DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EXPLOITING CHAOS IN SYSTEM IDENTIFICATION AND CONTROL

by
Mehmet ÖLMEZ

March, 2013
İZMİR

EXPLOITING CHAOS IN SYSTEM IDENTIFICATION AND CONTROL

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of PhD in Electrical and Electronics Engineering

by
Mehmet ÖLMEZ

March, 2013
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"EXPLOITING CHAOS IN SYSTEM IDENTIFICATION AND CONTROL "** completed by **MEHMET ÖLMEZ** under supervision of **PROF. DR. CÜNEYT GÜZELİŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

[illegible]

Prof. Dr. Cüneyt GÜZELİŞ

Supervisor

S. Sarıgül

Prof. Dr. Saide SARIGÜL

Thesis Committee Member

[illegible]

Yrd. Doç.Dr. Güleser KALAYCI DEMİR

Thesis Committee Member

[illegible]

Doc. Dr. Mustafa I. Yelcin

Examining Committee Member

[illegible]

Prof. Dr. Haldun Karaca

Examining Committee Member

[illegible]

Prof. Dr. Ayşe OKUR
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Prof. Dr. Cüneyt Güzeliş for his invaluable guidance, constructive feedback, motivating attitude. Without the support and constant supervision of such an eminent professional, this thesis would never have materialized. His inspiring advice and guidance helped me overcome every impasse and obstacle and allowed me to realize my true potential. This thesis is a testament to his dedication, brilliance and generosity.

I also would like to gratefully acknowledge the supervision I received from Prof. Dr. Saide Sarıgül and Asist. Prof. Dr. Güleser Kalayci Demir during the writing of this thesis. Their careful guidance, insightful comments and useful suggestions were crucial for the successful completion of this thesis.

I would like to extend my heartfelt thanks to all my colleagues for their understanding manner, encouragement and motivational talks. Their friendly support was definitely of great importance during this process.

I take great pleasure in thanking my beloved parents Suzan Ölmez and Zafer Ölmez, whose caring and encouraging attitude has always provided me with strength and confidence. They have always been there for me. I believe special thanks need to go to them as their valuable contributions helped me manage my time better during the process.

I would like to thank Ümran Board and Iain Board for providing unending encouragement, moral support and valuable advice. Their support and help sustained me throughout this process.

I wish to thank Robert Board and Ann Board for their good will. It is good to know that they have faith in me and my abilities.

Last but not least, I cannot forget my brilliant and beautiful wife Sinem Ölmez. She stood by me through thick and thin, good times and bad, unconditionally extending endless love, patience, inspiration and support. She was the star that guided me during this journey. I shall forever be in her debt.

Mehmet ÖLMEZ

EXPLOITING CHAOS IN SYSTEM IDENTIFICATION AND CONTROL

ABSTRACT

The thesis presents three main contributions: Two methods for system identification and a method for robustifying a chaotification method are developed in the thesis. The identification methods which are based on input-output data are used for the identification of linear, nonlinear and chaotic plants. Both identification methods construct nonlinear state equations for systems to be identified by using white Gaussian noise or chaotic state sequences in learning state recursions. Nonlinear state equations and output equations in these identification methods are realized by artificial neural networks. Extensive computer experiments on a set of benchmark plants for noisy input and initial conditons show that the developed methods provide significantly better performances when compared to the known identification methods. The third contribution of the thesis is another application in the direction of exploiting chaos now for forcing a nonchaotic plant to chaos in a robust way.

Keywords: Linear system identification, nonlinear system identification, chaotic system identification, robust chaotification

SİSTEM TANILAMA VE DENETİMİNDE KAOSUN KULLANIMI

ÖZ

Tezde ilişkin bilimsel yazına yapılan üç ana katkı sunulmaktadır: Sistem tanılama için iki ve gürbüz olarak kaotik yörüngeleri izleyebilmek için bir yöntem. Sistem tanılama için önerilen yöntemler, giriş-çıkış verilerine dayalı olarak doğrusal, doğrusal olmayan ve kaotik sistemlerin tanılanması için geliştirilmiştir. Yöntemlerde, doğrusal olmayan durum geçişlerinin verilerden öğrenilmesinde, beyaz Gauss gürültüsü veya kaotik durum dizileri kullanılmıştır. Her iki system tanılama yönteminde, doğrusal olmayan durum denklemlerinin ve çıkış denklemlerinin gerçekleştirilmesinde yapay sinir ağları model olarak alınmıştır. Önerilen tanılama yöntemlerin başarımları bilimsel yazında iyi bilinen doğrusal, doğrusal olmayan ve kaotik sistemlerin tanılanmasında gürültülü giriş ve farklı başlangıç koşulları altında sınanmıştır. Diğer iyi bilinen tanılama yöntemleriyle karşılaştırıldığında her iki yöntemin daha iyi başarımı verdiği görülmüştür. Tezdeki üçüncü katkı da kaostan yararlanma doğrultusunda bir diğer uygulama olarak gürbüz olarak kaotik yörüngelerin izlenmesi için yöntem geliştirilmesidir.

Anahtar sözcükler : Doğrusal sistem tanılama, doğrusal olmayan sistem tanılama, kaotik sistem tanılama, gürbüz kaotikleştirme

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

A control system or said plant which is illustrated in Figure 1.1 is a system whose input is used to force its output usually to a desired trajectory which might be constant or time-varying (Nagrath & Gopal, 2005; Bolton, 2004; Distefano, Stubberud, & Willams, 1990; Phillips & Harbor, 2000).

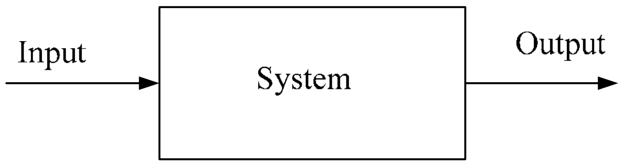

Figure 1.1 System to be controlled

The input of control systems which is called as control input is provided by a controller in a feedforward way shown by Figure 1.2 or in a feedback way shown by Figure 1.3 (Kumar, 2007; Doebelin, 1985; Bhattacharya, 2011; Ogata, 2009; Dorf, 1992; Aström & Murray, 2008; Kuo, 1995).

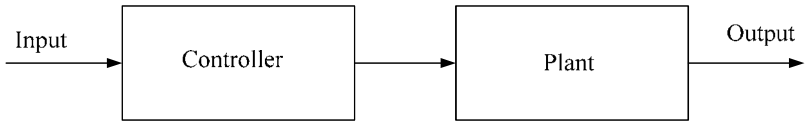

Figure 1.2 Open-loop control system

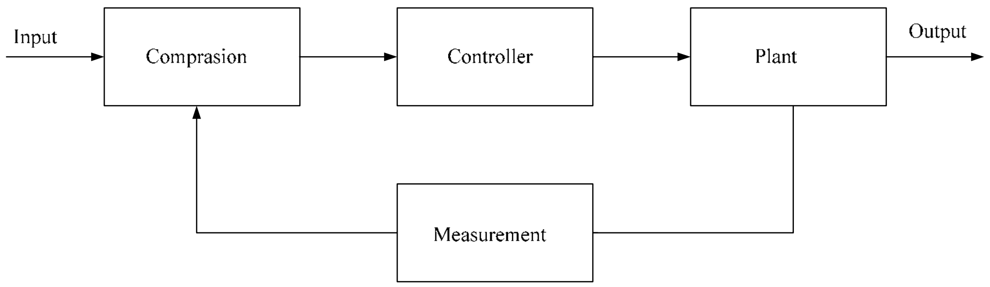

Figure 1.3 Closed-loop control system

In order to design a controller, one needs to have a model for the plant. A plant model may be obtained either by exploiting the physical laws which are assumed to govern the plant dynamics or by applying an identification method based on the input-output measurements (Zadeh, 1956). In earlier works (Bode, 1940; Bode 1945; Nyquist, 1932; James, Nichols, & Philips, 1947), the identification techniques were restricted to Single-Input Single-Output (SISO) systems (Gevers, 2006). Then, Aström and Bohin introduced a system identification method based on maximum likelihood which is a method widely studied in mathematical statistics, including time-series models (Aström & Bohin, 1965; Gevers, 2006; Koopman, Rubin, & Leipnik, 1950; Hannan, 1960). The maximum likelihood method based on minimizing a parameter-dependent criterion serves to estimate the parameters of differential equation models which are known in statistical literature as ARMA (Auto-Regressive Moving Average) or ARMAX (Auto-Regressive Moving Average with eXogeneous inputs) (Nelles, 2001; Larsson & Massberg, 2003; Wang & Garnier, 2012; Anderson,1971; Hannan,1970).

Another direction in the system identification is to employ state equation model as the alternative to the input-output models such as ARMA and ARMAX. Ho and Kalman presented method on how to determine linear minimal state space model from impulse response data (Ho & Kalman, 1966). State space approach is a mathematical model of a physical system defined by a set of coupled first order differential equations in terms of so-called internal variables known as state variables together with the input and output variables. State space model is especially appropriate for MIMO (Multi-Input Multi-Output) systems in contrast to formerly used models.

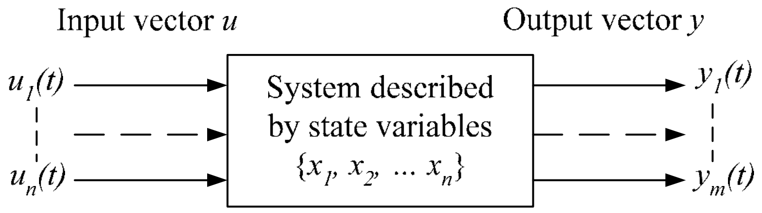

Figure 1.4 System by states, inputs and outputs

The state space model of a LTI (Linear Time Invariant) system with r inputs and m outputs can be described as

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t) \tag{1.1}$$

$$y(t) = Cx(t) + Du(t) \tag{1.2}$$

with $A \in \mathbb{R}^{\mathrm{nxn}}$, $B \in \mathbb{R}^{\mathrm{nxr}}$, $C \in \mathbb{R}^{\mathrm{mxn}}$, and $D \in \mathbb{R}^{\mathrm{mxr}}$ where u, y, x are the input, the output, and the state of the system, respectively. The problem in the LTI state space system identification methods is to obtain a minimal state space realization for a given data forming impulse response of the system (Shutter, 2000).

The improvements in system identification were followed by Box and Jenkins approach (1970) and Akaike approach (1976). Box and Jenkins approach employed an initial data analysis to the identification of a model. Its drawback is: It is being developed for single output case (Deistler, 1994; Gevers, 2006). Between 1975 and 1985, prediction error methods dominated in system identification field because of increasing computer speed and development of identification softwares (Ljung, 1978). Thereafter, identification methods for MIMO systems were explained in 1976 (Hazewinkel & Kalman, 1976; Clark, 1976). In the same year, the methods were developed for closed loop systems too (Söderström, Ljung, & Gustavsson, 1976). Following these studies, in the 1980s and 1990s many papers and researchs contributed to the developments in the area of system identification from different perspectives but in the two main directions, i.e. in the input-output and state space frameworks (Sin & Goodwin, 1980; Anderson & Gevers, 1982; Larimore, 1990; Juang, Phan, Horta, & Longman, 1993; Overschee, Moor, 1994; Verhaegen, 1994; Viberg, 1995; Chui & Maciejowski, 1996).

All research papers about system identification try to find the best representation of systems. During this procedure, there are several methods and models in defining systems such as prediction error method, state-space model based methods e.g. subspace methods, Hammerstein, Wiener, Hammerstein-Wiener, nonlinear ARX, and nonlinear ARMAX model based methods.

Artificial neural networks which have been emerged as mathematical models inspired by biological nerves systems are used as nonlinear system identification models designed by learning algorithms applied on measurement data (Li & Tongwen, 1993; Chu, Shoureshi, & Tenorio, 1990; Sugimoto, Matsumoto, & Kabe, 1992; Ljung & Sjöberg, 1992; Jovanovic, 1997; Kim & others, 2004; Fekih, Xu, & Chowdhury, 2007; Seyad & Cao, 2008).

Although system identification is a well established discipline in control area, there is still a need for developing efficient and high performance system identification methods especially for complex systems such as chaotic ones. Herein, chaos can be defined as the tiniest of variations in a system may cause unpredictable response of that system. Chaos that is a phenomenon of nonlinear dynamical systems having topologic transitivity and sensitive dependence on initial conditions (Kellert, 1993; Chen, 1997; Chen & Shi, 2006) has several application areas from astronomy to engineering. Secure communication and liquid mixing are among the most important engineering applications of chaotic systems (Kocarev and others, 2001; Chen, 1997; Ottino, 1989). The former one which exploits the noisy like spectrum of chaotic signals needs the identification of a chaotic system, i.e. the identification of the chaotic system model and its parameters used in the transmitter. The latter also exploits the chaos but for an interesting control purpose: As supposing the usefulness of a chaotic behavior for a plant, for instance a dc motor in liquid mixing, the plant under consideration which is originally not chaotic is chaotified to become a chaotic system (Chen & Dong, 1998; Chen, 1998; Chen, 2001; Chen, 2003; Chen & Shi, 2006). There is a growing interest in chaotification because of feasibility in real world applications (Chen & Shi, 2006; Jakimoski & Kocarev, 2001; Kocarev and others, 2001; Schiff and others, 1994; Ditto and others, 2000).

This thesis is a work in the direction of exploiting the chaos in the identification of complex systems and also in the chaotification also said anti-control of originally non-chaotic systems. The thesis introduces two new system identification methods which are used for constructing nonlinear state space models employing artificial neural networks by exploiting chaotic signals in learning nonlinear state recursions. The thesis also introduces a robust chaotification method for the systems which are not originally chaotic.

Chapter 2 delivers a background on system identification methods and the models used in these methods. Chapter 3 and Chapter 4, respectively, presents LSS-CSR (Learning State Space with Forced Chaotic State Recursion) and LSS-CSO (Learning State Space with Chaotic State Observer) system identification method for identifying linear, nonlinear and chaotic plants. In both chapters, the performances of the methods are evaluated with linear, nonlinear and nonautonomous chaotic plant. The performences of the methods are also evaluated under additive input noise and under different initial states.

Chapter 5 introduces the proposed robust chaotification method for anti-control of non-chaotic systems. It is shown in this chapter that the parameter region leading a desired chaotic behavior for the overall closed-loop system is enlarged by the proposed robustification method providing a better choice of controller parameters. The performance of the proposed robust chaotification method is tested under an additive input noise and the bifurcation diagram of the noisy system is given. Also in this chapter, a user friendly graphical user interface is presented where the user can change the plant and controller parameters, add input noise to the system by specifying the mean and standard deviation, draw the phase portraits of the system with noise and without noise, can observe the frequency spectrum of the system with noise and without noise, and see the bifurcation diagram of the system while changing the plant and controller parameters. Finally, in Chapter 6, the conclusion is given.

CHAPTER TWO
BACKGROUND ON SYSTEM IDENTIFICATION

This chapter gives a background on system identification, modeling, input-output representations, parameter estimation and anti-control (chaotification). The concepts presented in this chapter will be used in following chapters.

2.1 What is System Identification?

System identification is a process of finding a mathematical system model from measured data with some predetermined criteria (Chen and Chang, 2008). System identification is an approach for modeling a plant when physical laws are not enough to model the plant and input-output measurements are available only. Identification procedure is data driven and observations about the plant are very critical while a candidate model is chosen.

As a summary, the system identification procedure has three main steps;

- First; get the data,
- Second; define candidate models,
- Third; choose the best model.

2.2 System Modeling

In the analysis and design of control systems, it is necessary to have a mathematical model of the given plant. Such a mathematical model, called a simply model, must describe the system dynamics as completely as possible (Ogata, 1994).

In general, an accurate model may not be obtained by applying physical laws only. Then, making experiments is needed on the unknown plant. A mathematical model involving parameters is referred to as a parametric description of the plant (Ogata, 1994).

Every modeling technique does not contain an identification sub-procedure, which are collecting data from the plant, choosing a candidate model, estimating the parameters of the candidate model and validation with the real plant. For example finding a representative model for a plant by using physical laws is not a system identification task. System identification is considered a modeling approach but every modeling approach is not system identification. System identification procedure should be data driven. For example; from observation of the plant, a plant is said to be a LTI system by testing its time invariancy and linearity, then only getting the impulse response of the plant is enough for identification, but analyzing an electronic circuit with Kirchoff's Laws is not system identification. To analyze a control system and to design a controller which generates an appropriate control input, it is needed a model of the plant which works very near to the real life.

Mathematical models may be developed along two routes (or a combination of them). One route is to split up the system, into subsystems, whose properties are well understood from previous experience. This basically means that we rely on laws of nature and other well-established relationships that have their roots in earlier empirical work. These subsystems are then joined mathematically and a model of the whole system is obtained. This route is known as modeling and does not necessarily involve any experimentation on the actual systems. The other route, which is system identification, is directly based on experimentation. Input and output signals from the system are recorded and subjected to data analysis in order to infer a model (Ljung, 1987).

Modeling is a technique to introduce a mathematical model which is not too complex but very near to the reality. Too complex models can give better results for some situations but less complex models have good generalization ability and also working with less complex models gives us to design the system quickly with less computation time, and beside these benefits finding a simple model for the plant will cause a simple model for controller, this results a low cost controller and also system realizations.

Modeling could be generated in two ways;

- with the help of physical laws,
- by choosing a model, which has a global approximation property.

The first procedure can be applied when the plant can be modeled with physical laws, but the second one is used generally for gray box or black box systems. A white-box model is a system where all necessary information is available. Black box systems are systems which can be viewed only their inputs and outputs (Hellerstein, Diao, Parekh, & Tilbury, 2004). A black-box model is a system of which there is no information available. Only the inputs are given to the system and outputs from it are gotten then the internal dynamics of the system are tried to find. Practically all systems are somewhere between the black-box and white-box models, that type of models are grey-box models. Gray box systems are the combinations of white box and black box systems (Nelles, 2001; Sjoberg and others, 1995). In the second way of modeling, finding an appropriate model for the plant and estimating the correct parameters of the proposed model is the main task of the designers.

Mathematical models can be classified in several ways, some of which are described below (Kapur, 1988; Ugwa, 2012).

1. Linear versus nonlinear: In mathematical models variables are used. And these variables are used with operators. If all operators used in a mathematical model are linear, then it is said that mathematical model is linear. Otherwise if at least one of the operators are nonlinear it is said that mathematical model is nonlinear.

2. Deterministic versus probabilistic (stochastic): In a deterministic model, variables are described by unique values or they are determined by parameters in the model. If the parameter values, inputs and initial states are known, then the future of the system states and also the outputs are completely determined in a unique way. On the other hand, in a stochastic model variables are not described by unique values. They are described by probability distributions. So, the future of the system states and outputs can be specified with certain probabilities only.

3. Static versus dynamical: In a static or said algebraic model, the system outputs at a specific time instant depend only on the input at the same time instant. In contrast, the outputs of a dynamical system at a time instant can be defined with initial states summarizing the past/future inputs in addition to the input at the considered time instant. Discrete dynamic models are represented by difference equations and continuous time dynamical models are represented by differential equations.

4. Lumped parameters versus distributed parameters: If the model is a homogeneous model in terms of spatial coordinates, then the parameters are lumped, so is the model. If the model is a heterogeneous model, then the parameters and so the model are distributed. Distributed dynamical models are defined with the variables which are functions of spatial coordinates in addition to the time variable, so they are represented by partial differential equations in contrast to the lumped parameter models defined by ordinary differential equations.

One can choose a mathematical model for the plant to be identified as considering the above classification. During determining a mathematical model of a plant, it is important to choose a simple model that satisfies the specific objectives of the analysis and design. Then, the designer should try to identify the system with minimum number of parameters yielding a simple yet sufficient model. The stages of a typical system identification procedure are given in Figure 2.1.

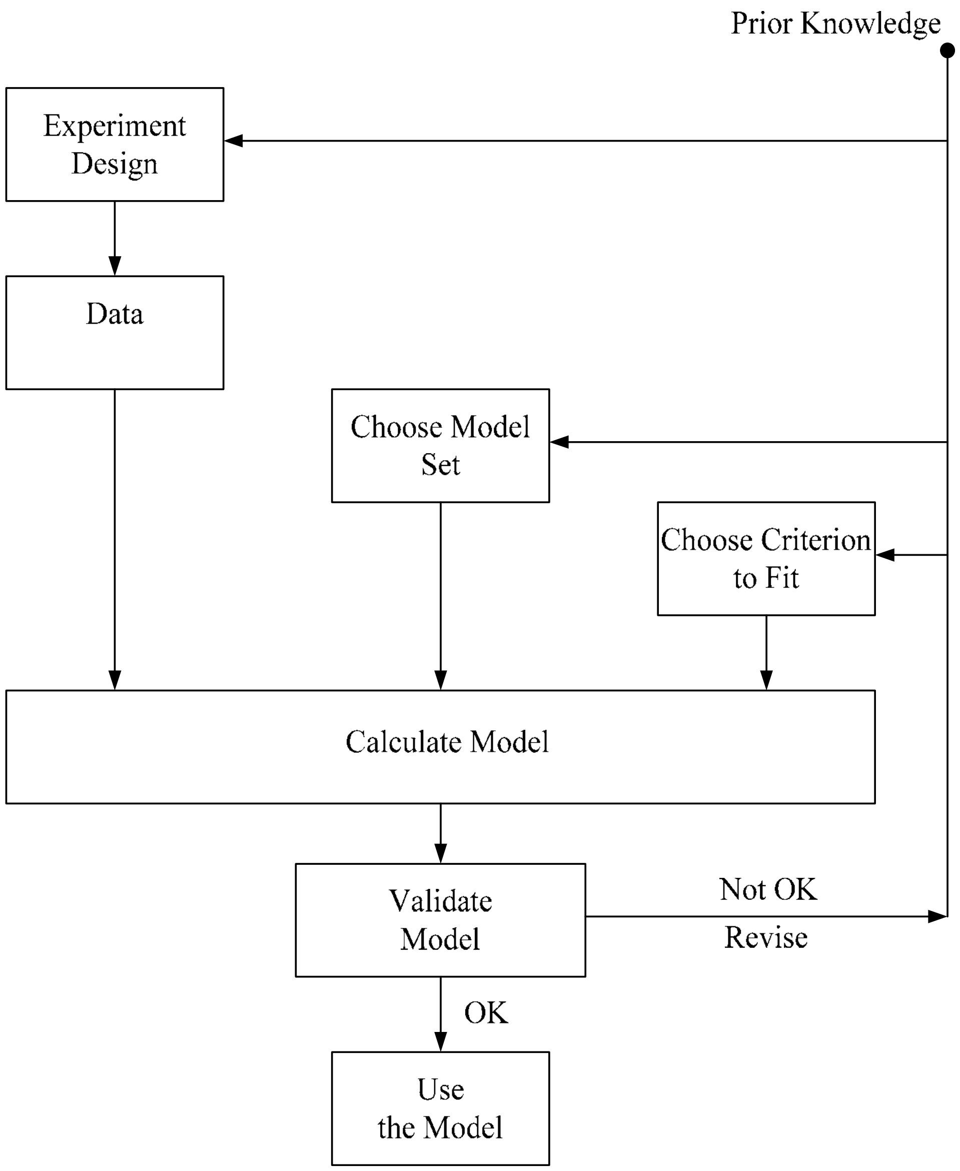

Figure 2.1 Loop of system identification (Ljung,1987)

2.2.1 System Identification by Least Squares

In mathematically formulating the identification problem, it must be used a function or performance index that will measure how well the model in question fits the experimental data. If it is chosen the performance index as to be the sum of error squares, then this identification method is called as the least squares method.

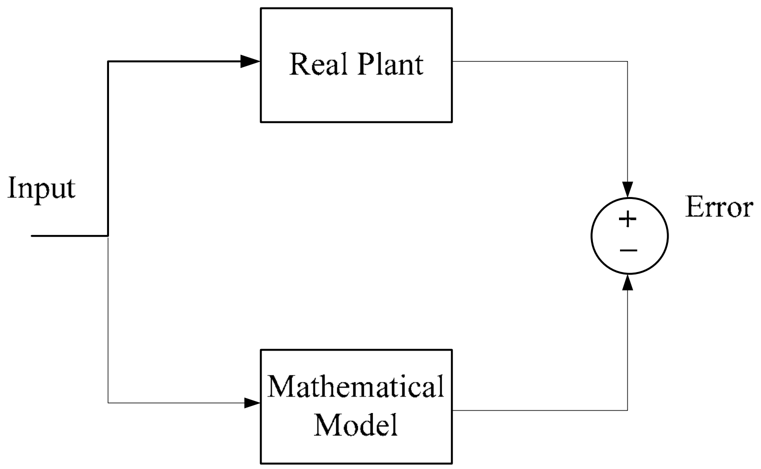

Figure 2.2 Error between real plant and mathematical model

When the real plant and its model (Ogata,1994) in Figure 2.2 are considered, it is assumed that the plant dynamics may be given by the following pulse transfer function for a discrete-time LTI model:

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots\dots\dots\dots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots\dots\dots\dots + a_n z^{-n}} \tag{2.1}$$

Where, the coefficients $a_1, a_2, \dots\dots\dots, a_n$ and $b_0, b_1, \dots\dots\dots, b_n$ of the denominator and numerator polynomials are the parameters of the system.

In the following, it is assumed that the completely known input sequence $u(0)$, $u(1)$, . . . $u(\mathrm{N})$ is applied to the plant and the corresponding plant output sequence $y(0), y(1), \dots, y(\mathrm{N})$ is observed.

Then, it is determined the least squares estimates of the system parameters $a_1, a_2, \dots\dots\dots, a_n$ and $b_0, b_1, \dots\dots\dots, b_n$. From equation (2.1) the output y(k) is estimated on the basis of y(k-1), y(k-2), . . . y(k-n) and u(k), u(k-1), ..., u(k-n) according to the following equation

$$\bar{y} = -a_1 y(k-1) - a_2 y(k-2) - .. - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + .. + b_n u(k-n) \tag{2.2}$$

Where $\bar{y}(k)$ is the estimated value of y(k). Error e(k) is defined as the difference between the actual output y(k) and the estimate output $\bar{y}(k)$

$$e(k) = y(k) - \bar{y}(k)$$

$$= y(k) + a_1 y(k-1) + a_2 y(k-2) + .. + a_n y(k-n) - b_0 u(k) - b_1 u(k-1) - .. - b_n u(k-n) \tag{2.3}$$

The error e(k) depends on the measured values of y(k), y(k-1), ... , y(k-n) and u(k), u(k-1),. . . . . , u(k-n). Equation (2.3) can be written as follows:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - .. - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + .. + b_n u(k-n) + e(k) \tag{2.4}$$

and also in a vector matrix form as:

$$\begin{bmatrix} y(n) \\ y(n+1) \\ . \\ . \\ y(N) \end{bmatrix} = \begin{bmatrix} y(n-1) & y(n-2) & . & . & y(0) & u(n) & u(n-1) & . & . & u(0) \\ y(n) & y(n-1) & . & . & y(1) & u(n+1) & u(n) & . & . & u(1) \\ . & . & & & . & . & . & & & . \\ . & . & & & . & . & . & & & . \\ y(N-1) & y(N-2) & . & . & y(N-n) & u(N) & u(N-1) & . & . & u(N-n) \end{bmatrix} . \begin{bmatrix} -a_1 \\ -a_2 \\ . \\ . \\ -a_n \\ b_0 \\ b_1 \\ . \\ . \\ b_n \end{bmatrix} + \begin{bmatrix} e(n) \\ e(n+1) \\ . \\ . \\ e(N) \end{bmatrix} \tag{2.5}$$

One defines

$$y(N)=\begin{bmatrix} y(n) \\ y(n+1) \\ . \\ . \\ y(N) \end{bmatrix}, \quad e(N)=\begin{bmatrix} e(n) \\ e(n+1) \\ . \\ . \\ e(N) \end{bmatrix}, \quad x(N)=\begin{bmatrix} -a_1(N) \\ -a_2(N) \\ . \\ . \\ -a_n(N) \\ b_0(N) \\ b_1(N) \\ . \\ . \\ b_n(N) \end{bmatrix}$$

$$C(N)=\begin{bmatrix} y(n-1) & y(n-2) & . & . & y(0) & u(n) & u(n-1) & . & . & u(0) \\ y(n) & y(n-1) & . & . & y(1) & u(n+1) & u(n) & . & . & u(1) \\ . & . & & & . & . & . & & & . \\ . & . & & & . & . & . & & & . \\ y(N-1) & y(N-2) & . & . & y(N-n) & u(N) & u(N-1) & . & . & u(N-n) \end{bmatrix} \tag{2.6}$$

then y(N) can be written as

$$y(N)=C(N)x(N)+e(N) \tag{2.7}$$

Defining performance index (cost) as

$$J_N=\frac{1}{2}\sum_{k=n}^{N}e^2(k)=\frac{1}{2}e^T(N)e(N) \tag{2.8}$$

Then, the problem becomes to determine *x(N)* such that the parameter values of $a_1, a_2, ..., a_n$ and $b_0, b_1, ..., b_n$ will best fit the observed data. x(N) can be found by minimizing the performance index

$$J_N=\frac{1}{2}e^T(N)e(N)=\frac{1}{2}[y(N)-C(N)x(N)]^T[y(N)-C(N)x(N)]$$

$$=\frac{1}{2}[-x^T(N)C^T(N)+y^T(N)][y(N)-C(N)x(N)]$$

$$= \frac{1}{2}\left[-x^T(N)C^T(N)y(N) + y^T(N)y(N) + x^T(N)C^T(N)C(N)x(N) - y^T(N)C(N)x(N) \right] \tag{2.9}$$

To do that, one may get the following normal equations obtained in terms of the partial derivatives with respect to x(N),

$$\frac{\partial J_N}{\partial x(N)} = C^T(N)C(N)x(N) - C^T(N)y(N) = 0 \tag{2.10}$$

$$C^T(N)C(N)x(N) = C^T(N)y(N) \tag{2.11}$$

By assuming $C^T(N)C(N)$ is nonsingular, the inverse of $C^T(N)C(N)$ exists. Then, the optimal parameters can be found as;

$$x(N) = \left[C^T(N)C(N)\right]^{-1} C^T(N)y(N) \tag{2.12}$$

2.2.2 System Identification Using State Space Models

Suppose that the given plant is described by the discerete-time state space model (Haykin, 1999).

$$\begin{aligned} x(n+1) &= f(x(n), u(n)) \\ y(n) &= h(x(n)) \end{aligned} \tag{2.13}$$

Where, the output equation and the state equation are defined by vector valued nonlinear functions $h(\bullet)$ and $f(\bullet,\bullet)$, respectively.

One may employ two neural networks to identify the system, one for dealing with the state equation and the other for dealing with the output equation. Let $\hat{x}(n+1)$ denote the estimate of $x(n+1)$ produced by the first neural network labeled Network 1 in Figure 2.4. This network operates on a concatenated input consisting of the external input $u(n)$ and the state $x(n)$ to produce $\hat{x}(n+1)$. The estimate $\hat{x}(n+1)$ is subtracted from the actual state $x(n+1)$ to produce the error vector:

$$e_1(n+1) = x(n+1) - \hat{x}(n+1) \tag{2.14}$$

Where, $x(n+1)$ which is the actual state of the plant constitutes the desired output for Network 1.

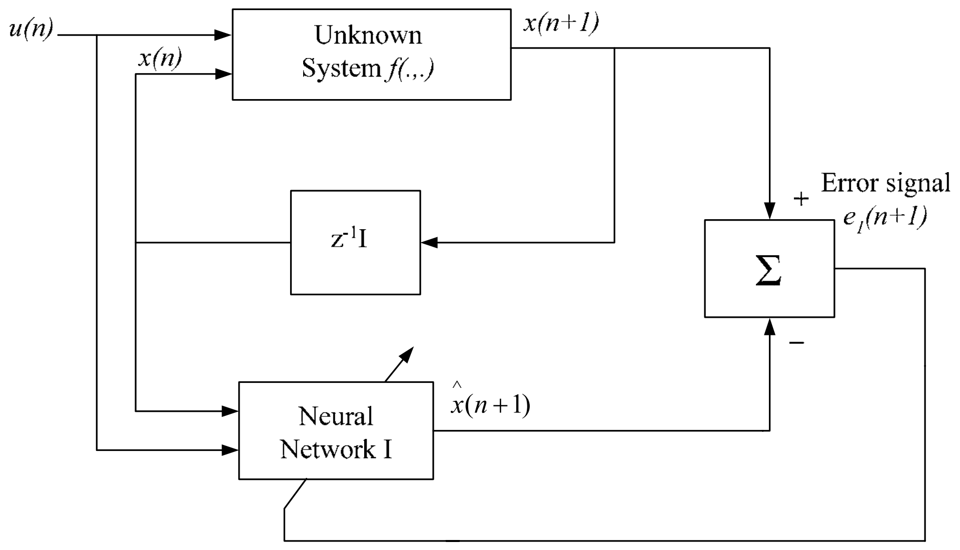

Figure 2.3 Training Network I used for the identification of the state function

The error is used to adjust the synaptic weights of the neural network in order to minimize the cost function based on the error vector $e_1(n+1)$.

The second neural network labeled as Network 2 operates on the actual state $x(n)$ of the unknown plant to produce an estimate $\hat{y}(n)$ of the actual output $y(n)$. The estimate $\hat{y}(n)$ is subtracted from $y(n)$ to produce the second error vector.

$$e_2(n) = y(n) - \hat{y}(n) \tag{2.15}$$

Where, $y(n)$ plays the role of desired response for the neural network. The error vector $e_2(n)$ is used to adjust the synaptic weights of Network 2 to minimize the Euclidean norm of the error vector.

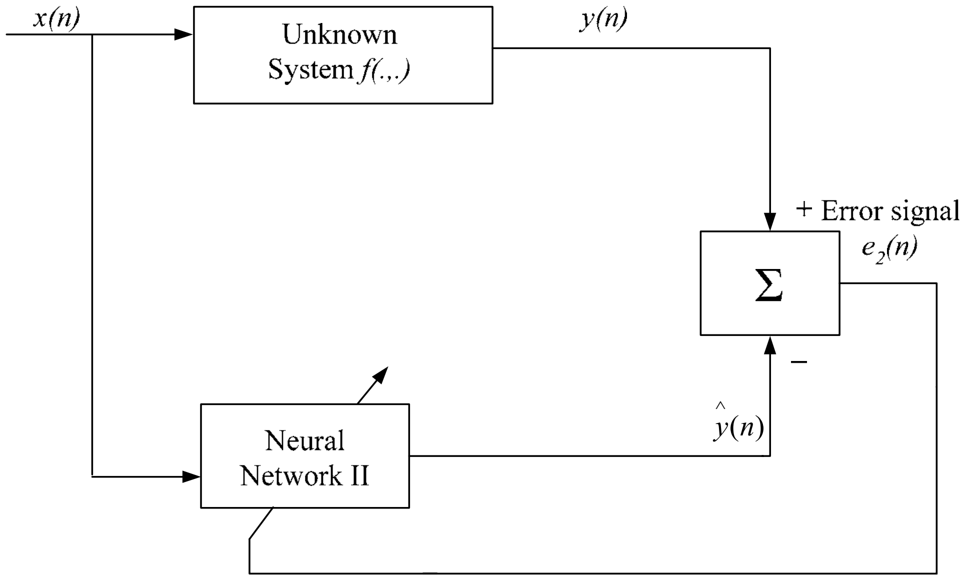

Figure 2.4 Training Network II for learning the output function

2.2.3 ARMA Model

ARMA model structures are the well known linear model structures. ARMA models are very widely used models for plants which are to be used for identification, control, communication, power systems, biomedical engineering, signal processing areas (Wies, Pierre, & Trudnowski, 2003; Hernandez & Arkun, 1993; Daniel & William, 1975; Perrott & Cohen, 1996; Chon & Cohen, 1997; Imer & Basar, 1999; Nikias & Mendel, 1993).

An ARMA model consists of two parts: Auto-regressive and moving-average**.**

$$\frac{y[z]}{u[z]} = \frac{b_1 z^{-1} + \ldots\ldots + b_m z^{-m}}{1 + a_1 z^{-1} + \ldots\ldots + a_n z^{-n}} \tag{2.16}$$

To include noise effects to linear system some exogenous inputs are applied to the linear system. Then, the system becomes ARX or ARMAX.

2.3 Nonlinear Blocks

2.3.1 Neural Networks

Neural networks are widely used in describing the nonlinear behaviors of the systems. Due to their approximation capabilities, algebraic neural networks such as multi layer perceptron and radial basis function network are widely used in defining especially next outputs in terms of current outputs and inputs for system identification purposes. Such an identification scheme is described below.

Assume an unknown plant which is only accessible through its output. Let the system be of a single input, single output. Let *y*(n) denotes the output of the system due to the input *u*(n) for varying discrete time n. Then, choosing to work with the NARX model, the identification model takes the form.

$$\hat{y}(n+1) = \varphi(y(n),...,y(n-q+1),u(n),...,u(n-q+1)) \tag{2.17}$$

Where, *q* is the order of the unknown system. At time *n+1*, the *q* past values of the input and the *q* past values of the output are all available. The model output $\hat{y}(n+1)$ represents an estimate of the actual output *y*(n+1). The estimate $\hat{y}(n+1)$ is subtracted from *y*(n+1) to produce the error signal.

$$e(n+1) = y(n+1) - \hat{y}(n+1) \tag{2.18}$$

Where *y*(n+1) plays the role of the desired response. The error *e*(n+1) is used to adjust the synaptic weights of the neural network so as to minimize the error. The identification model in Figure 2.5 (Haykin, 1999) is of a series parallel form because the actual output of the system is fed back to the input of the model.

