Jurnal Teknik Elektro Vol. 7, No. 2, September 2007: 82 - 87

Parameter Estimation using Least Square Method for MIMO Takagi-Sugeno Neuro-Fuzzy in Time Series Forecasting

Indar Sugiarto[1], Saravanakumar Natarajan[2]
[1]Department of Electrical Engineering – Petra Christian University
[2]Institut für Automatisierungstechnik – Universität Bremen
Email: indi@peter.petra.ac.id, natarajan@uni-bremen.de

ABSTRACT

This paper describes LSE method for improving Takagi-Sugeno neuro-fuzzy model for a multi-input and multi-output system using a set of data (Mackey-Glass chaotic time series). The performance of the generated model is verified using certain set of validation / test data. The LSE method is used to compute the consequent parameters of Takagi-Sugeno neuro-fuzzy model while mean and variance of Gaussian Membership Functions are initially set at certain values and will be updated using Back Propagation Algorithm. The simulation using Matlab shows that the developed neuro-fuzzy model is capable of forecasting the future values of the chaotic time series and adaptively reduces the amount of error during its training and validation.

Keywords: forecasting, time series, gaussian membership function, neuro-fuzzy, least square.

INTRODUCTION

When considering fuzzy logic for analyzing and solving certain problem in engineering which requires computational intelligence, practically ones choose between Mamdani model and Takagi-Sugeno model. Of course another model also could be used as well. But, Takagi-Sugeno model is preferred in the case of data-based (or numerically-data-driven) fuzzy modeling [1, 2, 3] as well as forecasting time series data where in many cases it can be seen as system with locally linear model. Another advantage is that the inference formula of the Takagi-Sugeno model is only two-step procedure, based on a weighted average defuzzifier, whereas Mamdani type of fuzzy model basically consists of four steps [4, 5].

In this paper, we implement neuro-fuzzy modeling which combines the ability of reasoning on a linguistic level and handling of uncertain or imprecise data with the ability of learning and adaptation certain aspects of intelligent system. Here we choose Takagi-Sugeno type fuzzy model along with differentiable operators and membership function and also the weighted average defuzzifier for defuzzification of output data. The corresponding output inference can then be represented in a multilayer feedforward network structure such as described in [4, 5] which is identical to the architecture of ANFIS [6]. Since we use Takagi-Sugeno fuzzy model with Gaussian Membership Function (GMF), then we have three types of free parameters: mean and variance of each GMFs and also Takagi-Sugeno Rule's Consequent Parameters (we usually name them as theta). Optimizing these parameters so that the speed of convergence can be accelerated becomes special interest in the research field of neuro-fuzzy technique. This paper concentrates our main interest in the optimization of theta by estimation using least square method (LSE) for MIMO system. So, we arrange our presentation as follows. First, we explain the importance of MIMO Takagi-Sugeno Neuro-Fuzzy implementation for time series forecasting and treatment procedure of data, in this case we use Mackey-Glass chaotic time series. Next, we explain how we implement least square method for optimizing the free parameters of Takagi-Sugeno Rule's Consequent Parameter. All important equations or equation(s) derived directly from its definition will be numbered, while its derivation steps will not be numbered. Finally, the result from Matlab simulation will be discussed.

Note: Discussion is impected before December, 1st 2007, and will be publiced in the "Jurnal Teknik Elektro" volume 8, number 1 March 2008.

MIMO TAKAGI-SUGENO NEURO-FUZZY FOR FORECASTING CHAOTIC TIME SERIES

Here we consider short-term forecasting as the implementation case of MIMO Takagi-Sugeno Neuro-Fuzzy for chaotic time series. The chaotic time series are generated from deterministic nonlinear systems that are sufficiently complicated as they appear to be random. The chaotic time series we use here is Mackey-Glass Time Series which available in Matlab. This chaotic time series generated by solving the Mackey-Glass time delay differential equation [7]:

$$(dx/dt) = (0.2*x(t-\delta)/(1+x^{10}(t-\delta)))-0.1*x(t) \quad (1)$$

where $x(0)=1.2$, $\delta=17$ and $x(t)=0$ for $t<0$.

