Automatic Generation Fuzzy Neural Network Controller with Supervisory Control for Permanent Magnet Linear Synchronous Motor

Hung-Ching Lu
Department of Electrical Engineering
Tatung University
Taipei, Taiwan
luhung@ttu.edu.tw

Ming-Hung Chang
Department of Electrical Engineering
Tatung University
Taipei, Taiwan
minghung21@yahoo.com.tw

Abstract—The automatic generation fuzzy neural network (AGFNN) controller with supervisory control for permanent magnet linear synchronous motor (PMLSM) is proposed in this paper. It comprises an AGFNN controller, which has ability of rule automatic generation with on-line learning and a supervisory controller, which is designed to stabilize the system states around a bounded region. The Mahalanobis distance (M-distance) formula is employed that the neural network has the ability of identification of the rules will be generated or not. To improve the learning speed of back-propagation algorithm in AGFNN controller, a switching law and a momentum term are used in this study. The design of supervisory controller is derived in the Lyapunov sense; thus, the stability of the control system can be guaranteed. Finally, simulation results show that the proposed controller is robust with regard to plant parameter variations and external load disturbance.

Index Terms—Fuzzy neural network, back-propagation algorithm, switching law, momentum term

I. INTRODUCTION

The fuzzy neural network (FNN) possesses both advantages of fuzzy logic process and neural network. Structure and parameter phase learning algorithms not only extract the fuzzy logic rule from input data and adjust the fuzzy partitions of the input and output spaces but also adjust the parameters of the FNN. Generally speaking, these two phases are done sequentially [1]; that is, the structure learning phase is employed to decide the structure of fuzzy rules first. Then the parameter learning phase is used to tune the coefficients of each rule (for example, membership functions). The disadvantages of this sequential learning scheme are that it is suitable only for offline instead of online operation [1] and a large amount of representative data must be collected in advance for the implementation of this scheme. Moreover, the independent realization of the structure and parameter learning is too times consuming. To overcome this problems and to achieve accelerated learning, a self-constructing neural fuzzy inference network (SONFIN) was proposed in [2] to perform the structure and parameter learning phases concurrently. This kind of research is focus on how to generate an optimal number of neurons or fuzzy rules in FNN and investigate automated methods of adding and pruning neurons. However, the network structure and parameter learning algorithms proposed in [2] are both complicated and not suitable for practical implementation. To overcome this difficulty, a self-constructing fuzzy neural network (SCFNN) is proposed in [3]. But the similarity checking of this learning scheme was performed on all input variables. It will be result in the generation of unnecessary neurons and large transient error since huge external load disturbance happened. The performance of the control system would be deteriorated since redundant neurons were generated. By this reason, the learning scheme increases the complexity of the algorithm significantly and it is not a practical realization. In this paper, Mahalanobis distance (M-distance) method in [4] takes into account the distance of the mean and standard deviation to identify the rules will be generated or not. Then the controller can directly and immediately generate a suitable number of neurons without pruning neurons.

The learning speed of the traditional back-propagation algorithm is too slow to cope with transient state responses due to its weight is adjusted too slowly for high speed tracking, and this problem may cause system oscillation. A momentum term [5] is employed in this paper to increase the learning speed and mitigate the oscillation in FNN system. The general error term cannot be determined due to the uncertainties of the plant dynamic such as parameter variations and external load disturbance. In this paper, a switching law in [6] is employed to obtain a suitable error term that the transient state error is relieved.

From the viewpoints of the above discussion, the automatic generation fuzzy neural network (AGFNN) controller with supervisory control for PMLSM is proposed in this paper. The supervisory controller is designed to compensate the AGFNN controller. The AGFNN controller is the main tracking controller. The supervisory controller is designed so that the states of the control system are stabilized around a predetermined bound region. If the nonlinear system tends to be unstable by the AGFNN controller, especially in the transient period, the supervisory controller will be activated to work with the AGFNN controller to stabilize the whole system. On the other hand, if the AGFNN controller works well that the system stable, the supervisory controller will be deactivated.
The AGFNN controller generates the rules which are desired by using M-distance method. A momentum term and a switching law are used to improve the learning speed of back-propagation algorithm in FNN.