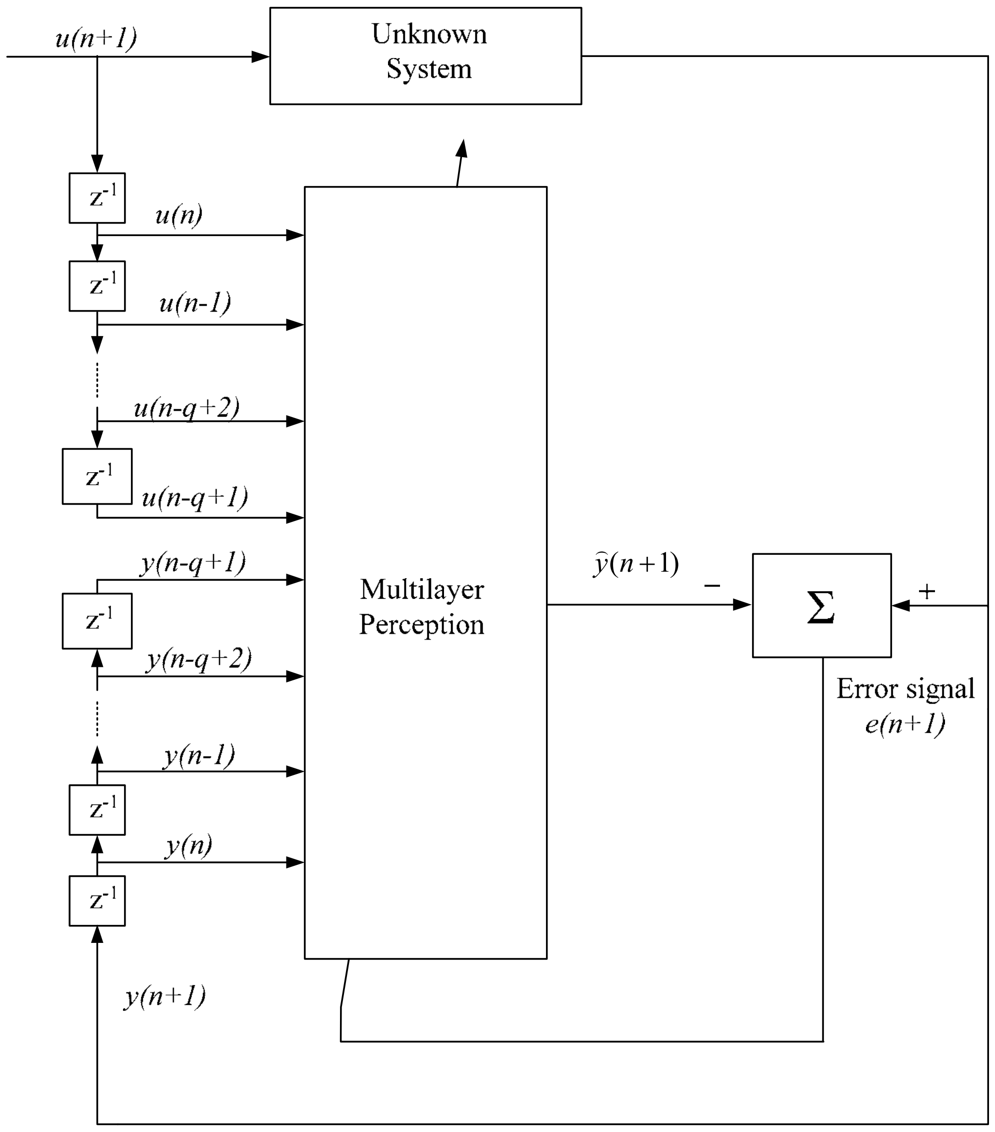

Figure 2.5 Multi layer neural network used for identification

2.3.2 Power Series

Power series is the powers of a variable such as;

$$P_i(x) = x^i \tag{2.19}$$

where *i* is from 0 to N. They are used in identification for defining static blocks.

2.3.3 B-Splines

Splines are piecewise polynomials used to approximate unknown functions from data. By adding n’th order splines with breakpoints a nonlinear function can be fit. A first order spline is a piecewise constant function, second order spline is a piecewise linear function, and a third order spline is a quadratic function (Moran, Agamennoni, Figueroa, 2005).

2.3.4 Chebyshev Polynomials

Chebyshev polynomial basis functions which are given below are also used as defining static blocks in identification.

$$H_i(x) = \cos(i \arccos x) \tag{2.20}$$

2.3.5 Wavelets

The wavelet basis functions consist of mother wavelets. A basis function can be written as;

$$B_{i,j}(x) = \frac{1}{\sqrt{2^i}} \psi\left(\frac{x - 2^i j}{2^i}\right) \tag{2.21}$$

Where, ψ is the mother wavelet. They are especially useful for nonstationary measurement data.

2.4 Input-Output Representations for Nonlinear Systems

2.4.1 Volterra Series Representation

Volterra mapping and Volterra series representation is one of the most important input-output nonlinear analytic mapping of nonlinear dynamical systems.

Volterra mapping is a nonlinear mapping $V(\bullet): X \to Y$ transforming an input function $x(\bullet) \in$ X into an output function $y(\bullet) \in$ Y. Then, an operator $V_n(\bullet): X \to Y$ can be defined as:

$$V_n(\bullet)(t) = \int_0^{t_1} ... \int_0^{t_n} K_n(t; t_1, ..., t_n)(\bullet)(t_1)...(\bullet)(t_n) dt_1 ... dt_n \tag{2.22}$$

Where, $K_n(t; t_1, ..., t_n)$ are the Volterra kernels of the operator $V_n(\bullet)$.

By using the Volterra operator above, one can write the Volterra mapping from $x(\bullet) \in$ X into $y(\bullet) \in$ Y as:

$$y(t) = V_n(x)(t) = \int_0^{t_1} ... \int_0^{t_n} K_n(t; t_1, ..., t_n)(x)(t_1)...(x)(t_n) dt_1 ... dt_n \tag{2.23}$$

Volterra series is a representation which consists of infinite sum of Volterra operator. Volterra series representation is given as:

$$V(\bullet) = \sum_{n=0}^{\infty} V_n(\bullet)(t) = V_o(t) + \sum_{n=1}^{\infty} \int_0^{T} ... \int_0^{T} K_n(t; t_1, ..., t_n)(\bullet)(t_1)...(\bullet)(t_n) dt_1 ... dt_n \tag{2.24}$$

If $V_n \equiv 0$ for all n different than 1, then the Volterra mapping $V(\bullet)$ becomes linear (Defigueiredo & Chen, 1993).

The output of Volterra series is a sum of zeroth, first, second and higher order models.

$$Y[x(n)] = Y_0[x(n)] + Y_1[x(n)] + ... + Y_n[x(n)] \tag{2.25}$$

The zeroth order Volterra model is a constant,

$$Y_0[x(n)] = H_0 \tag{2.26}$$

When the first order Volterra model like a linear system is considered, the output $y_1(n)$ is the linear convolution of the input with $h_1(n)$ which is;

$$Y_1(n) = x(n) * h_1(n) = \sum_{k=0}^{\infty} h_1(k)x(n-k) \tag{2.27}$$

When the Volterra series are extended as the combination of zeroth, first and second order Volterra models the equation in discrete time becomes (Ogunfunmi, 2007) as:

$$y(n) = h_0 + \sum_{k_1=0}^{\infty} h_1(k_1)x(n-k_1) + \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} h_2(k_1,k_2)x(n-k_1)x(n-k_2) \tag{2.28}$$

Volterra series can give the system response when input is applied to the Volterra series representation if the kernels are chosen correctly.

2.4.2 Hammerstein Models

Another input-output representation of nonlinear systems is Hammerstein model. Hammerstein model of any nonlinear system is composed of a nonlinear static block and a linear dynamic block (Chou, 2006). The Hammerstein model structure is shown in Figure 2.7, where $F(\bullet)$ is the nonlinear static function which represents the nonlinear characteristics of the model and $G(\bullet)$ is the dynamic linear function which represents the dynamic behaviour of the model.

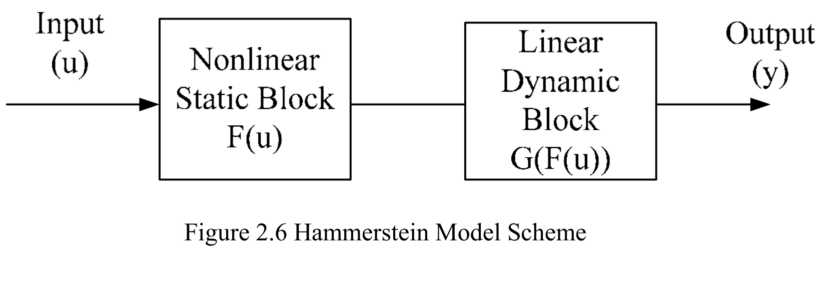

Figure 2.6 Hammerstein Model Scheme

Hammerstein models are widely used in identification of nonlinear systems, such as in control systems (Kung & Womack 1984), in identification of nonlinear systems

(Eskinat, Johnson, & Luyben, 1991), in nonlinear data transmission channels (Maqusi, 1985) and in many other nonlinear system applications.

The nonlinear static block may be modeled by an artificial neural network or a piecewise linear model and the linear dynamic block may be modeled with an autoregressive-moving average model (Alduwaish & Karim, 1997; Hatanaka & Uosaki, 2001; Alduwaish, Karim, & Chandrasekar, 1997; Paoletti, 2003).

2.4.3 Wiener Models

Wiener model of any nonlinear system is composed of a linear dynamic block and a static nonlinear block (Moran, Agamennoni, & Figueroa, 2005). The Wiener model structure is shown in Figure 2.7 where the static block comes after the linear dynamical block as oppose to the Hammerstein model.

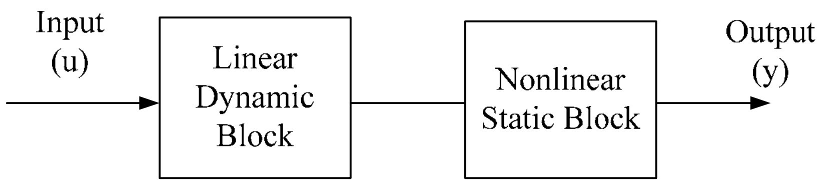

Figure 2.7 Wiener Model Scheme

2.4.4 Hammerstein - Wiener Models

Hammerstein-Wiener model contains Hammerstein and Wiener models together (Hu, 2011).

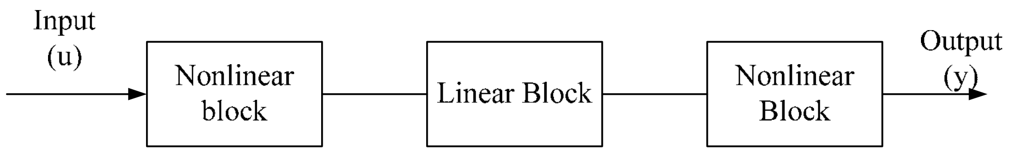

Figure 2.8 Hammerstein- Wiener model scheme

2.4.5 Prediction Error Method

Prediction error methods constitute a family of parameter estimation methods. Let measurements from the system input and output be denotes as u and y, respectively. $Z^N = \{u(1), y(1), u(2), y(2), \ldots, u(N), y(N)\}$ denote the input output pairs (Ljung, 2002).

The basic idea in prediction error method is describing the model as a predictor of the next output as in Equation (2.29).

$$\hat{y}_m(t|t-1) = f(Z^{t-1}) \tag{2.29}$$

$\hat{y}_m(t|t-1)$ denotes the one step ahead prediction of the output and f is an arbitrary function of past observed data.

Then, one may parameterize the predictor in terms of parameter vector in Equation (2.30).

$$\hat{y}_m(t|\theta) = f(Z^{t-1}, \theta) \tag{2.30}$$

Prediction error methods can handle systems that operate in closed loop. However, they require an explicit parametrization of the model.

2.4.6 State Space Model with Subspace Method

State space representations for a linear time invariant discrete time system can be given as follows.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \\ y_k &= Cx_k + Du_k + v_k \end{aligned} \tag{2.31}$$

Where, u(k) and y(k) are, respectively, the input and output measurements, x(k) is the state sequence, w(k) and v(k) are the unmeasurable sequences representing noise which are assumed as zero mean white noise sequences. A is the dynamical system matrix, B is the input, C is the output matrix, D represents the relation between input and output directly. Subspace method enables to determine a state space representation of a system.

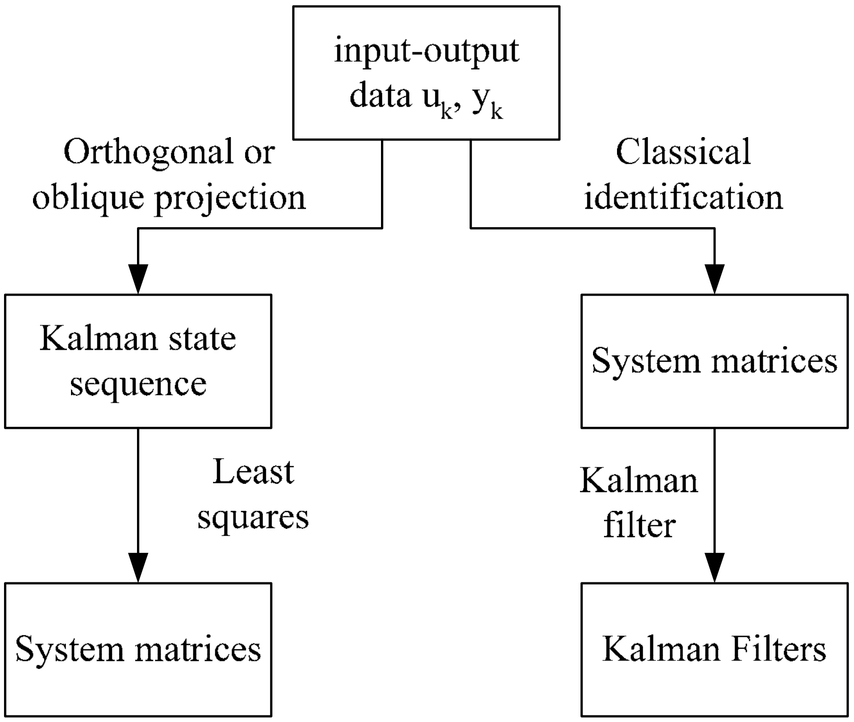

Figure 2.9 Subspace method versus classical approach (Overschee & Moor,1996)

Figure 2.9 shows the difference between classical approach and subspace method. In the subspace method, Kalman state sequence can be developed from the input output measurements using QR-decomposition or singular value decomposition. Then, the system matrices can be obtained with the least squares method. The classical approach firstly obtains the system matrices then using Kalman filter to compute the states. On the other hand, the subspace method first determines a state sequence by using projection into the subspace, then provide system matrices.

2.4.7 Elman Network

Elman network which is a neural network having feedback connection is a type of recurrent neural network. It is suitable for dynamic system modeling. Figure 2.10 illustrates its architecture. It consists of input, output, hidden, and context units. Context units memorize the previous activations of hidden units (Pham & Liu, 1991).

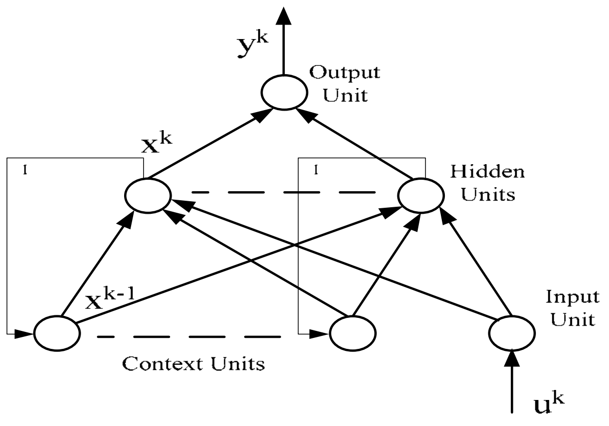

Figure 2.10 Architecture of Elman Network (Pham & Liu, 1991)

2.5 Chaos Control and Anti-Control

Chaos is known as a dynamical behavior of a system which is sensitive to initial conditions and having a topological transitivity property, so that it is usually an undesired property for systems of practical interest. So, chaos is tried to be suppressed in any way, for instance by chaos control methods (Chen, 1998).

However, in some systems including chemical reactions, biological and economic and in some applications such as liquid mixing and secure communication, chaos is a desirable property. So, anti-control (i.e. chaotification) of an originally nonchaotic system may be the control objective rather than suppressing the chaos (Chen, 1998; Wang & Chen, 2000; Wang, Chen, & Man, 2001; Lü, Zhou, Chen, & Yang, 2002).

CHAPTER THREE
LEARNING STATE SPACE WITH FORCED CHAOTIC STATE RECURSION

This chapter presents a method for identification of a system which might be linear or nonlinear. The method presented named as Learning State Space with forced Chaotic State Recursion (LSS-CSR) is used to construct a state equation model of the system based on input output data. The idea behind the method is that, a chaotic signal and white Gaussian noise includes almost all possible state recursion under the force by the system input. It will be seen that usage of chaotic states in the identification of state equation gives a better performance compare to the other well known methods also to the method using white Gaussian noise in training the state equations.

(LSS-CSR) employs two artificial neural networks, one is for the construction of the state equation the other is for estimating the output equation. Four benchmark plants (one is linear, the other is chaotic and the other two are nonlinear) are used for testing the performance of the method. (LSS-CSR) is used to identify the linear and nonlinear plants with different input types, the chaotic one which is forced duffing equation, uses the input which is forcing the system. Four different inputs are applied to linear and nonlinear plants which are step, chaotic and sinoisoidal with two different frequencies. The method is also tested with different initial conditions for the benchmark plants. In all cases signal to error ratio and normalized mean square error are calculated for both evaluating training and test performances of the identification method. The proposed methods are also compared with the benchmark identification methods, Nonlinear ARX, Nonlinear ARMAX, Hammerstein Model, Wiener Model, Hammerstein-Wiener Model, State Space Model with Subspace Method, State Space Model with Prediction Error Method and Elman Network. Noise performance of the proposed method is evaluated at the last stage and their performance are also compared with the benchmark models and methods stated above.

3.1 Learning State Space with forced Chaotic State Recursion

The first method which will be called as (LSS-CSR) deals with the identification of a linear or nonlinear plant which is given as a black box defined with input output data. The aim of this method is that we need to construct the nonlinear state space equation which is given with the Equation (3.1).

$$\begin{aligned} x^{k+1} &= f(x^k, u^k) \\ y^k &= h(x^k, u^k) \end{aligned} \tag{3.1}$$

The method consists of two artificial neural network structures named State ANN and Output ANN. State ANN is constructed to estimate the states of the plant and Output ANN is used for estimating the output of the plant. The block diagram of the (LSS-CSR) is shown in Figure 3.1.

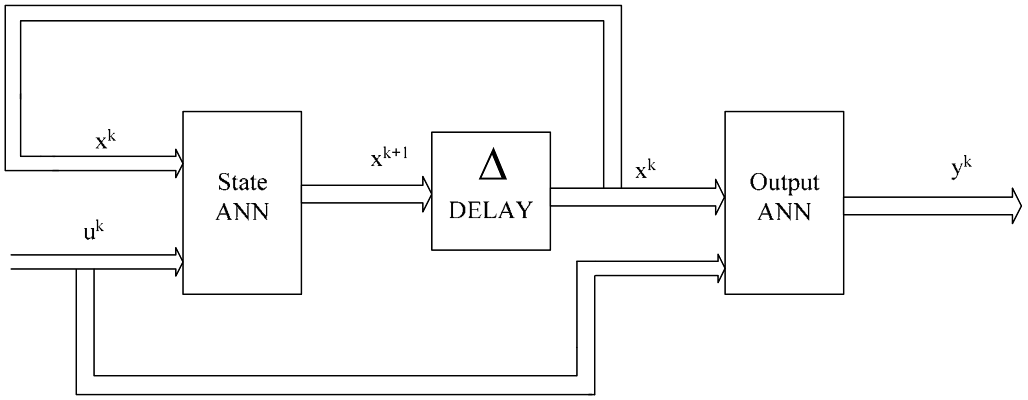

Figure 3.1 General block diagram of the proposed method LSS-CSR

The train and test phases of the proposed method LSS-CSR is as follows. The key point of the method is that white gaussian noise consists of all states for a linear or nonlinear plant. Then the first step is to assign the states x^{k+1}'s as with a white gaussian noise. Then with the known inputs u^k's and the assigned delayed states which are white gaussian noise train the State ANN to learn x^{k+1}'s. After training of the State ANN take the State ANN fixed and train the Output ANN with the known inputs u^k's, and the delayed state ANN outputs to learn the known outputs y^k's. We can summarize the method step by step as follows:

- First step is assigning the states x^{k+1}'s as with a white gaussian noise, if the system to be identified is chaotic then use a known chaotic system states.
- Train the State ANN to learn x^{k+1}'s with known inputs u^k's and the assigned delayed states.
- Take the State ANN fixed and train the Output ANN with the known inputs u^k's, and the delayed state ANN outputs to learn the known outputs y^k's.

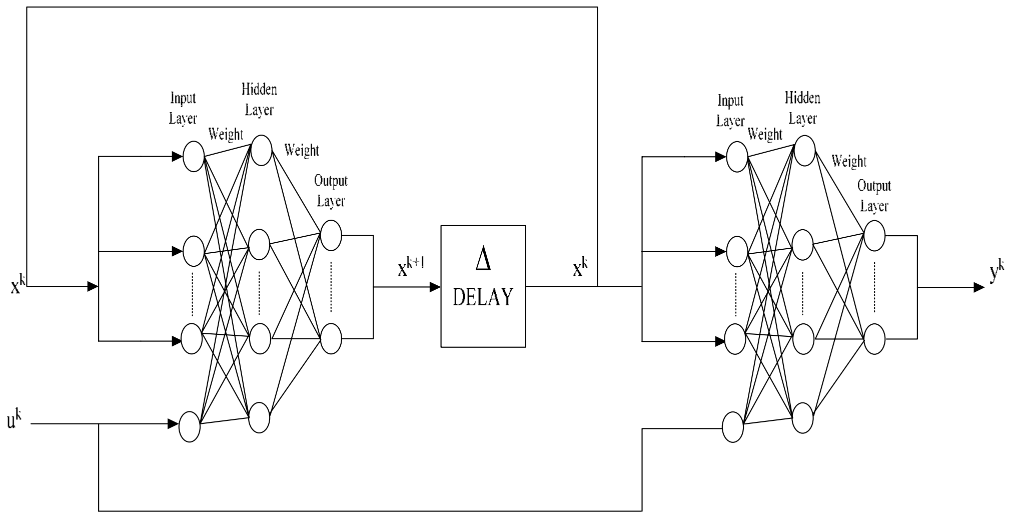

Figure 3.2 Artifical neural network structure of proposed method LSS-CSR

In all simulations of LSS-CSR method the State ANN have 3 input neurons, one for the system input and two others for the current value of 2 state variables, and to have 2 output neurons each of which represents the next value of one of 2 state variables. The output ANN has 3 input neurons again one for the system input and two others for the current value of 2 state variables, and it has a single output neuron representing the system output. Both ANNs are trained with the gradient descent type back propagation with the momentum term 0.9 and with a constant learning 0.05 for tansig (i.e. the hyperbolic tangent sigmoid) activation function in the input and hidden layers, linear transfer function in the output layer. 3 hidden neurons are observed to be sufficient for all of the state, and output ANNs and so the identification results are obtained for 3 hidden neuron ANNs in all experiments

3.2 Benchmark Plants in Proposed Method

The performance of the proposed method LSS-CSR is tested with some benchmark plants. Plant I, Plant II, Plant III and Plant IV are stated in Table 3.1 with the references where the plant dynamics are borrowed.

Table 3.1 Benchmark plants used for testing the proposed method LSS-CSR

Benchmark Plant	**State Equations**	**Plant Description and Reference**
Plant I	$x_1(k+1)=0.5x_2(k)$ $x_2(k+1)=0.5x_1(k)+0.1x_2(k)+(1+v(k))$ $y(k)=x_1(k)+x_2(k)$	Second order plant, *u(k)* and *v(k)* are control inputs, *y(k)* is the output. (Narendra, 1996)
Plant II	$x(k+1)=10\sin(x(k))+u(k)[0.1+\cos(x(k)u(k))]$ $y(k)=0.025x(k)+0.5$	First order plant, *u(k)* is the control input and *y(k)* is the output. (Narendra, 1996)
Plant III	$x(k+1)=\dfrac{9.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)}$ $y(k)=x(k)+1$	*u(k)* is the control input and *y(k)* is the output (Uykan & others, 2000)
Plant IV	$\ddot{x}=7.6\cos(t)-0.04\dot{x}-x^3$	Forced Duffing Oscillator (Chen, Chen, & Öğmen, 1997)

Plant I which is a modified form of plant (Narendra,1996), is a second order linear state model which is having two states and the output equation is independent from the output. Plant II is the modified form of the plant in the literature (Narendra, 1996) which is a first order nonlinear state model having one state and output. Plant III is a modified form of the plant given in the literature (Uykan, Güzeliş, Çelebi, & Koivo, 2000) which is a third order nonlinear difference equation model. Plant IV is a modified version of the chaotic system in the literature (Chen, Chen, & Öğmen, 1997) known as the forced duffing oscillator.

The performance of the method is compared with some benchmark models which are stated below with their properties:

- **Hammerstein Model:** It uses 10 piecewise linear blocks at the nonlinear block, and uses a linear equation $y[z]=\frac{B[z]}{F[z]}u[z]z^{-n}$ where *B* representing the order of zeros, *F* representing the order of poles and *n* is the input delay. In the linear block the parameters *B*, *F*, and *n* are taken as 2, 3, and 1, respectively.

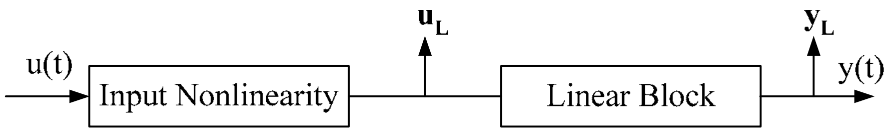

Figure 3.3 Structure of Hammerstein model

- **Wiener Model:** It uses 10 piecewise linear blocks at the nonlinear block, and uses a linear equation $y[z]=\frac{B[z]}{F[z]}u[z]z^{-n}$ where *B* representing the order of zeros, *F* representing the order of poles and *n* is the input delay. In the linear block the parameters *B*, *F*, and *n* are taken as 2, 3, and 1, respectively.

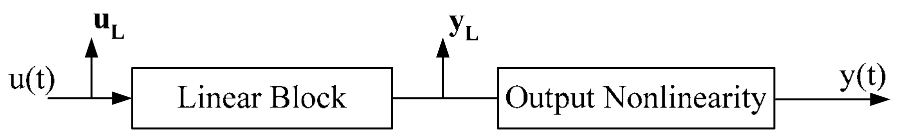

Figure 3.4 Structure of Wiener model

- **Hammerstein-Wiener Model:** Uses 10 piecewise linear blocks at the input and output nonlinear blocks, and uses a linear equation $y[z] = \frac{B[z]}{F[z]} u[z] z^{-n}$ where B representing the order of zeros, F representing the order of poles and n is the input delay. In the linear block the parameters B, F, and n are taken as 2, 3, and 1, respectively.

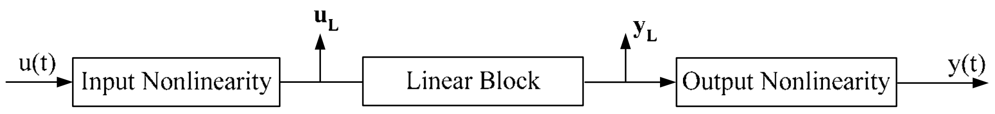

Figure 3.5 Structure of Hammerstein-Wiener model

- **Nonlinear ARMAX Model:** Four regressors (u(t-1), u(t-2), y(t-1), y(t-2)) and as a nonlinear block 10 sigmoid networks are used.

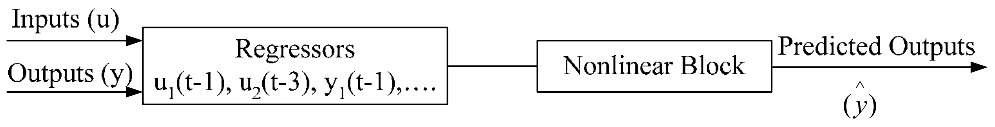

Figure 3.6 Structure of nonlinear ARMAX model

- **Nonlinear ARX Model:** Two regressors (y(t-1), y(t-2)) and as a nonlinear block 10 sigmoid networks are used

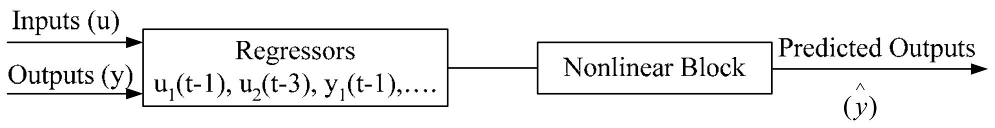

Figure 3.7 Structure of nonlinear ARX model

- **State Space Model with Prediction Error Method:** A linear state space model using prediction error method is used as in the equation below with the order of two.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{3.2}$$

- **State Space Model with Subspace Method:** A linear state space model using subspace method is used as in the equation below with the order of two.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{3.3}$$

- **Elman Network:** The elman network which is used have a one hidden layer which is having 3 neurons.

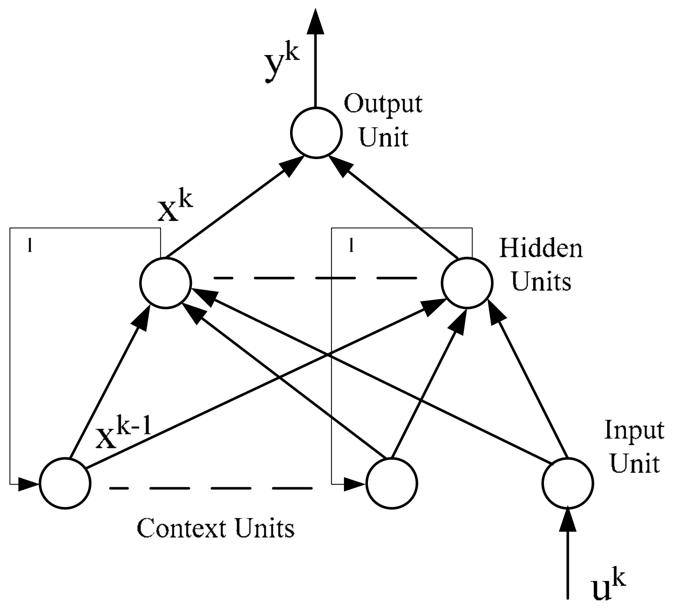

Figure 3.8 Architecture of Elman Network (Pham & Liu, 1991)

3.3 Simulation Results of the Proposed Method LSS-CSR

3.3.1 Case 1: A second order linear state model

In order to test the proposed method LSS-CSR, the modified form of the benchmark plant in the literature is borrowed (Narendra, 1996) and given in Equation (3.4).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (1+v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{3.4}$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are white gaussian noise with the power of 1dBW, 1 of them is the input which is *v*(k)), 3 hidden layer neurons and 2 output neurons which are used for learning the states. The Output ANN has 3 input neurons which are states (2 states) and the plant input *v*(k). Like the State ANN, the Output ANN has 3 hidden neurons.

The method is tested with the structure above and with 4 different types of input (*v(k)*) cases listed in Table 3.2.

Table 3.2 Inputs used for testing method LSS-CSR with plant in Equation (3.4)

Input I	*v(k)*=step input
Input II	*v(k)* = sin(k)
Input III	*v(k)*= sin(20k)
Input IV	*v(k+1)*=4*v(k)*(1-*v(k)*) , *v(0)*=0.1 (Chaotic input)

For all cases, the method performance is tested with the Mean Square Error (MSE) and, Signal to Error Ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized Mean Square Error (nMSE) and Signal to Error Ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial conditions are also fixed to constant values $x_1(0)$, $x_2(0)$ are both equal to 0.5. The algorithm ran 100 times to calculate SER values to find out the best, worst and the average performances of the method LSS-CSR. Table 3.3 shows the performance of the proposed method LSS-CSR to the plant I, nMSE and SER values for both training and test phases are shown. Tables from 3.3 to 3.11 show the performance of the benchmark methods and proposed method LSS-CSR to the plant I shown in Equation (3.4). Figures from 3.9 to 3.12 also show the identified output results versus desired outputs according to the different applied inputs. Comparison of outputs for different methods versus desired output for chaotic input with plant I is illustrated in Figure 3.13.

y_i^p is assumed the predicted output of the desired output y. Then the mean square error is calculated with the Equation 3.5 below

$$\text{mean square error (mse)} = \frac{1}{n}\sum_{i=1}^{n}(y_i^p - y_i)^2 \tag{3.5}$$

Normalized mean square errors are calculated as in Equation 3.6

$$\text{Normalized mean square error (nmse)} = \frac{\text{mean square error}}{\text{desired output signal power}} \tag{3.6}$$

$$\text{signal to error ratio (SER)} = 10\log_{10}\frac{\text{mean square of desired signal}}{\text{mean square Error (mse)}} \tag{3.7}$$

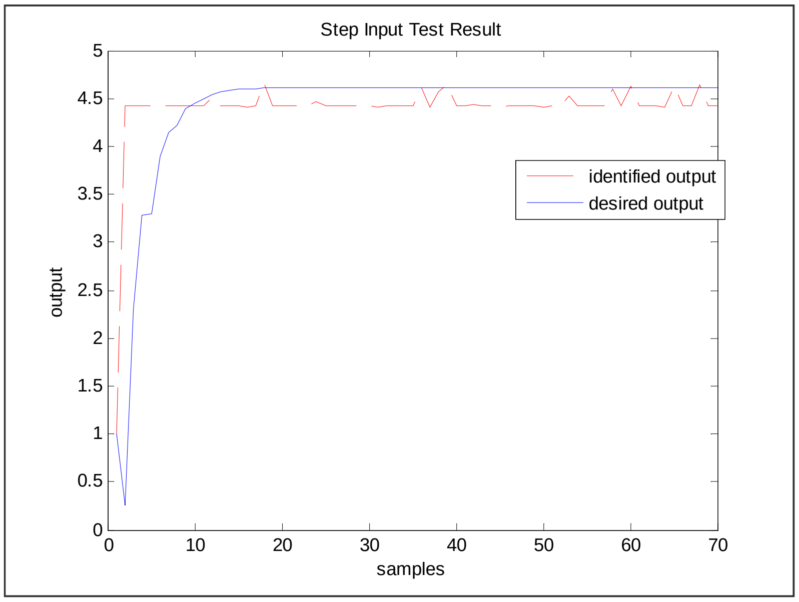

Figure 3.9 LSS-CSR method output versus desired output for step input with plant I

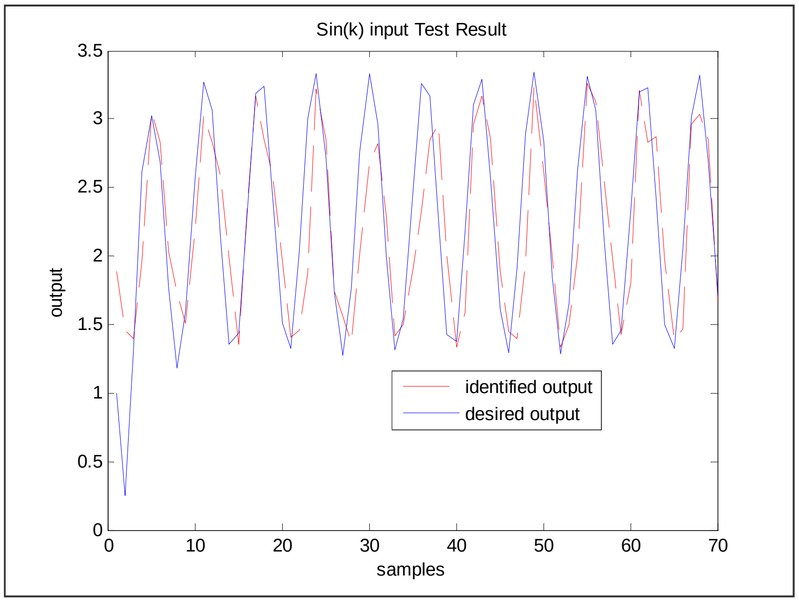

Figure 3.10 LSS-CSR method output versus desired output for sin(k) input with plant I

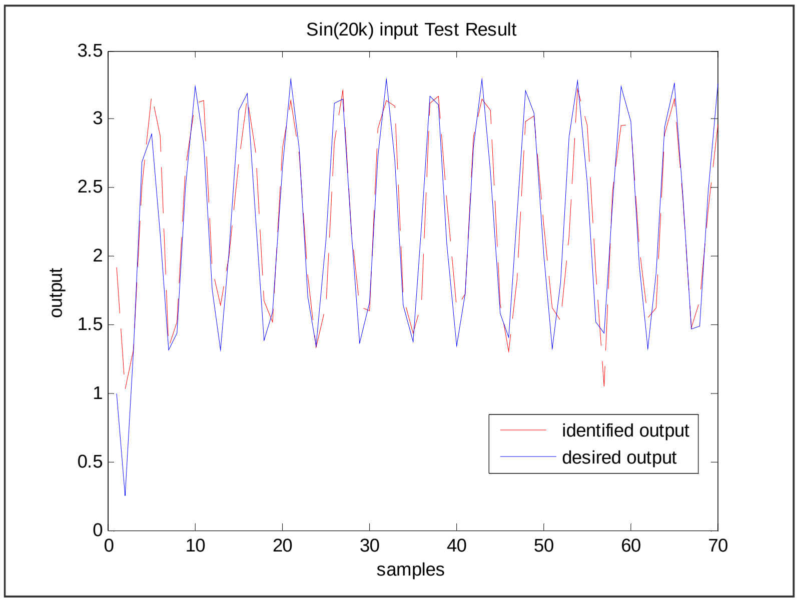

Figure 3.11 LSS-CSR method output versus desired output for sin(20k) input with plant I

