American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

Model Reference Adaptive Control based on a Simplified Recurrent Neural Network Trained by Gbest-Guided Gravitational Search Algorithm to Control Nonlinear Systems

Omar Farouq Lutfy[a]*, Ahmed Lateef Jassim[b]

[a]*Assistant Prof. Dr., University of Technology, Baghdad, Iraq*
[b]*Engineer, University of Technology, Baghdad, Iraq*
[a]*Email: 60157@uotechnology.edu.iq*
[b]*Email: cse.61120@uotechnology.edu.iq*

Abstract

This paper presents an intelligent Model Reference Adaptive Control (MRAC) strategy based on a Simplified Recurrent Neural Network (SRNN) for nonlinear dynamical systems. This network is an enhanced version of a previously reported modified recurrent network (MRN). More precisely, the enhancement in the SRNN structure was realized by employing unity weight values between the context and the hidden layers in the original MRN structure. The newly developed Gbest-guided Gravitational Search Algorithm (GGSA) was adopted for optimizing the parameters of the SRNN structure. To show the efficiency of the proposed SRNN-based MRAC, three different nonlinear systems were considered as case studies, including complex difference equations and the water bath temperature control system. From an extensive set of evaluation tests, which includes a control performance test, a disturbance rejection test, and a generalization test, the proposed SRNN-based MRAC system demonstrated its effectiveness with regards to precise control and good robustness and generalization abilities. Furthermore, compared to other Neural Network (NN) structures, including the original MRN and the Multilayer Perceptron (MLP) NN, the SRNN structure attained superior results due to the utilization of a reduced set of parameters. From another study, the GGSA accomplished the best optimization results in terms of control precision and convergence speed compared to the original Gravitational Search Algorithm (GSA).

Keywords: Modified Elman neural network; modified recurrent neural network; artificial neural network; gbest-guided gravitational search algorithm; model reference adaptive control.

---

* Corresponding author.

1. Introduction

With the increased ability to understand aspects about systems, external disturbances, and operating conditions, intelligent control has become an effective strategy in various engineering and industrial applications. Heuristic reasoning and learning from previous experiences are the main features that enable intelligent control methods to achieve a human-like processing in solving a particular problem. In particular, Artificial Neural Networks (ANNs), fuzzy logic, and Evolutionary Algorithms (EAs) are the most popular techniques for developing intelligent control systems. Due to their generalization and learning capabilities, many control and identification problems have been successfully addressed with the aid of ANNs. However, static ANNs suffer from some limitations due to the absence of dynamical characteristics, which negatively affect the overall network approximation ability. For this reason, recurrent neural networks (RNNs) have attracted extra attention among researchers recently, particularly in the process control field [1, 2]. RNNs are used to acquire sequential or time-varying patterns. In essence, a RNN has feedback (closed loop) connections [3]. RNNs can be farther divided into two types, depending upon the connections between layers as fully and partially recurrent networks. In Fully Recurrent Networks (FRNs), each node is connected to all other nodes. In addition, there might be self-feedback connections in some nodes. On the other hand, in Partially Recurrent Networks (PRNs), only certain nodes have feedback connections with other nodes or with themselves. In fact, PRNs combine the advantage of feedforward and recurrent networks [3, 4], and they have been widely used in linear and nonlinear control design for many control problems.

In the literature, one of the most widely used neural network types is the Elman Network (ELN), which was proposed by Elman [5]. Aiming at improving the dynamic characteristics and the approximation capability of the original ELN, Pham and Liu [6] proposed a modified ELN structure, which was called the Modified Elman Neural Network (MELN). The MELN was successfully employed for solving different modelling and control problems. For instance, Shiltagh [4] proposed to use adjustable weights that connect the hidden and the context layers to improve the performance of the MELN. This network structure, which was called the Modified Recurrent Network (MRN), was exploited to control nonlinear dynamical systems. Shyu and his colleagues [7] presented a neural networks based on a model reference adaptive speed control. To establish the training patterns, the robust observer-based model reference tracking control technique was used. Then, to robustly track a reference model for an induction motor drive, the trained neural network was used as the adaptive speed controller. Ge and his colleagues [8] utilized the MELN to control the speed of an ultrasonic motor. In another work, Thammano and Ruxpakawong [9] suggested a new strategy in defining the weights of the original ELN. More specifically, the authors suggested to use multi-valued weights based on the value of the input samples. Ma and Zhang [10] designed a model reference adaptive control (MRAC) system for the aero-engine based on the neural network sliding mode variable structure decoupling controller. Specifically, the Radical Basis Function (RBF) neural network was used as the output of the sliding mode control, and the sliding mode switching function was used as the neural network input. The weights of the RBF network were adjusted adaptively by the error between the reference model output and actual output. Zhou and his colleagues [11] designed a control method to control the air chamber pressure in the slurry shield tunneling utilizing the MELN. In order to accomplish a fast response for the real power control in hybrid generation systems, Huang [12] suggested an intelligent controller which combines a RBF neural network and a MLEN to achieve maximum

power point tracking in the power generation system. Fang and his colleagues [13] proposed a RBF NN to control the single-phase active power filter (APF)
using a model reference adaptive sliding mode structure. The RBF NN was utilized to approximate the nonlinear function and eliminate the modeling error in the APF system. Reference [14] proposed the application of the RBF based on a special type of neural networks which belongs to a class of associative memory neural networks. The RBF NN was used in a new method as a controller in the MRAC. In another work, Lutfy and Dawood [15] proposed another type of recurrent networks, namely a Self-Recurrent Wavelet Neural Network, which was used in the MRAC to control some nonlinear systems.

It is worth to highlight that the above works used the general MELN structure that contains several sets of connection weights, which adds a complexity to the control system design.

With regards to the training process, gradient-based methods are the most widely used techniques for training the ELN and the MELN [16]. Nevertheless, these training techniques have slow convergence speed and they might easily trapped at local minima of the optimization problem [11-17]. As better alternative training methods, Evolutionary Algorithms (EAs) are increasingly utilized to avoid the limitations of gradient-based optimization methods.

As a newly developed optimization algorithm, the gravitational search algorithm (GSA), which was proposed by Rashedi and his colleagues [18], is a population-based search algorithm that uses Newton's universal law of gravitation, mass interaction, and law of motion [19]. The GSA uses certain objects, which are known as agents, to perform the optimization process. The positions of these agents represent possible solutions for the optimization problem and the agent's performance is measured by the size of its mass. In this context, agents with heavy masses, which move slowly, apply strong gravitational forces and attract other agents with smaller masses. This process causes all agents to gradually move towards the global optimal solution [18, 20]. In the present work, to improve the approximation ability of the MRN proposed in [4], a Simplified Recurrent Neural Network (SRNN) is proposed and used within the MRAC structure. More precisely, the improvement was attained by using unity values for the weights that connect the context and the hidden layers in the original MRN structure. The proposed SRNN structure is used within the MRAC to control nonlinear systems. Moreover, to avoid drawbacks of gradient-based optimization methods, the newly developed Gbest-guided Gravitational Search Algorithm (GGSA), which is classified as an EA, is employed for optimizing the weights of the SRNN structure. The remaining parts of the paper are arranged as follows: Section 2 explains the structure of the proposed SRNN-based MRAC. A background of RNNs and an overview of training the SRNN are given in Section 3. Basic concepts of the GSA, the GGSA, and the procedure of applying the latter are discussed in Section 4. In order to show the efficiency of the proposed SRNN-based MRAC, an extensive set of evaluation tests and two comparative studies are conducted in Section 5. Finally, a few remarks are provided in Section 6 to conclude the paper.