II. CONTROL OF THE ROTOR POSITION OF THE PMLSM DRIVE SYSTEM

PMLSM is a nonlinear system and its mathematical model is difficult to derive completely. The mechanical equation of a PMLSM drive in [7] is represented as

\[ J\ddot{\theta}(t)+B\dot{\theta}(t)+T_f+f(\dot{\theta})=T_e, \]  

(1)

where \( J \) is the moment of inertia, \( B \) is the damping coefficient, \( \theta \) is the position, \( T_f \) represents the external load disturbance, \( f(\dot{\theta}) \) is the friction force, and \( T_e \) denotes the electric torque. The friction force can be formulated as follows:

\[ f(\dot{\theta}) = F_C \text{sgn}(\dot{\theta}) + (F_S - F_C) \exp\left(-\frac{\dot{\theta}}{\dot{\theta}_s}\right) \cdot \text{sgn}(\dot{\theta}) + K_\theta \dot{\theta}, \]  

(2)

where \( F_C \) is the Coulomb friction; \( F_S \) is the static friction; \( \dot{\theta}_s \) is the Stribeck-velocity parameter; \( K_\theta \) is the coefficient of viscous friction; \( \text{sgn}(\cdot) \) is a sign function. All the parameters of \( f(\dot{\theta}) \) are time varying. With the implementation of field-oriented control, the electric torque can be simplified as

\[ T_e = K_i i^*_q, \]  

(3)

\[ K_i = \frac{3n_p^2 L_s}{2J}, \]  

(4)

where \( K_i \) is the torque constant, \( i^*_q \) is the torque current command, \( L_s \) is the flux current command, \( n_p \) is the number of pole pairs, \( L_m \) is the magnetizing inductance per phase, and \( L_r \) is the rotor inductance per phase. The PMLSM drive system from (1) can be represented in the following state-space form:

\[
\begin{bmatrix}
\dot{\theta}(t) \\
\ddot{\theta}(t)
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & B/J
\end{bmatrix}
\begin{bmatrix}
\theta(t) \\
\dot{\theta}(t)
\end{bmatrix} +
\begin{bmatrix}
0 \\
1/J
\end{bmatrix}(-f(\dot{\theta}) + K_i u - T_f),
\]  

(5)

and \( u = i^*_q \) is the control effort. For compactness, (5) is rewritten as

\[
\dot{\theta} = A\theta + B(-f(\dot{\theta}) + K_i u - T_f), \quad \text{where} \quad A = \begin{bmatrix} 0 & 1 \\ 0 & -B/J \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1/J \end{bmatrix}. \]  

(6)

By computation, the dynamic equation can be denoted as

\[
\dot{\theta}_d = A\theta_d + B(J\ddot{\theta}_d + B\dot{\theta}_d). \]  

(7)

Now, define the tracking error vector as follows

\[
E = [\theta_e - \theta_d - \dot{\theta}_d - \dot{\theta}_e]^T = [\theta_e \quad \dot{\theta}_e]^T, \]  

(8)

and it is desired to construct a controller effort \( U \) such that \( \lim_{t \to \infty} E(t) = 0 \). Next, the tracking error dynamics are analyzed. From (6) and (7), we have

\[
\dot{E} = (A - BK)E + B(KE + J\ddot{\theta}_d + B\dot{\theta}_d + f(\dot{\theta}) - K_i u + T_f), \]  

(9)

and \( K \) is chosen so that the characteristic polynomial of \( A - BK \) whose roots lie strictly in the open left half of the complex plane, then it implies that \( A - BK \) is asymptotically stable. If the parameters of the PMLSM drive system, the friction force function, and \( T_f \) are well known, the computational control law will be defined as

\[ \dot{i}_q = u^* = \frac{1}{K_i} [f(\dot{\theta}) + J\ddot{\theta}_d + B\dot{\theta}_d + KE], \]  

(10)

and (9) can be rewritten as

\[ \dot{E} = (A - BK)E + B[K_i (u^* - u) + T_f]. \]  

(11)

The block diagram of the whole control system in this paper is shown in Fig.1. The control law is assumed to take the following form:

\[ u = u_{FNN} + u_s, \]  

(12)

where \( u_{FNN} \) is the AGFNN control effort and \( u_s \) is the supervisory control effort, respectively.

III. AGFNN CONTROLLER WITH SUPERVISORY CONTROL

A. Structure of the AGFNN

A five-layer AGFNN is comprised by the input, membership, rule, normalization, and output layer, is adopted to implement the AGFNN controller in this section.

