Nonlinear System Control Using Functional-Link-Based Neuro-Fuzzy Network Model Embedded with Modified Particle Swarm Optimizer

Miin-Tsair Su, Chin-Teng Lin, Sheng-Chih Hsu, Dong-Lin Li, Cheng-Jian Lin, and Cheng-Hung Chen

Abstract

This study presents an evolutionary neural fuzzy system (NFS) for nonlinear system control. The proposed NFS model uses functional link neural networks (FLNNs) as the consequent part of the fuzzy rules. This study uses orthogonal polynomials and linearly independent functions in a functional expansion of the functional link neural networks. A learning algorithm, which consists of structure learning and parameter learning, is presented. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on the particle swarm optimization (PSO) algorithm, can adjust the shape of the membership function and the corresponding weighting of the FLNN. The distance-based mutation operator, which strongly encourages a global search giving the particles more chance of converging to the global optimum, is introduced. The simulation results have shown the proposed method can improve the searching ability and is very suitable for the nonlinear system control applications.

Keywords: Functional link neural networks (FLNNs), mutation operator, neuro-fuzzy networks (NFNs), particle swarm optimization (PSO), perturbation operator.

1. Introduction

Nonlinear system control is an important tool that is adopted to improve control performance and achieve robust fault-tolerant behavior. Among nonlinear control techniques, those based on artificial neural networks and fuzzy systems have become popular topics of research in recent years [1]-[3] because classical control theory usually requires a mathematical model to design the controller. However, the inaccuracy of the mathematical modeling of plants usually degrades the performance of the controller, especially for nonlinear and complex control problems [4]. On the contrary, both the fuzzy system controller and the artificial neural network controller provide key advantages over traditional adaptive control systems. Although traditional neural networks can learn from data and feedback, the meaning associated with each neuron and each weight in the network is not easily interpreted. Alternatively, the fuzzy logical models are easily appreciated, because they use linguistic terms and the structure of IF–THEN rules. However, fuzzy systems have a lack of an effective learning algorithm to refine the membership functions to minimize output errors. According to the literature review mentioned before, it can be said that, in contrast to pure neural or fuzzy methods, the hybrid neuro-fuzzy networks (NFNs) [5]-[11] possess the advantages of both neural networks and fuzzy systems. NFNs bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Training of the parameters is the main problem in designing a neural fuzzy system. Backpropagation (BP) [11], [12] training is commonly adopted to solve this problem. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent approach is used in BP training to minimize the error function, the algorithms may reach the local minima very quickly and never find the global solution. The aforementioned disadvantages lead to suboptimal performance, even for a favorable neural fuzzy system topology. Therefore, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure are required.

Accordingly, a new optimization algorithm, called particle swarm optimization (PSO), appears to be better than the backpropagation algorithm. PSO is a stochastic optimization technique developed by Kennedy and Eberhart in 1995 [13], [14], inspired by the social behavior of bird flocking or fish schooling. Bird flocking has some underlying rules that enables large numbers of birds to flock synchronously, often changing direction suddenly. PSO solves optimization problems by modeling the underlying rules. To simulate social behavior, bird flocking searches for food in an area. Each bird flies according to self-cognition and social information. Self-cognition is the generalization produced by past
experience. The social information is the message that is shared by the society. The strategy of the birds is to maintain the good experiences by referring to the knowledge of the others. The two main models of the PSO algorithm, called $g_{best}$ (global best) and $l_{best}$ (local best), which differ in the way they define particle neighborhood. Kennedy and Poli [27], [28] showed that the $g_{best}$ model has a high convergence speed with a higher chance of getting stuck in local optima. On the contrary, the $l_{best}$ model is less likely become trapped in local optima but has a slower convergence speed than $g_{best}$.

Similar to genetic algorithms (GAs) [25], [26], PSO is a population based optimization tool. The system is initialized with a population of random solutions and searches for optima by updating potential solutions after each iteration. However, unlike GA, PSO has no evolution operators such as crossover and mutation. Compared to GA, PSO is more efficient, with fewer parameters, and easy to implement. During the past several years, PSO has been successfully applied to a diverse set of optimization problems, such as multidimensional optimization problems [21], multi-objective optimization problems [22], [24], classification problems [6], [15], [16] and artificial neural network training [12], [17], [23].

This study presents a PSO-based learning algorithm for the neural fuzzy system (NFS) in nonlinear system control applications. The proposed NFS model is based on our previous research [18]. The NFS model, which combines a neural fuzzy system with a functional link neural network (FLNN) [10], [19], [20], is designed to improve the accuracy of functional approximation. The consequent part of the fuzzy rules that corresponds to an FLNN comprises the functional expansion of input variables. The orthogonal polynomials and linearly independent functions are adopted as functional link neural network bases. PSO is an efficient tool for optimization and search problems. However, it is easy to become trapped in local optima due to its information sharing mechanism. Many research works have shown that mutation operators can help PSO prevent premature convergence [30]-[32]. To prevent basic PSO from becoming trapped in local optima, we modified the basic PSO by adding a diversity scheme, called the distance-based mutation operator, which strongly encourages a global search giving the particles more chance of converging to the global optimum.

