ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

İnayet Özge AKSU

MULTIFEEDBACK-LAYER NEURAL NETWORK CONTROLLER DESIGN USING PARTICLE SWARM OPTIMIZATION ALGORITHM FOR HARD DISK DRIVE CONTROL

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2013

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MULTIFEEDBACK-LAYER NEURAL NETWORK CONTROLLER DESIGN USING PARTICLE SWARM OPTIMIZATION ALGORITHM FOR HARD DISK DRIVE CONTROL

İnayet Özge AKSU

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 25/06/2013.

.....	.....	.....
Assoc. Prof. Dr. Ramazan ÇOBAN	Prof. Dr. İlyas EKER	Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

ABSTRACT

MSc THESIS

MULTIFEEDBACK-LAYER NEURAL NETWORK CONTROLLER DESIGN USING PARTICLE SWARM OPTIMIZATION ALGORITHM FOR HARD DISK DRIVE CONTROL

İnayet Özge AKSU

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Assoc. Prof. Dr. Ramazan ÇOBAN
Year: 2013, Pages: 53
Jury : Assoc. Prof. Dr. Ramazan ÇOBAN
: Prof. Dr. İlyas EKER
: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ

In this paper, the weights of the Multifeedback-Layer Neural Network (MFLNN) which has recently proposed are trained by the Particle Swarm Optimization (PSO) algorithm. To improve training capability of the PSO, it is enhanced by some modifications. This method (MFLNN-PSO) is applied to two different problems to prove accomplishment of the method. Then, the closed loop identification of the reader head position of a disk drive system is proposed by using the MFLNN-PSO algorithm. Finally, a new type of neuro controller is put forward by using the MFLNN-PSO. Initially, this neuro controller is applied to two different kinds of dynamic systems. Later, it is applied to a hard disk drive system as a real physical example. Simulation results show that the MFLNN-PSO controller is effective and efficient on the control of dynamic systems and hard disk drive system.

Keywords: Hard Disk Driver, Identification, Multifeedback-Layer Neural Network, Nonlinear Controller, Particle Swarm Optimization.

ÖZ

YÜKSEK LİSANS TEZİ

HARD DISK SÜRÜCÜ KONTROLÜ IÇIN PARÇACIK SÜRÜSÜ OPTIMIZASYON ALGORITMASI KULLANILARAK ÇOK KATMANLI-GERI BESLEMELI SINIR AĞI YAPISINDA KONTROLÖR TASARIMI

İnayet Özge AKSU

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Danışman : Doç. Dr. Ramazan ÇOBAN
Yıl: 2013, Sayfa: 53
Jüri : Doç. Dr. Ramazan ÇOBAN
: Prof. Dr. İlyas EKER
: Doç. Dr. Zekeriya TÜFEKÇİ

Bu çalışmada, ilk olarak, yakın zamanda geliştirilen Çok Katmanlı-Geri Beslemeli Sinir Ağının (MFLNN) ağırlıkları, Parçacık Sürü Optimizasyonu (PSO) algoritması ile eğitilmiştir. PSO algoritmasının eğitme yeteneğini geliştirmek için, algoritmada bazı iyileştirmeler yapılmıştır. Yöntemin başarısını göstermek için bu metot (MFLNN-PSO) iki farklı probleme uygulanmıştır. Daha sonra, MFLNN-PSO algoritması ile bir disk sürücüsünün okuma kafasının konumlanması için tanılanmıştır. Son olarak, MFLNN-PSO kullanılarak yeni bir neuro kontrolör tasarlanmıştır. Bu neuro kontrolör iki farklı dinamik sisteme ve ayrıca gerçek bir fiziksel örnek olarak hard disk sürücü sistemine uygulanmıştır. Simülasyon sonuçları, MFLNN-PSO kontrolörün dinamik sistemler ve hard disk sürücü sisteminin kontrolü üzerinde etkin ve etkili olduğunu göstermektedir.

Anahtar Kelimeler: Hard Disk Sürücüsü, Tanımlama, Çok Katmanlı-Geri Beslemeli Sinir Ağı, Nonlineer Kontrolör, Parçacık Sürü Optimizasyonu.

ACKNOWLEDGEMENTS

I would like to thank my advisor Assoc. Prof. Dr. Ramazan ÇOBAN for his constructive criticism, unfailing guidance, encouragement, useful suggestions and his valuable time.

I would like to thank each and every member of the evaluation committee for their guidance.

My sincere thanks to my husband (Arif AKSU) for his patience, kindness and understanding during my graduate studies.

I give my greatest appreciation to my parents (Kemal HAKAN and Hamiyet HAKAN), my sisters (Gözde, Ezgi and Ece).

Finally, I would like to thank Nezahat GÜNENÇ TUNCEL for her support and encouragements for my studies.

CONTENTS **PAGE**

LIST OF TABLES **PAGE**

LIST OF FIGURES **PAGE**

LIST OF ABBREVIATIONS

PSO	: The Particle Swarm Optimization
PSO-FNNC	: The PSO Fuzzy Neural Network Control
TCP	: Transmission Control Protocol
HDD	: Hard Disk Driver
MFLNN	: The Multifeedback-Layer Neural Network
CLLRNN	: The Context Layered Locally Recurrent Neural Network
RNN	: The Recurrent Neural Network
LM	: Levenberg-Marquart Algorithm
FNN	: The Feedforward Neural Network
KR	: Keep Range coefficient
RMSE	: The Root Mean Square Error
N	: Number of pairs of the data
$y(k)$	: The output of the control system
$\hat{y}(k)$	: The reference output
$G(s)$	: Model of the plant
J	: Inertia of the arm
b	: Friction constant for the hard disk driver
K_a	: Amplifier Gain
R	: Armature Resistance
L	: Armature Inductance
K_m	: Motor constant value
MIMO	: Multiple-Input-Multiple-Output
TRFN-G	: The TSK-type Recurrent Fuzzy Network with Genetic Learning (Juang, 2002)

1. INTRODUCTION

Over the last few decades, neural approaches have been preffered to identify and control the nonlinear systems in engineering and industrial fields. It is seen that desired performances are obtained in different systems which are controlled by neural network controllers in Beyhan and Alcı (2010); Kim and Lewis (2000); Lewis et al. (1996); San et al. (2011). In Lin et al. (2004) and Lin and Wai (2003), a recurrent fuzzy neural network is used for control of the linear synchronous motor drive system. Besides, recurrent fuzzy neural networks are also succesfully used in controlling and identification of the dynamic systems (Lee and Teng, 2000; Mastorocostas and Theocharis, 2002). Some types of recurrent neural networks are applied to the nonlinear dynamic systems for controlling or identification in Chow and Fang (1998); Coban (2013) and Kim et al. (1997). Different computational optimization algorithms can be used in neural network controllers. The genetic algorithms among them are one of the methods used to building the controller structures. In the work of Juang (2002), a TSK-type recurrent fuzzy network with the genetic learning is applied to dynamic system control. The Particle Swarm Optimization (PSO) algorithm is also preffered for building the contol structure. In Dong et al. (2011), the PSO Fuzzy Neural Network Control (PSO-FNNC) is used for the ball and plate system. In that study, the PSO algorithm is used for the fuzzy neural network optimization. In Sheikhan et al. (2012), neural based controller is used for adaptive queue control in transmission control protocol (TCP) communication. In the study, the PSO algorithm is used for opitimization of the weights of the controller.

Hard disk drivers (HDDs) are the data storage medium and used in many areas, not only in computer systems but also in mobile communication industries as digital data storage. In recent years, the track density has been increased in the disk to develop the storage capacity. Thus, reader head must be correctly located on desired track and moved from one track to another. Because of this reason, position of the reader head comes into prominence.

Especially in mobile environment, positioning the reader/writer head is crucial and critical problem because of external shocks and vibrations. This situation reduces the performance of HDDs. Additionally, higher track density in hard disk drives necessitates the higher accuracy of track position in the mobile environment. The reducing positioning error of reader/writer head is significant issue in HDD design (Ren et al., 2009). For these reasons, a high performance control system is needed to reach the data effectively and precisely. Some methods are proposed in Atsumi and Messner (2013), Chen et al. (2006), and Pérez-Arancibia, Tsao, and Gibson (2010) to enhance the hard disk performance.

Neural networks are models that their neurons are connected to each other as a human brain. Since most physical systems are complex and non-linear, they may not be modeled by using conventional methods. For modeling such non-linear and complex problems, intelligent systems such as neural networks can be preferred. Also, neural networks are used in dynamic system control and identification. Identification and control of dynamic systems are so complicated process. Since output is a function of past-output or past-input or both of them, identification and control of dynamic systems are not comprehensible such as static systems. Also, when dynamic systems are used with long taped delay, input dimension excessively increase.

Overcome the input dimension increase, Elman (1990) and Jordan (1988) developed recurrent neural networks. In these networks, the delayed states of the hidden or output nodes copy to the hidden layer neurons with an extra set of context nodes. Hopfield (1985) and Tank (1986) proposed some neural networks to solve optimization problems. Pearlmutter (1989) developed a fully recurrent neural network which all nodes are connected to each other. In the Radial basis function network (Billings and Fung, 1995), past output values of the network are fed to input and output nodes. In (Sree Hari Rao and Yadaiah, 2005), a new approach was improved for artificial neural networks to identify parameters of dynamical systems. Coban (2013) suggested the Context Layered Locally Recurrent Neural Network (CLLRNN) to identify linear and nonlinear characteristics of the dynamic systems.

Savran (2007) proposed a new recurrent neural network (RNN), which is called the Multifeedback-Layer Neural Network (MFLNN).

The neural networks are the key structure for the designing and analyzing of the nonlinear control systems (Choi et al., 2001). In HDD control, Ge et al. (1998) and Lewis et al. (1999) proposed adaptive neural networks. In Ge et al. (1998) and Wang et al. (2001), a neural network controller with estimator is developed for the voice coil motor mass/inertia and a neural network control element to eliminate disturbances. In Plett (2003), it is shown that dynamic neural networks have significant benefits to prevent unknown disturbances. Mukhopadhyay and Narendra (1993) proposed a Multilayer neural network to reduce the adverse effects of the disturbances.

Recurrent neural network techniques are attractive alternatives to accomplish to control. In this study, the MFLNN proposed by Savran (2007) is used to control the reader head in a hard disk drive. The MFLNN is the very new version of the Recurrent Neural Networks (RNN). The MFLNN has three feedback layers for recurrence unlike the other RNNs which has simple feedback elements (Savran, 2007).

In this study, the Particle Swarm Optimization (PSO) algorithm is chosen for the MFLNN training. The derivative based algorithms such as the back-propagation and Levenberg-Marquart (LM) algorithms require desired values of the network' s output or Jacobian of the system during the training. Therefore, it is not suitable for controller design (Aksu and Coban, 2013) . To this end, The PSO is preferred to train the MFLNN. To improve training ability of the PSO, it is enhanced by some modifications.

In the literature, the usage of the PSO as the training algorithm of a neural network structure is very common. In Mendes et al. (2002), the Feedforward Neural Networks (FNN) is trained by the PSO and this system are applied to the classification and regression tasks. The results of the study are compared with the results which are obtained by using gradient-based algorithm. In Junyou (2007), the PSO algorithm is choosen to train the feedforward neural network. In the study, the success of the PSO is shown by comparing the results of back-propogation method

and the PSO. In this paper, the MFLNN is used for the dynamic systems whereas the other network structures are used for static systems. This shows that the MFLNN structure is more powerful. In this study, the MFLNN is trained with the PSO for the first time.

2. STRUCTURE OF THE MFLNN, THE PSO ALGORITHM AND THE HARD DISK DRIVER

Nowadays, neuro-control of the dynamic systems comes into prominence. To achieve this purpose, different types of neural controller structures are developed. In this study, a new type of neuro controller which is achieved by the MFLNN and the PSO is utilized for control of different dynamic systems and a disk drive system. In the following subsections, the MFLNN and as its training algorithm the PSO, and the HDD are presented, briefly.

2.1. The MFLNN

In this study, the MFLNN (Savran, 2007) is employed as a neuro-controller owing to the fact that it has nonlinear structure, good convergent property, and fast learning capability. The MFLNN is a type of recurrent neural networks which has four hidden layers (three of which are feedback layers), an input layer and an output layer. Structure of the MFLNN is depicted in Figure 2.1. A linear activation function is used as an activation function in the input layer. The hyperbolic tangent activation function is used in hidden layers and output layer. The input layer takes information from outside and transmits it to hidden layers with no alteration. The output layer takes information from hidden layers and sends it outside through the activation function. Whereas the input and output layers have physical links to outside, the hidden layers do not have.

