NEURO – SLIDING MODE CONTROLLERS FOR SYSTEMS WITH UNCERTAINTIES

by

YILDIRAY YILDIZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2004

NEURO – SLIDING MODE CONTROLLERS
FOR SYSTEMS WITH UNCERTAINTIES

APPROVED BY:

Prof. Dr. Asif Sabanovic …….

(Dissertation Supervisor)

Dr. Serhat Yesilyurt …….

(Dissertation Co-Supervisor)

Dr. Ahmet Onat …….

Dr. Kemalettin Erbatur …….

Dr. Mustafa Unel …….

DATE OF APPROVAL: …….

ABSTRACT

A Neuro – Sliding Mode Controller was designed for systems that have uncertainties like unknown external disturbances and unknown system parameters. First, the controller was designed for single input single output (SISO) systems and then it was generalized for a certain class of multi input multi output systems. Stability proof was given using Lyapunov Stability Criteria and finally, the theory was supported by simulation and experimental results.

The Neuro – Sliding Mode Controller proposed in this thesis consists of a one layered neural network whose activation functions are linear. The main working principle of the controller is minimizing a cost function which is determined from the requirements of the Lyapunov Stability Criteria and Sliding Mode Control Theory.

The major contribution of this work is that, different from the similar works in the field, the stability of the overall control system was shown by analyzing the properties of the cost function introduced to the neural network for minimization.

Two different experimental setups were used for SISO and MIMO cases respectively. For the SISO case, the position of an electrical motor that actuates a linear servo – drive was controlled. For the MIMO case, a system consisting of two piezoelectric actuators connected to each other via a load cell, which was used for force measurement, was used. In this system, the position of one actuator and the internal force created were controlled simultaneously. Both experiments were successful and supported the theory.

ÖZET

Bilinmeyen dışşal etkenlerin etki ettigi ve parametrelerinin tam olarak bilinemedigi sistemlerin denetimi icin yapay sinir aglari kullanilarak kayan kipli bir denetim algoritması geliştirilmiştir. Bu algoritma öncelikle tek giriş ve tek çıkışlı (TGTÇ) sistemler için dizayn edilmiş, daha sonra da, bir sınıf çok girişli ve çok çıkışlı (ÇGÇÇ) sistemler için genelleştirilmiştir. Ortaya çıkan denetim sisteminin kararlılığı gösterilmiş ve son olarak da simulasyon ve deneylerle teori desteklenmiştir.

Dizayn edilen denetim sistemi, aktivasyon fonksiyonları doğrusal olan, tek katmanlı bir yapay sinir ağı içermektedir. Bu denetleyicinin temel çalışma prensibi, Lyapunov Kararlılık Kriterleri ve Kayan Kipli Denetim Teorisi kullanilarak seçilmiş olan bir maliyet fonksiyonun minimizasyonudur.

Bu çalışmanın en önemli katkısı, bu alanda yapılmış daha önceki çalışmalardan farklı olarak, yapay sinir ağı ile minimizasyonu yapılan maliyet fonksiyonunun özellikleri incelenerek, tüm denetim sisteminin kararlılık ispatının yapılmış olmasıdır.

TGTÇ ve ÇGÇÇ durumları için iki ayrı deney düzeneği kullanılmıştır. TGTÇ için doğrusal eksenli konumlandırma sürücüsünde kullanılan elektrik motorunun pozisyon denetimi yapılmıştır. ÇGÇÇ durumu içinse, birbirlerine aralarındaki kuvvet ölçmek için kullanılan bir yük hücresi - “load cell”-, vasıtasıyla etki eden iki piezoelektrik malzemeden oluşan bir deney düzeneği kullanılmıstır. Bu deneyde piezoelektrik malzemelerden bir tanesinin pozisyon denetimi yapılırken, aynı anda sistemde yük hücresinin tepkisinden dolayı oluşan kuvvetin de denetimi yapılmıştır. Her iki deney de başarılı olmuş ve öne sürülen teoriyi desteklemiştir.

Bilimsel merakımı tetikleyen ve bana her zaman cesaret veren babam,
Şakir Yıldız'a...

ACKNOWLEDGMENTS

The whole story started with my talk with Professor Asif Sabanovic and his proposing the research problem to me. I want to thank him not only for this reason but also for his limitless help and support during my studies.

Special thanks to Dr. Serhat Yesilyurt for helping me with the preparation of a good thesis.

I also thank Dr. Mustafa Unel for long discussions we had about my studies and other subjects. I learned a lot from him about scientific thinking, reasoning and ethics. I really believe that, meeting with him at Sabanci University was one of the biggest chances for me, during my life time.

Cenk Yıldız, my cousin, helped me with the arrangement of the material in the thesis. I thank him for sharing the burden.

Finally, I want to thank to my family members for their love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	:	Amount that the main node of the network is changed
M	:	Amount that the other nodes in the network is changed
W	:	Weight matrix
i, j	:	Index numbers
b	:	Threshold value for a node
$\Gamma[\cdot]$	:	Nonlinear operator
$\gamma(\cdot)$	:	Smooth activation function
u	:	Control input
y	:	Output of a system
R	:	Activation function of radial basis neural network
c	:	Center of a receptive field in radial basis networks
σ_{ij}	:	Width of a receptive field
$f(\cdot)$	:	Function
T	:	Number of nodes in a layer
f_T	:	Map obtained by neural network
C_f	:	First moment of the Fourier transform
y_d	:	Desired trajectory
P	:	An operator
$\mathbf{C}$	:	Controller
n	:	Number of the order of a system
x	:	State vector
B	:	Input Matrix
d	:	External Disturbance
$y^{(n)}$	:	Time derivative of y, taken n times.

x_d	:	Desired state vector
e_t	:	Tracking error vector
e	:	Output error
$f(x)$	:	Function of x
σ	:	Sliding function
$\dot{e}$	:	Time derivative of the output error
C	:	A positive constant
$V(\sigma)$	:	Lyapunov function candidate with σ being the argument
G	:	A row vector
$\boldsymbol{G}$	:	A matrix
D	:	A positive constant
$\boldsymbol{D}$	:	A positive definite matrix
w	:	Neural network weight – scalar –
E	:	The error function to be minimized
$\dot{\sigma}$	:	Time derivative of the sliding function
η	:	Learning constant
$\overline{\eta}$	:	Modified learning constant
h	:	Sampling time
χ	:	Longitudinal position of the load
θ_1	:	Angular position of the pulley driven by the servomotor
θ_2	:	Angular position of the undriven pulley
v	:	Longitudinal velocity of the load
$\dot{V}$	:	Time derivative of the Lyapunov function candidate
T_L	:	Friction torque at the servomotor side
F_1, F_2, F_3	:	Forces in the different part of the belt
F_L	:	Friction force at the load side
J	:	Equivalent moment of inertia on the motor side
θ	:	Angular position on the servomotor's shaft
ω	:	Angular velocity of the servomotor's shaft
F_B	:	Belt elasticity force proportional to the belt stretch
m	:	Mass of the load

F_D	:	Belt damping force proportional to the derivative of stretch
F_L	:	Friction force at the load side
r	:	Radius of pulleys
$D(t)$	:	Time varying disturbance
K_t	:	Torque constant
u_h	:	Voltage due to hysteresis effect
S	:	Strain tensor
s^E	:	Elastic compliance matrix when subjected to constant electric field
ε^T	:	Permittivity measured at a constant stress
$\boldsymbol{T}$	:	Stress tensor
$\boldsymbol{d}$	:	Matrix of a piezoelectric material constants
$\boldsymbol{E}$	:	Electric field vector
$\boldsymbol{D}$	:	Electric displacement vector
H	:	Hysterisis effect
T_{em}	:	Electromechanical transducer with transformer ratio T_{em}
C	:	Sum of the capacitances of the individual PZT wafers.
$\dot{q}$	:	Total current flowing through the circuit
q_p	:	Transduced charge from the mechanical side.
u_p	:	Voltage due to the piezo effect
u_{in}	:	The total voltage over the piezo actuator
F_{ext}	:	Externally applied force
x	:	Elongation of the piezo actuator
M	:	Mechanical relation between F_p and x
F_p	:	Transduced force from the electrical side
α, a, b	:	Constants
q_c, q_{ll}	:	Upper right- and lower left-hand-side points of a hysteresis loop
A	:	Cross-sectional area of the piezo actuator
$\boldsymbol{A}$	:	Input amplitude of the voltage feeding piezo actuator
c_p	:	Damping coefficient of the piezo actuator
k_p	:	Stiffness of the piezo actuator

L	:	Length of the piezo actuator
m_p	:	Mass of the piezo actuator
T	:	The torque developed by the servomotor
t	:	Time
η	:	Viscosity
ρ	:	Mass density

LIST OF ABBREVIATIONS

ADALINE	:	Adaptive linear element
ANN	:	Artificial neural network
DD	:	Direct drive
DOF	:	Degree of freedom
MIMO	:	Multi input, multi output
MNN	:	Multi layer neural network
NN_1, NN_2	:	Neural network one, neural network two
PD	:	Proportional and derivative
PI	:	Proportional and integral
PID	:	Proportional, integral and derivative
PZT	:	Led zirconium titanate
RBFN	:	Radial basis function network
RNN	:	Recurrent neural network
SISO	:	Single input, single output

NEURO – SLIDING MODE CONTROLLERS FOR SYSTEMS WITH UNCERTAINTIES

by

YILDIRAY YILDIZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2004

NEURO – SLIDING MODE CONTROLLERS
FOR SYSTEMS WITH UNCERTAINTIES

APPROVED BY:

Prof. Dr. Asif Sabanovic …….