The idea behind the proposed DMPSO algorithm is that there are only two kinds of convergence, local optimum convergence and global optimum convergence. If local optimum convergence occurred, meaning that the basic PSO is trapped in a local optimum, this is a good time to apply the mutation operator to help the PSO to escape from the local optimum. If global optimum convergence occurred, applying the mutation operator will cause the PSO to naturally converge again at the global optimum.

This paper is organized as follows. Section 2 describes the structure of the suggested model. Section 3 presents the proposed learning algorithms for the FLNFFN model. Next, Section 4 presents the results of simulations of nonlinear system control problems. Finally, Section 5 draws conclusions.

### 2. Structure of Functional-Linked-Based Neuro-Fuzzy Network

This section describes the structure of functional-linked-based neuro-fuzzy network (FLNFN) model, which uses a nonlinear combination of input variables. The FLNFN is based on our previous research [18]. Each fuzzy rule corresponds to a sub-FLNN [10], [19], [20] comprising a functional expansion of input variables. The functional link neural network (FLNN) is a single layer network in which the need for hidden layers is removed. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions (i.e., the use of suitable orthogonal polynomials for a functional expansion) and then evaluating these functions with the variables as the arguments. Figure 1 presents the structure of the proposed FLNFN model.

The FLNFN model realizes a fuzzy IF-THEN rule in the following form.
Rule \( j \):

IF \( \hat{x}_i \) is \( A_{i,j} \) and \( \hat{x}_2 \) is \( A_{2,j} \) and ... and \( \hat{x}_n \) is \( A_{n,j} \)

and \( \hat{x}_N \) is \( A_{N,j} \)

THEN \( \hat{y}_j = \sum_{i=1}^{N} w_{ij} \phi_k = w_{ij1} \phi_{k1} + w_{ij2} \phi_{k2} + \ldots + w_{ijM} \phi_{kM} \)

where \( \hat{x}_i \) and \( \hat{y}_j \) are the input and local output variables, respectively; \( A_{i,j} \) is the linguistic term of the pre-condition part with a Gaussian membership function; \( N \) is the number of input variables; \( w_{ij} \) is the link weight of the local output; \( \phi_k \) is the basis trigonometric function of input variables; \( M \) is the number of basic functions, and Rule \( j \) is the \( j^{th} \) fuzzy rule.

The operation functions of the nodes in each layer of the FLNFN model are now described. In the following description, \( u^{(i)} \) denotes the output of a node in the \( i^{th} \) layer.

Layer 1 (Input node): No computation is performed in this layer. Each node in this layer is an input node, which corresponds to one input variable, and only transmits input values to the next layer directly

\[ u^{(1)} = \hat{x}_i \]  \hspace{1cm} (2)

Layer 2 (Membership function node): Nodes in this layer correspond to a single linguistic label of input variables in layer 1. Therefore, the calculated membership value specifies the degree to which an input value belongs to a fuzzy set in layer 2. The implemented Gaussian membership function in layer 2 is

\[ u^{(2)} = \exp \left( -\frac{[u^{(1)} - m_y]^2}{\sigma_y^2} \right) \]  \hspace{1cm} (3)

where \( m_y \) and \( \sigma_y \) are the mean and variance of the Gaussian membership function, respectively, of the \( j^{th} \) term of the \( i^{th} \) input variable \( \hat{x}_i \).

Layer 3 (Rule Node): Nodes in this layer represent the pre-condition part of a fuzzy logic rule. They receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described above is adopted to perform the IF-condition matching of the fuzzy rules. As a result, the output function of each inference node is

\[ u^{(3)} = \prod_{i} u^{(2)} \]  \hspace{1cm} (4)

where the \( \prod_{i} u^{(2)} \) of a rule node represents the firing strength of its corresponding rule.

Layer 4 (Consequent Node): Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are nonlinear combinations of input variables from the FLNN. For such a node,

\[ u^{(4)} = u^{(3)} \sum_{i=1}^{M} w_{ij} \phi_k \]  \hspace{1cm} (5)

where \( w_{ij} \) is the corresponding link weight of the functional link neural network (FLNN) and \( \phi_k \) is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by \( [\hat{x}_1, \sin(\pi \hat{x}_1), \cos(\pi \hat{x}_1), \hat{x}_2, \sin(\pi \hat{x}_2), \cos(\pi \hat{x}_2)] \) for two-dimensional input variables. Therefore, \( M \) is the number of basic functions, \( M = 3 \cdot N \), where \( N \) is the number of input variables. Moreover, the output nodes of the FLNN depend on the number of fuzzy rules of the FLNFN model.

Layer 5 (Output Node): Each node in this layer corresponds to a single output variable. The node integrates all of the actions recommended by layers 3 and 4 and acts as a center of area (COA) defuzzifier with

\[ y = u^{(5)} = \frac{\sum_{j=1}^{R} u^{(4)} \sum_{i=1}^{N} w_{ij} \phi_k}{\sum_{j=1}^{R} u^{(3)} \sum_{i=1}^{N} w_{ij} \phi_k} = \frac{\sum_{j=1}^{R} u^{(5)} \hat{y}_j}{\sum_{j=1}^{R} u^{(5)}} \]  \hspace{1cm} (6)

where \( R \) is the number of fuzzy rules, and \( y \) is the output of the FLNFN model. As described above, the number of tuning parameters for the FLNFN model is known to be \( (2 + 3P) \cdot N \cdot R \), where \( N \), \( R \), and \( P \) denote the number of inputs, existing rules, and outputs, respectively.