2. The SRNN-based MRAC Structure

The idea of the MRAC is to define the required response of the controlled system by a reference model, which is

simply a plant of a known dynamical structure. The task of this model is to provide the desired output by applying a given input signal. Subsequently, by utilizing a suitable optimization technique, the main goal of MRAC design is to regulate the controller parameters by minimizing the difference between the outputs of the reference model and the controlled system [21]. Based on this design approach, the SRNN structure was used in this study as the controller to realize a nonlinear MRAC scheme.

Figure 1 illustrates a schematic diagram of the suggested SRNN-based MRAC system. From this figure, it is clear that $r(k)$ is the reference input, $u(k)$ is the control action, $y_m(k+1)$ is the reference model output, $y(k+1)$ is the actual system output, and $e(k+1)$ is the error between the outputs of the reference model and the system [21].

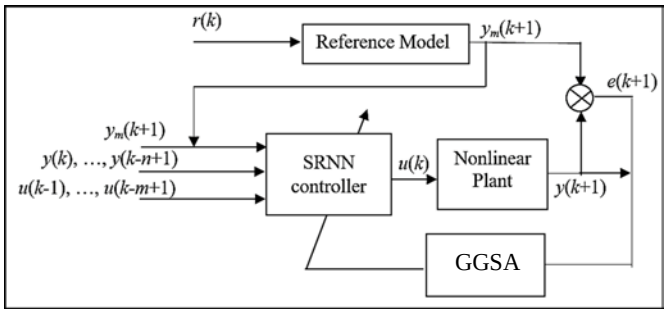


Figure 1: A schematic diagram of the SRNN-based MRAC

It is important to notice that the design of the SRNN controller within the MRAC depends on the generalized inverse control technique. Hence, the SRNN structure is trained to act as an inverse controller, as demonstrated in Figure1. To illustrate this idea, consider the following nonlinear autoregressive moving average (NARMA) model, which is utilized for representing single-input single output (SISO) nonlinear systems [21-23]:

$$y(k+1) = f\big(y(k),y(k-1),\ldots,y(k-n+1),u(k),\ldots,u(k-m+1)\big), \tag{1}$$

where $y(k)$ is the system output and $u(k)$ is the system input, $f(.)$ is a nonlinear function, n and m are the orders of the system, and k represents the discrete-time instant. In order to derive the control law, it is assumed that Equation (1) above is invertible, which results in the following equation:

$$u(k) = h\big(y_m(k+1),y(k),y(k-1),\ldots,y(k-n+1),u(k-1),\ldots,u(k-m+1)\big), \tag{2}$$

where, $y_m(k+1)$ represents the reference model output at time instant $(k+1)$ and $h(.)$ is the inverse function of f in Equation (1), such that: $h(.) = f^{-1}(.)$. In order to realize the control law given in Equation (2), a suitable function approximator must be used to approximate the inverse function, $h(.)$. In particular, due to its remarkable nonlinear function approximation capability, the SRNN is employed in this work to achieve this

task. It can be seen from Figure (1) that the training of the SRNN structure is accomplished by decreasing the error signal between the outputs of the reference model and the actual system. More specifically, this signal of the error is given by the following equation [24]:

$$E = \frac{1}{2}\sum_{k=1}^{N}(y(k+1) - y_m\,(\mathrm{k}+1))^2, \qquad (3)$$

where *N* represents the samples number. In particular, this error signal is employed as the performance index to be reduced by a suitable optimization method, specifically the GGSA in this work, as illustrated in Figure 1.

3. Background of Recurrent Neural Networks

This section clarifies the structure of the SRNN controller. At first, an outline of the basic and the modified Elman networks is given. After that, the structure of the SRNN is explained in details.

3.1 Basic and Modified Elman Networks

The most commonly known architecture of RNNs is the ELN. This network extends the feedforward network using context nodes whose task is to remember the network's previous action. At a specific time *k*, the input nodes take the first input pattern and together with the context nodes activate the nodes in the hidden layer. Then, the hidden layer nodes activate the output nodes and at the same time activate the context nodes. At the next time step, *k*+1, the above steps are repeated and this time the context nodes store the previous outputs of the hidden nodes at time *k*. In the original ELN, the feedback connection weights between the hidden and the context layers are fixed and all the other connection weights in the network are adjustable [4, 5, 25, 26].

To improve the approximation capability of the basic ELN, a modified version has been proposed in [6], which was called the Modified Elman Network (MELN). The idea of the MELN is to add other feedback connections to the context units, which are known as the "self-feedback" links, each of which with a fixed gain to boost the approximation capability of the basic ELN. Specifically, each self-feedback connection gain is fixed, and can be found manually from 0 to 1 [4].

3.2 The Simplified Recurrent Neural Network (SRNN)

The problem of finding suitable values for the gains, α, in the self-feedback connection in the MELN is inconvenient and time consuming, particularly when the network has a large number of hidden layer nodes. Hence, a suggestion was made to find the optimal gain values by a particular optimization method, and moreover to adopt adjustable weights between the hidden and context layers [4]. This network structure was called the Modified Recurrent Network (MRN). As a matter of fact, the MRN structure includes a large number of parameters which might negatively affect the network approximation ability, since more parameters cause more uncertainty. Thus, it was suggested to use unity values for the weights connecting the context and the hidden layers in the original MRN structure. This structure is called the simplified recurrent neural network (SRNN), which can be considered as a simplified version of the original MRN. The SRNN structure is shown in Figure 2.

