An efficient recurrent fuzzy CMAC model based on a dynamic-group–based hybrid evolutionary algorithm for identification and prediction applications

Chin-Ling LEE¹, Cheng-Jian LIN²,

¹Department of International Business, National Taichung University of Science and Technology, Taichung, Taiwan
²Department of Computer Science & Information Engineering, National Chin-Yi University of Technology, Taichung, Taiwan

Abstract: This article presents an efficient TSK-type recurrent fuzzy cerebellar model articulation controller (T-RFCMAC) model based on a dynamic-group–based hybrid evolutionary algorithm (DGHEA) for solving identification and prediction problems. The proposed T-RFCMAC model is based on the traditional CMAC model and the Takagi–Sugeno–Kang (TSK) parametric fuzzy inference system. Otherwise, the recurrent network, which imports feedback links with a receptive field cell, is embedded in the T-RFCMAC model, and the feedback units are used as memory elements. The DGHEA, which is a hybrid of the dynamic-group quantum particle swarm optimization (QPSO) and the Nelder–Mead method, is proposed for adjusting the parameters of the T-RFCMAC model. In DGHEA, an entropy-based grouping technique is adopted to improve the searching capability and the convergent speed of quantum particles swarm optimization. Experimental results show that the proposed DGHEA-based T-RFCMAC model is more effective at identification and prediction than other models.

Key words: Fuzzy cerebellar model articulation controller, entropy, Nelder–Mead, particle swarm optimization, prediction, identification

1. Introduction

For nonlinear system processing, neural networks or neural fuzzy networks [1–3] are the most commonly used model. If a feedforward network is applied in the dynamic system, we first need to obtain the delay numbers of outputs and inputs [3]. In this situation, the accurate order of the dynamic system is usually not clear. To solve this problem, recurrent networks [4,5] are adopted for processing the dynamic system. In the present study, we present a novel recurrent network for solving different problems.

The cerebellar model articulation controller (CMAC) model was proposed by Albus [6,7]. It is a simple network architecture with a high convergence rate, high learning speed, good generalization capability, ease of hardware implementation, etc. The CMAC network has been employed in various areas successfully, such as robot control [8], pattern recognition [9], and signal processing [10]. However, Albus’s CMAC network still has three major limitations. First, it is difficult to select the memory structure parameters. While the common CMAC network has a constant value allocated to each hypercube, the derivative information is not maintained and the data for a quantized state are constant. In order to solve this problem, inconstant differentiable basis functions are used, such as fuzzy membership functions by Jou [11] and spline functions by Lane et al.
Second, for solving high-dimensional problems, the CMAC model requires a great amount of memory space. Even though some approaches have been used to deal with this problem, such as hashing functions, the "collision" problem and the complexity problem in high dimensions still exist. The third limitation is that the CMAC model needs a more rigorous theory for function approximation. Several modified CMAC models were proposed to overcome this problem by using fuzzy concept or B-spline functions [13,14]. The basic function in the CMAC model is a binary case function and the input vectors produce the same output in the same hypercube from the network. That is why the output of the CMAC model is usually not as smooth as the target function. Many researchers combine the fuzzy concept into the CMAC network [14,15]. That is, they replace the basic functions with fuzzy membership functions. The model is called fuzzy CMAC (FCMAC). In this study, a novel TSK-type recurrent fuzzy cerebellar model articulation controller (T-FRCMAC) is proposed.