3. Learning Algorithms for the FLNFN Model

This section presents the learning algorithm for constructing the FLNFN model. The proposed learning algorithm comprises a structure learning phase and a parameter learning phase.

Figure 2 presents flowchart of the learning scheme for the FLNFN model. Structure learning is based on the entropy measure (EM) used to determine whether a new rule should be added to satisfy the fuzzy partitioning of input variables. Parameter learning is based on the proposed evolutionary learning algorithm, which minimizes a given cost function by adjusting the link weights in the consequent part and the parameters of the membership functions. Initially, there are no nodes in the network except the input–output nodes, i.e., there are no nodes in the FLNFN model. The nodes are created automatically as learning proceeds, upon the reception of incoming training data in the structure and parameter learning processes. In this study, once the learning process is completed, the trained-FLNFN can act as the nonlinear system controller. The rest of this section details the initialization phase, the structure learning phase and the parameter learning phase.
Figure 2. Flowchart of the proposed learning scheme for the FLNFN model.

A. Initialization Phase

a. Particle Representation

The first step in PSO learning algorithm is the coding of a fuzzy rule into a particle. Figure 3 shows an example of a fuzzy rule coded into a particle where \( i \) and \( j \) are the \( i^{th} \) dimension and the \( j^{th} \) rule. In Figure 3, \( m_{ij} \) and \( \sigma_{ij} \) are the mean and variance of a Gaussian membership function, respectively, and \( w_{ij} \) represents the corresponding link weight of the consequent part that is connected to the \( j^{th} \) rule node.

![Figure 3. Coding a fuzzy rule into a particle in the proposed DMPSO method.](image)

b. Data Normalization

For training data, finding the optimal solution is difficult because the range of training data is wide. Therefore, the data must be normalized. Let training data be transformed to the interval of \([0, 1]\)

\[ x_i = \frac{\hat{x}_i - \hat{x}_{i_{\min}}}{\hat{x}_{i_{\max}} - \hat{x}_{i_{\min}}} \]  

(7)

where \( x_i \) is the value after normalization; \( \hat{x}_i \) is the vector of the \( i^{th} \) dimension to be normalized; \( \hat{x}_{i_{\min}} \) is the minimum value of vector \( \hat{x}_i \); \( \hat{x}_{i_{\max}} \) is the maximum value of vector \( \hat{x}_i \).

c. Fitness Function Definition

In this study, we adopt a fitness function (i.e., objective function) to evaluate the performance of these composed fuzzy systems. The fitness function is defined as follows.

\[ F = \frac{1}{1 + \frac{1}{N_D} \sum_{k=1}^{N_D} (y_k - y_k^d)^2} \]  

(8)

where \( y_k \) represents the model output of the \( k^{th} \) data, \( y_k^d \) represents the desired output of the \( k^{th} \) data, and \( N_D \) represents the number of the training data.

B. Structure Learning Phase

The first step in structure learning is to determine whether a new rule should be extracted from the training data and to determine the number of fuzzy sets in the universe of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which \( m_{ij} \) and \( \sigma_{ij} \) represent the mean and variance of that cluster, respectively. For each incoming pattern \( x_i \), the rule firing strength can be regarded as the degree to which the incoming pattern belongs to the corresponding cluster. The entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will have lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule. For computational efficiency, the entropy measure can be calculated using the firing strength from (2) as

\[ EM_j = \sum_{i=1}^{N} D_j \log_2 D_j \]  

(9)

where \( D_j = \exp(-1/u_{ij}^{(2)}) \) and \( EM_j \in [0, 1] \). According to (9), the measure is used to generate a new fuzzy rule, and new functional link bases for new incoming
data are described as follows. The maximum entropy measure
\[ EM_{\text{max}} = \max_{1 \leq j \leq R(t)} EM_j \]
is determined, where \( R(t) \) is the number of existing rules at time \( t \). If \( EM_{\text{max}} \leq EM \), then a new rule is generated, where \( EM \in [0, 1] \) is a prespecified threshold that decays during the learning process.

In the structure learning phase, the threshold parameter \( EM \) is an important parameter. The threshold is set between zero and one. A low threshold leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value equals zero, then all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value \( EM \) will critically affect the simulation results. As a result of our extensive experiments and by carefully examining the threshold value \( EM \), which uses the range \([0, 1]\), we concluded that there was a relationship between threshold value \( EM \) and the number of input variables \( N \). Accordingly, \( EM = \tau N \), where \( \tau \) belongs to the range \([0.26, 0.3]\).

Once a new rule has been generated, the next step is to assign the initial mean and variance to the new membership function and the corresponding link weight for the consequent part. Since the goal is to minimize an objective function, the mean, variance, and weight are all adjustable later in the parameter learning phase. Hence, the mean, variance, and weight for the new rule are set as
\[ m_{i}^{R(i+1)} = x_i \]
\[ \sigma_{i}^{R(i+1)} = \sigma_{i\text{init}} \]
\[ v_{i}^{R(i+1)} = \text{random}[-1, 1] \]
where \( x_i \) is the new input and \( \sigma_{i\text{init}} \) is a prespecified constant.