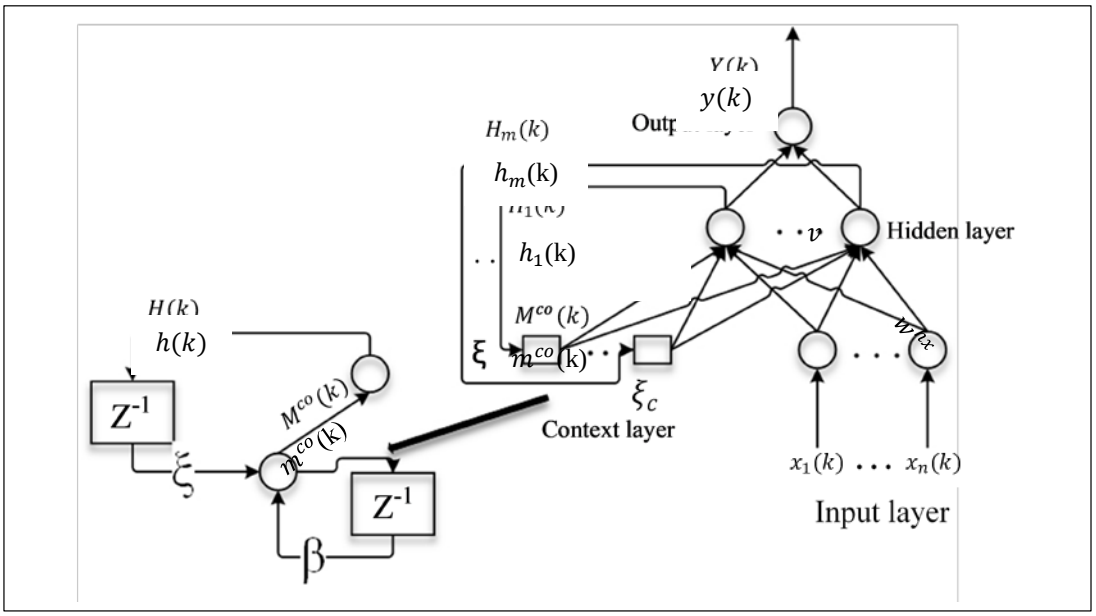


Figure 2: Structure of the SRNN controller.

Evidently, Figure 2 shows that the SRNN structure consists of an input layer, a hidden layer, and an output layer, which are explained below.

Layer 1: This layer is the input layer, which consists of two parts, namely real inputs and context units. The real units convey the input variables, (x_1, x_2,…, x_n), to the hidden layer. The output of each context node is calculated by the expression below:

$$m_c^{co}(k)=\beta_c(k)m_c^{co}(k-1)+\xi_c(k)h_C(k-1), \qquad (4)$$

Where c=1, 2, ⋯, *C*, and *C* is the number of nodes in the context layer. Beta (β) and Zeta (ξ) are adjustable connections from context and hidden layers, respectively. $h_c(k-1)$ and $m_c^{co}(k-1)$ are past outputs of the hidden and the context layers, respectively.

Layer 2: This is the hidden layer whose task is to activate both the output and the context layers. The response of the m^{th} hidden node is expressed as follows:

$$h_m(k) = f\left[\sum_{c=1}^{C} m_c^{co}(k) + \sum_{i=1}^{n} w_{mi}^{hx} x_i(k)\right], \qquad (5)$$

where m=1, 2, ..., *C*, and *C* is the number of nodes in the hidden layer which is also equal to the number of nodes in the context layer, x_i is the i^{th} input variable, where i = 1, 2, …, n and n represents the number of nodes

in the input layer, *f*(.) represents a nonlinear activation function, and w_{mi}^{hx} represents a weight between the i^{th} input node and the m^{th} hidden node.

Layer 3: There is a single node in this layer, known as the output layer, which produces the output of the SRNN structure using the following equation:

$$y(k) = \sum_{j=1}^{m} v_j h_j(k) , \tag{6}$$

where v_j is the j^{th} connection weight between thej^{th}hidden node and the output node, while $y(k)$ denotes the control action at time instant (*k*).

3.3 Training the SRNN controller

In light of the preceding discussion, it is obvious that the SRNN structure possesses different modifiable weights, as shown below:

$$S = [\beta_j \zeta_j w_{mi}^{hx} v_j] \tag{7}$$

For achieving the required SRNN performance, the optimal values for the weights in Equation (7) must be obtained. To achieve this objective, the GGSA is utilized in this work as the optimization method for the SRNN-based MRAC controller.

4. Gravitational Search Algorithm

To explain the GSA optimization method, assume that there are *N* agents, each with a dimension *D*, that are scattered in a particular search space. In this search space, each agent has a specific position, as shown below:

$$x_i=[x_i^1, \ldots,x_i^d,\ldots,x_{i,}^D], \tag{8}$$

where x_i^d denotes the i^{th} agent position in the d^{th}dimension.

The top $Kbest$ agents apply a gravitational force that can be determined by the following formula [18]:

$$F_i^d(t) = \sum_{j \in Kbest,\, j \neq i} Rand_j \;.\; F_{ij}^d(t) \qquad (9)$$

$$F_{ij}^d(t) = G(t).\frac{M_i(t)\,.\,M_j(t)}{R_{ij}(t)+\varepsilon}\;.\;\left(x_j^d(t) \;-\; x_i^d(t)\right) \qquad (10)$$

where i = 1, 2, . . . , N and j = 1, 2, . . . , D. $Rand_j$ is a random number from [0, 1], $G(t)$ is the gravitational coefficient, $M_i(t)$ and $M_j(t)$ signify masses of solutions i and j, respectively, $R_{ij}(t)$ is the Euclidian distance from solution i to solution j, ε is a small constant, and $Kbest$ denotes a set with the first K agents having the best fitness values.

The parameters $G(t)$ and $R_{ij}(t)$ in Equation (10) are obtained as given below:

$$G(t) = G_0 * exp(-\delta * \frac{L}{L_{max}}) \qquad (11)$$

$$R_{ij}(t) = \left\| x_i^d(t), x_j^d(t) \right\|_2 \qquad (12)$$

where G_0 is the initial gravitational constant, δ is the decrease coefficient, L is the current iteration, and L_{max} is the maximum number of iterations, respectively. Additionally, for the i^{th} solution X_i, its mass is defined by:

$$M_i = \frac{S_i(t)}{\sum_{j=1}^{N} S_j(t)} \qquad (13)$$

$$S_i(t) = \frac{Fit_i(t) - \; worst(t)}{best(t) \; - \; worst(t)}, \qquad (14)$$

where $Fit_i(t)$ represents agent's i fitness at time t, and $worst(t)$ and $best(t)$ represent the minimum and the maximum fitness values (for a minimization problem) at time t. In particular, $worst(t)$ and $best(t)$ are found according to the following expressions [18]:

$$worst(t) = min_{j \in [1,\ldots,N]} Fit_j(t) \qquad (15)$$

$$best(t) = max_{j \in [1,\ldots,N]} Fit_j(t) \quad (16)$$

Then, the next step is to find the acceleration of each agent, as given below [18]:

$$a_{ij}^d = \frac{F_{ij}^d}{M_i} \quad (17)$$

Subsequently, the new velocity and position of each agent are determined as follows:

$$v_{ij}^d(t+1) = rand_i * v_{ij}^d(t) + a_{ij}^d(t) \quad (18)$$

$$x_{ij}^d(t+1) = x_{ij}^d(t) + v_{ij}^d(t) \quad (19)$$

4.1 Gbest-Guided Gravitational Search Algorithm

Population-based heuristic algorithms are built using two main operations, namely exploration and exploitation. Expanding the search space is the task of exploration, while further searching the promising solution areas to find the optimal solution is the objective of the exploitation. In general, exploration is performed during the early iterations. With progression of iterations, exploration gradually decreases while exploitation gradually increases to avoid the problem of getting stuck at local minima. To ensure the best possible optimization performance, there should be a reasonable compromise between the exploration and the exploitation operators [18]. Nonetheless, several studies proved that the original GSA has a relatively slow and ineffective exploitation [27- 29]. Therefore, the authors in [27] proposed a modified variant of the GSA and they called it the Gbest-Guided Gravitational Search Algorithm (GGSA) for enhancing the exploitation of the GSA using a low-cost method. Specifically, the GGSA preserves and utilizes the position of the best agent, which is known as the global best (gbest) solution, achieved so far to guide the movement of other agents towards the global optimal solution. This is done by adding an additional velocity component related to the gbest agent, which aids in preventing the agents from stagnation in local minima of the search space. Therefore, two advantages are gained from the above searching strategy. Firstly, unlike the procedure in the original GSA, the agent with the best fitness function, i.e. the gbest, obtained thus far is saved. Secondly, this gbest agent is used to accelerate the movement of the other agents towards the global solution of the optimization problem. In more details, this searching proposal is provided as follows:

$$v_i(t+1) = rand * v_i(t) + \acute{q}_1 * ac_i(t) + \acute{q}_2 * (gbest - x_i(t)), \quad (20)$$

where $v_i(t)$ is agent's i velocity at time t, $\acute{q}_1$ and $\acute{q}_2$ are accelerating coefficients, *rand* denotes a random number between [0, 1], $ac_i(t)$ is agent's i acceleration at time t, and *gbest* represents the best solution position

obtained thus far.

In the GGSA, all the agents are randomly initialized. Then, the gravitational force and the gravitational constant are obtained utilizing Equations (9) and (11), respectively. Next, Equation (17) is used to find the acceleration of each agent. After updating the position of the *gbest* achieved until the current iteration, Equation (20) is used to compute the velocity of each agent. Finally, Equation (19) is used to update the position of each agent. This process terminates by satisfying a certain stopping condition [27].

4.2 The Procedure of Applying the GGSA for Optimizing the SRNN Controller

In this work, the proposed SRNN-based MRAC controller is trained by the GGSA. Figure 3 shows the flowchart of the GGSA [27]. The following steps clarifies the procedure of the optimization method:

***Step* 1:** Specify the agents' number, the maximum number of iterations, and the coefficients of the gravitational constant, namely G_0 and δ.

***Step* 2:** Randomly generate an initial population of N agents within specific limits. Each of these agents is the complete modifiable weights of a single SRNN controller.

***Step* 3:** Set $t = 1$, where t is the iteration counter.

***Step* 4:** Determine each agent's cost function utilizing the Integral Square of Error (ISE) having the following expression:

(21)

$$\text{ISE}=0.5 \sum_{t=1}^{T} e^2(k),$$

Where $e(k)$ represents the control error between the reference signal and the actual system output at time sample k and T is the total number of time samples. Subsequently, the fitness of each agent is obtained as follows:

(22)

$$fitness = \frac{1}{ISE + \varepsilon}$$

where ε is a small number for evading the zero division.

***Step* 5:** For the first iteration, the agent with the largest fitness function is considered as the global best solution. However, for the remaining iterations, if an agent achieves a larger fitness function compared with the global best solution, this agent is assigned as the current global best solution.

***Step* 6:** Update the gravitational constant according to Equation (11) and find the mass of each agent utilizing Equation (13).

***Step* 7:** For each agent, obtain the gravitational force applied by the top *Kbest* solutions using Equation (9).

***Step* 8:** Determine the acceleration of each agent utilizing Equation (17).

***Step* 9:** Find the velocity of each agent utilizing Equation (20).

***Step* 10:** Update the position of each agent using Equation (19).

Step 11: When the maximum number of iterations is achieved, then the current *gbest* position represents the final optimized SRNN weights. Otherwise, set $t = t + 1$ and go back to Step 4.

5. Simulation Results

Several simulation tests and comparative studies have been conducted in this section to investigate the effectiveness of the proposed SRNN-based MRAC, as depicted in Figure 1. By adopting the training procedure described in Section 4.2, the GGSA was employed to optimize the parameters of the SRNN controller. The main parameters in the GGSA including number of iterations, initial gravitational constant, and decrease coefficient were set to 500, 10 and 10, respectively. Moreover, for all controlled systems, only six hidden layers were used for the SRNN structure. The above settings for the GGSA and the SRNN controller were satisfactory to guarantee accurate control results.

5.1 Normal Control Tests

The efficiency of the proposed SRNN-based MRAC controller is evaluated in controlling three different nonlinear systems including a nonlinear non-minimum phase system, a water bath temperature control system, and a nonlinear minimum phase system.

Case Study 1:

As a first case study to evaluate the control system, the SRNN controller is used to control a nonlinear non-minimum phase system described by the following discrete-time equation [30]:

$$y(k+1) = \frac{y(k)y(k-1)}{1+y^2(k)+y^2(k-1)} + u(k) + 1.5(k-1) \tag{23}$$

The SRNN-based MRAC, shown in Figure (1), is required to track the following reference input:

$$r(k) = 0.5sin\left(\frac{2\pi k}{150}\right) + 1.2sin\left(\frac{2\pi k}{250}\right) \tag{24}$$

