regressive Integrated Movin Average (CARIMA) model given by:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + C(q^{-1})\frac{e(t)}{\Delta} \quad (39)$$

where $A(q^{-1})$, $B(q^{-1})$ and $C(q^{-1})$ are polynomials in the backward shift operator $q^{-1}$ and $\Delta=1-q^{-1}$ represents the differencing operator. e(t) is an uncorrelated random sequence, u(t) and y(t) denote the input and output, respectively. For simplicity, the $C(q^{-1})$ polynomial is chosen to be 1. It is noted that if $C(q^{-1})$ can be truncated, it can be absorbed into $A(q^{-1})$ and $B(q^{-1})$ polynomials.

Now, a prediction of the system output, given measured output up to time t and control input u(t+i) for $i \le -1$, is:

$$\hat{y}(t+j\,|\,t) = G_j(q^{-1})\Delta u(t+j-d-1) + F_j(q^{-1})y(t) \quad (40)$$

where j denotes the number of future time steps being predicted, $G_j(q^{-1}) = E_j(q^{-1})B(q^{-1})$, and $E_j(q^{-1})$ results from a recursive solution of the Diophantine relation:

$$1 = E_j(z^{-1})\tilde{A}(z^{-1}) + z^{-j}F_j(z^{-1})$$

Hence, $E_j$ and $F_j$ are polynomials uniquely difined, given $A(q^{-1})$ and the integer j.

The GPC control law can be derived by minimizing the following cost function:

$$J(N_1, N_2, N_u) = \sum_{j=N_1}^{N_2} \delta(j)\left[\hat{y}(t+j\,|\,t) - w(t+j)\right]^2 + \sum_{j=1}^{N_u} \lambda(j)\left[\Delta u(t+j-1)\right]^2 \quad (41)$$

where $N_1$ and $N_2$ are the minimum and maximum costing horizons, $N_u$ is the control horizon, $\delta(j)$ and $\lambda(j)$ are weighting sequences and w(t+j) is the future reference trajectory.

Minimizing Eq. (41) yields the following incremental control vector:

$$\Delta U = (G^T G + \lambda I)^{-1} G^T (w - f) \quad (42)$$

where:

$$w = [w(t+1)w(t+2)...w(t+N_2)] \quad (43)$$

$$f = [f(t+1)f(t+2)...f(t+N_2)] \quad (44)$$

$$G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots\vdots & g_0 \\ g_{N_2-1} & g_{N_2-2} & \cdots & g_{N_2-N_u} \end{bmatrix} \quad (45)$$

with f(t+j) being those components of $\hat{y}(t+j\,|\,t)$ which are known at time t as the system free response. $g_i$ are elements of the polynomial $G_i(q^{-1})$ itself obtained from the recursive Diophantine relation. Thus, at each sampling interval the parameters of the CARIMA model are first estimated via the linearization of the identified MGAP-RBF neural network in Eqs. (37) and (38). Then, the estimated parameters are utilized in the GPC controller design phase. Finally, the first element of the calculated incremental control vector in Eq. (42) is implemented to accomplish a receding horizon optimization procedure.

## SIMULATION STUDY

This section presents a set of simulation experiments on Piovoso CSTR to demonstrate the performance of:

- The proposed on-line system identification approach to model a highly non-linear and time-varying CSTR benchmark process.
- The proposed adaptive NMPC and LMPC controllers to achieve servo and regulatory objectives.

### *A. Piovoso CSTR*

*Process Description*

Piovoso CSTR- The reactor considered in this example is a CSTR [22] in which the reactor temperature controller is cascaded to the coolant temperature loop. It is assumed that the reactor composition is not available as an on-line measurement and no observer is constructed to estimate it. An irreversible first-order reaction

proceeding in the CSTR, which has the following dimensionless mass and energy balances forming the state equations:

$$\left[\frac{dx_1}{dt} = -x_1 + D_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right)\right] \quad (46)$$

$$\frac{dx_2}{dt} = -x_2 + BD_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right) + \beta(y_c - x_2) \quad (47)$$

where $x_1$ is the composition and $x_2$ the temperature and $y_c$ the cooling jacket temperature.

As only the temperature is measurable, the output equation is given by:

$y=x_2$

The parameters used for this model are given in Table 1.

The response of the CSTR with initial condition x0 = [0.9996, 10.8] and zero input ($y_c$) is illustrated in Fig. 2.