In general, a backpropagation (BP) algorithm [16] is used in parameter learning and has powerful training ability to employ with a forward structure on networks. Because the BP algorithm depends on the gradient descend method, it has some disadvantages, such as the lower convergent speed and being easily trapped into local minimum. Thus, many evolutionary algorithms have been proposed to conquer the above-mentioned disadvantages, such as genetic algorithm (GA) [17], differential evolution (DE) [18], and particle swarm optimization (PSO) [19]. Recently, PSO has been commonly used to perform parameter learning in various models. PSO was introduced by Eberhart and Kennedy and inspired by the social behavior of animals, such as bird flocking, fish schooling, and the swarm theory. Because PSO has some attractive characteristics, such as simple implementation, high speed convergence, and direct logic, it has been widely developed in many areas. Although traditional PSO has a fast convergence speed, it is also easily trapped into local minima. Recently, several hybrid models have been studied to improve the disadvantage of the traditional PSO. Sun et al. [20] proposed a quantum-behaved PSO algorithm that guaranteed theoretically finding an optimal solution in search space. Zhang et al. [21] proposed a hybrid algorithm by integrating GA and gravitational search algorithm (GSA) to avoid premature convergence and to improve the search ability. In [21], crossover and mutation operators are used for jumping out of the local optima. Mohan and Albert [22] proposed a hybrid algorithm that integrates GA and PSO algorithms. The GA will be the main optimizer and the PSO will be used to guide the GA to locate optimal solutions quickly and effectively. An improved hybrid method applying both PSO and GA was also developed in [23]. To avoid premature convergence of PSO, Idoumghar et al. [24] presented a hybrid evolutionary algorithm based on the idea that PSO ensures fast convergence, while simulated annealing brings the search out of local optima because of its strong local-search ability. Abadin and Rezaei [25] presented a combinational method including PSO and continuous ant colony optimization in order to improve the search process. Singh et al. [26] developed a combination of mean Gbest particle swarm optimization (MGBPSO) and GSA. The basic inspiration is to integrate the ability of exploitation in MGBPSO with the ability of exploration in GSA to synthesize the strength of both approaches. The presented approach has an automatic balance between local and global searching abilities.

Because the above-mentioned algorithms do not change the characteristics of the original evolutionary algorithms, they also have problems with premature convergence and falling into local optima. To overcome the above-mentioned drawbacks, an efficient TSK-type recurrent fuzzy cerebellar model articulation controller (T-RFCMAC) model with a dynamic-group–based hybrid evolutionary algorithm (DGHEA) is proposed for solving prediction and identification problems in this study. The proposed DGHEA is a hybrid algorithm, integrating the dynamic-group quantum particle swarm optimization (QPSO) and the Nelder–Mead (NM) method. Unlike
traditional QPSO, the formation of the dynamic group in the proposed improved QPSO is not only used for particle-position update, but also for diversity of particle reference. An entropy-based criterion is used to determine whether or not the particle belongs to the group.

2. The structure of the T-RFCMAC model

In this section, a TSK-type recurrent fuzzy cerebellar model articulation controller (T-RFCMAC) model is proposed. The architecture of the T-RFCMAC model is shown in Figure 1, which includes five layers: the input space partition, the recurrent unit, the association space, the TSK-type output, and the defuzzification.

The proposed T-RFCMAC model involves the Gaussian basic function as the receptive field functions and the TSK-type output as the linear parametric function of the model output for learning. The one-dimensional Gaussian basic function is given as follows:

$$\mu(x) = e^{-\left(\frac{x-m}{\sigma}\right)^2}$$

where $x$ represents a specific input state and $m$ and $\sigma$ represent the corresponding center and variance. A $N_D$-dimension Gaussian basic function is given as follows:

$$\alpha_j = \prod_{i=1}^{N_D} e^{-\left(\frac{x_i-m_{ij}}{\sigma_{ij}}\right)^2}$$

Figure 1. The architecture of the T-RFCMAC model.
where $\Pi$ denotes the product operation, $\alpha_j$ represents the $j$th component of the association vector, $x_i$ is the input value of the $i$th dimension for a specific input state $x$, and $m_{ij}$ and $\sigma_{ij}$ represent the center and the variance of the receptive field functions. Furthermore, the input of each membership function for discrete time $t$ is given as follows:

$$r_{ij}(t) = x_i(t) + r_{ij}(t-1)\theta_{ij}, \quad (3)$$

where $\theta_{ij}$ represents the recurrent weight of the feedback unit, $r_{ij}(t)$ denotes the input of each membership function, $x_i$ denotes the input value of the $i$th dimension for a specific input state $x$, and $r_{ij}(t-1)$ denotes the storage blocks that store the last time information of the model. Each hypercube element of the receptive field functions is deduced to generate a partial fuzzy output by applying its corresponding association vector value as matching degree of input; the formula is given as follows:

$$a_{0j} + \sum_{i=1}^{N_D} a_{ij}x_i \quad (4)$$

where $a_{0j}$ and $a_{ij}$ represent the scalar values, $N_D$ is the number of the input dimensions, and $x_i$ is the $i$th input dimension. After employing Eq. (4), the single fuzzy output is defuzzified by the centroid of area approach into a scalar output $y$ and the real output $y$ is calculated as follows:

$$y = \frac{\sum_{j=1}^{N_c} \alpha_j \left( a_{0j} + \sum_{i=1}^{N_D} a_{ij}x_i \right)}{\sum_{j=1}^{N_c} \alpha_j}, \quad (5)$$

where $N_c$ represents the number of hypercube cells.

3. The proposed hybrid evolutionary algorithm

In this section, a hybrid evolutionary algorithm is proposed. The proposed algorithm consists of the QPSO and the NM algorithms.

3.1. Review of the NM method

The NM method, originally proposed by Nelder and Mead [27], is a simplex method for finding a local minimum in an optimal problem. A “simplex” is a convex hull that is structured by $n+1$ pointers $(X_0X_1X_2, \ldots, X_{n-1}X_n)$ in $n$ dimensions. Because the NM method has few parameters and is easy to comprehend and implement, it is an effective method in various fields, such as engineering, biology, chemistry, and physics. The NM method includes four parameters, namely reflection ($\alpha$), expansion ($\beta$), contraction ($\gamma$), and shrinkage ($\delta$).

In each iteration, we start the process with $n+1$ points and the objective function is optimized at each point of the simplex. After each transformation, a better point will replace the current worst point. The NM process is given as follows:

Step 1: Set the basic parameters: reflection ($\alpha$), expansion ($\beta$), contraction ($\gamma$), and shrinkage ($\delta$).

Step 2: Order the objective values at the points as $f(X_0) \leq f(X_1) \leq f(X_2) \cdots \leq f(X_n)$

Step 3: Center Point $M$: The centroid $M$ of the points excluding $X_{n+1}$ is calculated and the formula is given as follows:

$$M = \frac{1}{n} \sum_{i=0}^{n-1} X_i \quad (6)$$

where $n$ denotes the number of points and $X_i$ is the $i$th point.
Step 4: Reflection Point \( X_r \): Generate the reflection point \( X_r \).

\[
X_r = M + \alpha(M - X_n) \tag{7}
\]

Step 5: According to Eq. (7), three cases are described as follows:

Case 1: If \( f(X_0) \leq f(X_r) \leq f(X_n) \), we set \( X_n = X_r \).

Case 2: If \( f(X_r) < f(X_0) \), we use Eq. (8) to calculate the expansion point \( X_e \).

\[
X_e = X_r + \gamma(X_r - M) \tag{8}
\]

If \( f(X_e) < f(X_0) \), we set \( X_n = X_e \). Otherwise, we set \( X_n = X_r \).

Case 3: If \( f(X_r) > f(X_n) \), we use Eq. (9) to calculate the contraction point \( X_c \).

\[
X_c = M + \beta(X_r - M) \tag{9}
\]

If \( f(X_c) < f(X_n) \), we set \( X_n = X_c \). Otherwise, we use Eq. (10) to replace all the points \( X_i \) and generate a new simplex.

\[
X_i = X_0 + \delta(X_i - X_0) \text{ for } i \in \{1, 2, 3, \cdots, n\} \tag{10}
\]

Step 6: If the terminal criterion is satisfied, stop the process. Otherwise, move to Step 2. According to the above process, the NM method is a speedy algorithm for searching a local minimum without complex formulas.

