Identification of Dynamical Systems Using Recurrent Neural Networks with Structure Optimization Utilizing Genetic Algorithms

Ahmed I. Abdual Kareem Al –Dulaimy

Received On: 9/8/2004
Accepted On: 5/1/2005

Abstract

This paper presents a genetic learning for training recurrent neural networks (RNNs) using series-parallel modeling scheme. All weights of these networks are adjusted simultaneously with the optimization of RNNs structure. The evolutionary technique is based on genetic algorithms (GAs) with real coding operators used as a mean for training the RNNs of variable structure (GAs are used for selecting an optimal number of hidden nodes for neuro-identifier, as well as training the network to minimize the error). The mean square error (MSE) function between the plant and the model output are optimized globally through generations of the genetic search with elitism and hybrid selection method. Due to the mechanism of a hybrid selection method, elitism strategy and real-coding operators, the GA can find the best model for a given plant early from the first generations. The significance of the proposed identification approach is illustrated with simulated different examples for linear and non-linear plants off-line.

تعريف نظم السيطرة للأنظمة الديناميكية باستخدام الشبكات العصبية المرتدة مع الحصول على الهيكلية المثلى للشبكة العصبية بالأعتماد على الخوارزميات الجينية

الخلاصة

في هذا البحث تم اعتماد التعليم الجيني لتعليم الشبكات العصبية المرتدة باستخدام نسق النمذجة التوالية المتوازية حيث تم تعديل كافة اوزان هذه الشبكات بنفس الوقت الذي تتم فيه تعريف الهيكلية المثلى لهذه الشبكات. وتم استخدام تقنيات النشوء الحاسوبية المعتمدة على الخوارزميات الجينية ذات معاملات الترميز بالقيم الحقيقية كوسيلة لغرض الحصول على القيمة المثلى لعدد الارتباطات في الطبقة المخفية للشبكة العصبية المرتدة علاوة على تعليم الشبكة، لتقليل الخطأ الناتج بين خرج المنظومة وخرج النموذج. وتم كذلك حساب القيمة المثلى لدالة معدل مربع الخطأ بين الخرج من العملية تحت التعريف و للنموذج خلال الأجيال من البحث الجيني المسند بطريقة اختيار مهجنة وحكم النخبة. وبسبب ميكانيكة الاختيار المهجنة وستراتيجية اختيار النخبة ومعاملات الترميز بالقيم الحقيقية فان الخوارزمية الجينية المقترحة يمكنها ايجاد افضل نموذج لعملية معينة مبكرا ومنذ الاجيال الاولى للتعليم الجيني. وتم اختبار أهمية التوجه لنظام التعريف المقترح بتمثيل منظومات خطية ولاخطية.

1. Introduction

The ability of the humans to be able to recognize and explain their environment is based on the capability of establishing relations between information and information units [1].

The brain's powerful capabilities in thinking, interpreting, remembering and problem-solving have led scientists to simulate the brain's functionally with very simplified computer models.

One result of such work is a computing approach different than the commonly known approach of sequential digital computing. It is referred to as **neural-network computing** or **artificial neural networks**.

By simulating some of the features of biological networks of neurons, artificial neural networks are able to analyze data for patterns, and then make predications on the basis of those patterns [2].

* Dept. of Control & Systems Eng., University of Technology, Baghdad-IRAQ

https://doi.org/10.30684/etj.24.2A.7
2412-0758/University of Technology-Iraq, Baghdad, Iraq
This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

Artificial neural networks represent an important area of research, which opens a variety of new possibilities in different fields including control systems engineering [3].

A neural network is a massively parallel-distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1-Knowledge is acquired by the network through a learning process.

2-Interneuron connection strengths, known as weights, are used to store the knowledge [3].

Artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks.

2.Identification and Optimization Methods

System identification can be divided into structure and parameter identification. Structure identification (modeling) is the process of finding the input variables of a function system followed by the determination of the input-output relation. The identification of the involved coefficients of the functional system is called parameter identification or estimation [4],[5].

The identification of a linear system is a mature field and there exist a large number of powerful algorithms for structure and parameter identification [6]. Unfortunately, most of those algorithms cannot be extended for the structure identification of nonlinear models without a considerable loss of generality. This shortcoming is caused by the large amount of possible relations between the input and output variables. It is evident that more powerful search and optimization algorithms are required to solve such complex tasks [7].

Many conventional optimization methods were introduced for parameter estimation. The suitability of a method depends on the quality of information contained in the data, the conceptual model structure and the application concerned [8].

