Omar F. Lutfy
University of Technology,
Control and Systems
Engineering Department.
Baghdad, Iraq.
60157@uotechnology.edu.iq

Ahmed L. Jassim
University of Technology,
Control and Systems
Engineering Department.
Baghdad, Iraq.
cse.61120@uotechnology.edu.iq

Received on: 25/7/2018
Accepted on: 8/11/2018
Published online: 25/12/2018

A Simplified Recurrent Neural Network Trained by Gbest-Guided Gravitational Search Algorithm to Control Nonlinear Systems

Abstract- *This paper presents a feedback control strategy using a Simplified Recurrent Neural Network (SRNN) for nonlinear dynamical systems. As an enhancement for a previously reported modified recurrent network (MRN), the proposed SRNN structure is used as an intelligent Proportional-Integral-Derivative (PID)-like controller. 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 PID-like SRNN controller, three different nonlinear systems were considered as case studies, including a control valve, and a complex difference eq.. From an extensive set of evaluation tests, which includes a control performance test, a disturbance rejection test, and a generalization test, the proposed PID-like SRNN controller 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, genetic algorithm.*

How to cite this article: O.F. Lutfy and A.L. Jassim, "A Simplified Recurrent Neural Network Trained by Gbest-Guided Gravitational Search Algorithm to Control Nonlinear Systems," *Engineering and Technology Journal*, Vol. 36, Part A, No. 12, pp. 1290-1301, 2018.

1. Introduction

With the ability to comprehend 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. The most popular techniques for developing intelligent control systems include Artificial Neural Networks (ANNs), fuzzy logic, and Evolutionary Algorithms (EAs). In particular, owing 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. As such, recurrent neural networks (RNNs) have gained additional focus recently, particularly in 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,

https://doi.org/10.30684/etj.36.12A.11
2412-0758/University of Technology-Iraq, Baghdad, Iraq

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. Ji and Qi [7] proposed a proportional–integral–derivative (PID) MLEN which has two context layers to improve the approximation accuracy of the original MLEN. Ge et al. [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. Zhou et al. [10] 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 [11] suggested an intelligent controller which combines a radial basis function neural network and a MLEN to achieve maximum power point tracking in the power generation system.

It is worth to highlight that the above works used the general MELN structure which contains several sets of connection weights, which add 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 [12]. Nevertheless, these training techniques have slow convergence speed and they might easily trapped at local minima of the optimization problem [10, 13]. 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 et al. [14], is a population-based search algorithm which uses Newton's universal law of gravitation, mass interaction, and law of motion [15]. 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 [14, 16].

In the present work, to improve the approximation ability of the MRN proposed in [4], a Simplified Recurrent Neural Network (SRNN) is put forward. More precisely, the improvement was attained by adopting unity values for the weights that connect the context and the hidden layers in the original MRN structure. The proposed SRNN structure is used as a PID-like feedback controller 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. An overview of training the SRNN is given in Section 3. Basic concepts of the gravitational search algorithm (GSA), the GGSA, and the procedure of applying the latter are discussed in Section 4. In order to show the efficiency of the proposed PID-like SRNN controller, an extensive set of evaluation tests with two comparative studies are conducted in Section 5. Finally, a few remarks are provided in Section 6 to conclude the paper.

2. Background of Recurrent Neural Network

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

I. 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. These context nodes offer a limited recurrent architecture, hence the ELN is also called the simple recurrent network. 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 preserve the previous outputs of the hidden nodes at time k. In the original ELN, 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, 17,18].

Aiming at improving 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].

II. Simplified Recurrent Neural Network (SRNN) algorithm

The problem of finding suitable values of the gain 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). In fact, the MRN structure consists of a large number of parameters which might negatively affect the network approximation ability, since more parameters result in more uncertainty in their values. Therefore, it was proposed to utilize unity values for the weights that connect the context and the hidden layers in the original MRN structure. This proposed structure is called the simplified recurrent neural network (SRNN), which can be considered as a simplified version of the original MRN. The structure of the SRNN controller is depicted in Figure 1. Obviously, this figure shows that the SRNN structure comprises an input layer, a hidden layer, and an output layer. The task of each of these layers is explained below.