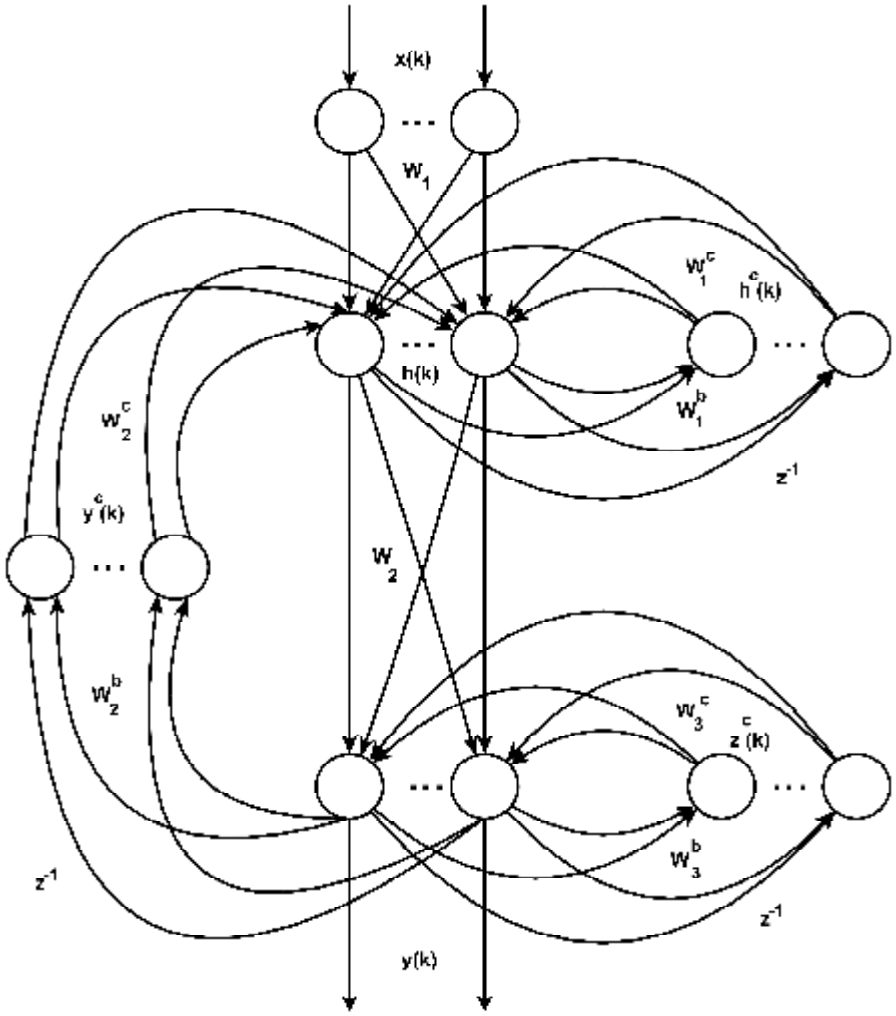


Figure 2.1. Structure of the MFLNN

Two local feedback layers and one global feedback layer are sited in the MFLNN structure (Savran, 2007). The local feedback layers provide self-recurrency for the output layer and the feedforward hidden layer while the other supplies a global recurrency from the output layer to the feedforward hidden layer. $x(k)$ is the input of the network and $y(k)$ is the output of the network. W_1 is the weight between the input and the hidden layer, whereas W_2 is the weight between the hidden and the output layer in the feedforward path. In the feedback and feedforward layers, weighted sums of the delayed outputs of the hidden and output layers enter the activation functions. W_1^b, W_2^b, and W_3^b denote the input weights of the feedback

layer neurons. W_1^c, W_2^c and W_3^c symbolize the output weights of the feedback layers. The information via these output weights is transmitted into the hidden and output layer neurons. $h^c(k)$, $y^c(k)$, and $z^c(k)$ are the outputs of the feedback layer neurons. One time delay is shown by z^{-1} as in the z- domain notation, the bias connections are not illustrated in the figure for the sake of simplicity.

Determination of the number of neurons in a hidden layer is extremely important. The number of the hidden-to-hidden layer neurons must be equal to number of the hidden layer neurons. Then, the number of the output-to-hidden layer neurons must be equal to the number of the output layer neurons. The number of the output-to-output neurons must be equal to the number of the output layer neurons. These values can be properly changed to increase the accuracy of the network.

For training the network, error value must be calculated when the training is performed from $k=1$ to $k=T$. The error value is calculated for each k in the following:

$$e(k)=y(k)-\hat{y}(k) \tag{2.1}$$

where $y(k)$ is the output of the MFLNN and $\hat{y}(k)$ is the desired output. Initial value of the hidden layer output (h) and the output layer output (y) is equal to zero.

$$\mathrm{h}(0)=0, \qquad \mathrm{y}(0)=0 \tag{2.2}$$

The input values which entered to the activation function of feedback neurons are

$$net_h^c(k)=\left[W_1^b h(k-1)\right]+B_1^b \tag{2.3}$$

$$net_y^c(k)=\left[W_2^b y(k-1)\right]+B_2^b \tag{2.4}$$

$$net_z^c(k) = \left[W_3^b y(k-1)\right] + B_3^b \tag{2.5}$$

W_1^b, W_2^b, W_3^b indicate the input weights of the feedback layers. B_1^b, B_2^b, B_3^b are the bias values of neurons of the feedback layers. The outputs of the feedback layer neurons are

$$h^c(k) = j_h^c(net_h^c(k)) \tag{2.6}$$

$$y^c(k) = j_y^c(net_y^c(k)) \tag{2.7}$$

$$z^c(k) = j_z^c(net_z^c(k)) \tag{2.8}$$

where j_h^c, j_y^c, and j_z^c represent the activation functions of the feedback layer neurons. $net_h(k)$ and $h(k)$ are the local fields of the hidden layer neurons and output of the hidden layers, respectively, and calculated by

$$net_h(k) = \left[W_1 x(k)\right] + \left[W_1^c h^c(k)\right] + \left[W_2^c y^c(k)\right] + B_1 \tag{2.9}$$

$$h(k) = j_h\left(net_h(k)\right) \tag{2.10}$$

where W_1 is the weight between the input and hidden layer, B_1 is the bias for hidden layer neuron, W_1^c and W_2^c represent the output weights of the feedback layers, and j_h activation function of hidden layer, $net_y(k)$ is the local field of output layer neurons, and y is the output of the output layer. They are computed in the following:

$$net_y(k) = \left[W_2 h(k)\right] + \left[W_3^c z^c(k)\right] + B_2 \tag{2.11}$$

$$y(k) = j_y\left(net_y(k)\right) \tag{2.12}$$

where W_2 is the weights between the hidden and output layer, B_2 is the bias for output layer neuron, j_y is the activation function of the output layer, and W_3^c is the output weight of the feedback layers. The feedback structure of the MFLNN may be altered according to the number of the feedback neurons and different types of functions can be used as activation functions.

The network input data can be normalized (as the case may be) before data is presented to the neural network. The formula, which the input data is linearly normalized with, is as follows:

$$X_{norm} = \frac{V_{\max} - V_{\min}}{X_{\max} - X_{\min}} X + \frac{V_{\min} X_{\max} - V_{\max} X_{\min}}{X_{\max} - X_{\min}} \tag{2.13}$$

where X_{norm} is normalized input data, $V_{\min}$ is the minimum of the normalization range, $V_{\max}$ is the maximum of the normalization range, X is input data, $X_{\min}$ is minimum value for input data and $X_{\max}$ maximum value for input data. Then, at the output of the neural network, the network output data is denormalized as follows:

$$Y_{denorm} = Y_{norm} - \frac{V_{\min} Y_{\max} - V_{\max} Y_{\min}}{Y_{\max} - Y_{\min}} \times \frac{Y_{\max} - Y_{\min}}{V_{\max} - V_{\min}} \tag{2.14}$$

where Y_{norm} is the output of the network which is normalized, $Y_{\min}$ is the minimum value for output data and $Y_{\max}$ is the maximum value for output data. In this study, $V_{\min}$ and $V_{\max}$ are chosen -1.0 and +1.0 , respectively.

The activation functions calculate the output of the neurons by the net information come from layers. In neural networks, the different activation functions can be used for neurons. The most fitting activation function is found by trying the

different functions. In this study, linear, sigmoid, and hyperbolic tangent activation functions given by Eqs. (2.15) - (2.17) respectively, are used.

$$y=x \tag{2.15}$$

$$y=\frac{1}{1+e^{-x}} \tag{2.16}$$

$$y=\tanh(x)=\frac{\sinh(x)}{\cosh(x)}=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.17}$$

Adjustment of the network weights is crucial process. The network weights are mostly determined by the back-propagation algorithm or the Levenberg-Marquart algorithm for system identification. In the system identification there exist both desired output information and network output information. Hence, the error between them can be easily utilized for error back-propagation. However, in the control scheme, no information about the desired value of the network output exists. Even if there exists it may be so much expensive to obtain such information.The derivative based algorithms such as the back-propagation and Levenberg- Marquart (LM) algorithms require such desired values of the network' s output or Jacobian of the system during the training. Therefore, it is not suitable for controller design (Aksu and Coban, 2013) . To this end, The PSO is preferred to train the MFLNN. To improve training ability of the PSO, it is enhanced by some modifications. In subsequent chapters, how the network weights are determined with the PSO algorithm is mentioned.

2.2. The PSO

Optimization is a process to find out the best solution within the alternative solutions of a specific problem under certain conditions. The PSO algorithm is optimization technique which uses intuitive methods. It is developed by inspiring

from birds and fish swarms. It is based on population and developed by Kennedy and Eberhart (1995).

The number of parameters in the PSO is less than the other population based optimization techniques, so the application of the PSO is simple and requires less memory. In Chen et al. (2013), the PSO is used for optimization of parameters of an intelligent multi-category classifiers. In Tsekouras (2013), designing of radial basis function networks with the PSO is examined. The timetabling problem is designed with the PSO in Tassopoulos and Beligiannis (2012). Furthermore, the PSO algorithm can be used in various engineering problems such as binary problems or highly complex, multimodal, nonlinear, noisy, or non-differentiable dynamic problems (Coban, 2011).

The PSO algorithm is based on following the bird which is closest the food. In the PSO, the birds and the bird flocks are named as particles and swarms, respectively. Algorithm is started with the population that consists of random solutions. So, the velocity and location of any particle are randomly assigned. During the training, each particle repositions its location in accordance with its flight experience and it' s neighbors' flight experience in the search space.

In the literature, there are two different types of neighborhood topologies: global neighborhood and local neighborhood. In the global neighborhood topology, all particles are neighborhood of each other. This approach converges fast, but there is a possibility of trap into the local optima. In the local neighborhood approach, a particle has limited neighborhoods. The number of neighborhoods affects the convergence rate. The local neighborhood topology has greater chance to fall into global optima (Kelemen et al., 2008). In this work the local neighborhood topology is preferred. The number of neighborhoods is selected %50.

Let the $pbest$ designate the best location of each particle and $gbest$ the best location that neighborhoods of each particle have. With respect to the values of $pbest$ and $gbest$, the optimal solution is sought as updating the velocities and positions of the particles in the following:

$$v_{id} = w \times v_{id} + c_1 \times r_1 \times \left(p_{id} - x_{id} \right) + c_2 \times r_2 \times \left(p_{gd} - x_{id} \right) \tag{2.18}$$

$$x_{id} = x_{id} + v_{id} \tag{2.19}$$

where d is the index of dimension of the search space, i is the index of the particle in the swarm, r_1 and r_2 are the random numbers which are uniform distribution in the range [0,1]. c_1 and c_2 are positive coefficients and w is inertia value. v_{id} and x_{id} are the particles velocity and the particles current position, respectively. p_{id} is the position of the *pbest* and p_{gd} is the position of the *gbest* which is the best solution of the neighbors.

Determination of c_1, c_2, and w substantially affects the performance of the PSO. c_1 and c_2 are stochastic acceleration coefficients. c_1 provides that the particle moves along the *pbest* direction, while c_2 ensures that the particle moves along the *gbest* direction. The values of the c_1 and c_2 are calculated as follows:

$$c_1 = c_{1\max} - \frac{\left(c_{1\max} - c_{1\min} \right) k}{K} \tag{2.20}$$

$$c_2 = c_{1\min} + c_{1\max} - c_1 \tag{2.21}$$

where $c_{1\min}$ and $c_{1\max}$ are the minimum and maximum values for c_1, respectively. k is the iteration number and K is the number of total iterations. In this study, c_1 and c_2 values are choosen as 1.5 and 2.5, respectively, following some trial and errors. The inertia weight factor w provides balance between local and global search. Smaller value for w indicates the local search and larger value indicates the global search, contrarily.

In this study, linearly decreasing weight is used for inertia value. Initially, global search is enabled by using a large inertia value. Towards the end of the run, local search is enabled with small inertia value (Shi and Eberhart, 1999).

$$w_k = w_{\max} - \frac{(w_{\max} - w_{\min})k}{K} \tag{2.22}$$

where $w_{\max}$ is the maximum value of inertia weights, $w_{\min}$ is the minimum value of inertia weights, k is the iteration number and K is maximum iterations number. In this study, $w_{\max}$ and $w_{\min}$ are selected 0.9 and 0.4, respectively.

Since swarm size affects the performance of the PSO it must be determined formerly. The desired solution may be obtained with larger swarm size. In contrast, the use of a very large number of particles can increase the complexity of the calculation. This situation requires more time than the process of finding the optimal solution. In most of the studies, the swarm size is prefered between 20 and 60. Here, the swarm size is taken 60 after a number of trials is attempted.

In this study, the particle swarm optimization algorithm is improved. The improvements made are as follows:

- to improve the performance of the PSO, the particle which has the worst position is found and changed with the best position in the swarm during training.
- the coefficient c_3 is inserted into the Equation (2.19) for updating the particle location given by

$$x_{id} = x_{id} + c_3 \times v_{id} \tag{2.23}$$

The value of the coefficient c_3 is chosen in the range [0.1, 0.15], randomly. Equation (2.23) is a modified version of Equation (2.19).

- to enhance the success of the algorithm, the mutation operation is added to algorithm. The main purpose of the mutation is to prevent the problem solutions from falling to a local value during the iterations. The mutation generally:

    - avoids that solutions pass exactly same to next iterations after a certain stage during the iterations.
    - avoids solutions to catch local optima.
    - is a method which is used to obtain the desired solutions more quickly.