Since we work with time series data which already exist, then data preparation for forecasting consists only structuring of data and determination of training and validation data set. According to Matlab documentation, Mackey-Glass data consists of 1200 data points. For our experiment purpose, we neglected the first 100 data due to its harmonic nature, then we took next 200 data for training, and finally we used the next 800 data for validation. So, we have time series in the form of $X = \{X_1, X_2, X_3, \ldots, X_n\}$ which is taken at regular intervals of time $t = \{1, 2, 3, \ldots, n\}$. For this forecasting problem, usually a set of known values of the time series up to a point in time, say t, is used to predict the future value of the time series at some point, let's say $(t+P)$. Then we created a mapping from S sample data points, sampled every s units in time, to a predicted future value of the time series at time point which resulted an S-dimensional vector of the form:

$$XI(t) = [X\{t-(S-1)s\}, X\{t-(S-2)s\}, \ldots, X\{t\}]$$

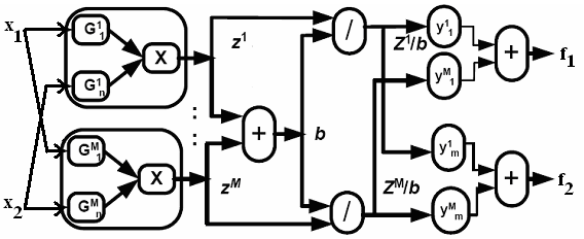
Palit [3] shows that for predicting the Mackey-Glass time series, the value of S=4 and s = P = 6 are good choices. The output data of the fuzzy predictor corresponds to the trajectory prediction, where for MIMO case, it can be written as:

$$XO(t) = [X\{t+P\}, X\{t+2P\}, X\{t+3P\}, \ldots]$$

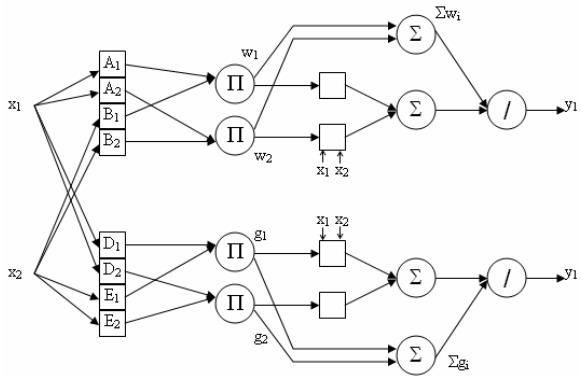
For n-times of sampling data, we combine vector XI and XO, name it as XIO which stands for Input-Output Matrix. For our MIMO case, we use 4 input and 3 output so that the XIO matrix can be written in the form:

$$XIO = \underbrace{\begin{bmatrix} X_1 & X_2 & X_3 & X_4 \\ X_4 & X_5 & X_6 & X_7 \\ \ldots & \ldots & \ldots & \ldots \\ X_{n-6} & X_{n-5} & X_{n-4} & X_{n-3} \end{bmatrix}}_{\text{XI matrix}} \underbrace{\begin{bmatrix} X_5 & X_6 & X_7 \\ X_8 & X_9 & X_{10} \\ \ldots & \ldots & \ldots \\ X_{n-2} & X_{n-1} & X_n \end{bmatrix}}_{\text{XO matrix}}$$

The structure of Takagi-Sugeno Neuro-Fuzzy we use here is the same with that one described in [4, 5] which is similar with MANFIS (Multiple-output ANFIS) model [1, 6]. For convenient, here we redraw the neuro-fuzzy model proposed by [5] in comparison with MANFIS model [1], both for two input and two output system.



(a)



(b)

Figure 1. Structure of Takagi-Sugeno neuro-fuzzy as proposed by Palit (a) and Jang (b)

Referring to figure 1.a above, the fuzzy logic system considered here for constructing neuro-fuzzy structures is based on Takagi-Sugeno fuzzy model with Gaussian membership functions. It uses product inference rules and a weighted average defuzzifier-defines as:

$$f_j(x^p) = \frac{\sum_{l=1}^{M} y_j^l z^l}{\sum_{l=1}^{M} z^l}, \qquad (2)$$

where $j = 1,2,3,\ldots m$ reflects the number of outputs, and $l = 1,2,3,\ldots M$ reflects the number of rules (Gaussian membership function).