The following formula is used as the reference model for the system,

$$y_m(k+1) = 0.6y_m(k) + r(k) \quad (25)$$

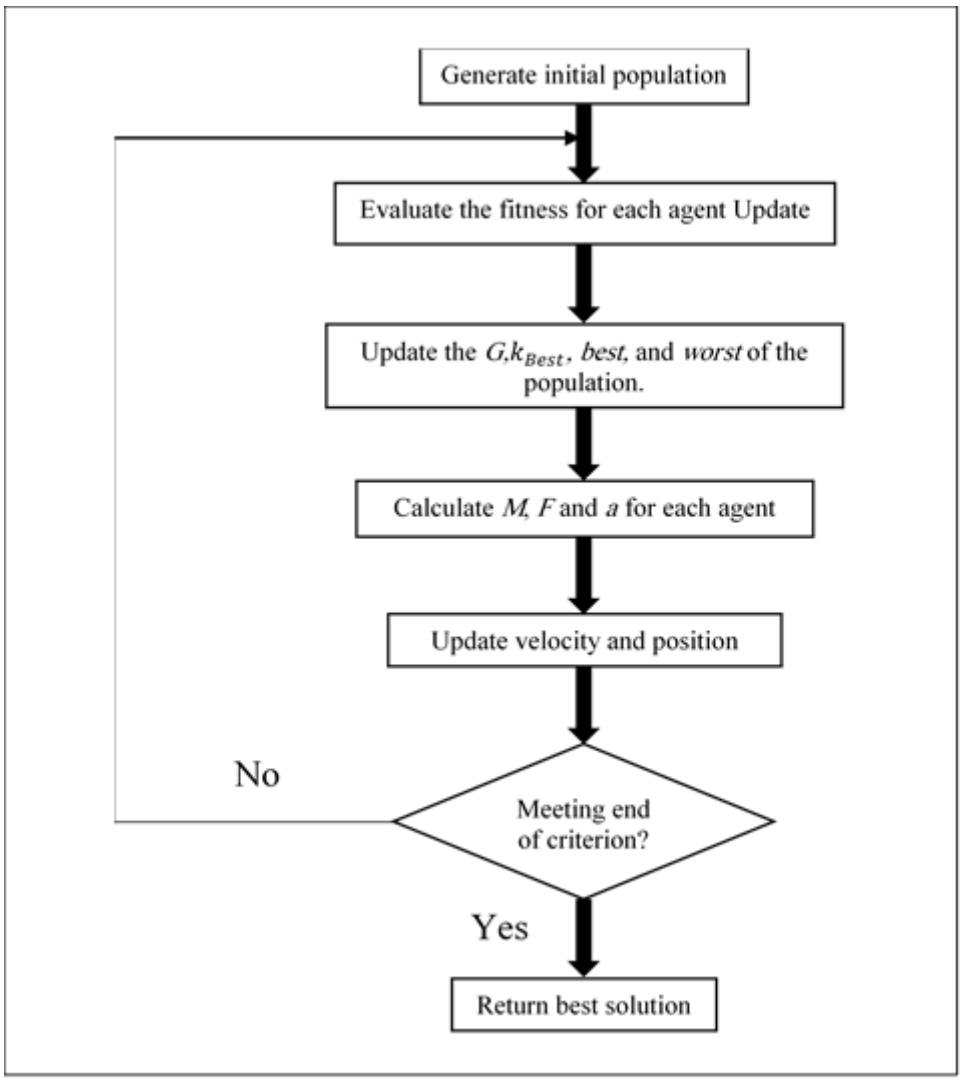
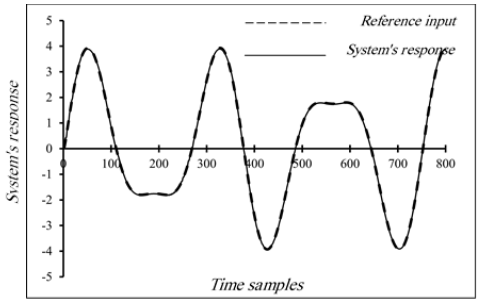
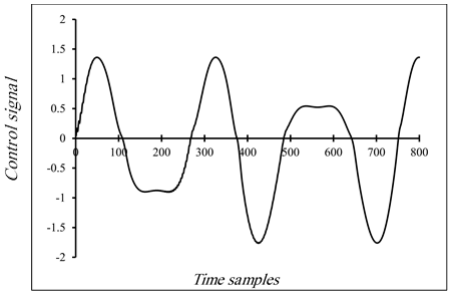


Figure 3: The Aflowchart of the GGSA

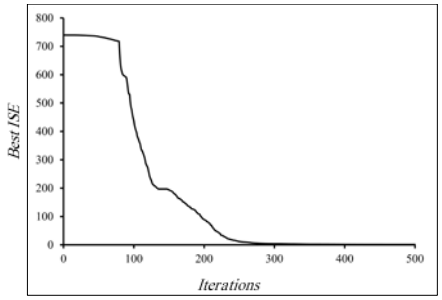
As a control objective, it is desired to force the system output to track the reference signal. Figure 4 illustrates the system response, the control action, and the best objective function against iterations. Specifically, Figure 4(a) demonstrates the remarkable capability of the suggested SRNN-based MRAC controller in controlling the nonlinear system model, where it is evident that the system response tracks the desired reference signal. The control signal can be clearly seen from Figure 4(b). Figure 4(c) illustrates the best objective function against iterations.



(a)



(b)



(c)

Figure 4: Case Study 1 (a) system's output and reference signal (b) control action (c) best cost function against iterations.

Case Study 2:

This is a nonlinear plant expressed by the following difference equation [30]:

$$y(k+1) = \frac{y(k) + y(k-1)}{1 + y^2(k) + 2y^2(k-1)} + u(k) \qquad (26)$$

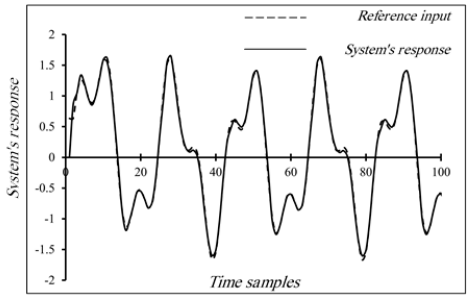
The reference signal is as follows:

$$r(k)=0.5[\sin(10\pi k) + \sin(25\pi k+0.5)] \qquad (27)$$

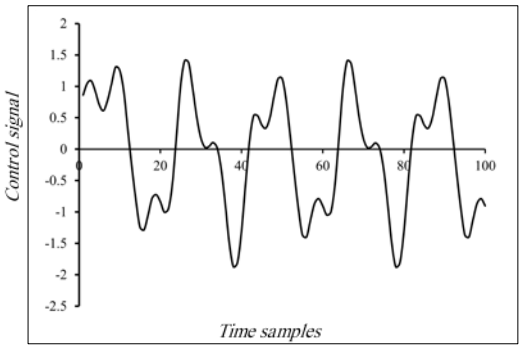
While the required system behavior is given by the following reference model equation:

$$y_m(k+1)=0.5y_m(k)+0.3y_m(k-1)+r(k) \quad (28)$$

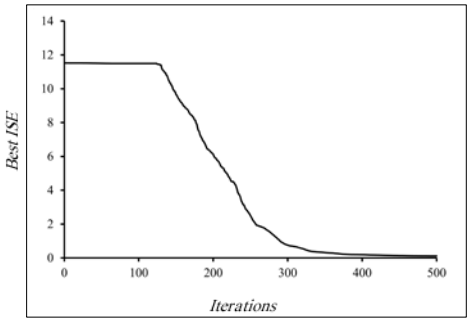
The simulation results of this case study are given in Figure (5). With a zero steady-state error and with no oscillations, Figure 5(a) shows that the SRNN-based MRAC controller has achieved a good performance in controlling this system. As for the control signal, it was indicated in Figure 5(b). Minimization of the 0.5ISE criterion is illustrated in Figure 5(c).



(a)



(b)



(c)

Figure 5: Case Study 2 (a) system's output and reference signal (b) control action (c) best cost function against iterations.

Case Study 3:

In this case study, the water bath temperature control system is considered with the following equation [31, 32]:

$$y(k+1) = a(T_s)\mathrm{y(k)} + \frac{b(T_s)}{1+e^{0.5y(k)-y}}u(k) + [1-a(T_s)]Y_0, \quad (29)$$

where $y(k)$ is the output temperature of the system in ℃, $u(k)$ is the control input, Y_0 represents the room temperature in ℃, $a(T_s) = e^{-\alpha T_s}$, and $b(T_s) = \frac{\beta}{\alpha}(1-e^{-\alpha T_s})$. Specifically, the following settings were used: α=1.00151 x 10^{-4}, β=8.67973 x 10^{-3},$\gamma = 40$, $Y_0 = 25\ ^{\circ}C$ and $T_s = 30s$. These values correspond to a real water bath system [32].