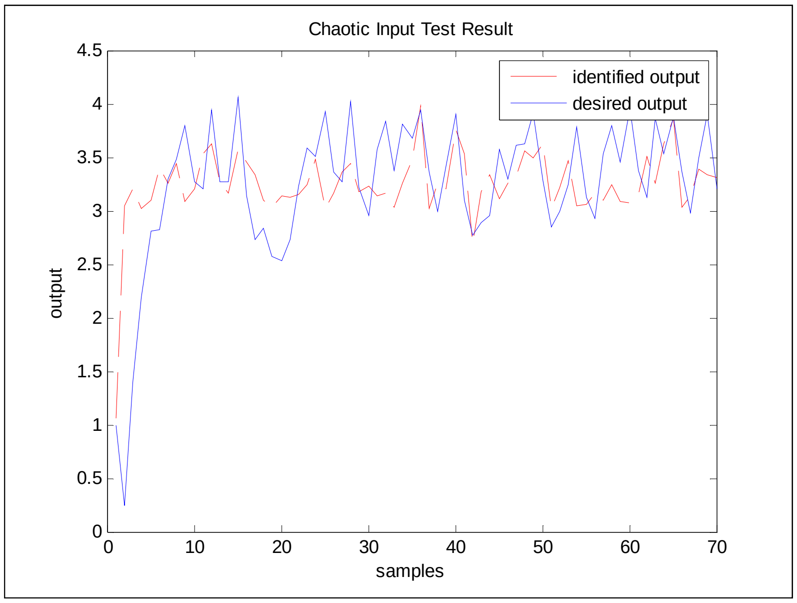

Figure 3.12 LSS-CSR method output versus desired output for chaotic input with plant I

Table 3.3 Performance of the proposed method LSS-CSR with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
LSS-CSR	Step	0.0193	14.876	17.071	18.113	0.0223	14.451	16.510	17.212
	Sin(k)	0.0340	13.842	14.721	15.002	0.0379	13.902	14.203	14.573
	Sin(20k)	0.0209	16.163	16.701	17.104	0.026	15.422	15.702	16.009
	Chaotic Input	0.0180	16.313	17.503	18.675	0.0420	12.910	13.604	15.551

Table 3.4 Performance of Hammerstein model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	Step	0.0011	27.293	29.7	31.245	0.0022	25.0021	26.64	28.109
	Sin(k)	0.015	16.934	18.3	19.993	0.033	13.992	14.8	16.043
	Sin(20k)	0.1	9.345	9.85	10.450	2.78	-5.483	-4.43	-3.453
	Chaotic Input	0.003	22.988	24.8	26.030	0.016	16.833	17.8	18.2042

Table 3.5 Performance of Wiener model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	Step	$5*10^{-9}$	79.893	81.3	83.3329	0.0015	27.118	28.3	29.004
	Sin(k)	0.023	15.893	16.2	16.554	0.033	14.493	14.7	14.995
	Sin(20k)	0.07	10.942	11.3	11.539	0.13	8.2839	8.6	8.9983
	Chaotic Input	0.0046	22.0453	23.35	24.593	3.66	-7.773	-5.64	-3.2958

Table 3.6 Performance of Hammerstein-Wiener model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	**Step**	$8*10^{-9}$	78.984	80.8	82.298	0.0014	27.048	28.4	29.857
	Sin(k)	0.011	18.4893	19.44	20.002	0.035	13.984	14.58	14.779
	Sin(20k)	0.077	10.748	11.11	11.586	0.34	3.860	4.58	4.447
	Chaotic Input	0.008	20.116	20.9	21.796	0.014	17.7745	18.54	19.0238

Table 3.7 Performance of Elman network with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
	Step	0.02	15.6013	16.92	22.85	0.024	14.640	16.21	17.1134
ELMAN Network	**Sin(k)**	0.01	16.284	17.7	19.65	0.02	15.736	16.8	18.27
	Sin(20k)	0.019	16.458	17.01	19.004	0.023	14.985	16.3	17.697
	Chaotic Input	0.0418	10.151	13.79	15.8429	0.044	4.8977	13.58	15.185

Table 3.8 Performance of nonlinear ARX model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
	Step	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent
Nonlinear ARX	**Sin(k)**	0.046	12.744	13.31	13.698	Non convergent	Non convergent	Non convergent	Non convergent
	Sin(20k)	0.09	9.904	10.4	10.842	0.177	7.220	7.5	7.702
	Chaotic Input	0.0234	15.839	16.31	16.696	0.057	11.995	12.39	12.893

Table 3.9 Performance of nonlinear ARMAX model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	Step	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent
	Sin(k)	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent	Non convergent
	Sin(20k)	25.6	-15.694	-14.1	-13.685	32.34	-15.74	-15.1	-14.802
	Chaotic Input	$2*10^{-7}$	62.034	66.3	68.201	Non convergent	Non convergent	Non convergent	Non convergent

Table 3.10 Performance of prediction error method with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	Step	4.10^{-5}	41.204	43.8	44.403	0.002	26.103	26.9	27.336
	Sin(k)	0.88	0.301	0.53	0.687	1.24	-1.112	-0.9	-0.791
	Sin(20k)	0.84	0.592	0.72	0.884	1.33	-1.559	-1.23	-1.172
	Chaotic Input	0.02	16.405	17	17.503	0.022	15.958	16.5	16.794

Table 3.11 Performance of subspace method with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	Step	$2.6*10^{-6}$	52.203	55.7	57.804	0.003	24.495	25.25	25.796
	Sin(k)	0.83	0.613	0.8	0.934	0.87	0.419	0.59	0.793
	Sin(20k)	0.58	1.984	2.31	2.652	1.022	-0.113	-0.095	0.091
	Chaotic Input	0.019	16.849	17.22	17.558	0.045	12.994	13.5	13.894

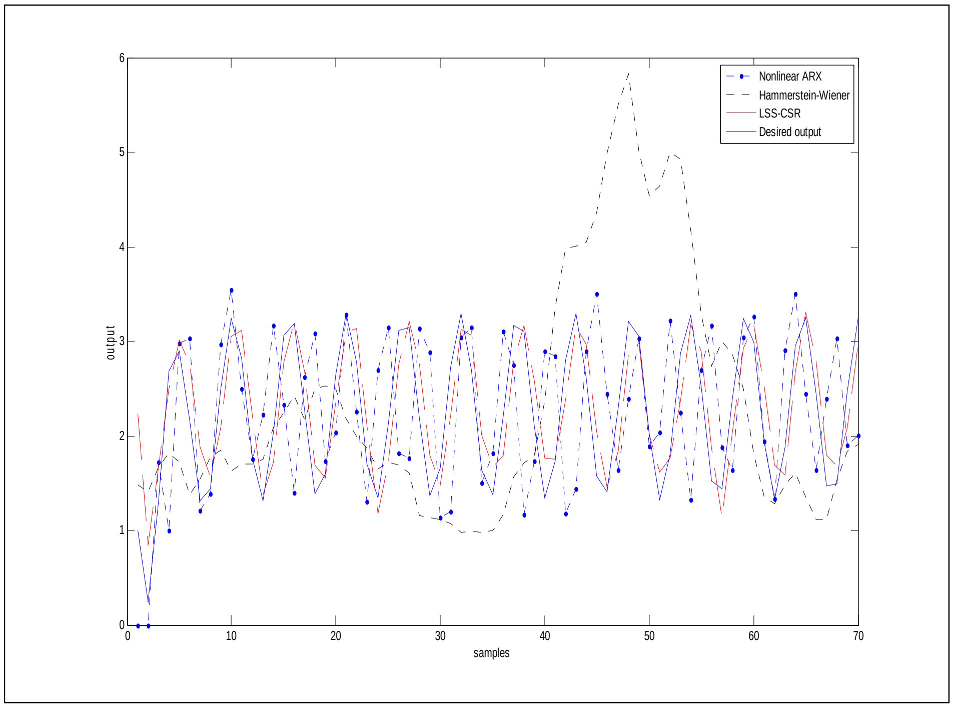

Figure 3.13 Comparison of outputs for different methods versus desired output for sin(20k) with plant I

3.3.2 Case 2: A first order nonlinear state model

In order to test the proposed method LSS-CSR with a nonlinear plant which is a nonlinear state space equation, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) which is given in Equation (3.8)

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{3.8}$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. To construct the State ANN there are three inputs (two of them are white gaussian noise with the power of 1dBW which represents the state of the plant, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which is used for learning the states. The Output ANN has 3 input neurons which of 2 are states (taken from the output of the State

ANN) and the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. The Output ANN has 1 output neuron which represents the output of the plant.

The method is tested with the artificial neural network structure above and with 4 different types of input (*u*(k)) cases listed in Table 3.12.

Table 3.12 Inputs used for testing method LSS-CSR with plant in Equation (3.8)

Input I	*u(k)*=step input
Input II	*u(k)* = sin(k)
Input III	*u(k)*= sin(20k)
Input IV	*u(k+1)*=4*u(k)*(1-u*(k)*) , *u(0)*=0.1 (Chaotic input)

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial condition is fixed to a constant value which is $x_1(0)$ is equal to 0.5. The algorithm ran 100 times to calculate SER values to find out the best, worst and the average performances of the method to the plant II which is given with the Equation (3.8). Tables from 3.13 to 3.21 show the performance of the benchmark methods and proposed method LSS-CSR to the plant II. Figures from 3.14 to 3.17 also show the identified output results versus desired outputs according to the different applied inputs. Comparison of outputs for different methods versus desired output for chaotic input with plant II is illustrated in Figure 3.18.

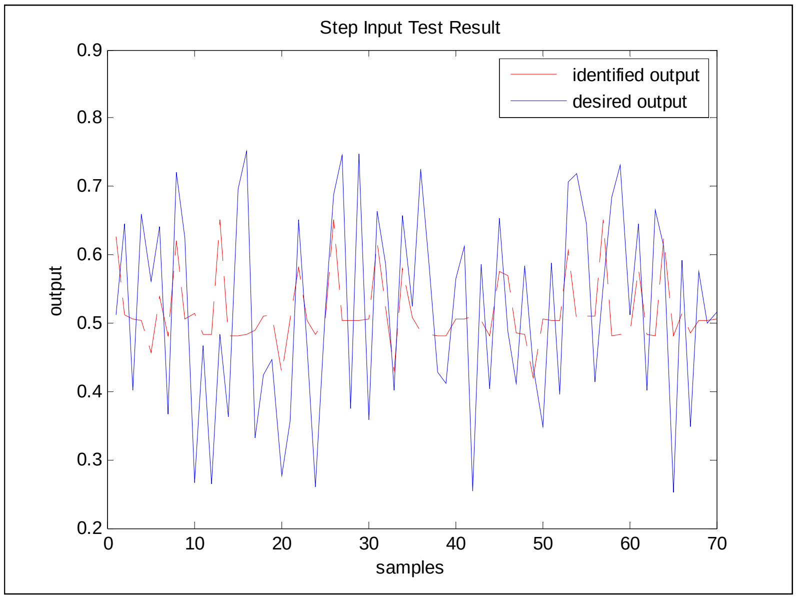

Figure 3.14 LSS-CSR method output versus desired output for step input with plant II

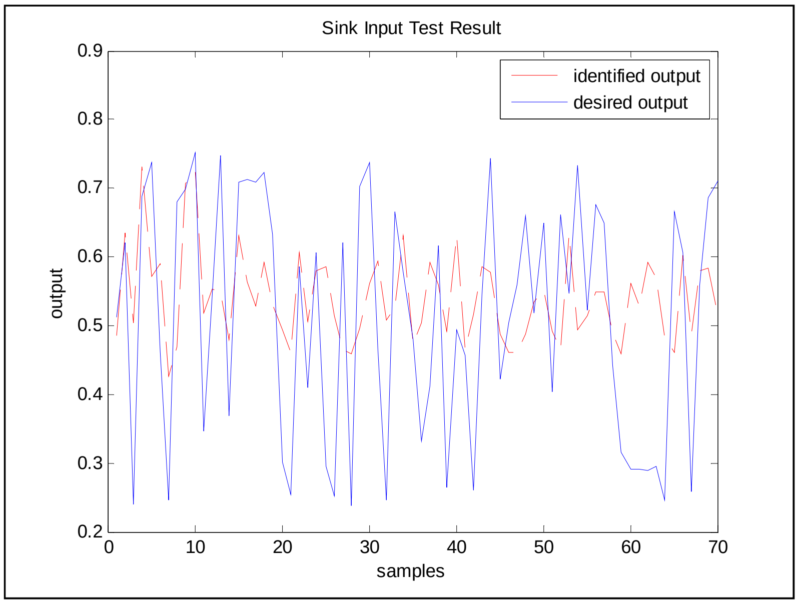

Figure 3.15 LSS-CSR method output versus desired output for sin(k) input with plant II

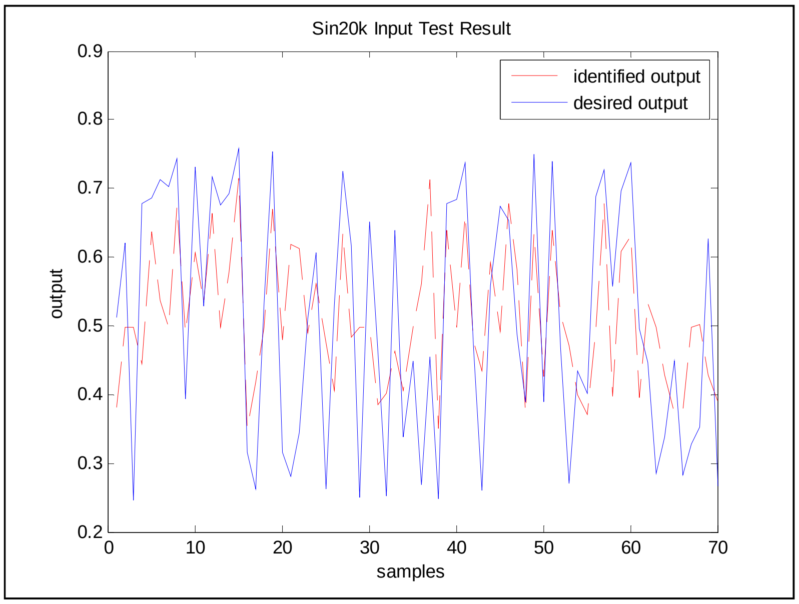

Figure 3.16 LSS-CSR method output versus desired output for sin(20k) input with plant II

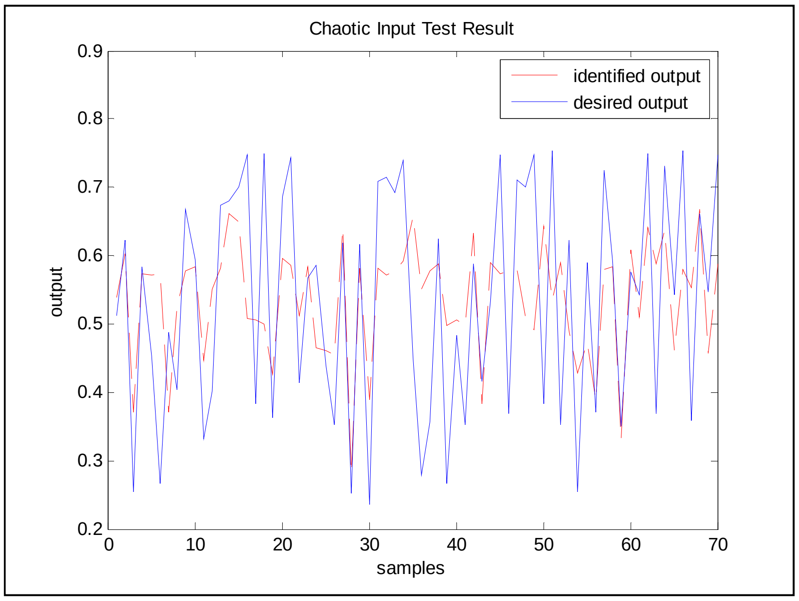

Figure 3.17 LSS-CSR method output versus desired output for chaotic input with plant II

Table 3.13 Performance of the proposed method LSS-CSR with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
LSS-CSR	Step	0.067	11.002	11.670	12.122	0.078	10.590	11.030	11.442
	Sin(k)	0.077	10.831	11.100	11.603	0.084	10.338	10.751	11.210
	Sin(20k)	0.066	11.260	11.702	12.104	0.072	11.010	11.487	11.966
	Chaotic Input	0.076	10.905	11.242	11.685	0.077	10.672	11.122	11.438

Table 3.14 Performance of Hammerstein model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	Step	0.088	10.0221	10.54	10.774	0.105	9.0903	9.78	10.001
	Sin(k)	0.52	2.519	2.82	2.9957	1.24	-2.73	-0.94	-0.052
	Sin(20k)	0.06	11.873	12.14	12.583	0.15	7.757	8.26	8.403
	Chaotic Input	0.08	10.589	10.84	10.9086	0.69	1.322	1.58	1.7128

Table 3.15 Performance of Wiener model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	Step	0.063	11.790	12.00	12.205	0.084	10.396	10.75	10.884
	Sin(k)	0.038	13.956	14.18	14.446	0.208	6.665	6.81	6.942
	Sin(20k)	0.064	11.674	11.9	12.034	0.122	8.854	9.13	9.438
	Chaotic Input	0.054	12.283	12.64	12.805	0.097	9.850	10.11	10.426

Table 3.16 Performance of Hammerstein-Wiener model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	**Step**	0.073	10.994	11.33	11.559	0.079	10.744	11.02	11.259
	Sin(k)	0.057	11.773	12.4	12.694	0.17	7.573	7.70	7.902
	Sin(20k)	0.072	10.954	11.4	11.778	0.115	8.785	9.39	9.505
	Chaotic Input	0.055	11.998	12.59	12.874	0.53	1.992	2.75	2.995

Table 3.17 Performance of Elman network with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
ELMAN Network	**Step**	0.088	10.2265	10.55	10.971	0.091	10.002	10.42	10.769
	Sin(k)	0.1026	9.521	9.888	9.9555	0.1058	9.3862	9.6453	9.8197
	Sin(20k)	0.1030	9.623	9.86	10.1693	0.1232	8.7340	9.0928	9.4004
	Chaotic Input	0.084	10.362	10.76	11.0676	0.1004	9.752	9.98	10.3296

Table 3.18 Performance of nonlinear ARX model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	$7.7*10^3$	-42.349	-38.89	-35.340	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 3.19 Performance of nonlinear ARMAX model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	11.65	-11.129	-10.66	-9.902	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 3.20 Performance of prediction error method with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	**Step**	0.082	9.964	10.84	10.993	0.101	9.504	9.94	10.023
	Sin(k)	0.74	1.112	1.32	1.432	2.09	-4.032	-3.2	-2.893
	Sin(20k)	1.028	-0.304	-0.12	0.003	1.18	-1.114	-0.74	-0.504
	Chaotic Input	0.102	9.603	9.90	10.005	0.141	7.920	8.48	8.7732

Table 3.21 Performance of subspace method with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	**Step**	0.081	10.703	10.9	11.003	0.113	8.945	9.46	9.603
	Sin(k)	0.77	0.794	1.115	1.203	1.118	-0.505	-0.48	-0.385
	Sin(20k)	0.95	0.159	0.203	0.293	1.004	-0.100	-0.018	0.030
	Chaotic Input	0.075	10.403	11.23	11.584	0.18	6.892	7.44	7.771

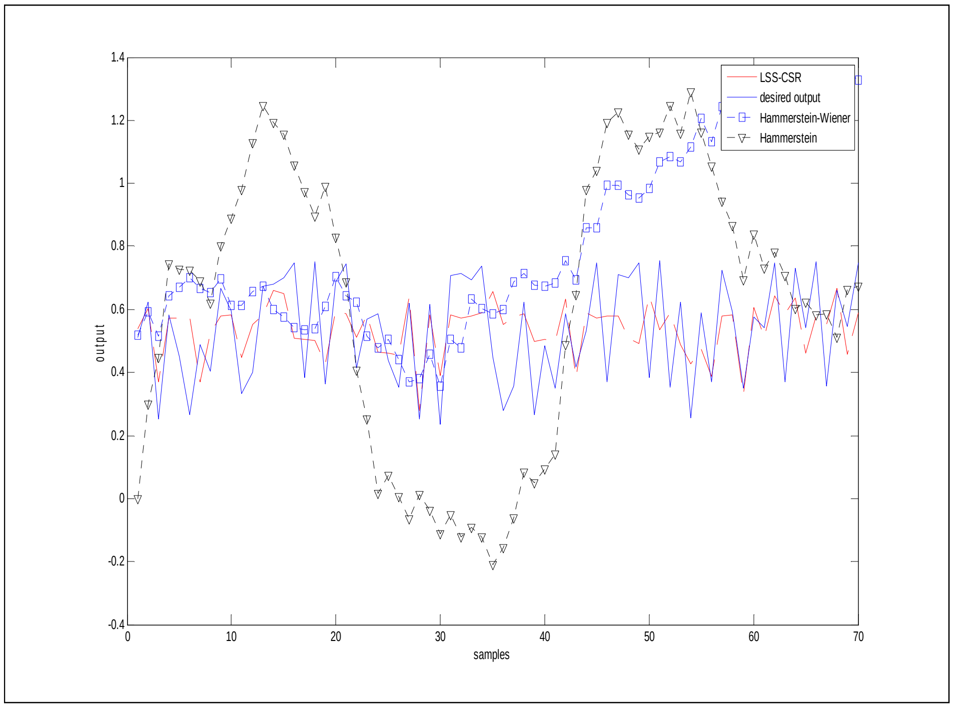

Figure 3.18 Comparison of outputs for different methods versus desired output for chaotic input with plant II

3.3.3 Case 3: A third order nonlinear difference equation model

In order to test the proposed method LSS-CSR, the modified dynamics of the nonlinear benchmark plant in literature is borrowed (Uykan & others, 2000) which is shown in Equation 3.9. This plant called plant III is more complicated than the plant given with the Equation 3.8 since this model has more dynamics in the state equation. The plant III also tested with different inputs like plant I and plant II. For the step and chaotic inputs Equation (3.9) is used, for sinusoidal inputs sin(k) and sin(20k) the plant given in Equation (3.10) is used to make the plant stable.

$$\begin{aligned} x(k+1) &= \frac{9.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \\ y(k) &= x(k)+1 \end{aligned} \tag{3.9}$$

$$x(k+1) = \frac{2.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \tag{3.10}$$
$$y(k) = x(k)+1$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For constructing the State ANN there are three inputs (two of them are white gaussian noise with the power of 1dBW representing the state of the plant, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which is used for learning the states. The Output ANN has 3 input neurons, two are coming from the State ANN output and the other is the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. The Output ANN has one output neuron representing the output.

The method is tested with the structure above and with 4 different types of input (*u*(k)) cases listed in Table 3.22.

.

Table 3.22 Inputs used for testing method LSS-CSR with plant III

Input I	*u*(k)=step input
Input II	*u*(k) = sin(k)
Input III	*u*(k)= sin(20k)
Input IV	*u*(k+1)=4u(k)(1-u(k)) , *u*(0)=0.1 (Chaotic input)

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial conditions *x*(-2), *x*(-1), *x*(0), *y*(0), *y*(-1) are chosen as 1,1.5, 2, 0.5, and 0.5 respectively.

The algorithm ran 100 times to calculate SER values to find out the best, worst and the average performances of the method to the plant III. Tables from 3.23 to 3.31 show the performance of the benchmark methods and proposed method LSS-CSR to the plant II. Figures from 3.19 to 3.22 also show the identified output results versus desired outputs according to the different applied inputs. Comparison of outputs for different methods versus desired output for chaotic input with plant II is illustrated in Figure 3.23.

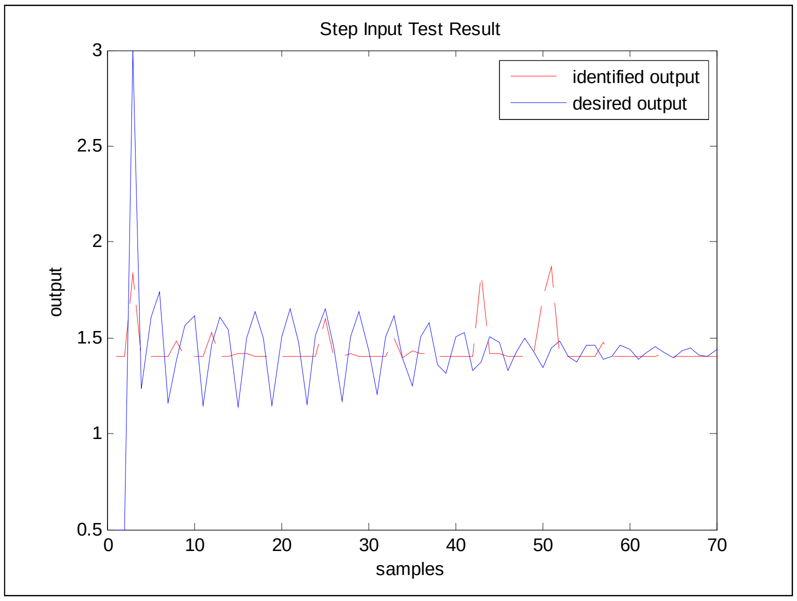

Figure 3.19 LSS-CSR method output versus desired output for step input with plant III

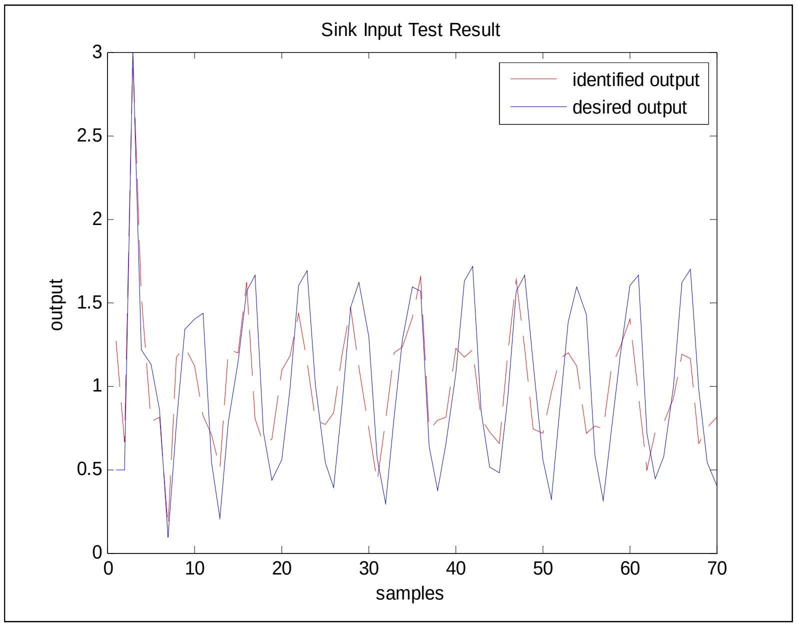

Figure 3.20 LSS-CSR method output versus desired output for sin(k) input with plant III

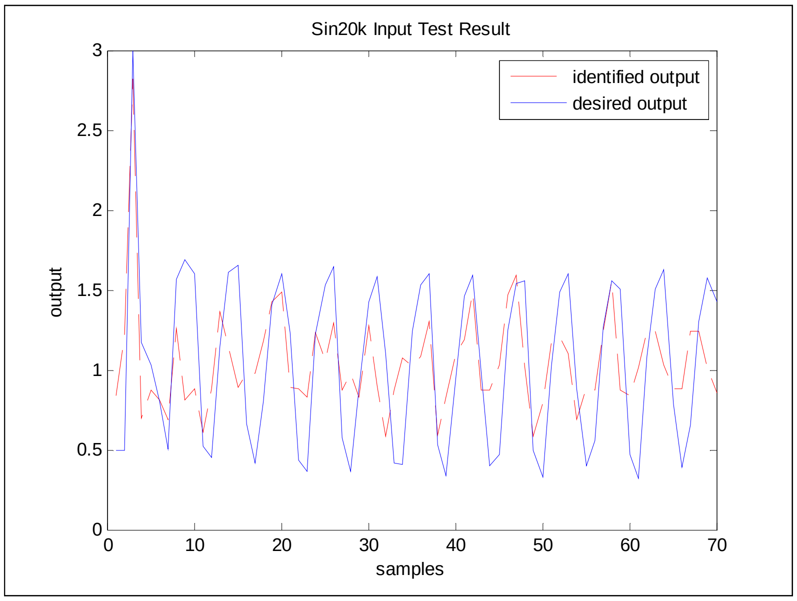

Figure 3.21 LSS-CSR method output versus desired output for sin(20k)input with plant III

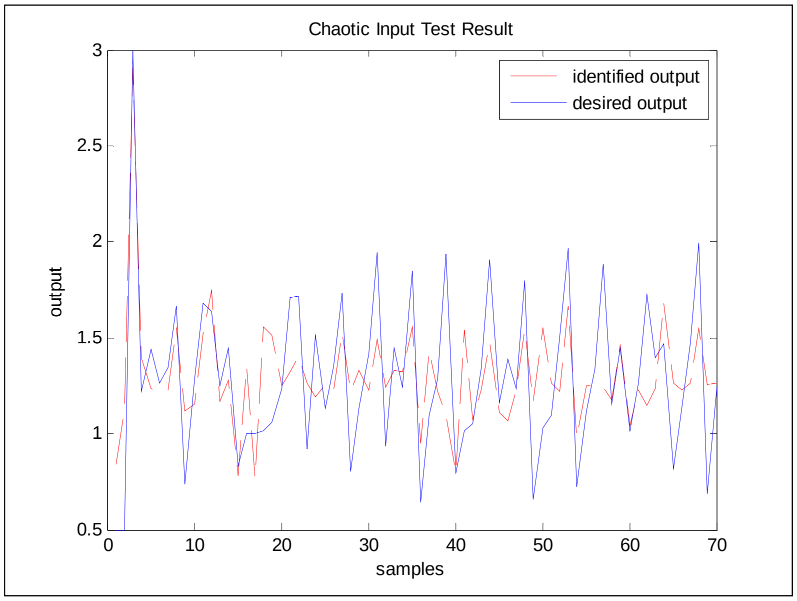

Figure 3.22 LSS-CSR method output versus desired output for chaotic input with plant III

Table 3.23 Performance of the proposed method LSS-CSR with Plant III

	INPUT	**nMSE (training)**	**SER (dB) (training) (Min)**	**SER (dB) (training) (Mean)**	**SER (dB) (training) (Max)**	**nMSE (test)**	**SER (dB) (test) (Min)**	**SER (dB) (test) (Mean)**	**SER (dB) (test) (Max)**
LSS-CSR	**Step**	0.0270	14.703	15.700	16.103	0.028	14.348	15.293	15.641
	Sin(k)	0.136	8.0300	8.572	9.002	0.156	7.673	8.110	8.247
	Sin(20k)	0.144	8.040	8.388	8.709	0.149	7.672	8.205	8.483
	Chaotic input	0.078	10.466	11.002	11.163	0.081	10.198	10.888	11.157

Table 3.24 Performance of Hammerstein model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	**Step**	0.0097	19.493	20.16	20.592	0.029	14.993	15.36	15.704
	Sin(k)	0.112	8.994	9.49	9.743	0.951	0.003	0.217	0.702
	Sin(20k)	0.038	13.895	14.11	14.443	0.853	-0.034	0.69	1.008
	Chaotic Input	0.047	12.904	13.28	13.603	0.098	9.856	10.08	10.395

Table 3.25 Performance of Wiener model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	**Step**	0.004	22.9403	24.35	25.503	0.032	14.553	14.92	15.254
	Sin(k)	0.078	10.873	11.03	11.110	0.204	6.201	6.89	6.970
	Sin(20k)	0.066	11.593	11.800	11.990	0.659	1.504	1.809	2.002
	Chaotic Input	0.0351	13.984	14.54	14.785	0.097	9.795	10.13	10.376

Table 3.26 Performance of Hammerstein-Wiener model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	**Step**	0.007	20.693	21.52	22.012	0.026	15.405	15.85	15.996
	Sin(k)	0.016	17.302	17.89	18.003	0.286	5.274	5.43	5.596
	Sin(20k)	0.024	15.785	16.19	16.394	0.62	1.806	2.05	2.226
	Chaotic Input	0.032	14.468	14.88	15.067	0.103	9.572	9.83	10.043

Table 3.27 Performance of Elman network with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
ELMAN Network	Step	0.025	15.643	16.05	17.0735	0.042	13.483	13.83	14.5639
	Sin(k)	0.105	7.3596	9.79	12.385	0.152	5.9402	8.18	10.362
	Sin(20k)	0.224	6.0054	6.501	7.0129	0.272	5.236	5.64	5.952
	Chaotic Input	0.08	10.102	10.99	11.8677	0.1028	9.473	9.8791	10.6933

Table 3.28 Performance of nonlinear ARX model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARX	Step	0.01	19.784	19.95	20.506	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	0.22	6.382	6.55	6.770	0.37	3.336	4.30	4.508
	Sin(20k)	0.096	9.784	10.15	10.384	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	0.137	8.404	8.63	8.732	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 3.29 Performance of nonlinear ARMAX model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	Step	0.01	19.094	19.95	20.498	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	0.071	10.890	11.44	11.774	0.79	0.403	1.012	1.205
	Sin(20k)	2.032	-3.645	-3.08	-2.403	2.083	-3.676	-3.18	-2.806
	Chaotic Input	8.76	-10.003	-9.43	-8.884	10.36	-10.54	-10.15	-9.709

Table 3.30 Performance of prediction error method with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	**Step**	0.01	-19.504	19.98	20.302	0.018	16.985	17.23	17.406
	Sin(k)	0.49	2.589	3.07	3.453	1.51	-1.903	-1.81	-0.040
	Sin(20k)	0.949	0.002	0.22	0.303	2.26	-3.903	-3.54	-2.705
	Chaotic Input	0.088	9.905	10.52	10.813	0.105	9.604	9.76	9.895

Table 3.31 Performance of subspace method with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	**Step**	0.017	17.306	17.69	18.003	0.022	15.903	16.5	16.781
	Sin(k)	0.63	1.403	1.94	2.004	1.181	-1.023	-0.70	-0.068
	Sin(20k)	0.91	0.302	0.40	0.443	1.18	-1.059	-0.72	-0.174
	Chaotic Input	0.07	10.903	11.51	11.604	0.365	3.958	4.36	4.980

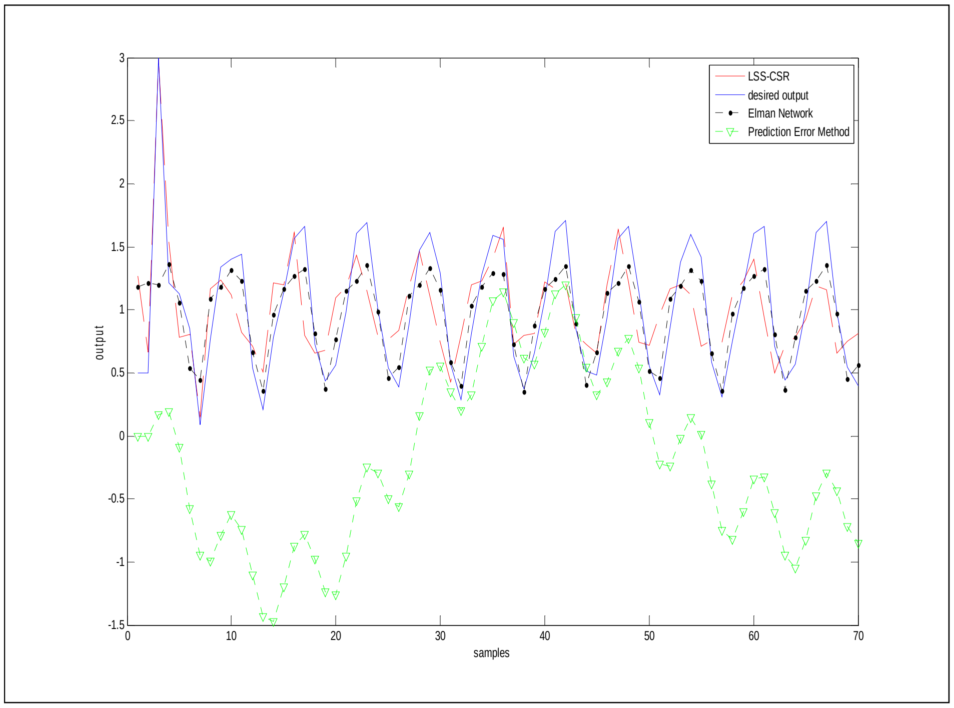

Figure 3.23 Comparison of outputs for different methods versus desired output for sin(k) with plant III

3.3.4 Case 4: A chaotic system (forced duffing oscillator)

In order to test the proposed method LSS-CSR, the modified dynamics of the chaotic benchmark plant in the literature is borrowed which is shown in Equation (3.12). This equation can also be written with two state equations in Equation (3.13). This is chosen in order to test the performance of the proposed method LSS-CSR with a chaotic plant. In cases I, II, and III nonlinear and linear plants were tested with chaotic inputs. Forced Duffing oscillator were chosen as a chaotic plant because of its forcing input, since the proposed method needs input-output data of the plant which is to be identified. Forced duffing oscillator states and the phase portrait is shown in Figures 3.24, Figure 3.25, and Figure 3.26 under the initial conditions in Equation (3.11).

$$x_1(0)=3 \text{ and } x_2(0)=4 \tag{3.11}$$

$$\ddot{x} = 7.6\cos(t) - 0.04\dot{x} - x^3 \tag{3.12}$$

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= 7.6\cos(t) - 0.04x_2 - x_1^3 \\ y &= 0.05x_1 + 10\end{aligned} \tag{3.13}$$

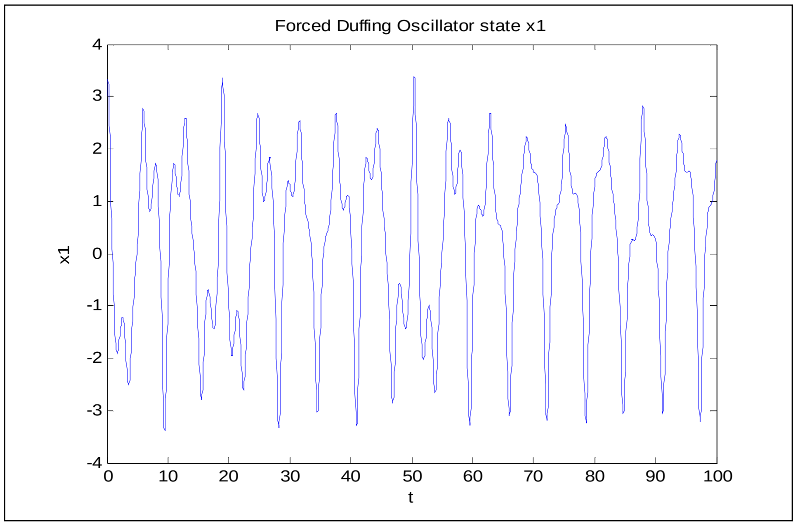

Figure 3.24 Forced Duffing Oscillator state x1

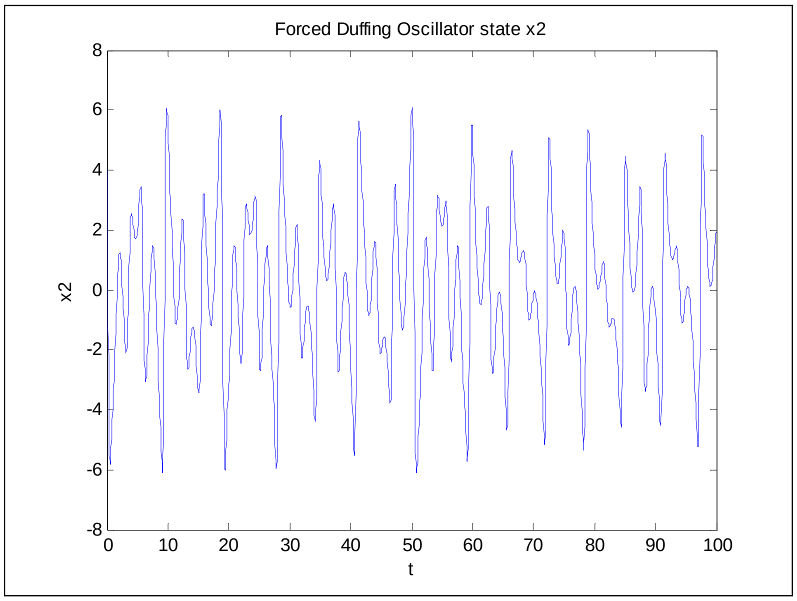

Figure 3.25 Forced Duffing Oscillator state x2

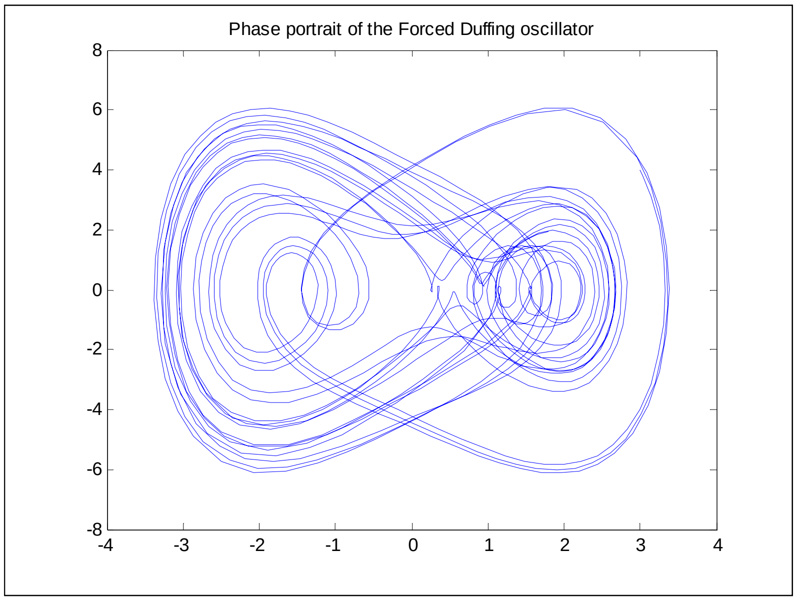

Figure 3.26 Forced Duffing Oscillator phase portrait

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For constructing the State ANN there are three inputs (two of them are the states of a henon system and a lorenz system, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Output ANN has also 3 input neurons, two are coming from the State ANN output and the other is the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. Also the Output ANN has 1 output neuron for learning the output. In order to train the State ANN used modified lorenz system is given in Equation (3.14) (Lorenz, 1963; Cuomo & Oppenheim, 1993).