  Here, the uniform mutation with a predifined ratio (probability) is chosen. According to the predifined ratio the uniform mutation sums up the value of the chosen particle with a uniform random value within the parameter' s range. In this study, as the mutation ratio is increased by 0.0001 at each iteration, the possibility of falling into local best is avoided. Besides, on the mutation stage, not only position but also velocity is subjected to the mutation. When the mutation ratio is provided, position and velocity values of particle are increased by the random value which obtained in the range $(KR \times [x_{\min}, x_{\max}])$, where KR stands for a positive constant.

- when the network coefficients assume larger values in the range [-10, 10], the convergence is not ensured during the training. Therefore, initial value of the network coefficients are set in the range [-1,1], randomly at the beginning of training. Afterwards, as iteration continues this interval is expanded 10 times by a proper number (KR). Keeping the positions and velocities in this range $(KR \times [-1,1])$ enables the algorithm to converge in a predefined number of iterations.

In order to demonstrate the success of improvements in the PSO, Easom function is used in what follows:

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)e^{-(x_1-p)^2-(x_2-p)^2} \tag{2.24}$$

where $-100 \le x_i \le 100$ for $i = 1, 2$. While the desired result with the standard PSO algorithm is found in the 7th run for 5000 iterations, with the improved PSO it is found in the 49th iteration in the 1st run. This optimization results in an error of 1×10^{-5}.

In this study, the MFLNN weights are trained by the improved PSO. The error value is obtained from the difference between reference signal and output signal of the control system. At the beginning of each epoch, the initial values of parameters are assigned randomly in the range just mentioned before. The error value must be computed for the fitness function calculations in the PSO. The Root Mean Square Error (RMSE) which is used as a fitness function, are calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y(k) - \hat{y}(k)\right)^2} \tag{2.25}$$

where N is number of pairs of the data which are used during the training, $y(k)$ is the output of the control system, and $\hat{y}(k)$ is the reference output. The key steps of the PSO are shown in Figure 2.2.

Set the parameters of the PSO
//Initialization
For Each particle in the swarm
Initialize position of particles randomly in the range $[X_{min}, X_{max}]$
Initialize velocity of particles randomly in the range $[V_{min}, V_{max}]$
End
//Iteration
Do
  For Each particle in the swarm
    Evaluate the fitness value
    If the fitness value is better than the best fitness value (*pbest*) in history
      Set present value as the new *pbest* .
  End
  Select the particle with the best fitness value in the neighborhood as the *lbest*
  For Each particle in the swarm
    Find the particle with the worst fitness value and change it
    to the particle with the best fitness value
    Compute particle velocity according to Equation (2.18)
    Clamp the velocity in each dimension to the range $[V_{min}, V_{max}] \times KR$
    Update position of particle according to Equation (2.23)
    Clamp the position in each dimension to the range $[X_{min}, X_{max}] \times KR$
    Apply mutation
  End
While Termination criteria is not fulfilled.

Figure 2.2. The pseudo-code of the PSO algorithm (Coban, 2011)

2.3. Hard Disk Driver

The track density and storage capacity of modern personal computers and work stations increase rapidly. Hence, it would be very difficult to control the HDDs. All the data are recorded in tracks which are concentric circles on a disk (Franklin et al., 1990). Disks are driven by spindle motor. In a hard disk driver, data is read from or written to by the READ/WRITE heads which mounted on the head slider. The actuator provides the head to move around the surface of the disk.

In the early 1990s the data storage capacity and data access times of the disk drives increased over 60 percent in every year. Towards the end of each year, the ratio increased to 100 percent. Nowadays, the designers transfer the task of the disk drives to the CPU to make improvements in computer environment (Hughes, 2002). Off-line error recovery, disk drive failure warnings, and storing data across multiple disk drives are studied in intelligent procedures (Dorf and Bishop, 2008).

The head-positioning servo mechanism is a control system that provides repositioning on the each track with minimum error and time. For the head-positioning servo mechanism, track-seeking and track-following are two main functions. Track seeking provides that READ/WRITE head to move from the current track to desired track in the minimum-time. Track-following ensures that READ/WRITE head is repositioned exactly over of given track with minimum position error in spite of under disturbances while information is read from or written to the disk.

The purpose of the disk drive system is to position the reader head on the desired track in minimum time. Firstly the plant, the sensor and the controller are decided. After that, model of the plant $G(s)$ and the sensor are obtained. In Fig. 2.3, the block diagram of a disk drive system is shown. Figure 2.3 shows the permanent magnet DC motor which rotates the reader arm and a linear amplifier (Dorf and Bishop, 2008). In this work, the model of the armature-controlled DC motor is preferred.

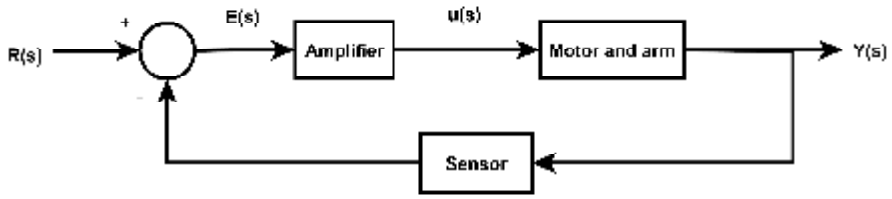


Figure 2.3. Block diagram of the HDD (Dorf and Bishop, 2008)

The transfer function of the disk drive system is given by (Dorf and Bishop, 2008)

$$G(s) = \frac{K_m}{s(Js+b)(Ls+R)} \tag{2.26}$$

where J is inertia of the arm and the read head, b is friction constant for the hard disk driver. K_a is amplifier gain, R is armature resistance, L is armature inductance

and K_m is motor constant value for the disk drive system. The values of these parameters are given in Table 2.1.

Table 2.1. Characteristic parameters of disk drive reader (Dorf and Bishop, 2008)

Parameter	Typical Value
J	1 N m s^2/rad
B	20 N m s/rad
K_a	10-1000
R	1 Ω
K_m	5 N m/A
L	1 mH

By using these values of the parameters the simplified transfer function of the open-loop head reader is obtained by

$$G(s) = \frac{Y(s)}{V(s)} = \frac{5000}{s(s+20)(s+1000)} \tag{2.27}$$

The difference equation obtained by using the zero-order hold for 0.1 second sampling period can be found as follows:

$$\begin{aligned} y(k) = {} & 1.1353 \times y(k-1) - 0.1353 \times y(k-2) \\ & + 0.0140 \times u(k-1) + 7.64 \times 10^{-3} \times u(k-2) \\ & + 6.9049 \times 10^{-7} \times u(k-3) \end{aligned} \tag{2.28}$$

3. STRUCTURE OF THE NEURO CONTROLLER

In this section, firstly, the MFLNN which is a new type of recurrent neural networks is trained with the PSO algorithm. At the end of the section, the MFLNN-PSO structure is tested with the chaotic time series and a Multiple-Input-Multiple-Output (MIMO) type nonlinear dynamic system. Afterwards, how the closed loop identification of the reader head position of a disk drive system with the MFLNN-PSO algorithm is described. Then, the system is tested with two different data sets. In the last section, a new type of neuro controller is proposed by using the MFLNN-PSO structure and then it is applied to two different kinds of dynamic systems and a hard disk drive system.

3.1 Training the MFLNN using the PSO Algorithm

In this chapter, the Multifeedback-Layer Neural Network (MFLNN) weights are trained by the Particle Swarm Optimization (PSO). This method (MFLNN-PSO) is applied to two different problems to prove accomplishment of the study. Firstly, a chaotic time series prediction problem is used to test the MFLNN-PSO. Also, the method is used for identification of a non-linear dynamic system. This study shows that the MFLNN-PSO can be used for dynamic system identification as well as controller design.

The PSO updates the network weights, which are used for training purpose. For each training epoch, initial parameters are defined randomly. A set of input and output data pairs is used for determining the error function to evolve parameters. The error is calculated by the output of the MFLNN as in Equation (2.1). This error is used in a fitness function of the PSO. The weights of neural networks are updated via the fitness value of the PSO. As a fitness function the root mean squared error (RMSE) criteria in Equation (3.1) is used.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y(k)-\hat{y}(k)\right)^2} \tag{3.1}$$

where N is size of data pairs, $y(k)$ is the output of the network and $\hat{y}(k)$ is the desired output of the neural network. The above process will be continued until the desired criteria are fulfilled.

3.1.1 Simulation Results

In this section, the proposed algorithm is applied to non-linear dynamic systems for prediction and identification. Training and prediction performances are determined by using RMSE criteria as in Equation (3.1).

Example 1

In this example, the learning and generalization ability of the MFLNN is tested with chaotic time series. The mathematical expression of the model is guided by

$$\dot{x}(t) = \frac{0.2x(t-t)}{1+x^{10}(t-t)} - 0.1x(t) \tag{3.2}$$

In 1993, Jang studied prediction of future values of this time series. Future values are calculated by using method in (Jang, 1993). The integration of above equation is done through the fourth order Runge-Kutta method. Time interval is 0.1, initial value of x, is equal to $x(0)=1.2$ and for $t<0$ x(t)=0. τ is defined as delay term and it is 17.

In this example, 1000 input-output data pairs are used. These data are obtained from

$$\begin{Bmatrix} x^d = \left[x(k-18)\ \ x(k-12)\ \ x(k-6)\ \ x(k)\right], \\ y^d = \left[x(k+6)\right] \end{Bmatrix} \tag{3.3}$$

where k is from 118 to 1117. First 500 data pairs and last 500 data pairs are used for the training and testing data set, respectively. The MFLNN is formed four inputs, five hidden and one output layer neurons. Figure 3.1 illustrates the outputs of the plant and MFLNN. The convergence behavior of the neural network is shown in Figure 3.2 which illustrates success of the proposed algorithm. The training continued along 9000 time steps and during this process 96 parameters are trained. The training and testing RMSEs, which are obtained in the end of training, are 0.0038 and 0.0044, respectively. The prediction error is shown in Figure 3.3.

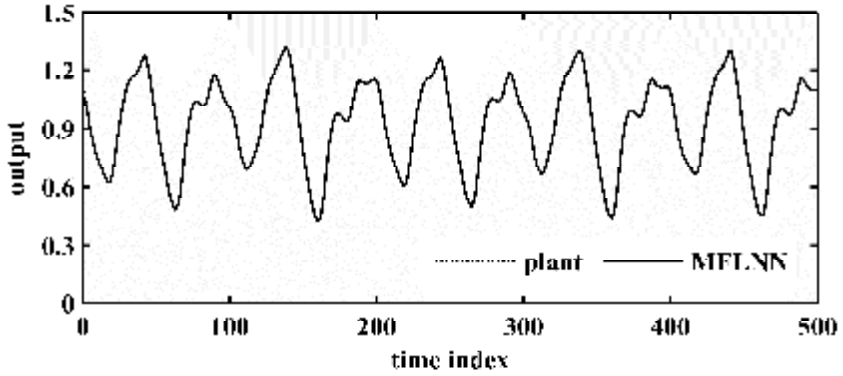


Figure 3.1. Output of the plant and the MFLNN

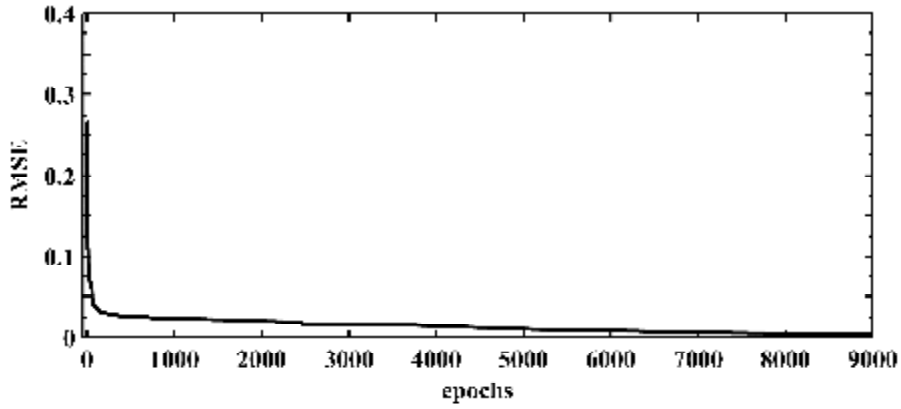


Figure 3.2. RMSE curves of the MFLNN

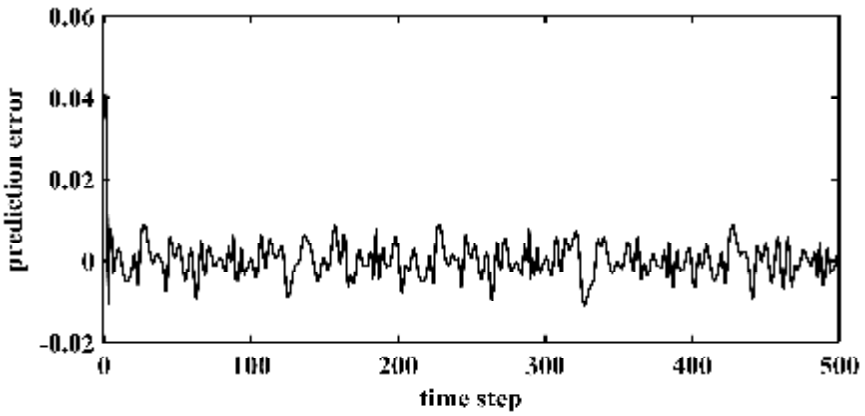


Figure 3.3. The prediction error in Example 1

Example 2

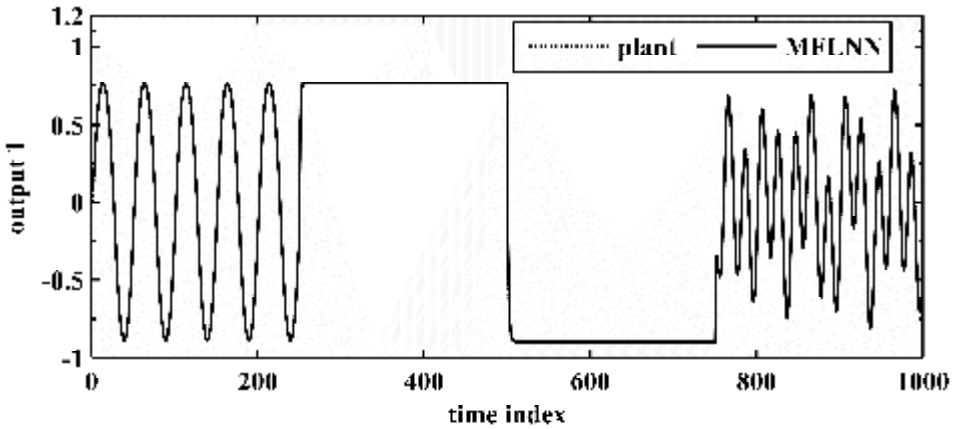
In the second example, a Multiple-Input-Multiple-Output (MIMO) type non-linear dynamic system which has two inputs and two outputs is identified. The system which is identified was used as in (Sastry et al., 1994) and (Juang and Lin, 1999). The equation of the plant is

$$\begin{bmatrix} y_{p1}(k) \\ y_{p2}(k) \end{bmatrix} = 0.5 \begin{bmatrix} \dfrac{y_{p1}(k-1)}{1+y_{p2}{}^2(k-1)} + u_1(k) \\ \dfrac{y_{p1}(k-1)y_{p2}(k-1)}{1+y_{p2}{}^2(k-1)} + u_2(k) \end{bmatrix} \tag{3.4}$$