Least square (LS) and recursive least squares (RLS) are basic methods for parameter estimation, but they have two constraints. The first is that the noise needs to be uncorrelated with the measurement of the dependent variable. The second constraint is that the quality of the estimates are shown to depend on the richness of the information contained in the data [9]. There are many other estimation methods, like the instrumental variable method, the maximum likelihood method, and the Koopmans-Levin method, but all these methods have drawbacks like long computation time and complex computation [5],[9].

The aim of this paper is to utilize GAs to find the dynamics of linear and nonlinear systems with emphasis on using two types of RNNs for modeling. This paper contains two fields .The first one is to use GAs as a learning algorithm to train the weights of RNNs of pre-defined structure and the second is utilizing GAs as an optimization technique to optimize the structure of two types of RNNs.

3. Neural Networks for Identification

Neural networks (NNs) offer some of the most versatile ways of modeling nonlinear processes of a diverse nature. A neural network attempts to mimic the function of the brain in a crude but simplistic manner. The nonlinear relationship between the input(s) and the output(s) is modeled using a number of basic blocks, called neurons or nodes. The nodes are interconnected and are usually arranged in multiple layers. Each inter-nodal link

or interconnection is weighted. At each node, the weighted inputs (from other nodes or from external inputs to the network) are summed together with an external bias known as the threshold, and the result is passed through a nonlinear function (also known as the activation function), which forms the output of the node [10].

Two basic architectures for NNs are the feedforward and the recurrent networks. Networks may also be designed combining the features of both. A feedforward neural network (FNN) has a multi-layered structure. The signals flow between the nodes only in the forward direction, i.e. towards the output end (see Fig. 1). The nodes of a non-input layer can have inputs from nodes of any of the earlier layers, so it is used with a one-step ahead prediction models of plants. In the recurrent networks architecture, the output from a node can flow in the forward direction (i.e. to nodes towards the output), or in the reverse direction, or may be feedback as an input to the node itself and is called self-feedback or self-connection. So, it is employed to predict several steps ahead into the future, then it will improve representation capabilities (see Fig. 2) [11][12]. In NNs applications, usually the nodal characteristics remain unchanged. The adjustment of the weights on the nodal interconnection and the thresholds is usually referred to as training or learning of the network. This is analogous to the estimation of parameters in an identification problem. The learning may be supervised or unsupervised. Unsupervised learning is based on maximization of some predefined function or criterion. Supervised learning expects operator intervention. It requires a training data set, comprising a set of input data and a corresponding set of data for the desired outputs and the learning is based on the minimization of the error between the computed and the desired outputs [9].

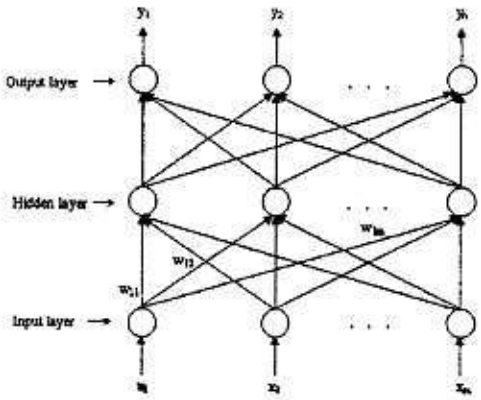

Fig (1) Feedforward Neural Networks

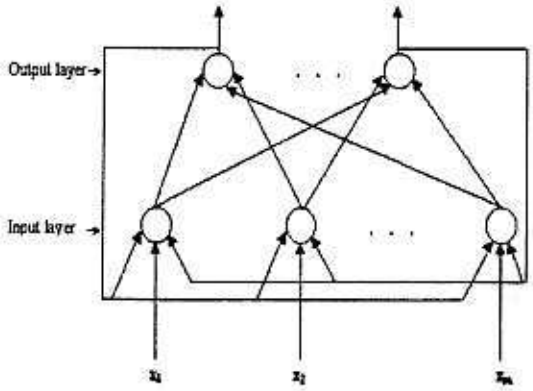

Fig (2) A Recurrent Neural Networks

3.1Identification of Dynamical Systems Using RNNs

As mentioned above, neural networks can be classified as FNNs and RNNs. The FNNs have been widely applied to dynamic system identification with success [13],[14]. The tapped–delay-line (TDL) method is usually adopted to enable a FNN to represent a dynamic system, because a FNN does not have dynamic memory. The method employs as inputs the current and past inputs and outputs of the system to be modeled by the network. The next output of the system is used as a teaching signal. However, there are several drawbacks associated with the TDL method. One of the drawbacks is its slow computation speed due to the large number of units in the input layer. This is because, if the order of the system to

be identified is unknown, it must be over-estimated and hence a large number of units in the input layer must be given to accommodate high system orders. The large number of units in the input layer also makes the identifier highly susceptible to external noise [3][6].