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1) \\ \dot{x}_2 &= rx_1 - x_2 - 20x_1x_3 \\ \dot{x}_3 &= 5x_1x_2 - bx_3 \end{aligned} \tag{3.14}$$

with the parameters σ, *r*, and *b* are 10, 56.6, 5.02, respectively.

The states and the phase portrait of the used Lorenz system is given in Figure 3.27, Figure 3.28, Figure 3.29, and Figure 3.30. under the initial conditions in Equation (3.15).

$$x_1(0)=1,\ x_2(0)=0, \text{ and } x_3(0)=1 \tag{3.15}$$

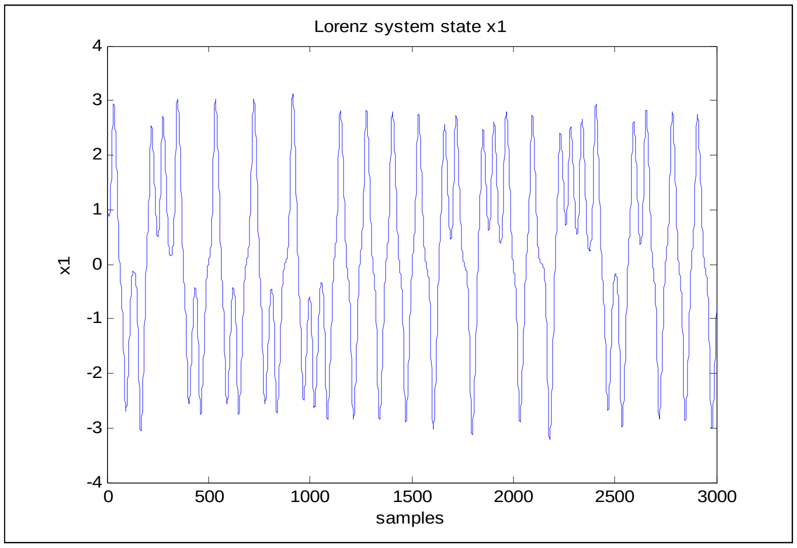

Figure 3.27 Lorenz system state x_1

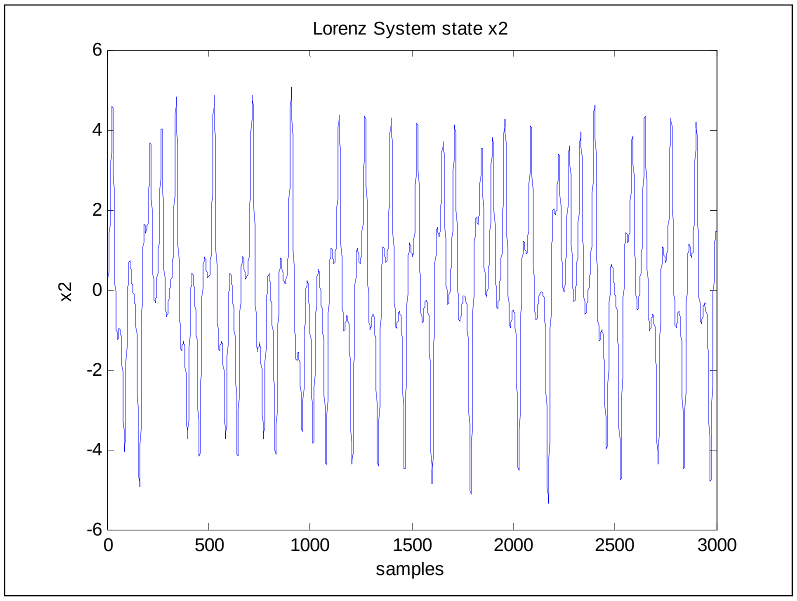

Figure 3.28 Lorenz system state x_2

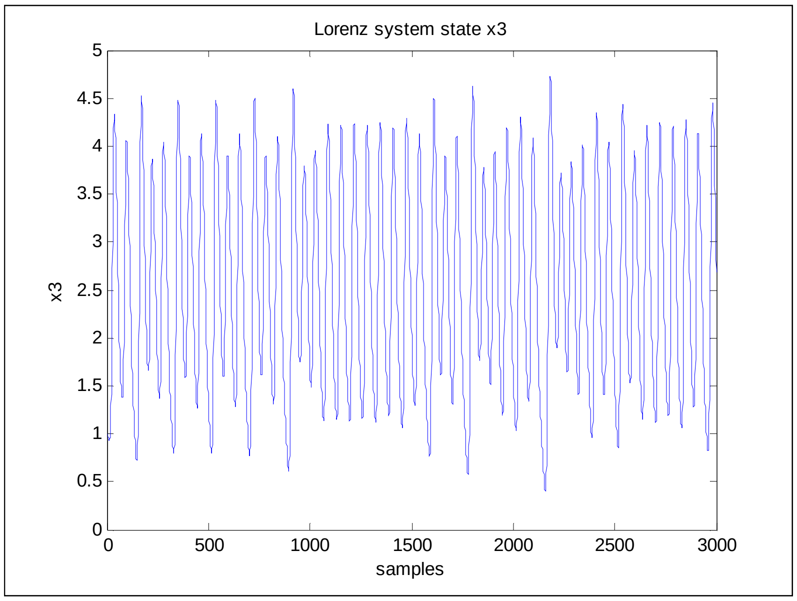

Figure 3.29 Lorenz system state x_3

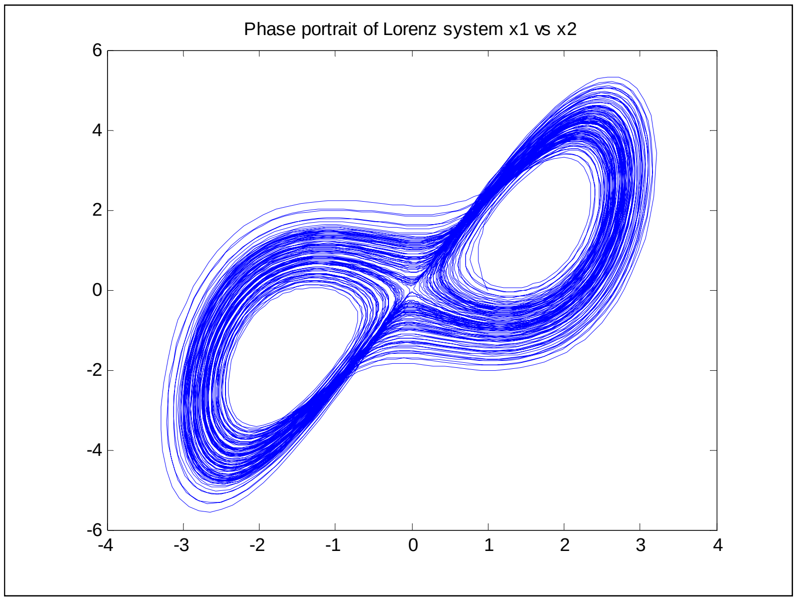

Figure 3.30 Lorenz system phase portrait x_1 versus x_2

In order to train the State ANN used modified henon system is given in Equation (3.16) (Dick & Kandel; 2005).

$$y(k+1) = 1.4y(k)^2 + 0.3y(k-1) + 1 \qquad (3.16)$$

The phase portrait of the henon system in Equation 3.16 is given in Figure 3.31 under the initial conditions $y(-1)=0.5$ and $y(0)=0.5$.

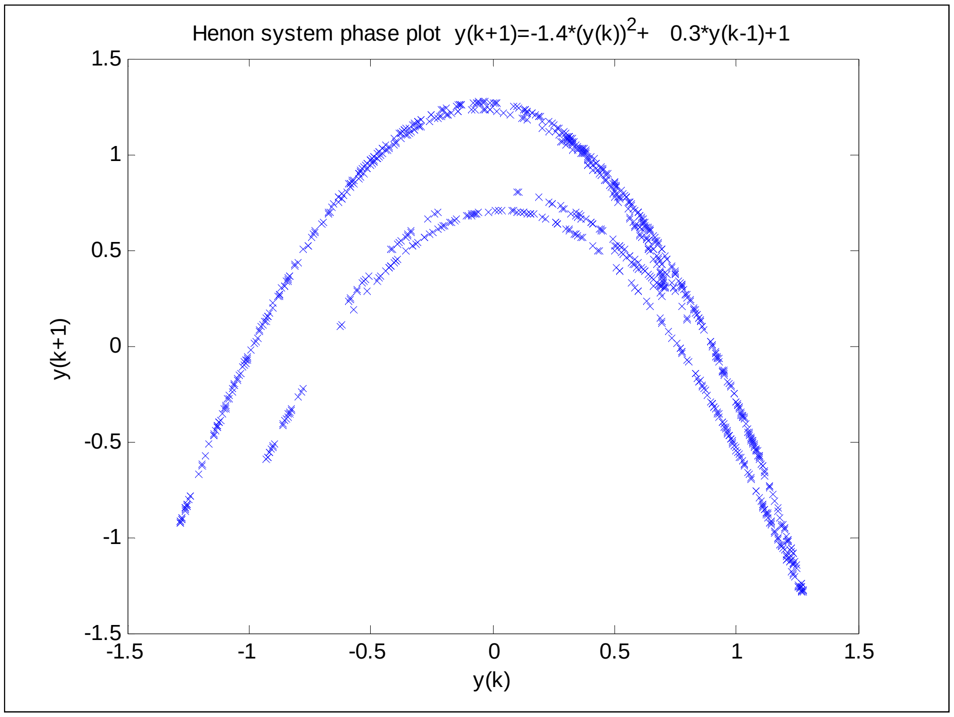

Figure 3.31 Henon System phase plot

Train and test performances of the proposed method and the benchmark methods taken from the literature are given in Table 3.32.

Table 3.32 Comparison of train and test performances of the proposed method and the benchmark methods

Model	**nMSE training**	**SER (dB) (training) (Min)**	**SER (dB) training(Mean)**	**SER (dB) training (Max)**	**nMSE (test)**	**SER (dB) (test) (Min)**	**SER (dB) (test) (Mean)**	**SER (dB) (test) (Max)**
LSS-CSR	0.0073	20.912	21.330	22.603	0.0118	17.996	19.288	21.801
Nonlinear ARX	0.028	15.102	15.53	15.721	0.0688	11.403	11.62	11.832
Prediction Error Method	0.197	6.663	7.03	7.904	2.541	-4.589	-4.05	-3.453
Subspace Method	0.749	0.432	1.25	1.698	1.013	-0.332	-0.056	0.102
Wiener Model	0.029	14.895	15.40	15.793	0.151	7.773	8.204	8.502
Hammerstein Model	0.0548	12.093	12.61	12.856	0.167	7.551	7.75	7.896
Hammerstein-Wiener Model	0.0275	15.398	15.604	15.873	0.295	4.706	5.293	5.554
Nonlinear ARMAX	0.039	13.794	14.092	14.443	0.069	11.394	11.56	11.707
Elman Network	0.0320	14.607	14.9446	15.254	0.0335	14.392	14.750	14.967

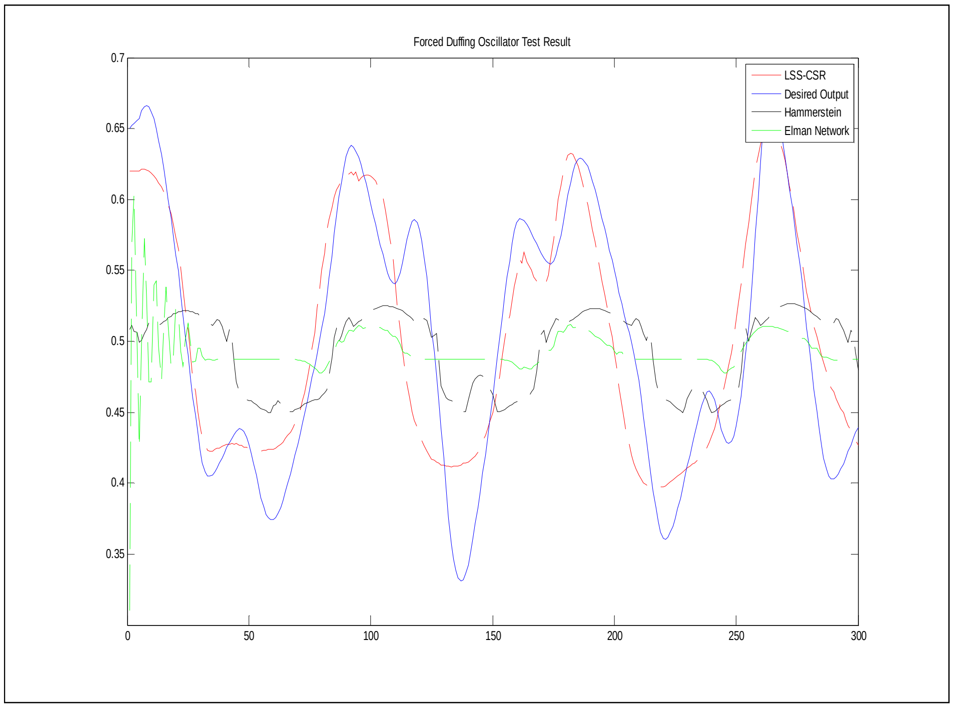

Figure 3.32 Forced Duffing Oscillator Test Result

3.3.5 Performance of the proposed method LSS-CSR with different initial conditions

3.3.5.1 Test a

In order to test the proposed method LSS-CSR to different initial conditions; the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) given in Equation (3.17).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (1+v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{3.17}$$

The benchmark plant is executed with different initial conditions. State ANN and Output ANN are trained with some possible values of initial conditions. For the plant above $x(1)$ is varied between 0 and 19.99 and the State ANN and Output ANN are trained with 70 different initial conditions. Then the method is tested with different initial conditions that are not placed in the training set.

The Figure 3.33 shows the identified and desired outputs which are the responses of x_1(k)=5 and x_2(k)=0.5. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets.

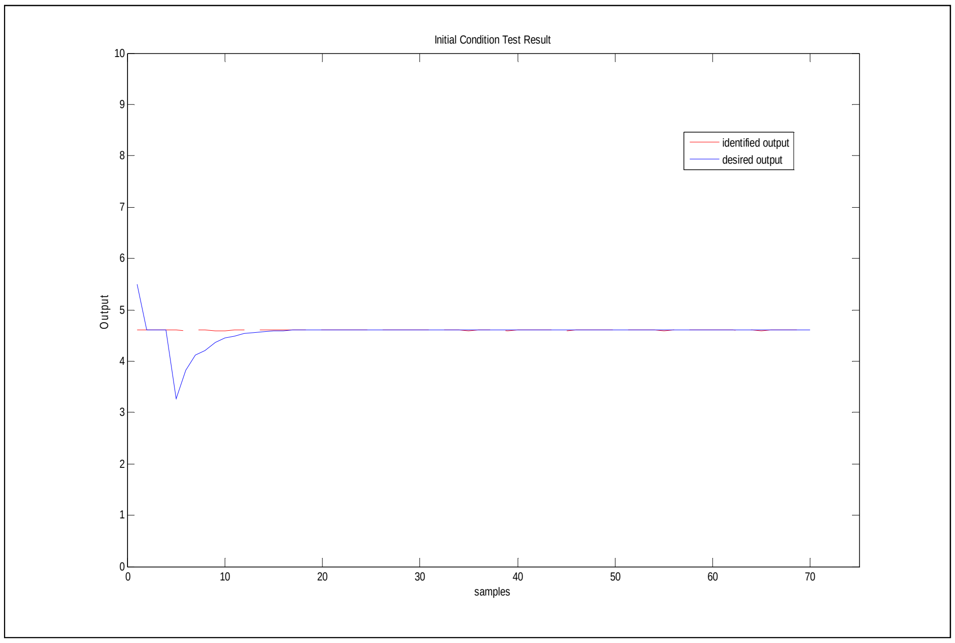

Figure 3.33 Initial condition Test Result of Proposed Method LSS-CSR with Plant I

The performance parameters of the test point x_1(k)=5 and x_2(k)=0.5 which is shown in the Figure 3.33 are calculated and they are shown in Table 3.33 as follows;

Table 3.33 Initial condition Performance parameters of method LSS-CSR with Plant I

SER (dB)	25.998
MSE	0.0530
nMSE	0.0026

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 3.34.

Table 3.34 Initial condition SER results of method LSS-CSR with Plant I

SER (worst) (dB)	24.103
SER (mean) (dB)	26.692
SER (best) (dB)	28.872

3.3.5.2 Test b

The proposed method LSS-CSR performance is tested with different initial conditions for a nonlinear plant which the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) and shown in Equation (3.18).

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{3.18}$$

The plant input is a unit step input. The benchmark plant is executed with different initial conditions. State ANN and Output ANN are trained with some possible values of initial conditions. For the plant above x is varied between 0 and 89.99 and the State ANN and Output ANN are trained with 70 different initial conditions. Then the method is tested with different initial conditions that are not placed in the training set.

The Figure 3.34 shows the identified and desired outputs which are the responses of x(k)=5. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets.

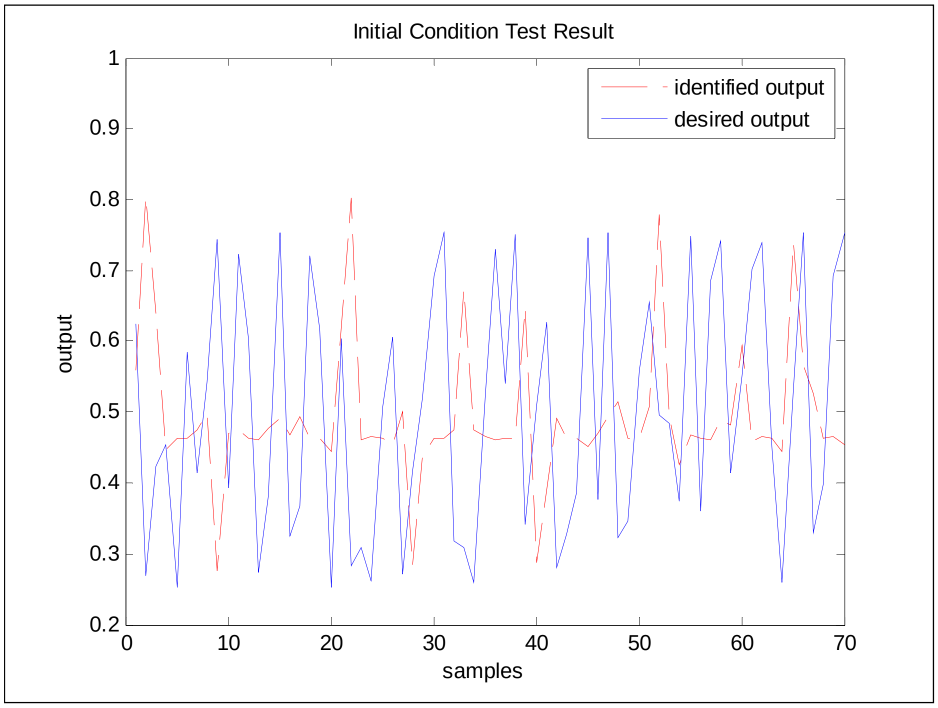

Figure 3.34 Initial condition test result of the proposed method LSS-CSR with Plant II

The performance parameters of the test point *x*(k)=5 which is shown in the Figure 3.34 are calculated and they are shown in Table 3.35 as follows;

Table 3.35 Initial condition Performance parameters of Method LSS-CSR with Plant II

SER (dB)	7.98
MSE	0.044
nMSE	0.159

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 3.36.

Table 3.36 Initial condition SER results of Method LSS-CSR with Plant II

SER (worst) (dB)	7.104
SER (mean) (dB)	8.066
SER (best) (dB)	10.317

3.3.5.3 Test c

In order to test the proposed model to different initial conditions; the dynamics of the benchmark plant in literature is borrowed (Uykan & others, 2000)

$$x(k+1) = \frac{9.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \qquad (3.19)$$
$$y(k) = x(k)+1$$

To test the proposed model, the benchmark plant is executed with different initial conditions and with the step input u. State ANN and output ANN are trained with some possible values of initial conditions. The benchmark plant above $x(0)$ is varied from 0 to 2.99.

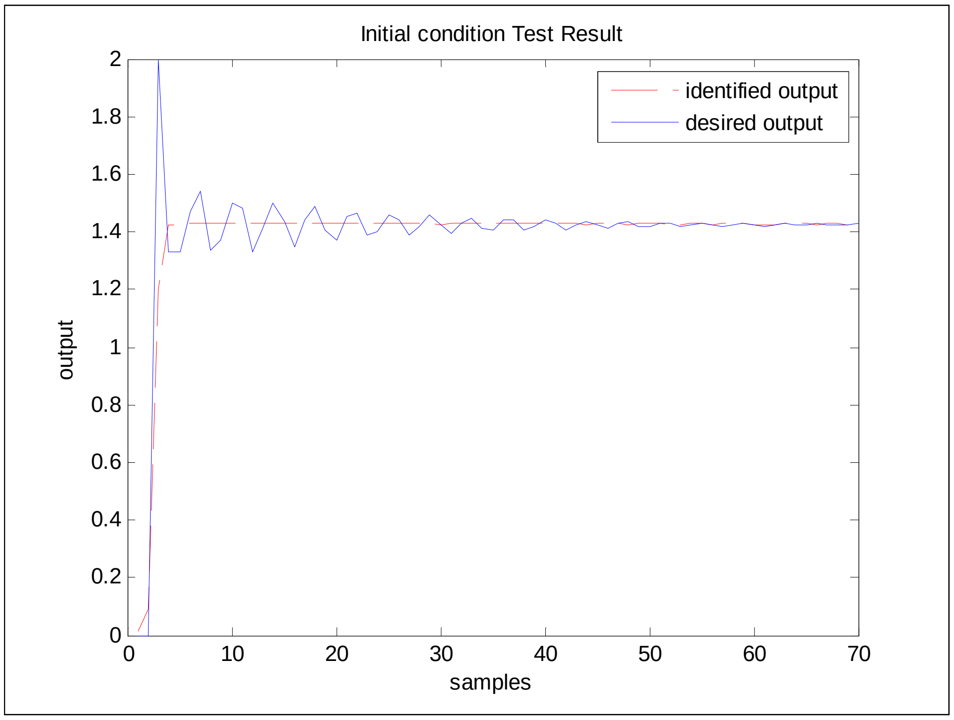

Figure 3.35 The identified and desired outputs at $x(0)$=1

The above figure shows the identified and desired outputs which are the responses of $x(0)$=1. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets. Calculated parameters are listed in Table 3.37.

Table 3.37 Calculated parameters of Method LSS-CSR when $x(0)=1$

SER	22.674dB
MSE	0.0112
nMSE	0.0054

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 3.38.

Table 3.38 SER results of Method LSS-CSR when $x(0)=1$

SER(worst)	20.334dB
SER(mean)	22.651dB
SER(best)	24.495dB

3.3.6 Noise performance of the proposed method LSS-CSR

3.3.6.1 Case 1: A second order linear state model

In order to test the proposed method LSS-CSR, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) and given in Equation (3.20).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (u(k)+v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{3.20}$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. To construct the State ANN there are three inputs (two of them are white gaussian noise with the power of 1dBW which represents the state of the plant, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which is used for learning the states. The Output ANN has 3 input neurons which are states (taken from the output of the State ANN) and the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. The Output ANN has 1 output neuron which represents the output of the plant.

Noise performance is tested with a uniformly distributed noise between [-0.40 0.40]. A uniformly distributed noise is added to the plant input, and the system performance is tested. Figure (3.36) shows the plant input with additive noise and without noise. Plant input is chosen a sinusoidal input which is a sinus sin(20k).

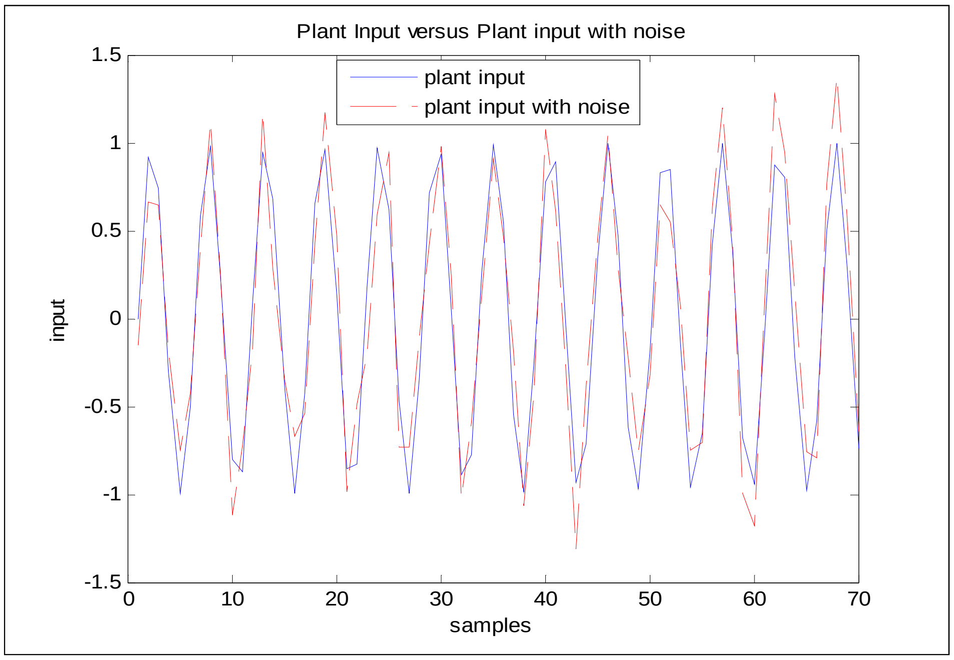

Figure 3.36 Plant input versus plant input with additive noise

Table 3.39 Noise performance results of plant I

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSR	0.026	15.702	0.0274	15.61
Nonlinear ARX	0.177	7.5	0.177	7.5
Prediction Error Method	1.33	-1.23	1.473	-1.68
Subspace Method	1.022	-0.095	1.028	-0.122
Wiener Model	0.130	8.6	0.159	7.98
Hammerstein Model	2.78	-4.430	58.45	-17.667
Hammerstein-Wiener Model	0.34	4.58	2.032	-3.08
Nonlinear ARMAX	32.34	-15.1	33.198	-15.21
Elman Network	0.023	16.3	0.0281	15.50

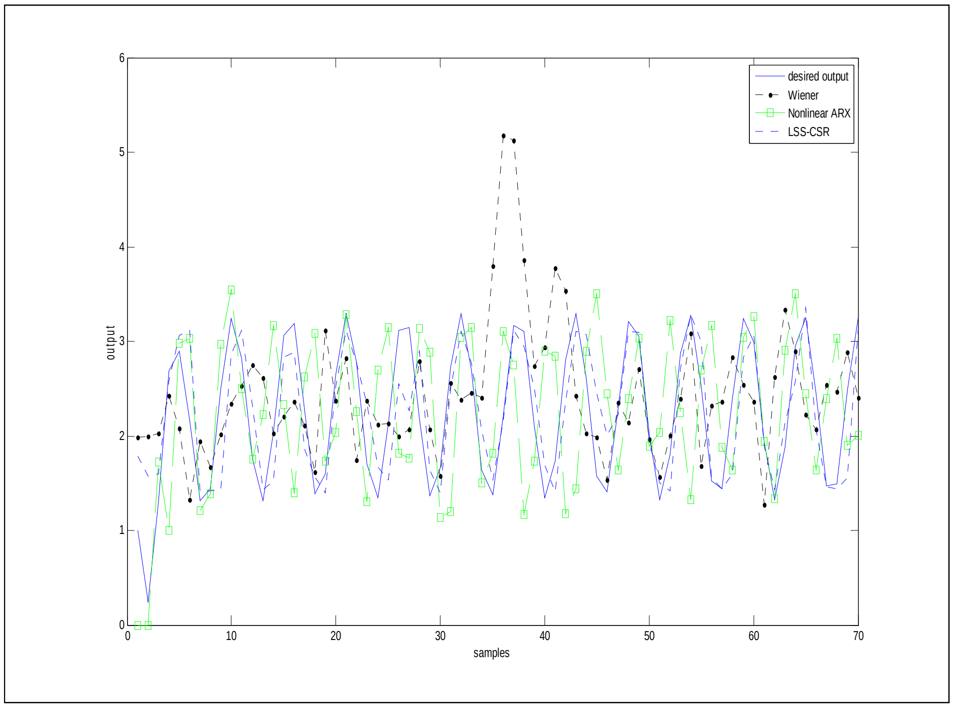

Figure 3.37 Comparison of outputs for different methods versus desired output of plant I with noise

3.3.6.2 Case 2: A first order nonlinear state model

In order to test the proposed method LSS-CSR with a nonlinear plant which is a nonlinear state space equation, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) which is given in Equation (3.21).

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{3.21}$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. To construct the State ANN there are three inputs (two of them are white gaussian noise with the power of 1dBW which represents the state of the plant, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which is used for learning the states. The Output ANN has 3 input neurons which are states (taken from the output of the State ANN) and the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. The Output ANN has 1 output neuron which represents the output of the plant.

Noise performance is tested with a uniformly distributed noise between [-0.25 0.25]. A uniformly distributed noise is added with the plant input, and the system performance is tested. Below the Figure 3.38 shows the plant input with additive noise and without noise.

Plant input is chosen a chaotic input which is given in Equation (3.22).

$$v(m+1) = 4v(m)(1-v(m)), \quad v(0) = 0.1\,(\text{chaotic}) \tag{3.22}$$

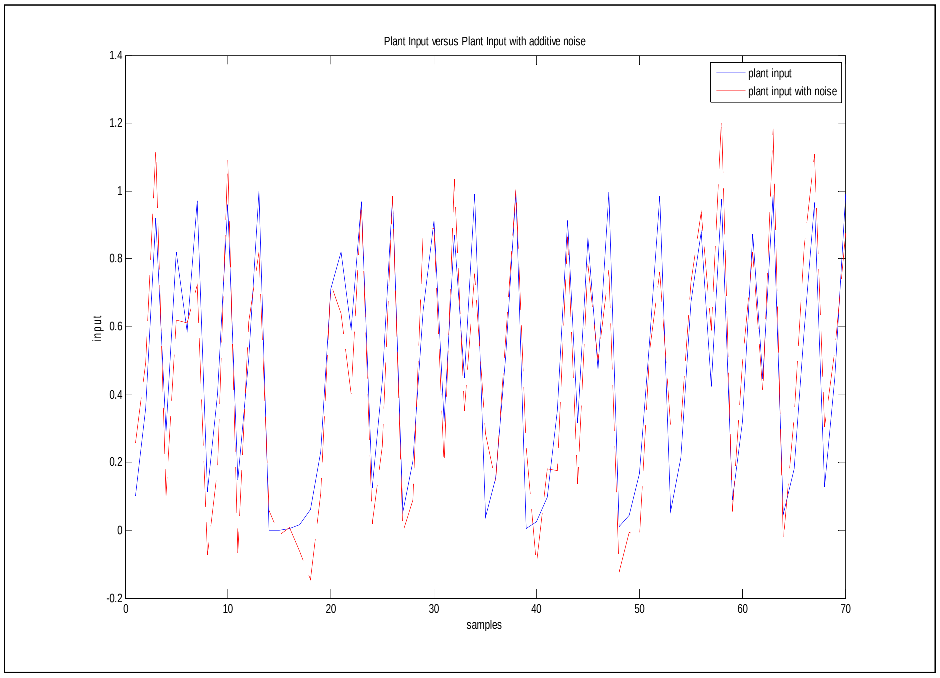

Figure 3.38 Plant input versus plant input with additive noise

Table 3.40 Noise performance results of plant II

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSR	0.077	11.122	0.084	10.704
Nonlinear ARX	Nonconvergent	Nonconvergent	Nonconvergent	Nonconvergent
Prediction Error Method	0.141	8.48	0.161	7.91
Subspace Method	0.180	7.44	0.196	7.070
Wiener Model	0.097	10.11	0.104	9.80
Hammerstein Model	0.69	1.58	33.022	-15.188
Hammerstein-Wiener Model	0.53	2.75	141.778	-21.51
Nonlinear ARMAX	Nonconvergent	Nonconvergent	Nonconvergent	Nonconvergent
Elman Network	0.1004	9.98	0.1043	9.81

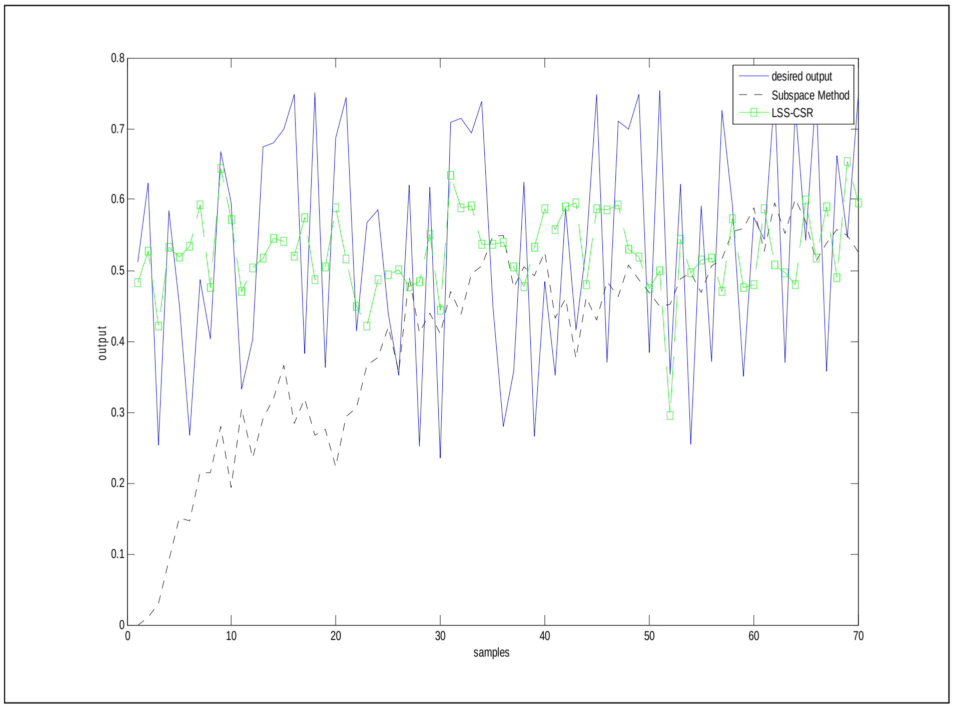

Figure 3.39 Comparison of outputs for different methods versus desired output of plant II with noise

3.3.6.3 Case 3: A third order nonlinear difference equation model

In order to test the proposed method LSS-CSR, the dynamics of the nonlinear benchmark plant in literature is borrowed (Uykan & others, 2000) which is shown in Equation (3.23). This plant is more complicated than the plant given with the Equation (3.21) since this model has more dynamics in the state equation.

$$x(k+1) = \frac{2.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \qquad (3.23)$$

$$y(k) = x(k)+1$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For constructing the State ANN there are three inputs (two of them are white gaussian noise with the power of 1dBW representing the state of the plant, the other is the input which is *u(k)*), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Output ANN has 3 input neurons, two of them are coming from the State ANN output and the other is the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons.