The output consequent of the l^{th} rule for each output j^{th} is:

$$y_j^l = \left(\theta_{0j}^l + \sum_{i=1}^{n} \theta_{ij}^l x_i \right) \qquad (3)$$

where $i = 1, 2, 3, \ldots n$ reflects the number of input.

Then we generate degree of fulfillment and fuzzification procedure as:

$$z^l = \prod_{i=1}^{n} \mu_{G_i^l}(x_i) \text{ where } \mu_{G_i^l}(x_i) = e^{-\frac{(x_i - c_i^l)^2}{\sigma_i^{l2}}} \qquad (4)$$

For fuzzification using Gaussian membership function with the corresponding mean and variance parameters c_i^l and σ_i^l, we assume that $c_i^l \in I_i, \sigma_i^l > 0,$ and $y_j^l \in O_j$, where I_i and O_j are the input and output universes of discourse respectively. The corresponding l^{th} rule from the above fuzzy logic system can be written as

$$R_l = If\ x_1\ is\ G_1^l\ and\ x_2\ is\ G_2^l\ and\ ...x_n\ is\ G_n^l$$
$$then\ \ y_l^j = \theta_{oj}^l + \theta_{1j}^l x_1 + \theta_{2j}^l x_2 + ... + \theta_{nj}^l x_n$$

Where, x_i are the *n* system inputs, y_l^j are *M* rules output consequents, and f_j are its *m* outputs. If all the parameters for neuro-fuzzy network are properly selected, then the system can correctly approximate any nonlinear system based on given *XIO* matrix data.

For training that neuro-fuzzy, which equivalent with multi-input multi-output (MIMO) feedforward network, we use backpropagation algorithm (BPA). Generally, BPA based on Steepest Descent Gradient Rule uses a low learning rate (η) for network training in order to avoid the oscillations in the final phase of the training. That is why BPA needs large number of epochs (recursive steps) for the smallest SSE (Sum of Squared Error). The steepest descent gradient rule, used for Neuro-Fuzzy network training is based on the recursive expression:

$$\theta_{0j}^l(k+1) = \theta_{0j}^l(k) - \eta.(\partial S / \partial \theta_{0j}^l)$$
$$\theta_{ij}^l(k+1) = \theta_{ij}^l(k) - \eta.(\partial S / \partial \theta_{ij}^l)$$
$$c_i^l(k+1) = c_i^l(k) - \eta.(\partial S / \partial c_i^l)$$
$$\sigma_i^l(k+1) = \sigma_i^l(k) - \eta.(\partial S / \partial \sigma_i^l)$$

where *j = 1,2,3,...m* reflects the number of outputs, *l = 1,2,3,...M* reflects the number of rules (Gaussian membership function), and *i = 1, 2, 3, ... n* reflects the number of input. In the sense of BPA, *θ, c*, and *σ* are parameters that will be updated during the training phase. Here we differentiate θ_0 from θ_i since it will not contain any part of input vector, hence will be calculated independently. The SSE for each epoch reflects the training performance function. Here we use notation S for the squared error, which can be described as:

$$S_j = 0.5 \cdot \sum_{p=1}^{N} (e_j^p)^2 = 0.5 \cdot E_j^T E_j \qquad (5)$$

and the SSE will be

$$SSE = \sum_{j=1}^{m} S_j$$

where E is the error vector in which its element e_j is the difference between the real sampled data from XIO matrix and its prediction, or, $e_j = (d_j - f_j)$.