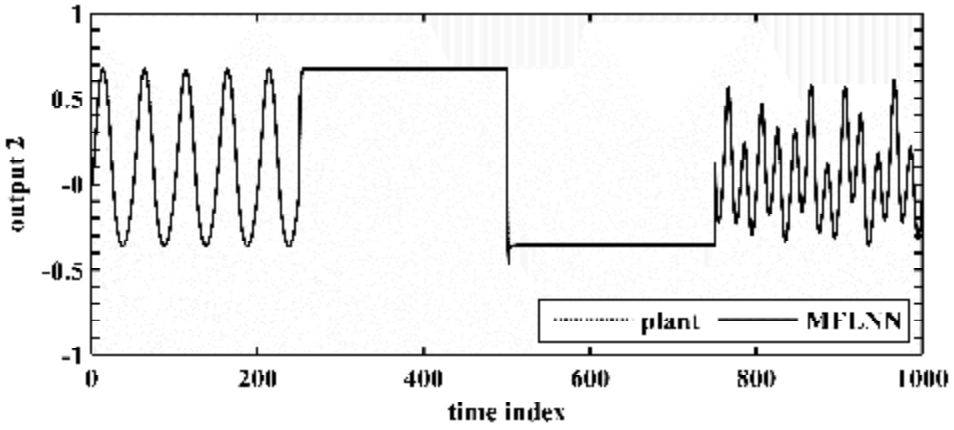
where $u_1(k)$ and $u_2(k)$ are the inputs of the system, $y_{p1}(k)$ and $y_{p2}(k)$ are outputs of the system, respectively, and k is the discrete time step. The MFLNN is consisted of two inputs, two outputs and two hidden layer neurons. First 500 samples of the training data, which are independent and identically distributed (i.i.d.) uniform sequence over [-2, 2], are obtained. Remaining 500 samples for two inputs are obtained through sinusoidal signal given by $\sin(p\,k/45)$. The testing data for two inputs is obtained as follows

$$u(k)=\begin{cases}\sin(p\,k/25), & k<250\\ 1, & 250\le k<500\\ -1, & 500\le k<750\\ 0.3\sin(p\,k/25)+0.1\sin(p\,k/32)+0.6\sin(p\,k/10), & 750\le k<1000\end{cases} \tag{3.5}$$

In Figure 3.4, the outputs of dynamic system and outputs of the MFLNN for the testing data are shown. First and second outputs of the system are demonstrated in Figure 3.4 (a) and Figure 3.4 (b), respectively. The RMSE curve is shown in Figure 3.5. For the MFLNN 42 parameters are trained in 9000 training time step. In the testing phase, the RMSE values of outputs y_1 and y_2 are 0.0096 and 0.0054, respectively. The prediction errors for both outputs are shown in Figure 3.6.



(a)



(b)

Figure 3.4. (a) First and (b) second outputs of the plant and the MFLNN

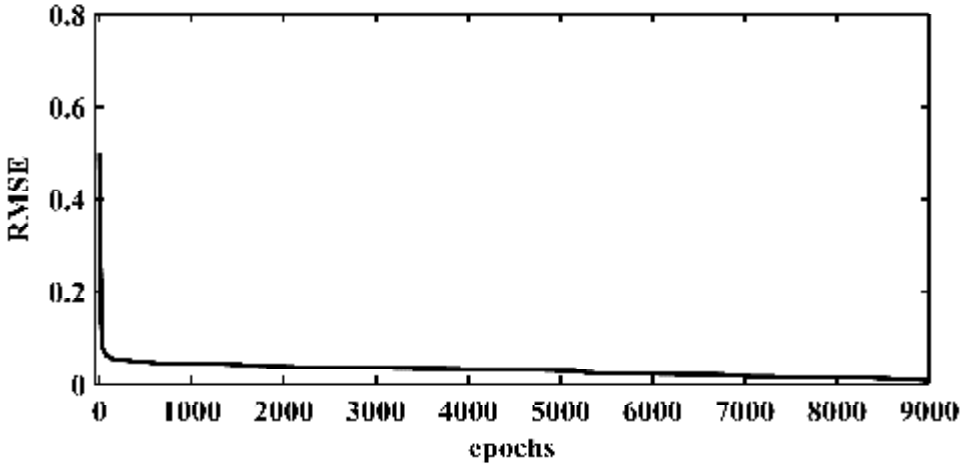
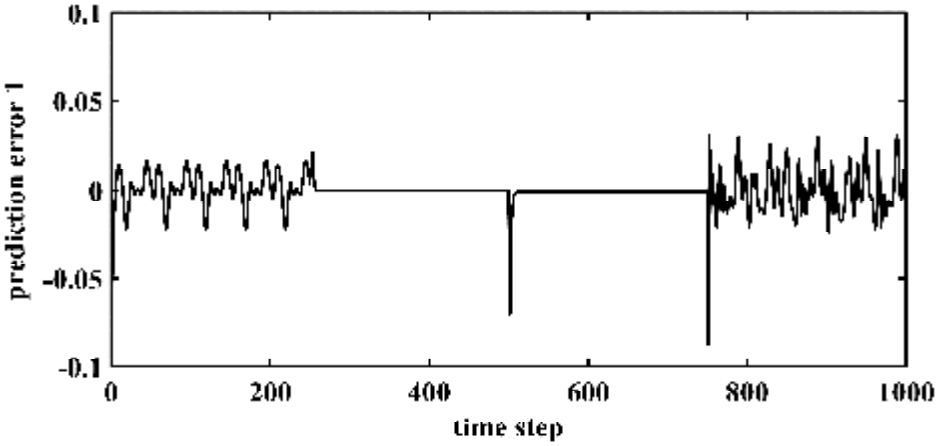
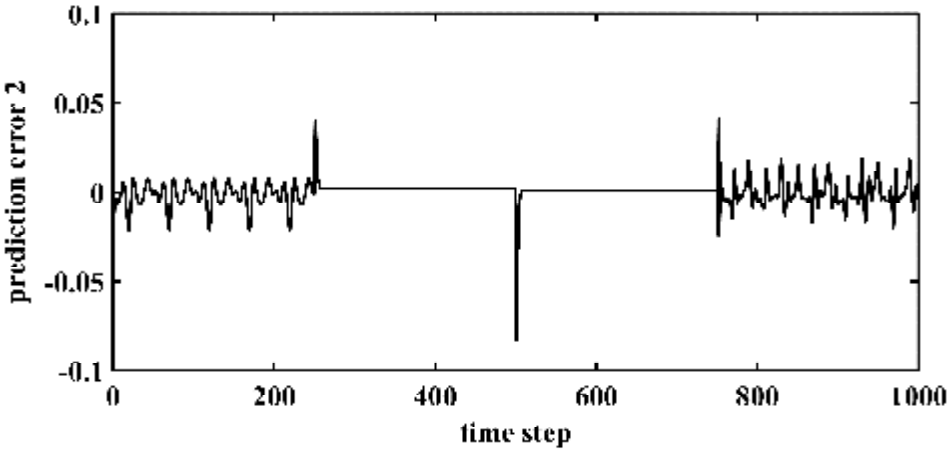


Figure 3.5. RMSE curves of the MFLNN



(a)



(b)

Figure 3.6. The prediction error in Example 2 (a) for first output and (b) for second output

3.2 Identification of Disk Drive Systems using the MFLNN and the PSO Algorithm

In this chapter, the closed loop identification of the reader head position of a disk drive system is proposed by using the Multifeedback-Layer Neural Network. To identify the system, the connection weights of the Multifeedback-Layer Neural Network (MFLNN) are trained by the Particle Swarm Optimization (PSO) algorithm. Simulation results show the effectiveness of the proposed method.

In this section to identify the closed-loop disk drive system the MFLNN-PSO algorithm is used. Performance of identification algorithm is determined by RMSE criteria as in Equation (3.1). For identification process the MFLNN has one input, one output and four hidden layer neurons.

3.2.1 Simulation Results

In this work, in the training phase 1000 samples are used. First 500 samples of the training data are obtained in the range [-2, 2] using a random number generator. And the remaining 500 samples are obtained from a sinusoidal signal given by $\sin(pk/45)$. The convergence behavior of the neural network is shown in Figure 3.7 which illustrates achievement of the suggested algorithm. In the training, 58 parameters are trained by the PSO. This process is done in 9000 training time step. The RMSE value of output in the training phase is 0.0070.

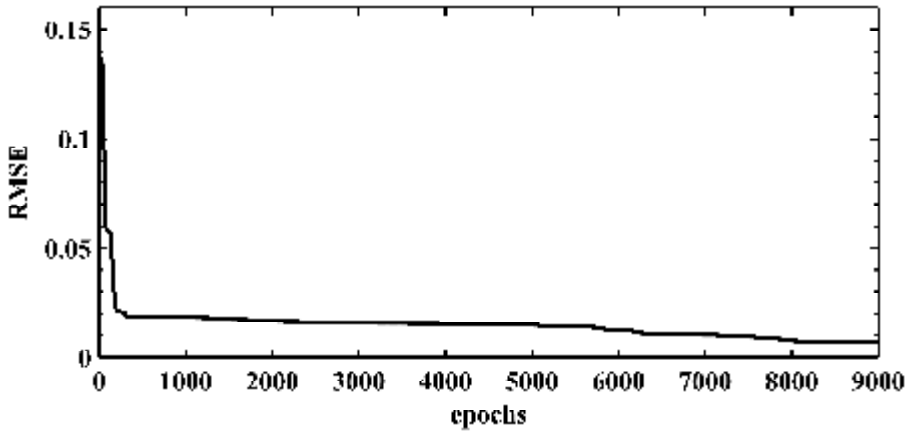


Figure 3.7. RMSE curves of the MFLNN

The system is verified with two different data sets. The first data set is obtained as follows

$$u(k)=\begin{cases}\sin(p\,k/25), & k<250\\ 1, & 250\le k<500\\ -1, & 500\le k<750\\ 0.3\sin(p\,k/25)+0.1\sin(p\,k/32) & \\ +0.6\sin(p\,k/10), & 750\le k<1000\end{cases} \tag{3.6}$$

The reference and prediction outputs of the system are shown in Figure 3.8. The RMSE value of output in the testing phase is 0.0086. The prediction error for the first test data set is shown in Figure 3.9. For the second data set 1000 samples are obtained by applying independent and identically distributed (i.i.d.) uniform sequence over [-2, 2]. The identification performance of the system for this testing data set is demonstrated in Figure 3.10. The RMSE value of output in the testing phase is 0.0090. The prediction error for the second test data set is shown in Figure 3.11.