Noise performance is tested with a uniformly distributed noise between [-0.40 0.40]. A uniformly distributed noise is added to the plant input which is a sinusoidal input sin(k) where k is from 0 to 70 and the system performance is tested. Below the figure shows the plant input with additive noise and without noise.

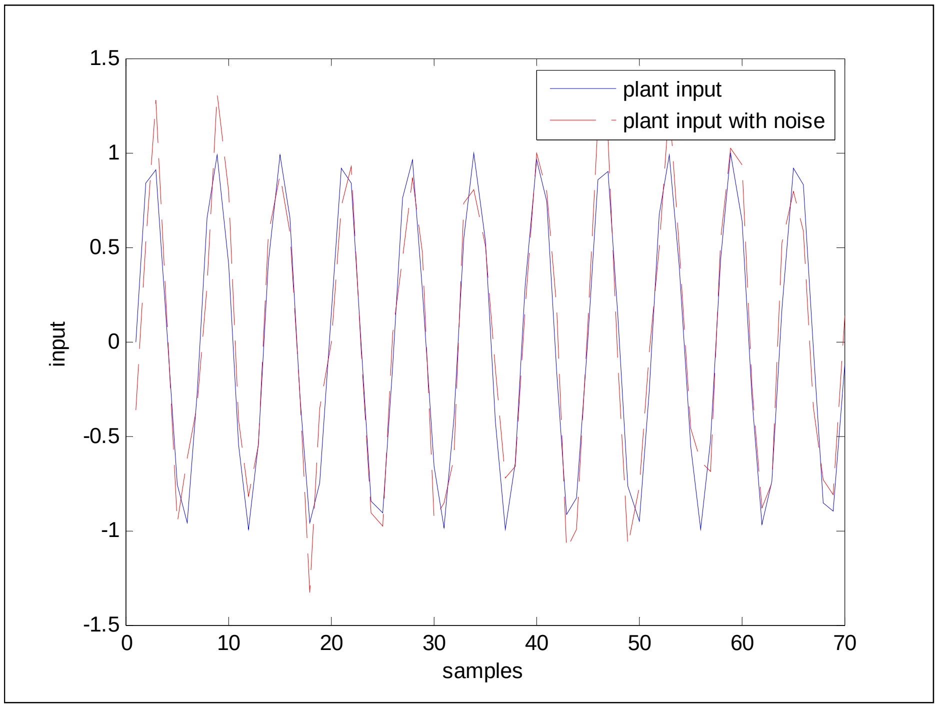

Figure 3.40 Plant input versus plant input with additive noise

Table 3.41 Noise performance results of plant III

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSR	0.156	8.110	0.164	7.84
Nonlinear ARX	0.37	4.30	0.336	4.73
Prediction Error Method	1.51	-1.81	2.214	-3.45
Subspace Method	1.181	-0.70	1.223	-0.87
Wiener Model	0.204	6.89	0.235	6.27
Hammerstein Model	0.951	0.217	2.77	-4.42
Hammerstein-Wiener Model	0.286	5.43	0.71	1.44
Nonlinear ARMAX	0.79	1.012	0.823	0.842
Elman Network	0.152	8.18	0.235	6.287

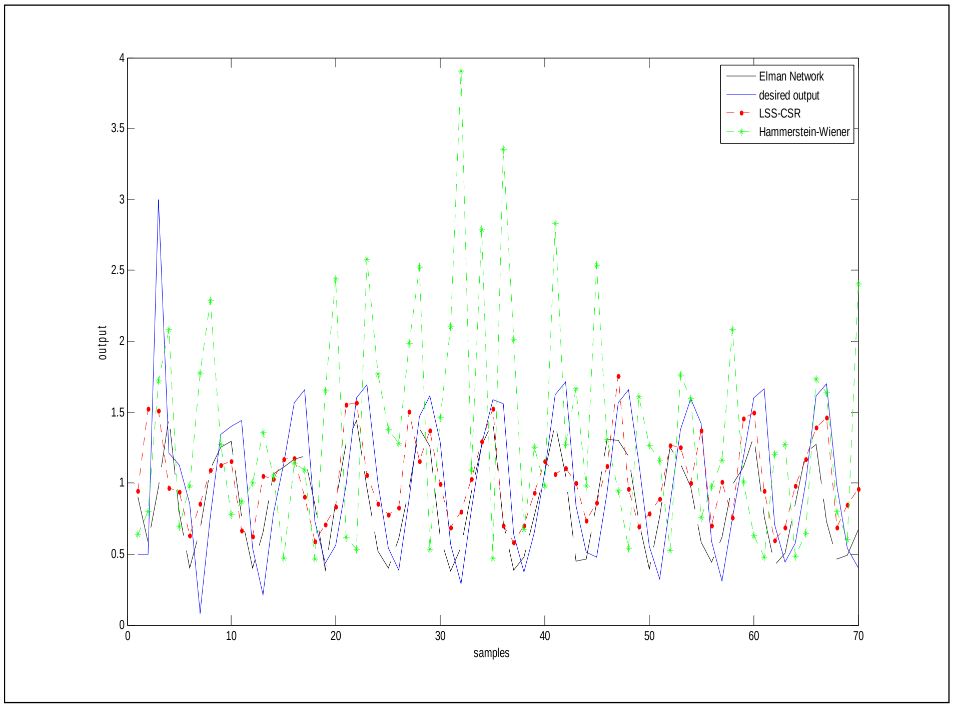

Figure 3.41 Comparison of outputs for different methods versus desired output of plant III with noise

3.3.6.4 Case 4: A chaotic system (Forced Duffing Oscillator)

In order to test the proposed method LSS-CSR under noise with a chaotic plant, the modified dynamics of the chaotic benchmark plant in the literature is borrowed which is shown in Equation 3.24. We can also write this equation with two state equations in Equation 3.25. This is chosen in order to test the performance of the proposed method with a chaotic plant. In cases I,II and III nonlinear and linear plants were tested with chaotic inputs. Forced Duffing oscillator were chosen as a chaotic plant because of its forcing input, since the proposed method needs input output data of the plant which is to be identified. Forced duffing oscillator states and the output are calculated under the initial conditions $x_1(0)$=3 and $x_2(0)$=4.

$$\ddot{x} = 7.6\cos(t) - 0.04\dot{x} - x^3 \tag{3.24}$$

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= 7.6\cos(t) - 0.04x_2 - x_1^3\\ y &= 0.05x_1 + 10\end{aligned} \tag{3.25}$$

The State ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For constructing the State ANN there are three inputs (two of them are the states of a henon system and a lorenz system, the other is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Output ANN has also 3 input neurons, two are coming from the State ANN output and the other is the plant input *u*(k). Like the State ANN, the Output ANN has 3 hidden neurons. Also the Output ANN has 1 output neuron for learning the output. In order to train the State ANN used modified Lorenz system is given in Equation (3.26) (Lorenz, 1963; Cuomo & Oppenheim, 1993).

$$\begin{aligned}\dot{x}_1 &= \sigma(x_2 - x_1)\\ \dot{x}_2 &= rx_1 - x_2 - 20x_1x_3\\ \dot{x}_3 &= 5x_1x_2 - bx_3\end{aligned} \tag{3.26}$$

with the parameters σ, *r*, *b*, and initial conditions $x_1(0)$, $x_2(0)$, and $x_3(0)$ are 10, 56.6, 5.02, 1, 0, and 1, respectively.

In order to train the State ANN, the following modified Henon system (Dick & Kandel, 2005) is used.

$$y(k+1) = 1.4y(k)^2 + 0.3y(k-1) + 1 \tag{3.27}$$

The phase portrait of the Henon system in Equation (3.27) is given under the initial conditions *y*(-1)=0.5 and *y*(0)=0.5.

Noise performance is tested with a uniformly distributed noise between [-2 2]. A uniformly distributed noise is added with the plant input which is a sinusoidal input sin(k) where k is from 0 to 70 and the system performance is tested. Figure 3.42 shows the plant input with additive noise and without noise.

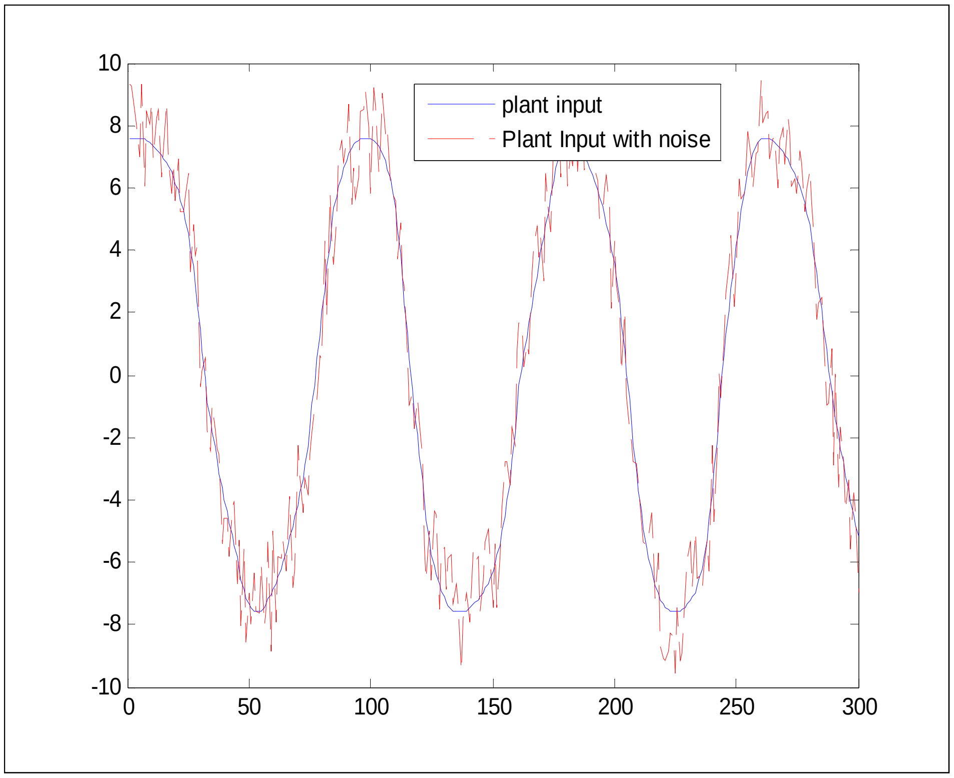

Figure 3.42 Plant input versus plant input with additive noise

Table 3.42 Noise performance results of plant IV

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSR	0.0118	19.288	0.0145	18.338
Nonlinear ARX	0.0688	11.62	0.0688	11.62
Prediction Error Method	2.541	-4.05	2.781	-4.44
Subspace Method	1.013	-0.056	1.017	-0.072
Wiener Model	0.151	8.204	0.153	8.153
Hammerstein Model	0.167	7.75	0.5276	2.776
Hammerstein-Wiener Model	0.295	5.293	47.729	-16.78
Nonlinear ARMAX	0.069	11.56	0.07	11.456
Elman Network	0.0335	14.750	0.0356	14.48

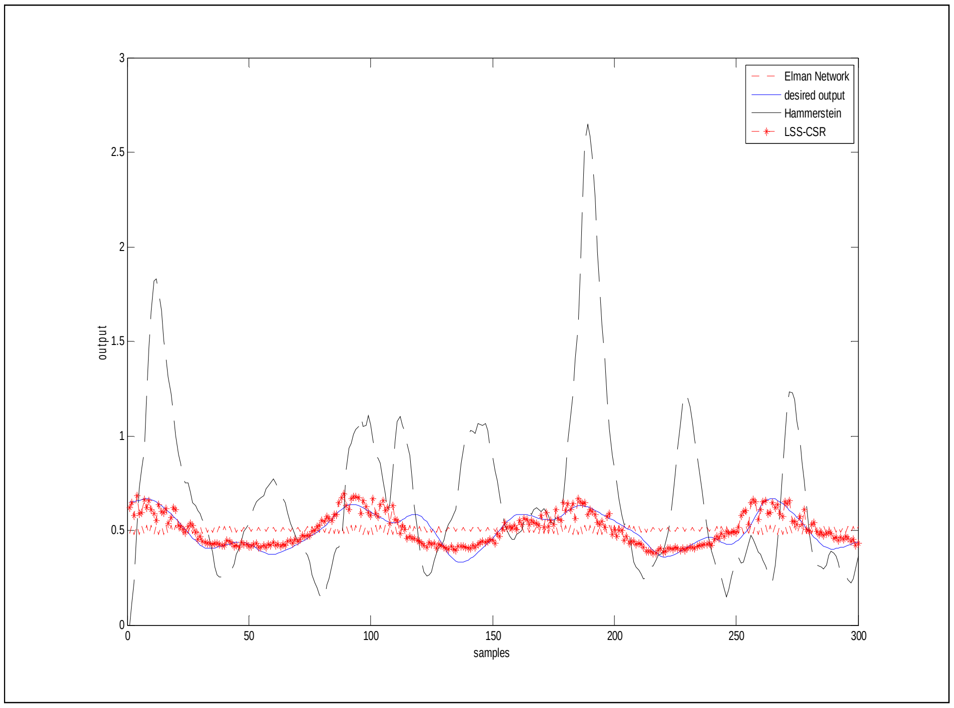

Figure 3.43 Comparison of outputs for different methods versus desired output of plant IV with noise

CHAPTER FOUR
LEARNING STATE SPACE WITH CHAOTIC STATE OBSERVER

Chapter 4 presents the identification method named as Learning State Space with Chaotic State Observer (LSS-CSO). LSS-CSO is designed to construct the state equation of the system based on input output data. This method employs a state observer in addition to exploiting chaotic signals and white Gaussian noise in the training of state equations. LSS-CSO employs three artificial neural networks which are designed for observing the states and estimating the output. Four benchmark plants (one is linear the other is chaotic and the other two are nonlinear) are used for testing the performance of the method. The method is used to identify the linear and nonlinear plants with different input types; the chaotic one which is forced duffing equation uses the input which is forcing the system. Four different inputs are applied to linear and nonlinear plants which are step, chaotic and sinoisoidal with two different frequencies. The method is also tested with different initial conditions for the benchmark plants. In all cases signal to error ratio and normalized mean square error are calculated for both evaluating training and test performances of the identification method. The proposed method is also compared with the benchmark identification methods, Nonlinear ARX, Nonlinear ARMAX, Hammerstein, Wiener, Hammerstein-Wiener, Subspace, Prediction Error Method and Elman Network. Noise performance of the proposed method is evaluated at the last stage and its performance is also compared with the benchmark methods stated above.

4.1 Learning State Space with Chaotic State Observer

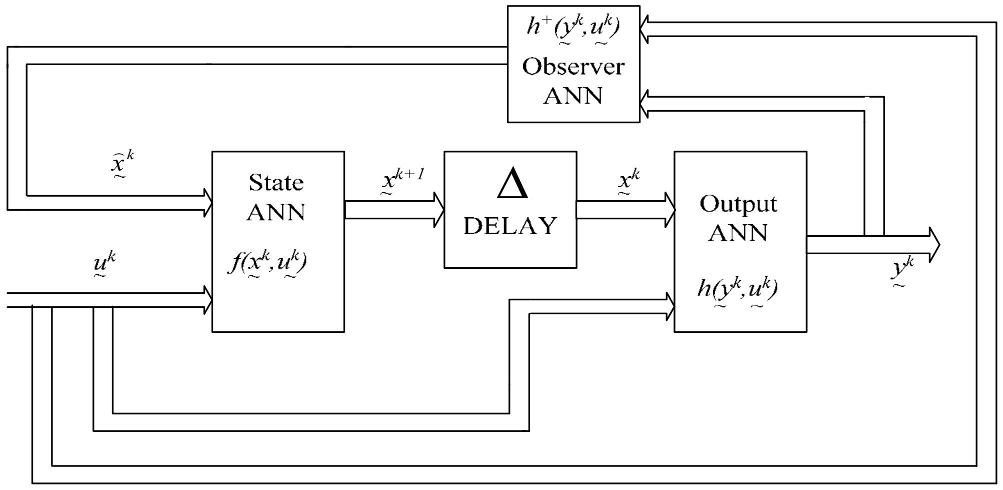
$h^{+}(\underset{\sim}{y}^k,\underset{\sim}{u}^k)$
Observer ANN
$\underset{\sim}{\hat{x}}^k$
State ANN
$f(\underset{\sim}{x}^k,\underset{\sim}{u}^k)$
$\underset{\sim}{x}^{k+1}$
Δ DELAY
$\underset{\sim}{x}^k$
Output ANN
$h(\underset{\sim}{y}^k,\underset{\sim}{u}^k)$
$\underset{\sim}{u}^k$
$\underset{\sim}{y}^k$

Figure 4.1 Learning State Space with Chaotic State Observer Block Diagram

The proposed method named LSS-CSO works as follows; firstly assign x^k as white gaussian noise (if the black-box system to be identified is chaotic then assign x^k's as a known chaotic systems states). Then with the known input u^k and known output y^k train the Output ANN and Observer ANN as an Auto association network to learn the states x^k's. Use Observer ANN output to generate $\hat{x}^k$'s. At the last stage use $\hat{x}^k$'s, x^{k+1}'s, and u^k training set to design State ANN

- Train Output ANN + Observer ANN together as an autoassociation network. Then hold this couple fixed.
- Now use u^k, y^k to generate $\hat{x}^k$'s.
- Use u^k, x^k, $\hat{x}^k$ training set to design state ANN.

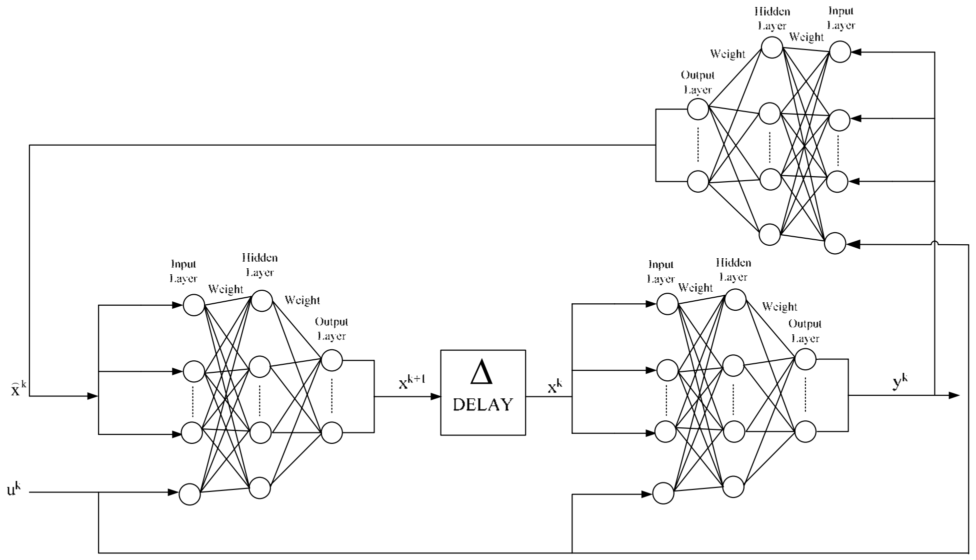

Figure 4.2 Learning State Space with Chaotic State Observer Method Structure

The State ANN architecture is taken like in the LSS-CSR method which is having 3 input neurons, one for the system input and two others for the current value of 2 state variables, and to have 2 output neurons each of which represents the next value of one of 2 state variables. And also the Output ANN is also taken same with the LSS-CSR method which is having 3 input neurons again one for the system input and two others for the current value of 2 state variables, and it has a single output neuron representing the system output. The observer ANN of the LSS-CSO method has 2 input neurons, one for the system input and the other for the system output and 2 output neurons representing the current values of the state variables. 3 hidden neurons are observed to be sufficient for all of the state, observer and output ANNs and so the identification results are obtained for 3 hidden neuron ANNs in all experiments. All ANNs are trained with the gradient descent type back propagation with the momentum term 0.9 and with a constant learning 0.05 for tansig (i.e. the hyperbolic tangent sigmoid) activation function in the input and hidden layers, linear transfer function in the output layer.

4.2 Simulation Results of the Proposed Method LSS-CSO

4.2.1. Test I

In order to test the proposed method LSS-CSO, the dynamics of the benchmark plant in the literature is borrowed (Narendra, 1996).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (1+v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{4.1}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are estimated states, 1 of them is the input which is *v(k)*), 3 hidden layer neurons and 2 output neurons which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neurons which are the estimates of the states. The Output ANN has 3 input neurons which are states (2 states) and the plant I input *v*(k), has 3 hidden neurons and 1 output neuron which represents the system output. The method is tested with the structure above and with 4 different types of input (*v*(k)) cases listed in Table 4.1.

Table 4.1 Inputs used for testing method LSS-CSO with plant in Equation (4.1)

Input I	*v*(k)=step input
Input II	*v*(k) = sin(k)
Input III	*v*(k)= sin(20k)
Input IV	*v*(k+1)=4*v*(k)(1-v(k)) , v(0)=0.1 (Chaotic input)

.

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The both initial conditions $x_1(0)$, $x_2(0)$, are also fixed to constant values as 0.5.

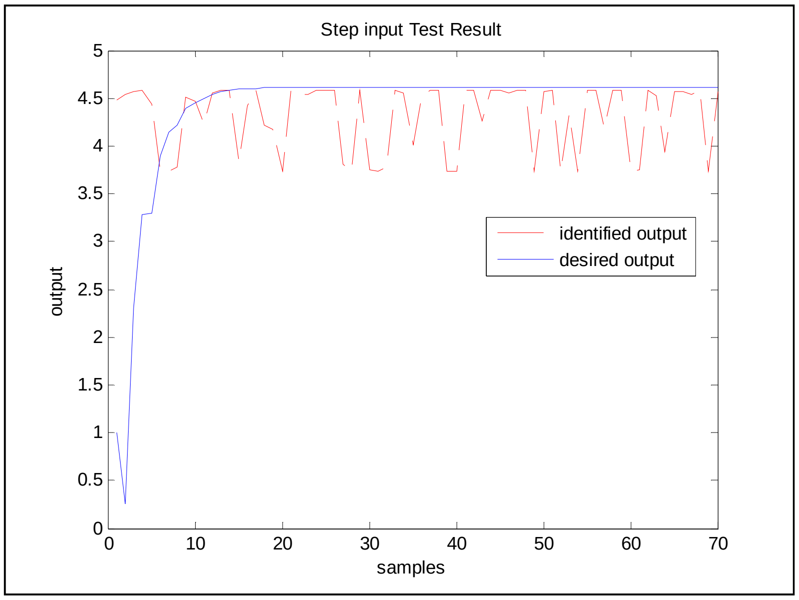

Figure 4.3 LSS-CSO method output versus desired output for step input with plant I

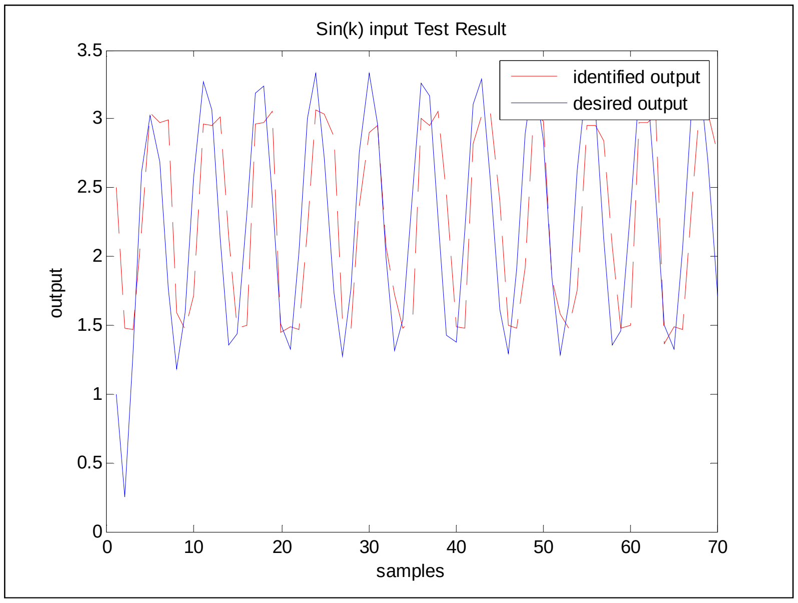

Figure 4.4 LSS-CSO method output versus desired output for sin(k) input with plant I

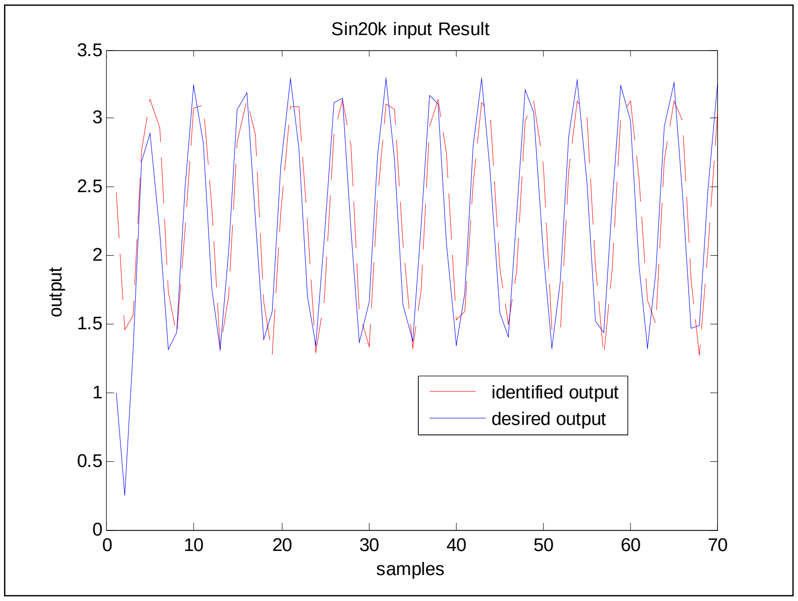

Figure 4.5 LSS-CSO method output versus desired output for sin(20k) input with plant I

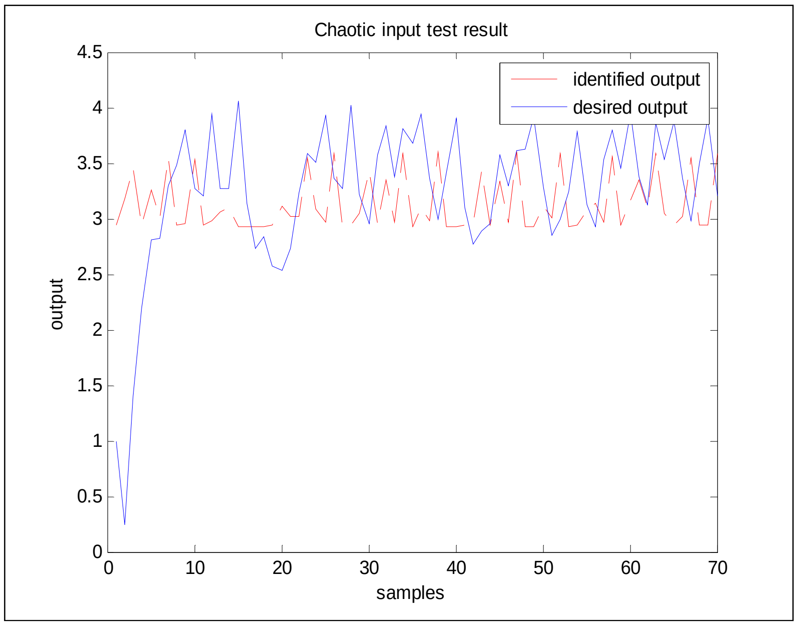

Figure 4.6 LSS-CSO method output versus desired output for chaotic input with plant I

Table 4.2 Performance of the proposed method LSS-CSO with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
LSS-CSO	**step**	0.0449	14.328	15.003	15.746	0.0380	13.015	14.197	15.0170
	Sin(k)	0.0550	13.405	14.218	15.466	0.0581	11.206	12.444	13.821
	Sin(20k)	0.0299	15.607	16.303	17.108	0.0330	14.600	15.202	16.185
	Chaotic input	0.0291	16.405	16.904	17.877	0.0562	12.404	13.0258	14.006

Table 4.3 Performance of Hammerstein model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	**Step**	0.0011	27.293	29.7	31.245	0.0022	25.0021	26.64	28.109
	Sin(k)	0.015	16.934	18.3	19.993	0.033	13.992	14.8	16.043
	Sin(20k)	0.1	9.345	9.85	10.450	2.78	-5.483	-4.43	-3.453
	Chaotic Input	0.003	22.988	24.8	26.030	0.016	16.833	17.8	18.2042

Table 4.4 Performance of Wiener model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	**Step**	$5*10^{-9}$	79.893	81.3	83.3329	0.0015	27.118	28.3	29.004
	Sin(k)	0.023	15.893	16.2	16.554	0.033	14.493	14.7	14.995
	Sin(20k)	0.07	10.942	11.3	11.539	0.13	8.2839	8.6	8.9983
	Chaotic Input	0.0046	22.0453	23.35	24.593	3.66	-7.773	-5.64	-3.2958

Table 4.5 Performance of Hammerstein-Wiener model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	**Step**	$8*10^{-9}$	78.984	80.8	82.298	0.0014	27.048	28.4	29.857
	Sin(k)	0.011	18.4893	19.44	20.002	0.035	13.984	14.58	14.779
	Sin(20k)	0.077	10.748	11.11	11.586	0.34	3.860	4.58	4.447
	Chaotic Input	0.008	20.116	20.9	21.796	0.014	17.7745	18.54	19.0238

Table 4.6 Performance of Elman network with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
ELMAN Network	**Step**	0.02	15.6013	16.92	22.85	0.024	14.640	16.21	17.1134
	Sin(k)	0.01	16.284	17.7	19.65	0.02	15.736	16.8	18.27
	Sin(20k)	0.019	16.458	17.01	19.004	0.023	14.985	16.3	17.697
	Chaotic Input	0.0418	10.151	13.79	15.8429	0.044	4.8977	13.58	15.185

Table 4.7 Performance of nonlinear ARX model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	0.046	12.744	13.31	13.698	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	0.09	9.904	10.4	10.842	0.177	7.220	7.5	7.702
	Chaotic Input	0.0234	15.839	16.31	16.696	0.057	11.995	12.39	12.893

Table 4.8 Performance of nonlinear ARMAX model with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	25.6	-15.694	-14.1	-13.685	32.34	-15.743	-15.1	-14.802
	Chaotic Input	$2*10^{-7}$	62.034	66.3	68.201	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 4.9 Performance of prediction error method with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	**Step**	4.10^{-5}	41.204	43.8	44.403	0.002	26.103	26.9	27.336
	Sin(k)	0.88	0.301	0.53	0.687	1.24	-1.112	-0.9	-0.791
	Sin(20k)	0.84	0.592	0.72	0.884	1.33	-1.559	-1.23	-1.172
	Chaotic Input	0.02	16.405	17	17.503	0.022	15.958	16.5	16.794

Table 4.10 Performance of subspace method with Plant I

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	**Step**	$2.6*10^{-6}$	52.203	55.7	57.804	0.003	24.495	25.25	25.796
	Sin(k)	0.83	0.613	0.8	0.934	0.87	0.419	0.59	0.793
	Sin(20k)	0.58	1.984	2.31	2.652	1.022	-0.113	-0.095	0.091
	Chaotic Input	0.019	16.849	17.22	17.558	0.045	12.994	13.5	13.894

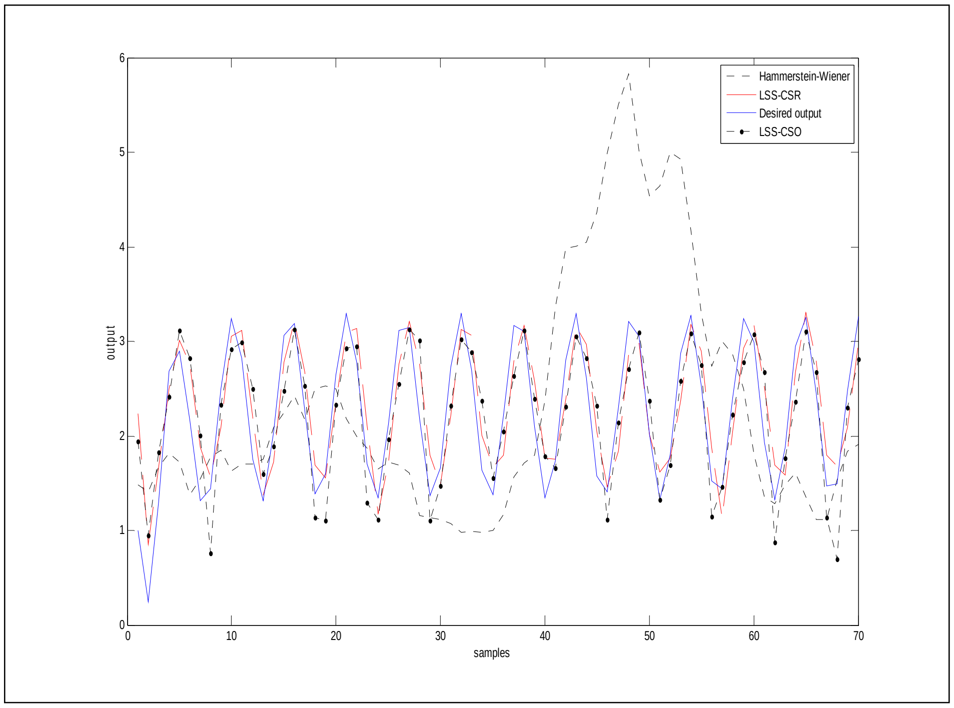

Figure 4.7 Comparison of outputs for different methods versus desired output for sin(20k) with plant I

4.2.2 Test II

In order to test the proposed method LSS-CSO, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996).

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{4.2}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are the estimated states, 1 of them is the input which is *u(k)*), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neuron which are estimates of the states. The Output ANN has 3 input neurons which are the states and the plant input *u*(k), has 3 hidden neurons and 1 output neuron which represents the system output. The method is tested with the structure above and with 4 different types of input (u(k)) cases listed in Table 4.11.

Table 4.11 Inputs used for testing method LSS-CSO with plant II in Equation (4.2)

Input I	u(k)=step input
Input II	u(k) = sin(k)
Input III	u(k)= sin(20k)
Input IV	u(k+1)=4u(k)(1-u(k)) , u(0)=0.1 (Chaotic input)

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial condition $x(0)$ is fixed to a constant value 0.5.