Layer 1: Each node in this layer is an input node, which corresponds to input variables.

\[ x_i = \theta_i \quad \text{and} \quad x_s = \dot{\theta}_s. \]  

(13)

Layer 2: The Gaussian function is adopted as the membership function. For the \( j \) th node

\[ u_{ij} = \exp\left(-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right), \quad i = 1, 2, j = 1, \ldots, M(k), \]  

(14)

where \( c_{ij} \) and \( \sigma_{ij} \) are, respectively, the mean and the standard deviation of the Gaussian function in the \( j \) th term of the \( i \) th group which input is the linguistic variable \( x_i \) to the node of layer 2, and \( M(k) \) is the total number of the rule in each groups at time \( t \) which is generated by the control system.

Layer 3: Each node in this layer is denoted by \( \Pi \), which is used to compute each rule’s firing strength and the output is given by
\( \Phi_j(x_i) = \exp \left[ -\frac{1}{2} \sum_{j=1}^{2} \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right], \quad i = 1, 2, \ j = 1, \ldots, M(k). \) (15)

**Layer 4:** The data is normalized in this layer which can be denoted as

\[
\Phi_j = \frac{\Phi_j}{\sum_{j=1}^{M(k)} \Phi_j}.
\] (16)

**Layer 5:** This layer acts as a defuzzifier. The single node in this layer is labeled \( \Sigma \) and it sums all incoming signals to obtain the final inferred result

\[
u_{\text{FNN}} = \sum_{j=1}^{M(k)} \Phi_j w_j.
\] (17)

**B. Structure Learning Phase**

From the viewpoint of fuzzy logic control, a fuzzy rule is a local representation over a defined region. If a new input falls within the local region, the AGFNN will not generate a new rule but adjust the parameters of existing rules. The firing strength of each rule shown in (15) can be regarded as a function of regularized Mahalanobis distance (M-distance), i.e.

\[
\Phi_j = \exp(-md(j)^2) = \exp \left[ -\frac{1}{2} \sum_{j=1}^{2} \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right],
\] (18)

where

\[
md(j) = \sqrt{\frac{1}{\sigma_{ij}^2} (x_i - c_{ij})^2 + \frac{1}{\sigma_{ij}^2} (x_i - c_{ij})^2},
\] (19)

\[
\sigma_j = \begin{bmatrix}
\frac{1}{\sigma_{ij}^2} & 0 & \cdots & 0 \\
0 & \frac{1}{\sigma_{ij}^2} & \cdots & 0 \\
\vdots & 0 & \ddots & 0 \\
0 & 0 & \cdots & \frac{1}{\sigma_{ij}^2}
\end{bmatrix}, \quad j = 1, 2, \ldots, M(k).
\] (20)

Substitute Eq.(20) into Eq.(19) and consider the case that \( i = 1, 2 \); then

\[
md(j) = \sqrt{\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} + \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}.
\] (21)

Next, find

\[
MJ = \min_{i \in 1, 2, \ldots, M(k)} (md(j))
\] (22)

\[
MJ > k_d,
\] (23)

where \( k_d \) is a preset positive constant. Since the capability of learning is decay in AGFNN, the existing rules are not sufficient such that a new rule should be considered. If (23) is satisfied, a new rule will be generated. On the other hand, the generation of rules is influenced by the variation of \( k_d \) seriously. Once a new rule is generated, the next step is to assign initial mean \( c_{i}^{\text{new}} \) and the standard deviation \( \sigma_i^{\text{new}} \) of the corresponding membership function that are selected as

\[
c_{i}^{\text{new}} = x_i,
\] (24)

and \( \sigma_i^{\text{new}} \) is preset constant. Then, the new membership function is adopted and the number \( M(k) \) is incremented as

\[
M(k + 1) = M(k) + 1.
\] (25)

**C. Parameter Learning Phase with Switching Law**

This method is generally referred to as the back-propagation learning method. To describe the online parameter learning algorithm of the AGFNN by using the supervised gradient descent method, first the cost function \( P_c \) is defined as

\[
P_c = \frac{1}{2} (\theta_d - \theta)^2 = \frac{1}{2} \theta^2,
\] (26)

where \( \theta_d \) denotes the output error between the desired trajectory \( \theta_d \) and the rotor position \( \theta \) of the PMLSM. The parameter learning algorithm based on back-propagation is described in the following. In layer 5, the error term to be propagated is computed as

\[
\delta^P = -\frac{\partial P_c}{\partial u_{\text{FNN}}} = \begin{bmatrix}
-\frac{\partial P_c}{\partial \theta_d} & -\frac{\partial P_c}{\partial \theta} \\
-\frac{\partial P_c}{\partial \theta_d} & -\frac{\partial P_c}{\partial \theta} & -\frac{\partial P_c}{\partial \theta}
\end{bmatrix}.
\] (27)