Layer 1: This layer is called the input layer and it consists of two parts, namely real inputs and context units. The real units transfer 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:

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

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 mission is to activate both the output and the context layers. The response of the m^{th} hidden node is expressed as follows:

$$h_m(k) = f\left[\sum_{c=1}^{C} m_c^{co}(k) + \sum_{i=1}^{n} w_{mi}^{hx} x_i(k)\right] \quad (2)$$

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 $i = 1, 2, \ldots, n$ and n represents the number of nodes in the input layer, *f(.)* is a nonlinear activation function, and w_{mi}^{hx} represents the weight between the i^{th} input node and the m^{th} hidden node.

Layer 3: The single node in this layer, known as the output layer, produces the output of the SRNN structure according to the following formula:

$$y(k) = \sum_{j=1}^{m} v_j h_j(k) \quad (3)$$

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

3.Training the SRNN controller

From the above discussion, it is evident that the SRNN has different modifiable weights, as given below:

$$S = [\beta_j \zeta_j w_{mi}^{hx} v_j] \quad (4)$$

For achieving the required SRNN performance, the optimal values for the weights in Eq. (4) must be obtained. To accomplish this objective, the GGSA is utilized in this work as the optimization method for the PID-like SRNN controller.

4.Gravitational Search Algorithm

To clarify the GSA optimization procedure, assume that there are *N* agents which are scattered in a given search space whose dimension is *D*. In this search space, each agent has a particular position, as given below:

$$x_i=[x_i^1, \ldots, x_i^d, \ldots, x_{i,}^D], \quad (5)$$

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

The top $Kbest$ agents apply a gravitational force which can be calculated using the following eq. [14]:

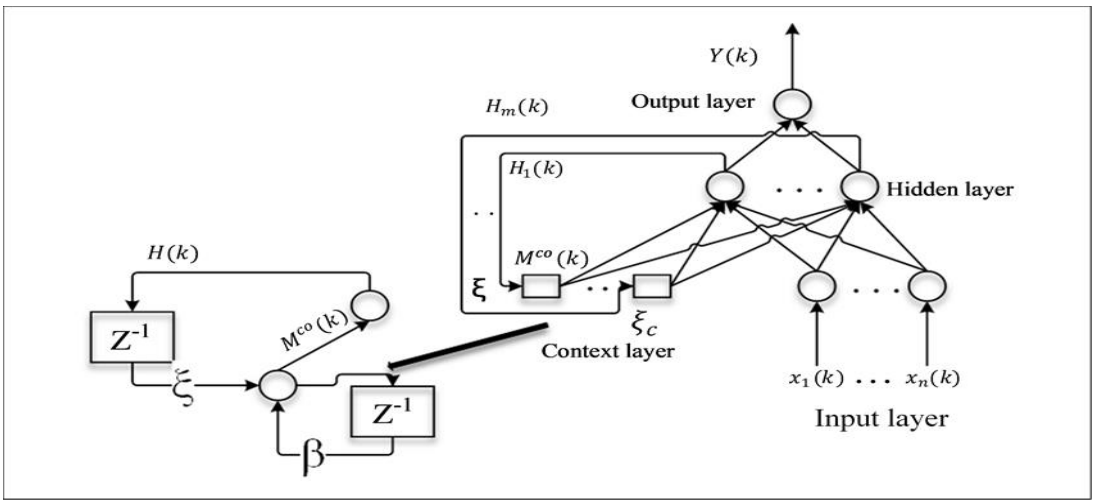


Figure 1: Structure of the SRNN controller.

$$(t)= \sum_{j \in Kbest,\, j \neq i} Rand_j \;.\; F_{ij}^{d}(t) \qquad (6)$$

$$F_{ij}^{d}(t)=G(t).\frac{M_i(t)\,.\,M_j(t)}{R_{ij}(t)+\varepsilon}\;.\;\left(x_j^d(t)\;-\;x_i^d(t)\right) \qquad (7)$$

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 Eq. (7) are obtained as given below:

$$G(t) = G_0 * exp(-\delta * \frac{L}{L_{max}}) \qquad (8)$$

$$R_{ij}(t)=\left\|x_i^d(t),x_j^d(t)\right\|_2 \qquad (9)$$

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:

$$M_i = \frac{\mathrm{S_i(t)}}{\sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{S_j(t)}} \qquad (10)$$

$$\mathrm{S_i(t)} = \frac{\mathrm{Fit_i(t)-\ worst(t)}}{\mathrm{best(t)\ -\ worst(t)}}, \qquad (11)$$