*Identification Study*

CSTR output for zero input and initial condition $x_0$ =[0.9996,10.8] by adding noise to it, presents in Fig.12. This noisy output should be identified by MGAP-RBF-UKF. The identification results illustrated in Fig. 4 and Fig. 5. The time-history profiles of neuron updating progress are shown in Fig. 5.

In the case of non-zero input, the input sequence that presents in Fig. 6 used to excite the process.

Figs.7 and 8 illustrate the obtained on-line identification results. It is noted that the GAP-RBF approach uses the EKF estimation algorithm while the proposed MGAP-RBF approach is based on the UKF algorithm. The simulation tests were run under similar initialized conditions, given by:

$\varepsilon_{max} = 0.91$, $\varepsilon_{min} = 0.001$, $\kappa = 0.1$, $\gamma = 0.95$, $e_{min}$=0.001, $e'_{min} = 0.001$ Number of inputs = 4 (u, $y_{-1}$, $y_{-2}$, $y_{-4}$).

Network model accuracy in terms of Integral of Square Error (ISE) measure (ISE for GAP-RBF is 0.2646 while ISE for MGAP-RBF is 0.0747).

*Implementation of the Proposed Adaptive MPC Controllers*

The proposed adaptive neural-based NMPC and LMPC controllers were used to control the temperature $x_2$ of the CSTR process and the results were compared to those obtained using a well-tuned conventional PID controller. In all simulation tests, the initialized parameters of the MGAP-RBF were set to the same values mentioned in part (A.2). The neural network input vector was selected as:

$$X_n = [y(t-1), y(t-2), ..., y(t-n), u(t), ..., u(t-m)] \quad (48)$$

The settings of the NMPC and GPC controller were:

$N_1$ =1, $N_2$ =15, $N_u$=7, R=1, P=1, Q=0.01, $\delta$ =1, $\lambda$ =0.8.

***Table 1: Nominal Piovoso CSTR Operating Condition.***

| Nominal CSTR Operating Condition | |
|---|---|
| $D_a$ = 0.085 | $\gamma$ = 20.0 |
| B = 22 | $\beta$ = 3.0 |

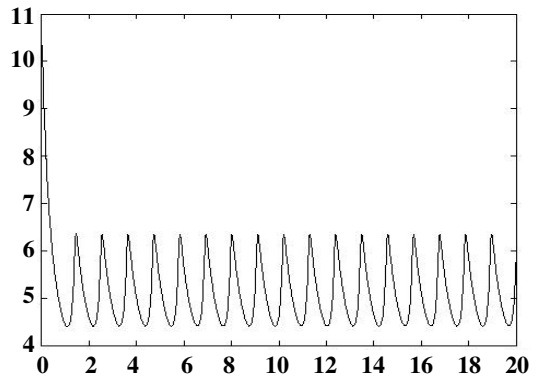


***Fig. 2: The response of the CSTR with initial condition $x_0$ = [0.9996,10.8] and zero input ($y_c$).***

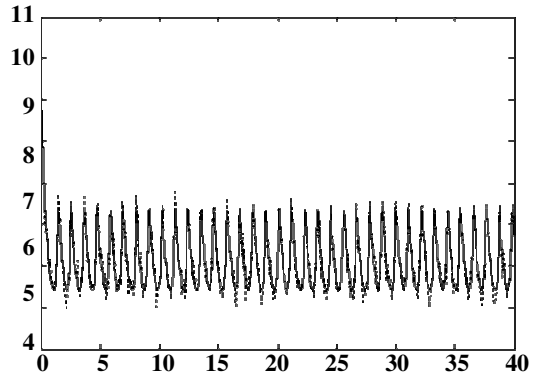


***Fig. 3: Real (dashed line) and clean (solid line) CSTR output.***

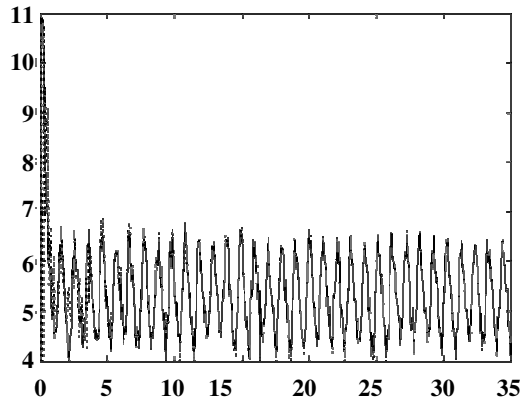


*Fig. 4: Approximated network (dashed line) and real output (solid line) for MGAP-RBF-UKF output networks.*