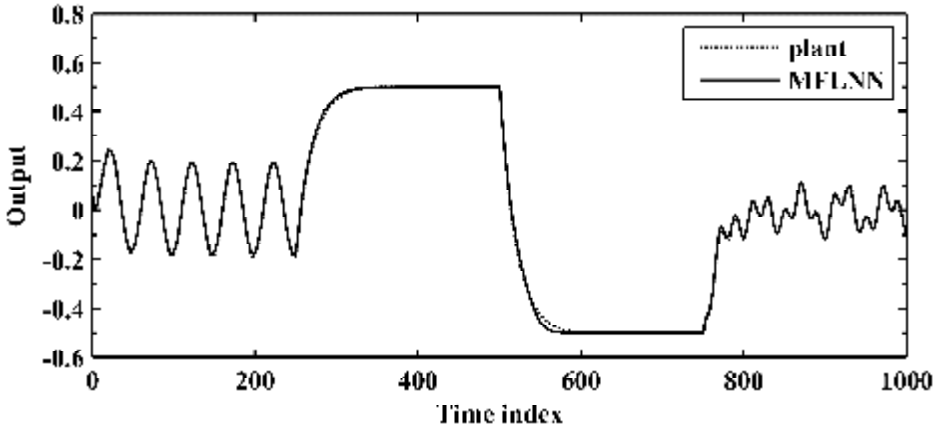


Figure 3.8. Outputs of the plant and the MFLNN for the first data set

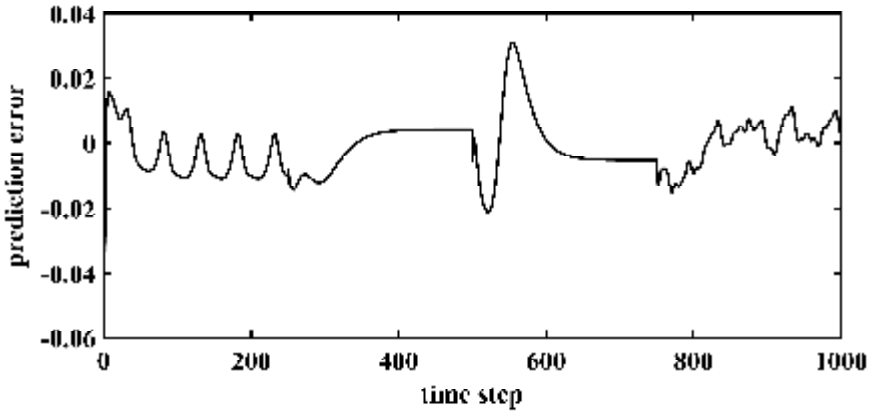


Figure 3.9. The prediction error for the first test data set

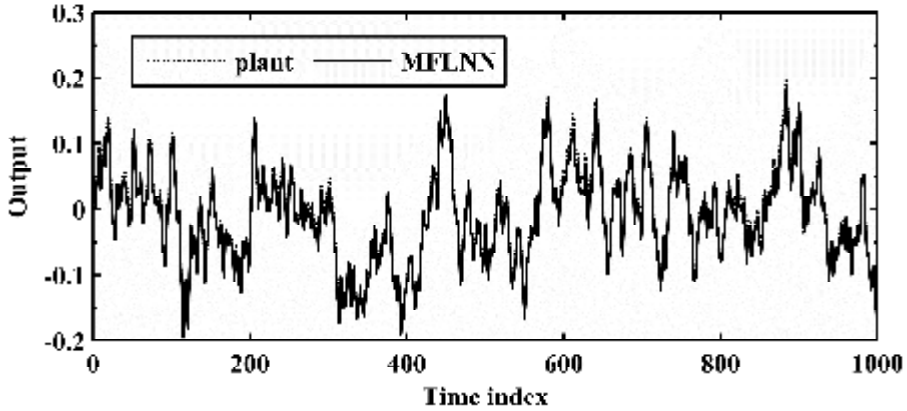


Figure 3.10. Outputs of the plant and the MFLNN for the second data set

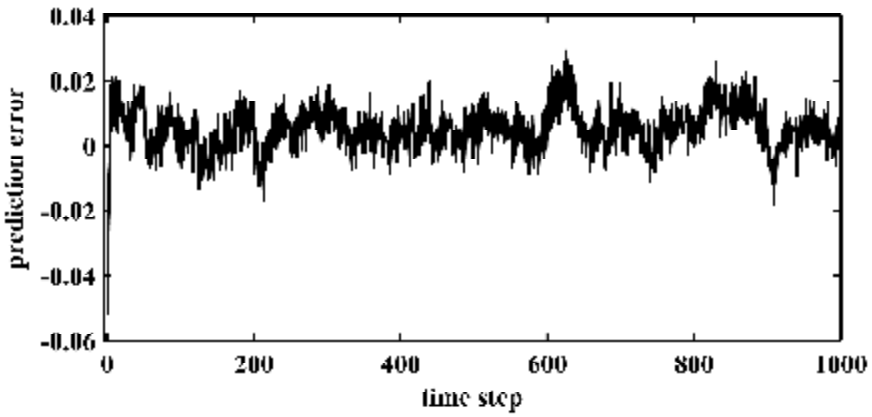


Figure 3.11. The prediction error for the second test data set

3.3 Controller Design with MFLNN-PSO

In this chapter, a new type of neuro controller is put forward by using the Multifeedback-Layer Neural Network (MFLNN) which has recently proposed. The connection weights of the MFLNN, which is used in this study, are trained by the Particle Swarm Optimization (PSO) algorithm. To improve training capability of the PSO, it is enhanced by some modifications. Firstly, this MFLNN-PSO controller is applied to two different kinds of dynamic systems. Then, it is applied to a hard disk drive system as a real physical example. Simulation results show that the MFLNN-PSO controller is effective and efficient on the control of dynamic systems and hard disk drive system.

The aim of this section is to describe how the MFLNN-PSO controller is designed and to explain how it can be used for control of a dynamic system. The neural networks are successfully applied to the dynamic system control (Hagan and Demuth, 1999). The network structure is crucial for the success and adaptation of the system. The MFLNN is chosen for this system, due to the fact that the MFLNN is a nonlinear model based on recurrent neural networks. Therefore, the controller structure which is designed in this study is nonlinear. A scheme in Figure 3.12 shows the neuro controller structure which is used in this study. In this model, fitness value is used to adjust the parameters of the neural network. The training is performed using the PSO algorithm. The fitness value that is necessary for PSO training is

obtained from difference between plant output signal and reference signal. We can see that the reference signal and past signal of the plant are input signals which enter the controller. The output signal of the controller is amplified and entered the plant. A similar control architecture with the genetic algorithm is used in Juang (2002) for dynamic system control.

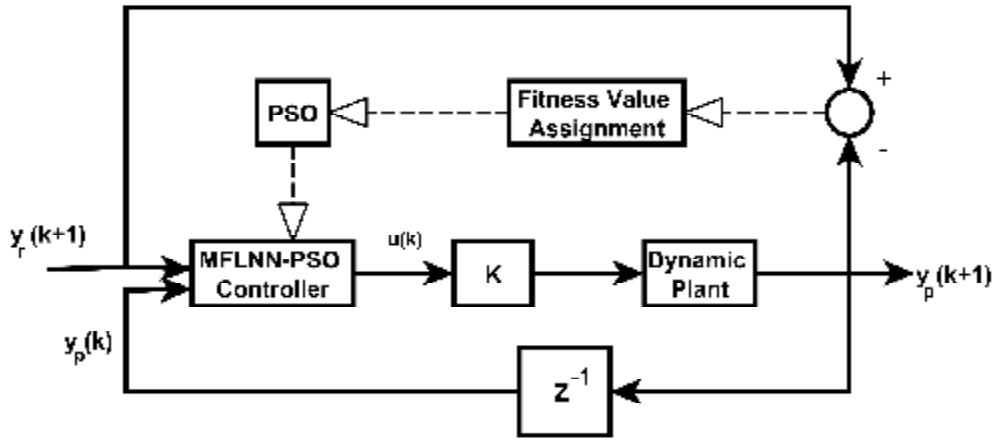
To train the network, the PSO algorithm is preferred instead of the derivative based algorithms. The PSO is one of the intelligent algorithms whose application is simple and easy. Initially, the particles in the PSO algorithm spread to whole space and find the global optimal solution without derivative and differential information.

For designing an effective neural controller, the behavior of the system which shall be controlled should be understood very well. The training of the neural network is related to determination of adaptable parameters of network, effectively. During the training, the properties of the plant are taught to the MFLNN. The training is continued until the difference between desired and actual plant response will be desired value. In each iteration, the weights are updated to minimize the error value. The success of the system is measured by the accuracy of the system' s response to input data that is not used in the learning phase. For all examples controller gain (K) is set 10.

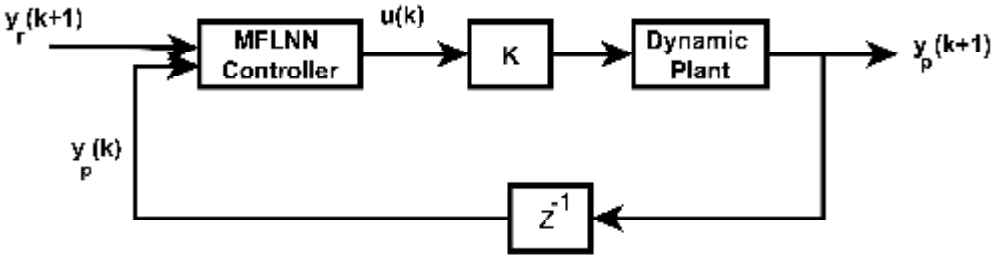
3.3.1 Simulation Results

In this section, the MFLNN-PSO is applied to control of some dynamic systems. Control configuration of the MFLNN-PSO for training is shown in Figure 3.12 (a) and for testing is given in Figure 3.12 (b). The inputs to the neural network controller consist of present reference signal and past output signal of the plant. For the purpose of producing control signal, the MFLNN-PSO controller takes advantage of both of these signals. Adaptation of the controller to different dynamic systems and behavior on the hard disk drive system is tested via different examples. In these examples, the MFLNN has four hidden layer nodes, two input layer nodes and one output layer node. In the input layer, linear activation function is used. In the output and hidden layers, hyperbolic tangent activation function is preferred for the

MFLNN to delineate nonlinear behaviors of the dynamic system. The PSO parameters used in this study is given in Table 3.1. Firstly, two different dynamic system control problems are considered. The results in these examples are compared with those in the TSK-type Recurrent Fuzzy Network with Genetic learning (TRFN-G) which has same controller design configuration as the MFLNN-PSO controller (Juang, 2002). In examples 1 and 2, the reference signals given in Juang (2002) is used. In addition, the same reference signals are applied to HDD system in example 3 and 4.



(a) (figure continued next page)



(b)

Figure 3.12. Controller configurations (a) for training and (b) for testing

Table 3.1. The PSO algorithm parameters used in examples

Parameter	**Symbol**	**Value**
Number of neighbors	NN	30
Swarm size	S_S	60
Number of parameters	D	62
First confidence coefficient (minimum value)	c1min	1.5
First confidence coefficient (maximum value)	c1max	2.5
Keep range coefficient	KR	10
Mutation rate	PM	0.001
Inertia (minimum value)	wmin	0.4
Inertia (maximum value)	wmax	0.9

Example 1

In the first example a nonlinear dynamic system which includes three past outputs is used (Juang, 2002). The difference equation of the plant to be controlled is given as follows:

$$y_p(k+1)=\frac{0.6y_p(k)+y_p(k-1)\left(y_p(k)+2.5\right)}{1+y_p^2(k)+y_p^2(k-1)}+u(k) \tag{3.7}$$

where y_p is output of the plant. 250 data which are used for training are obtained as follows:

$$\begin{aligned} y_r(k+1) &= 0.6y_r(k)+0.2y_r(k-1)+0.6\sin(2pk/25), && 1\le k\le 110 \\ &= 0.6y_r(k)+0.2y_r(k-1)+0.2\sin(2pk/25) \\ &\quad +0.4\sin(pk/32), && 110<k\le 250 \end{aligned} \tag{3.8}$$

where y_r is the reference output. The reference signal used for testing performance of the system is obtained as follows:

$$\begin{aligned} y_r(k+1) &= 0.6y_r(k)+0.2y_r(k-1)+0.2\sin(2pk/25) \\ &\quad +0.4\sin(pk/32), && 250<k\le 500 \end{aligned} \tag{3.9}$$

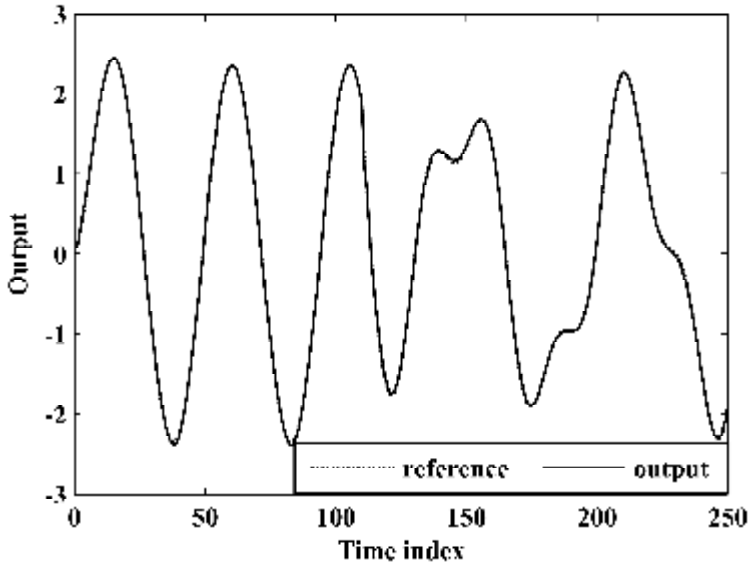
Training is continued for 9000 time steps. The fitness value used for PSO algorithm in training is calculated as follows:

$$RMSE = \sqrt{\frac{1}{250}\sum_{k=1}^{250}\left(y_p(k+1) - y_r(k+1)\right)^2} \tag{3.10}$$

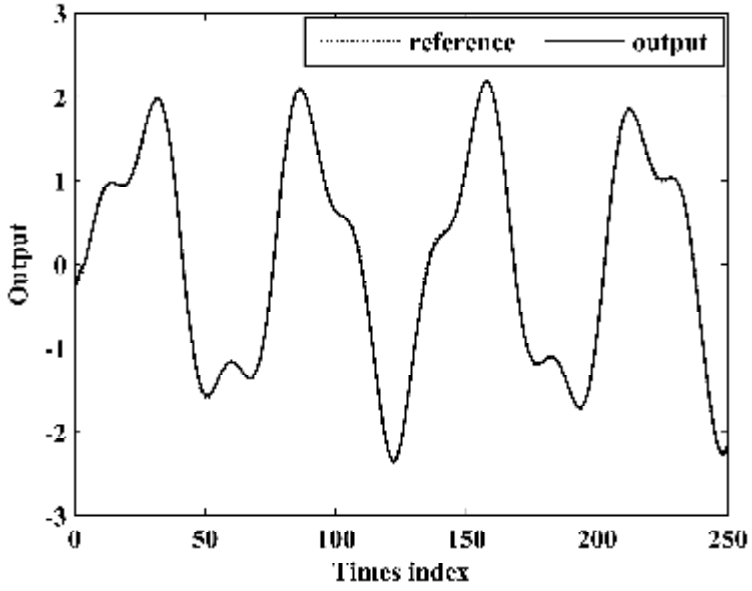
The RMSE values obtained in the results of the training and test stages are presented in Table 3.2. In order to show the results following the training, the reference and plant output are depicted in Figure 3.13. The result with the training data is illustrated in Figure 3.13 (a), while that with the testing data is shown in Figure 3.13 (b). It seems that the MFLNN-PSO has a better performance than the TRFN-G which has more hidden layers. In other words, the MFLNN-PSO has a higher accuracy for control of this system.