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 [14]:

$$worst(t)= \min_{j \in [1,\ldots,N]} Fit_j(t) \qquad (12)$$

$$best(t)= \max_{j \in [1,\ldots,N]} Fit_j(t) \qquad (13)$$

The next step is to compute the acceleration of each agent, as given below [14]:

$$a_{ij}^{d}=\frac{F_{ij}^{d}}{M_i} \qquad (14)$$

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

$$v_{ij}^{d}(t+1)=rand_i*v_{ij}^{d}(t)+a_{ij}^{d}(t) \qquad (15)$$

$$x_{ij}^{d}(t+1)=x_{ij}^{d}(t)+v_{ij}^{d}(t) \qquad (16)$$

I. 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 [14]. Nonetheless, several studies proved that the original GSA has a relatively slow and ineffective exploitation [19-21]. Therefore, the authors in [19] 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 preserve and utilize 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:

$$v_i(t+1) = rand * v_i(t) + \acute{q}_1 * ac_i(t) + \acute{q}_2 * (gbest - x_i(t)), \quad (17)$$

where $v_i(t)$ is agent's i velocity at time t, $\acute{q}_1$ and $\acute{q}_2$ are accelerating coefficients, *rand* denotes 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 Eq.s (6) and (8), respectively. Next, Eq. (14) is used to find the acceleration of each agent. After updating the position of the gbest achieved until the current iteration, Eq. (17) is used to compute the velocity of each agent. Finally, Eq. (16) is used to update the position of each agent. This process terminates by satisfying a certain stopping condition [19].

II. The Procedure of Applying the GGSA for Optimizing the SRNN Controller

In this work, the proposed PID-like SRNN controller is trained by the GGSA. Figure 2 shows the flowchart of the GGSA [19]. The following steps explain 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:

$$\text{ISE}=0.5 \sum_{t=1}^{T} e^2(k), \quad (18)$$

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:

$$fitness = \frac{1}{ISE+\varepsilon} \quad (19)$$

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 Eq. (8) and find the mass of each agent utilizing Eq. (10).

***Step* 7:** For each agent, obtain the gravitational force applied by the top *Kbest* solutions using Eq. (6).

***Step* 8:** Determine the acceleration of each agent utilizing Eq. (14).

***Step* 9:** Find the velocity of each agent utilizing Eq. (17).

***Step* 10:** Update the position of each agent using Eq. (16).

***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.

3. Simulation Results

Several simulation tests and comparative studies have been conducted in this section to investigate the effectiveness of the proposed PID-like SRNN structure to act as a feedback controller, as depicted in Figure 3. By adopting the training procedure described in Section 4 paragraph II, the GGSA was employed to optimize the parameters of the SRNN controller. 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. The above settings for the GGSA and the SRNN controller were sufficient to guarantee the desired control objective.

I. Normal Control Tests

The efficiency of the proposed SRNN controller is evaluated in controlling the following nonlinear dynamical systems. The input signal for all plants has the following definition:

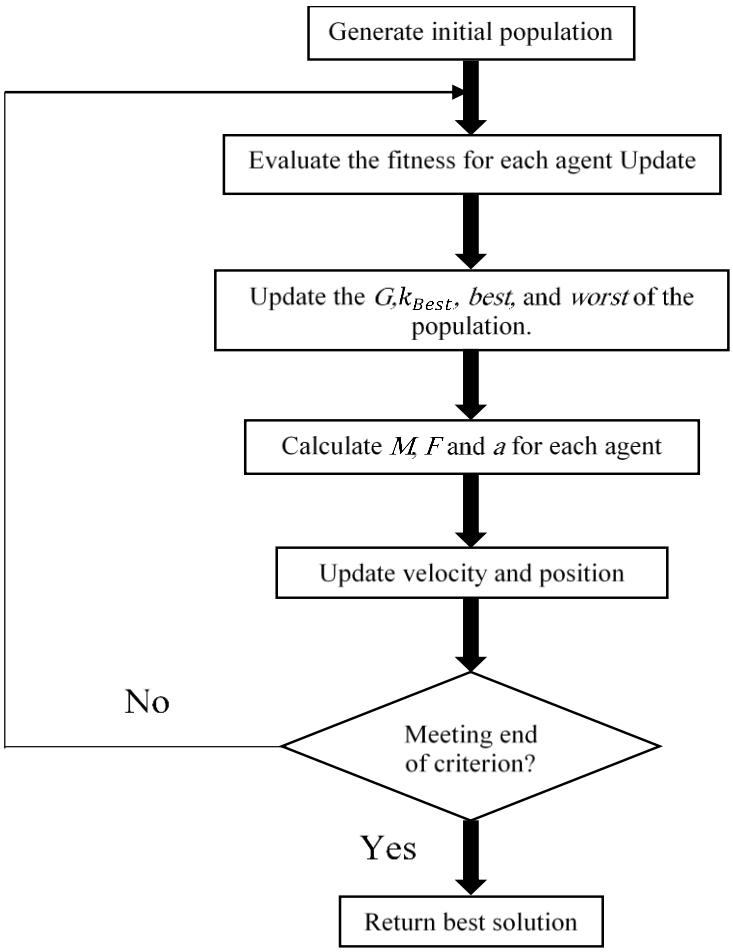


Figure 2: A flowchart of the GGSA [14]

$$r_{train}(k)=\begin{cases}0.6, & 0 \le k \le 100\\ 0.7, & 101 \le k \le 200\\ 0.8, & 201 \le k \le 300\\ 0.7, & 301 \le k \le 400\end{cases} \quad (20)$$

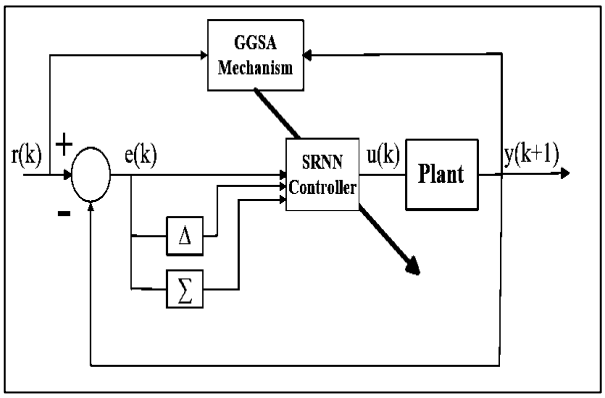


Figure 3: A schematic diagram for the control system, in which the SRNN is used as a feedback controller.

Case Study 1:

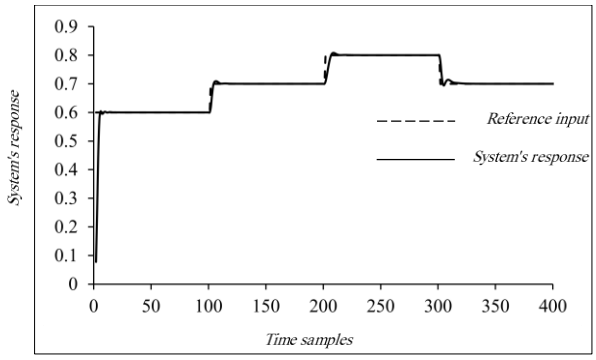
In this case study, the SRNN controller is used to control a valve representing an opening which has an adjustable area. This valve includes an actuator, a valve body, and a valve plug. The task of the actuator is to transform the control signal into a movement of the stem and valve plug. In particular, the following Wiener model is used to represent the dynamics of the control valve [22]:

$$x(k)=\frac{0.0616q^{-1}+0.0543q^{-2}}{1-1.5714q^{-1}+0.6873q^{-2}}u(k) \quad (21)$$