(Dissertation Supervisor)

Dr. Serhat Yesilyurt …….

(Dissertation Co-Supervisor)

Dr. Ahmet Onat …….

Dr. Kemalettin Erbatur …….

Dr. Mustafa Unel …….

DATE OF APPROVAL: …….

ABSTRACT

A Neuro – Sliding Mode Controller was designed for systems that have uncertainties like unknown external disturbances and unknown system parameters. First, the controller was designed for single input single output (SISO) systems and then it was generalized for a certain class of multi input multi output systems. Stability proof was given using Lyapunov Stability Criteria and finally, the theory was supported by simulation and experimental results.

The Neuro – Sliding Mode Controller proposed in this thesis consists of a one layered neural network whose activation functions are linear. The main working principle of the controller is minimizing a cost function which is determined from the requirements of the Lyapunov Stability Criteria and Sliding Mode Control Theory.

The major contribution of this work is that, different from the similar works in the field, the stability of the overall control system was shown by analyzing the properties of the cost function introduced to the neural network for minimization.

Two different experimental setups were used for SISO and MIMO cases respectively. For the SISO case, the position of an electrical motor that actuates a linear servo – drive was controlled. For the MIMO case, a system consisting of two piezoelectric actuators connected to each other via a load cell, which was used for force measurement, was used. In this system, the position of one actuator and the internal force created were controlled simultaneously. Both experiments were successful and supported the theory.

ÖZET

Bilinmeyen dışşal etkenlerin etki ettigi ve parametrelerinin tam olarak bilinemedigi sistemlerin denetimi icin yapay sinir aglari kullanilarak kayan kipli bir denetim algoritması geliştirilmiştir. Bu algoritma öncelikle tek giriş ve tek çıkışlı (TGTÇ) sistemler için dizayn edilmiş, daha sonra da, bir sınıf çok girişli ve çok çıkışlı (ÇGÇÇ) sistemler için genelleştirilmiştir. Ortaya çıkan denetim sisteminin kararlılığı gösterilmiş ve son olarak da simulasyon ve deneylerle teori desteklenmiştir.

Dizayn edilen denetim sistemi, aktivasyon fonksiyonları doğrusal olan, tek katmanlı bir yapay sinir ağı içermektedir. Bu denetleyicinin temel çalışma prensibi, Lyapunov Kararlılık Kriterleri ve Kayan Kipli Denetim Teorisi kullanilarak seçilmiş olan bir maliyet fonksiyonun minimizasyonudur.

Bu çalışmanın en önemli katkısı, bu alanda yapılmış daha önceki çalışmalardan farklı olarak, yapay sinir ağı ile minimizasyonu yapılan maliyet fonksiyonunun özellikleri incelenerek, tüm denetim sisteminin kararlılık ispatının yapılmış olmasıdır.

TGTÇ ve ÇGÇÇ durumları için iki ayrı deney düzeneği kullanılmıştır. TGTÇ için doğrusal eksenli konumlandırma sürücüsünde kullanılan elektrik motorunun pozisyon denetimi yapılmıştır. ÇGÇÇ durumu içinse, birbirlerine aralarındaki kuvvet ölçmek için kullanılan bir yük hücresi - “load cell”-, vasıtasıyla etki eden iki piezoelektrik malzemeden oluşan bir deney düzeneği kullanılmıstır. Bu deneyde piezoelektrik malzemelerden bir tanesinin pozisyon denetimi yapılırken, aynı anda sistemde yük hücresinin tepkisinden dolayı oluşan kuvvetin de denetimi yapılmıştır. Her iki deney de başarılı olmuş ve öne sürülen teoriyi desteklemiştir.

Bilimsel merakımı tetikleyen ve bana her zaman cesaret veren babam,
Şakir Yıldız'a...

ACKNOWLEDGMENTS

The whole story started with my talk with Professor Asif Sabanovic and his proposing the research problem to me. I want to thank him not only for this reason but also for his limitless help and support during my studies.

Special thanks to Dr. Serhat Yesilyurt for helping me with the preparation of a good thesis.

I also thank Dr. Mustafa Unel for long discussions we had about my studies and other subjects. I learned a lot from him about scientific thinking, reasoning and ethics. I really believe that, meeting with him at Sabanci University was one of the biggest chances for me, during my life time.

Cenk Yıldız, my cousin, helped me with the arrangement of the material in the thesis. I thank him for sharing the burden.

Finally, I want to thank to my family members for their love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	:	Amount that the main node of the network is changed
M	:	Amount that the other nodes in the network is changed
W	:	Weight matrix
i, j	:	Index numbers
b	:	Threshold value for a node
$\Gamma[\cdot]$	:	Nonlinear operator
$\gamma(\cdot)$	:	Smooth activation function
u	:	Control input
y	:	Output of a system
R	:	Activation function of radial basis neural network
c	:	Center of a receptive field in radial basis networks
σ_{ij}	:	Width of a receptive field
$f(\cdot)$	:	Function
T	:	Number of nodes in a layer
f_T	:	Map obtained by neural network
C_f	:	First moment of the Fourier transform
y_d	:	Desired trajectory
P	:	An operator
$\mathbf{C}$	:	Controller
n	:	Number of the order of a system
x	:	State vector
B	:	Input Matrix
d	:	External Disturbance
$y^{(n)}$	:	Time derivative of y, taken n times.

x_d	:	Desired state vector
e_t	:	Tracking error vector
e	:	Output error
$f(x)$	:	Function of x
σ	:	Sliding function
$\dot{e}$	:	Time derivative of the output error
C	:	A positive constant
$V(\sigma)$	:	Lyapunov function candidate with σ being the argument
G	:	A row vector
$\boldsymbol{G}$	:	A matrix
D	:	A positive constant
$\boldsymbol{D}$	:	A positive definite matrix
w	:	Neural network weight – scalar –
E	:	The error function to be minimized
$\dot{\sigma}$	:	Time derivative of the sliding function
η	:	Learning constant
$\overline{\eta}$	:	Modified learning constant
h	:	Sampling time
χ	:	Longitudinal position of the load
θ_1	:	Angular position of the pulley driven by the servomotor
θ_2	:	Angular position of the undriven pulley
v	:	Longitudinal velocity of the load
$\dot{V}$	:	Time derivative of the Lyapunov function candidate
T_L	:	Friction torque at the servomotor side
F_1, F_2, F_3	:	Forces in the different part of the belt
F_L	:	Friction force at the load side
J	:	Equivalent moment of inertia on the motor side
θ	:	Angular position on the servomotor's shaft
ω	:	Angular velocity of the servomotor's shaft
F_B	:	Belt elasticity force proportional to the belt stretch
m	:	Mass of the load

F_D	:	Belt damping force proportional to the derivative of stretch
F_L	:	Friction force at the load side
r	:	Radius of pulleys
$D(t)$	:	Time varying disturbance
K_t	:	Torque constant
u_h	:	Voltage due to hysteresis effect
S	:	Strain tensor
s^E	:	Elastic compliance matrix when subjected to constant electric field
ε^T	:	Permittivity measured at a constant stress
$\boldsymbol{T}$	:	Stress tensor
$\boldsymbol{d}$	:	Matrix of a piezoelectric material constants
$\boldsymbol{E}$	:	Electric field vector
$\boldsymbol{D}$	:	Electric displacement vector
H	:	Hysterisis effect
T_{em}	:	Electromechanical transducer with transformer ratio T_{em}
C	:	Sum of the capacitances of the individual PZT wafers.
$\dot{q}$	:	Total current flowing through the circuit
q_p	:	Transduced charge from the mechanical side.
u_p	:	Voltage due to the piezo effect
u_{in}	:	The total voltage over the piezo actuator
F_{ext}	:	Externally applied force
x	:	Elongation of the piezo actuator
M	:	Mechanical relation between F_p and x
F_p	:	Transduced force from the electrical side
α, a, b	:	Constants
q_c, q_{ll}	:	Upper right- and lower left-hand-side points of a hysteresis loop
A	:	Cross-sectional area of the piezo actuator
$\boldsymbol{A}$	:	Input amplitude of the voltage feeding piezo actuator
c_p	:	Damping coefficient of the piezo actuator
k_p	:	Stiffness of the piezo actuator

L	:	Length of the piezo actuator
m_p	:	Mass of the piezo actuator
T	:	The torque developed by the servomotor
t	:	Time
η	:	Viscosity
ρ	:	Mass density