With E = [e_1 e_2 e_3 … e_n], then we can derive $\partial S / \partial \theta_0^l$ as follows:

$$S = 0.5 \cdot E^T \cdot E = 0.5 \cdot [e_1^2 + e_2^2 + ... e_n^2]$$
$$\frac{\partial S}{\partial \theta_0^l} = \frac{1}{2} \cdot \left[2 \cdot \frac{\partial e_1^2}{\partial \theta_0^l} \cdot e_1 + ... + 2 \cdot \frac{\partial e_n^2}{\partial \theta_0^l} \cdot e_n \right] = \left[\frac{e_1 \partial e_1^2}{\partial \theta_0^l} + \frac{e_2 \partial e_2^2}{\partial \theta_0^l} ... + \frac{e_n \partial e_n^2}{\partial \theta_0^l} \right] \qquad (6)$$

Recall that $e_j = (d_j - f_j)$ and also $f_j = \frac{\sum_{l=1}^{M} y_j^l z^l}{\sum_{l=1}^{M} z^l}$, whereas d_j taken directly from XIO matrix, then we can derive $\partial f_j / \partial \theta_0^l$ as follows:

$$\frac{\partial f_j}{\partial \theta_0^l} = \frac{\partial}{\partial \theta_0^l} \left(y_j^1 \cdot \frac{z^l}{\sum_{l=1}^{M} z^l} + ... + y_j^m \cdot \frac{z^l}{\sum_{l=1}^{M} z^l} \right) \qquad (7)$$

With $\frac{\partial e_j}{\partial \theta_0^l} = \frac{\partial}{\partial \theta_0^l}(d_j - f_j) = -\frac{\partial f_j}{\partial \theta_0^l}$

Because z^l is independent from θ_0^l then

$$\frac{\partial f_j}{\partial \theta_0^l} = \left(\frac{z^l}{b} \cdot \frac{\partial y_j^1}{\partial \theta_0^l} + ... + \underbrace{\frac{z^M}{b} \cdot \frac{\partial y_j^m}{\partial \theta_0^l}}_{=0 \text{ if } l \neq m} \right) = \frac{z^l}{b} \cdot 1 = \frac{z^l}{b}$$

where, $b = \sum_{l=1}^{M} z^l$

Then we have

$$\frac{\partial S}{\partial \theta_0^l} = \left[\frac{e_1 \partial e_1^2}{\partial \theta_0^l} + \frac{e_2 \partial e_2^2}{\partial \theta_0^l} ... + \frac{e_n \partial e_n^2}{\partial \theta_0^l} \right] = \left[(f_1 - d_1) \cdot \frac{\partial f_1}{\partial \theta_0^l} + ... + (f_n - d_n) \cdot \frac{\partial f_n}{\partial \theta_0^l} \right] \qquad (8)$$

or in matrix form we can write it as:

$$\frac{\partial S}{\partial \theta_0^l} = \left[(f_j - d_j) \right] \bullet \left[\frac{z_j^l}{b_j} \right] \qquad (9)$$

Using the same method described above, the rest of derivative for the adjustable parameters can be obtained as:

$$(\partial S / \partial \theta_{ij}^l) = (f_j - d_j) \cdot (z^l / b) \cdot x$$
$$(\partial S / \partial c_i^l) = A \cdot \{2 \cdot (z^l / b) \cdot (x_i - c_i^l) / (\sigma_i^l)^2\}$$
$$(\partial S / \partial \sigma_i^l) = A \cdot \{2 \cdot (z^l / b) \cdot (x_i - c_i^l)^2 / (\sigma_i^l)^3\}$$

where

$$A = \left(\sum_{j=1}^{m} \left(y_j^l - f_j \right) \cdot \left(f_j - d_j \right) \right) = \left(\sum_{j=1}^{m} \left(\left(\theta_{0j}^l + \sum_{i=1}^{n} \theta_{ij}^l \cdot x_i \right) - f_j \right) \cdot \left(f_j - d_j \right) \right) \quad (10)$$

LEAST SQUARE METHOD FOR OPTIMIZING THE FREE PARAMETERS OF TAKAGI-SUGENO RULE'S CONSEQUENT PARAMETER

Consider the following rules in Takagi-Sugeno fuzzy model:

$$R_1 : If\ x_1\ is\ G_1^1\ and\ x_2\ is\ G_2^1\ and\ \ldots x_n\ is\ G_n^1$$
$$then\quad y_l^1 = \theta_{oj}^1 + \theta_{1j}^1 x_1 + \theta_{2j}^1 x_2 + \ldots + \theta_{nj}^1 x_n$$