Since the increasing of error is not due to the parameter learning in AGFNN, the error term from (27) is not suitable for the system. The exact calculation of the Jacobian of the system \( \partial \theta / \partial u_{\text{FNN}} \) which is contained in \( \partial E / \partial u_{\text{FNN}} \) cannot be determined due to the uncertainties of the plant dynamic such as parameter variations and external disturbances [3]. To overcome this problem and to increase the online learning speed of the network parameters, a suitable error term in layer 5 is selected by the switching law [6] and proposed in this section. Firstly, a switching condition is defined and it is denoted as

\[
H(s) = ss\ ,
\] (28)

where \( s = C\theta_d + \dot{\theta}_d \) and \( C \) is a positive constant.

The switching regulator is designed in this paper to judge whether the switching condition \( H(s) < 0 \) is satisfied. If the condition is not satisfied, the switching regulator is back-propagated to the AGFNN, and then \( \delta^P \) in (27) is taken to tune the parameters of the AGFNN. It is seen that the inputs of the switching regulator are \( s \) and \( \dot{s} \). The switching condition \( H(s) < 0 \) will be achieved as the following statements.

Case 1: \( H(s) \geq 0 \). It is obvious that the switching condition is not satisfied. Therefore, the switching regulator is designed as
\[ s_r(s) = s. \] (29)

Case 2: \( H(s) < 0 \). It means that the switching condition is satisfied and the switching regulator can be selected as
\[ s_r(s) = \beta \cdot s, \] (30)
where \( \beta > 0 \) is an arbitrary small number.

For the purpose of tuning the parameter of AGFNN, the index function \( E_s \) of the switching regulator is defined as
\[ E_s = \frac{1}{2} (s_r(s))^2 \] (31)
where \( E_s \) is the other kind of cost function which we chose. It is found that the increasing of error is not caused by the output of the control system \( u_{FNN} \) but caused by the uncertainties of the plant dynamic. Therefore, the new calculation of the Jacobian \( \frac{\partial E_s}{\partial \theta_e} \) is used to instead of \( \frac{\partial P}{\partial u_{FNN}} \) and the suitable error term is obtained. Next, there are also two circumstances must be considered.

Case 1: If \( H(s) \geq 0 \), the error term can be obtained as follows:
\[ \delta^s = -\frac{\partial E_s}{\partial \theta_e} = -\frac{\partial E_s}{\partial s_r} \frac{\partial s_r}{\partial s} \frac{\partial s}{\partial \theta_e} = C \cdot s. \] (32)

Case 2: If \( H(s) < 0 \), the error term can be obtained as follows:
\[ \delta^s = -\frac{\partial E_s}{\partial \theta_e} = -\frac{\partial E_s}{\partial s_r} \frac{\partial s_r}{\partial s} \frac{\partial s}{\partial \theta_e} = C \cdot \beta^2 \cdot s. \] (33)

D. Parameter Learning Law

The error term of the layer 5 is obtained from (32) and (33). The link weight is updated by the amount
\[ \Delta w_j = -\eta \frac{\partial P}{\partial u_y} \left[ \frac{\partial P}{\partial \theta_j} \frac{\partial \theta_j}{\partial u_{FNN}} \left[ \frac{\partial u_{FNN}}{\partial w_j} \right] \right] = \eta \delta^s \Phi_j, \] (34)
where \( \eta \) is the learning rate of the link weight. The inclusion of a momentum term has been found to increase the rate of convergence dramatically [5]. With this method, (34) will take the form:
\[ \Delta w_j(N) = \eta \delta^s \Phi_j + \alpha \cdot \Delta w_j(N-1), \] (35)
where \( \alpha \) is the momentum parameter. The link weights in layer 5 are updated according to the following equation:
\[ w_j(N+1) = w_j(N) + \Delta w_j, \] (36)
where \( N \) denotes the iteration number of the \( j \)th link.