Due to their structure, RNNs do not suffer from the above drawbacks. Therefore the present work is based on the use of such networks, namely the modified Elman network and the Jordan network, as will be explained in the next sections.

Recurrent networks can be classified as fully and partially recurrent. Fully recurrent networks can have arbitrary feedforward and feedback connections, all of which are trainable like Hopfield network. In partially recurrent networks, the main network structure is feedforward. The feedforward connections can be trainable. The feedback connections are formed through a set of "context" units and are not trainable (i.e. fixed), if a BP learning algorithm is to be used. While by using GAs, all the connections in the network can be trainable [3],[6].

The context units keep memories of some past states of the hidden units, and so the outputs of the networks depend on an aggregate of the previous states and the current input. It is because of this property that partially recurrent networks possess the characteristic of a dynamic memory. Two types of recurrent networks have been shown below, the first is modified Elman network (used in this paper) shown in Fig. (3), and the second is the Jordan network as illustrated in Fig (4).

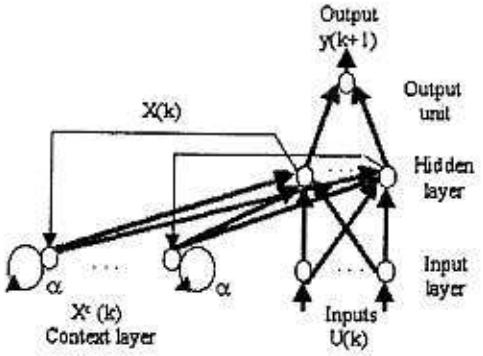

Fig (3) The Modified Elman Network

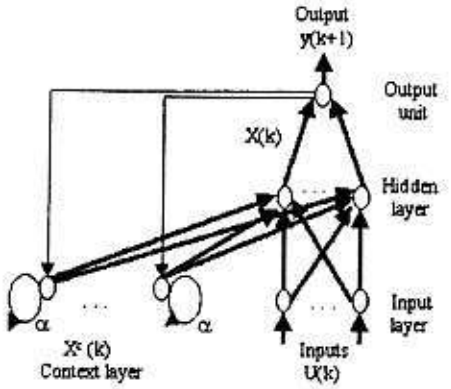

Fig (4) Jordan Network

This network have a multi- layered structure similar to the structure of MLPs. In this net, in addition to an ordinary hidden layer there is a context layer, this layer receives feedback signals from the ordinary hidden layer in the case of a modified Elman network.

The modified Elman network can be described by the following equations:

$$X(k) = W^{xc} X^{c}(k) + W^{xu} U(k)$$

$$X^{c}(k) = W^{cx} X(k-1) + \alpha X^{c}(k-1)$$

$$y(k) = W^{yx} X(k)$$

where,

$X(k)$ = output vector of hidden units.

$X^{c}(k)$ = output vector of context units.

$y(k)$ = output of the network.

$U(k)$ = input vector.

W^{xc} = weights matrix from context units to hidden units.

W^{xu} = weights matrix from input units to hidden units.

W^{yx} = weights vector from hidden units to output units.

W^{cx} = weights vector from hidden units to context units.

Referring to [6],[14],[15], one finds the series-parallel identification model as shown in Fig. (5) gives better approximation of the input-output function mapping compared to the parallel identification model shown in Fig. (6). When a parallel model is used, there is no guarantee that the identified model will converge to a given plant output and therefore; the output error (plant output yp minus RNN output ym to an input u) will not tend to zero [15]. For these reasons, the series-parallel model will be used in this paper.