C. Parameter Learning Phase

a. Particle Swarm Optimization (PSO) Algorithm

There are several major versions of the PSO algorithms. The following version modified by Shi and Eberhart [29] is used in this paper. Each particle is represented by a position and a velocity, which are updated as follows:
\[ v_{id} = w \cdot v_{id}^{-1} + c_1 \cdot r_1 \cdot (p_{id} - x_{id}) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id}) \]
\[ x_{id} = x_{id}^{-1} + v_{id} \]
Here, \( x_{id} \) and \( v_{id} \) are the \( d^{th} \) dimensional component of the position and velocity of the \( i^{th} \) particle at time step \( t \). \( p_{id} \) is the \( d^{th} \) component of the best (fitness) position the \( i^{th} \) particle has achieved by time step \( t \), and \( p_{gd} \) is the \( d^{th} \) component of the global best position achieved in the population by time step \( t \). The constants \( c_1 \) and \( c_2 \) are known as the “cognition” and “social” factors, respectively, as they control the relative strengths of the individual behavior of each particle and the collective behavior of all particles. The parameter \( w \in (0, 1) \) is the inertia of the particle, and controls the exploratory properties of the algorithm. Finally, \( r_1 \) and \( r_2 \) are two different random numbers in the range of 0 to 1, and are used to enhance the exploratory nature of the PSO.

b. Distance-based Mutation PSO (DMPSO) Algorithm

Ratnaweera et al. [33] stated that the lack of population diversity in PSO algorithms is a factor in their convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance. There are mainly two types of mutation operators: one type is based on particle position [32] and the other type is based on particle velocity [31]. The former method is the most common technique, and the mutation operator we proposed in this paper is also based on particle position.

In [30], Li, Yang, and Korejo modified the PSO by adding a mutation operator; the mutation operator provides a chance to escape from local optima. They focused on determining which random generator of the mutation operator is good for improving the population. The most important thing is the timing of application of the mutation operator. If mutation operator is applied too early, when the particles are not nearly convergent, the local search ability (local exploitation) of PSO is destroyed. If the mutation operator is applied too late, the parameter learning algorithm will be very inefficient. Hence, it is an important issue to consider when to apply mutation operator. In our study, we used the distances between each particle as a measure to determine whether the mutation operator needed to be applied or not, and the modified PSO we used is the so called distance-based mutation particle swarm optimization (DMPSO). Comparing the basic PSO with DMPSO, a convergent detection unit used to detect the particle convergent status is introduced. If the particles are convergent, the mutation operator will be processed. Otherwise, the mutation operator will be skipped.

The convergence detection unit computes the average distance from every particle to the particle that has global best value using (14)
where \( P'_i \) and \( G'_\text{best} \) indicate the \( i^{th} \) particle and the particle that has the best value at the \( t^{th} \) iteration, and \( S \) is the population size.

After the average distance is computed, the threshold \( Th_{\text{conv}} \) is used to determine whether the particles are close enough or not according to (15). If all particles are close enough, meaning that all particles are converging to the same position, the mutation operator will be applied. Otherwise, the mutation operator will be skipped.

\[
o(t) \leq Th_{\text{conv}} \tag{15}
\]

In this paper, every particle has its own mutation probability. If the average distance is greater than \( Th_{\text{conv}} \), implying that the majority of particles are not convergent, the mutation probability is set to zero, meaning that every particle does not mutate and the behavior of every particle is like a generic PSO. If the average distance is less than \( Th_{\text{conv}} \), meaning that all particles are converging to the same position, named \( G'_\text{best} \), the mutation probability (MP) of each particle is computed by (16).

\[
\text{success}_{i}(t) = \begin{cases} 1, & \text{if } F(P'_i) > F(P'_{\text{best}}) \\ 0, & \text{otherwise} \end{cases}
\]

\[
\text{progress}(t) = \sum_{i=1}^{s} \text{success}_{i}(t)
\]

\[
\text{MP} = \exp\left(-\frac{\text{progress}}{S}\right)
\]

where \( F(\cdot) \) denotes the fitness value of the particle, and \( \text{progress}(t) \) is the number of successful evolution particles at time step \( t \). The value of \( \text{success}_{i}(t) \) is set to 1 only when the \( i^{th} \) particle is successfully evolved at the \( t^{th} \) iteration, meaning that the local best fitness value is improved at the \( t^{th} \) iteration.

The design of mutation probability is based on the ratio of improved population. If the ratio of the improved population is higher, the mutation probability becomes smaller. Most particles are moving toward the best value that they have currently found. The lower probability guarantees the direction of the moving group will not be destroyed by the mutation operator. On the other hand, if most particles do not improve their fitness value, the population is in the stable status. There are two possibilities: the first possibility is that the particles have converged to the global optimum (or near global optimum). The application of the mutation operator at the moment will not destroy the moving group, because the particles still remember the global optimum, and the mutated particles will move toward the global optimum in the next generation. The second possibility is that the particles have converged to the local optimum, or in other words, they have fallen into a trap. The mutation operator provides a chance to escape from the trap. If some particles mutate and the new position the particle reaches has a better fitness value than the local optima, the other particles that are trapped will fly to the new position in the next iteration according to the PSO, meaning that the trapped particles can escape from the local optimum.