LIST OF ABBREVIATIONS

ADALINE	:	Adaptive linear element
ANN	:	Artificial neural network
DD	:	Direct drive
DOF	:	Degree of freedom
MIMO	:	Multi input, multi output
MNN	:	Multi layer neural network
NN_1, NN_2	:	Neural network one, neural network two
PD	:	Proportional and derivative
PI	:	Proportional and integral
PID	:	Proportional, integral and derivative
PZT	:	Led zirconium titanate
RBFN	:	Radial basis function network
RNN	:	Recurrent neural network
SISO	:	Single input, single output

1. INTRODUCTION

1.1. Artificial Neural Networks

In the last fifteen years, the field of neural networks became popular among researchers and this resulted in considerable amount of work in the field. Being a subject to the interest of many researchers, neural networks do not have a standard, agreed definition. In the literature, among the attempts to make a neat introduction to the field, the work of J.M. Bishop and R.J. Mitchell [1] attracts attention. The introduction given here is taken from this work [1]:

The field of neural networks, also known as "connectionism", parallel distributed processing, connection science or neural computing, is a mode of computing which seeks to include the style of computing used in the brain. In their book An Introduction to Neural Computing [2], Alexander and Morton have the following definition:

> *"Neural computing is the study of networks of adaptable node which, through a process of learning from task examples, store experimental knowledge and make it available for use."*

A neural network is a processor of information consisting of simple processing elements connected together. Each processing element is a very simple model of a neuron in the brain, hence the term neural networks. Thus, a neural network could be described as an attempt to create an artificial brain. However, present neural networks try to mimic the way the brain does things in order to harness its ability to infer and induce from incomplete and confusing information.

What makes these networks powerful is their potential for performance improvement over time as they acquire more knowledge about a problem, and their ability to handle fuzzy, real world data. That is, a network "taught" certain data patterns,

is able to recognize both these patterns and those which are similar: the network is able to generalize. Also, neural networks are inherently parallel in their operation, and so have the capacity to operate much faster than conventional computers.

Work in the neural networks began in 1940s and flourished under McCulloch, Pitts, Hebb, Widrow, Minsky and Rosenblatt until 1969. Unfortunately, the claims by Rosenblatt as to what his machine could do were somewhat exaggerated, and in 1969 Minsky and Papert wrote a book, *Perceptrons* [3], which showed that there were various problems with these neural networks and so the subject went out of fashion until the mid 1980s. During this time, however, various people were still working in the field, including Alexander, Kohonen, Hopfield, and the PDP group including Rumelhart, McClelland and Hinton. Neural networks have become fashionable again because it has been realized that there are limitations as to what Expert Systems do, and the success of Hopfield and the PDP group. In 1986, Rumelhart and McClelland published Parallel Distributed Processing [4] in which it is shown how the objections of Minsky and Papert could be overcome, and work in the field flourished.

A neural network consists of simple processing elements connected together. There are various ways in which these elements can be connected, in single or multiple layers, fully or partially connected, etc., and there are various forms of processing elements. However, many factors are common.

A neural network is not programmed to complete a given task, rather it adapts and acquires knowledge over time in order to complete a task. One stage of operation, therefore, requires the network to learn, and this can be supervised or unsupervised. Once the network has learned to perform a task, it can then be set to undertake that task. For example, a neural network which is to be used to recognize objects would first be taught what those objects looked like, and then would be put in the mode whereby it reported whether the object it was being shown was one it had been taught.

There are many forms of neural network of which the main types are “Perceptron” types, Hopfield nets, Boltzmann machines, Weightless or n-tuple nets, Kohonen nets, the neocognitron and ART classifiers. The two types which are used the most, and which are more applicable in the area covered by colloquium, are “Perceptron” and Kohonen nets.

1.1.1. Perceptron Networks

In these networks, the main element is an extension of the McCulloch and Pitts neuron, as shown in Fig. 1.1. The output of the element is some function of the weighted sum of all the inputs. Sometimes the output value is used directly, or the output is processed by, say, a threshold or the sigmoid function. In some networks it is important to know if the element has "fired" – in some way that a neuron in the brain produces an active output- and this occurs if the weighted sum exceeds a given threshold. For other networks, the actual value of the output is required.

Learning is the process of deciding what the values of the weights should be. This is often achieved by comparing the output of the element with what it should be – a form of supervised learning – and adjusting the weights appropriately.

A single layer network of such neurons can perform simple functions. However one of the criticisms raised by Minsky and Papert was that such a network could not solve a problem that is not linearly separable, which is one where one straight line cannot separate opposing classes. However, a multilayer network can solve such problems.

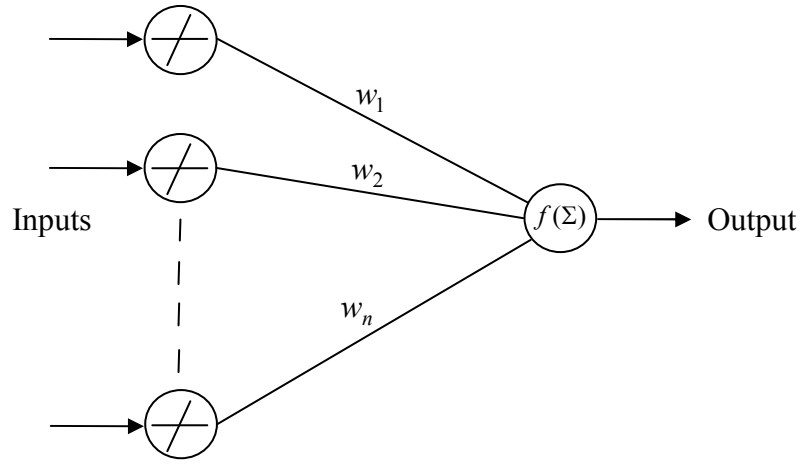

Figure 1.1. Basic neuron model

In learning mode, the network is repeatedly presented the data in the training set, both the required input and output for each item in the set, and the weights adjusted accordingly using the generalized delta rule, by propagating errors between the actual output and the desired output back through the network. This can take time as the whole data set must be presented many hundred times. This is a severe disadvantage of the method. However, a great many researchers in the field use backpropagation learning on multi layer perceptrons.

1.1.2. Kohonen Networks

Teuvo Kohonen postulated a different form of network with a different learning strategy. The learning of these networks is unsupervised, which in some ways is more like the form in which humans learn. In Kohonen networks, each node has a number of values associated with it, one value for each input to the network, and each is connected to each node. In use, the values associated with each node are compared with the input, and the node which most closely resembles the input is the one which fires. The system must be taught patterns to fire, and also be able to recognize similar patterns. This is achieved by the following way.

In learning mode, the node which most resembles the input is determined. Then, its weights are adjusted so as to reduce the difference between the weights and the inputs. This allows the node to resemble the input more closely. So that, similar patterns are also remembered, those nodes which are connected to this node, that is those which are adjacent to it, also have their weights adjusted, but the amount by which they are adjusted is determined by the distance of each node from that which fired. If the main node is changed by an amount δ, the other nodes are changed by $M\delta$, where M is determined by the "maxican hat" function shown in Fig. 1.2. Therefore, the system should be able to remember the taught input patterns, and similar patterns, that is the network can generalize. Kohonen has used these networks successfully in the field of speech recognition and many other researchers use Kohonen networks.

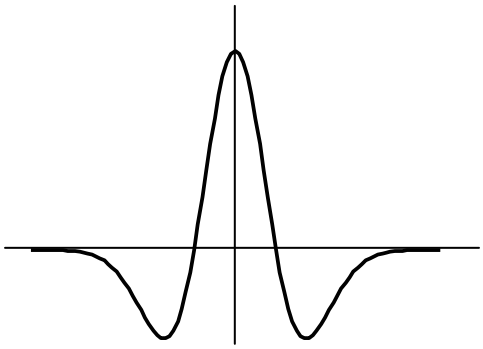

Figure 1.2. “Mexican Hat”

1.2. Neural Networks for Control

The past fifteen years have witnessed a great deal of progress in both theory and practice of control using neural networks. A review of control practice using neural networks and the theory related to it can be found in the work of Kumpati S. Narendra [5], from which the following summary is taken:

The field of control is inherently interdisciplinary in nature and extends from design, development, and production on the one hand to mathematics on the other. Since the very beginning over six decades ago, control research has been driven by the diverse and changing nature of technology. Today, control techniques have become pervasive in a wide spectrum of applications which are major scientific, technological, and economic importance. Mathematical control theory has been described as an indispensable component of a partnership between mathematical theory, engineering practice, and hardware and software capabilities.

The objective of control is to influence the behavior of dynamical systems. It includes maintaining the outputs of systems at constant values, regulation, or forcing them to follow prescribed time functions, tracking. The control is to find the necessary control inputs to the system using all available data. Achieving fast and accurate control while assuring stability and robustness is the aim of all control systems design.

The best developed part of control theory deals with linear systems and powerful methods for designing controllers for such systems are currently available. In fact, most of the controllers used in modern industry belong this class. However, as applications become more complex, the processes to be controlled are increasingly characterized by

poor models, distributed sensors and actuators, multiple subsystems, high noise levels and complex information patterns. The difficulties encountered in designing controls for such processes can be broadly classified under three headings: 1) complexity, 2) nonlinearity, 3) uncertainty.

