Journal of Universal Computer Science, vol. 13, no. 9 (2007), 1332-1343
submitted: 12/6/06, accepted: 24/10/06, appeared: 28/9/07 © *J.UCS*

Genetic Algorithm Based Recurrent Fuzzy Neural Network Modeling of Chemical Processes

Jili Tao, Ning Wang
(National Laboratory of Industrial Control Technology, Institute of Advanced Process Control, Zhejiang University, Hangzhou 310027, China
jltao@iipc.zju.edu.cn, nwang@iipc.zju.edu.cn)

Xuejun Wang
(Department of Information Science and Engineering, Central South University, Changsha 410075, China
wxjgya@126.com)

Abstract: A genetic algorithm (GA) based recurrent fuzzy neural network modeling method for dynamic nonlinear chemical process is presented. The dynamic recurrent fuzzy neural network (RFNN) is constructed in terms of Takagi-Sugeno fuzzy model. The consequent part is comprised of the dynamic neurons with output feedback. The number and the parameters of membership functions in the premise part are optimized by the GA considering both the approximation capability and structure complexity of RFNN. The proposed dynamic model is applied to a PH neutralization process and the advantages of the resulting model are demonstrated.

Key Words: Genetic algorithm, Recurrent fuzzy neural network, Modeling, PH neutralization process

Category: H.3.3

1 Introduction

Process modeling is a very important step in the control, diagnosis and optimization of the process system. The chemical process modeling is especially difficult because of its nonlinear and dynamic characters. The field of Artificial Intelligence (AI), such as Neural Network (NN), Fuzzy Theory, Expert System (ES), Genetic Algorithms (GA), has been rapidly developed in recent years. Each algorithm has their own strengths but there are still some limitations of them. In order to reduce these limitations, the hybrid algorithms were developed by combining two or three AI approaches and then drew on the strength of each to offset the weakness of the others. Thereinto, fuzzy-neural network (FNN) had emerged as one of the most active and fruitful areas of research in fuzzy logic and neural networks with the better degree of accuracy and with the smaller computational times [Buckley and Hayashi 1994][Rutkowski 2004]. However, in the field of dynamic system modeling, the most commonly used models were the recurrent neural networks (RNN), which were regarded as closed-loop systems,

with the feedback paths introducing dynamics to the model [Qin et al. 1992]. They exhibited greater prediction capabilities compared with the static ones [Mastorocostas and Theocharis 2002]. In [Lin et al. 2005], a dynamic RNN was proposed for system identification and derived good result. The feedback signal was taken from the output of the activation function, after delayed for several times, it was fed as input to the neuron. Therefore, the dynamic chemical process modeling in this work combines the recurrent neural network with Takagi-Sugeno (T-S) fuzzy model, and a recurrent fuzzy neural network (RFNN) is constructed, which is composed of the premise part and the consequent part.

Though fuzzy systems have been applied successfully in the field of industry, the generation of the fuzzy rules and the adjustment of its membership functions (MFs) were done by trial and error and/or operator's experience. Subsequently, the designers find it difficult to develop adequate fuzzy rules and membership functions to reflect the essence of the process. Moreover, some information gets lost or ignored on purpose when human operators articulate their experience in the form of linguistic rules. To overcome these drawbacks, an elitism GA for optimization of the parameters of MFs in the premise part is proposed. Since the number of MFs determines the fuzzy rules directly, which is closely related to the complexity of RFNN, a special fitness function considering both the approximation capability and structure complexity is designed in this work. Once the MFs and rules are decided, the parameters in the consequent part can be tuned by recursive least-squares (RLS) method.

The rest of the paper is organized as follows. Section 2 describes the basic architecture of T-S fuzzy model, and combines it with recurrent neural networks. Section 3 gives a detailed description of the proposed GA approach to optimize both the number and the parameters of MFs in the premise part. The simulation tests on a pH neutralization process are made in section 4. The conclusions are summarized at section 5.