\section{Illustrative Examples}

In this section, we demonstrate the performance of the proposed FLNFN model using DMPSO algorithm for nonlinear system control. The FLNFN model is adopted to design controllers in three simulations of nonlinear system control problems: multi-input multi-output (MIMO) plant control [3], control of the truck backing system [34], and a water bath temperature control system [35].

\subsection{Multi-Input Multi-Output Control}

In this example, the MIMO plants [3] to be controlled are described by

\[
\begin{bmatrix}
    y_{r1}(k+1) \\
    y_{r2}(k+1)
\end{bmatrix}
=\begin{bmatrix}
    0.5y_{p1}(k) \\
    0.5y_{p2}(k)
\end{bmatrix}
\cdot
\begin{bmatrix}
    \sin(k\pi/45) \\
    \cos(k\pi/45)
\end{bmatrix}
+\begin{bmatrix}
    u_1(k) \\
    u_2(k)
\end{bmatrix}
\tag{17}
\]

The controlled outputs should follow the desired output \( y_{r1} \) and \( y_{r2} \), as specified by the following 250 pieces of data:

\[
\begin{bmatrix}
    y_{r1}(k) \\
    y_{r2}(k)
\end{bmatrix}
=\begin{bmatrix}
    \sin(k\pi/45) \\
    \cos(k\pi/45)
\end{bmatrix}
\tag{18}
\]

The inputs of the FLNFN-DMPSO are \( y_{p1}(k) \), \( y_{p2}(k) \), \( y_{r1}(k) \), and \( y_{r2}(k) \), and the outputs are \( u_1(k) \) and \( u_2(k) \).

Figure 4 plots the learning curves of the best performance of the FLNFN-DMPSO model for the affinity/fitness value, the CNFC–ISEL [36], the SEFC [37], and the Mamdani-type fuzzy system using symbiotic evolution algorithm (MFS–SE) [38], after the learning process of 600 generations. To demonstrate the control result, Figure 5 plots the control results of the desired output (solid line) and the model output (dotted line) after the learning process of 600 generations, and Figure 6 shows the errors of the proposed method. Table 1 presents the best and averaged affinity/fitness values after 600 generations of training. The comparison indicates that the best and averaged affinity/fitness values of FLNFN-DMPSO are better than those of other methods.
M.-T. Su et al.: Nonlinear System Control Using Functional-Link-Based NFN Model Embedded with Modified PSO

Figure 4. Learning curves of best performance of the FLNFN-DMPSO, CNFC-ISEL, SEFC, and MFS-SE in MIMO plant control.

Figure 5. Desired (solid line) and model (dotted line) output generated by FLNFN-DMPSO in MIMO plant control.

Figure 6. Errors of proposed FLNFN-DMPSO in MIMO plant control.

Table 1. Comparison of performance of FLNFN-DMPSO, FLNFN-PSO, CNFC-ISEL, SEFC, and MFS-SE in MIMO plant control.

<table>
<thead>
<tr>
<th>Method</th>
<th>Affinity/Fitness Value (Best)</th>
<th>Affinity/Fitness Value (Avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLNFN-DMPSO</td>
<td>0.9898</td>
<td>0.9856</td>
</tr>
<tr>
<td>FLNFN-PSO</td>
<td>0.9506</td>
<td>0.9149</td>
</tr>
<tr>
<td>CNFC-ISEL [36]</td>
<td>0.9786</td>
<td>0.9721</td>
</tr>
<tr>
<td>SEFC [37]</td>
<td>0.9581</td>
<td>0.9553</td>
</tr>
<tr>
<td>MFS-SE [38]</td>
<td>0.8560</td>
<td>0.8503</td>
</tr>
</tbody>
</table>

B. Control of Backing Up the Truck

Back ing a truck into a loading dock is difficult. It is a nonlinear control problem for which no traditional control method exists [34]. Figure 7 shows the simulated truck and loading zone. The truck’s position is exactly determined by three state variables $\phi$, $x$, and $y$, where $\phi$ is the angle between the truck and the horizontal, and the coordinate pair $(x, y)$ specifies the position of the center of the rear of the truck in the plane. The steering angle $\theta$ of the truck is the controlled variable. Positive values of $\theta$ represent clockwise rotations of the steering wheel, and negative values represent counterclockwise rotations. The truck is placed at some initial position and is backed up while being steered by the controller. The objective of this control problem is to use backward only motion of the truck to make it arrive at the desired loading dock $(x_{\text{desired}}, y_{\text{desired}})$ at a right angle ($\phi_{\text{desired}} = 90^\circ$). The truck moves backward as the steering wheel moves through a fixed distance $(d_j)$ in each step. The loading region is limited to the plane $[0, 100] \times [0, 100]$.