$$R_2 : If\ x_1\ is\ G_1^2\ and\ x_2\ is\ G_2^2\ and\ \ldots x_n\ is\ G_n^2$$
$$then\quad y_l^2 = \theta_{oj}^2 + \theta_{1j}^2 x_1 + \theta_{2j}^2 x_2 + \ldots + \theta_{nj}^2 x_n$$

$$R_M : If\ x_1\ is G_1^M\ and\ x_2\ is G_2^M\ and \ldots x_n\ is G_n^M$$
$$then\quad y_l^M = \theta_{oj}^M + \theta_{1j}^M x_1 + \theta_{2j}^M x_2 + \ldots + \theta_{nj}^M x_n$$

Therefore, for a given set of inputs the corresponding Takagi-Sugeno inference will be:

$$f_j = \frac{\sum_{l=1}^{M} y_j^l \cdot z^l}{\sum_{l=1}^{M} z^l} = \frac{\left(z^1 \cdot y_j^1 + z^2 \cdot y_j^2 + \ldots + z^M \cdot y_j^M \right)}{z^1 + z^M + \ldots + z^M} \quad (11)$$

Where z^l is the degree of fulfillment of the l^{th} rule, which is computed for the n input system using the product operator as:

$$z^l = \prod_{i=1}^{n} \mu_{G_i^l}(x_i) \quad (12)$$

We can use the normalized degree of fulfillment for l^{th} rule in the form: $\gamma^l = \dfrac{z^l}{\sum_{l=1}^{M} z^l}$

so that the corresponding Takagi-Sugeno inference for i^{th} training sample will be:

$$f_i = \gamma_i^1 \cdot \left(\theta_0^1 + \theta_1^1 \cdot x_{1,i} + \ldots + \theta_n^1 \cdot x_{n,i} \right) + \gamma_i^2 \cdot \left(\theta_0^2 + \theta_1^2 \cdot x_{1,i} + \ldots + \theta_n^2 \cdot x_{n,i} \right) + \ldots . + \gamma_i^M \cdot \left(\theta_0^M + \theta_1^M \cdot x_{1,i} + \ldots + \theta_n^M \cdot x_{n,i} \right)$$

Recall that XIO matrix is composed of XI and XO matrices. We can apply Least Square Estimation using XI matrix into matrix f_j. But first, we append 1 along with n inputs in XI which take care of θ_0^l from the rule consequent and name this new matrix as XIe so that the consequent parameters can be written as:

$$\theta^l = \left[\theta_0^l, \theta_1^l, \theta_2^l, \ldots, \theta_n^l \right]^T$$

and the XIe vector will have the form of:

$$XIe = [1, X\{t-(S-1)s\}, X\{t-(S-2)s\}, \ldots, X\{t\}]$$

here we use S = 4 and s = 6 (as described in the previous section).

Then the matrix form of the corresponding Takagi-Sugeno inference can be written as:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} \gamma_1^1 \cdot XIe_1 & \gamma_1^2 \cdot XIe_1 & \ldots. & \gamma_1^2 \cdot XIe_1 \\ \gamma_2^1 \cdot XIe_2 & \gamma_2^2 \cdot XIe_2 & \ldots. & \gamma_2^2 \cdot XIe_2 \\ & \vdots & & \\ \gamma_N^1 \cdot XIe_N & \gamma_N^2 \cdot XIe_N & \ldots. & \gamma_N^2 \cdot XIe_N \end{bmatrix} \cdot \begin{bmatrix} \theta^1 \\ \theta^2 \\ \vdots \\ \theta^M \end{bmatrix}$$

The Least Square Estimation in matrix form, therefore

$$[f] = [\gamma XIe] \cdot [\theta], \text{ or,}$$
$$[\gamma XIe]^T \cdot [\gamma XIe] \cdot [\theta] = [\gamma XIe]^T \cdot [f]$$

which yield:

$$[\theta] = ([\gamma XIe]^T \cdot [\gamma XIe])^{-1} \cdot [\gamma XIe]^T \cdot [f] \quad (13)$$

SIMULATION USING MATLAB

We wrote all of the equations described above in matrix form and we use Matlab to simulate them so that we can observe its performance. For the first experiment, we generate matrix c (mean) and σ (variance) of the Gaussian membership function randomly and matrix θ is also generated randomly. Here is the result from the simulation using randomly generated free parameters of Takagi-Sugeno neuro-fuzzy network.