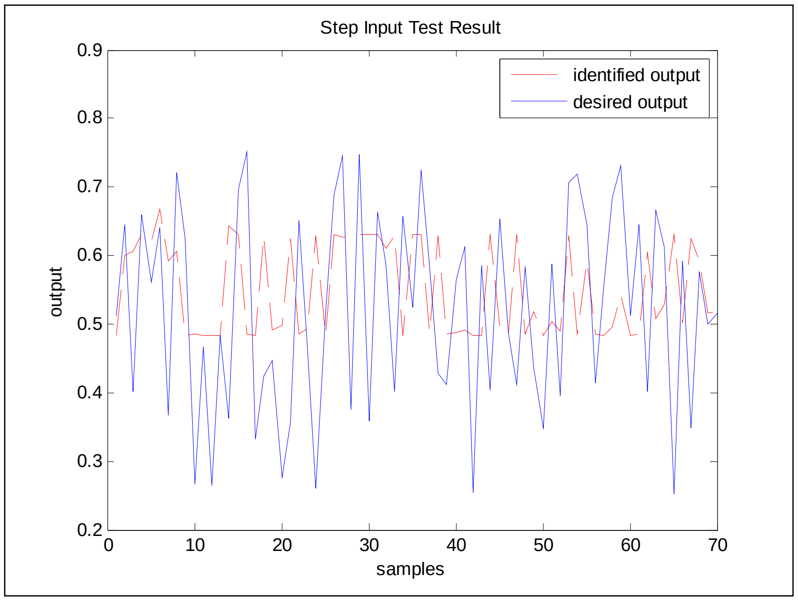

Figure 4.8 LSS-CSO method output versus desired output for step input with plant II

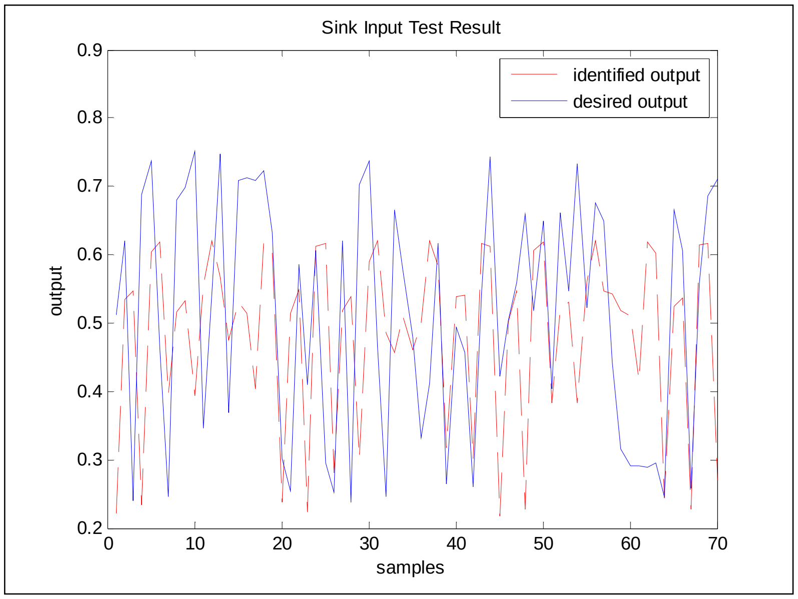

Figure 4.9 LSS-CSO method output versus desired output for sin(k) input with plant II

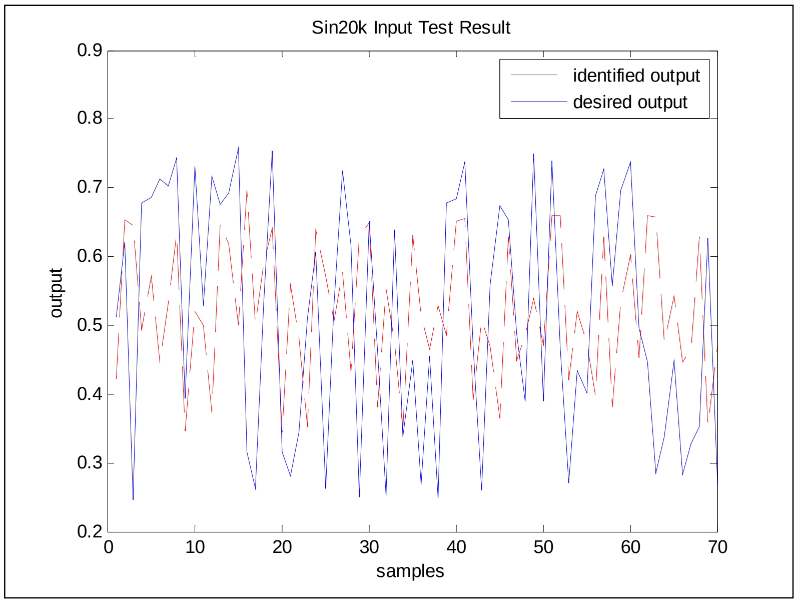

Figure 4.10 LSS-CSO method output versus desired output for sin(20k) input with plant II

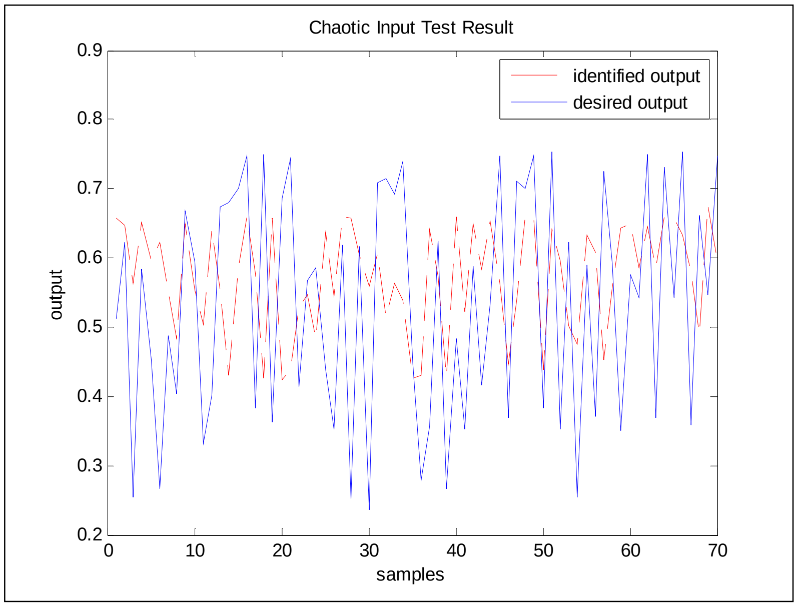

Figure 4.11 LSS-CSO method output versus desired output for chaotic input with plant II

Table 4.12 Performance of the proposed method LSS-CSO with Plant II

	INPUT	**nMSE (training)**	**SER (dB) (training) (Min)**	**SER (dB) (training) (Mean)**	**SER (dB) (training) (Max)**	**nMSE (test)**	**SER (dB) (test) (Min)**	**SER (dB) (test) (Mean)**	**SER (dB) (test) (Max)**
LSS-CSO	**step**	0.078	12.442	12.646	12.905	0.068	11.022	11.638	11.910
	Sin(k)	0.091	11.682	11.944	12.481	0.094	10.152	10.225	10.496
	Sin(20k)	0.085	12.104	12.380	12.682	0.115	9.118	9.478	9.904
	Chaotic input	0.070	12.705	13.188	13.506	0.086	10.275	10.609	10.974

Table 4.13 Performance of Hammerstein model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	Step	0.088	10.0221	10.54	10.774	0.105	9.0903	9.78	10.001
	Sin(k)	0.52	2.519	2.82	2.9957	1.24	-2.73	-0.94	-0.052
	Sin(20k)	0.06	11.873	12.14	12.583	0.15	7.757	8.26	8.403
	Chaotic Input	0.08	10.589	10.84	10.9086	0.69	1.322	1.58	1.7128

Table 4.14 Performance of Wiener model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	Step	0.063	11.790	12.00	12.205	0.084	10.396	10.75	10.884
	Sin(k)	0.038	13.956	14.18	14.446	0.208	6.665	6.81	6.942
	Sin(20k)	0.064	11.674	11.9	12.034	0.122	8.854	9.13	9.438
	Chaotic Input	0.054	12.283	12.64	12.805	0.097	9.850	10.11	10.426

Table 4.15 Performance of Hammerstein-Wiener model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	Step	0.073	10.994	11.33	11.559	0.079	10.744	11.02	11.259
	Sin(k)	0.057	11.773	12.4	12.694	0.17	7.573	7.70	7.902
	Sin(20k)	0.072	10.954	11.4	11.778	0.115	8.785	9.39	9.505
	Chaotic Input	0.055	11.998	12.59	12.874	0.53	1.992	2.75	2.995

Table 4.16 Performance of Elman network with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
ELMAN Network	**Step**	0.088	10.2265	10.55	10.971	0.091	10.002	10.42	10.769
	Sin(k)	0.1026	9.521	9.888	9.9555	0.1058	9.3862	9.6453	9.8197
	Sin(20k)	0.1030	9.623	9.86	10.1693	0.1232	8.7340	9.0928	9.4004
	Chaotic Input	0.084	10.362	10.76	11.0676	0.1004	9.752	9.98	10.3296

Table 4.17 Performance of nonlinear ARX model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	$7.7*10^3$	-42.349	-38.89	-35.340	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 4.18 Performance of nonlinear ARMAX model with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	**Step**	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	11.65	-11.129	-10.66	-9.902	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(20k)	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 4.19 Performance of prediction error method with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	**Step**	0.082	9.964	10.84	10.993	0.101	9.504	9.94	10.023
	Sin(k)	0.74	1.112	1.32	1.432	2.09	-4.032	-3.2	-2.893
	Sin(20k)	1.028	-0.304	-0.12	0.003	1.18	-1.114	-0.74	-0.504
	Chaotic Input	0.102	9.603	9.90	10.005	0.141	7.920	8.48	8.7732

Table 4.20 Performance of subspace method with Plant II

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	**Step**	0.081	10.703	10.9	11.003	0.113	8.945	9.46	9.603
	Sin(k)	0.77	0.794	1.115	1.203	1.118	-0.505	-0.48	-0.385
	Sin(20k)	0.95	0.159	0.203	0.293	1.004	-0.100	-0.018	0.030
	Chaotic Input	0.075	10.403	11.23	11.584	0.18	6.892	7.44	7.771

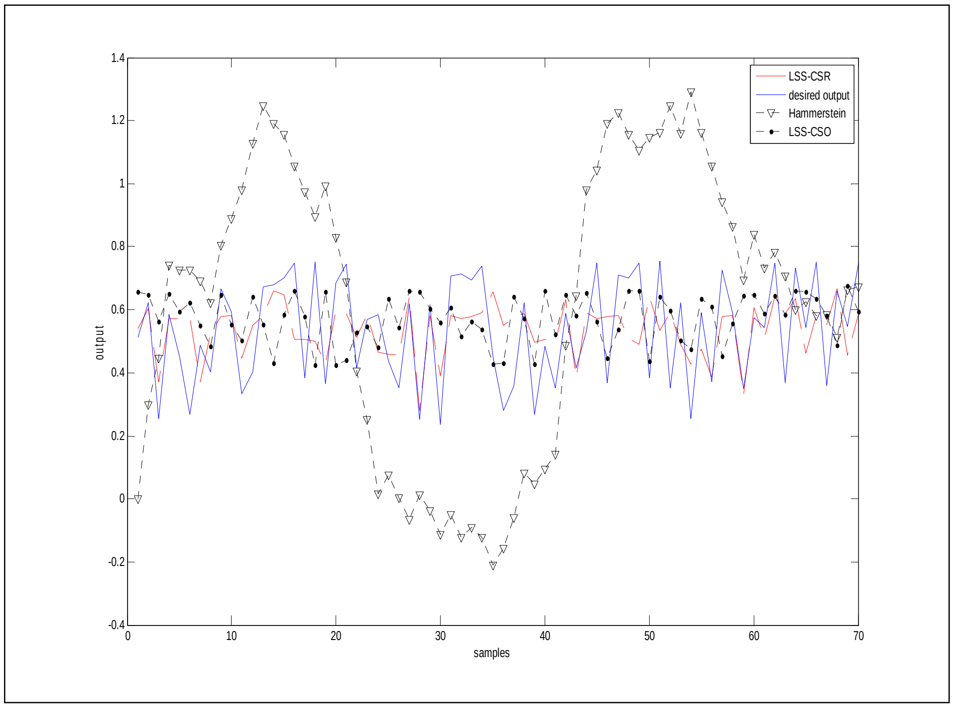

Figure 4.12 Comparison of outputs for different methods versus desired output for chaotic input with plant II

4.2.3 Test III

In order to test the proposed method LSS-CSO with a more complicated nonlinear plant, the dynamics of the benchmark modified plant in literature is borrowed (Uykan & others,2000) .

Four different inputs are applied to the plant III which are step, chaotic and sinusoidal signals with two different frequencies. For the step and chaotic inputs the Equation (4.3) is used, but for sinusoidal inputs Equation (4.4) is used to provide stability.

$$\begin{aligned} x(k+1) &= \frac{9.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \\ y(k) &= x(k)+1 \end{aligned} \tag{4.3}$$

$$x(k+1) = \frac{2.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \qquad (4.4)$$
$$y(k) = x(k)+1$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are the estimates of states, 1 of them is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which is used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neuron which are the estimates of states. The Output ANN has 3 input neurons which of two are the states and the plant input *u*(k), has 3 hidden neurons and 1 output neuron which represents the system output.

The method is tested with the structure above and with 4 different types of input (*u*(k)) cases listed in Table 4.21.

Table 4.21 Inputs used for testing method LSS-CSO with the plant III

Input I	*u*(k)=step input
Input II	*u*(k) = sin(k)
Input III	*u*(k)= sin(20k)
Input IV	*u*(k+1)=4*u*(k)(1-u(k)) , *u*(0)=0.1 (Chaotic input)

.

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial condition *x*(-2), *x*(-1), and *x*(0) are fixed to a constant values 1, 1.5, and 2 respectively.

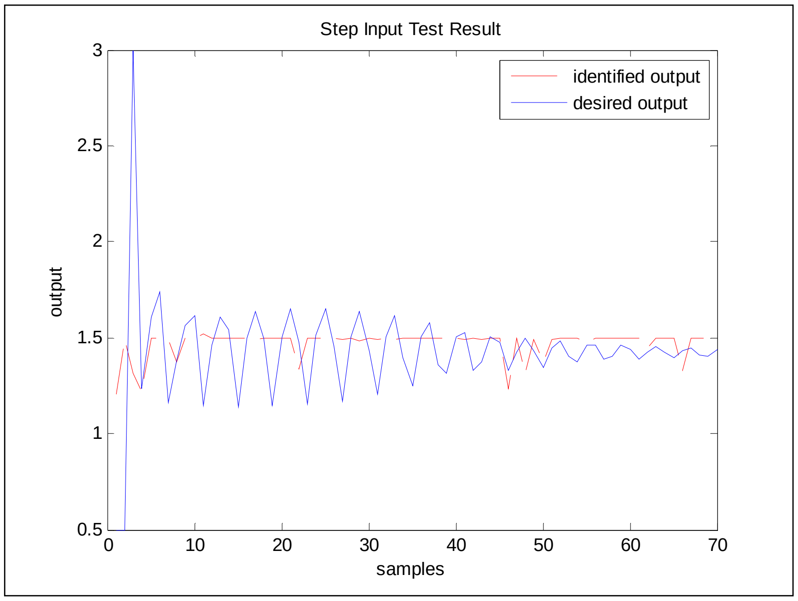

Figure 4.13 LSS-CSO method output versus desired output for step input with plant III

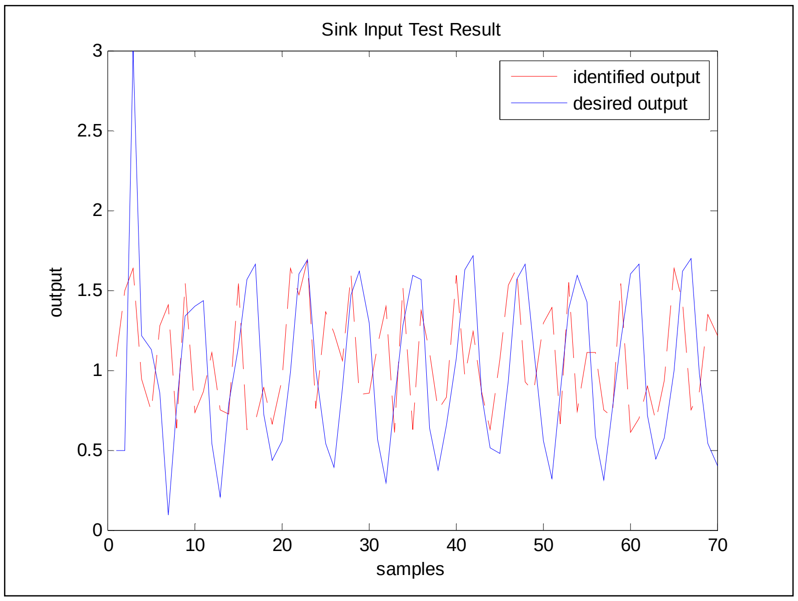

Figure 4.14 LSS-CSO method output versus desired output for sin(k) input with plant III

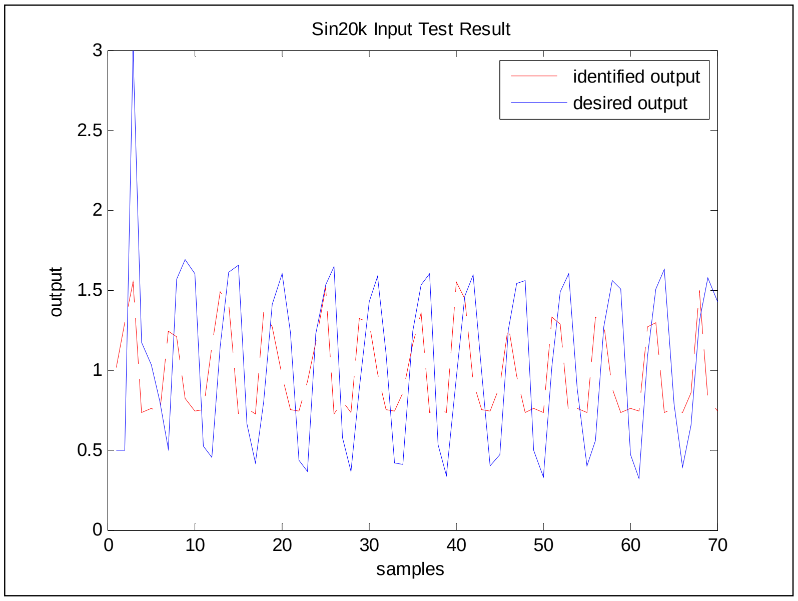

Figure 4.15 LSS-CSO method output versus desired output for sin(20k) input with plant III

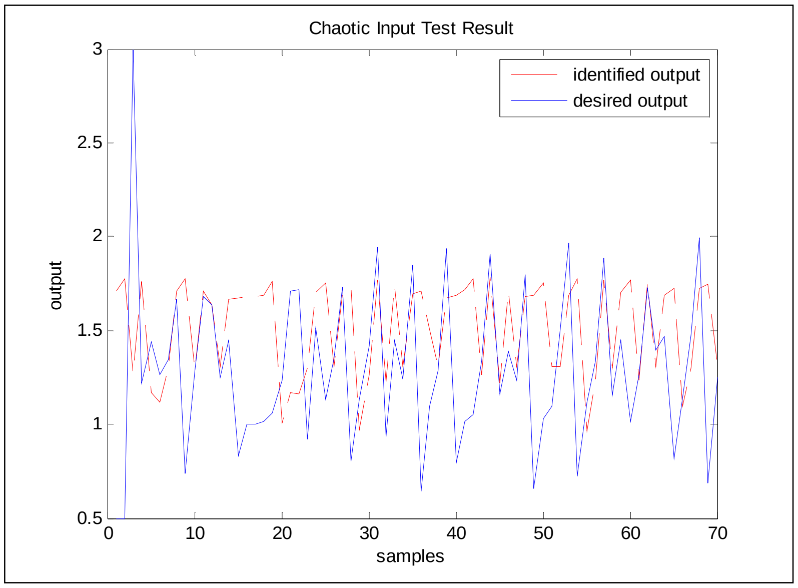

Figure 4.16 LSS-CSO method output versus desired output for chaotic input with plant III

Table 4.22 Performance of the proposed method LSS-CSO with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
LSS-CSO	**step**	0.0265	16.514	17.202	17.609	0.0364	14.128	14.301	14.725
	Sin(k)	0.148	9.348	9.917	10.168	0.220	8.027	8.213	8.526
	Sin(20k),	0.166	9.194	9.402	9.824	0.187	8.319	8.971	9.284
	Chaotic input	0.0762	12.215	12.718	12.993	0.100	9.875	10.073	10.256

Table 4.23 Performance of Hammerstein model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein	**Step**	0.0097	19.493	20.16	20.592	0.029	14.993	15.36	15.704
	Sin(k)	0.112	8.994	9.49	9.743	0.951	0.003	0.217	0.702
	Sin(20k)	0.038	13.895	14.11	14.443	0.853	-0.034	0.69	1.008
	Chaotic Input	0.047	12.904	13.28	13.603	0.098	9.856	10.08	10.395

Table 4.24 Performance of Wiener model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Wiener	**Step**	0.004	22.9403	24.35	25.503	0.032	14.553	14.92	15.254
	Sin(k)	0.078	10.873	11.03	11.110	0.204	6.201	6.89	6.970
	Sin(20k)	0.066	11.593	11.800	11.990	0.659	1.504	1.809	2.002
	Chaotic Input	0.0351	13.984	14.54	14.785	0.097	9.795	10.13	10.376

Table 4.25 Performance of Hammerstein-Wiener model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Hammerstein–Wiener	Step	0.007	20.693	21.52	22.012	0.026	15.405	15.85	15.996
	Sin(k)	0.016	17.302	17.89	18.003	0.286	5.274	5.43	5.596
	Sin(20k)	0.024	15.785	16.19	16.394	0.62	1.806	2.05	2.226
	Chaotic Input	0.032	14.468	14.88	15.067	0.103	9.572	9.83	10.043

Table 4.26 Performance of Elman network with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
ELMAN Network	Step	0.025	15.643	16.05	17.0735	0.042	13.483	13.83	14.5639
	Sin(k)	0.105	7.3596	9.79	12.385	0.152	5.9402	8.18	10.362
	Sin(20k)	0.224	6.0054	6.501	7.0129	0.272	5.236	5.64	5.952
	Chaotic Input	0.08	10.102	10.99	11.8677	0.1028	9.473	9.8791	10.6933

Table 4.27 Performance of nonlinear ARX model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARX	Step	0.01	19.784	19.95	20.506	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	0.22	6.382	6.55	6.770	0.37	3.336	4.30	4.508
	Sin(20k)	0.096	9.784	10.15	10.384	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Chaotic Input	0.137	8.404	8.63	8.732	Non Convergent	Non Convergent	Non Convergent	Non Convergent

Table 4.28 Performance of nonlinear ARMAX model with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Nonlinear ARMAX	Step	0.01	19.094	19.95	20.498	Non Convergent	Non Convergent	Non Convergent	Non Convergent
	Sin(k)	0.071	10.890	11.44	11.774	0.79	0.403	1.012	1.205
	Sin(20k)	2.032	-3.645	-3.08	-2.403	2.083	-3.676	-3.18	-2.806
	Chaotic Input	8.76	-10.003	-9.43	-8.884	10.36	-10.543	-10.15	-9.709

Table 4.29 Performance of prediction error method with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Prediction Error Method	Step	0.01	-19.504	19.98	20.302	0.018	16.985	17.23	17.406
	Sin(k)	0.49	2.589	3.07	3.453	1.51	-1.903	-1.81	-0.040
	Sin(20k)	0.949	0.002	0.22	0.303	2.26	-3.903	-3.54	-2.705
	Chaotic Input	0.088	9.905	10.52	10.813	0.105	9.604	9.76	9.895

Table 4.30 Performance of subspace method with Plant III

	INPUT	nMSE (training)	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
Subspace Method	Step	0.017	17.306	17.69	18.003	0.022	15.903	16.5	16.781
	Sin(k)	0.63	1.403	1.94	2.004	1.181	-1.023	-0.70	-0.068
	Sin(20k)	0.91	0.302	0.40	0.443	1.18	-1.059	-0.72	-0.174
	Chaotic Input	0.07	10.903	11.51	11.604	0.365	3.958	4.36	4.980

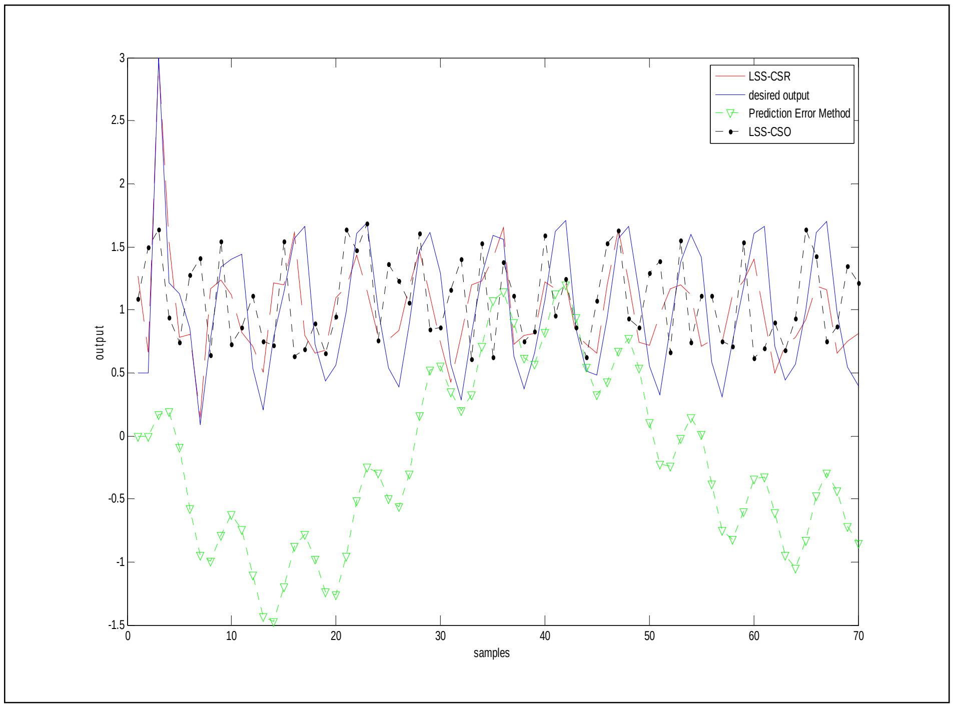

Figure 4.17 Comparison of outputs for different methods versus desired output for sin(k) with plant III

4.2.4 Test IV

In order to test the proposed method LSS-CSO, the modified dynamics of the chaotic benchmark plant in literature is borrowed which is shown in Equation (4.6). This equation can also be written with two state equations in Equation (4.7). This is chosen in order to test the performance of the proposed method LSS-CSO with a chaotic plant. In tests I, II and III nonlinear and linear plants were tested with chaotic inputs. Forced Duffing oscillator were chosen as a chaotic plant because of its forcing input, since the proposed method needs input output data of the plant which is to be identified. Forced duffing oscillator ran under the initial conditions in Equation (4.5).

$$x_1(0)=3 \text{ and } x_2(0)=4 \tag{4.5}$$

$$\ddot{x} = 7.6\cos(t) - 0.04\dot{x} - x^3 \tag{4.6}$$

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= 7.6\cos(t) - 0.04x_2 - x_1^3 \\ y &= 0.05x_1 + 10\end{aligned} \tag{4.7}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are the estimates of states, 1 of them is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neuron which are the states. The Output ANN has 3 input neurons which are the states and the plant input *u*(k), has 3 hidden neurons and 1 output neuron which represents the system output. In order to train the Output ANN used modified Lorenz system is given in Equation (4.8) (Lorenz, 1963 ,Cuomo & Oppenheim, 1993).

$$\begin{aligned}\dot{x}_1 &= \sigma(x_2 - x_1) \\ \dot{x}_2 &= rx_1 - x_2 - 20x_1x_3 \\ \dot{x}_3 &= 5x_1x_2 - bx_3\end{aligned} \tag{4.8}$$

with the parameters σ, *r*, *b*, and initial conditions $x_1(0)$, $x_2(0)$, and $x_3(0)$ are 10, 56.6, 5.02, 1, 0, and 1, respectively.

In order to train the State ANN used modified henon system is given in Equation (4.9) (Dick & Kandel; 2005).

$$y(k+1) = 1.4y(k)^2 + 0.3y(k-1) + 1 \tag{4.9}$$

The phase portrait of the henon system in Equation (4.9) is given in Figure 4.38 under the initial conditions *y*(-1)=*0.5* and *y*(0)=*0.5*. Train and test performances of the proposed method and the benchmark methods taken from the literature are given in Table 4.31.

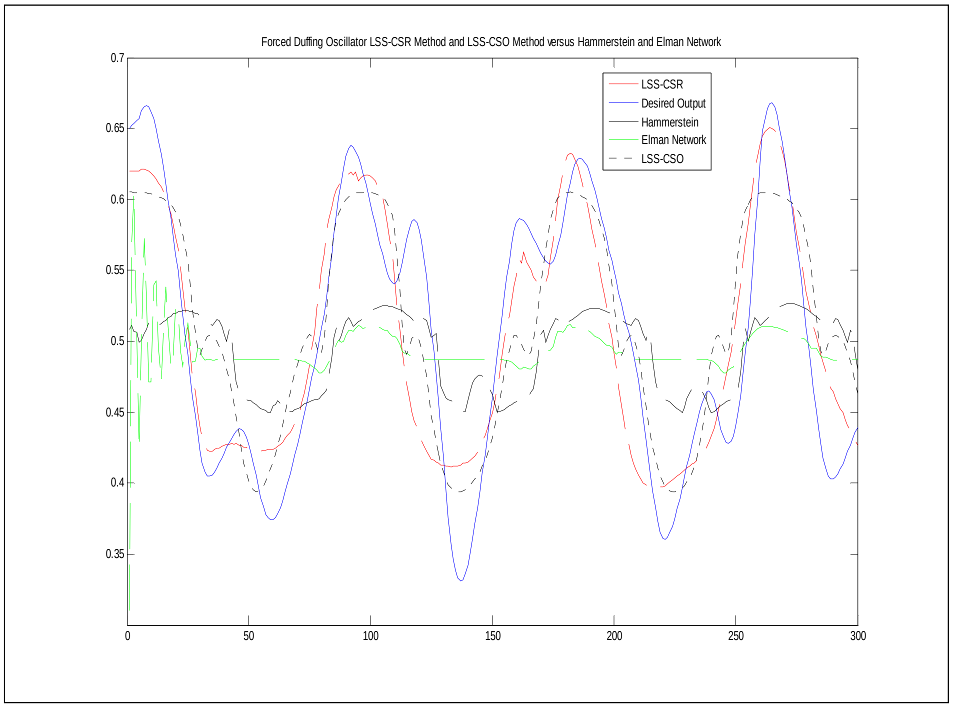

Figure 4.18 Comparison of outputs for different methods versus desired output with plant IV

Table 4.31 Comparison of train and test performances of the proposed method and the benchmark methods

Method - Model	nMSE training	SER (dB) (training) (Min)	SER (dB) (training) (Mean)	SER (dB) (training) (Max)	nMSE (test)	SER (dB) (test) (Min)	SER (dB) (test) (Mean)	SER (dB) (test) (Max)
LSS-CSO	0.0092	19.993	20.305	21.412	0.010	19.673	19.971	20.552
Nonlinear ARX	0.028	15.102	15.53	15.721	0.0688	11.403	11.62	11.832
Prediction Error Method	0.197	6.663	7.03	7.904	2.541	-4.589	-4.05	-3.453
Subspace Method	0.749	0.432	1.25	1.698	1.013	-0.332	-0.056	0.102
Wiener Model	0.029	14.895	15.40	15.793	0.151	7.773	8.204	8.502
Hammerstein Model	0.0548	12.093	12.61	12.856	0.167	7.551	7.75	7.896
Hammerstein-Wiener Model	0.0275	15.398	15.604	15.873	0.295	4.706	5.293	5.554
Nonlinear ARMAX	0.039	13.794	14.092	14.443	0.069	11.394	11.56	11.707
Elman Network	0.0320	14.607	14.9446	15.254	0.0335	14.392	14.750	14.967

4.2.5 Performance of the proposed method LSS-CSO with different initial conditions

4.2.5.1 Test a

In order to test the proposed method LSS-CSO to different initial conditions; the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) given in Equation (4.10).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (1+v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{4.10}$$

The plant input is a unit step input (*u*(k)) The benchmark plant is executed with different initial conditions. State ANN and Output ANN are trained with some possible values of initial conditions. For the plant above x(1) is varied between 0 and 19.99 and the State ANN and Output ANN are trained with 70 different initial conditions. Then the method is tested with different initial conditions that are not placed in the training set.

The Figure 4.19 shows the identified and desired outputs which are the responses of x_1(k)=5 and x_2(k)=0.5. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets.

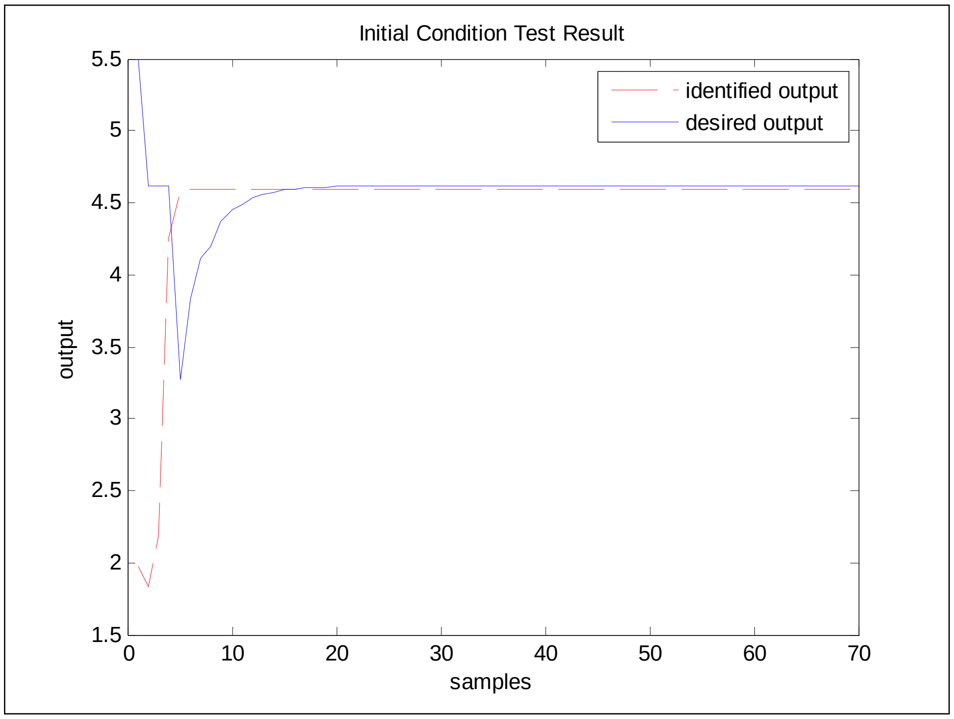

Figure 4.19 Initial condition Test Result of Proposed Method LSS-CSO with Plant I

The performance parameters of the test point x_1(k)=5 and x_2(k)=0.5 which is shown in the Figure 4.19 are calculated and they are shown in Table 4.32 as follows;

Table 4.32 Initial condition Performance parameters of method LSS-CSO with Plant I

SER (dB)	17.0138
MSE	0.4160
nMSE	0.0199

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 4.33.

Table 4.33 Initial condition SER parameters of Method LSS-CSO with Plant I

SER (worst) (dB)	16.788
SER (mean) (dB)	17.512
SER (best) (dB)	18.174

4.2.5.2 Test b

The proposed method LSS-CSO performance is tested with different initial conditions for a nonlinear plant which the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996) and shown in Equation 4.11.

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{4.11}$$

The plant II input is a sinusoidal input (sink) where k is from 0 to 70. The benchmark plant is executed with different initial conditions. State ANN and Output ANN are trained with some possible values of initial conditions. For the plant above $x(0)$ is varied between 0 and 89.99 and the State ANN and Output ANN are trained with 70 different initial conditions. Then the method is tested with different initial conditions that are not placed in the training set.

The Figure 4.20 shows the identified and desired outputs which are the responses of x_1(k)=80. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets.

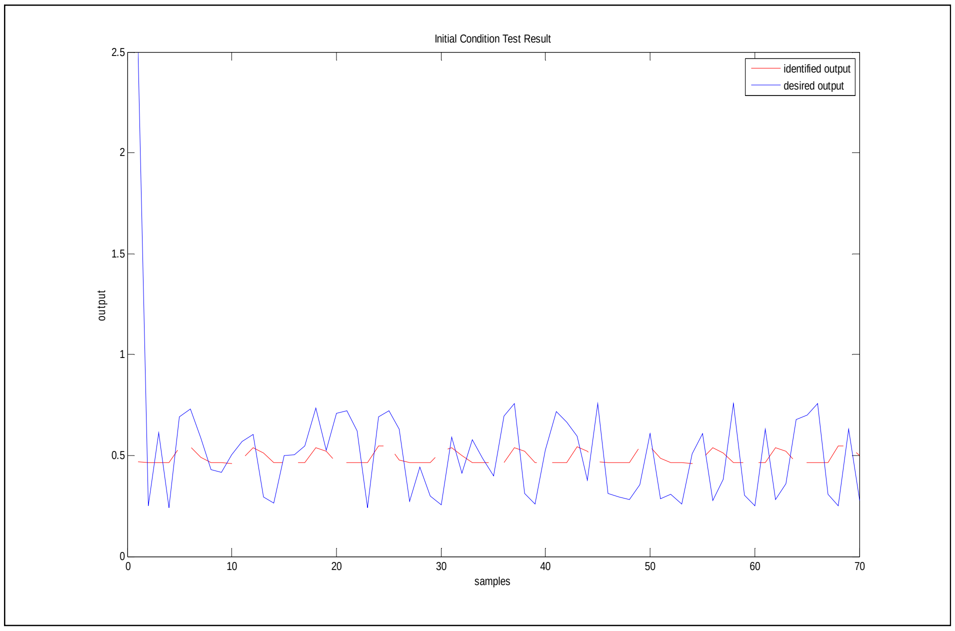

Figure 4.20 Initial condition Test Result of Proposed Method LSS-CSO with Plant II

The performance parameters of the test point x_1(k)=80 which is shown in the Figure 4.20 are calculated and they are shown in Table 4.34 as follows.

Table 4.34 Initial condition Performance parameters of Method LSS-CSO with Plant II

SER (dB)	5.975
MSE	0.100
nMSE	0.255

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 4.35.

Table 4.35 Initial condition SER parameters of Method LSS-CSO with Plant II

SER (worst) (dB)	5.669
SER (mean) (dB)	5.814
SER (best) (dB)	6.107

4.2.5.3 Test c

In order to test the proposed method LSS-CSO to different initial conditions; the dynamics of the benchmark plant in literature is borrowed (Uykan & others, 2000).

$$\begin{aligned} x(k+1) &= \frac{9.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \\ y(k) &= x(k)+1 \end{aligned} \tag{4.12}$$

To test the proposed model, the benchmark plant is executed with different initial conditions and with the step input u. Plant ANN and output ANN are trained with some possible values of initial conditions. For the benchmark plant above x_1(k) is varied from 0 to 2.99.