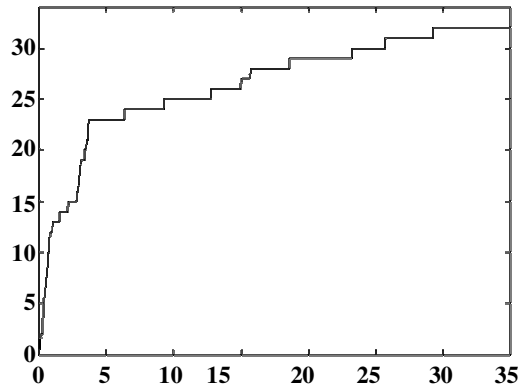


*Fig. 5: Neuron updating progress for MGAP-RBF-UKF output network.*

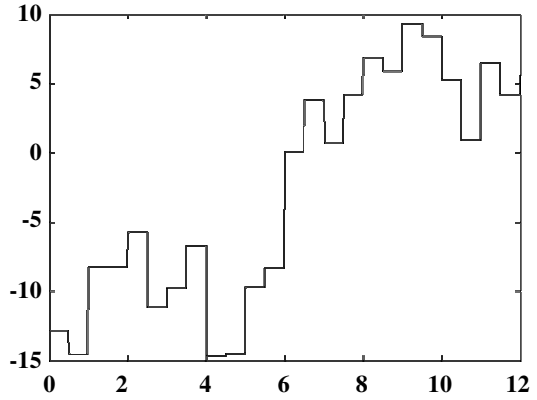


*Fig. 6: Input sequence for identification procedure.*

Moreover, the NMPC constraints were enforced as following:

$0.6\times\text{setpoint} < y < 1.5\times\text{setpoint}$ , $\Delta u < 40$ , $-15 < u < 15$

Fig. 9 illustrates the setpoint tracking performance of the NMPC, LMPC or GPC, RLS-GPC (i.e., identification by RLS method combined with the GPC control strategy) and PID controllers. The PID parameters were set to $K_c = 10$, $T_I = 0.053\ \text{min}^{-1}$, $T_D = 0.0012\ \text{min}^{-1}$ by the IMC method and then fine tuned by Simulunk Response Optimization blockset of MATLAB. The parameters of the RLS-GPC were set to: $N_1 = 1$, $N_2 = 5$, $N_u = 6$, $\delta = 1$, $\lambda = 0.68$. For illustrating the performane of controllers more precisely; the constraint of NMPC and GPC controllers are imposed to the PID controller.

The result of PID controller performance has been shown in Figs. 9, 10. Fig. 11 showes the disturbance rejection performances of NMPC, GPC, RLS-GPC and PID controllers to 2 unit step changes made in the temperature $x_2$ of the CSTR. As shown, adaptive GPC exhibits a faster response toward the steady-state setpoint.

In the mentioned method, the SQP algorithm has been implemented via the function "fmincon" in MATLAB Optimization Toolbox. It has been found that the average time needed to calculate the solution during each sampling time was 0.1 minute. Of course, more computationally efficient codes are likely to be produced for this specific real-time problem rather than the general-purpose code provided through the "fmincon" function.

Comparing the simulation results indicates the superiority of the adaptive neural-based GPC controller in this case study. These demonstrate the effectiveness of the dynamic linearization approach adopted in the proposed adaptive GPC controller.

The resulting controller scheme is much simpler than the proposed adaptive NMPC controller in terms of the required computational complexity. Nevertheless, it should be noted that the computational burden of the RLS-GPC controller is much lower than the proposed neural-based MPC methods.

Obviously, the RLS-GPC approach is simpler than the proposed adaptive GPC in both identification and control methodology. But, it was practically observed that the approach was not able to present a good performance in this case study to cope with the constraint boundries enforced on the control action by changing its controller parameters ($N_1$, $N_2$, $N_u$, $\delta$, $\lambda$).