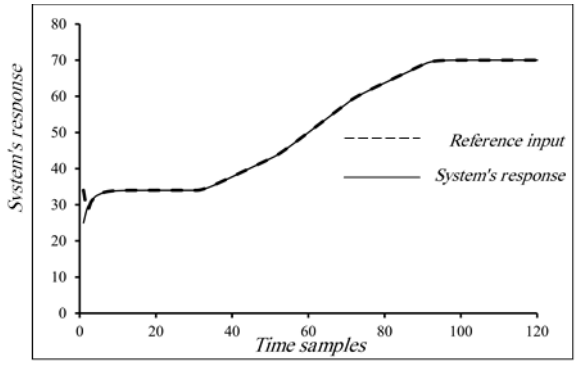
The desired water temperature in ℃ is described by the following signal:

$$r(k) = \begin{cases} 34 & k \le 30 \\ 34 + 0.4(k-30) & 30 \le k \le 50 \\ 44 + 0.8(k-50) & 50 \le k \le 70 \\ 60 + 0.5(k-70) & 70 \le k \le 90 \\ 70, & 90 \le k \le 120 \end{cases} \quad (30)$$

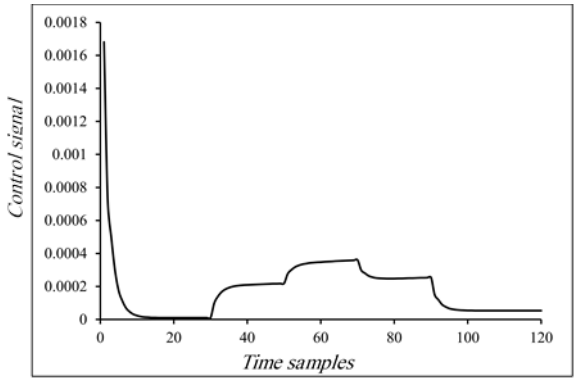
The following reference model was used [33]:

$$y_m(k+1) = 0.6y_m(k) + 0.4r(k) \quad (31)$$

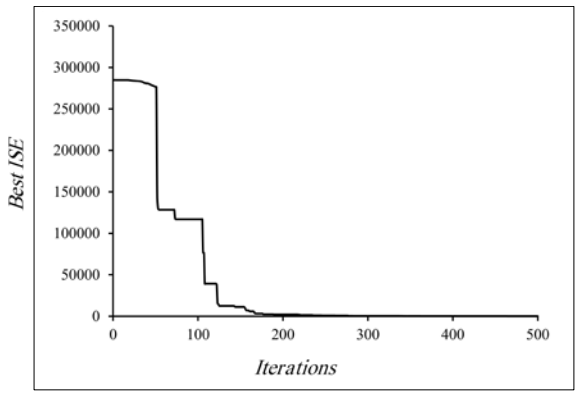
Figure 6 illustrates the output response, the control signal, and the best 0.5ISE against the iterations. In particular, Figure 6(a) illustrates the excellent performance of the SRNN-based MRAC controller, where it is obvious that the system response is practically identical to the reference signal. Figure 6(b) shows the control action, while the decrease in the 0.5ISE against 500 iterations is depicted in Figure 6(c).



(a)



(b)

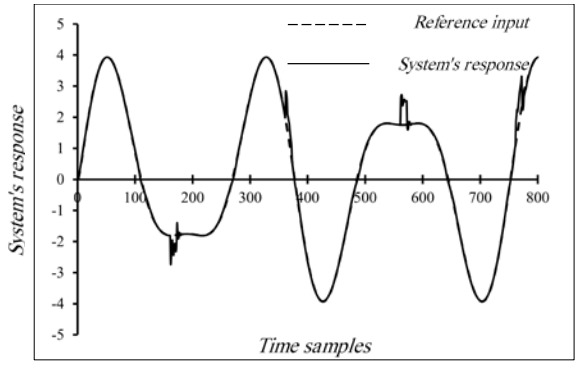


(c)

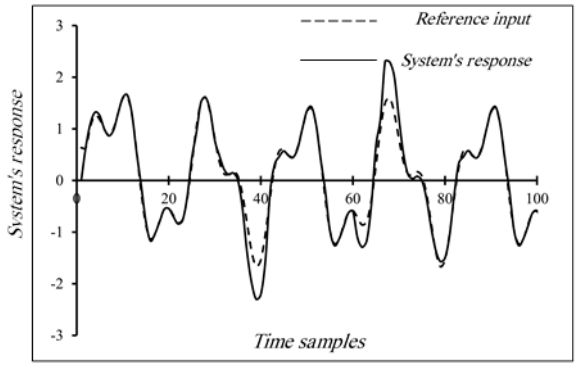
Figure 6: Case Study 3 (a) system's output and reference signal (b) control action (c) best cost function against iterations.

5.2 Robustness Tests

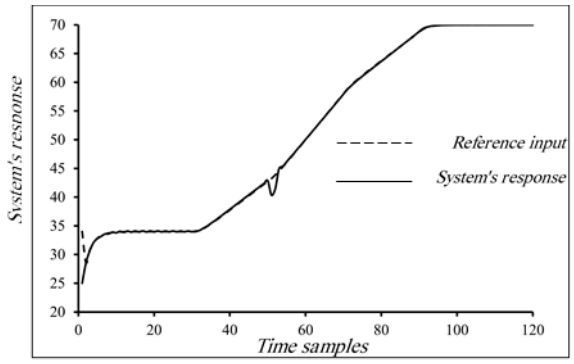
The aim of these tests is to examine the robustness of the SRNN-based MRAC system in handling unexpected external disturbances. Particularly, these tests were performed by injecting external disturbances of 50 percent of the controlled system response during only the testing phase. During different periods of the simulation time, these disturbances last a period of 10 samples. More precisely, these disturbances were encountered during the following intervals: "$161 \leq k \leq 171$, $361 \leq k \leq 371$, $561 \leq k \leq 571$ and $761 \leq k \leq 771$", for plant 1, and at the intervals $30 \leq k \leq 39$ and $60 \leq k \leq 69$, for Plant 3. For the water bath temperature control system, a disturbance of $-3℃$ was applied at the 50^{th} sample. As an important issue in these tests, the above disturbances have been only applied throughout the controller testing stage and not throughout the training stage, which further complicates the SRNN-based MRAC controller since it is not trained to deal with such unexpected disturbances. Figure 7(a), (b), and (c) clearly shows the robustness of the proposed controller in attenuating the disturbances applied for Plants 1, 2, and 3, respectively, where it is clear that the controller has done well both during and after the effect of each disturbance.



(a)



(b)



(c)

Figure 7: Robustness tests for (a) system 1 (b) system 2 (c) system 3.

5.3 Generalization Tests

These tests were made for demonstrating the generalization capability of the proposed controller by following testing signals that are completely different from the training signals. In this regard, the same training signals defined in Equations (24), (27) and (30) were used for Plants 1, 2 and 3, respectively.