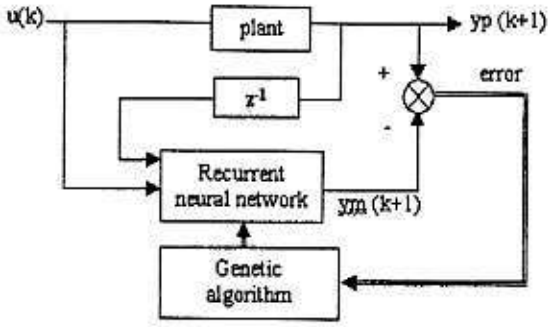

Fig (5) Series-Parallel Identification Model

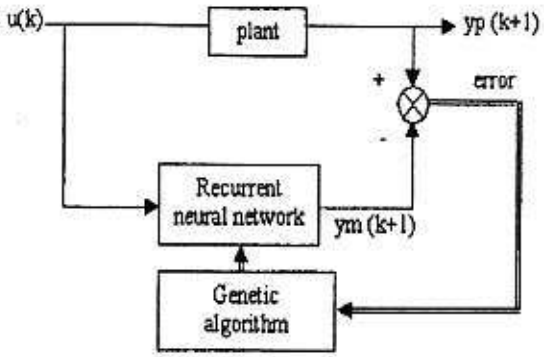

Fig (6) Parallel Identification Model

4. Evolutionary Computation

In the past few years, search and optimization algorithms inspired by biological evolution are introduced, and called evolutionary computation (EC) [7].

Evolutionary Computation (EC) methodologies are sometimes used in combinations with other methodologies; for example, Mitchell [16] described the use of evolutionary algorithm such as GAs to train NN. Instead of replacing the entire population-each generation, only one or two individuals were produced which then had to compete to be included in the new population. The network weights were represented as real, rather than binary numbers.

Typical paradigms of the EC include genetic algorithms (GAs), evolutionary strategies (ESs), evolutionary programming (EP), and genetic programming (GP). The scope of this work is to describe and focus only on GAs and see how the GA searches for solutions in the problem of identification of different linear and nonlinear systems using two types of RNNs.

GAs have proven to be very powerful search and optimization tools especially when only little is known about the underlying structure in the data. GAs are search algorithms based on the mechanics of natural selection and genetics. They employ operations similar to those found in natural genetics to guide their path through the search space. Essentially, they combine a survival the fittest optimization strategy with a structured yet randomized information exchange [17].

4.1 GA Operators

The genetic algorithms used involve the three types of operators: a hybrid selection method, single point crossover, and mutation [16].

The hybrid selection method is a robust strategy inspired from the modified simplex method, is to accept in the new population only those strings

that have better fitness values than the worst individual in the old population. This strategy is expected to ensure good guidance in the Complex and nonlinear search space ([6, 18]).

The crossover operation used in traditional GA is utilized successfully for real-coded GA.

Mutation can occur in a string with very small probability. In real-coded mutation, the random number between small upper and lower limits was added to the parameter itself (the connection weights).

Elitism:

In this operation the current fittest (best) individual in the population is copied directly to the next population without being changed by the other operations. This operator is sometimes used to try to make sure that there will be a reasonable fit individual present in the population at every time step. It helps to avoid having all the strings get modified by crossover and mutation in a way so that no good solution exits at some time [19].

4.2 Hybrid Neural/Genetic Learning

Genetic algorithms were first used instead of back propagation (BP) as a way of finding a good set of weights for a fixed set of connections by Montana and Davis [16]. Several problems associated with BP algorithm (e.g., the tendency to get stuck at local optima in weight space, or the unavailability of a "teacher" to supervise learning in some task), often make it desirable to find alternative weight training schemes.

The GA is used as follows; each chromosome is considered as a list (or "vector") of the total weights of neural networks.

Fig. (7) shows how the encoding is done for a sample of RNN. Each "gene" in the chromosome is a real number (this type of coding was used in this paper rather than bit). To calculate the fitness of a given chromosome, the weights in the chromosome are assigned to the links in the corresponding network, the network is run on the training set, and the mean square of error (*MSE*) was returned.

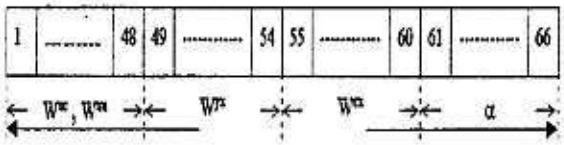

Fig (7) Chromosome Representation

4.3 Objective Function

The network is trained to provide the desired function approximation, where all its weights and the number of hidden nodes are adapted to minimize the mean square modeling error between the output of plant and the output of RNN over a batch of training patterns as:

$$MSE=\frac{\sum_{k=1}^{Np}[yp(k)-ym(k)]^2}{Np} \qquad (2)$$

where

MSE=mean square of error.

yp(k)= output of linear plant at sample *k*.

ym(k)=output of linear reference model at sample *k*.