A neural network can be considered as a conveniently parameterized class of nonlinear maps, from a systems theoretic point of view. During the 1980's and the early 1990's, conclusive proofs were given by numerous authors that multilayer feedforward networks (MNN) are capable of approximating any continuous function on a compact set in a very precise and satisfactory sense [6]-[9] . As a result, such networks found wide application in many fields, both for function approximation and pattern recognition. Since dynamics is an essential part of physical systems, it was proposed in [10], that neural networks should be used as components in dynamical systems and that a study of such networks should be undertaken within a unified framework of systems theory. A great deal of progress has been made since that time both in the theory and practice of control using neural networks, i.e., the field of neurocontrol. Numerous dynamical systems have been identified and controlled in computer simulations, and a few have been practically implemented. It has become increasingly evident that artificial neural networks (ANN) are capable of coping with all three categories of difficulties which arise in complex control systems.

In the 1980's the MNN shown in Fig. 1.3 was introduced for approximating continuous functions. Later, the radial basis function network (RBFN) was proposed as a viable alternative. The n layer MNN with input u and output y is described by the

$$\Gamma\left[W_n\Gamma\left[W_{n-1}...\Gamma\left[W_1u+b_1\right]+...+b_{n-1}\right]+b_n\right]=y \tag{1.1}$$

where W_i is the weight matrix in the i^{th} layer, and the vectors b_i $(i=1,2,..,n)$ represents the threshold values for each node in the i^{th} layer. $\Gamma[\cdot]$ is a nonlinear operator with $\Gamma(x)=\left[\gamma_1(x),\gamma_2(x),...,\gamma_n(x)\right]$ where $\gamma_i(\cdot)$ is a smooth activation function , which is generally sigmoid.

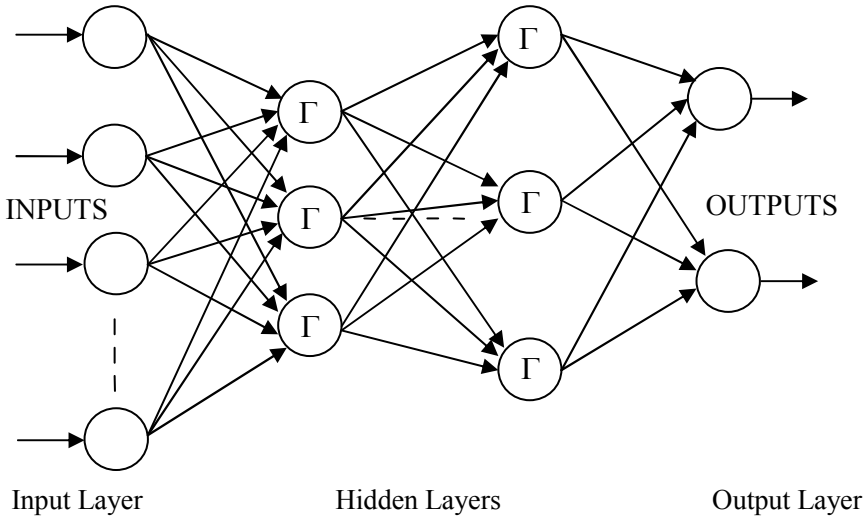

Figure 1.3. Structure of the multi layer neural network

In Fig. 1.4, the structure of the RBFN with an input $u \in \Re^n$ and output $y \in \Re$ is shown. The output is described by the equation

$$y = f(u) = \sum_{i=1}^{N} W_i R_i(u) + W_0 \tag{1.2}$$

where W_i $(i = 0,1,2,...,N)$ are the weights, and the function $R_i : \Re^n \to \Re$ have generally the form

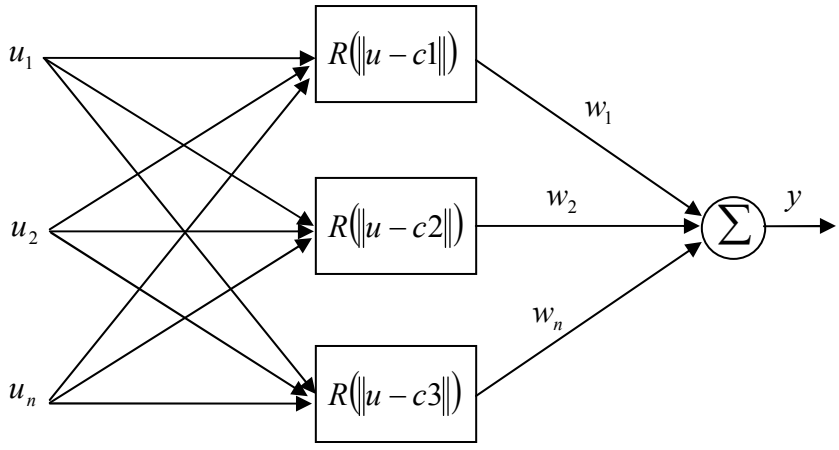

Figure 1.4. Structure of the radial basis function network

$$R_i(u) = \exp\left[-\sum_{j=1}^{n} \frac{\left(u_j - c_{ij}\right)^2}{2\sigma_{ij}^{\ 2}} \right] \tag{1.3}$$

where $c_i^{\ T} = [c_{i1},...,c_{in}]$ is the center of the i^{th} receptive field and σ_{ij} is referred to as its width.

Both MNN and RBFN are capable, at least in theory, of coping with complexity, nonlinearity, and uncertainty encountered in complex systems. The massive parallel nature of the MNN permits computation to be performed at high rates. Since they can approximate nonlinear maps to any desired degree of accuracy, they can also be used to identify and control nonlinear dynamical systems. Finally, the fact that various algorithms are currently available for the adjustment of the parameters of the networks implies that they can deal with uncertainty, by realizing approximations of unknown static and dynamic mappings, from IO data.

While it has been shown that both MNN and RBFN can approximate arbitrary functions from one finite dimensional space to another to any desired degree of accuracy, it is also true that polynomials, trigonometric series, splines, and orthogonal functions share the same properties. Hence, questions naturally arise as to why ANN should be preferred over such methods. Extensive computer studies carried out during the past years have revealed that neural networks enjoy numerous practical advantages over conventional methods. In view of their architecture they are more fault tolerant and less sensitive to noise and they are more easily implementable in hardware because of the parameterization used. Barron's work [11] has also provided partial theoretical justification for using ANN's over its competitors. If the class of functions F is restricted to those whose Fourier transforms have a first moment, and $f : \Re^d \to \Re$ is a member of this class, the problem of approximating it with a network having one hidden layer and T nodes in that layer is considered [11]. If f_T is the map obtained by the neural network, it is shown that the L_2 norm of the approximation error $\|f - f_t\|_2$ is bounded by $O\left(C_f / T^{1/2}\right)$ where C_f is the first moment of the Fourier transform. In contrast to this, no linear combination of T fixed basis functions can achieve an approximation error smaller than $O\left(C_f / T^{1/d}\right)$ Hence, as the dimensionality of input space increases, it is clear that MNN are preferable to approximation schemes in which the adjustable parameters arise linearly.

The above theoretical result has great significance for the design of practical controllers for dynamical systems, since the dimensionality of the input space is invariably large in such cases. Both MNN, in which the parameters occur nonlinearly, and RBFN in which c_{ij} and σ_{ij} are adjusted, require substantially fewer parameters for a desired degree of accuracy.

1.2.1. Applications of Neurocontrol

Most of the current interest in the application of neural networks is in static systems, particularly in recognition, where they have been very successful. The problem of control is a considerably difficult one, and the presence of a feedback loop implies that stability problems are invariably present. Even though such stability questions will eventually become very important in industrial applications, the neurocontrol problems being attempted at the present time are those in which improving performance rather than assuring stability is the main consideration.

It is true in control practice that the simplest controller which satisfies all the constraints while meeting performance specifications, is the one most likely to be chosen in any application. This is because simplicity generally correlates strongly with robustness as well as low cost. Neural network based controllers, on the other hand, contain a large number of parameters and are invariably complex. Hence, simple controllers such as constant gain controllers, PI and PID, state feedback controllers, and linear adaptive controllers must all be tried and found inadequate in some sense before neural controllers are considered seriously in applications.

In the initial stages of any field, when theory is not well developed, it is natural for heuristic techniques to be used in the solution of practical problems. There are numerous examples of such periods in the history of automatic control. Before 1868 automatic control systems were designed through intuition and invention. Efforts to increase the accuracy of the system led to problems of instability and the field had to wait Maxwell's theory for an explanation. In the 1950's and 1960's, when the area of adaptive control was in its initial stages, ingenious schemes were proposed with little theoretical justification. History repeated itself in 1980's, when a number of approximate schemes, based on simple concepts, were proposed in the neurocontrol

literature for the output y of a plant to track a desired output y^* with a small error. The basic idea of the above schemes was to determine the inverse of an operator P, which represents the plant, and use it as a controller $\mathbf{C}$, so that the plant together with the controller, PC, would approximate a unit operator over the range of values of the reference input. The different methods of computing the inverse gave rise to different control architectures, some of which are shown in Fig. 1.5. In "Direct Inverse Control", Fig. 1.5.a, the input to the controller is y and the controller, a neural network, is trained to approximate the input to the plant. In "Feedforward Inverse Control", Fig. 1.5.b, a neural network NN_2 is trained so that its output $\hat{y}$ approximates the output of the plant. A controller NN_1 is then trained so that $(NN_2)(NN_1)$ approximates a unit operator. In internal model control, the error e between model and plant is fed back to the input so that NN_1, NN_2, and P form part of a feedback loop as shown in Fig. 1.5.c. This model has been found to be very successful in practical applications in chemical engineering.