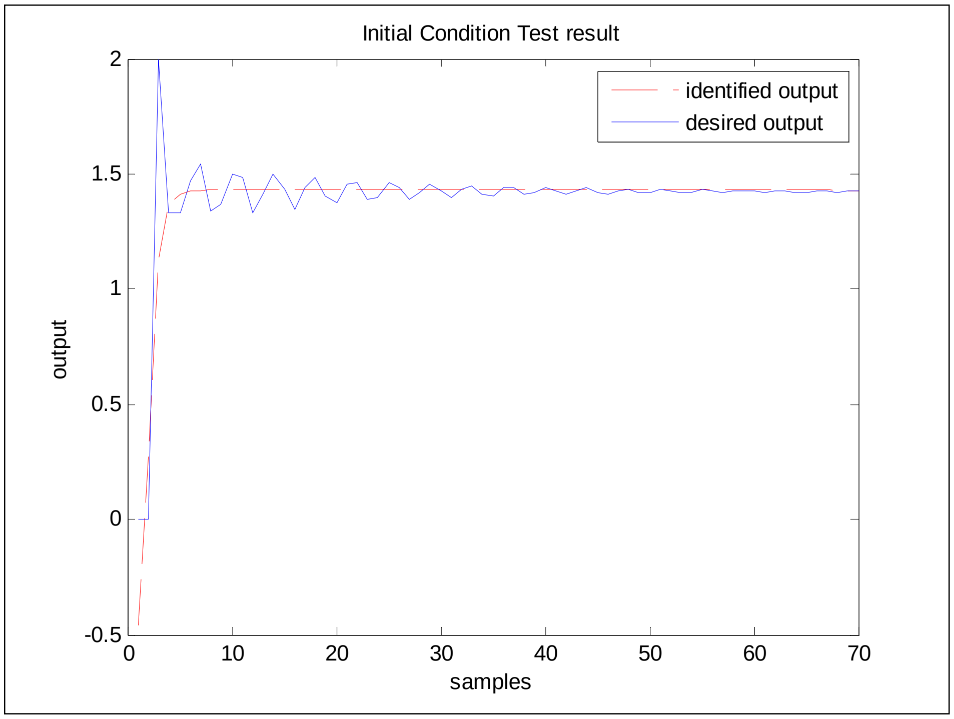

Figure 4.21 Initial condition Test Result of Proposed Method LSS-CSO with Plant III

The above figure shows the identified and desired outputs which are the responses of x_1(k)=1. Mean Square error, normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for test data sets and they are shown in Table 4.36 as follows;

Table 4.36 Initial condition Performance parameters of method LSS-CSO with Plant III

SER (dB)	20.872dB
MSE	0.0165
nMSE	0.0083

The performance of the system tested for various initial conditions, 100 different cases for train and test data are chosen and SER values are calculated the best worst and the mean are listed in Table 4.37.

Table 4.37 Initial condition SER parameters of Method LSS-CSO with Plant II

SER (worst) (dB)	19.784dB
SER (mean) (dB)	20.772dB
SER (best) (dB)	21.496dB

4.2.6 Noise Performance of the Proposed method LSS-CSO

4.2.6.1 Case 1: A second order linear state model

In order to test the Proposed Method LSS-CSO, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996).

$$\begin{aligned} x_1(k+1) &= 0.5x_2(k) \\ x_2(k+1) &= 0.5x_1(k) + 0.1x_2(k) + (u(k) + v(k)) \\ y(k) &= x_1(k) + x_2(k) \end{aligned} \tag{4.13}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are estimates of the states, 1 of them is the input which is *v*(k)), 3 hidden layer neurons and 2 output neurons which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neurons which are states. The Output ANN has 3 input neurons which are states (2 states) and the plant input *v*(k), has 3 hidden neurons and 1 output neuron which represents the system output.

Noise performance is tested with a uniformly distributed noise between [-0.40 0.40]. A uniformly distributed noise is added with the plant input, and the system performance is tested. Below the Figure 4.22 shows the plant input with additive noise and without noise. Plant I input is chosen a sinusoidal input which is a sinus sin(20k).

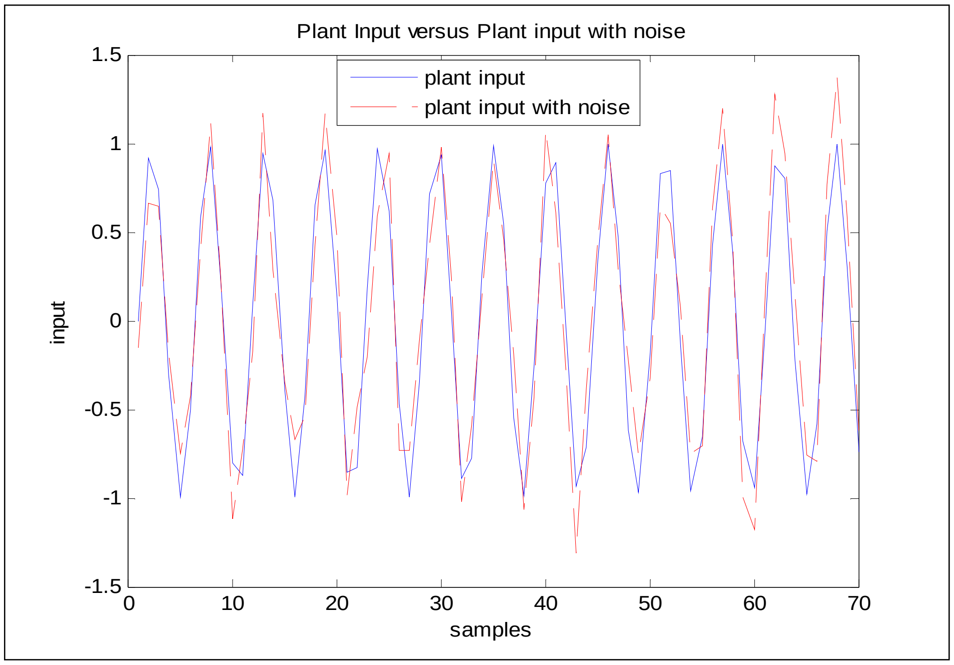

Figure 4.22 Plant input versus plant input with additive noise

Table 4.38 Noise performance results of plant I

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSO	0.0330	15.202	0.0333	14.796
Nonlinear ARX	0.177	7.5	0.177	7.5
Prediction Error Method	1.33	-1.23	1.473	-1.68
Subspace Method	1.022	-0.095	1.028	-0.122
Wiener Model	0.130	8.6	0.159	7.98
Hammerstein Model	2.78	-4.430	58.45	-17.667
Hammerstein-Wiener Model	0.34	4.58	2.032	-3.08
Nonlinear ARMAX	32.34	-15.1	33.198	-15.21
Elman Network	0.023	16.3	0.0281	15.50

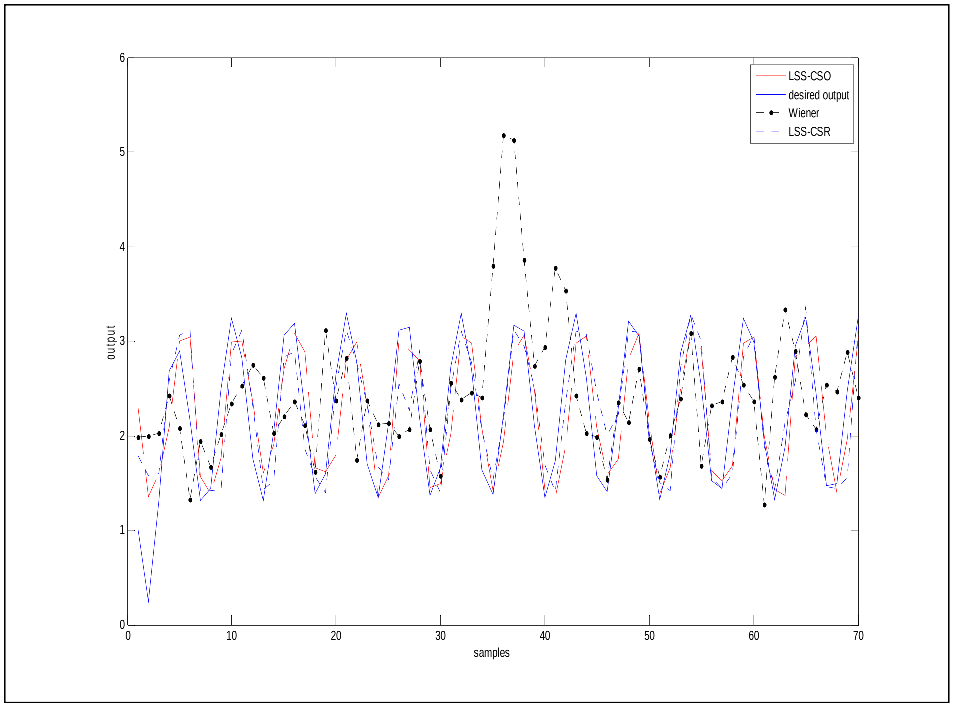

Figure 4.23 Comparison of outputs for different methods versus desired output of plant I with noise

4.2.6.2 Case 2: A first order nonlinear state model

In order to test the Proposed Method LSS-CSO, the dynamics of the benchmark plant in literature is borrowed (Narendra, 1996).

$$\begin{aligned} x(k+1) &= 10\sin(x(k)) + u(k)\left[0.1 + \cos(x(k)u(k))\right] \\ y(k) &= 0.025x(k) + 0.5 \end{aligned} \tag{4.14}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them is the estimates of states, 1 of them is the input which is *v*(k)), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 1 output neuron which is the state. The Output ANN has 3 input neurons which are the states and the plant input $v(k)$, has 3 hidden neurons and 1 output neuron which represents the system output.

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial condition $x_1(0)$ is fixed to a constant value 0.5.

Noise performance is tested with a uniformly distributed noise between [-0.25 0.25]. A uniformly distributed noise is added with the plant input, and the system performance is tested. Figure (4.24) shows the plant input with additive noise and without noise. Plant II input is chosen a chaotic input which is given in Equation (4.15).

$$v(m+1)=4v(m)(1-v(m)) \ , \ v(0)=0.1 \text{ (Chaotic input)} \qquad (4.15)$$

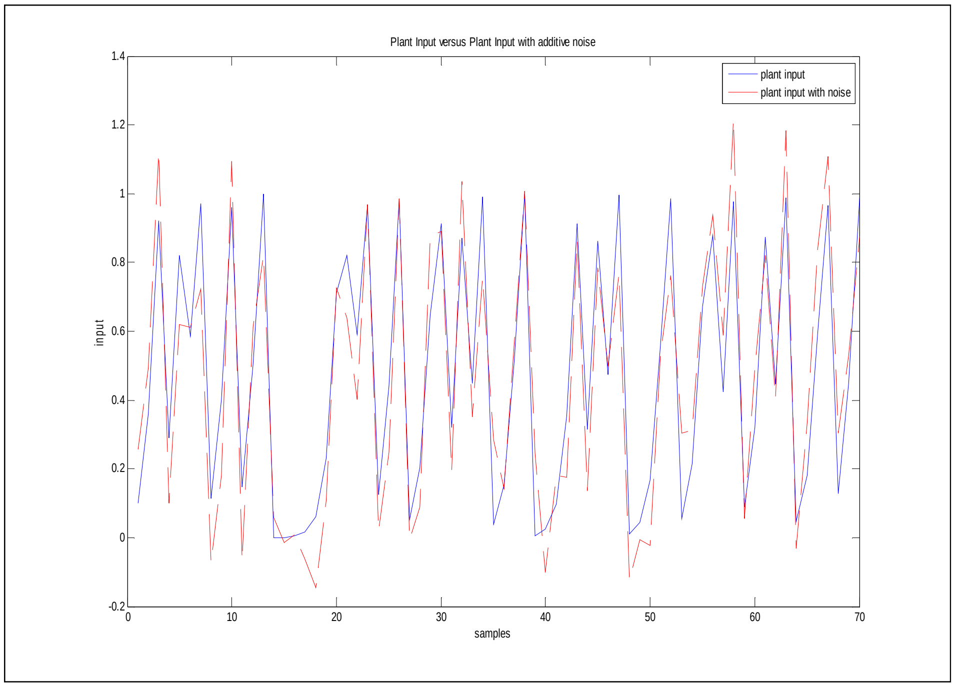

Figure 4.24 Plant input versus plant input with additive noise

Table 4.39 Noise performance results of plant II

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSO	0.086	10.609	0.094	10.298
Nonlinear ARX	Nonconvergent	Nonconvergent	Nonconvergent	Nonconvergent
Prediction Error Method	0.141	8.48	0.161	7.91
Subspace Method	0.180	7.44	0.196	7.070
Wiener Model	0.097	10.11	0.104	9.80
Hammerstein Model	0.69	1.58	33.022	-15.188
Hammerstein-Wiener Model	0.53	2.75	141.778	-21.51
Nonlinear ARMAX	Nonconvergent	Nonconvergent	Nonconvergent	Nonconvergent
Elman Network	0.1004	9.98	0.1043	9.81

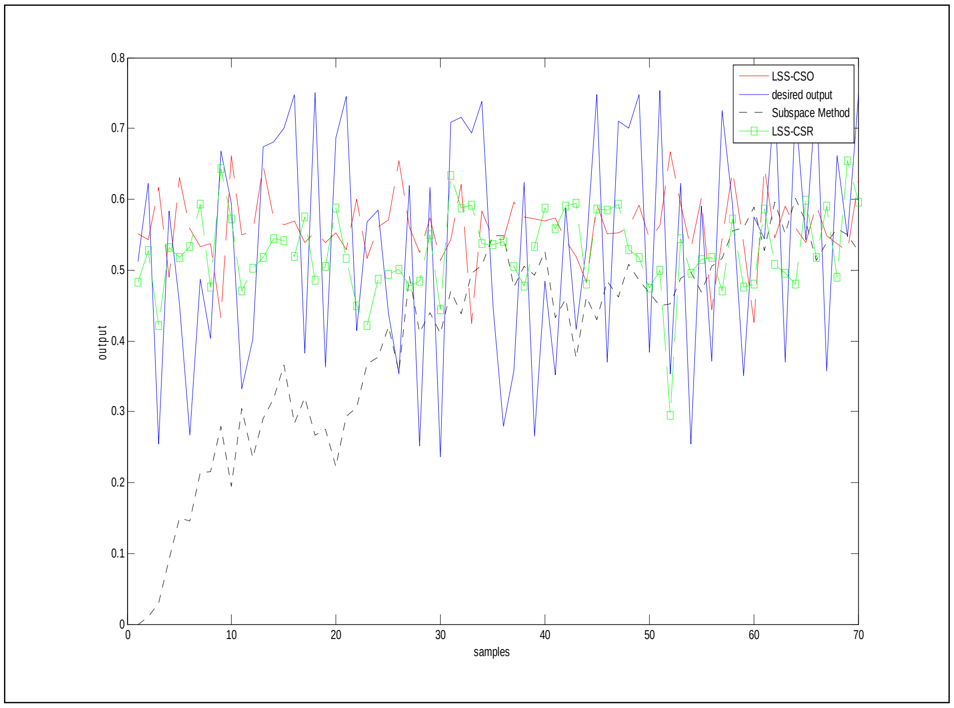

Figure 4.25 Comparison of outputs for different methods versus desired output of plant II with noise

4.2.6.3 Case 3: A third order nonlinear difference equation model

In order to test the Proposed Method LSS-CSO, the dynamics of the benchmark plant in literature is borrowed (Uykan & ,2000).

$$x(k+1) = \frac{2.6x(k)x(k-1)x(k-2)u(k-1)(x(k-2)-1)+u(k)}{1+x^2(k-1)+x^2(k-2)} \tag{4.16}$$
$$y(k) = x(k)+1$$

The State ANN, Observer ANN and the Output ANN is constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are the estimates of the state, 1 of them is the input which is u(k)), 3 hidden layer neurons and 2 output neurons which are used for learning the state. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neurons which is the state. The Output ANN has 3 input neurons which are the states and the plant input u(k), has 3 hidden neurons and 1 output neuron which represents the system output.

For all cases the method performance is tested with the mean square error (MSE) and, signal to error ratio (SER) parameters. The samples of data for inputs and outputs are 70. Training data samples are 48, and the other 22 of them are reserved for test. Normalized mean square error (nMSE) and signal to error ratio (SER) parameters are calculated for both training and test data sets. Mean square error is divided by the output signal power to calculate the normalized mean square error, in order to compare the values. The initial condition $x_1(0)$ is fixed to a constant value 0.5.

Noise performance is tested with a uniformly distributed noise between [-0.40 0.40]. A uniformly distributed noise is added with the plant input which is a sinusoidal input sin(k) where k is from 0 to 70 and the system performance is tested. Figure 4.26 shows the plant input with additive noise and without noise.

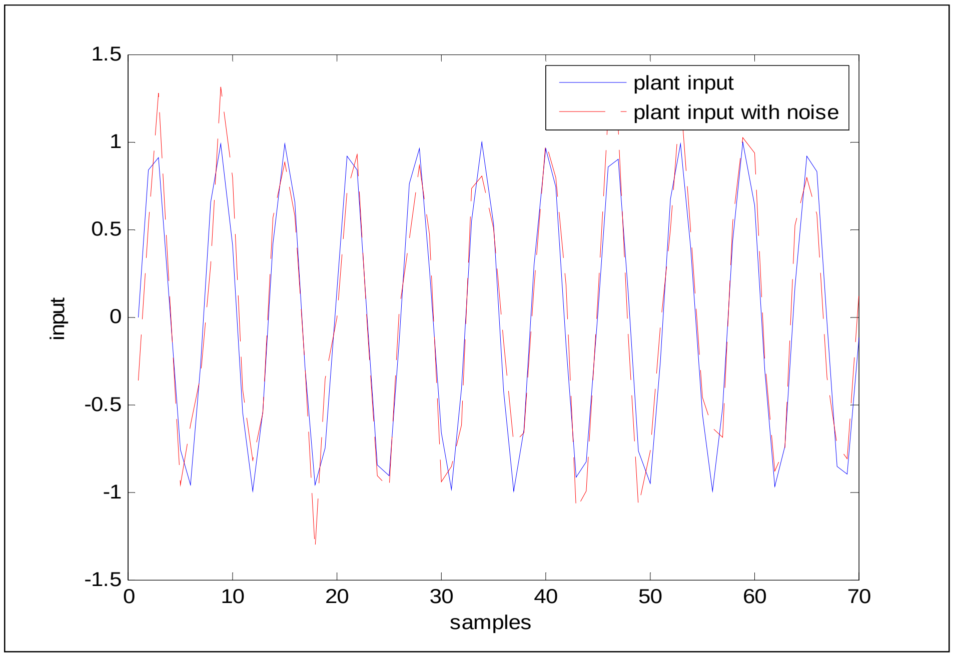

Figure 4.26 Plant input versus plant input with additive noise

Table 4.40 Noise performance results of plant III

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSO	0.220	8.213	0.196	7.102
Nonlinear ARX	0.37	4.30	0.336	4.73
Prediction Error Method	1.51	-1.81	2.214	-3.45
Subspace Method	1.181	-0.70	1.223	-0.87
Wiener Model	0.204	6.89	0.235	6.27
Hammerstein Model	0.951	0.217	2.77	-4.42
Hammerstein-Wiener Model	0.286	5.43	0.71	1.44
Nonlinear ARMAX	0.79	1.012	0.823	0.842
Elman Network	0.152	8.18	0.235	6.287

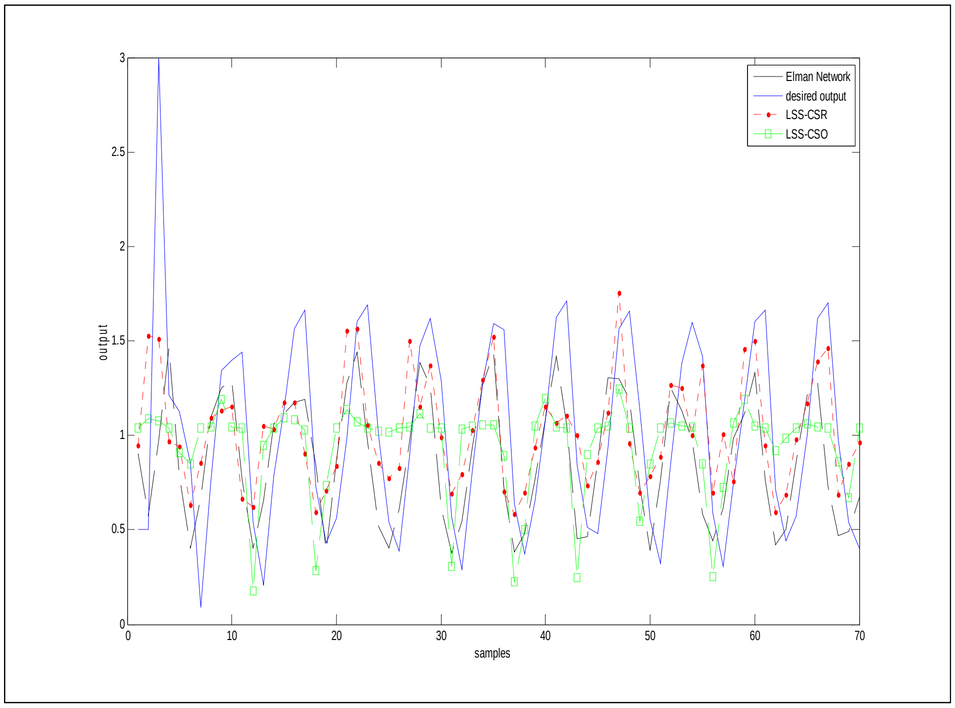

Figure 4.27 Comparison of outputs for different methods versus desired output of plant III with noise

4.2.6.4 Case 4: A chaotic system (Forced Duffing Oscillator)

In order to test the Proposed Method LSS-CSO under noise with a chaotic plant, the modified dynamics of the chaotic benchmark plant in the literature is borrowed which is shown in Equation (4.17). We can also write this equation with two state equations in Equation (4.18). This is chosen in order to test the performance of the proposed method LSS-CSO with a chaotic plant. In cases I, II, and III nonlinear and linear plants were tested with chaotic inputs. Forced Duffing oscillator were chosen as a chaotic plant because of its forcing input, since the proposed method needs input output data of the plant which is to be identified. Forced duffing oscillator states and the output are calculated under the initial conditions $x_1(0)=3$ and $x_2(0)=4$.

$$\ddot{x} = 7.6\cos(t) - 0.04\dot{x} - x^3 \tag{4.17}$$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= 7.6\cos(t) - 0.04x_2 - x_1^3 \\ y &= 0.05x_1 + 10 \end{aligned} \tag{4.18}$$

The State ANN, Observer ANN and the Output ANN are constucted with a nonlinear type of neural network which is a multilayer perceptron. For the State ANN there are three inputs (2 of them are the estimates of the states, 1 of them is the input which is *u*(k)), 3 hidden layer neurons and 2 output neuron which are used for learning the states. The Observer ANN has 2 inputs (output and the input), 3 hidden neurons and 2 output neurons which are the states. The Output ANN has 3 input neurons which are the states and the plant input *u*(k), has 3 hidden neurons and 1 output neuron which represents the system output. In order to train the Output ANN used modified lorenz system is given in Equation (4.19) (Lorenz, 1963; Cuomo & Oppenheim, 1993)

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1) \\ \dot{x}_2 &= rx_1 - x_2 - 20x_1x_3 \\ \dot{x}_3 &= 5x_1x_2 - bx_3 \end{aligned} \tag{4.19}$$

with the parameters σ, *r*, *b*, and initial conditions $x_1(0)$, $x_2(0)$, and $x_3(0)$ are 10, 56.6, 5.02, 1, 0, and 1, respectively.

In order to train the Output ANN used modified henon system is given in Equation (4.20) (Dick & Kandel, 2005).

$$y(k+1) = 1.4y(k)^2 + 0.3y(k-1) + 1 \tag{4.20}$$

The phase portrait of the henon system in Equation (4.20) is given under the initial conditions $y(-1)=0.5$ and $y(0)=0.5$.

Noise performance is tested with a uniformly distributed noise between [-2 2]. A uniformly distributed noise is added with the plant input which is a sinusoidal input sin(k) where k is from 0 to 70 and the system performance is tested. Figure (4.28) shows the plant input with additive noise and without noise.

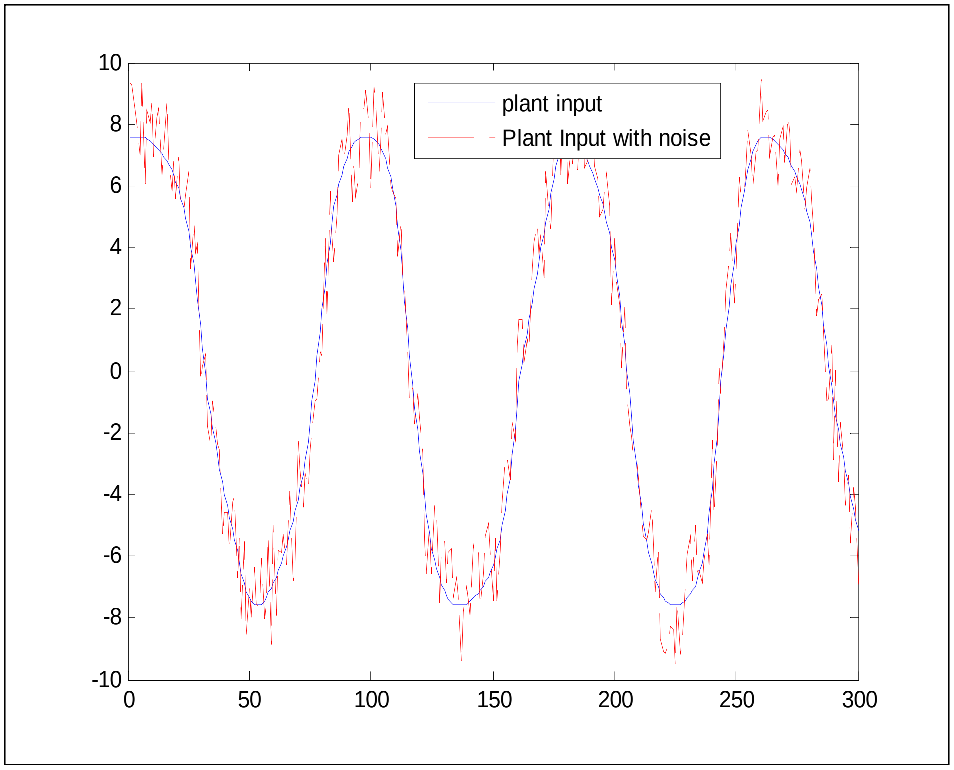

Figure 4.28 Plant input versus plant input with additive noise

Table 4.41 Noise performance results of plant IV

Method – Model	**nMSE**	**SER (dB)**	**nMSE with noise**	**SER (dB) with noise**
LSS-CSO	0.010	19.971	0.0111	19.532
Nonlinear ARX	0.0688	11.62	0.0688	11.62
Prediction Error Method	2.541	-4.05	2.781	-4.44
Subspace Method	1.013	-0.056	1.017	-0.072
Wiener Model	0.151	8.204	0.153	8.153
Hammerstein Model	0.167	7.75	0.5276	2.776
Hammerstein-Wiener Model	0.295	5.293	47.729	-16.78
Nonlinear ARMAX	0.069	11.56	0.07	11.456
Elman Network	0.0335	14.750	0.0356	14.48

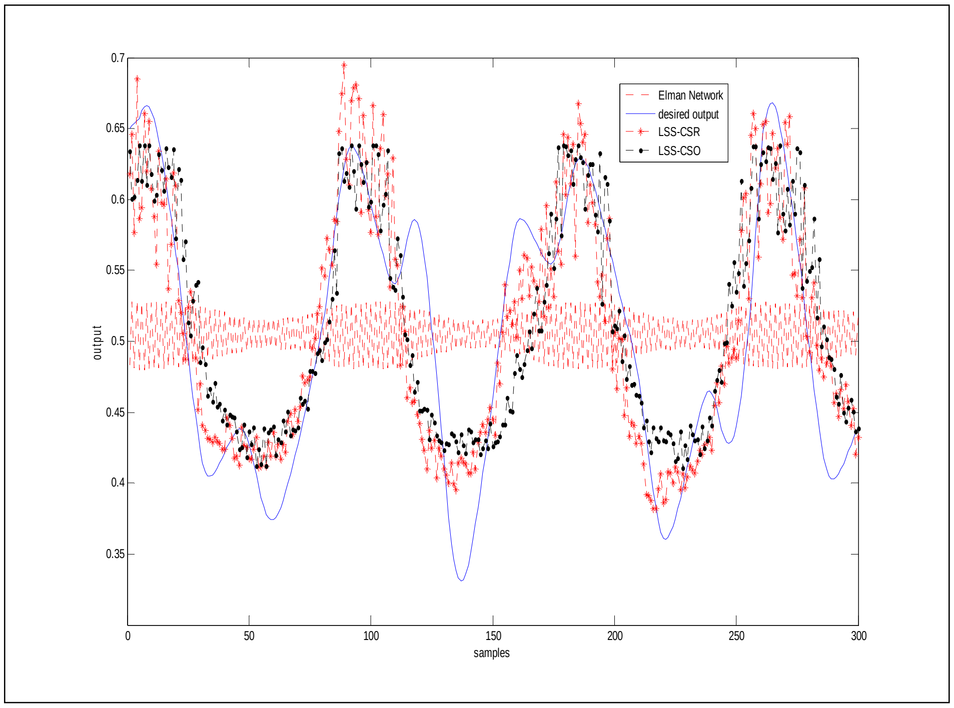

Figure 4.29 Comparison of outputs for different methods versus desired output of plant IV with noise

CHAPTER FIVE
ROBUST CHAOTIFICATION

Chapter 5 presents a method for robustification of a chaotic reference model based dynamical state feedback chaotification method which can be applied to any input-state linearizable (nonlinear) system including linear controllable types.

The developed robust chaotification method gives a broad chaotification area to the system which can change its parameters and get disturbance noise. A user friendly graphical user interface (GUI) is presented which is designed for all users to use this robust chaotification method. The GUI helps users to change the parameters; both controller parameters and plant parameters. Users can add additional noise to the plant and test whether the overall system is still in chaos. The maximum Lyapunov exponent and the phase portrait of the tested system is printed on the screen. If the tested system has a positive Lyapunov exponent (is in chaos) the bifurcation diagram is plotted in a cumulative way.

5.1 Lorenz Type System and Current Result

Considering the Lorenz system defined in Equation (5.1) (Lorenz, 1963; Cuomo & Oppenheim, 1993) as the reference chaotic system.

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= rx-y-20xz\\ \dot{z} &= 5xy-bz\end{aligned} \tag{5.1}$$

Then, redefine the above Lorenz states x, y and z as

$$\begin{aligned}x_n &\hat{=} x\\ x_{n+1} &\hat{=} y\\ x_{n+2} &\hat{=} z\end{aligned} \tag{5.2}$$

and choosing the control input u as

$$u = a_1 x_1 + \cdots + a_{n-1} x_{n-1} - (\sigma - a_n) x_n + \sigma x_{n+1} \tag{5.3}$$

The linear system given in Equation (5.1) is augmented to the following nonlinear system of *(n+2)* nd order.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \\ \dot{x}_{n+1} \\ \dot{x}_{n+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & -\sigma & \sigma & 0 \\ 0 & 0 & \cdots & r & -1 & -20x_n \\ 0 & 0 & \cdots & 0 & 5x_n & -b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \\ x_{n+1} \\ x_{n+2} \end{bmatrix} \tag{5.4}$$

With the above choice of the control input u, the last state equation of the system in Equation (5.1) becomes identical to the first Lorenz equation in Equation (5.2) under the change of variables in Equation (5.2).

The chaotification scheme is proposed by (Şahin & Güzeliş, 2010) is given in Figure 5.1. Where, a part of the Lorenz system constitutes a dynamical controller together with the linear state feedbacks.

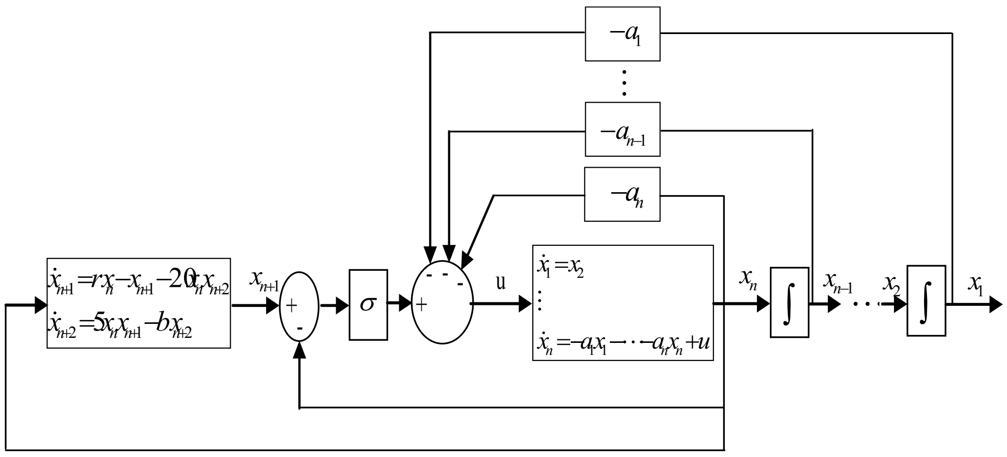
$-a_1$
$-a_{n-1}$
$-a_n$
$\dot{x}_{n+1}=rx_n-x_{n+1}-20x_nx_{n+2}$
$\dot{x}_{n+2}=5x_nx_{n+1}-bx_{n+2}$
x_{n+1}
σ
u
$\dot{x}_1=x_2$
$\dot{x}_n=-a_1x_1-\cdots-a_nx_n+u$
x_n
x_{n-1}
x_2
x_1

Figure 5.1 The Chaotification scheme (Şahin & Güzeliş, 2010)

5.2 Chaotified DC Motor System by Lorenz Chaotic System

An rf-310ta series DC motor which is easy to use and can be found easily with very low cost is used, so it is very easy to reproduce the experiments and observe how a robust chaotified DC motor runs. For simplicity, a first order simplified model is chosen for the permanent magnet DC motor: $\dot{x} = -a_m x + b_m u$ where x is the angular velocity of the motor, i.e. the rpm and u stands for the armature input voltage. The first order system parameters of the used rf-310ta series DC motor were identified as $a_m = 2$ and $b_m = 5600$ by measuring its response due to the step input. (Şahin, 2010)

$$\begin{aligned} \dot{x} &= -a_m x - (\sigma - a_m)x + \sigma y = -a_m x + b_m u \\ \dot{y} &= rx - y - 20xz \\ \dot{z} &= 5xy - bz \end{aligned} \tag{5.5}$$

the control input is chosen as

$$u = \frac{1}{b_m}\left[-(\sigma - a_m)x + \sigma y\right] \tag{5.6}$$

5.3 Robust Chaotification of the System

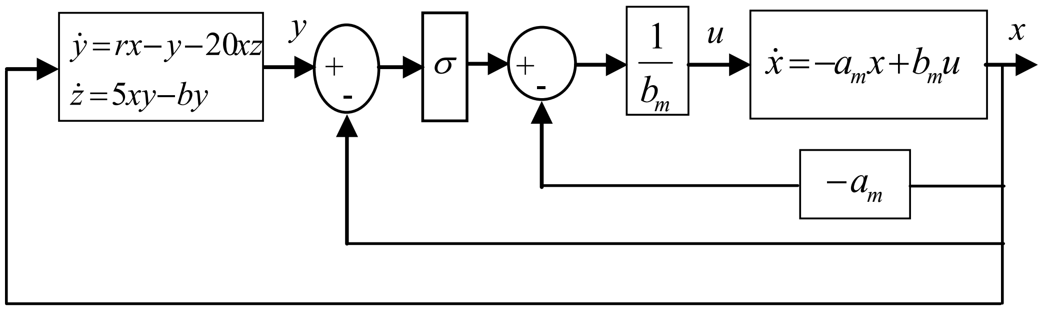

Figure 5.2 The chaotification of DC motor

In Figure 5.2 Lorenz chaotification process is obtained theoretically but indeed DC motor used is identified with a first order model and also the DC motor parameters will change during operation and will be effected from disturbances (e.g. noise, vibrations, thermal effects etc..). Since the control system is designed model dependent, change of plant (DC motor) parameters will change the system behavior. So the cascade block $\frac{1}{b_m}$ and the feedback block $-a_m$ will not be able to work well in a wide range, during the change of plant parameters.

Here, one may call this cascade block and feedback block as controller parameters and they should be chosen well for the chaotification of the overall system in a wider range so the chaotification scheme will run in a wider range of plant parameters. To achieve this we fixed the plant parameters in the model and changed the controller parameters. In every step the system is tested in chaos or not if it is in chaos it is marked. Chaos is tested with Lyapunov exponents with the program of Vasiliy Govorukhin (2004) given in Mathworks in Matlab Central. The algorithm employed in this m-file for determining Lyapunov exponents was proposed in (Wolf, Swift, Swinney, & Vastano, 1985). Then in Figure 5.3 the points, a_m =3.455 and b_m = 3300 are chosen as controller parameters which are the center of gravity of chaos points. After determining the controller parameters the system is operated again for searching a wider chaotification area.

The chaotification area is increased with the correct choose of parameters and it is shown in Figure 5.4.