Layer 4: In this layer only the error term needs to be calculated and propagated.

\[ \delta^l = -\frac{\partial P}{\partial \theta_j} = \frac{\partial P}{\partial u_{FNN}} \left[ \frac{\partial u_{FNN}}{\partial \Phi_j} \frac{\partial \Phi_j}{\partial \theta_j} \right] = \delta^l \Phi_j. \] (37)

Layer 3: In this layer only the error term needs to be calculated and propagated.

\[ \delta^s = -\frac{\partial P}{\partial u_y} = \delta^s \left[ \frac{1}{\sum_{i=1}^{m} \Phi_i} - \frac{\Phi_j}{(\sum_{i=1}^{m} \Phi_i)^2} \right]. \] (38)

Layer 2: The error term is computed as:
\[ \delta^s = -\frac{\partial P}{\partial u_y} = \delta^s \left[ \frac{1}{\sum_{i=1}^{m} \Phi_i} - \frac{\Phi_j}{(\sum_{i=1}^{m} \Phi_i)^2} \right]. \] (39)

At the same time, the update law of \( c_y \) and \( \sigma_y \) are obtained as
\[ \Delta c_y = -\eta \frac{\partial P}{\partial c_y} = \eta \left[ \frac{\partial P}{\partial u_y} \left[ \frac{\partial u_y}{\partial c_y} \right] \right] = \eta \delta^s \left[ \frac{2(u_y - c_y)^2}{\sigma_y^2} \right], \] (40)
\[ \Delta \sigma_y = -\eta \frac{\partial P}{\partial \sigma_y} = \eta \left[ \frac{\partial P}{\partial u_y} \left[ \frac{\partial u_y}{\partial \sigma_y} \right] \right] = \eta \delta^s \left[ \frac{2(u_y - c_y)^2}{\sigma_y^2} \right], \] (41)
where \( \eta \) are the learning rate of the mean and the standard deviation, respectively. The mean and the standard deviation of the membership functions in this layer will be updated as
\[ c_y(N+1) = c_y(N) + \Delta c_y, \] (42)
\[ \sigma_y(N+1) = \sigma_y(N) + \Delta \sigma_y. \] (43)

E. The Design of Supervisory Controller

Generally speaking, the system is stable since the above AGFNN ids work well. But the update laws shown in (36), (42), and (43) can not be guaranteed the stability of the whole control system. To overcome this problem, the design of supervisory controller is used to stabilize the whole control system. First, the Lyapunov function is defined as
\[ V = \frac{1}{2} E^T P E, \] (44)
where \( P \) is a symmetric positive definite matrix which satisfies the following Lyapunov equation:
\[ A_n^T P + P A_n = -Q, \] (45)
and \( Q \) is positive definite and selected by the designer.

Take the derivative of the Lyapunov function and use (11), (12), and (45), then
\[ \dot{V} = E^T P E \]
\[ = E^T P \left\{ (A-BK)E + B \left[ K, (u'-u)+T_e \right] \right\} \]
\[ = \frac{1}{2} E^T Q E + E^T P B \left[ K, (u'-u_{f_{NN}}-u)+T_e \right] \]
\[ \leq \frac{1}{2} E^T Q E + E^T P B \left[ K, (\|u\|_2+\|u_{f_{NN}}\|+\|T_e\|) - E^T P B K, u. \right] \]

To satisfy \( \dot{V} \leq 0 \), the supervisory control effort \( u_s \) has to be designed as
\[
u_s = I \cdot \text{sgn}(E^T P B) \left[ \nu_{f_{NN}} + \frac{1}{K} (f'(\theta) + \theta) + \nu_{f_{NN}}^2 + \|E\| + \|T_e\|_2 \right], \tag{47} \]
where
\[
I = \begin{cases} 
1, & \text{if } V \geq F \\
0, & \text{if } V < F,
\end{cases} \tag{48}
\]

\( K \) is a positive constant, \( f'(\theta) < f''(\theta) < \infty \), external disturbance \( |T_e| \leq T_{e_{max}} \), and \( F \) is a positive constant. In order to let \( \|u_{f_{NN}}\| \) be bounded, some conditions are considered to design as
\[
|w_j| \leq \bar{w}_j < \infty, \tag{49}
\]
where \( \bar{w}_j \) is the upper bound of \( |w_j| \). Because the output value of layer 4 is between 0-1, the link weight is bounded by (51), and the number of neurons in layer 4 is finite, then \( \|u_{f_{NN}}\| \) is bounded can be obtained. Further, supervisory control effort \( u_s \) is also bounded due to all of elements from supervisory control effort is bounded.

Substitute (10) and (47) into (46) and consider the case that \( |\theta| \) is bounded, some conditions are considered to design as
\[
|w_j| \leq \bar{w}_j < \infty, \tag{49}
\]
where \( \bar{w}_j \) is the upper bound of \( |w_j| \). Because the output value of layer 4 is between 0-1, the link weight is bounded by (51), and the number of neurons in layer 4 is finite, then \( \|u_{f_{NN}}\| \) is bounded can be obtained. Further, supervisory control effort \( u_s \) is also bounded due to all of elements from supervisory control effort is bounded.