All the above approaches are approximate, since the plant is a dynamic rather that a static map. The latter raises many mathematical questions such as existence of solutions, left versus right inverses of operators, as well as those related to the domains and range spaces of the operators being approximated. Nevertheless, these questions are mentioned here because successful heuristic techniques are the precursors to good theory and the latter is quite often merely a precise way of stating intuitive concepts.

The problem of nonlinear control is very complex and even at the present time a great deal of heuristic reasoning is used in applications. Simple schemes, such as those discussed earlier, have yielded impressive results in simulation studies as well as in some applications with long time constants. But they do not offer guarantees about their applicability under very different operating conditions. Theoretical control methods become increasingly relevant to provide such guarantees. This is not to imply that such methods can be directly applied to practical problems. Practical control system design invariably involves compromises and approximations, but a sound theory provides a sound basis for such engineering choices. In the following a few applications of neurocontrol selected from a large pool is presented.

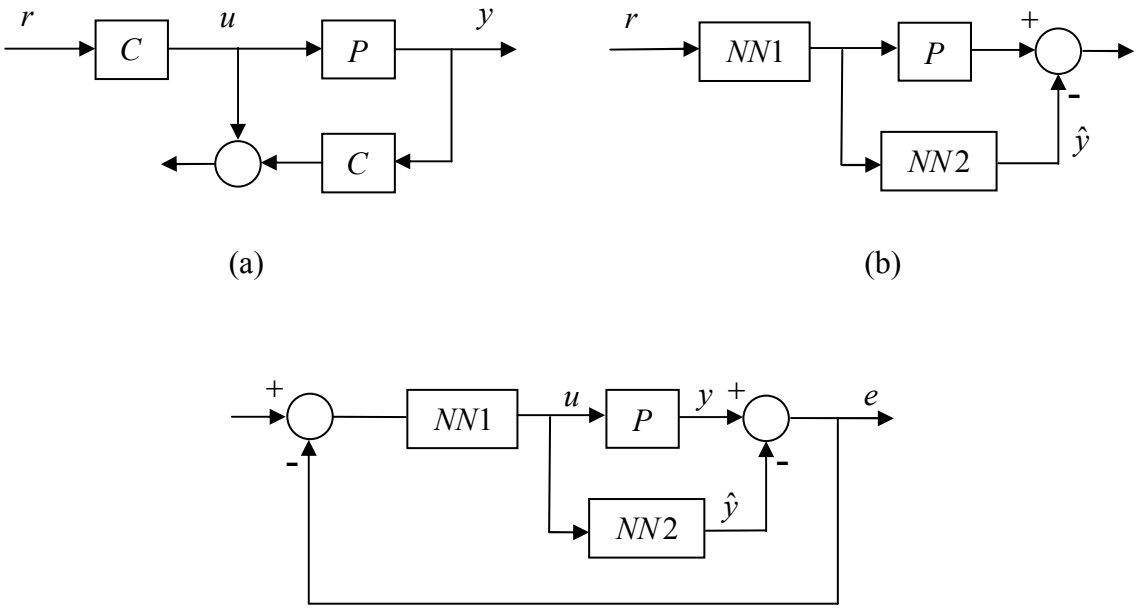

Figure 1.5. Control architectures

1.2.1.1 Robotics

Neural networks have been extensively used in robotics and the areas where neurocontrollers have found application can be conveniently classified as: 1) manipulator control, 2) contact control, 3) coordination, grasping, and manipulation, 4) locomotion, and 5) planning and navigation.

In manipulator control, the use of neural networks has been strongly motivated by the enormous complexity of computing kinematics in real time. Since adaptive laws with kinematics terms computed based on desired rather that actual trajectories had been shown to be stable, neural networks could be trained off-line to learn the dynamics of the robot.

This has been the main approach used in this area starting with the work of Kawato [12], and also used in the work of Zomoya and Nabhan [13]. In that works, the authors have shown that satisfactory tracking can be achieved for unknown payloads and that neural networks can adapt rapidly to changing payloads on-line. With few exceptions [14]-[16], most work on robot control has been in the form of simulation studies where dynamic model parameter values have been obtained from experimental studies.

Learning impedance relationships for the purpose of transferring human skills to robots in tool handling, and learning environmental models for overall system stability and performance are two areas where some effort has been made to apply neural networks to contact control. Asada [17] has applied neural networks for learning skill from real force and motion data. Learning of a skill in many cases implies the learning of the combined impedance of the human and the tool, and this is accomplished using a neural network. Transferring the skill to a robot implies that the robot must be able to change its impedance through control to match what was learned in the neural network. Using recurrent networks, Venkataraman [18], has developed a robotic system that has the ability to autonomously acquire models of soil and rock samples found on planetary surfaces.

In view of the inherent complexity of the coordination, grasping, and manipulation, neural networks have been used very few. A notable work is Hwang [19], who has considered the problem of cooperative control of two robots that grasp and carry an object. Two neural networks, one for each arm, are trained with data from the system. Another problem area where neural networks have been used effectively is due to Hanes [20] and is related to control of contact forces between the fingers of a robot hand and the object it grasps. The problem of determining grasp locations that will result in a desired force distribution is complex from the computational viewpoint and neural networks are ideally suited for it. Neural networks were found to yield stable grasp configurations over a wide range of object sizes and clinch levels.

In the area of locomotion, Miller [21] has been systematically using neural networks to control biped walking robots. He and his co-workers have designed, implemented and tested an adaptive dynamic balance scheme on such an experimental robot. According to them, while the control problems for dynamic walking are more complicated than with static balance, dynamic walking promises higher walking speeds, improved walking structures and greater efficiency.

A truly interesting and novel application of neural adaptive control is described by Pomerleau in [22] and is the area of vision-based autonomous driving. Based on images from an on-board video camera, a robot van, equipped with motors for the steering wheel, braking and the acceleration pedal, determines its own trajectory. The noteworthy feature of the system is the supervisory control method used to train it. The neural network is taught to imitate the driving reactions of a person. As the person drives, the neural network is trained using backpropagation. The şnout to the neural

network is a 30 x 32 unit video image and the output layer consists of 30 nodes each of which corresponds to steering direction.

A very exciting application in a somewhat different direction is by Beer and his colleagues who believe that neural network architectures abstracted from biological systems can be directly applied to the control of autonomous agents [23]. Since they have evolved over a long periods of time, even simple animals are capable of feats of sensorimotor control that are far superior to the most of the sophisticated robots. In [24], a fully distributed neural network architecture for controlling the locomotion of a Hexapod robot is described. It is shown that the controller can be utilized to direct the robust locomotion of an actual six-legged robot to achieve a wide range of gaits.

1.2.1.2 Neural Networks in Aeronautics

Another area where neural networks are finding application is aeronautics. For an excellent review of the problems that arise in such systems and manner in which neural networks may be used, [25] can be consulted.

Driven by cost and operational requirements operating envelopes of aircraft are being extended toward regimes governed by unsteady fluid mechanics. These, in turn, are bringing in their wake complex stability and control problems. Under such operating conditions, to maintain safety and operational readiness, new types of control systems are needed. Neural network technologies are being explored for fault diagnosis, control reconfiguration, identification of nonlinear dynamics and adaptive control.

The use of the theory and technology of ANNs for problems of identification, diagnosis, and control in large complex space systems is proposed by Rauch and Schaecter [26]. Improvement of performance without interrupting the control loop, evaluation of the output sensors to determine the existence of spurious structural vibrations and implementation of health monitoring which allows the system to recognize immediate faults as well as long term degradation are some of the problems considered in this article.

Another interesting feature of aeronautical systems described in [25] is the possibility of using thousands of sensors and actuators based on microelectronic mechanical systems. Faller and Schreck [25] and Rauch and Schaecter [26] believe that

one of the major advantages of neural networks is the tremendous computational speed achieved by massively parallel hardware.

1.3. Neuro – Sliding Mode Control

Sliding mode control is a well established control scheme for the control of nonlinear systems due its robustness to parameter uncertainties and external disturbances [27]-[29]. However, because it is necessary to know the system dynamics for the calculation of the control input, it is hard to apply to systems that has very complex dynamics or the ones whose dynamics are not very well known. For these systems a control scheme that does not need full information of system dynamics is needed. In the literature there exist some approximate solutions [30] but their stability is not proven yet. The controller should also adapt itself to large parameter variations and unexpected external disturbances. Merging a well established control structure like sliding mode control with intelligent algorithms appeared to be a good idea and many researchers published various control structures based on this. A comprehensive historical investigation and a literature survey can be found in [31]. To mention some, in [32], Jezernic, Rodic, Safaric and Curk applied the idea on a 3.D.O.F PUMA type DD – robot system. They used continuous sliding mode theory to establish a robust control scheme. To avoid the chattering effect, they estimated the equivalent control and used this estimation in the sliding mode control algorithm. The estimation of the equivalent control was done using an online neural network estimator. In [33], Rodie, Jezernic, Sabanovic and Safarie used a sliding-mode based learning algorithm for robust accurate tracking of a single axis DD robotic manipulator driven with an induction motor. In another work, [34], Fang, Y., Chow; T.W.S. and Li, X.D. proposed a control system on the basis of a discrete Lyapunov function. Part of the equivalent control is estimated by a recurrent neural network (RNN) and a real-time iterative learning algorithm is developed and used to train the RNN. They also proved the stability of the system by showing that the learning error converges to zero.