Table 3.2. Performance comparison between the MFLNN-PSO and TRFN-G in example 1

	TRFN_G	**MFLNN-PSO**
Training RMSE	0.0631	0.008703
Testing RMSE	0.0536	0.007544
Parameters	48	62



(a)



(b)

Figure 3.13. Reference signal and output of the control system for example 1 (a) for training (b) for testing

The RMSE curve of the system which indicates the performance of the system is shown in Figure 3.14. Since output of the control system for the testing signal and reference signal are almost the same, the prediction error of the system for the testing signal is given in Figure 3.15. Control signal in response to testing signal

is shown in Figure 3.16. According to the results, the MFLNN-PSO obtains better results as compared with the TRFN-G.

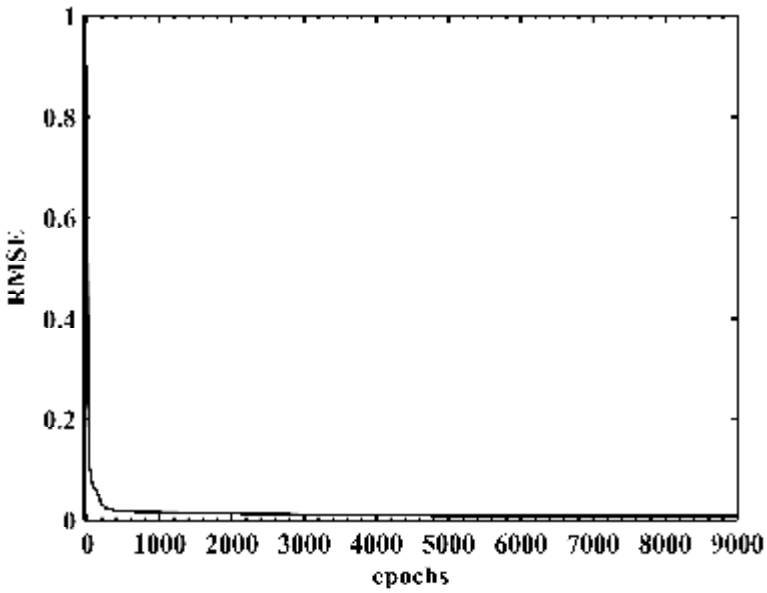


Figure 3.14. RMSE curve of the MFLNN-PSO for the example 1

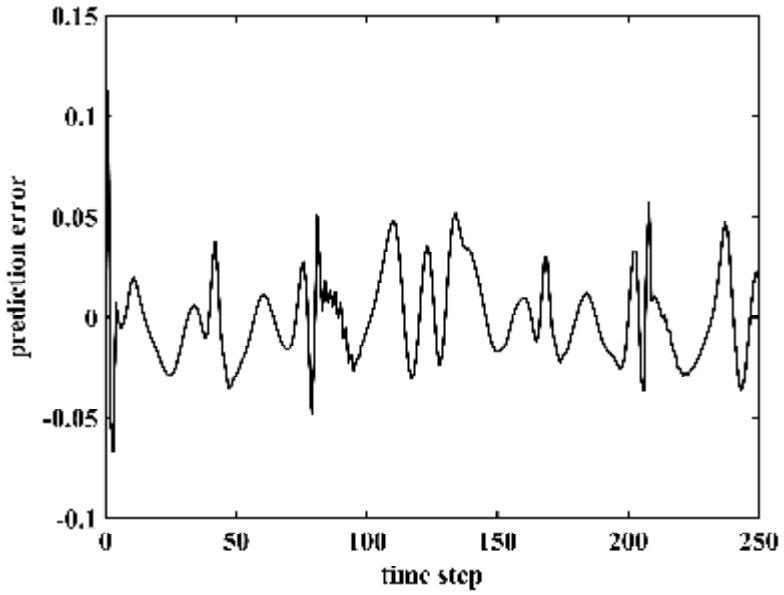


Figure 3.15. The prediction error of the control system for the example 1

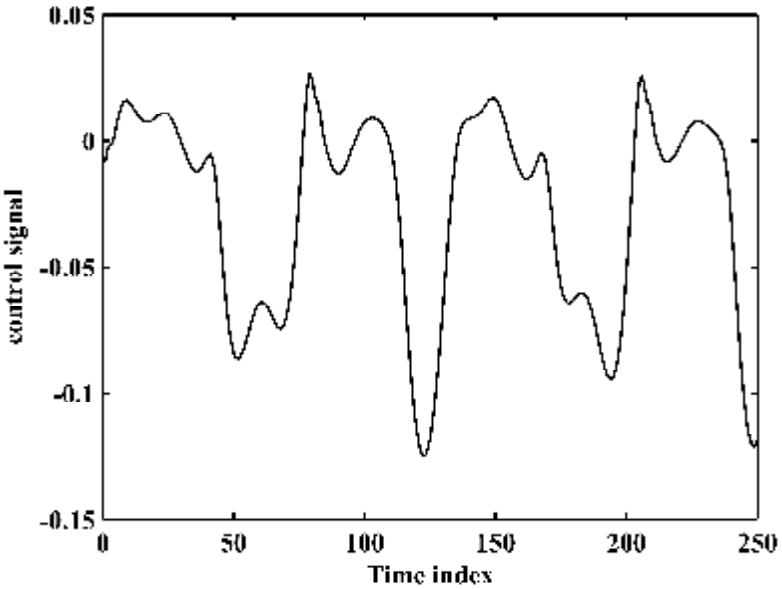


Figure 3.16. Control signal for the example 1

Example 2

In the second example, a dynamic system which is nonlinear with longer input delays is controlled. The difference equation of the time-delayed dynamic plant is given by (Juang, 2002):

$$y_p(k+1) = 0.72\,y_p(k) + 0.025\,y_p(k-1)u(k-1) + 0.01u^2(k-2) + 0.2u(k-3) \tag{3.11}$$

In this plant, the instantaneous output is a function of two previous outputs and four previous inputs. The reference signal which is used during training is given by

$$y_r(k+1) = \begin{cases} 10, & \text{if } k \le 50 \text{ or } 100 < k \le 150 \\ 15, & \text{if } 50 < k \le 100 \text{ or } 150 < k \le 200 \end{cases} \tag{3.12}$$

Besides, the same signal is used for the testing stage. The model which is illustrated in Figure 3.12 is used as a control configuration. The fitness value which is necessary for the training is calculated by

$$RMSE = \sqrt{\frac{1}{200}\sum_{k=1}^{200}\left(y_p(k+1) - y_r(k+1)\right)^2} \tag{3.13}$$

After 9000 time step training, in order to make comparison between the MFLNN-PSO and the TRFN-G, simulation results are given in Table 3.3. When comparing results, the proposed controller in this study have explicitly better results than the TRFN-G controller which is designed by Juang (2002). As seen in Table 3.3, the MFLNN-PSO achieves higher accuracy. The plant output and reference signal that obtained after the training are shown in Figure 3.17. The converge behavior of the MFLNN-PSO is indicated in Figure 3.18 that represents the system performance. For this system, the prediction error of the system for the testing signal is shown in Figure 3.19 which indicates the success of the system. Control signal which is response to testing signal is demonstrated in Figure 3.20. It is seen in Figure 3.20, the proposed controller uses less control efforts to track the reference signal.

Table 3.3. Performance comparison between the MFLNN-PSO and TRFN-G in example 2

	TRFN_G	**MFLNN-PSO**
Training RMSE	1.1374	0.0082937
Parameters	48	62

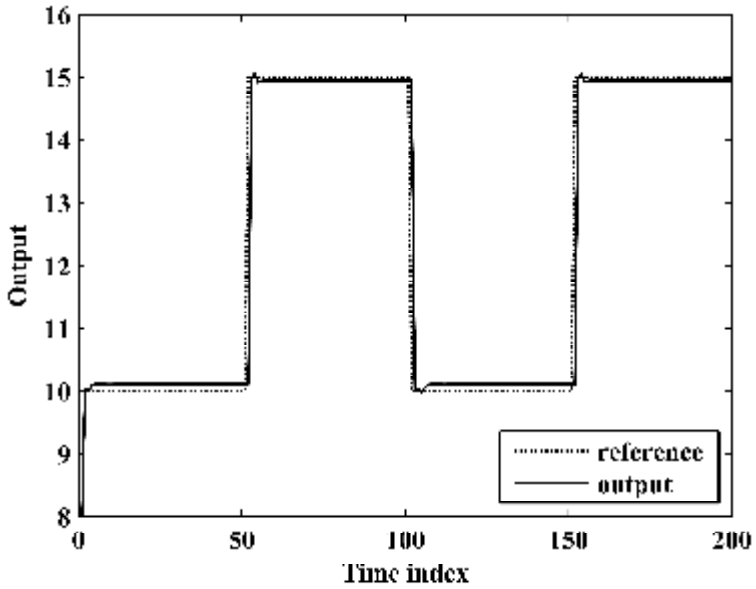


Figure 3.17. Reference signal and output of the control system for the example 2

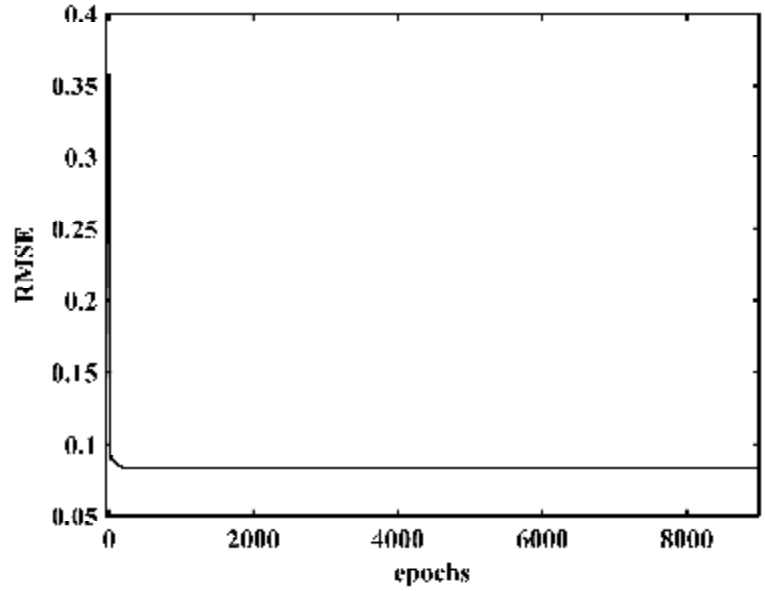


Figure 3.18. RMSE curve of the MFLNN-PSO for example 2

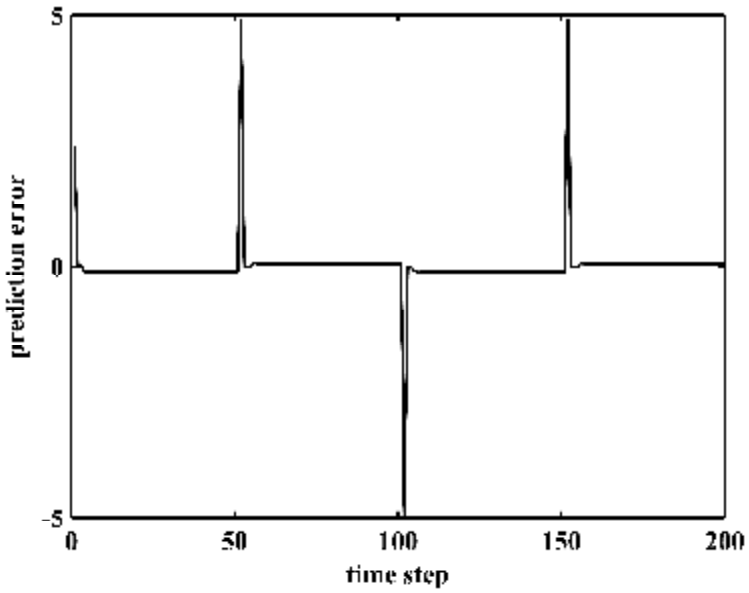


Figure 3.19. The prediction error of the control system for the example 2

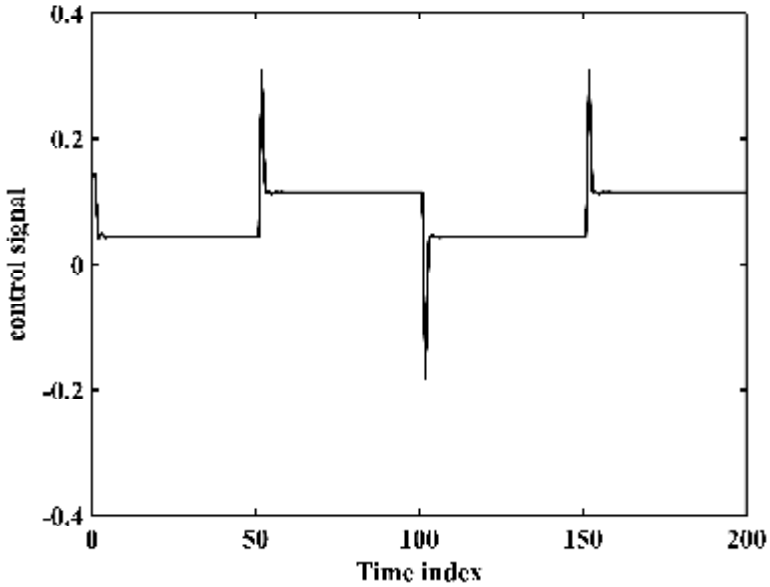


Figure 3.20. Control signal for the example 2

Example 3

In this example, the training and testing signals given by Equation (3.8) and Equation (3.9) which are the same as those in the example 1 is used for controlling the hard disk driver whose equation is given by Equation (2.28). Following the same way in the earlier examples, the control model shown in Figure 3.12 is used. 250 data