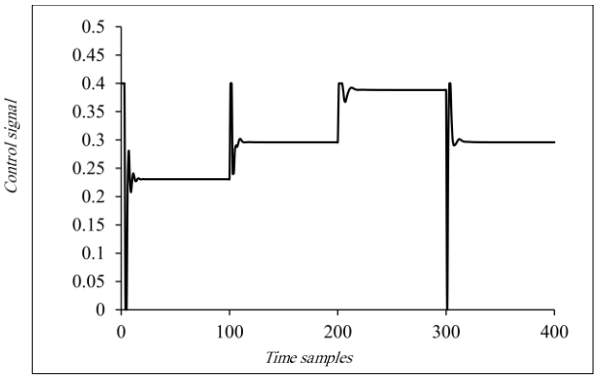
$$y(k)=\frac{x(k)}{\sqrt{0.10+0.90x^2(k)}} \quad (22)$$

where the control pressure, the stem position, and the flow through the valve are represented by the variables $u(k)$, $x(k)$ and $y(k)$, respectively. As a constraint in this control problem, the input to the valve should be limited within the range of [0, 0.4]. As a control objective, it is desired to force the system output to track the reference signal. Figure 3 illustrates the valve response, the control action signal, and the best objective function against iterations. Specifically, Figure 4(a)

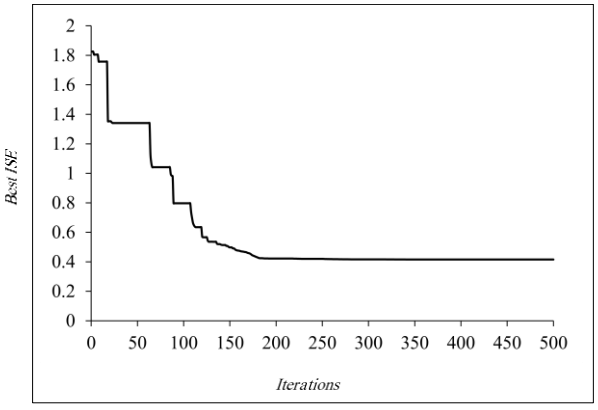
demonstrates the remarkable capability of the suggested SRNN controller in controlling the nonlinear valve model, where it is evident that the system response reaches the desired reference signal. The control signal was within the allowable range of [0, 0.4], as can be clearly seen from Figure 4(b). Figure 4(c) illustrates the best objective function against iterations.



(a)



(b)



(c)

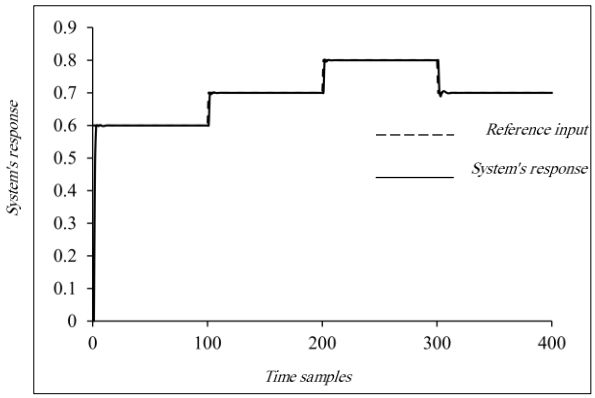
Figure 4: Control valve (a) system's output and reference signal (b) control action (c) best performance index against iterations.

Case Study 2:

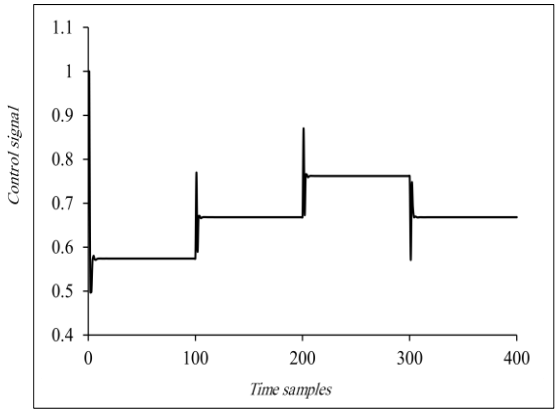
The Continuous Stirred Tank Reactor (CSTR), which is a highly nonlinear chemical process, is controlled by the SRNN controller in this case study. The following nonlinear difference equation is used in this work to represent the CSTR model [23]:

$$y(k+1)= 0.7653\,y(k)- 0.231\,y(k-1)+ 0.4801\,u(k) - 0.6407y^2(k)+ 1.014\,y(k-1)y(k) - 0.3921\,y^2(k-1)+ 0.592\,y(k)u(k)- 0.5611\,y(k-1)u(k) \quad (23)$$