In this thesis, the controller proposed is designed based on the minimization of a cost function that is obtained by satisfying Lyapunov stability criteria. This cost function is the same cost function used in [33] but different from their approach, the aim

is not calculating the equivalent control but computing the whole control signal using the minimization process. Also, the neural network used is a one layer neural network which holds the linearity of parameters. The major contribution in this work is that, the stability of the system is proven by investigating the shape of the cost function and showing that there is no danger of sticking to local minima.

In chapter 2, the proposed control scheme is presented for single input single output (SISO) systems. The stability is proven and the performance of the controller is verified by simulations and experiments. In chapter 3, the theory is generalized to enclose a class of multiple input multiple output systems (MIMO). Again the stability is proven and simulation and experimental results showing the controller performance is presented. In the last chapter, chapter 4, the conclusion of the overall work is given.

2. NEURO-SLIDING MODE CONTROLLER DESIGN FOR SISO SYSTEMS

2.1. Problem Definition

Consider an n^{th} order, SISO, nonlinear system

$$\dot{x} = f(x) + B(x)u + d \tag{2.1}$$

where $x = \left[y,..., y^{n-1}\right]^T \in \Re^n$ is the state vector, $y \in \Re$ is the output of the system, $u \in \Re$ is the control input, $f(x) \in \Re^n$ is an unknown, continuous and bounded nonlinear function, $B(x) \in \Re^n$ is a known input vector whose elements are continuous and bounded and $d \in \Re^n$ is an unknown, bounded external disturbance. Also, $y^{(n)} = d^n y / dt^n$. It is assumed that the system is controllable. The goal is to find a control input such that, the system output y, will follow a desired trajectory y_d, while desired state vector is defined as $x_d = \left[y_d,..., y_d^{(n-1)}\right]^T$. The tracking error vector is defined as $e_t = \left[e,..., e^{(n-1)}\right]^T \in \Re^n$, where $e = y_d - y$. Sliding mode control is used as the control scheme.

The claim here is to design a neurocontroller for the system with partially known dynamics. Although the nonlinear function $f(x)$ is assumed to be unknown, the order of the system is assumed to be known and this is itself information about system dynamics. Because of this reason, the word *uncertain* is used instead of *unknown* to define the level of information about system dynamics.

2.2. Controller Design

To design a sliding mode controller for the defined problem, firstly, a sliding manifold must be chosen. Secondly, to ensure the existence of the sliding mode, a Lyapunov function must be determined in terms of the sliding function and the necessary control input must be computed that will fulfill the requirements of the Lyapunov stability criteria. In the following sections, these design steps is described in detail and the reason for the necessity of a neural network controller is given.

2.2.1. Selection of the Sliding Manifold

If the output of the system (2.1) y follows the desired trajectory y_d, it means that actually, system states, x, are following the desired states, x_d. This is equivalent to $e_t = x_d - x = 0$. Generally, the sliding mode manifold is selected as $\sigma = Ge_t = 0$. The $(1xn)$ row vector G is selected such that the sliding mode manifold takes the following form.

$$\sigma = \left(\frac{d}{dt} + C \right)^{n-1} e = 0 \tag{2.2}$$

where $e = y_d - y$. Explicitly the manifold can be written as

$$\binom{n-1}{0} e^{(n-1)} + \binom{n-1}{1} Ce^{(n-2)} + \binom{n-1}{2} C^2 e^{(n-3)} + \ldots + \binom{n-1}{n-1} C^{n-1} e = 0 \tag{2.3}$$

For example, if the system is of second order, the sliding manifold becomes

$$\dot{e} + Ce = 0 \tag{2.4}$$

which is a straight line in e vs $\dot{e}$ plane. The selection of the positive constant C, determines the speed of the system. Once the desired speed is decided, the sliding manifold can be designed by selecting an appropriate C value.

2.2.2 Lyapunov Function Selection and Finding the Necessary Control Input

According to the Lyapunov stability criteria, if a function $V(\sigma)$ is positive definite and the time derivative of it is negative definite and the function vanishes only when $\sigma = 0$, then σ has to go to zero. To achieve these requirements, the following Lyapunov function is selected.

$$V = \frac{1}{2}\sigma^2 \tag{2.5}$$

The function selected is positive definite and it vanishes only when $\sigma = 0$. The only requirement left is its time derivative be negative definite. Choosing its time derivative as

$$\dot{V} = -D\sigma^2 \tag{2.6}$$

where, D is positive constant, restricts the derivative to be negative definite. Substituting (2.5) into (2.6), the following equation is obtained.

$$\dot{\sigma}\sigma = -D\sigma^2 \tag{2.7}$$

where D is a positive constant. Going one step further,

$$\sigma\left(\dot{\sigma} + D\sigma\right) = 0\,. \tag{2.8}$$

Hence, for the Lyapunov stability criteria to be held,

$$\dot{\sigma} + D\sigma = 0 \tag{2.9}$$

must be satisfied for $\sigma \neq 0$. By substituting (2.2) into (2.9) and using (2.1), the necessary control input can be found as

$$u = (GB)^{-1}(Gx_d - Gf(x) + Gd + D\sigma). \tag{2.10}$$

This control input does not result in a VSS structure because the system reaches the manifold asymptotically instead of hitting the manifold in finite time. This is called pseudo sliding mode control.

It is clearly seen that, to compute the necessary control input, full information about system dynamics and disturbance is needed. In most real – life engineering problems, this information can not be found or can be obtained partially. Some approximations can be done (e.g. see [33]), but the resulting control input may not give the desired performance.

A solution to this problem can be using a neural network structure that minimizes the function $\dot{\sigma} + D\sigma$ and eventually makes it zero, without need for the information about the system dynamics or the disturbance but using only the information about the error, the order of the system and the vector B.

2.2.2.1 Structure of the Controller

The proposed structure of the neurocontroller is shown in Fig. 2.1. There is an input layer which has linear activation (i.e. whatever comes in directly goes out without any manipulation), an output node that also has a linear activation function. w_i's are adaptable weights that are updated during the operation. Overall, this structure is called an ADALINE –adaptive linear element-, the simplest type of neural network structures. The output of the controller is the weighted sum of the components of the error vector e_t, which can be written as

$$u = \sum_{i=1}^{n} w_i e^{(i-1)} . \tag{2.11}$$

As stated above, the goal is to push the function $\dot{\sigma} + D\sigma$ to zero. To achieve this goal the error function

$$E = \frac{1}{2}(\dot{\sigma} + D\sigma)^2 \tag{2.12}$$

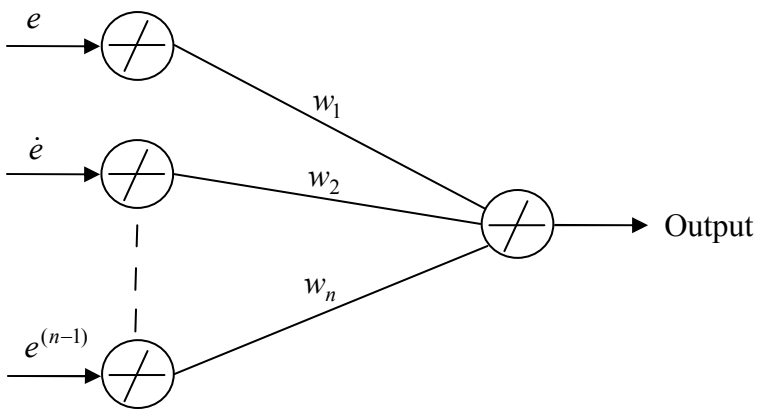

Figure 2.1. Structure of the neurocontroller

is introduced to the network and weights are updated accordingly. The weight update algorithm is described in the next section.