3.2. The proposed DGHEA

In this subsection, a new DGHEA is introduced. The proposed DGHEA contains a swarm of particles. Each particle represents a parameter solution vector of the T-RFCAMC model. In traditional QPSO, the initial \( P_p \) particles are randomly generated to form a swarm. In the proposed DGHEA, the formation of the dynamic group is not only used for particle-position update, but also for diversity of particle references. The number of groups in the proposed DGHEA is not fixed, and it can confirm that the particles are not classified in unreasonable group. For group formation, the fitness values of all particles are ordered from the greatest to least. That is, the first particle has the best (the greatest) fitness value in the sorted swarm and becomes the group leader \( L_g \) in the first group. An entropy-based criterion is adopted as a measure to evaluate the group membership of the particles in sequence. Entropy is defined as follows:

\[
EM = - \sum_{i=1}^{P-1} D_i \log_2 D_i \tag{11}
\]

\[
D_i = L_1 - P_i \tag{12}
\]

where \( p \) represents the number of particles, \( P_i \) denotes the particles in sequence, and \( D_i \) is the distance between the group leader and other particles. In order to facilitate calculation, \( D_i \) is normalized to \( D_i \in [0, 1] \). By using Eq. (11), the threshold \( EM \) is used to determine whether a new group is generated or not. If \( D_i \) is larger than the threshold \( EM \), set the \( i \)th particle as a new leader and form a new group. The schematic of dynamic groups is shown in Figure 2.

QPSO was first inspired by analysis of the convergence of the traditional PSO and quantum systems. Sun et al. [20] proposed quantum-behaved PSO. The updated position of each particle is defined as follows:

\[
P_i = \varphi \times P_{best_i} + (1 - \varphi)G_{best} \tag{13}
\]
Figure 2. The schematic of dynamic groups.

\[ M_{\text{best}} = \frac{1}{P} \sum_{i=1}^{P} P_{\text{best}} \]

\[ x_i(t+1) = P_i \pm \beta |M_{\text{best}} - x_i(t)| \times \ln \left( \frac{1}{u} \right), \quad i = 1, 2, \ldots, P \]

where \( P_i \) represents local attractor, \( M_{\text{best}} \) is the mean best position defined as the mean of all the best positions of a swarm, \( \varphi \) and \( u \) are random numbers with uniform distribution on \([0, 1]\) respectively, and \( P \) is the number of particles. \( \beta \) is called the contraction–expansion coefficient and can be tuned to control the convergence speed of the algorithm. The symbol \( \pm \) is used to determine the minus sign and the plus sign. If the random value is larger than 0.5, the minus sign (−) is selected; otherwise, the plus sign (+) is selected. In the DGHEA, particles do not refer to \( P_{\text{best}} \) position. In contrast, the group leader is used to substitute the position of \( P_{\text{best}} \) and all the particles have suitable reference points. The new formula is given as follows:

\[ P_i = \varphi \times L_g + (1 - \varphi)G_{\text{best}} \]
Initialization of \( N+1 \) particles position and parameters set:
\[
g = 0; \\
G = 0;
\]
Calculate the fitness value of particles \( F_i \)
Rank particles from best to worse
Update the position of particles by using NM algorithm
NM
Update the particle position by using QPSO
Calculate the distance \( D_i \) between best performance of particle and other particles
Calculate the degree \( EM \) of confusion of particles by using formula of entropy
Is \( EM < D_i \) ?
Yes \( g = g + 1 \)
No
Update the particle position by using QPSO
Calculate the fitness value and update \( P_{best} \) and \( G_{best} \) position
Is \( F_i > F_{ith} \) ?
Yes \( G = G + 1 \)
No
Is \( G > G_{th} \)?
Yes
End
No

Figure 3. The DGHEA flowchart.

Particle

Fuzzy Hypercube_1  Fuzzy Hypercube_2  \ldots  \ldots  Fuzzy Hypercube_j  \ldots  \ldots  Fuzzy Hypercube_p

\[
\begin{align*}
m_{1j} & \quad m_{2j} & \ldots & \quad m_{nj} \\
\sigma_{1j} & \quad \sigma_{2j} & \ldots & \quad \sigma_{nj} \\
\theta_{1j} & \quad \theta_{2j} & \ldots & \quad \theta_{nj}
\end{align*}
\]

Figure 4. Coding the parameters of T-RFCMAC model into a particle.
Step 5: Updating \( P_{\text{best}} \) (personal best) and \( G_{\text{best}} \) (global best)

After updating the position of each particle, the \( P_{\text{best}} \) and \( G_{\text{best}} \) will be updated if necessary. First, the fitness values of each particle are compared with the personal best fitness value \( (P_{\text{best}}) \). If the current fitness value is larger than \( P_{\text{best}} \), the \( P_{\text{best}} \) will be replaced by the current fitness value and also the position of the \( P_{\text{best}} \) in \( N \)-dimensional space will be updated. The updating approach of the global best fitness value \( (G_{\text{best}}) \) is similar to the updating approach of \( P_{\text{best}} \).