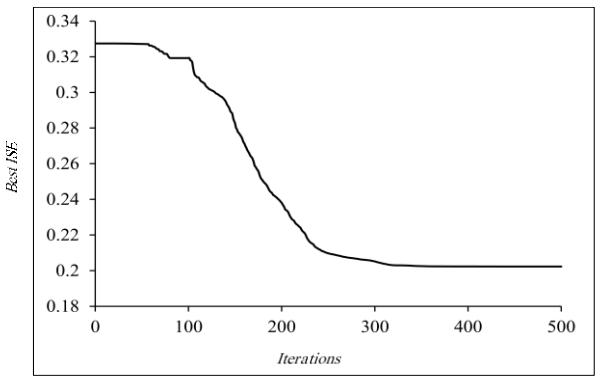
The simulation results of this case study are given in Figure (5). With a zero steady-state error and with no oscillations, Figure 5(a) shows that the SRNN controller has achieved a good performance in controlling the CSTR process. As for the control signal, it was within the allowable range of [-1, 1], as indicated in Figure 5(b). Minimization of the 0.5ISE criterion is illustrated in Figure 5(c).



(a)

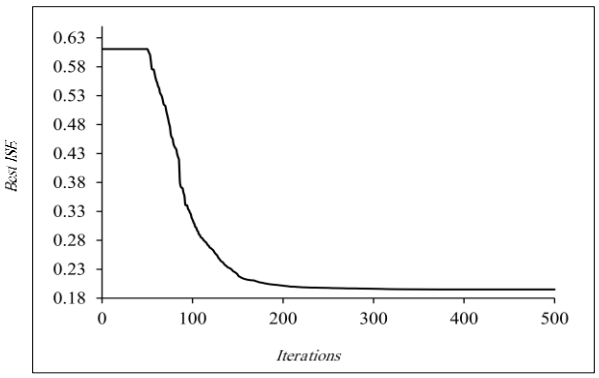


(b)



(c)

Figure 5: CSTR (a) system's output and reference signal (b) control action (c) best performance index against iterations.



(c)

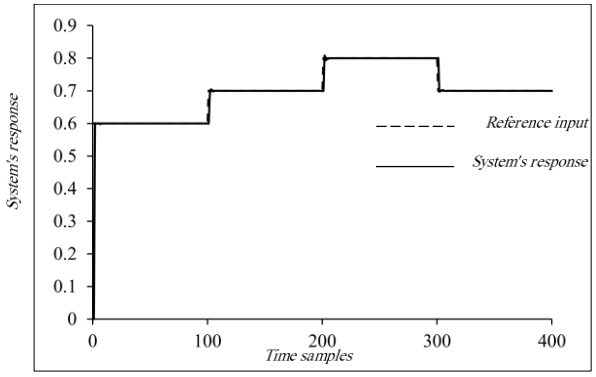
Figure 6: The nonlinear plant (a) system's output and reference signal (b) control action (c) best performance index against iterations.

Case Study 3:

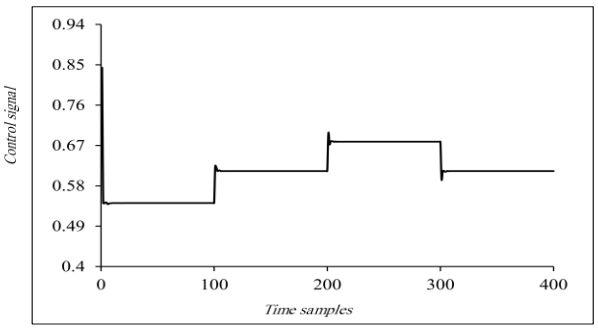
The system considered in this case study represents a nonlinear (in both output and input) plant. The following nonlinear discrete-time eq. is used for this plant [24]:

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k) \qquad (24)$$

Figure 6 illustrates the output response, the control action signal, and the best 0.5ISE against the iterations.