2 T-S Recurrent Fuzzy Neural Networks (RFNN)

2.1 Basic Structure of the T-S Fuzzy Model

Considering a SISO system, we assume the process to be modeled is described as following:

$$y(k+1) = f(\boldsymbol{X}(k)) \tag{1}$$

where $\boldsymbol{X}(k) = [y(k), ..., y(k-n), u(k-d), ...u(k-d-m)]$, u is the input, m,n are the orders of the process while d is the time delay, respectively. A T-S type fuzzy model can be constructed to model process with its fuzzy IF-THEN rules

expressed as follows:

$$\text{Rule } j: \text{ If } x_1(k) \text{ is } A_{1j} \text{ and } x_2(k) \text{ is } A_{2j} \text{ and ... and } x_N(k) \text{ is } A_{Nj}$$
$$\text{Then } y(k+1) = \boldsymbol{B}_j^T \boldsymbol{X}(k), j = 1, 2..., M, M \leq \prod_{i=1}^{N} m_i \tag{2}$$

where $\boldsymbol{B}_j = [b_{j1}, b_{j2}, ..., b_{jN}]^T$ are parameters in the consequent part, $y(k+1)$ is the prediction output of the fuzzy model, m_i is the number of MFs for x_i, M is the number of rules. The output of fuzzy model can be expressed as follows:

$$y(k+1) = \frac{\sum_{j=1}^{M} a_j[\boldsymbol{X}(k)] \boldsymbol{B}_j^T \boldsymbol{X}(k)}{\sum_{j=1}^{M} a_j[\boldsymbol{X}(k)]} \tag{3}$$

where $a_j[\boldsymbol{X}(k)]$ is the Gaussion membership function of the inferred fuzzy set A_j ($A_j = \prod_{i=1}^{N} A_{ij}$), defined by

$$a_j[\boldsymbol{X}(k)] = \mu_1^{i_1} \mu_2^{i_2} ... \mu_N^{i_n} \tag{4}$$

$$\mu_i^j = exp\left(-\frac{||x_i - c_{ij}||}{\sigma_{ij}^2}\right) \tag{5}$$

where $i_1 \in 1, 2, ..., m_1$; $i_2 \in 1, 2, ..., m_2$; ...; $i_N \in 1, 2, ..., m_N$; c_{ij} is the center of the Gaussion membership function of the fuzzy set A_{ij}, while σ_{ij} is the width. Define fuzzy basis function (FBF) as

$$\Phi_j[\boldsymbol{X}(k)] = \frac{a_j[\boldsymbol{X}(k)]}{\sum_{l=1}^{T} a_l[\boldsymbol{X}(k)]} \tag{6}$$

The output $y(k+1)$ can then be expressed as a linear combination of FBFs in the following form.

$$y(k+1) = \sum_{j=1}^{M} \Phi_j[\boldsymbol{X}(k)] \boldsymbol{B}_j^T \boldsymbol{X}(k) \tag{7}$$

2.2 The Structure of Recurrent Fuzzy Neural Networks

Mastorocostas *et al.* presented a general structure of the dynamic recurrent neuron [Mastorocostas and Theocharis 2002]. Hence, it is easy to construct the dynamic neuron according to the consequent part of the T-S fuzzy model as shown in Figure 1. Combining with the premise part, the dynamic RFNN can be obtained in Figure 2, where $\boldsymbol{x} = [\boldsymbol{u}, \boldsymbol{y}]$ is the input vector used to connect the network to its environment. The whole RFNN consists of two parts— the premise part and the consequent part.

In Figure 2, the premise part is made up of 4 layers. The first layer is the input layer, the number of the nodes in this layer is N. Each node in the second