Np= training patterns.

it is necessary therefore to map the objective function (*MSE*) to a fitness function. The most commonly used objective-to-fitness transformation is of the form [12]: -

$$\text{Fitness}=\frac{10}{1+MSE} \qquad (3)$$

5. Problem Description

Simulation examples are chosen to demonstrate the effect of using GA as a learning algorithm for recurrent neural

identifier of variable structure. Generally, the number of input and output nodes is fixed by definition of the problem. The number of hidden nodes and hidden layers, however, can be varied, and as it is known that, the number of network weights increased when the number of hidden nodes and hidden layers are increased [20].
In order to simplify the task of choosing the number of hidden nodes, GAs is well suited for this problem [16].

6. The Proposed Method

The following genetic procedure is introduced for training the neural network identifier for the plant:

Step 1:Initialize the genetic operators: crossover probability (Pc), mutation probability (Pm), population size (N_{pop}) and the maximum number of generations.
Step 2:Generate an initial population of the network at random. The number of hidden nodes and the initial connection density for each network is uniformly generated at random within certain ranges. The random initial weights are uniformly distributed inside a small range, typically between –1 and +1.
Step 3:First take the number of hidden nodes (in this work it is considered to be between 2 to 8 nodes) as an integer number from the chromosome of each population, then partially train each network on the training set to evaluate the objective function (*MSE*), and then calculate the fitness function as denoted in equation (3).
Step 4:Put in descending order all the chromosomes in the current population, (the first one is the fittest).
Step 5:Select individuals using hybrid selection method (Roulette Wheel plus deterministic selection).
Step 6:Stop if a maximum number of generations of GAs are achieved, otherwise increment the generations by one and go to **Step 3.**

7. Simulation Results

Simulation examples are chosen to demonstrate the effect of using GA as a learning algorithm for recurrent neural identifier of variable structure. Before describing the simulated examples, a brief relevant item of information concerning the simulations is given below:
1-In all the simulations carried out, a series-parallel modeling is used. For 3000 generations.
2- All hidden units are non-linear units with sigmoid function, while the other units are linear.
3-The real-coding genetic operators are used with N_{pop}=40, Pc = 0.85, Pm=0.05. The mutation process with real-valued parameters is implemented by summing the old parameter with randomly generated numbers between -0.5 and 0.5. A hybrid selection method [16] is used.
4-The following sequence of 1000 samples is chosen as a training signal to the RNN as random value between [-1,1]:

$u(k)=random[-1,1] \qquad 0 \le k \le 1000$

where k denotes the sampling instant.
5-The test signal is represented with the following input sequence:

$$u(k) = 0.2\ Sin\ (2\pi\ k / 50) \qquad 0 \le k \le 100$$

7.1 Example 1 [6]:

A nonlinear plant in output and in input with the following difference equation:

$$yp(k+1)=\frac{yp(k)}{1+yp^2(k)}+u^3(k) \qquad (4)$$

Is to be identified using RNNs identifier with series-parallel model.
The simulation results for this plant are shown in Fig (8 a, b, c, d), which show

the output of the plant with RNNs identifier, difference error between two outputs, the best *MSE* against the generations and the best-hidden node selection against the generations respectively.

7.2 Example 2 [21]:

In this example, the identification model scheme is applied to the gantry crane model.

Consider the block diagram of the gantry crane scale model shown in Fig. (9).

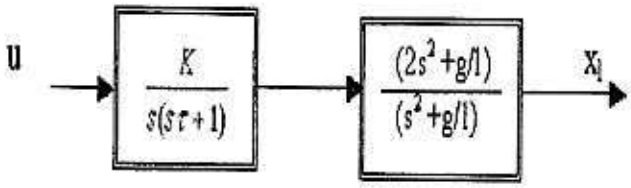

Fig (9) Block Diagram of The Gantry Crane System

The overall transfer function from actual horizontal load position x_l to the control voltage u is given in equation below:

$$\frac{x_l}{u} = \frac{K}{s(s\tau+1)}\left(1+\frac{s^2}{(s^2+g/l)}\right) \quad (5)$$

The approximate values of τ=0.065s, K=0.35, g=9.81 N/kg, l=1m. It is required that this crane to be identified using variable structure RNNs with minimum error.

A continuous time model representation is adopted to be numerically solved using the Runge Kuta fourth order method for the same performance index (*MSE*), the observation time T_{ob}=10s, and the simulation step size for this purpose h_s=0.01s.