2.2.2.2 Weight Update Algorithm

Weights are updated using simple gradient descent approach –in continuous form- or backpropagation:

$$\dot{w}_i = -\eta \frac{\partial E}{\partial w_i} \tag{2.13}$$

where, η is the learning constant, generally chosen between 0 and 1. To compute the weight updates, the derivative of the error function E w.r.t. w_i should be found. Using the chain rule, the derivative can be written as

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial w_i}. \tag{2.14}$$

Substituting (2.12) into (2.14) and taking the derivatives, the following equation is obtained:

$$\frac{\partial E}{\partial w_i} = \left(\dot{\sigma} + D\sigma\right)\frac{\partial\left(\dot{\sigma}\right)}{\partial u} e^{(i-1)} \tag{2.15}$$

Substituting $\sigma = Ge_t$ into (2.15),

$$\frac{\partial E}{\partial w_i} = \left(\dot{\sigma} + D\sigma\right)\frac{\partial\left(G\dot{x}_d - G\dot{x}\right)}{\partial u}e^{(i-1)} \tag{2.16}$$

hence,

$$\frac{\partial E}{\partial w_i} = -\left(\dot{\sigma} + D\sigma\right)GB(x)e^{(i-1)} \tag{2.17}$$

is obtained. As a result, the weight update algorithm can be stated as

$$\dot{w}_i = \left(\dot{\sigma} + D\sigma\right)\eta GB(x)e^{(i-1)}. \tag{2.18}$$

In this form the update mechanism is continuous. For the computer implementation, the discrete version is used:

$$w_i\left[(k+1)h\right] = w_i\left[(kh)\right] + \left(\dot{\sigma}\left[(kh)\right] + D\sigma\left[(kh)\right]\right)\eta hGB(x)e^{(i-1)}\left[(kh)\right] \tag{2.19}$$

where h refers to the sampling interval. For a system with constant B, the term ηhGB is a constant scalar, which can be denoted by $\overline{\eta}$ to make the expression (2.19) neater:

$$w_i\left[(kh+h)\right] = w_i\left[(kh)\right] + \overline{\eta}\left(\dot{\sigma} + D\sigma\right)e^{(i-1)}\left[(kh)\right] \tag{2.20}$$

2.2.2.3 Disturbance Rejection

The goal of the neurocontroller is to push (2.12) to zero. To see how this is achieved by the control action, error function is rewritten by substituting the sliding function $\sigma = Ge_t$ and using (2.1):

$$E = \frac{1}{2}\left[G\left(\dot{x}_d - f(x) - Bu - d\right) + DGe_t\right]^2 \tag{2.21}$$

Substituting (2.11) into (2.21),

$$E = \frac{1}{2}\left[G\left(\dot{x}_d - f(x) - B\sum_{i=1}^{n} w_i e^{(i-1)} - d \right) + DGe_t \right]^2 \tag{2.22}$$

From (2.22), it is clear that as e approaches zero, the weights w_i has to increase infinitely to compensate for the disturbance and the other terms in the equation. This decreases the controller performance and results in a sluggish system response. To avoid this, another term is added to the controller structure that is independent of the error e. The structure of the resulting controller is shown in Fig. 2.2.

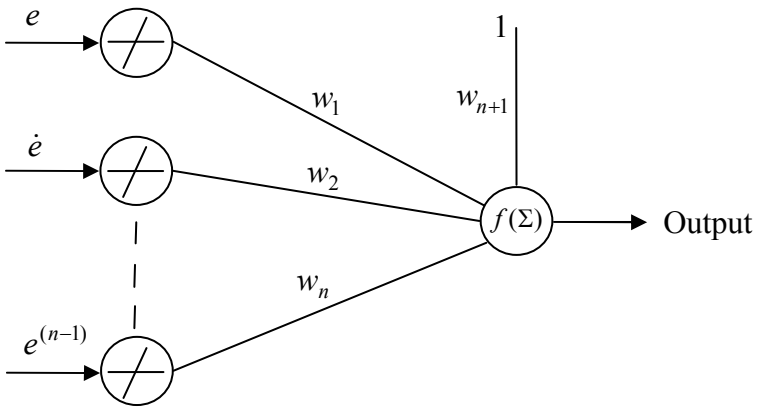

Figure 2.2. Structure of the upgraded neurocontroller

The new term, $1w_{n+1}$, can be seen as a bias for the ADALINE. After adding this term, the error function takes the following form.

$$E = \frac{1}{2}\left[G\left(\dot{x}_d - f(x) - B\sum_{i=1}^{n} w_i e^{(i-1)} - B1w_{n+1} - d \right) + DGe_t \right]^2 \tag{2.23}$$

(2.23) shows that the term, $1w_{n+1}$ is not multiplied by the error e, and hence can compensate the disturbance term without increasing indefinitely. The weight update of this new term is given as

$$w_{n+1}\left[(k+1)h\right] = w_{n+1}\left[(kh)\right] + \overline{\eta}\left(\dot{\sigma}\left[(kh)\right] + D\sigma\left[(kh)\right]\right) \tag{2.24}$$

2.2.2.4 Stability Proof

It is a known fact that one of the biggest problems in backpropagation weight update algorithm is sticking to local minima. In this section, it is shown that, using the proposed control algorithm, the error function is pushed to zero without sticking to any local minima, thus forcing the components of the error vector e_t to the sliding manifold. Then, the stability proof in terms of Lyapunov is presented.

2.2.2.4.1 The Shape of the Error Surface

If a function's second derivative does not change sign with respect to a variable, then the function does not have a change in the curvature sign through that variable, which means that the function does not have a local minimum through that variable. Examining the second derivative of the error function (2.12) w.r.t. its variables (weights), the following equation holds.

$$\frac{d^2E}{dw_i^2} = (GBe^{(i-1)})^2 \tag{2.25}$$

This result, (2.25), shows that the sign of the curvature of the error surface (2.9) is always positive; hence, there are no local minima. This indicates that, with a proper selection of the learning constant, the proposed network is capable of minimizing the function (2.9) up to its global minimum, which is nothing but zero. Thus, the tracking error vector has to converge to zero. Also, since η is a constant, G is a constant vector and $B(x)$ is bounded, weight update algorithm (2.18) shows that weights converge to a finite value at steady state. A finite value for the weights at steady state results in a bounded control input (2.11). As a result, all the signals in the control system are bounded.

2.2.2.4.2 Proof Based on Lyapunov

Let the Lyapunov function candidate be

$$V = \frac{1}{2}\left(\dot{\sigma} + D\sigma\right)^2 . \tag{2.26}$$

This is exactly the same function that is used for the error function (2.12). It is easily seen that $V > 0$ for $\dot{\sigma} + D\sigma \neq 0$. Taking the time derivative of V, one obtains the following equation.

$$\dot{V} = \frac{\partial V}{\partial w_i}\frac{dw_i}{dt} \tag{2.27}$$

Substituting (2.13) into (2.27) and using the identity $E = V$, gives the following expression.

$$\dot{V} = -\eta\left(\frac{\partial V}{\partial w_i}\right)^2 \tag{2.28}$$

In the expression (2.28), η is a positive scalar, thus $\dot{V} \leq 0$. However, since it is proven that the error surface – hence, the Lyapunov function – does not have any local minima, the expression $\partial V / \partial w_{ij}$ becomes zero only at the global minimum, which is zero. So, $\dot{V} < 0$ for $\dot{\sigma} + D\sigma \neq 0$. This proves that Lyapunov function converges to zero and the requirement (2.9) is satisfied, resulting a stable system.

2.3. Simulation Results

Simulations are carried on the model of a single axis, toothed belt, linear servo system, which is now in use at Sabanci University, Mechatronics Laboratory. This system is driven by an electrical motor. The belt can carry different loads by the help of

a carriage. Thus, the load carried by the belt, the friction forces between the carriage and the rail, and the friction on the motor bearings affect the motor as disturbances. The aim is to control the motor under different loading conditions, without the information about the load. The controller should be able to suppress the oscillations caused by the elastic belt, the system response should be fast enough and the steady state error should be zero.

The physical and mathematical model of the system together with open loop response is given in section 2.3.1, and the simulation results are presented in section 2.3.2. Also, the experimental results on the same system are presented in section 2.4.

2.3.1 The Model of the System

The system consists of the electrical servomotor connected to the toothed-belt driven mass *m,* referred as the load. The physical model of the system is presented in Fig. 2.4.

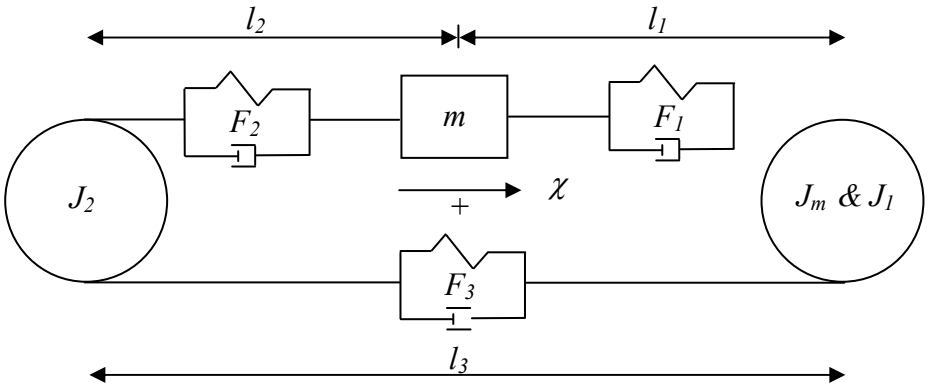

Figure 2.4. The physical model of the timing-belt driven system

The structure depicts the forces in each part of the timing belt along with the inertia of the motor and the pulleys.