Substitute (10) and (47) into (46) and consider the case that
\[
\dot{V} = E^T P E \]
\[ = \frac{1}{2} E^T Q E + E^T P B \left[ K, (u'-u_{f_{NN}}-u)+T_e \right] \]
\[ \leq \frac{1}{2} E^T Q E + E^T P B \left[ K, (\|u\|_2+\|u_{f_{NN}}\|+\|T_e\|) - E^T P B K, u. \right] \]

To show that \( \lim_{t \to \infty} E(t) = 0 \), the following term is defined.
\[
P_s(t) = \frac{1}{2} E^T Q E, \tag{51}
\]
and integrate (50) to obtain
\[
\int_0^t P_s(\tau)d\tau \leq V(0)-V(t) \leq V(0) < \infty, \tag{52}
\]

Also, because \( \dot{P_s}(t) \) is bounded, it can be shown that \( \lim_{t \to \infty} P_s(t) = 0 \) by Barbalat’s Lemma [8]. That is \( E(t) \to 0 \) as \( t \to \infty \), then the stability of the control system can be guaranteed. Using the designed supervisory control effort \( u_s \) as shown in (47), the inequality \( \dot{V} < 0 \) can be obtained for nonzero value of the tracking error vector \( E \) when \( V > F \).

The stability of the control system can be guaranteed by the design of supervisory controller in Lyapunov stability theorem. If the nonlinear system tends to be unstable by the AGFNN controller, especially in the transient period, the supervisory controller will be activated to work with the AGFNN controller to stabilize the whole system. On the other hand, if the AGFNN controller works well that the system is stable; the supervisory controller will be deactivated. Then the stability of the control system can be guaranteed by the design of supervisory controller in any kind of operated environment.

IV. SIMULATION RESULTS

The detailed parameters of the drive systems are given as: \( \bar{F} = 20N/A, J = 0.1254Ns/V, B = 5.2892N/V, \) where the over bar symbol represents the system parameter in the nominal condition. The coefficients of the friction model are selected as \( F_c = 0.08, F_s = 1.2, \theta_s = 0.08, K_{\theta} = 0.8. \)

To investigate the effectiveness of the proposed control algorithm, three simulated cases including parameters variations and external disturbance are considered.

Case 1: \( J = \bar{J}, T_e = 0 N, \) during \( t = 0 - 5 \) second. \( \tag{53} \)

Case 2: \( J = 10 \bar{J}, T_e = 0 N, \) after \( t = 5 \) second. \( \tag{54} \)

Case 3: \( J = 10 \bar{J}, T_e = 50 N, \) after \( t = 10 \) second. \( \tag{55} \)

To demonstrate the performance of the control system with different reference trajectories, simulation results owing to three kinds of periodic commands are given. The control objective is to control the rotor position \( \theta \) of the PMLSM between the range \( \pm 5 \) mm periodically.

The simulation results of the proposed controller are shown in Figs. 2 to 3. For the simulated results, the proposed controller works well in all presented conditions and only extra generates 3 rules as case 3 is happened with the external load disturbance. Although the periodic triangular command is non-smooth in Fig. 3, the presented scheme only generates 5 rules for neurons and has the better performance.

V. CONCLUSIONS

In this paper, AGFNN controller with supervisory control for PMLSM is developed successfully. The proposed controller completely combines the advantages of AGFNN and supervisory controller. First, M-distance is used in AGFNN structure learning phase that AGFNN system can generate suitable number of rules. Second, the similarity checking is only for generating rules automatically in AGFNN structure learning phase. Next, the learning speed of AGFNN parameter phase is reformed by the momentum and the switching law prominently. Finally, the stability of the whole control system is guaranteed by the supervisory controller. The simulation results show that the proposed controller is robust with regard to plant parameter variations and external load disturbance.
Figure 1. Block diagram of the whole control system.

Figure 2. Simulation results of the proposed controller for periodic sinusoidal command: (a) the tracking response, (b) control effort, (c) number of rule, (d) tracking error, and (e) supervisory control effort.

Figure 3. Simulation results of the proposed controller for periodic triangular command: (a) the tracking response, (b) control effort, (c) number of rule, (d) tracking error, and (e) supervisory control effort.

ACKNOWLEDGMENT

This work was supported by the National Science Council, Republic of China, under Grant NSC 97-2221-E-036-025.

REFERENCES