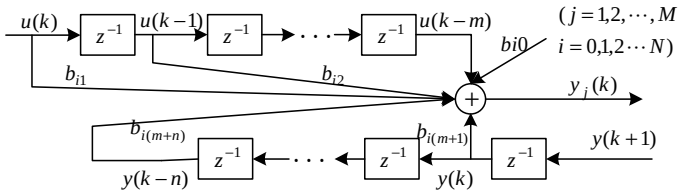

Figure 1: A representation of the dynamic neuron

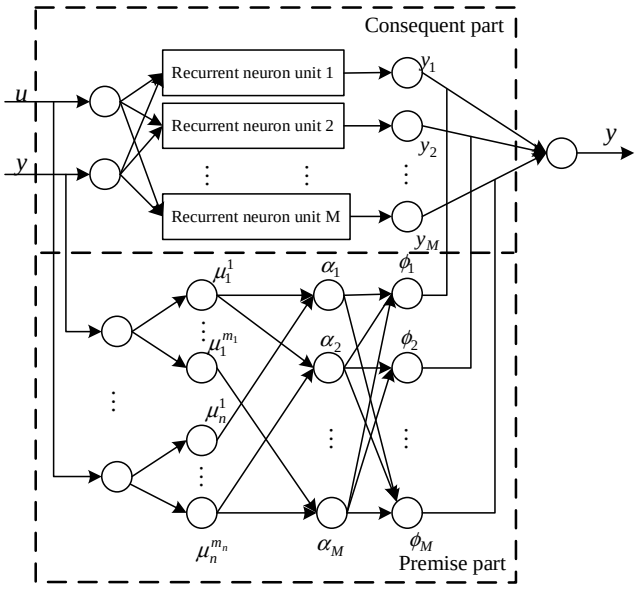

Figure 2: Schematic scheme of RFNN

layer delegates the value of a language variable, such as NM, PS, *etc*, which is used to calculate the value of membership function (μ_i^j) for each input variable. The number of nodes in layer 2 is $\sum_{i=1}^{N} m_i$. The node in the third layer is utilized to match the premises in every fuzzy rule and compute the fitness value of each rule (a_j). The number of nodes in the last layer is the same as layer 3, *i.e.*, $N_4 = N_3 = M$, which is used to figure out the value of FBFs (Φ_j). The consequent part is composed of 2 layers. The role of the first layer is the same as that in the premise part, and there exist M nodes in the second layer. Each node delegate one rule, which is used to yield y_j, $j = 1, 2, ..., M$, and is described in Figure 1. The output layer is used to produce the result of defuzzification as shown in Eq. (7).

The suggested RFNN, comprising the rules in the form of Eq. (2), is a quasi-nonlinear fuzzy model. The rules are not linked with each other, neither through external feedback nor through internal, they are connected merely via the defuzzification part. The premise and defuzzification block are static while the consequent block is dynamic, which are different from each other merely in the weighting factors of feedback. The input vector of the premise part may be different from that of consequent part.

3 Tuning the Parameter of RFNN

Suppose the input vector of RFNN is fixed in advance. The parameters need to learn are the weights $b_{jk}(j = 1, 2, ..., M,\ k = 1, 2, ..., N)$ and the centers (c_{ij}) and widths (σ_{ij}) of MFs.

3.1 Learning Algorithm of the Weights

If the membership function is decided a priori, *i.e.*, those FBFs $\Phi_j[\boldsymbol{X}(k)]$ have already been decided. The parameters in the consequent part $(\boldsymbol{B}_j)$ can then be easily estimated using least-squares (LS) algorithm. Denote

$$\Theta = [\boldsymbol{B}_1^T \boldsymbol{B}_2^T ... \boldsymbol{B}_M^T]^T \tag{8}$$

$$\Psi = [\Phi_1[\boldsymbol{X}(k)]\boldsymbol{X}(k)^T, \Phi_2[\boldsymbol{X}(k)]\boldsymbol{X}(k)^T, ..., \Phi_M[\boldsymbol{X}(k)]\boldsymbol{X}(k)^T]^T \tag{9}$$

Substitute Eq.(8) and Eq.(9) into Eq.(7) yields

$$y(k+1) = \Psi(k)^T \Theta \tag{10}$$

Also denote

$$\Phi = [\Psi(1), \Psi(2), ..., \Psi(N_n)] \tag{11}$$

$$\boldsymbol{Y}_d = [y_d(2), y_d(3), ..., y_d(N_n + 1)] \tag{12}$$

Rearrange Eq.(10) for $k = 1, 2, ..., N_n$, yields

$$\boldsymbol{Y}_d = \Phi\Theta \tag{13}$$

Eq.(13) is in the form of a linear regression model. If the reverse of matrix $(\Phi^T\Phi)$ exist, the parameters matrix Θ can be uniquely determined by the conventional LS method.

$$\Theta = [\Phi^T\Phi]^{-1}\Phi^T\boldsymbol{Y}_d \tag{14}$$

However, if Φ is poorly conditioned, the parameter obtained by Eq.(14) will change dramatically if the elements of $\boldsymbol{Y}_d$ are modified slightly and result in poor predictions of $\hat{\boldsymbol{Y}}_d$ when used Φ with new data [William and Ronald 2006]. This is always the case because sufficient training data are very difficult to be obtained for most processes. To solve this problem, the recursive least-squares (RLS) method is adopted in this work.