The input and output variables of the FLNFN-DMPSO must be specified. The controller has two inputs: truck angle $\phi$ and cross position $x$. When the clearance between the truck and the loading dock is assumed to be sufficient, the $y$ coordinate is not considered to be an input variable. The output of the controller is the steering angle $\theta$. The ranges of the variables $x$, $\phi$, and $\theta$ are as follows:

\[
0 \leq x \leq 100 \\
-90^\circ \leq \phi \leq 270^\circ \\
-30^\circ \leq \theta \leq 30^\circ
\]  

The equations of backward motion of the truck are

\[
x(k+1) = x(k) + d_j \cos \theta(k) + \cos \phi(k)
\]

\[
y(k+1) = y(k) + d_j \cos \theta(k) + \sin \phi(k)
\]

\[
\phi(k+1) = \tan^{-1} \left[ \frac{l \sin \phi(k) + d_j \cos \phi(k) \sin \theta(k)}{l \cos \phi(k) - d_j \sin \phi(k) \sin \theta(k)} \right]
\]

where $l$ is the length of the truck. Equation (20) yields
the next state from the present state.
Learning involves several attempts, each starting from
an initial state and terminating when the desired state is
reached; the FLNFN-DMPSO is thus trained. The train-
ing process continues for 2000 generations. The affinity
of the FLNFN-DMPSO is approximately 0.9637, and the
learning curve of FLNFN-DMPSO is compared with
those obtained using various existing models [36]-[38],
as shown in Figure 8. Figure 9 plots the trajectories of
the moving truck controlled by the FLNFN-DMPSO,
starting at initial positions \((x, y, \phi) = (40, 20, -30^\circ),
(10, 20, 150^\circ), (70, 20, -30^\circ), \) and \((80, 20, 150^\circ),\)
after the training process has been terminated. The considered
performance indices include the best affinity/fitness and
the average affinity/fitness value. Table 2 compares the
results. According to these results, the proposed
FLNFN-DMPSO outperforms various existing models.

![Figure 8](image1)

Figure 8. Learning curves of best performance of the
FLNFN-DMPSO, CNFC-ISEL, SEFC, and MFS-SE in control
of backing up the truck.

![Figure 9](image2)

Figure 9. Trajectories of truck, starting at four initial positions
under the control of the FLNFN-DMPSO after learning using
training trajectories.

<table>
<thead>
<tr>
<th>Method</th>
<th>Affinity/Fitness Value (Best)</th>
<th>Affinity/Fitness Value (Avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLNFN-DMPSO</td>
<td>0.9637</td>
<td>0.9502</td>
</tr>
<tr>
<td>FLNFN-PSO</td>
<td>0.9423</td>
<td>0.9355</td>
</tr>
<tr>
<td>CNFC-ISEL [36]</td>
<td>0.9558</td>
<td>0.9511</td>
</tr>
<tr>
<td>SEFC [37]</td>
<td>0.9516</td>
<td>0.9451</td>
</tr>
<tr>
<td>MFS-SE [38]</td>
<td>0.9398</td>
<td>0.9332</td>
</tr>
</tbody>
</table>
C. Control of Water Bath Temperature System

The goal of this subsection is to elucidate the control of the temperature of a water bath system according to
\[
\frac{dy(t)}{dt} = \frac{u(t) + \frac{Y_0 - y(t)}{T_R C}}{C}
\]  
(21)
where \(y(t)\) is the output temperature of the system in degrees Celsius; \(u(t)\) is the heat flowing into the system, \(Y_0\) is the room temperature; \(C\) is the thermal capacity of the system, and \(T_R\) is the equivalent thermal resistance between the borders of the system and the surroundings.

\(T_R\) and \(C\) are assumed to be essentially constant, and the system in (21) is rewritten in discrete-time form to some reasonable approximation. The system
\[
y(k + 1) = e^{-\alpha T_s} y(k) + \frac{\delta}{(1 - e^{-\alpha T_s})} u(k) + \left[1 - e^{-\alpha T_s}\right] y_0
\]  
(22)
is obtained, where \(\alpha\) and \(\delta\) are some constant values of \(T_R\) and \(C\). The system parameters used in this example are \(\alpha = 1.0015 e^{-4}\), \(\delta = 8.67973 e^{-3}\), and \(Y_0 = 25.0\) °C, which were obtained from a real water bath plant considered elsewhere [35]. The input \(u(k)\) is limited to 0 and 5V represents the voltage unit. The sampling period is \(T_s = 30\) s.

Figure 10 presents a block diagram for the conventional training scheme. This scheme has two phases – the training phase and the control phase. In the training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, training data with input vector \(I(k) = [y_p(k + 1) \ y_p(k)]\) and desired output \(y(k)\) can be defined, where the input vector of the FLNFN controller is the same as that used in the general inverse modeling [39] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal \(\hat{u}(k)\) is generated according to the input vector \(I'(k) = [y_{\text{ref}}(k + 1) \ y_p(k)]\), where \(y_p\) is the plant output and \(y_{\text{ref}}\) is the reference model output.

Figure 10 presents a block diagram for the conventional training scheme. This scheme has two phases – the training phase and the control phase. In the training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, training data with input vector \(I(k) = [y_p(k + 1) \ y_p(k)]\) and desired output \(y(k)\) can be defined, where the input vector of the FLNFN controller is the same as that used in the general inverse modeling [39] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal \(\hat{u}(k)\) is generated according to the input vector \(I'(k) = [y_{\text{ref}}(k + 1) \ y_p(k)]\), where \(y_p\) is the plant output and \(y_{\text{ref}}\) is the reference model output.