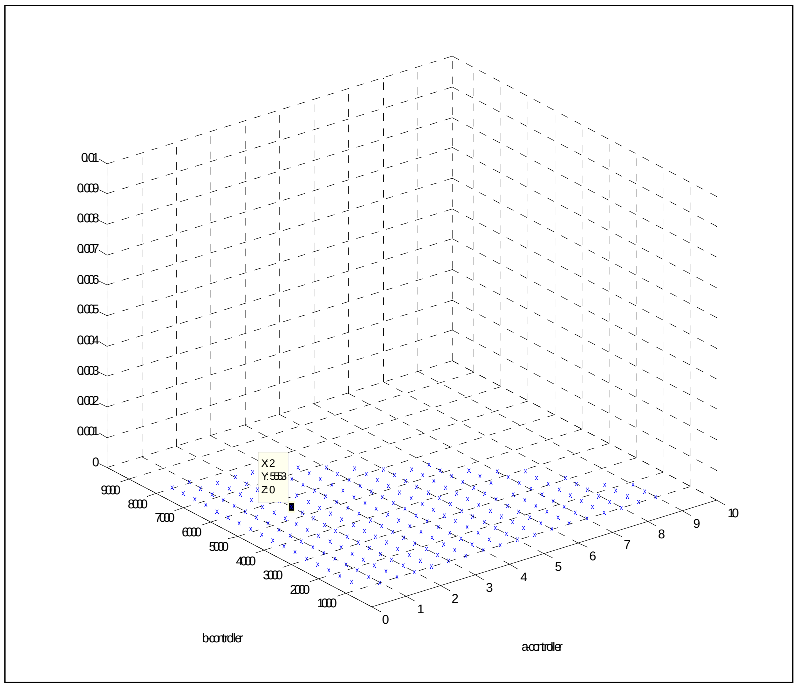

Figure 5.3 Bifurcation diagram while changing controller parameters

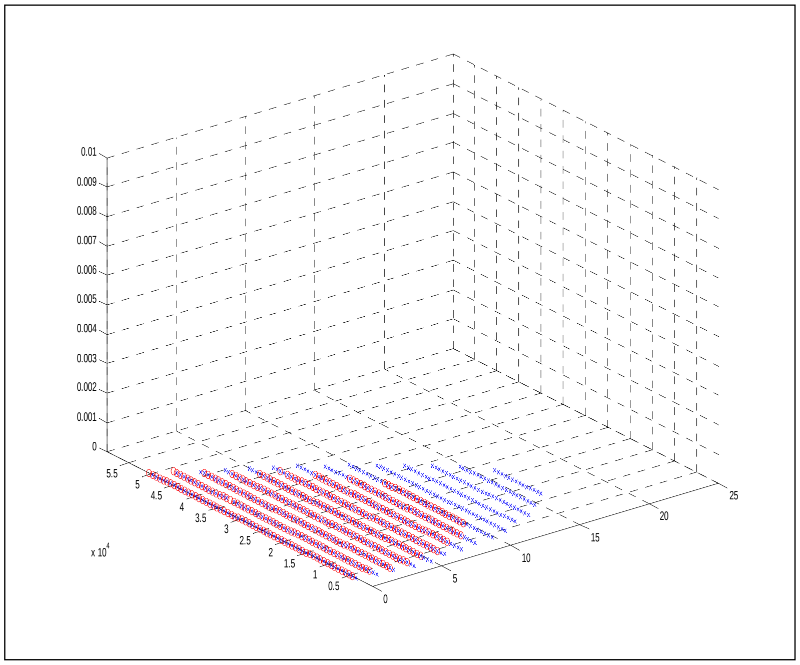

Figure 5.4 Comparison of Bifurcation diagram with different controller parameters

Figure 5.5 shows the bifurcation diagram of the system for region of interest where the plant parameters can vary, the red part is obtained when ac=2 and bc=5600, the blue part is for ac=3.455, bc=3300 which are the parameters found for robust chaotification.

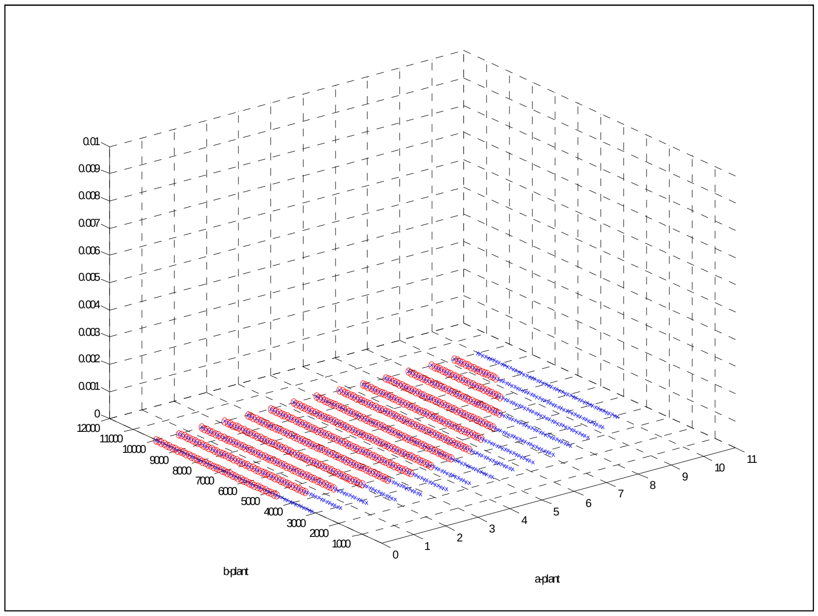

Figure 5.5 Comparison diagram of Bifurcation diagram in the region of interest

The parameter robust chaotification scheme is also tested under noise to test the system from outer disturbances. A signal which has a constant mean and constant variance (both should be changed by user on GUI) is added to the plant as a noise to test the system under disturbance. The chaotification system and noise added parameter robust chaotification system is compared about chaotification ability and result is given in Figure 5.6.

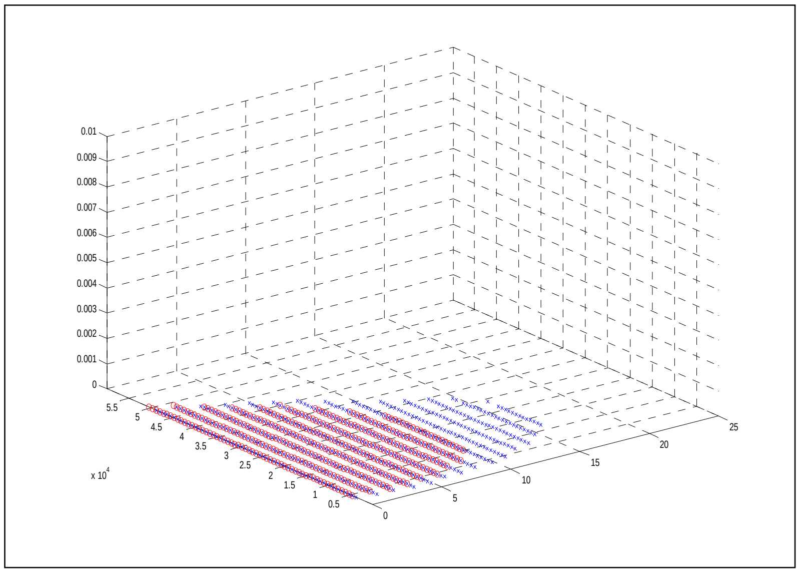

Figure 5.6 Comparison diagram of Bifurcation diagram with additive noise

5.4 Graphical User Interface (GUI) and User Dependent Parameters

In this section a graphical user interface (GUI) is presented which is designed for all users to use this robust chaotification method user friendly. The GUI helps users to change the parameters; both controller parameters and plant parameters. Users can add additional noise to the plant and test whether the overall system is still in chaos. The maximum Lyapunov exponent and the phase portrait of the tested system is printed on the screen. If the tested system has a positive Lyapunov exponent (is in chaos) the bifurcation diagram is plotted in a cumulative way.

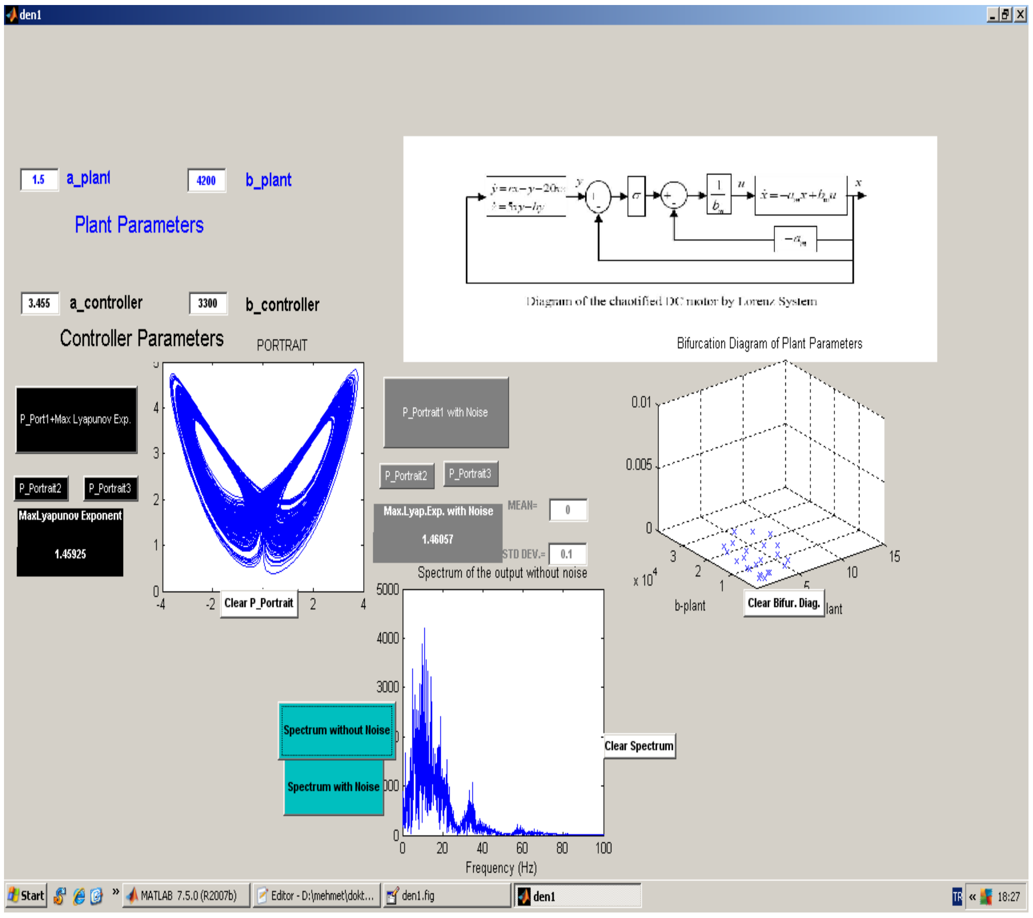

Figure 5.7 Graphical user interface of the robust chaotification method

Also the users can observe the spectrum of the overall system for the specified plant and controller parameters given by the user. The plant parameters vary during the operation and also the spectrum for the output of the system changes with the different choice of controller parameters. The proposed robust chaotification method provides a broad and powerful spectrum. The spectrum difference is shown in Figure 5.8 for the fixed plant parameters which are ap= 2.5 and bp =4500 with two different choice of controller parameters one of them is ac=2, bc=5600 and the other one which the proposed method gives us ac=3.455, bc=3300.

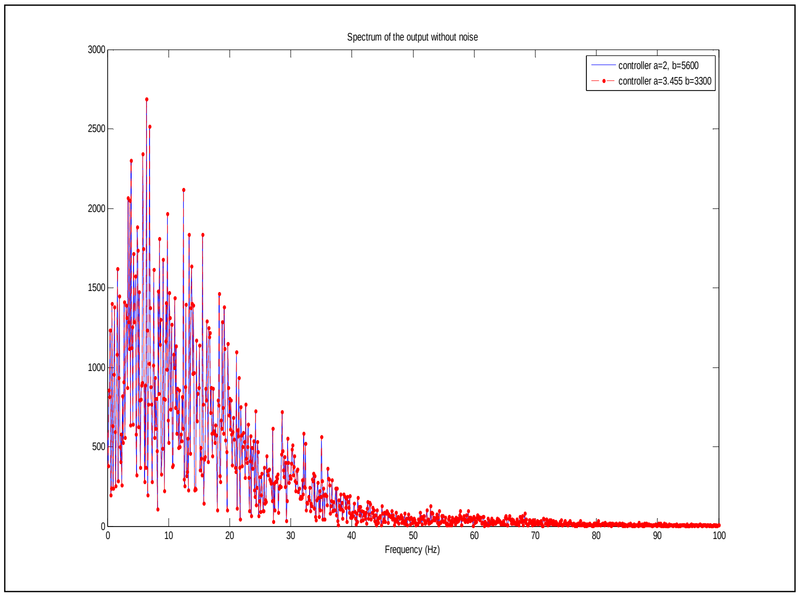

Figure 5.8 Spectrum with plant parameters ap= 2.5 and bp =4500

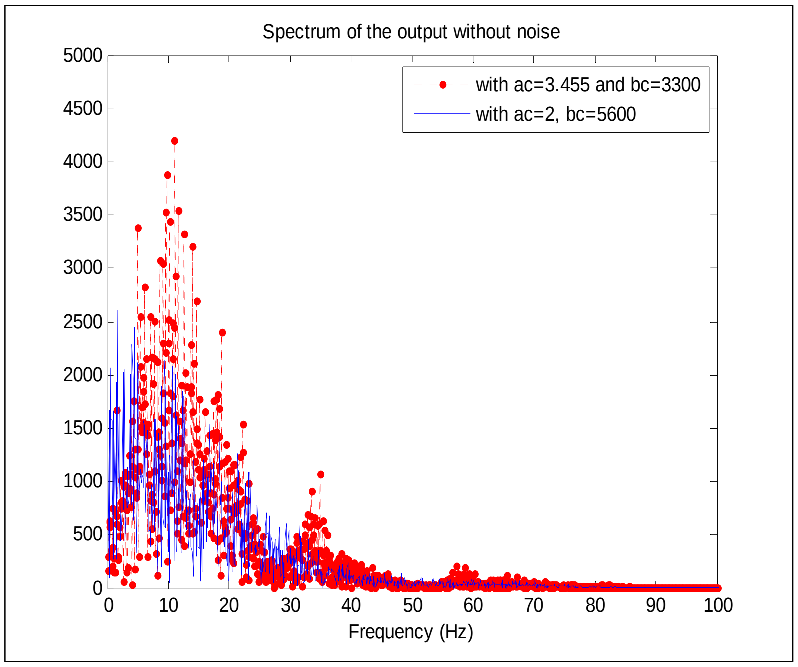

Figure 5.9 Spectrum with plant parameters ap=1.5 and bp=4200

The spectrum of the proposed robust chaotification method is also tested with an additive noise to the plant input an additive noise having a mean of 0 and standard deviation of 0.1 is added to the plant input and spectrum of the output is compared with the model which is designed with the controller parameters ac=2 and bc=5600. Figure 5.9 and Figure 5.10 show the spectrum difference between different choice of controller parameters, when the plant parameters are set to ap=1.5 and bp=4200 which are different from the nominal operating point ap=2 and bp =5600.

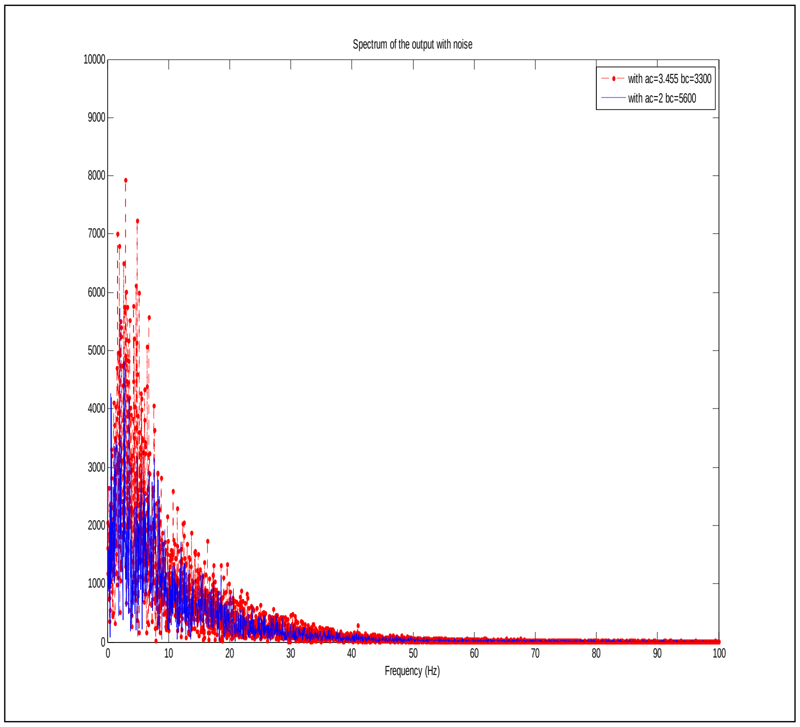

Figure 5.10 Spectrum with plant parameters ap=1.5 and bp=4200 with additive noise

CHAPTER SIX
CONCLUSION

Two new system identification methods, constructing the nonlinear state space plant models with employing artificial neural networks based on input output data and a robust chaotification method are proposed in this thesis.

(LSS-CSR) method is a system identification method based on input output data. For the identification of a system (linear, nonlinear or chaotic) only needed data is the input and output of the system which is to be identified. The developed LSS-CSR method is designed to learn the nonlinear state space equation of the system with two artificial neural networks, one is for constructing the state equation the other is for constructing the output equation. The method uses white gaussian noise states for linear and nonlinear plants independent of the order. But if the plant to be identified is a chaotic system, then using chaotic states of well-known chaotic plants as candidate states is preferred to raise the performance.

(LSS-CSO) method is also a system identification method based on input output data. For the identification of a system (linear, nonlinear or chaotic) only needed data is the input and output of the system which is to be identified. The developed LSS-CSO method is designed to learn the nonlinear state space equation of the system with three artificial neural networks, two of them are working as an auto association network which are output and observer networks. They both together try to observe the states. The third network is the state network which is used to construct the nonlinear state equation of the method. The method uses white gaussian noise states for linear and nonlinear plants independent of the order. But if the plant to be identified is a chaotic system then using chaotic states of well known chaotic plants as candidate states gives better performance.

Performance of both (LSS-CSR and LSS-CSO) methods are tested with four benchmark plants; one is a second order linear state model, one is a nonautonomous chaotic plant, the other two are nonlinear plants; first order nonlinear state model and a third order nonlinear difference equation model. Except the chaotic one the other linear and nonlinear plants are driven with step, chaotic and sinusoidal inputs with two different frequencies. Nonautonomous chaotic one is driven with its forcing input. The performance of the methods are tested in training and test phases with the performance parameters signal to error ratio (SER) and normalized mean square error (nmse) under four benchmark plants given above. Also the performance of both methods are tested with the noise coming to the input of the plant. With the all possible initial states of the linear and nonlinear plants, LSS-CSR and LSS-CSO methods are trained and tested with a random initial state, then performances of both methods are given in the thesis. The performance results of both methods are compared with eachother and other well known models, which are Hammerstein, Wiener, Hammerstein-Wiener, Nonlinear ARX, Nonlinear ARMAX, State space Model with Prediction Error Method, State Space Model with Subspace Method and Elman Network. The architectures of the well known models are specified with having higher order nonlinearities for instance Hammerstein and Wiener models having 10 piecewise-linear nonlinearities in the input and output part. Hammerstein-Wiener model having a total of 20 nonlinearities, and also Nonlinear ARX and Nonlinear ARMAX models having 10 sigmoid nonlinearities.

Both proposed methods can identify the benchmark plants in some cases clearly, in some cases generally but better than the other well known methods. The performance of the both proposed methods with the second order linear plant generally gives better training and test performances (LSS-CSR better than LSS-CSO with 1-2dB difference) when compared with other well known methods. Only the Elman Network gave competitively near results with LSS-CSR method, the other models provide poor performances as an average with four inputs. Two methods also provided better results in training and test phases again with 1-2 dB better in LSS-CSR method with two nonlinear plants which are first order nonlinear state model and a third order nonlinear difference equation model. With a first order nonlinear plant Nonlinear ARX and Nonlinear ARMAX models could not converge the plant, Subspace and Prediction Error methods provide very poor performances for sinusoidal inputs. The other well known models Hammerstein-Wiener family and

Elman Network gave lower performances in the average of all applied inputs. To the third order nonlinear difference equation both methods again were the best in comparison. Among other methods Elman was the first but can not reach LSS-CSR especially in sin20k input case. The others gave very poor performances, Nonlinear ARX and Nonlinear ARMAX could not converge to some input cases. Identification of the nonautonomous chaotic plant exhibits the performance of the proposed methods clearly where the other models could not settle the output of the plant. Since the plant to be identified is chaotic using chaotic states as candidate states boost the performance of the proposed methods. Adding noise to the inputs of the nonlinear, linear and chaotic plants did not effect the performance of the proposed methods, only 1 or less dB decreases shown, but with other methods in some cases drastic decreases in other well known methods is shown.

Linear and nonlinear plants ran with all possible states and input output pairs for different initial states are taken. Both methods are used to identify the plants with different initial states and tested with a random initial state that is not placed in the training set. LSS-CSR method again gave better performance than the LSS-CSO method but both methods can identify the systems generally.

The third contribution of the thesis which is a robust chaotification method is tested with a first order model of a DC motor. The proposed method shows the chaotic area of the system while changing the plant parameters. The method specifies the correct controller parameters to enlarge the chaotic area which is the bifurcation diagram of the system. With the correct choice of the controller parameters the system has a wider chaos area and the proposed model is also robust to the input noise. In the study a user friendly graphical user interface is also developed for the robust chaotification method. The user can change the controller and plant parameters of the system very easily, can add noise to the input of the system with specifiying the mean and standard deviation of the additive noise. The graphical user interface shows the maximum Lyapunov exponent of the system with and without noise, draw phase portraits of the system according to the specified inputs and specified noise parameters, can see the the bifurcation diagram of the system in a cumulative way where the diagram marks the operating point if the Lyapunuov exponent of that point is a positive number. Users can also see the spectrum of the overall system for the operating points specified by the user.

REFERENCES

Akaike, H. (1976). *Canonical correlation in R. Mehra and D. Lainiotis, editors Advanced in system identification*. New York: Academic Press.

Alduwaish, H. & Karim, M.N. (1997). A New Method For The Identification Of Hammerstein Model. *Automatica,* 1871-1875.

Alduwaish, H., Karim, M. N., & Chandrasekar. V. (1997). Hammerstein Model Identification by Multilayer Feedforward Neural Networks. *International Journal of Systems Science*, 49-54.

Anderson, T.W. (1971). *The statistical Analysis of Time Series*. New York, USA: Wiley.

Anderson, B.D.O. & Gevers, M. (1982). Identifiability of linear stochastic systems operating under linear feedback, *Automatica, 18* (2), 195–213.

Aström, K. J., & Bohin, T. (1965). Numerical identification of linear dynamic systems from normal operating records. *Proceeding IFAC Symposium Self. Adaptive Systems, Teddington, U.K.,* 96-111.

Aström, K. J., & Murray, R. H. (2008). *Feedback Systems: An introduction for Scientists and Engineers*. Princeton: Princeton University Press.

Bhattacharya, S. K. (2011). *Linear Control Systems* (1st ed.). Pearson.

Bode, H. W. (1940). Relations between attenuation and phase in feedback Amplifier Design. *Bell System Technical Journal*, *19*, 421-454.

Bode, H. W. (1945). *Network Analysis and Feedback Amplifier Design.* Newyork: D. van Nostrand.

Bolton, W. (2004). *Instrumentation and Control Systems*. Oxford U.K.:Elseiver Publishing Co.

Box, G.E. P., & Jenkins, G. M. (1970). *Time series Anaylsis*. San Francisco USA: Forecasting and Control, Holden Day.

Chen, B. S. E. & Chang, Y. T. (2008). A review of system identification in control engineering, signal processing, communication and systems biology. *Journal of Biomechatronics Engineering, 1* (1), 87-109.

Chen, G. (1997). Control and Anti-Control of Chaos. *1997 First International Conference, Control of Oscillations and Chaos Proceedings.*

Chen, G. (1998). Chaos: Control and Anti-Control. *IEEE Circuits and Systems ISSN, 9,* 1-5.

Chen, G. (2001). What does chaos have to do with systems and control engineering?. *J. Syst. Sci. Complex, 14,* (32-39).

Chen, G. (2003). *Chaotification via Feedback: The Discrete Case. In Chaos Control: Theory and Applications*. Berlin, Germany :Springer.

Chen, G., Chen, Y., & Öğmen, H. (1997). Identifying chaotic systems via a Wiener-type cascade model, *IEEE Control Systems Magazine*, *8*, 29-36.

Chen, G. & Dong, X. (1998). F*rom Chaos to Order: Methodologies, Perspectives and Applications*. Singapore:World Scientific.

Chen, G & Shi, Y. (2006). Introduction to Anti-Control of Discrete Chaos: Theory and Applications. *Philosophical Transactions of the Royal SocietyA, 364,* (2433-2447).

Chon, K. H. & Cohen R. J. (1997). Linear and Nonlinear ARMA Model Paameter Estimation Using an Artificial Neural Network. *IEEE Transactions on Biomedical Engineering, 44* (3), 168-174.

Chou, D.S. (2006). *Efficacy of Hammerstein Models in Capturing the Dynamics of Isometric Muscle Stimulated at Various Frequencies*. MSc Thesis MIT.

Chu, S.R., Shoureshi, R., & Tenorio, M. (1990). Neural Networks for system identification. *IEEE Control System Magazine, 10* (3), 31-35.

Chui, N. & Maciejowski, J. (1996). Realization of stable models with subspace methods. *Automatica*, *32* (11), 1587–1595.

Clark, J.M.C. (1976). The consistent selection of parameterizations in system identification. *Proc. Joint Automatic Control Conf., Purdue Univ. West Lafayette,* 576-580.

Cuomo, K. M. & Oppenheim, A. V. (1993). Circuit Implementation of Synchronized Chaos with Applications to Communications. *Physics Review Letters*, *71* (1), 65-68.

Daniel, G. & William, K. C. (1975). Functional Seperation of EMG Signals via ARMA Identification Methods for Prosthesis Control Purposes. *IEEE Transactions on Systems, Man and Cybernetics*, *5* (2), 252-259.

Defigueiredo, R.J.P. & Chen, G. (1993). *Nonlinear Feedback Control Systems: An Operator Theory Approach*. Academic Press.

Deistler, M. (1994). *System identification*. Vienna Austria: IIASA's 20th Anniversary, Novographic.

Dick, S. & Kandel, A. (2005). Computational Intelligence in Software Quality Assurance . *Series in Machine Perception and Artificial Intelligence) World Scientific Publishing Co Pte Ltd (1 Jun 2005).*

Distefano, J. J., Stubberud, A. R., & Williams, I. (1990). *Schaum's Outline of Theory and Systems* (2nd ed.). McGraw Hill.

Ditto, W. L., Spano, M. L., In, V., Neff, J., Meadows, B., Langberg, J. J., Bolmann. A., & McTeague, K. (2000). Control of Human Atrial Fibrillation. *Int. J. Bif. Chaos, 10*, 593-602.

Doebelin, E. O. (1985). *Control System Principles and Design* (1st ed.). John Wiley & Sons. Inc.

Dorf, R. C. (1992). *Modern Control Systems* (6th ed.). Addison Wesley.

Eskinat, E., Johnson, S. H., & Luyben, W. L. (1991). Use of Hammerstein Models in Identification of Nonlinear Systems. *Ameri*call *Institute of Chemical Enqineerinq Journal, 37*, 255-268.

Fekih, A., Xu, H., & Chowdhury, F. N. (2007). Neural Networks based system Identification Techniques for model based fault detection of nonlinear systems. *International Journal of Innovative Computing, Information and Control, 3* (5), 1073-1085.

Gevers, M. (2006). A personal view of the development of system identification. *IEEE Control system Magazine, 26* (6), 93-105.

Hannan, E. J. (1970). *Multiple Time series.* New York USA: Wiley.

Hannan, E. J. (1960). *Time series analysis*. New York: Methuen.

Hatanaka, T. & Uosaki, K. (2001). Hammerstein Model Identification Method Based on Genetic Programming. *IEEE Evolutionary Computation, Proceedings of the 2001 Congress,* 1430 – 1435.

Haykin, S. (1999). *Neural Networks a Comprehensive Foundation*. Prentice Hall Inc.

Hazewinkel, M. & Kalman, R. E. (1976). *On variants, canonical Forms and Moduşi for Linear Constant, Finite- Dimensional Dynamical Systems Lecture Notes in Economicss and Mathematical System, 131*, Berlin, Germany: Springer-verlag.

Hellerstein, J. L., Diao, Y., Parekh, S., & Tilbury, D. M. (2004). *Feedback Control of Computing Systems*. Wiley Interscience IEEE Press.

Hernandez, E. & Arkun, Y. (1993). Control of Nonlinear Systems Using Polynomial ARMA Models. *AIChE Journal, 39* (3)*,* 446-460.

Ho, B. L., & Kalman, R.E. (1966). Effective construction of linear state-variable models form input/output functions. *Regelungstecknik, 14* (12), 545-548.

Hu, W. (2011). *Electronics and Signal Processing*. Springer.

Imer, O. C. & Basar, T. (1999). Optimal Solution to a Team Problem with Information Delays: An Application in Flow Control for Communication Networks. *Proceedings of the 38th IEEE Conference on Decision and Control,* 3, 2697-2702.

Jakimoski, G. & Kocarev, L. (2001). Chaos and Cryptography: Block Encryption Ciphers Based on Chaotic Maps. *IEEE Trans. Circ. Syst-I, 48*, 163-169.

James, H. M., Nichols, N.B., & Phillips, R.S. (1947). *Theory of Servomechanisms*, M.I.T. Radiation Lab. Series 25. New York: McGraw-Hill.

Jovanovic, O. (1997). Identification of dynamic system using neural network. *The scientific journal Facta Universitatis Architecture and Civil Engineering, 1* (4), 525-532.

Juang, J. N., Phan, M., Horta, L.G., & Longman, R.W. (1993). Identification of observer Kalman filter Markov parameters—Theory and experiments. *J. Guidance, Contr. Dynam.*, *16* (2), 320–329.

Kapur, J. N. (1988). *Mathematical Modeling*. New Age International Press.

Kellert, S. H. (1993). *In Wake of Chaos: Unpredictable Order in Dynamical Systems*. Chicago: University of Chicago Press.

Kim, I. H., Fok, S., Frege, K., Lee, D., Oh, T., & Wang, D. W. L. (2004). Neural Network-Based system identification and controller synthesis for industrial sewing machine. *International Journal of Control, Automation, and systems. 2* (1), 83-91.

Kocarev, L., Maggio, G. M., Ogorzalek, M., Pecora, L., & Yao, K. (2001). Special Issue on Applications of Chaos in Modern Communication Systems. *IEEE Trans. Circ. Syst-I, 48*, 1385-1527.

Koopmans, T.C., Rubin, H., & Leipnik, R. B. (1950). *Measuring the equation systems of dynamic economics*. New York: Wiley.

Kumar, A. A. (2007). *Control Systems*. New Delhi: PHI Learning Private Limited.

Kung, M. C. & Womack, B. F. (1984). Discrete Time Adaptive Control of Linear Dynamical Systems with a Two-Segment Piecewise-Linear Symmetric Nonlinearity. *IEEE Transactions on Automatic Control.*

Kuo, B. (1995). *Automatic Control Systems* (7th ed.). Prentice Hall, Englewood Cliffs, N. J.

Larimore, W.E. (1990). Canonical variate analysis in identification, filtering, and adaptive control. *Proc.of 29th IEEE Conf. Decision and Control*, *Honolulu,* 596–604.

Larsson, E. K., & Mossberg, M. (2003). On possibilities for estimating continuoud-time ARMA parameters. *13th IFAC Symposium on System Identification, Rotterdam, Netherlands,* 621-626.

Li, K. & Tongwen, C. (1993). System identification using a new dynamic network. *Proceedings of 1993 International Joint Confrerence on Neural Networks,* 2363-2366.

Ljung, L. (1987). *System Identification: Theory for the User*. New Jersey: PTR Prentice Hall, Englewood Cliffs.

Ljung, L. (1978). Convergence analysis of parametric identification methods. *IEEE Trans. Automat. Contr., 23*, 770-783.

Ljung, L. (2002). Prediction Error Estimation Methods, *Circuits Systems Signal Processing*, *21* (1), 11-21.

Ljung, L. & Sjöberg, J. (1992). A system identification perspective on neural nets. *IEEE Neural Networks for signal processing*, 423-435.

Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of Atmospherics Science*, *20* (1), 130-141 .

Lü, J., Zhou, T. S., Chen, G., & Yang, X. S, (2002). Generating chaos with a switching piecewiselinear controller. *Chaos*, *12* (2), 344-349.

Maqusi, M. (1985). Performance of Baseband Digital Transmission in Nonlinear Channel with Memory. *IEEE Transactions* on *Communications, 33*, 715-718.

Moran, P. R. U., Agamennoni, O. E., & Figueroa, J. L. (2005). *On the identification of Wiener Model.* Enpromer.

Nagrath, L. J., & Gopal, M. (2005). *Control System Engineering* (4th ed.). New Age International Publisher.

Narendra, K. S. (1996). Neural Networks for Control Theory and Practice *Proceedings of the IEEE*, *84* (10).

Nelles, O. (2001a). *Nonlinear System Identification.* Springer Verlag.

Nelles, O. (2001b). *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* New York: Springer.

Nikias, C. L. & Mendel, J. M. (1993). Signal Processing with Higher Order Spectra. *IEEE Signal Processing Magazine, 10* (3).

Nyquist, H. (1932). Regeneration Theory. *Bell System Technical Journal, 11*, 126-147.

Ogata, K. (1994). *Discrete Time Control Systems*. Prentice Hall.

Ogata, K. (2009). *Modern Control Engineering* (5th ed.). Prentice Hall.

Ogunfunmi, T. (2007). *Adaptive Nonlinear System Identification: The Volterra and Wiener Model Approaches.* Springer.

Overschee, P. & Van and De Moor, B. (1994). N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, *30* (1), 75–93.

Ottino, J. M. (1989). *The Kinematics of Mixing : Streching Chaos, and Transport*. New York: Cambridge University Press.

Overschee P. V, Moor B. D. (1996). *Subspace Identification for Linear Systems Theory- Implementation- Applications*. Kluwer Academic Publishers.

Paoletti, S. (2003). Identification of Piecewise Affine Models. *PhD Thesis Siena.*

Perrott, M. H. & Cohen, R. J. (1996). An Efficient Approach to ARMA Modeling of Biological Systems with Multiple Inputs and Delays. *IEEE Transactions on Biomedical Engineering* ,*43* (1).

Pham, D.T. & Liu, X. (1991). Neural Networks for Discrete Dynamic System Identification. *J. of Systems Engineering*, *1* (1), 51-60.

Phillips, C. L. & Harbor, R. R. (2000). *Feedback control systems* (4th ed.).Prentice Hall.

Schiff, S. J., Jerger, K., Duong, D. H., Chang, T., Spano, M. L., & Ditto, W. L. (1994). Controlling Chaos in the Brain. *Nature, 370*, 615-620.

Schutter, B. D. (2000). Minimal state-space realization in linear system theory: An overview. *Journal of Computational and Applied Mathematics, Special Issue on Numerical Analysis in the 20th Century- Vol I: Approximation Theory, 121* (1-2) 331-354.

Seyad, R. K. A., Cao, Y. (2008). Nonlinear System Identification for Predictive Control using Continuous time Recurrent Neural Networks and Automatic Differentiation, *Journal of Process Control, 18* (6), 568-581.

Sjoberg, J., Zhang, Q., Ljung, L., Benveniste, A., Deylon, B., Glorennec, P.Y., Hjalmarsson, H., & Juditsky, A. (1995). Nonlinear Black Box Modeling in System Identification: a Unified Overview. *Automatica, 31*, 1691- 1724.

Sin, K. S. & Goodwin, G. C. (1980). Checkable conditions for identifiability of linear systems operating in closed loop. *IEEE Trans. Automat. Contr.*, *25*, 722–729.

Söderström, T., Ljung, L., & Gustavsson, I. (1976). Identifiability conditions for linear multivariable systems operating under feedback. *IEEE Trans. Automat. Contr., 21*, 837-840.

Sugimoto, S., Matsumoto, M., & Kabe, M. (1992). System Identification using neural networks. *IEEE International Conference on Computing & Processing*, 79-82.

Şahin, S. (2010). *Learning Algorithms for Nonlinear Controller Design*. Dokuz Eylül University Natural and Applied Sciences, PhD Thesis, İzmir

Şahin, S. & Güzeliş, C. (2010). "Chaotification" of Real Systems by Dynamic State Feedback". *IEEE Antennas and Propagation Magazine*, *52* (6), 222-233.

Ugwa, K. A. (2012). Mathematical Modeling as a Tool for Sustainable Development in Nigeria. *International Journal of Academic Research in Progressive Education and Development, 1* (2), 251-258.

Uykan, Z., Güzeliş, C., Çelebi, M. E., & Koivo, H. N. (2000). Analysis of Input-Output Clustrering for Determining Centers of RBFN. *IEEE Transaction on Neural Networks, 11* (14), 851-858.

Verhaegen, M. (1994). Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica*, *30* (1), 61–74.

Viberg, M. (1995). Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, *31* (12), 1835–1851.

Wang, L. & Garnier, H. (2012). *System identification, environmental modeling and control system design*. Springer.

Wang, X. F. & Chen, G. (2000). Chaotifiying a stable LTl system by tiny ftdhack control. *IEEE Trans. Circ. Syst- 1, 47,* 410-415.

Wang, X. F., Chen, G., & Man, K. F. (2001). Making a continuous-time minimum-phase system chaotic by using time-delay feedback. *IEEE: Trans. Circ. Syst.-I, 48*, 641-645

Wies, R. W., Pierre, J. W., & Trudnowski, D. J. (2003). Use of ARMA Block Processing for Estimating Stationary Low-Frequency Electromechanical Modes of Power Systems. *IEEE Transactions on Power Systems*, *18* (1), 167-173.

Wolf, A., Swift, J. B., Swinney, H. L., & Vastano, J. A. (1985). Determining Lyapunov Exponents from a Time Series. *Physica D.*, *16,* 285-317.

Zadeh, A.,L. (1956). On system identification problem. *IRE Transaction on circiut theory, 3*, 277-281.

Zhu Y. (2001). *Multivariable system identification for process control*. Oxford UK: Elsevier Science Ltd.