3.2 Optimization of the Numbers and Parameters of MFs

For almost every process, we can get some linguistic information from domain experts or operators. Once the fuzzy model is mapped into the networks shown in Figure 2, and the weights are obtained as described in section 3.1, there exist $2\sum_{i=1}^{N} m_i$ parameters to be optimized. However, optimization of those parameters cannot be easily solved by the standard optimization methods. An interesting alternative for solving this complicated problem can be offered by the recently developed evolutionary computation methods-GAs, which are the most popular and successful strategies among these evolutionary methods [Salomon 1996]. In a GA, the population is the set of possible solutions, and each individual of this population is characterized by chromosome-like structures. The possibilities of survival of each individual are evaluated by the cost function. The result of this evaluation is called fitness and plays an important role in selection and reproduction. The evolution is achieved by the application of genetic operators, such as selection, crossover and mutation. The optimization of the numbers and parameters of MFs in RFNN is described at following sections.

3.2.1 Coding Method

There are totally $2\sum_{i=1}^{N} m_i$ parameters to be optimized in the RFNN, which means one chromosome should be able to delegate $2\sum_{i=1}^{N} m_i$ real number. Hence, binary coding chromosome will become too complex, and decimal coding chromosome is adopted. The structure of lth chromosome is shown as follows.

$$\boldsymbol{C}_l = \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1m_1} & 0 & \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1m_1} & 0 \\ c_{21} & c_{22} & \ldots & c_{2m_1} & 0 & \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2m_1} & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ c_{N1} & c_{N2} & \ldots & c_{Nm_1} & 0 & \sigma_{N1} & \sigma_{N2} & \ldots & \sigma_{Nm_1} & 0 \end{bmatrix} \tag{15}$$

where $l = 1, 2, \ldots L$, L is the size of the population, m_i is produced randomly between 1 and D, D is the maximum number of MFs. Without loss of generality, D is set as 11, and N as 2. The elements of $\boldsymbol{C}_l$ are computed by the following equation:

$$c_{ij} = x_{imin} + r(x_{imax} - x_{jmin}) \quad 1 \leq i \leq N, 1 \leq j \leq m_i \tag{16}$$

$$\sigma_{ij} = 0.01 + r(\frac{x_{imax}}{2} - 0.01) \quad 1 \leq i \leq N, 1 \leq j \leq m_i \tag{17}$$

where r is a random number between 0 and 1, x_{jmin} and x_{jmax} is the minimum and maximum values of input variables given in the problem. When L premise parts are generated, the corresponding weights of Lconsequent parts can be obtained using RLS method described in section 3.1. Thus, the formulation of L RFNNs is completed, which can be represented by the pairs $(\boldsymbol{C}_1, \Theta_1), (\boldsymbol{C}_2, \Theta_2), ..., (\boldsymbol{C}_L, \Theta_L)$.

3.2.2 Fitness Function

Once L RFNNs are constructed, the predictions $\hat{Y}_{d1}, \hat{Y}_{d2}, ..., \hat{Y}_{dL}$ can be obtained and the corresponding errors E_l, are computed as follows:

$$E_l = ||Y_d - \hat{Y}_{dl}||_2^2 \tag{18}$$

In terms of section 2, the complexity of RFNNs mainly lies in the number of input (N)) and its number of MFs ($\sum_{i=1}^{N} m_i$). Since the number of input is decided a priori, the number of MFs is included in the fitness function to simplify the structure of RFNN. Hence, fitness function considering both approximation capability and structure complexity is shown as follows.

$$J_l(\boldsymbol{C}_i, \Theta_i) = E_l + \lambda \sum_{i=1}^{N} m_i \tag{19}$$

This criterion expresses a compromise between the cost of modeling errors and the cost of the complexity of network structure, where λ is the weighting factor.

3.2.3 Population Evolution

Once the codification and fitness function have finished, the next step will be the application of the evolutionary algorithm. The GA used in this work is a standard genetic algorithm (SGA) with elitism, and its flowchart is shown in Figure 3. The first step of the GA is the generation of an initial random population. And after the initialization, the fitness of every individual in the population is obtained through the cost function evaluation. If the ending condition is not satisfied, the next step will be population evolution (applying genetic operators). The ending condition can be the maximum evolution generation, the degree of convergence, or any other. Moreover, Elitism, the inclusion of the best current set in the next population, is used throughout.