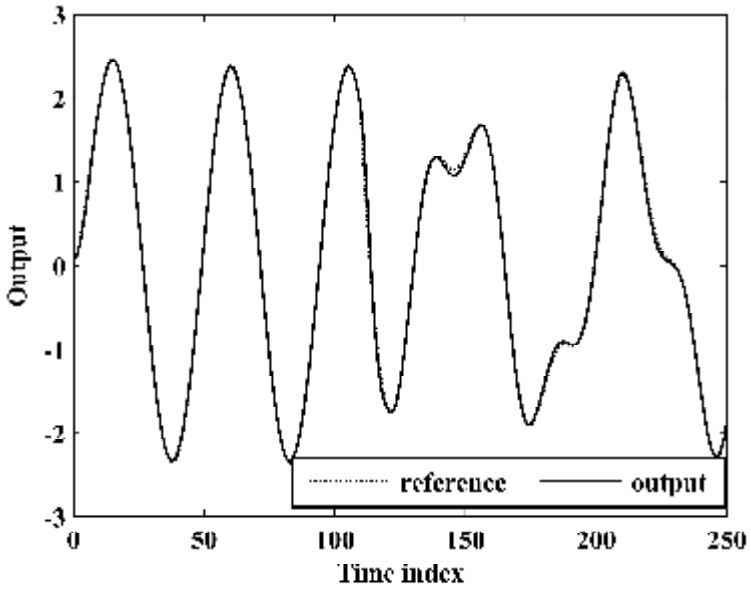
is used for both testing and training stages. The reference signals and outputs of the system are shown in Figure 3.21. The output for the training signal is given in Figure 3.21 (a), while that for testing signal is shown in Figure 3.21 (b). The RMSE values which are obtained from training and testing results are presented in Table 3.4. In this example, fitness value is calculated by

$$RMSE = \sqrt{\frac{1}{250}\sum_{k=1}^{250}\left(y_p(k+1) - y_r(k+1)\right)^2} \tag{3.14}$$

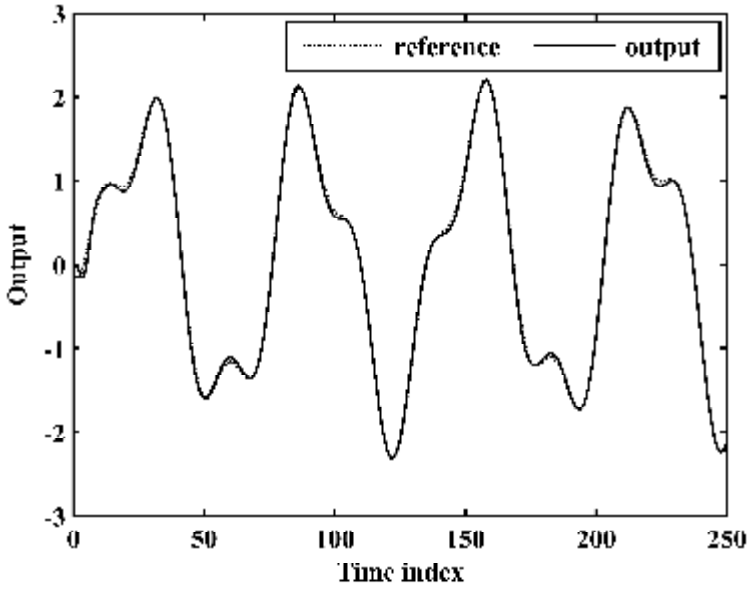
For 9000 training time steps, RMSE curve is shown in Figure 3.22. The tracking error between the desired and plant outputs during the training is indicated in Figure 3.23. As can be seen in Figure 3.23, desired and plant outputs are almost equal which means that the proposed controller follows the set-point efficiently. Control signal in response to testing signal is shown in Figure 3.24. The control signal which is obtained from the MFLNN-PSO controller shows the tracking performance of the system.

Table 3.4. Performance of the MFLNN-PSO in example 3

	MFLNN-PSO
Training RMSE	0.014370
Testing RMSE	0.012243
Parameters	62



(a)



(b)

Figure 3.21. Reference signal and output of the control system for the example 3 (a) for training (b) for testing

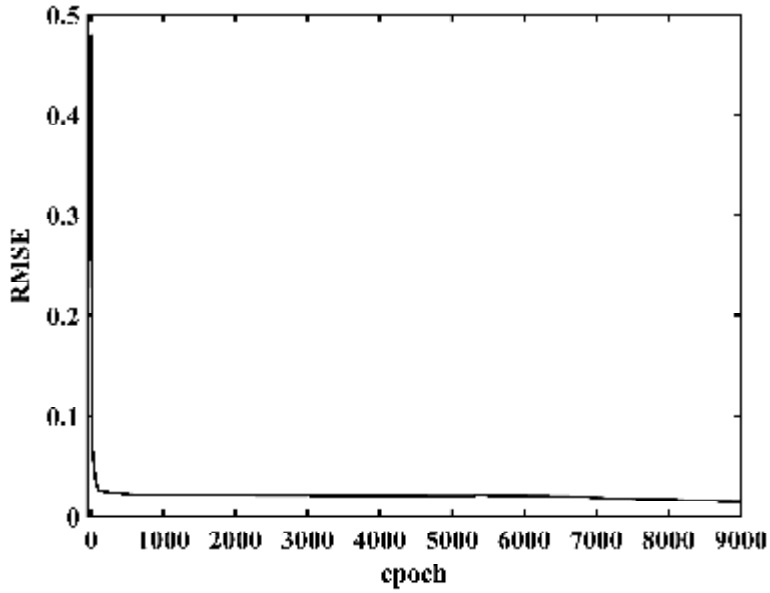


Figure 3.22. RMSE curve of the MFLNN-PSO for example 3

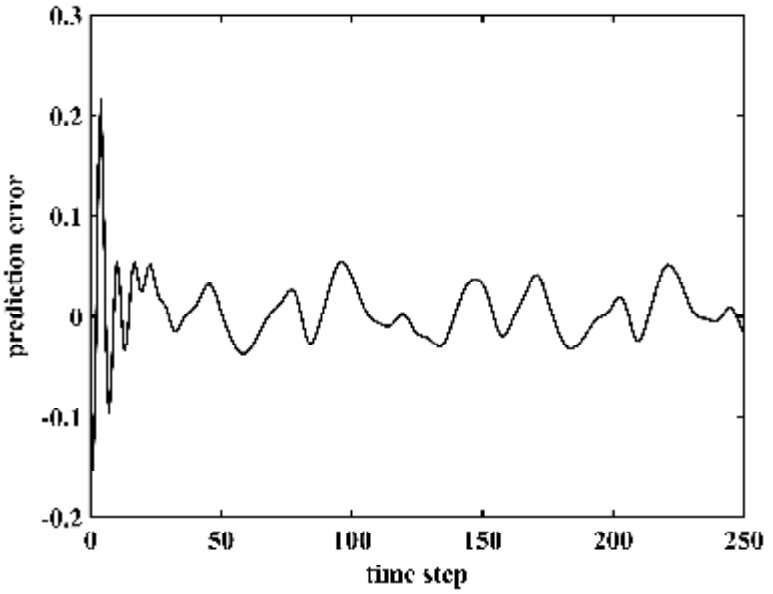


Figure 3.23. The prediction error of the control system for example 3

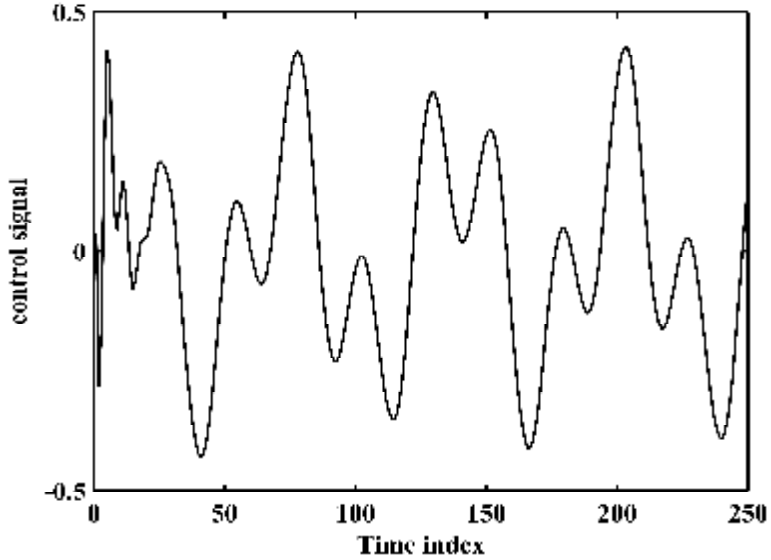


Figure 3.24. Control signal for the example 3

Example 4

In this example, for controlling the hard disk driver the same signals as in the example 2 are used. Both training and testing stage uses the same reference signals, which is given by Equation (3.12). The controlled plant, which is the HDD system, is given by Equation (2.28).

The convergent behavior which is obtained from training and testing results are given in Table 3.5. As in previous examples, the control model shown in Figure 3.12 is used. The fitness value used for PSO algorithm in training is obtained by

$$RMSE = \sqrt{\frac{1}{200}\sum_{k=1}^{200}\left(y_p(k+1) - y_r(k+1)\right)^2} \tag{3.15}$$

After the training, the desired signal and plant output are demonstrated in Figure 3.25 which shows the success of the system. Training is continued for 9000 time steps and the best network parameters are used for test phase. The RMSE curve of the system which indicates the performance of the system is shown in Figure 3.26. At the end of the training, the tracking error of the system for the testing signal is

illustrated in Figure 3.27 which shows the performance of the system. Control signal in response to testing signal is demonstrated in Figure 3.28. Figure 3.28 illustrates that, the controller requires less control efforts to track the reference signal.