(a)



(b)

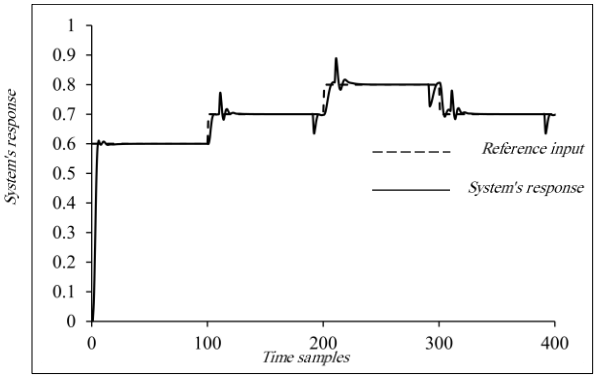
In particular, Figure 6(a) illustrates the excellent performance of the SRNN controller, where it is obvious that the system response is practically identical to the reference signal. Figure 6(b) shows the control action, while the decrease in the 0.5ISE against 500 iterations is depicted in Figure 6(c).

II. Robustness Tests

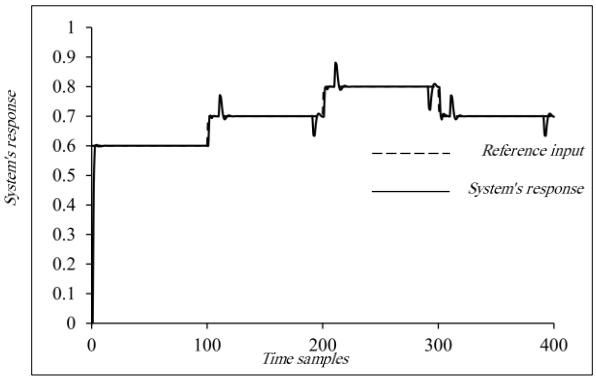
The purpose behind conducting these tests is for examining the robustness of the PID-like SRNN controller in handling unexpected external disturbances. Particularly, these tests were performed by injecting external disturbances of 10 percent of the controlled system response during only the testing phase. During different periods of the simulation time, these disturbances last a period of 80 samples. More precisely, these disturbances were encountered during the following intervals: "$110 \le k \le 190$, $210 \le k \le 290$ and $310 \le k \le 390$", for all the plants. 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 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.

III. Generalization Tests

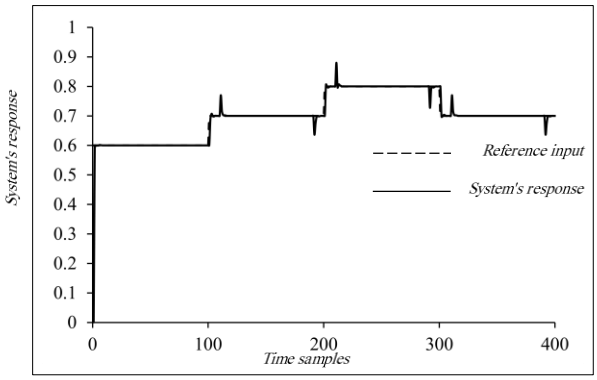
These tests are made for proving generalization capability of the proposed controller by following testing signals that are completely different from the training signals.



(a)



(b)



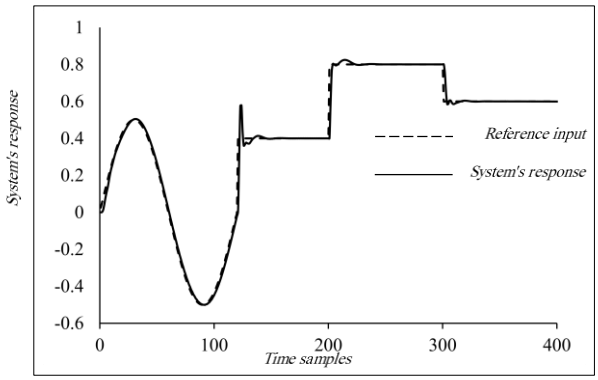
(c)