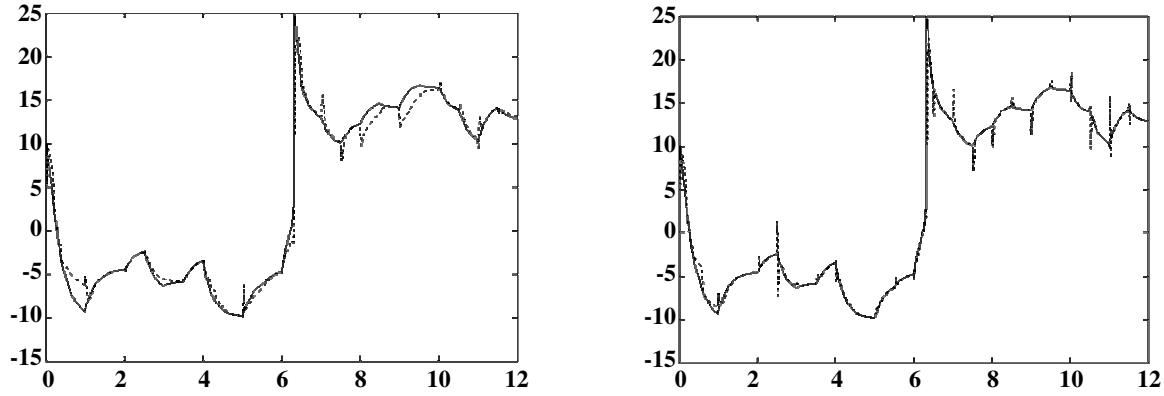

*Fig. 7: Approximated network (dashed line) and real output (solid line). (a) and (b) are GAP-RBF-EKF and MGAP-RBF-UKF output networks, respectively.*

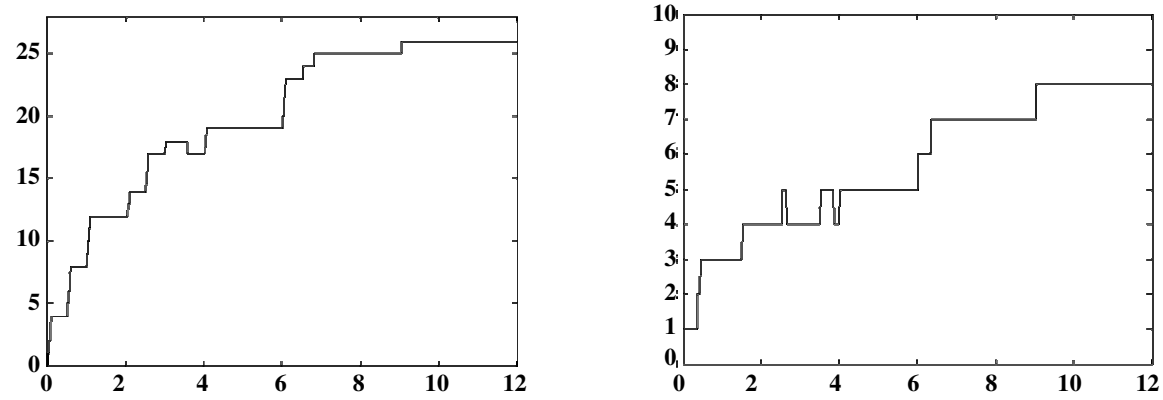

*Fig. 8: Neuron updating progress. (a) and (b) are GAP-RBF-EKF and MGAP-RBF-UKF neuron updating progress, respectively.*