Table 3.5. Performance of the MFLNN-PSO in example 4

	MFLNN-PSO
Training RMSE	0.074763
Parameters	62

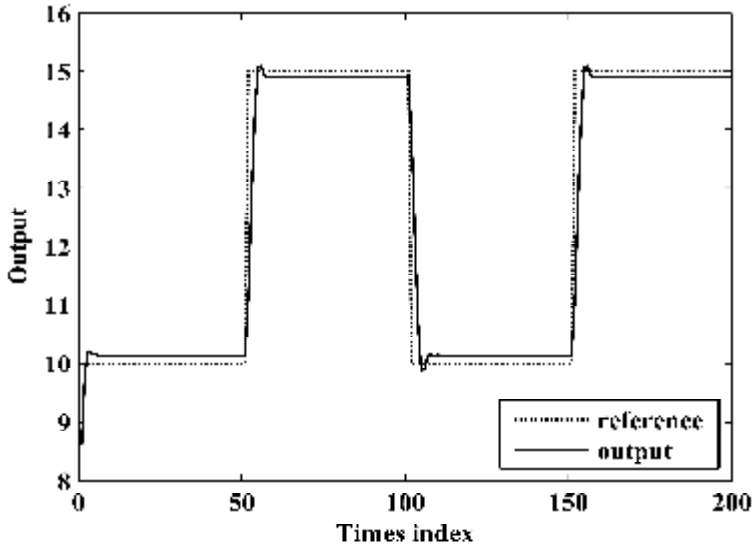


Figure 3.25. Reference signal and output of the control system for the example 4

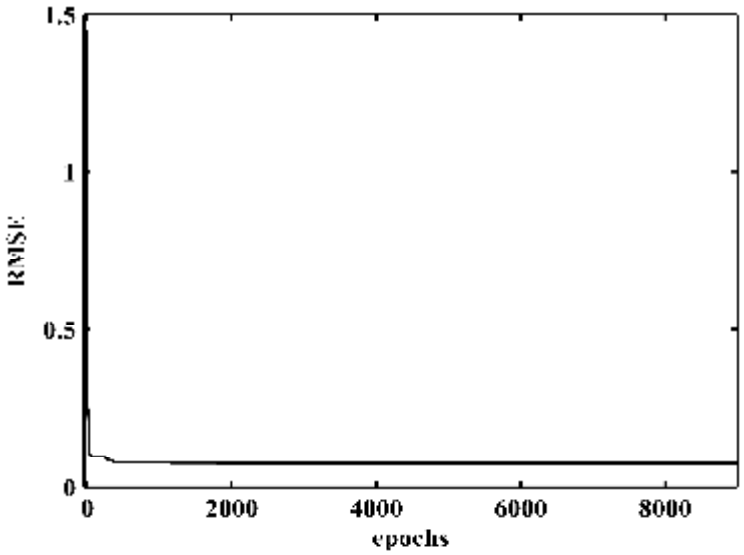


Figure 3.26. RMSE curve of the MFLNN-PSO for example 4

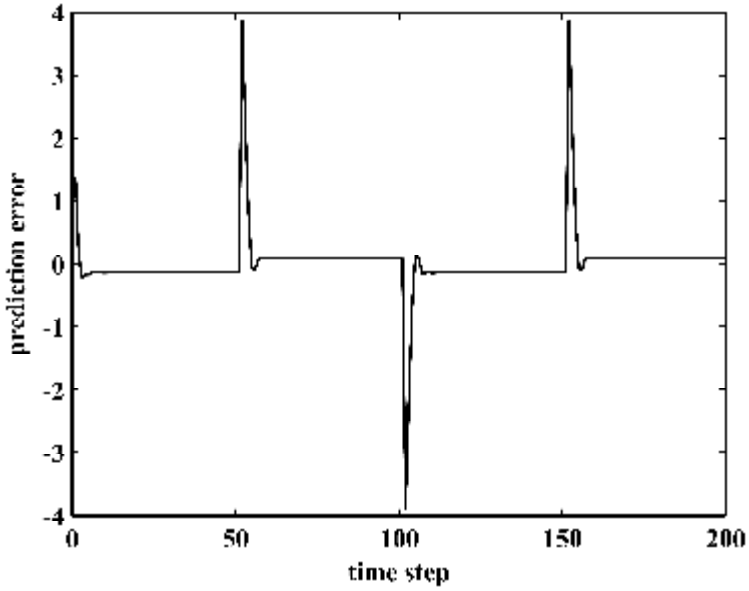


Figure 3.27. The prediction error of the control system for example 4

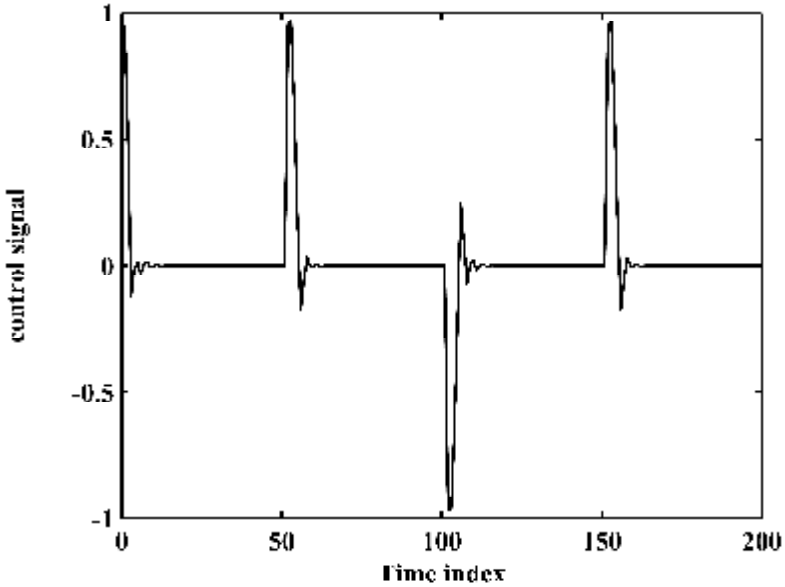


Figure 3.28. Control signal for the example 4

4. CONCLUSION

In this study, initially, the Multifeedback-Layer Neural Network (MFLNN) weights are trained by the Particle Swarm Optimization (PSO). The MFLNN-PSO structure is tested on chaotic time series prediction and non-linear dynamic system identification problems. According to the results, high prediction and identification accuracy is provided. Later, the closed loop identification of the reader head position of a disk drive system is proposed. The MFLNN is preferred to identify the system. Likewise, the MFLNN weights are trained by the PSO algorithm. The system is tested with two different data sets. According to the simulation results it has been shown that the disk drive system is identified successfully. Lastly, a new neuro controller using the Multifeedback-Layer Neural Network (MFLNN) is proposed for control of the disk drive system as well as nonlinear dynamical systems. In the same way, the PSO algorithm is used to tune the parameters of the MFLNN. By comparing the MFLNN-PSO controller with the other control configuration in the literature (TRFN-G), the effectiveness and efficiency of the proposed neuro controller is verified. It is seen that the MFLNN-PSO controller can be succesfully applied to the hard disk drive system as well as linear and nonlinear dynamic systems.

REFERENCES

AKSU I.O., and COBAN R., 2013. Identification of Disk Drive Systems using the Multifeedback-Layer Neural Network and the Particle Swarm Optimization Algorithm. The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE2013), Konya, TURKEY, 231-235.

ATSUMI, T., and MESSNER, W. C., 2013. Estimation method for unobservable settling vibration of head-positioning control in hard disk drives. Mechatronics, 23, 37-45.

BEYHAN, S., and ALCI, M., 2010. Stable modeling based control methods using a new RBF network. ISA Transactions, 49(4), 510-518.

BILLINGS S. A. and FUNG C. F., 1995. Recurrent Radial Basis Function Networks for Adaptive Noise Cancellation. Neural Netw., 8(2), 273-290.

CHEN, B. M., LEE, T. H., PENG, K., and VENKATARAMANAN, V., 2006. Hard disk drive servo systems (2nd ed). London:Springer-Verlag, 328p.

CHEN, Y.-L., CHENG, J., LIN, C., WU, X., OU, Y., and XU, Y., 2013. Classification-based learning by particle swarm optimization for wall-following robot navigation. Neurocomputing, 113, 27-35.

CHOI, Y.-K., LEE, M.-J., KIM, S., and KAY, Y.-C., 2001. Design and implementation of an adaptive neural network compensator for control systems. IEEE Trans. Ind. Electron., 48(2), 416-423.

CHOW, T. W. S., and FANG, Y., 1998. A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. IEEE Transactions on Industrial Electronics, 45(1), 151-161.

COBAN, R., 2011. A fuzzy controller design for nuclear research reactors using the particle swarm optimization algorithm. Nuclear Engineering and Design, 241(5), 1899-1908.

COBAN, R., 2013. A context layered locally recurrent neural network for dynamic system identification. Engineering Applications of Artificial Intelligence, 26(1), 241-250.

DONG, X., ZHAO, Y., XU, Y., ZHANG, Z., and SHI, P., 2011. Design of PSO fuzzy neural network control for ball and plate system. International Journal of Innovative Computing Information and Control, 7(12), 7091-7103.

DORF, R. C., and BISHOP, R. H., 2008. Modern control systems (11th ed.). Pearson. Prentice-Hall, Upper Saddle, NJ, 1056p.

ELMAN J. L., 1990. Finding Structures in Time. Cogn. Sci., 14, 179-211.

FRANKLIN, G. F., POWELL, J. D., and WORKMAN, M. L., 1990. Digital control of dynamic systems (3nd ed.). Reading, MA: Addison Wesley, 850p.

GE, S. S., LEE, T. H., and HARRIS, C. J., 1998. Adaptive neural network control of robotic manipulators. Singapore: World Scientific, 381p.

HAGAN, M. T., and DEMUTH, H. B., 1999. Neural networks for control. Proceedings of the 1999 American Control Conference, San Diego, CA, 1642-1656.

HOPFIELD, J.J. and TANK, D.W., 1985. Neural Computation of Decisions in Optimization Problems. Biol.Cybern., 52, 141-152.

HUGHES, G. F., 2002. Computers: wise drives. IEEE Spectrum, 39(8), 37-41.

JANG, J.-S. R., 1993. ANFIS: Adaptive-network-based Fuzzy Inference System. IEEE Trans. Syst., Man, Cybern., 23(3), 665– 685.

JORDAN, M. I., 1988. Supervised Learning and Systems with Excess Degrees of Freedom. COINS, Mass. Inst. Technol., Cambridge, MA, Tech. Rep. 88-27.

JUANG C.-F. and LIN C.-T., 1999. A Recurrent Self-organizing Neural Fuzzy Inference Network. IEEE Trans. Neural Netw., 10(4), 828– 845.

JUANG, C.-F., 2002. A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Trans. Fuzzy Syst., 10(2), 155-170.

JUNYOU, B., 2007. Stock Price forecasting using PSO-trained neural networks. IEEE Congress on Evolutionary Computation, 2879-2885.

KELEMEN, A., ABRAHAM, A., and CHEN, Y., 2008. Computational intelligence in bioinformatics. New York: Springer, 326p.

KENNEDY, J., and EBERHART, R.C., 1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, 1942-1948.

KIM, Y. H., and LEWIS, F. L., 2000. Optimal design of CMAC neural-network controller for robot manipulators. IEEE Trans. Syst., Man, Cybern. C, 30(1), 22-31.

KIM, Y. H., LEWIS, F. L., and ABDALLAH, C. T., 1997. A dynamic recurrent neural-network-based adaptive observer for a class of nonlinear systems. Automatica, 33(8), 1539-1543.

LEE, C.-H., and TENG, C.-C., 2000. Identification and control of dynamic systems using recurrent fuzzy neural networks. IEEE Transactions on Fuzzy Systems, 8(4), 349-366.

LEWIS, F.L., YESILDIREK, A., and LIU, K., 1996. Multilayer neural-net robot controller with guaranteed tracking performance. IEEE Trans. Neural Networks, 7(2), 388-399.

LEWIS, S., JAGANNATHAN, F. L., and YESILDIREK, A., 1999. Neural network control of robot manipulators and nonlinear systems. New York: Taylor and Francis, 442p.

LIN, F.-J., and WAI, R.-J., 2003. Robust recurrent fuzzy neural network control for linear synchronous motor drive system. Neurocomputing, 50, 365-390.

LIN, F.-J., LIN, C.-H., and HUANG, P.-K., 2004. Recurrent fuzzy neural network controller design using sliding-mode control for linear synchronous motor drive. IEE Proc.-Control Theory Appl., 151(4), 407-416.

MASTOROCOSTAS, P., and THEOCHARIS, J., 2002. A recurrent fuzzy neural model for dynamic system identification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 32(2) 176-190.

MENDES, R., CORTEZ, P., ROCHA, M., and NEVES, J., 2002. Particle swarms for feedforward neural network training. Proc. Int. Joint Conf. Neural Networks, 1895-1899.

MUKHOPADHYAY, S., and NARENDRA, K., 1993. Disturbance rejection in nonlinear systems using neural networks. IEEE Trans. Neural Networks, 4(1), 63-72.

PEARLMUTTER B. A., 1989. Learning State Space Trajectories in Recurrent Neural Networks. In Proc. Int. Joint Conf. Neural Networks, Washington, DC, 365-372.

PÉREZ-ARANCIBIA, N. O., TSAO, T.-C., and GIBSON, J. S., 2010. A new method for synthesizing multiple-period adaptive-repetitive controllers and its application to the control of hard disk drives. Automatica (Journal of IFAC), 46(7), 1186-1195.

PLETT, G., 2003.Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. IEEE Trans. Neural Networks, 14(2), 360-376.

REN, X. M., LEWIS, F. L., ZHANG, J., and GE, S. S., 2009. Feedforward control based on neural networks for hard disk drives. IEEE Trans. Magnetics, 45(7), 3025-3030.

SAN, P. P., REN, B., GE, S. S., LEE, T. H., and LIU, J.K., 2011. Adaptive neural network control of hard disk drives with hysteresis friction nonlinearity. IEEE Trans. Control Systems Technology, 19(2), 351-358.

SASTRY, P. S., SANTHARAM, G., and UNNIKRISHNAN, K. P., 1994. Memory Neuron Networks for Identification and Control of Dynamical Systems. IEEE Trans. Neural Netw., 5(2), 306-319.

SAVRAN, A., 2007. Multifeedback-layer neural network. IEEE Transactions on Neural Networks, 18(2), 373-384.

SHEIKHAN, M., HEMMATI, E., and SHAHNAZI, R., 2012. GA-PSO-optimized neural-based control scheme for adaptive congestion control to improve performance in multimedia applications. Majlesi Journal of Electrical Engineering, 6(1), 11-23.

SHI, Y., and EBERHART, R. C., 1999. Empirical study of particle swarm optimization. In: Congress on Evolutionary Computation. Washington, DC, USA, 1945-1950.

SREE HARI RAO V., YADAIAH N., 2005. Parameter Identification of Dynamical Systems. Int. J. Chaos, Solitons Fractals, 23, 1137-1151.

TANK D.W. and HOPFIELD J.J., 1986. Simple Neural Optimization Networks:An A/D Conventor ,Signal Decision Circuit, and a Linear Programming Circuit. IEEE Trans. Circ. Syst., CAS-33, 533-541.

TASSOPOULOS, I. X., and BELIGIANNIS, G. N., 2012. Solving effectively the school timetabling problem using particle swarm optimization. Expert Systems with Applications, 39(5), 6029-6040.

TSEKOURAS, G.E., 2013. A simple and effective algorithm for implementing particle swarm optimization in RBF network' s design using input-output fuzzy clustering. Neurocomputing, 108, 36-44.

WANG, J., GE, S. S., and LEE, T. H., 2001. Adaptive friction compensation for servo mechanisms. London: Springer Verlag.

CURRICULUM VITAE

İnayet Özge AKSU was born in Antakya, in November 9th 1987. She completed her elementary education and secondary education at Cumhuriyet İlköğretim Okulu. She graduated from Hacı Mehmet Koçarslan Anadolu Lisesi, in 2005. She completed university education at Department of Computer Engineering of Engineering - Architecture Faculty of Çukurova University in 2010. She worked at Osmaniye Korkut Ata University as a research assistant. She is working as a research assistant at Computer Engineering Department of Adana Science and Technology University.