The simulation results for this crane are shown in Fig. (10 a, b, c, d), which show the output of the crane with RNNs identifier, difference error between two outputs, the best *MSE* against the generations and the best-hidden node selection against the generations respectively.

8. Conclusions

In this paper a design method for the identification scheme using recurrent neural networks with evolutionary computation are introduced. The problem is formulated as a constrained optimization problem and a method based on evolutionary computation (GAs) with real codification and appropriate operators have been employed to solve the problem. In this way, the GA searches the proper identifier parameters (weights and structure of RNNs) in a reasonably large range. The proposed method can easily be extended to the design of other identification laws.

The following points are concluded from the work to solve different identification problems dealt within the proceeding examples.

1- It has been shown that RNNs are capable of representing linear and nonlinear plants models assuming no knowledge is available (about the orders or time delays) other than their input-output data sets (see for example figures (8a) and (10a)).

2- GA has the ability of starting with any values as initial weights for training RNN without any problem (e.g. local minimum and convergence failure) because GAs ensures broad converge over the entire search domain.

3- The results of RNN learning are not sensitive to the initial value of the weight vector, therefore GAs is employed to perform global search and to seek a good starting weight vector for subsequent neural network learning algorithm. The results are an improvement in the convergence speed of the algorithm.

4- GA can be used for optimizing RNN architecture. Only one parameter can define the whole structure of the RNNs

studied in this work (as given in Fig. (8d and 10d)). One limitation on the population size is noticed because of the computer memory. However, this limitation can be overcome with better computer hardware or language.

9. References

[1] Omatu S., Khalid M. and Yusof R.,- "Neuro-control and itsapplications"-Springer-Verlag,2nd,printing,1996.

[2]Banko Soucek-"Neural and concurrent real-time systems, the sixth generation"-Gohn Wiley and sons, 1989.

[3]Duc Truohg Pham and Liu Xing- "Neural networks for identification, prediction and control"-Springer-Verlag, 2nd,printing, 1995.

[4]Kalaba R. and Spingarn K.; "Control, Identification, and Input Optimization"; Plenum Press; 1982.

[5]Clausen S.T.; "System Identification and Robust Control"; Springer-Verlag; 1996.

[6] Al-Said I.A.M.; "Genetic Algorithms Based Intelligent Control"; Ph.D. thesis; Department of Control and Computer Engineering; University of Technology; January 2000.

[7] Ruan D.; "Intelligent Hybrid System: Fuzzy Logic, Neural Network, and Genetic Algorithms"; Kluwer Academic Publishers; 1997.

[8]Bubnicki Z.; "Identification of Control Plants"; PWN-Polish Scientific Publishers; 1980.

[9]Kanjilal P.P.; "Adaptive Prediction and Predictive Control"; Short Run Press Ltd.; 1995.

[10]Picton P.; "Neural Networks"; Palgrave Publishers; Second Edition; 2000.

[11] Lippmann R.P.; "An Introduction to Computing with Neural Nets"; IEEE ASSP Magazine; p.p. 4-21; April 1987.

[12] Chitra S.P.; "Use Neural Networks for Problem Solving"; Measurement and Control; p.p. 34-52; April 1993.

[13]Kadhim R.O.; "Design and Evaluation of Neuro-Identification and Controller for Nonlinear Dynamical System"; M.Sc. thesis; Nuclear Engineering Department; University of Baghdad; November 1999.

[14] Al-Araji A.S.A.; "A Neural Controller With A Pre-assigned Performance Index"; M.Sc. thesis; Department of Control and Computer Engineering; University of Technology; November 2000.

[15] Ismaeel S.A.R.; "Fuzzy-Neural Modelling of Dynamical Systems"; M.Sc. thesis; Department of Control and Computer Engineering; University of Technology; October 1997.

[16] Mitchell M.;" An Introduction to Genetic Algorithms"; 1st MIT Press Paperback Edition; Cambridge 1998.

[17] Goldberg D.E; "Genetic Algorithm in Search, Optimization and Machine Learning"; Addison-Wesley Publishing Company, Inc.; 1989.

[18] Naima F.M. and Al-Said I.A.M.;" Design of High Frequency OTA-C Filters Based on an Optimization Procedure";
المؤتمر العلمي الهندسي العراقي الحادي عشر (الهندسة الكهربائية)،بغداد-العراق ١٩٩٣

[19] Passino K.M. and Yurkovich S.;" Fuzzy Control"; Wesely Longman Inc. 1998.