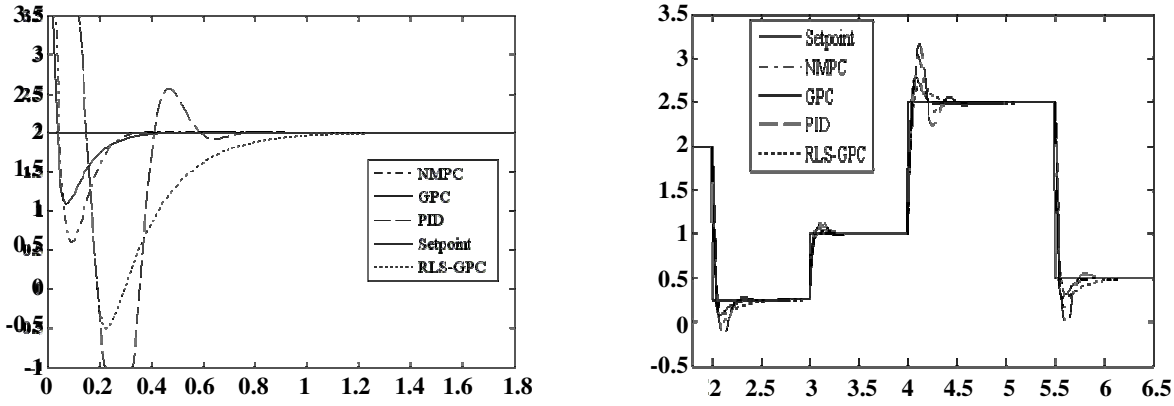


*Fig. 9: Setpoint tracking performances of neural network-based predictive controllers, RLS-GPC and PID. (a) times from 0-1.8. (b) times from 1.8-6.5.*

The PID controller is non-adaptive and hence has a very simple structure, being able to present good performance results. But, in the presence of noise and operational constraints, it can not perform well.

The performed simulation studies show that the average time needed to calculate the solution during each sampling time was 0.01 minute. This makes it as a suitable and feasible conntrol approach in most real-time process control applications.

## CONCLUSIONS

Two adaptive neural-based LMPC and NMPC controllers have been developed for non-linear systems. The developed controllers utilize the MGAP-RBF as an on-line neural-based identification strategy which incorporates the UKF learning algorithm with an exponential time-varying forgetting factor scheme to update the adaptive network structural and parametric variables. The model identification strategy needs no priori knowledge to start with and can adjust its dynamic structure complexity to varying non-linear system dynamics.

The simulation studies demonstrate the capability of the proposed identification strategy to effectively identify compact, accurate and transparent model for a non-linear CSTR benchmark problem.

It was observed that the UKF learning algorithm leads to better accuracy in comparison with the EKF algorithm due to utilization of a deterministic sampling approach to calculate mean and covariance terms. This makes the UKF algorithm to perform better in the face of process and measurement contaminating noises.

The developed adaptive neural-based MPC controllers differ from the conventional neural-based controllers because the identified network structure is very simple, having limited adaptive neurons in the hidden layer. This simplicity makes the controller design and implementation much easier than the classical approaches based on the fixed network structures including less significant neurons. The adaptive neural-based NMPC controller was introduced as a non-linear predictive control approach which utilizes the MGAP-RBF neural network as an explicit non-linear model of the controlled process. This led to a non-linear constrained optimization problem. The SQP method, which mimics Newton's approach, was used to generate the control solution. To reduce the computational burden, an alternative adaptive