Selection. A set of individuals from the previous population must be selected for reproduction. This selection depends on their fitness values. Individuals with better fitness values will more probably survive. There exist different types of selection operators, and in this work Roulette wheel method is applied. The probability of an individual being selected, $P(\boldsymbol{C}_l)$, is given by:

$$P(\boldsymbol{C}_l) = \frac{f(\boldsymbol{C}_l)}{\sum_{l=1}^{L} f(\boldsymbol{C}_l)} \tag{20}$$

where $f(\boldsymbol{C}_l)$ is the fitness value of the individual $\boldsymbol{C}_l$, which is obtained by $\frac{1}{J(\boldsymbol{C}_l, \Theta_l)}$. The roulette wheel is placed with L equally spaced pointers. A single spin of the roulette wheel will simultaneously pick all the members of the next

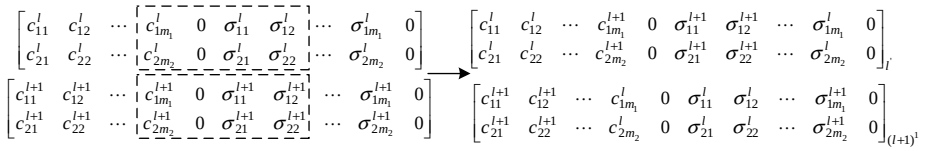

Figure 3: Schematic scheme of the crossover operation

population.

Crossover. The crossover operator is applied after selection, and this produces new individuals. Not all the selected individuals are crossed over, this depends on the crossover probability, p_c. Crossover operation is executed between the current chosen individual $\boldsymbol{C}_l$ and the next individual $\boldsymbol{C}_{l+1}$, and yields the offspring chromosomes $\boldsymbol{C'}_l$,$\boldsymbol{C'}_{l+1}$. Since the chromosome is composed of the centers and widths, two points are chosen among the centers and widths, respectively. The procedure is demonstrated with Figure 3, which contains a scheme of two-point crossover.

Mutation. To have a better exploration of the search space, mutation operator is carried out. And when the element of an individual is mutated with a probability p_m, it is replaced by a new generated element in terms of Eq. (16) and (17).

4 Simulation Results

The proposed GA based RFNN modeling approach will now be applied to a simulated PH neutralization process [Nahas et al. 1992]. The process model consists of three nonlinear ordinary differential equations and a nonlinear output equation:

$$\dot{h} = q_1 + q_2 + q_3 + C_v h^{0.5} W_{a4} \tag{21}$$

$$\dot{W}_{a4} = \frac{1}{Ah}[(W_{a1} - W_{a4})q_1 + (W_{a2} - W_{a4}) + (W_{a3} - W_{a4})] \tag{22}$$

$$\dot{W}_{b4} = \frac{1}{Ah}[(W_{b1} - W_{b4})q_1 + (W_{b2} - W_{b4}) + (W_{b3} - W_{b4})] \tag{23}$$

$$W_{a4} + 10^{pH_4 - 14} + W_{b4}\frac{1 + 2 \times 10^{pH_4 - pK_2}}{1 + 10^{pK_1 - pH_4} + 10^{pH_4 - pK_2}} - 10^{-pH_4} = 0 \tag{24}$$

where h is the liquid level, W_{a4} and W_{b4} are the invariants of the effluent stream, q_1, q_2 and q_3 are the acid, buffer and base flow rate, respectively. The sampling period is chosen as 0.25 min. It is desired to control pH_4 by the base flow rate q_3. The interesting operation range is defined by $0 \leq pH \leq 14$ and $0 \leq q_3 \leq 40$ ml\s The objective is to build discrete dynamical models for predicting the value of pH_4

using as input variables base flow rate q_3 and previous values of pH_4 According to the export knowledge of the pH process, the input of consequent part is chosen as:

$$\boldsymbol{X}(t) = [u(k), u(k-1, u(k-2), y_{FNN}(k), y_{FNN}(k-1), y_{FNN}(k-2)]^T \quad (25)$$

The input of premise part is selected as the current output (pH_4). For training and testing the GA based RFNN model, we created a set of 500 input-output data by selecting randomly the values of the base flow rate q_3 within the space [0,40].The first 300 data points is used for training RFNN, while the left is used