[20] Rzempoluck E.J.;" Neural Network Data Analysis Using Simulnet"; Springer-Verlag New York; Inc; 1998.

[21]Butter H.;" Model Reference Adaptive Control; Prentic Hall International (UK) Ltd; 1992.

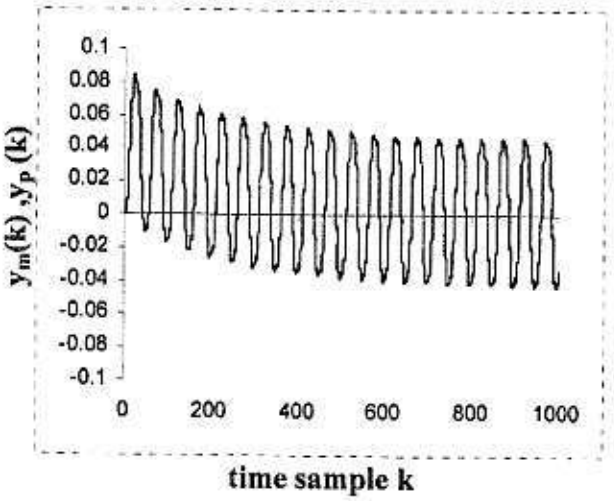

Fig (8) Simulation of Example 1: (a) Plant and Series-Parallel Model Outputs. (b) Difference Error between Two Outputs. (c) Best *MSE*. (d) Optimal Hidden Node Selection for RNNs.

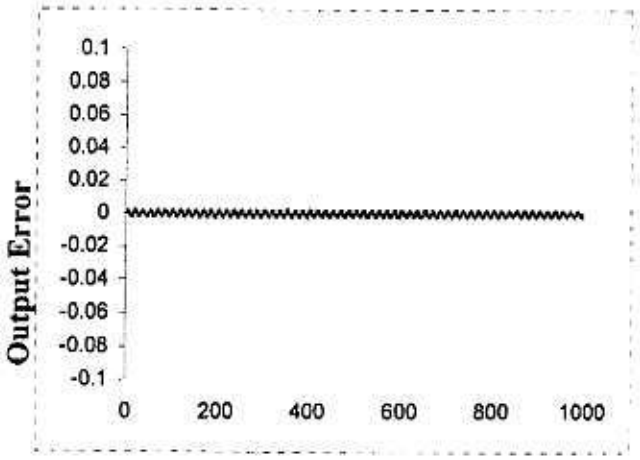

Fig (8 b) Cont.

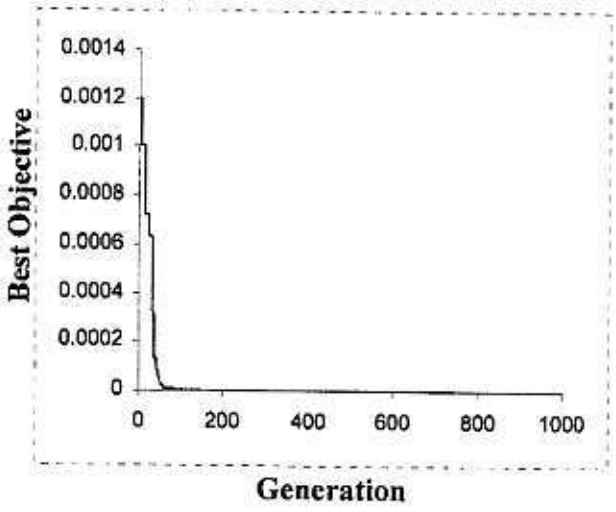

Fig (8 c) Cont.

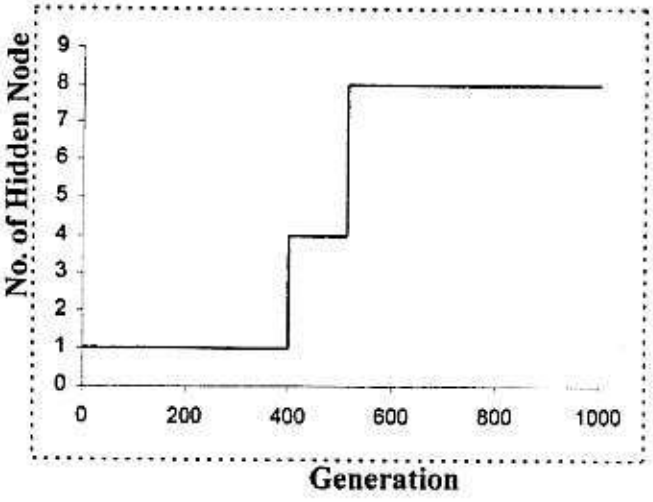

Fig (8 d) Cont.