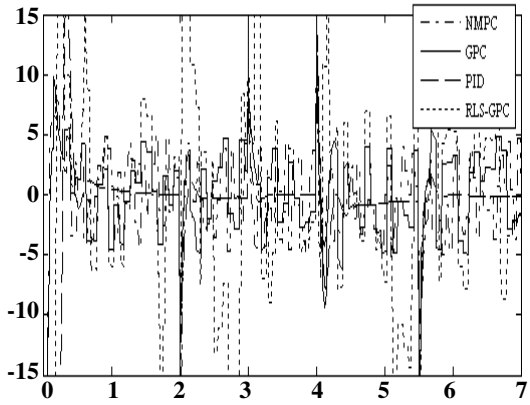


***Fig. 10: Control action u imposed by controller to coolant valve with the same constraints on PID.***

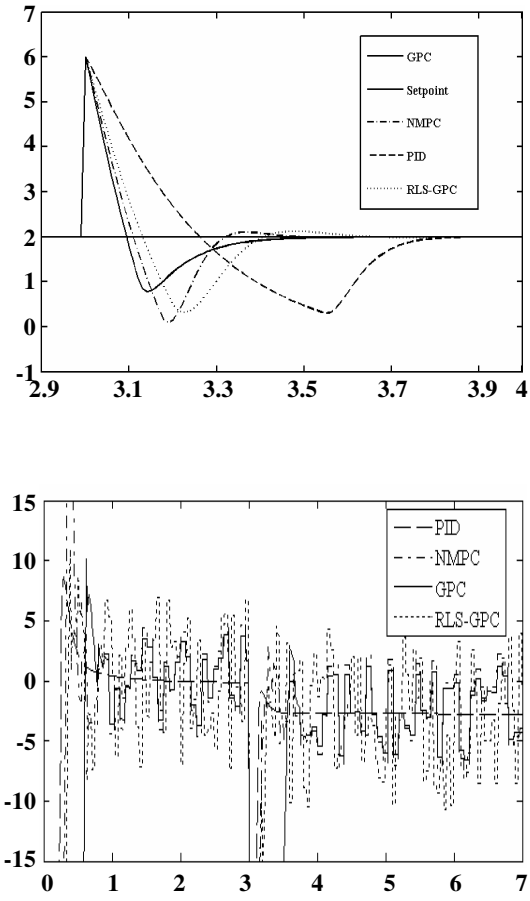


***Fig. 11: (a) Disturbance rejection performance for 2 unit change in the temperature $x_2$ of the CSTR. (b) Control action u imposed by controller to coolant valve.***

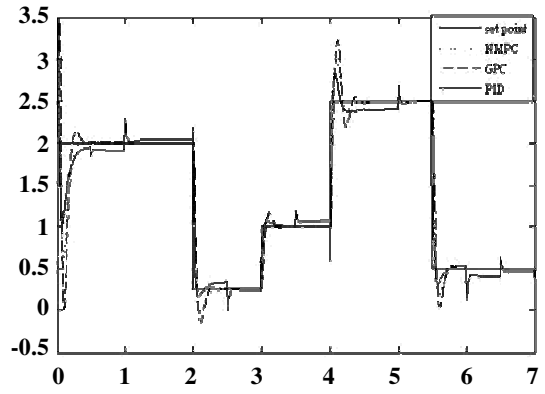


*Fig. 12: Neural network-based predictive controllers setpoint tracking performances with imposing the same noise on PID.*

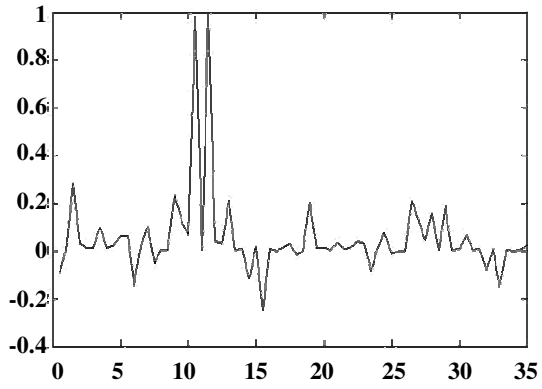

*Fig. 14: Varying of neural network coefficient μ when the number of neurons is constant in identification process.*

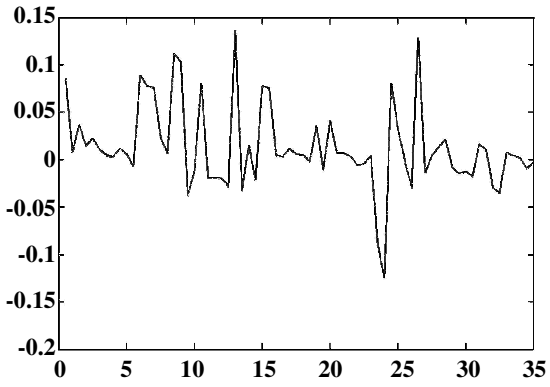

*Fig. 13: Varying of neural network coefficient α when the number of neurons is constant in identification process.*

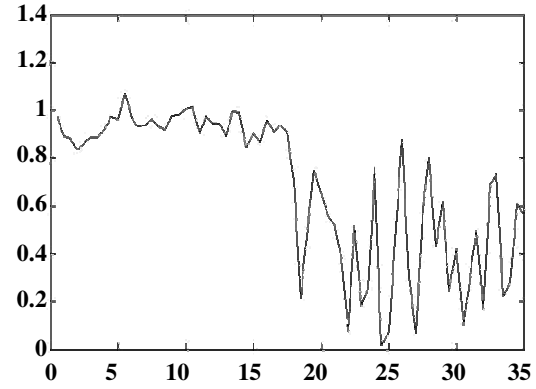

*Fig. 15: Varying of neural network coefficient σ when the number of neurons is constant in identification process.*

neural-based LMPC controller was proposed based on the well-known GPC framework. The GPC controller uses a dynamic linearization approach to extract a linear time-variant model from the most recent identified MGAP-RBF neural network around the actual system operating point at each sampling time instant. The ability of the resulting neural-based GPC controller to control the non-linear and time-varying CSTR process behaviour does not appear to suffer as a result of its linearized model simplicity.