Step 6: If the number of generations is satisfied, the algorithm will return to Step 2; otherwise, the algorithm will be terminated.

4. Simulation results

In this section, two examples are simulated. These simulation studies include the Mackey–Glass chaotic series prediction problem and the dynamic system identification problem. The chaotic time series and the dynamic system identification problems are difficult to predict and identify. Recently, several researchers [28–39] tried to improve the prediction and identification performance. Therefore, in the present study we compare the performance of the DGHEA-based T-RFCMAC model with those of other methods [28–39]. The root mean squared error (RMSE) is used for performance evaluation in this study. Two simulation comparisons are considered. One is the performance comparison of various evolutionary learning algorithms and the other is that of various models. All the programs were developed by using Visual Studio 2013 C++ software on an Intel i7-870 2.93 GHz personal computer.

4.1. Prediction of Mackey–Glass chaotic time series

The Mackey–Glass chaotic time series \( x(t) \) in consideration here is generated from the following delay differential equation:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t),
\]

where the initial value is given as \( \tau = 17 \) and \( x(0) = 1.2 \). Crowder extracted 1000 input–output data pairs \( \{x, y^d\} \) using four past values of \( x(t) \), i.e.

\[
[x(t - 18), x(t - 12), x(t - 6), x(t) ; x(t + 6)]
\]

Four inputs of the T-RFCMAC model correspond to these value of \( x(\Delta t) \), and one output represents the value \( x(t + \Delta t) \), where \( \Delta t \) is a time interval into the future. The first 500 pairs are used for training data, whereas the remaining 500 pairs are used for testing data. In this experiment, the parameters are set and shown in Table 1. Figure 5 shows the prediction results of the actual output and the output of T-RFCMAC model with DGHEA. Figure 6 describes the learning curves of the T-RFCMAC model with various learning algorithms, including the proposed DGHEA, QPSO [20], and NM [27]. Table 2 shows the performance comparison of the T-RFCMAC model with various learning algorithms. Table 3 lists the performance comparison of various models, including the proposed T-RFCMAC model with DGHEA, P-FCMAC [28], GEFREX [29], ANFIS [30], Kim and Kim [31], DENFIS [32], SEFC [33], and NFIS-SEELA [34]. Experimental results show that the T-RFCMAC model with DGHEA performs better than other methods.
Figure 5. Prediction results of the actual output and the output of T-RFCMAC model with DGHEA.

Figure 6. Learning curves of the T-RFCMAC model with various evolutionary learning algorithms.

Table 1. The parameters of the proposed method for chaotic time series prediction.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Generation</th>
<th>Number of particles</th>
<th>Number of fuzzy hypercubes</th>
<th>NM parameter $\alpha, \beta, \gamma, \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1000</td>
<td>52</td>
<td>3</td>
<td>1, 0.5, 2, 0.5</td>
</tr>
</tbody>
</table>

Table 2. Comparison results of the T-RFCMAC model with various evolutionary learning algorithms for chaotic time series prediction.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>0.0030</td>
<td>0.0057</td>
<td>0.0089</td>
<td>0.0065</td>
<td>0.0631</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0.0087</td>
<td>0.0182</td>
<td>0.0149</td>
<td>0.0176</td>
<td>0.0827</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>0.0057</td>
<td>0.0137</td>
<td>0.0125</td>
<td>0.0121</td>
<td>0.0727</td>
</tr>
<tr>
<td></td>
<td>Standard error</td>
<td>0.0046</td>
<td>0.0030</td>
<td>0.0022</td>
<td>0.0027</td>
<td>0.0078</td>
</tr>
</tbody>
</table>

Table 3. Comparison results of various models for chaotic time series prediction.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>0.0030</td>
<td>0.0094</td>
<td>0.0061</td>
<td>0.007</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.026</td>
<td>0.033</td>
<td>0.032</td>
<td>0.00675</td>
</tr>
</tbody>
</table>