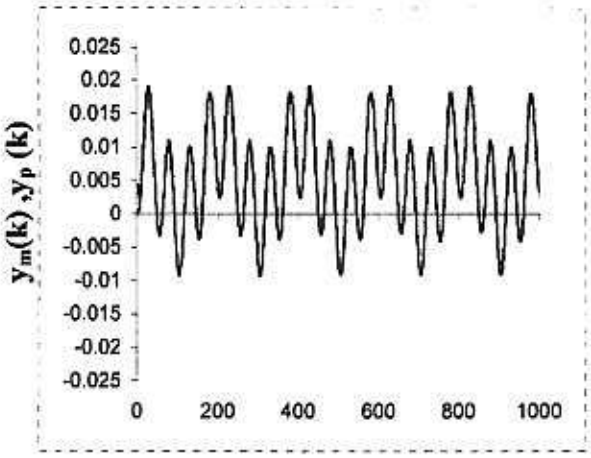

Fig (10) Simulation of Example 2: (a) Plant and Series-Parallel Model Outputs. (b) Difference Error between Two Outputs. (c) Best *MSE*. (d) Optimal Hidden Node Selection for RNNs.

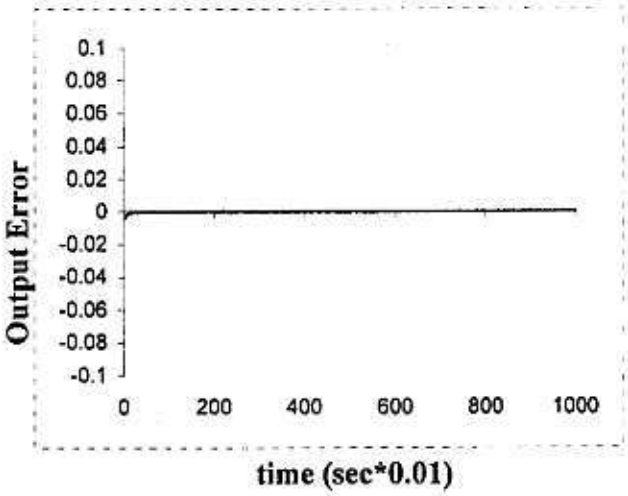

Fig (10 b) Cont.

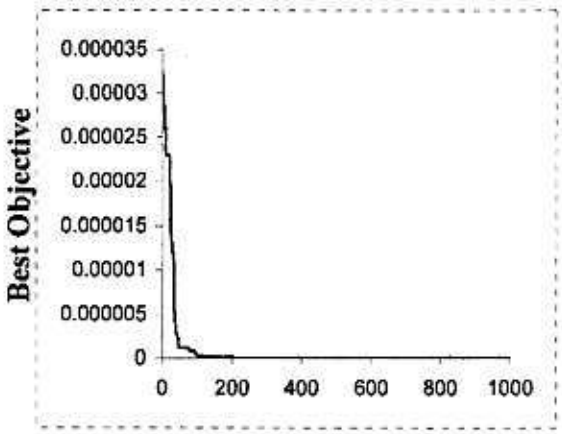

Fig (10 c) Cont.

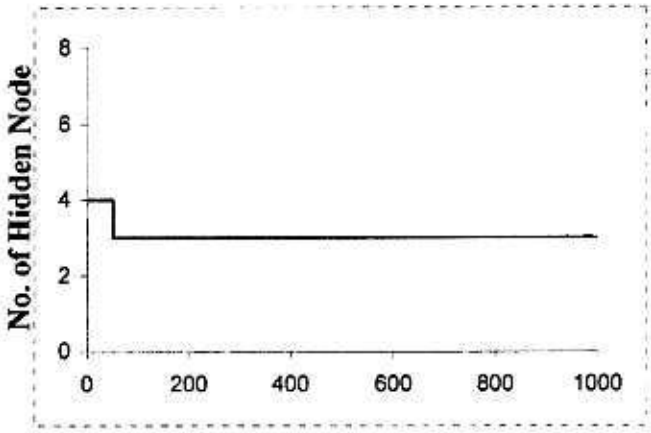

Fig (10 d) Cont.