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

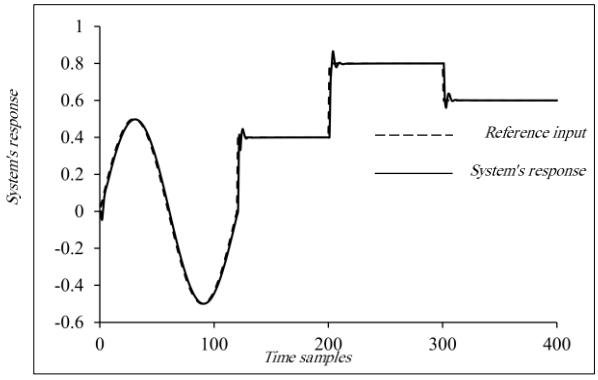
In this regard, the same training signal defined in Eq. (20) was used for all the plants.
On the other hand, the controller testing phase was made by the following signal:

$$r_{test}(k)=\begin{cases} 0.5\ \sin(\frac{2\pi k}{120}), & 0 \le k \le 120 \\ 0.4, & 121 \le k \le 200 \\ 0.8, & 201 \le k \le 300 \\ 0.6, & 301 \le k \le 400 \end{cases}$$

Figure 8(a), (b), and (c) illustrates generalization results of plants 1, 2, and 3, respectively. From this figure, one can infer that the PID-like SRNN controller successfully followed the testing signal, which was entirely unrelated to the training signal, for all plants under consideration.



(a)



(b)

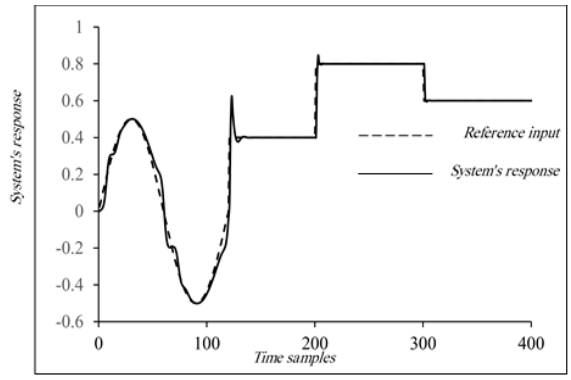


Figure 8: Generalization tests for (a) system 1 (b) system 2 (c) system 3.

IV. 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.

V. 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	0.79894	0.85882	1.15605
	Time	34.937	34.123	34.543
MRN	ISE	0.49347	0.77745	0.79704
	Time	32.168	32.284	40.381
SRNN	ISE	0.41583	0.20234	0.19533
	Time	19.643	19.282	18.496

Table 2: Comparison results of the GA, 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	1.21098	1.11727	1.42215
	Time	20.25	20.35	20.122
GGSA	ISE	0.41583	0.20234	0.19533
	Time	19.643	19.282	18.496

4.Conclusions

In this paper, a PID-like SRNN 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 and the GA, 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, "Dynamicproperties 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] Q.-l. Ji, and W.-M. Qi, "The property of PID Elman neural network and its application in identification of hydraulic unit," IEEE International Conference on Control and Automation, ICCA, Guangzhou, CHINA, PP. 1795-1798, 2007.

[8] H.-W. Ge, F. Qiana, Y.-C. Liang, W.-l. Du, L. Wang, "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] C. Zhou, L.Y. Ding, 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.

[11] 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.

[12] 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.

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

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

[15] C. Li, N. Zhang, X. Lai, J. Zhou, 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.

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

[17] S. Chong, S. Rui, L. Jie, Z. Xiaoming, T. Jun, S. Yunbo, L. Jun , 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.

[18] F.-J. Lin, L.-T. Teng, 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.

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

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

[21] 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.

[22] O. F. Lutfy, "Control of nonlinear systems utilizing a wavelet neural network controller optimized by micro artificial immune systems," The 2nd Engineering Conference of Control, Computers, and Mechatronics, Baghdad-Iraq, 2014.

[23] O. F. Lutfy, R. A. Majeed, "Internal Model Control Using a Self-Recurrent Wavelet Neural Network Trained by an Artificial Immune Technique for Nonlinear Systems," Engineering and Technology Journal, Vol. 36, Part A, No.7, pp.784–791, 2018.

[24] O. F. Lutfy, S. B. Mohd Noor, M. H. Marhaban, and K. A. Abbas, "A genetically trained adaptive neuro-fuzzy inference system network utilized as a proportional-integral-derivative-like feedback controller for non-linear systems," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, Vol. 223, pp.309-321, 2009.