4.2. Identification of dynamic system

In this example, the identification of the dynamic system is expressed as follows:

$$y_p(t+1) = 0.72y_p(t) + 0.025y_p(t-1)u(t-1) + 0.01u(t-2) + 0.2u(t-3)$$  \hspace{1cm} (19)
The output of dynamic system depends on three last inputs and two last outputs. Two current variables, \( y_p(t) \) and \( u(t) \), are fed as inputs into the T-RFCMAC model. The input signal \( u(t) \) is used to determine the results:

\[
u(t) = \begin{cases} 
\sin\left(\frac{\pi t}{25}\right), & t < 250 \\
1.0, & 250 \leq t < 500 \\
-1.0, & 500 \leq t < 750 \\
0.3\sin\left(\frac{\pi t}{25}\right) + 0.1\sin\left(\frac{\pi t}{12}\right) + 0.6\sin\left(\frac{\pi t}{15}\right), & 750 \leq t < 1000
\end{cases}
\]  

(20)

In this experiment, the adjustable parameters are set and shown in Table 4. Figure 7 displays the identification results of the actual output and the output of the T-RFCMAC model with DGHEA. Figure 8 shows the learning curves of the T-RFCMAC model with various evolutionary learning algorithms, including the proposed DGHEA, QPSO [20], and NM [27]. Table 5 shows the performance comparison of the T-RFCMAC model with various learning algorithms. Table 6 lists the performance comparison of various models, including the proposed T-RFCMAC model with DGHEA, ERNN [35], RFNN [36], WRFNN [37], SRFNN [38], Feedforward T2-FNN, and RIFNN [39]. Experimental results show that the T-RFCMAC model with DGHEA also performs better than other models.

Table 4. The parameters of the proposed method for dynamic system identification.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Generation</th>
<th>Number of particles</th>
<th>Number of fuzzy hypercubes</th>
<th>NM parameter ( \alpha, \beta, \gamma, \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1000</td>
<td>52</td>
<td>3</td>
<td>1, 0.5, 2, 0.5</td>
</tr>
</tbody>
</table>

Conclusion
This study proposes an efficient T-RFCMAC model with DGHEA for solving identification and prediction problems. The proposed DGHEA learning method is a hybrid of the modified QPSO and NM algorithms. The traditional QPSO is modified and added to the group concept to promote the reference information of particles and find the global optimal quickly. Two simulation studies included the dynamic system identification problem...
Table 5. Comparison results of the T-RFCMAC model with various evolutionary learning algorithms for dynamic system identification.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.0063</td>
<td>0.0090</td>
<td>0.0090</td>
<td>0.0087</td>
<td>0.1663</td>
</tr>
<tr>
<td>Worst</td>
<td>0.0090</td>
<td>0.0305</td>
<td>0.0091</td>
<td>0.0090</td>
<td>0.6710</td>
</tr>
<tr>
<td>Average</td>
<td>0.0086</td>
<td>0.0119</td>
<td>0.009003</td>
<td>0.0090</td>
<td>0.4196</td>
</tr>
<tr>
<td>Standard deviation (STD)</td>
<td>0.00060267</td>
<td>0.0046</td>
<td>0.0000038352</td>
<td>0.000104</td>
<td>0.1383</td>
</tr>
</tbody>
</table>

Table 6. Comparison results of various models for dynamic system identification.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>0.0063</td>
<td>0.036</td>
<td>0.072</td>
<td>0.0574</td>
</tr>
<tr>
<td>Models</td>
<td>SRFNN [38]</td>
<td>Feedforward</td>
<td>RIFNN [39]</td>
<td>-</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.02</td>
<td>0.0155</td>
<td>0.0125</td>
<td>-</td>
</tr>
</tbody>
</table>

and the Mackey–Glass chaotic series prediction problem. Simulation results show that the proposed T-FRCMAC model with DGHEA has a better RMSE performance and faster convergent speed than other methods.

Acknowledgment
The authors would like to thank the Ministry of Science and Technology of the Republic of China, Taiwan, for financially supporting this research under Contract No. MOST 106-2221-E-167-016.

References