In comparison with the LMPC, RLS-GPC and the well-tuned PID controllers, the NMPC controller showed better improvements with faster response time for both servo and regulatory objectives. However, it is noted that the computational burden of the RLS-GPC controller is much lower than the presented neural-based LMPC and NMPC methods. But, practical simulation studies revealed that the RLS-GPC approach was not able to present a good performance to cope with the constraint boundries enforced on the control action by changing its controller parameters ($N_1$, $N_2$, $N_u$, $\delta$, $\lambda$). Futhermore, the PID controller has a very simple non-adaptive structure which makes it practically attractive due to less computational demanding. But, its tuning requires expertise and can not perform well in the presence of noise and operational constraints.

*Received : Apr. 18, 2008 ; Accepted : Sept. 28, 2010*

**REFERENCES**


[1] Morari M., Lee J.H., Model Predictive Control: Past, Present and Future, *Computers and Chemical Engineering*, **21**, p. 965 (1977).

[2] Qin S.J., Badgwell T.A., "An overview of Industrial Model Predictive Control Technology", Fifth International Conference on Chemical Process Control, *AIChE*, p. 232 (1997).

[3] Narendra K.S., Parthasarathy K., Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Trans. on Neural Networks*, **1**(1), p. 4 (1990).

[4] Gori M., Tesi A., On the Problem of Local Minima in Backpropagation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **14**(1), p. 76 (1992).

[5] Chen S., Billings S.A., Cowan C.F.N., Grant P.M., Practical Identification of NARMAX Models Using Radial Basis Functions, *Int. J. Control*, **52**(6), p. 1327 (1990).

[6] S. Chen and A. Billings, Neural Networks for Nonlinear Dynamic System Modeling and Identification, *Int. J. Control*, **56**(2), p. 319 (1992).

[7] Girosi F., Poggio T., Networks and the Best Approximation Property, *Biological Cybernetics*, **63**, p. 169 (1990).

[8] Sundararajan N., Saratchandran P., Yingwei L., Radial Basis Function Neural Networks with Sequential Learning: MRAN and its Applications, River Edge: Singapore; *World Scientific: NJ*, (1999).

[9] Yingwei L., Sundararajan N., and Saratchandran P., A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function (RBF) Neural Networks, *Neural Computation*, **9**, p. 461 (1997).

[10] 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*, **16**(1), p. 57 (2005).

[11] Julier S.J., 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).

[12] 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).

[13] Huang G.-B., Saratchandran P., Sundararajan N., An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF) Networks, *IEEE Transactions on systems, man, and cybernetics, part B*, **34**(6), December (2004).

[14] Wang Y., Huang G.-B., Saratchandran P., Sundararajan N., "Time Series Study of GGAP-RBF Network: Predictions of Nasdaq Stock and Nitrate Contamination of Drinking Wate"r, Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, (2005).

[15] Julier S.J., Uhlmann J.K., Durrant-Whyte H., "A New Approach for Filtering Non-Linear Systems", Proceedings of the American Control Conference, p. 1628 (1995).

[16] Narendra K.S., Mukhopadhyay S., Adaptive Control Using Neural Networks and Approximate Models, *IEEE Trans. On neural networks*, **8**, p. 475 (1999).

[17] Powell M.J.D., A Fast Algorithm for Nonlinearly Constrained Optimization Calculations, Numerical Analysis, G.A.Watson ed., "Lecture Notes in Mathematics", Springer Verlag, **630**, (1978).

[18] Gill P.E., Murray W., Wright, M.H. Numerical Linear Algebra and Optimization, **1**, Addison Wesley, (1991).

[19] Clarke D.W., Mohtadi C., Tuffs. P.S., Generalized Predictive Control. Part I. The Basic Algorithm, *Automatica*, **23**(2), p. 137 (1987).

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

[21] Perry R.H., Chilton C.H., "Chemical Engineering Handbook", 5th ed., McGraw-Hill, New York, (1973).

[22] Piovoso M., Kosanovich K., Rokhlenko V., Guez A., "A Comparison of Three Nonlinear Controller Designs Applied to a Nonadiabatic First Order Exothermic Reaction in a CSTR", Proceedings of American Control Conference, Chicago IL, p. 490 (1992).