A sequence of random input signals \(u_{\text{ad}}(k)\) limited to 0 and 5V is injected directly into the simulated system described in (22), using the training scheme for the FLNFN-DMPSO controller. The 120 training patterns are selected based on the input–outputs characteristics to cover the entire reference output. The temperature of the water is initially 25 °C, and rises progressively when random input signals are injected.

This study compares the FLNFN-DMPSO controller to the FLNFN controller [18], the PID controller [40], the manually designed fuzzy controller [5], the FLNN controller [19], and the TSK-type NFN [7]. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set points regulation, the influence of impulse noise, a large parameter variation in the system, and the tracking capability of the controllers.

The first task is to control the simulated system to follow three set points
\[
y_{\text{ref}}(k) = \begin{cases} 
35^\circ C, & \text{for } k \leq 40 \\
55^\circ C, & \text{for } 40 < k \leq 80 \\
75^\circ C, & \text{for } 80 < k \leq 120 
\end{cases}
\]  
(23)

Figure 11 presents the regulation performance of the FLNFN-DMPSO controller. The regulation performance was also tested using the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller. To test their regulation performance, a performance index, the sum of absolute error (SAE), is defined by
\[
\text{SAE} = \sum_k |y_{\text{ref}}(k) - y(k)|
\]  
(24)
where \(y_{\text{ref}}(k)\) and \(y(k)\) are the reference output and the actual output of the simulated system, respectively.

The second set of simulations is performed to elucidate the noise rejection ability of the six controllers when some unknown impulse noise is imposed on the process. One impulse noise value of –5 °C is added to the plant output at the 60th sampling instant. A set point of 50 °C is adopted in this set of simulations. For the FLNFN-DMPSO controller, the same training scheme, training data, and learning parameters were used as in the first set of simulations. Figure 12 presents the behaviors of the FLNFN-DMPSO controller under the influence of impulse noise. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 352.32, 352.84, 418.5, 401.5, 379.22, and 361.96, which are shown in the second column of Table 3. The proposed FLNFN-DMPSO controller has a much better SAE value of regulation performance than the other controllers.

The second set of simulations is performed to elucidate the noise rejection ability of the six controllers when some unknown impulse noise is imposed on the process. One impulse noise value of –5 °C is added to the plant output at the 60th sampling instant. A set point of 50 °C is adopted in this set of simulations. For the FLNFN-DMPSO controller, the same training scheme, training data, and learning parameters were used as in the first set of simulations. Figure 12 presents the behaviors of the FLNFN-DMPSO controller under the influence of impulse noise. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 270.29, 270.41, 311.5, 275.8, 324.51, and 274.75, which are shown in the third column of Table 3. The FLNFN-DMPSO con-
controller performs quite well. It recovers very quickly and steadily after the occurrence of the impulse noise.

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. The value of \( 0.7u(k - 2) \) is added to the plant input after the 60th sample in the third set of simulations to test the robustness of the six controllers. A set point of 50 °C is adopted in this set of simulations. Figure 13 presents the behaviors of the FLNFN-DMPSO controller when the plant dynamics change. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 262.91, 263.35, 322.2, 273.5, 311.54, and 265.48, which are shown in the fourth column of Table 3. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN-DMPSO controller in the water bath system.

In the final set of simulations, the tracking capability of the FLNFN-DMPSO controller with respect to ramp-reference signals is studied. Define

\[
y_{\text{ref}}(k) = \begin{cases} 
34^\circ C, & \text{for } k \leq 30 \\
(34 + 0.5(k - 30))^\circ C, & \text{for } 30 < k \leq 50 \\
(44 + 0.8(k - 50))^\circ C, & \text{for } 50 < k \leq 70 \\
(60 + 0.5(k - 70))^\circ C, & \text{for } 70 < k \leq 90 \\
70^\circ C, & \text{for } 90 < k \leq 120 
\end{cases}
\] (25)

Figure 14 presents the tracking performance of the FLNFN-DMPSO controller. The SAE values of the FLNFN-DMPSO controller, the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 42.45, 44.28, 100.6, 88.1, 98.43, and 54.28, which are shown in the fifth column of Table 3. The results present the favorable control and tracking capabilities of the trained FLNFN-DMPSO controller in the water bath system. The aforementioned simulation results, presented in Table 3, demonstrate that the proposed FLNFN-DMPSO controller outperforms other controllers.

Table 3. Comparison of performance of various controllers to control of water bath temperature system.