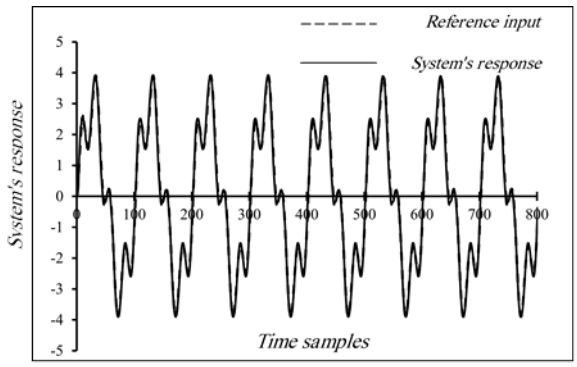
On the other hand, the controller testing phase was made by the following signals:

$$r(k)=0.5sin\left(\frac{2\pi k}{25}\right)+1.2sin\left(\frac{2\pi k}{100}\right) \quad (32)$$

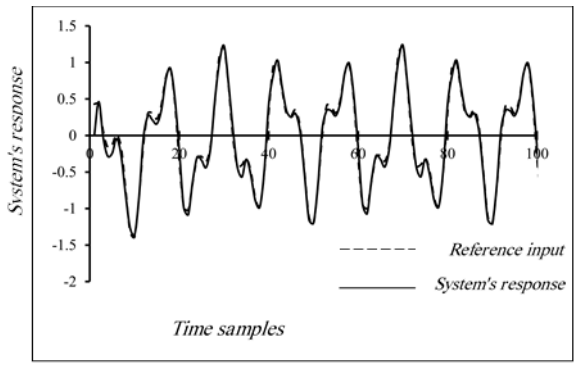
$$r(k)=0.5[cos(15\pi k)+cos(35\pi k+0.5)] \quad (33)$$

$$r(k)=\begin{cases}25, & k\le 20\\ 25+0.4(k-20) & 20\le k\le 40\\ 33 & 40\le k\le 50\\ 33-0.2(k-50) & 50\le k\le 70\\ 29 & 70\le k\le 80\\ 29+0.5(k-80) & 80\le k\le 100\\ 39 & 100\le k\le 120\end{cases} \quad (34)$$

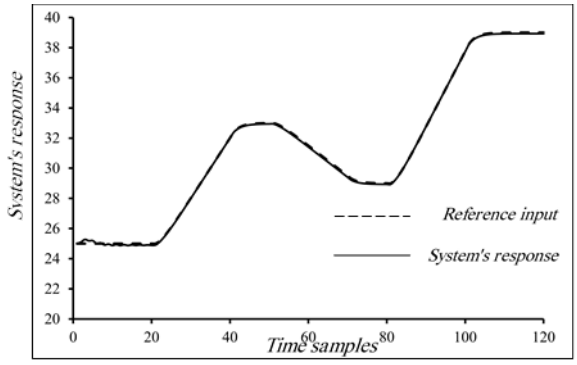
Equations (32), (33) and (34) represent the testing signals for Plants 1, 2 and 3, respectively. Figure 8(a), (b), and (c) illustrates generalization results of Plants 1, 2, and 3, respectively. From these results, it can be inferred that the SRNN-based MRAC successfully followed the testing signals, which were entirely unrelated to the training signals for all the plants.



(a)



(b)



(c)

Figure 8: Generalization tests for (a) Plant 1 (b) Plant 2 (c) Plant 3.

5.4 Comparing the Performance of the SRNN Controller with those of other Controllers

The performances of the proposed SRNN controller, the Modified Recurrent Network (MRN) controller, and the Multilayer Perceptron (MLP) controller are compared in this section in terms of control accuracy and execution time. The same optimization steps described in Section 4.2 were used for optimizing the parameters of each of the above networks to attain a fair and accurate comparative study.

Due to the utilization of several random operators by the GGSA, the final optimization results might slightly change in different runs. In order to deal with this issue, 10 runs have been conducted for each controller to account for the stochastic nature of the optimization method. The average of the 10 runs was then used to assess the result of each controller. As it is evident from Table 1, which summarizes the comparison results, the SRNN controller achieved a superior performance in comparison to the MRN and the MLP controllers. As an indication for the control accuracy, the SRNN controller produced the least values for the performance index compared to the other controllers. Moreover, as an indication for the processing speed, the SRNN controller took the shortest times among the times achieved by the other controllers.

5.5 Comparing the GGSA Optimization Performance with those of other Methods

This section is dedicated for comparing optimization results of the GGSA and the original GSA as the optimization methods for the proposed controller.

As was done in the previous section and for a fair comparison, 10 runs have been conducted for each optimization method and the average result has been taken. Obviously, Table 2, which illustrates the comparison results, demonstrates the advantage of using the GGSA algorithm compared to the original GSA. Specifically, the GGSA algorithm has accomplished the best control accuracy by producing the least ISE values for all the plants. Moreover, the GGSA has required the shortest processing times in comparison with the original GSA for all the plants.

Table 1: Comparison results of the MLP, the MRN, and the proposed SRNN.

NETWORK TYPE	CRITERION (AVERAGE OF TEN RUNS)	CONTROLLED PLANT		
		Plant 1	Plant 2	PLANT 3
MLP	ISE	12.68119	3.14045	6.36437
	Time (sec)	45.0728	13.1157	13.6983
MRN	ISE	2.9485	2.0952	3.17774
	Time (sec)	40.2345	12.5567	12.8119
SRNN	ISE	0.01912	0.079	0.46011
	TIME (SEC)	33.7519	9.9078	9.845

Table 2: Comparison results of the GSA and the GGSA in training the proposed SRNN controller.

NETWORK TYPE	CRITERION (AVERAGE OF TEN RUNS)	CONTROLLED PLANT		
		Plant 1	Plant 2	PLANT 3
GSA	ISE	40.6841	27.2595	31.1224
	Time (sec)	39.2053	10.0862	11.2893
GGSA	ISE	0.01912	0.079	0.46011
	TIME (SEC)	33.7519	9.9078	9.845

6. Conclusions

In this paper, a SRNN-based MRAC controller was proposed for controlling nonlinear dynamical systems. The structure of the SRNN is an enhanced version of a previously published MRN structure. The enhancement has been attained by using unity values for the weights connecting the context and the hidden layers in the original MRN structure. For the purpose of optimizing the weights of the proposed controller, the recently suggested GGSA was exploited. This training method has done well by reducing the ISE to the least values for all the considered plants. By controlling three different nonlinear systems, the results of an extensive assessment tests clearly indicates the efficiency of the proposed controller with regards to precise control, robustness ability, and generalization ability. Compared with other controllers, the SRNN structure has shown its superiority with regards to control performance and processing time. In addition, compared to the original GSA, the GGSA has achieved the best control precision and the shortest processing time.

References

[1] Y.-C. Cheng, W.-M. Qi, and W.-Y. Cai, "Dynamic properties of Elman and modified Elman neural network," Proceedings of the 1st Int. Conference on Machine Learning and Cybernetics, Beijing, PP. 637-640, 2002.