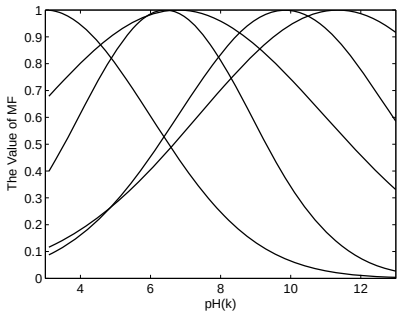

Figure 4: MFs optimized by GA

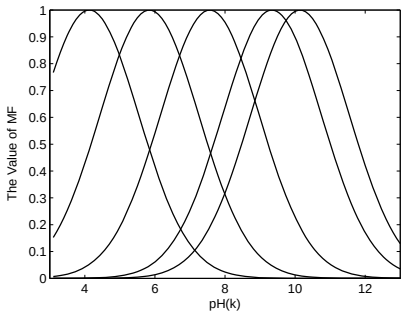

Figure 5: MFs obtained by k-means

to test the performance of the proposed RFNN. The operational parameters used in the proposed GA are listed as follows: number of chromosomes $L = 30$, maximum number of MFs $D = 11$, weighting factor $\lambda = 0.5$, number of generations $G = 150$, probability of crossover $p_c = 0.8$, probability of mutation $p_m = 0.01$. After the optimization of RFNN, 5 MFs of the premise part are shown in Figure 4, and the trained T-S type RFNN model is shown as following: If $pH_4(k)$ is A_1 then

$$y_{FNN} = 5.1878 - 0.0229u(k) + 0.0137u(k-1) + 0.0097u(k-2) \\ -0.0592y_{FNN}(k) - 0.0446y_{FNN}(k-1) - 0.0150y_{FNN}(k-2)$$

If $pH_4(k)$ is A_2 then

$$y_{FNN} = 21.2761 - 0.0716u(k) - 0.0022u(k-1) - 0.0024u(k-2) \\ +0.0109y_{FNN}(k) + 0.0100y_{FNN}(k-1) + 0.0034y_{FNN}(k-2)$$

If $pH_4(k)$ is A_3 then

$$y_{FNN} = -1.3595 - 0.0685u(k) + 0.0930u(k-1) + 0.061u(k-2) \\ -0.0394y_{FNN}(k) - 0.0287y_{FNN}(k-1) - 0.0095y_{FNN}(k-2)$$

If$pH_4(k)$ is A_4 then

$$y_{FNN} = 5.7630 - 0.0933u(k) + 0.0096u(k-1) + 0.0061u(k-2) \\ -0.0403y_{FNN}(k) - 0.0284y_{FNN}(k-1) - 0.0094y_{FNN}(k-2)$$

If$pH_4(k)$ is A_5 then

$$y_{FNN} = 4.5970 + 0.1954u(k) - 0.0207u(k-1) - 0.0133u(k-2) \\ +0.0873y_{FNN}(k) + 0.0627y_{FNN}(k-1) + 0.0210y_{FNN}(k-2)$$

Thus, the RFNN model is employed to give predictions on testing data set. The prediction result on the testing data sequence is compared with the result obtained by a general RFNN. In this method, the number of membership functions is the same as GA-RFNN, the centers are obtained by k-means method [Darken and Moody 1990], and the width is set as 2, as shown in Figure 5. And the parameters of the sequent part are obtained by RLS. From the modeling errors shown in Figure 6 and Figure 7, we can see that the proposed GA-RFNN model is superior to general RFNN model.

For the dynamic process modeling, if the input keeps unchanged, the output of dynamic process model will arrive at certain point if it is stable. This means that there exists a static relation between the input and the steady-state value of the output. This relation is well known as steady-state response in process industry. Here we examine the steady-state response of the two methods and the real process. The simulation results are shown in Figure 8 and Figure 9. From the above comparison, we can see that after the optimization of MFs in the premise part by GA, the structure of RFNN can be simplified while the approximation capability is improved greatly. Since the training data include the whole operating condition, the GA-RFNN fit the real process quite well. Without optimization, the RFNN using the same training data is inferior to GA-RFNN.

5 Conclusions

The issue of the dynamic modeling approach is investigated by GA based RFNN. The consequent part of RFNN is a linear combination of dynamic recurrent neurons, whose parameters are derived by RLS. The GA is designed which performs both structure and parameter optimization for the number and parameters of MFs in the premise part of RFNN. Based on the GA optimization method, the suggested RFNN modeling approach does not require accurate knowledge of the process. Performance comparisons on a nonlinear pH neutralization process are made with regard to general RFNN modeling method and demonstrate the efficiency of the proposed method.