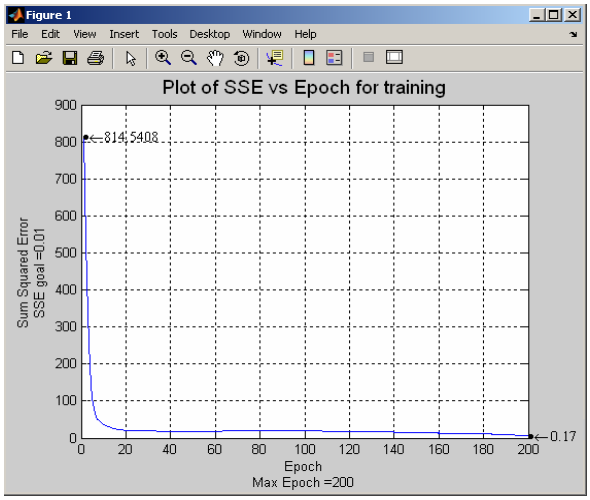


Figure 2. Plot of SSE vs Epoch for training using randomly generated free parameters of Takagi-Sugeno neuro-fuzzy network

Since all parameters are generated randomly, we got different results from the above figure. We recorded the best simulation which yield SSE = 0.14347. Here is the best simulation for initially random parameters during validation process.

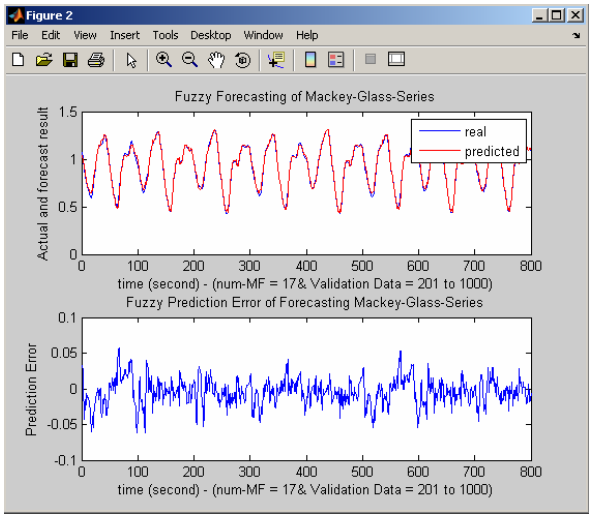


Figure 3. The best simulation for initially random parameters during validation process.

The next experiment is using LSE to optimize the rule consequent parameters θ. First we modify the equation for θ with small adaptation step μ to avoid singularities problem during matrix calculation so it became:

$$[\theta] = ([\gamma XIe]^T \cdot [\gamma XIe] + \mu I)^{-1} \cdot [\gamma XIe]^T \cdot [f] \qquad (14)$$

Here is the optimized result of the Gaussian membership function after being updated in 200 iterations (recall that we use first 200 data of XIO matrix for training).

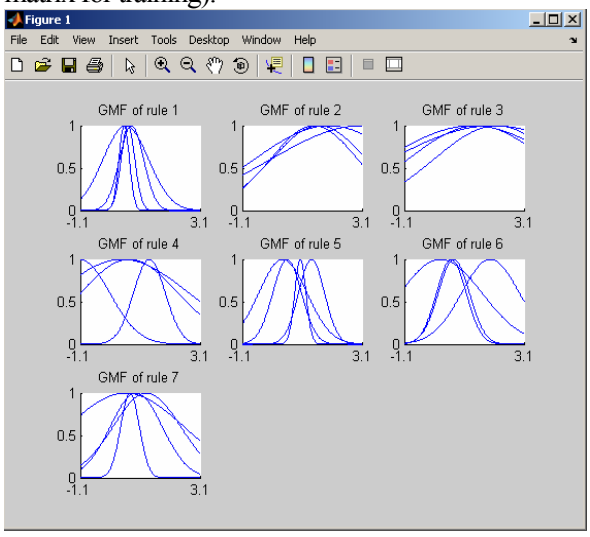


Figure 4. The optimized result of the Gaussian membership function after being updated in 200 iterations

The implementation of LSE method for rule consequent parameters reduces the value of SSE down to 0.0206 which is much smaller than the previous SSE. It means that the performance of neuro-fuzzy system is increased by factor up to 700%. Simulation during validation process yields the following result.