[2] A. I. Abdulkareem, "Identification of Dynamical Systems using Recurrent Neural networks with Structure Optimization Utilizing Genetic Algorithms," Engineering and Technology Journal, Vol.24, Part A, No.2, PP.129–139, 2005.

[3] L.C. Jain, and L.R. Medsker, "Recurrent Neural Networks: Design and Applications," CRC Press, 1st ed., London, 1999.

[4] N.A. Shiltagh, "A training algorithm for partial recurrent neural networks and its applications for system identification and control," Ph.D. Thesis, Control and Computer Eng. Dept., Univ. of Technology, Baghdad, Iraq, 2001.

[5] J.L. Elman, "Finding structure in time," Cognitive science, Vol. 14, PP. 179-211, 1990.

[6] D.T. Pham, and X. Liu, "Neural Networks for Identification, Prediction and Control," Springer-Verlag, 1st ed., London, Ch. 3, PP. 47-61, 1995.

[7] K.-K. Shyu, and H.-J. Shieh, S.-S. Fu, "Model Reference Adaptive Speed Control for Induction Motor Drive Using Neural Networks," IEEE Trans Ind Electron, Vol. 45, PP. 180-182. 1998.

[8] H.-W. Ge, et al., "Identification and control of nonlinear systems by a dissimilation particle swarm optimization-based Elman neural network," Nonlinear Analysis: Real World Applications, Vol. 9, PP. 1345-1360, 2008.

[9] A.Thammano, and P. Ruxpakawong, "Nonlinear dynamic system identification using recurrent neural network with multi-segment piecewise-linear connection weight," Memetic Computing, Vol. 2, PP. 273-282, 2010.

[10] J. Ma, and R. Zhang, "Model Reference Adaptive Neural Sliding Mode control for Aero-engine," AASRI Procedia, Vol. 3, PP. 508-514. 2012.

[11] C. Zhou, L.Y. Ding, and R. He,, "PSO-based Elman neural network model for predictive control of air chamber pressure in slurry shield tunneling under Yangtze River," Automation in Construction, Vol. 36, PP. 208-217, 2013.

[12] C.-H. Huang, "Modified neural network for dynamic control and operation of a hybrid generation systems," Journal of applied research and technology, Vol. 12, PP. 1154-1164. 2014.

[13] Y. Fang, J. Fei, and K. Ma, "Model reference adaptive sliding mode control using RBF neural network for active power filter," Electrical Power and Energy Systems, Vol. 73, PP. 249-258. 2015.

[14] M. Bahita, and K. Belarbi, "Model Reference Neural-Fuzzy Adaptive Control of the Concentration in a Chemical Reactor (CSTR)," IFAC-PapersOnLine, Vol. 49, PP. 158-162. 2016.

[15] O. F. Lutfy, and M. H. Dawood, "Model Reference Adaptive Control based on a Self-Recurrent Wavelet Neural Network Utilizing Micro Artificial Immune Systems," Al-Khwarizmi Engineering Journal, Vol. 13, PP. 107-122. 2017.

[16] X. Gao, X. Gao, and S. Ovaska, "A modified Elman neural network model with application to dynamical systems identification," IEEE International Conference in Systems, Man, and Cybernetics. Information Intelligence and Systems, Beijing, China, PP. 1376-1381, 1996.

[17] G. Ren, Y. Caoa, S. Wena, T. Huangc, and Z. Zeng, "A modified Elman neural network with a new learning rate scheme," Neurocomputing, Vol. 286, PP. 11-18, 2018.

[18] E. Rashedi, H. N. pour, and S. Saryazdi, "GSA: a gravitational search algorithm," Information sciences, Vol. 179, PP. 2232-2248, 2009.

[19] C. Li, N. Zhang, X. Lai, J. Zhou, and Y. Xu, "Design of a fractional-order PID controller for a pumped storage unit using a gravitational search algorithm based on the Cauchy and Gaussian mutation," Information Sciences, Vol. 396, PP. 162-181, 2017.

[20] S. Yazdani, H. N. pour, and S. Kamyab, "A gravitational search algorithm for multimodal optimization," Swarm and Evolutionary Computation, Vol.1, PP. 1-14, 2014.

[21] F. M. Ham, and I. Kostanic, "Principles of Neurocomputing for Science and Engineering," McGraw Hill, New York, NY, 2001.

[22] S. Li, J. Li, J. Qiu, H. Ji, and K. Zhu "Control design for arbitrary complex nonlinear discrete-time systems based on direct NNMRAC strategy," J Process Control, Vol. 21, No. 1, pp. 103-110, 2011.

[23] Y. Oussar, and G. Dreyfus," Initialization by selection for wavelet network training " Neurocomputing, Vol. 34, pp. 131-143, 2000.

[24] O. F. Lutfy, "Wavelet neural network model reference adaptive control trained by a modified artificial immune algorithm to control nonlinear systems," Arab. J. Sci. Eng., Vol. 39, pp. 4737-4751, 2014.

[25] S. Chong, S. Rui, L. Jie, Z. Xiaoming, T. Jun, S. Yunbo, L. Jun , and C. Huiliang, "Temperature drift modeling of MEMS gyroscope based on genetic-Elman neural network," Mechanical Systems And Signal Processing, Vol. 72, PP. 897-905, 2016.

[26] F.-J. Lin, L.-T. Teng, and H. Chu, "Modified Elman neural network controller with improved particle swarm optimisation for linear synchronous motor drive," IET Electric Power Applications, Vol. 2, PP. 201-214, 2008.

[27] S. Mirjalili, and A. Lewis, "Adaptive gbest-guided gravitational search algorithm," Neural Computing and Applications, Vol. 25, PP. 1569-1584, 2014.

[28] B. Yin, Z. Guo, Z. Liang, and X. Yue, "Improved gravitational search algorithm with crossover," Computers & Electrical Engineering, Vol. 66, PP. 505-516, 2017.

[29] S. Jiang, Y. Wang, and Z. Ji, "Convergence analysis and performance of an improved gravitational search algorithm," Applied Soft Computing, Vol. 24, PP. 363-384. 2014.

[30] A. Errachdi, and M. Benrejeb, "Model reference adaptive control based-on neural networks for nonlinear time-varying system," Proceedings of the International Conference on Systems, Control and Informatics, Italy, pp. 73-77, 2013.

[31] C. -F. Juang, and C. Lo, "Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm," Fuzzy Sets and Systems, Vol. 159, No. 21, pp. 2910-2926, 2008.

[32] C. -T. Lin, C. –H. Chen, and C. -J. Lin, "Nonlinear system control using functional-link-based neuro-fuzzy networks," In: Yu, W. (editor) Recent Advances in Intelligent Control Systems. Springer-Verlag London Limited, 2009.

[33] H. Husain, M. Khalid, and R. Yusof, "Direct model reference adaptive controller based-on neural-fuzzy techniques for nonlinear dynamical systems," American Journal of Applied Sciences, Vol. 5, No. 7, pp. 769-776, 2008.