<table>
<thead>
<tr>
<th>Controller</th>
<th>SAE</th>
<th>Regulation</th>
<th>Influence of Impulse Noise</th>
<th>Effect of Change in Plant Dynamics</th>
<th>Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLNFN-DMPSO</td>
<td>352.32</td>
<td>270.29</td>
<td>262.91</td>
<td>42.45</td>
<td></td>
</tr>
<tr>
<td>FLNFN [18]</td>
<td>352.84</td>
<td>270.41</td>
<td>263.35</td>
<td>44.28</td>
<td></td>
</tr>
<tr>
<td>PID [40]</td>
<td>418.5</td>
<td>311.5</td>
<td>322.2</td>
<td>100.6</td>
<td></td>
</tr>
<tr>
<td>Fuzzy [5]</td>
<td>401.5</td>
<td>275.8</td>
<td>273.5</td>
<td>88.1</td>
<td></td>
</tr>
<tr>
<td>FLNN [19]</td>
<td>379.22</td>
<td>324.51</td>
<td>311.54</td>
<td>98.43</td>
<td></td>
</tr>
<tr>
<td>TSK [7]</td>
<td>361.96</td>
<td>274.75</td>
<td>265.48</td>
<td>54.28</td>
<td></td>
</tr>
</tbody>
</table>
5. Conclusions

This study proposes an evolutionary neural fuzzy system, designed using functional-link-based neuro-fuzzy network (FLNFN) model embedded with distance-based mutation particle swarm optimization (DMPSO) algorithm. The proposed DMPSO learning algorithm consists of structure learning and parameter learning for the FLNFN model. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning method is based on the DMPSO algorithm, which has proved to be very effective for solving global optimization problem, can adjust the shape of fuzzy rule’s membership function and the corresponding weighting of FLNN. The simulation results have shown the proposed FLNFN-DMPSO method has more chance of converging to the global optimum and yields better performance than other existing models under some circumstances.

Acknowledgment

This work was supported in part by the UST-UCSD International Center of Excellence in Advanced Bio-engineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant Number: NSC-100-2911-I-009-101, and supported in part by the Aiming for the Top University Plan of National Chiao Tung University, the Ministry of Education, Taiwan, under Contract 100 W9633. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References


Miin-Tsair Su received the B.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1994 and the M.S. degree in power mechanical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1996. He is currently working toward the Ph.D. degree in electrical and control engineering at National Chiao Tung University, Hsinchu, Taiwan. His current research interests are fuzzy systems, neural networks, evolutionary algorithms, image processing, intelligent control, and pattern recognition.

Chin-Teng Lin received the B.S. degree in control engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1986 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, NCTU, where he is currently the University Provost and the Chair Professor of Electrical and Control Engineering. He served as the Founding Dean of Computer Science College of NCTU from 2005 to 2007. He is the textbook author of Neural Fuzzy Systems (Prentice-Hall) and Neural Fuzzy Control Systems with Structure and Parameter Learning (World Scientific). He has published over 140 journal papers, including about 80 IEEE Transactions papers. His research interests are in intelligent technology, soft computing, brain–computer interface, smart living technology, intelligent transportation systems, robotics and intelligent sensing, and nanobioinformation technologies and cognitive science (NBIC).

Dr. Lin is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies. He was a member of the Board of Governors (BoG) of the IEEE Systems, Man, Cybernetics Society (SMCS) from 2003 to 2005 and IEEE Circuit and Systems Society (CASS) (2005-2008), and the current BoG member of IEEE Computational Intelligence Society (CIS) (2008–2010). Dr. Lin is the IEEE Distinguished Lecturer from 2003 to 2005. He served as the Deputy EIC of the IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS—PART II, EXPRESS BRIEFS (2007–2008). Dr. Lin is the General Chair of FUZZ-IEEE2011 to be held in Taipei, and was the Program Chair of the 2006 IEEE International Conference on Systems, Man, and Cybernetics held in Taipei. He was the president of Board of Government (BoG) of Asia Pacific Neural Networks Assembly (APNNA) from 2004 to 2005. He has won the Outstanding Research Award granted by National Science Council (NSC), Taiwan, from 1997 to the present, the Outstanding Professor Award granted by the Chinese Institute of Engineering (CIE) in 2000, and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was also elected to be one of 38th Ten Outstanding Rising Stars in Taiwan (2000).

Sheng-Chih Hsu received the B.S. degree in Department of Automatic Control Engineering from Feng Chia University, Taichung, Taiwan, R.O.C., in 2006. He is currently working toward the Ph.D. degree in Institute of Electrical Control Engineering at National Chiao Tung University, Hsinchu, Taiwan. In addition to his current research on automatic vehicle detection and tracking, his interests include detection fire and smoke recognition, computer vision, and pattern recognition.

Dong-Lin Li received the B.S. degree in electrical engineering from National Chung Hsing University, Taichung, Taiwan, R.O.C, in 2006. Currently, he is working towards the Ph.D. degree in electrical control and engineering at National Chiao Tung University, Hsinchu, Taiwan, R.O.C. His research interest includes image processing, neural networks, fuzzy systems and evolutionary computation.

Cheng-Jian Lin received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, R.O.C., in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao Tung University, Taiwan, R.O.C., in 1991 and 1996. Currently, he is a full Professor of Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung County, Taiwan, R.O.C. His current research interests are soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and FPGA design.

Cheng-Hung Chen received the B.S. and M.S. degrees in computer science and information engineering from the Chaoyang University of Technology, Taiwan, R.O.C., in 2002 and 2004, respectively, and the Ph.D. degree in electrical and control engineering from the National Chiao Tung University, Taiwan, R.O.C., in 2008. Currently, he is an Assistant Professor of Electrical Engineering Department, National Formosa University, Yunlin County, Taiwan, R.O.C. His current research interests are fuzzy systems, neural networks, evolutionary algorithms, image processing, intelligent control, and pattern recognition.