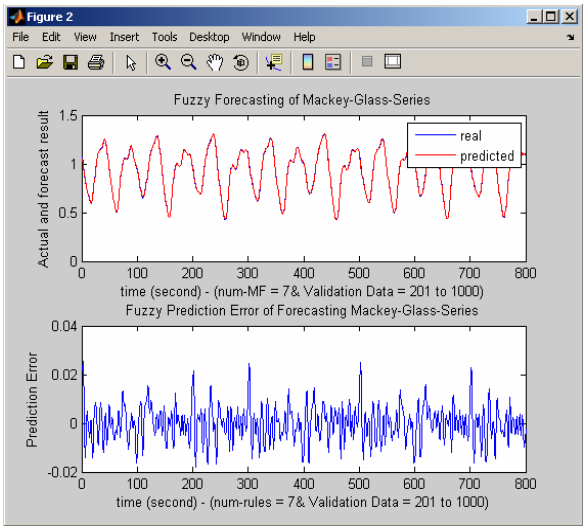


Figure 5. Simulation during validation process with LSE method.

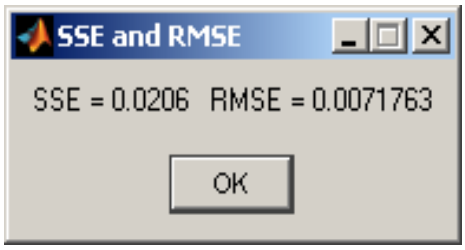


Figure 6. Final calculation of SSE which reflects the improvement from the previous result.

DISCUSSION AND CONCLUSION

After developing and testing program in Matlab, we conclude two important remarks here:

a. From the simulation, we can see that LSE method can enhance the performance of Takagi-Sugeno neuro-fuzzy for time series forecasting although it requires more computational power. Although GMFs aren't equally distributed along the universe of discourse, the system gives satisfactory prediction of Mackey-Glass Data.
b. The value of η for updating the parameters should be choosen properly since this value is sensitive. Too high value of this parameter makes the program tends to oscillating. This computation involves huge number of data. One must find optimization procedure to simplify the program or accelerate the calculation process.

Comparing to the previous work as described in [8], this paper certainly confirm that the same performance can only be achieved with the high number of rules which in turn will require much efforts to reduce redundancies and conflicting rules.

REFERENCES

[1] E. Mizutani, J.-S. Roger Jang, "Coactive Neural Fuzzy Modelling", Proceeding of the International Conference on Neural Network: 1995.

[2] Min-You Chen, D.A. Linkens, "Rule-base self-generation and simplification for data-driven fuzzy models", Fuzzy Set and System, 2003.

[3] Hossein Salehfar, Nagy Bengiamin, Jun Huang, "A Systematic Approach To Linguistic Fuzzy Modeling Based On Input-Output Data", Proceedings of the 2000 Winter Simulation Conference: 2000.

[4] Palit A. K, Popovic D., "Forecasting Chaotic Time Series using Neuro-Fuzzy Approach", Proceeding of IEEE-IJCNN: 1999.

[5] Palit A. K., Popovic D., "Computational Intelligence in Time Series Forecasting: Theory and Engineering Application", Springer-Verlag London: 2005.

[6] J.-S. Roger Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference Systems". IEEE Transaction on Systems, Man and Cybernetics: 1993.

[7] MATLAB, "The Language of Technical Computing", User's Guide and Programming, The MathWorks Inc: 1998.

[8] Felix Pasila, Indar Sugiarto, "Mamdani Model For Automatic Rule Generation of A MISO System", Proceedings of Soft Computing, Intelligent Systems and Information Technology (SIIT-2005): 2005.