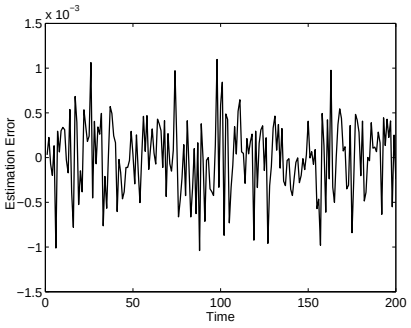

Figure 6: Modeling errors of GA-RFNN

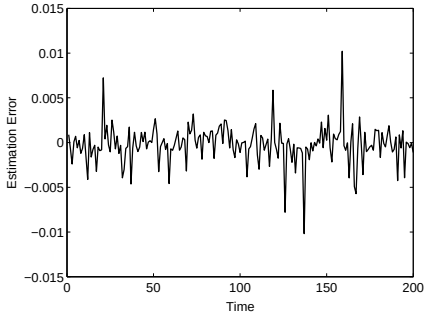

Figure 7: Modeling errors of RFNN

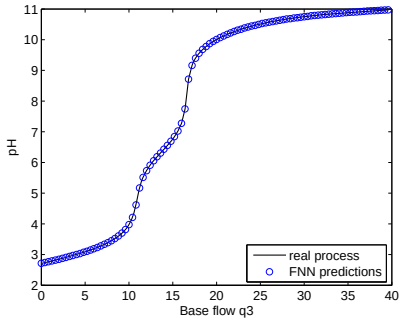

Figure 8: Prediction results by GA-RFNN

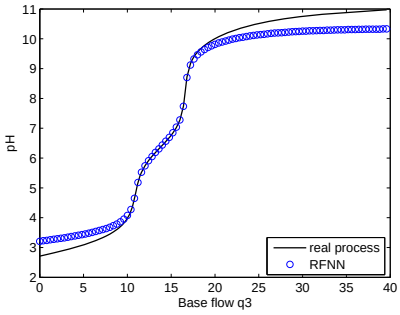

Figure 9: Prediction results by RFNN

Acknowledgements

This work is in part supported by the project of the National Natural Science Foundation of China (60421002).

References

[Buckley and Hayashi 1994] Buckley J., Hayashi Y.: "Fuzzy Neural Networks: A Survey"; Fuzzy Set. Syst., 66, 1(1994), 1-13

[Rutkowski 2004] Rutkowski L.: "New Soft Computing Techniques for System Modelin, Pattern Classification and Image Processing"; Springer Verlag, New York (2004)

[Qin et al. 1992] Qin S. Z., Su H. T. , McAvoy T. J.: "Comparison of Four Neural Net Learning Methods for Dynamic System Identificatio"; IEEE Trans. Neu. Net., 3, 1(1992), 122-130

[Mastorocostas and Theocharis 2002] Mastorocostas P. A., Theocharis J. B.: "A Recurrent Fuzzy-neural Model for Dynamic System Identificatio"; IEEE Trans Syst, Man, Cybern Part B: Cybernetics, 32, 2(2002), 176-190

[Lin et al. 2005] Lin C. J., Chen C. H., Chen H. J.: "Dynamic System Identification Using Pseudo-Gaussian-Based Recurrent Compensatory Fuzzy Neural Networks"; J. Chin. Inst. Chem. Eng, 28, 1(2005), 55-65

[William and Ronald 2006] William C. H., Ronald L. O.: "Results on the Bias and Inconsistency of Ordinary Least Squares for the Linear Probability Model"; Econ. Lett, 90, 3(2006), 321-327

[Salomon 1996] Salomon R.: "Re-evaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions"; Biosystem, 39, 3(1996), 263-278

[Nahas et al. 1992] Nahas E. P., Henson M. A. ,Seborg D. E.: "Nonlinear Internal Model Control Strategy for Neural Network Models"; Comp. Chem. Eng, 16, 12(1992), 1039-1057

[Darken and Moody 1990] Darken C., Moody J.: "Fast Adaptive K-means Clustering: Some Empirical Results"; Neu. Net, IJCNN, (1990), 233-238