The structure has the active torque developed by the motor, the inertia of which is given by J_m, six forces developed by the different part of the timing-belt and lumped motor and load disturbances. In each part of the timing belt the belt elasticity and belt friction force are developed. The belt elasticity forces are dependent on the stretch of the belt and the belt friction forces are dependent on the derivative of the belt stretch. In

general the behavior of the system could be described as a sixth order dynamical system where the motion of the pulley 2, the inertia of which is given by J_2, is defined by the forces F_3 and F_2.

The following equations (2.29) – (2.31) describe the mathematical model of the system.

$$J\frac{d^2\theta_1}{dt^2} = \mathrm{T} - T_L - r(F_1 - F_3) \tag{2.29}$$

$$J_2\frac{d^2\theta_2}{dt^2} = r(F_2 - F_3) \tag{2.30}$$

$$m\frac{d^2\chi}{dt^2} = -F_L + (F_1 + F_2) \tag{2.31}$$

The meanings of the variables and parameters are given in Table 2.1.

Table 2.1. Variables used in the model of the linear drive

θ_1	The angular position on the pulley driven by the servomotor
θ_2	The angular position on the undriven pulley
χ	The longitudinal position of the load
v	The longitudinal velocity of the load
T	The torque developed by the servomotor
T_L	The friction torque at the servomotor side
F_1, F_2, F_3	The forces in the different part of the belt
F_L	The friction force at the load side
J	Equivalent moment of inertia on the motor side
m	Mass of the load

The following assumptions could be made for the purpose of analyzing the system, presented in Fig. 2.4:

Inertia of pulleys 1 and 2 are small in comparison with the motor and the load inertia. Because of this, dynamics of the system could be simplified so that the resulting system is described by the dynamics of motor and the dynamics of the load mass subject to the action of the timing belt resulting force – representing the influence of all three parts of the timing belt. The simplified structure is represented in Fig. 2.5.

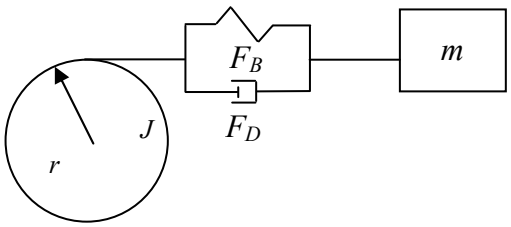

Figure 2.5. The simplified structure of the system

The following assumptions could be made for the purpose of analyzing the system presented in Fig. 2.5:

- Servomotor is assumed to be operating in the current control mode with high dynamics torque response on the motor axis with negligible time constant;
- Connection of the motor to driving pulley is assumed to be rigid;
- The inertia of the pulleys on both sides of the load (motor side and the belt-driven side) is negligible in comparison with inertia of the other components of the system;
- The elasticity of the toothed belt is treated as a nonlinear spring

Simplified description of the motor-belt-mass system can be modeled as a two-mass system with -nonlinear spring connection. Under these assumptions the motion of the system in Fig. 2.4 can be described by the following set of differential equations (2.32-2.36):

$$\frac{d\theta}{dt} = \omega \tag{2.32}$$

$$\frac{d\omega}{dt} = \frac{\mathrm{T}}{J} - \frac{T_L}{J} - \frac{r}{JG}\left(F_B + F_D\right) \tag{2.33}$$

$$\frac{d\chi}{dt} = v \tag{2.34}$$

$$\frac{dv}{dt} = \frac{F_B + F_D}{m} - \frac{F_L}{m} \tag{2.35}$$

$$F_{Belt} = F_B + F_D \tag{2.36}$$

The variables used in the simplified model (2.32-2.36) is given in Table 2.2.

Table 2.2 Variables used in the simplified model of linear drive

θ	The angular position on the servomotor's shaft
ω	The angular velocity of the servomotor's shaft
F_B	The belt elasticity force proportional to the belt stretch
F_D	The damping force developed by the belt proportional to the derivative of stretch
F_L	The friction force at the load side
r	Radius of pulleys

For the simpler treatment of the system as a whole in further analysis the equations (2.32) and (2.33) will be rewritten so to represent the motion of the belt on the pulley attached to the motor using transformation $x_m = r\theta$ one can write

$$\frac{dx_m}{dt} = v_m \tag{2.37}$$

$$\frac{dv_m}{dt} = \frac{F_{mot}}{m_{mot}} - \frac{F_B(x_m, x) + F_D(v_m, v)}{m_{mot}} - \frac{F_{Lmot}}{m_{mot}} \tag{2.38}$$

where meaning of the new variables is as follows:

$$F_{mot} = \frac{1}{r}K_T i = \frac{1}{r}\mathrm{T},\ F_{Lmot} = \frac{1}{r}T_L,\ m_{mot} = \left(\frac{1}{r}\right)^2 J \tag{2.39}$$

The behavior of the motor with a pulse servomotor reference current is shown in Fig. 2.6. In these pictures the transients are given for FESTO linear drive DGEL25-1500-ZR-KF equipped with servomotor MTR-AC-70-3S-AA. These results are obtained through experiments.

The disturbance current shown in Fig. 2.6 is the total disturbance – belt force and frictions – that affect the motor.

In the next section, the design of the neurocontroller for the position control of the motor used in the servo system is presented.

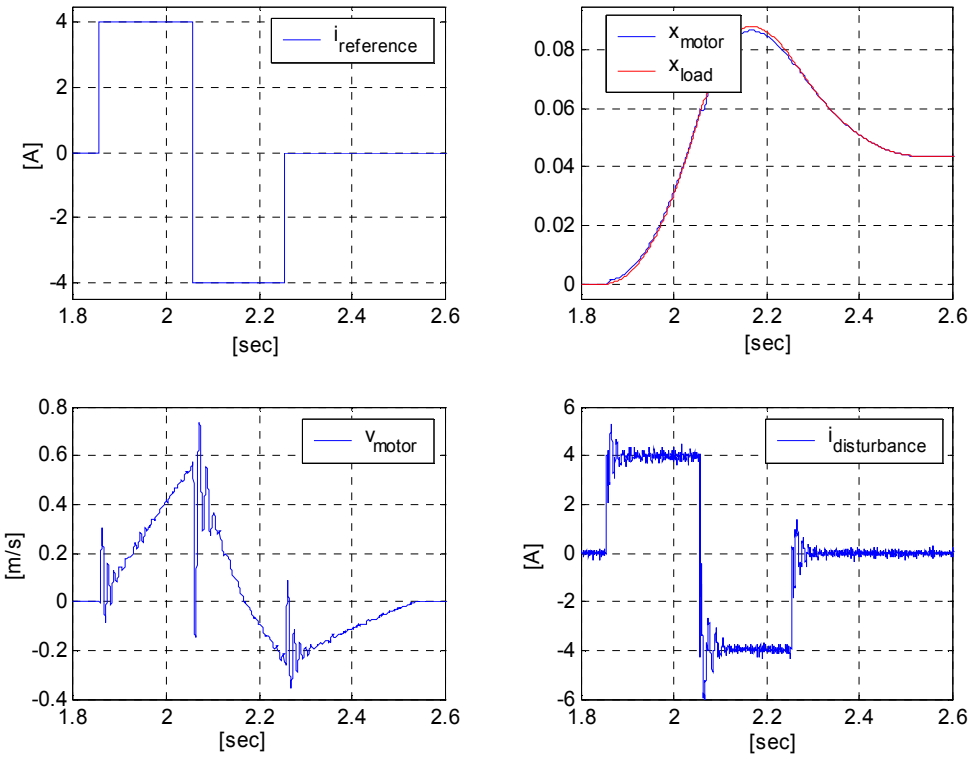

Figure 2.6. Open loop response of the system

2.3.2 Neurocontroller Design

As stated in the previous sections, the aim of the controller to be designed is to make the position control of the motor used in the linear servo system. In the system, the load on the belt and the friction on the slider, thus the belt force, is assumed to be unknown. Also, the friction at the motor bearings is not known. The only thing that is known about the system is that, when the belt force and the friction are taken as disturbance to the motor, the system is a second order system and the torque constant about the motor is known. Actually, because the learning rate is found generally by experience, the torque constant is not needed to be known precisely. It is shown by simulation results that the system is insensitive to parameter changes caused by say, adding extra load on the belt. In the end, the resulting controller is robust and stable.

Rewriting (2.37) and (2.38) by taking belt force and the friction acting on the motor as disturbance $D(t)$, the following equations are obtained.

$$\frac{dx_m}{dt} = v_m \tag{2.40}$$

$$\frac{dv_m}{dt} = \frac{F_{mot}}{m_{mot}} - \frac{D(t)}{m_{mot}} \tag{2.41}$$

Substituting (2.39) into (2.41)

$$\frac{dv_m}{dt} = i\frac{K_t r}{J} - \frac{D(t)}{m_{mot}} \tag{2.42}$$

and representing the whole system in state space, gives the following equation.

$$\begin{bmatrix} \dot{x}_m \\ \dot{v}_m \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_m \\ v_m \end{bmatrix} + \begin{bmatrix} 0 \\ K_t r / J \end{bmatrix} i + \begin{bmatrix} 0 \\ m_{mot} \end{bmatrix} D(t) \tag{2.43}$$

After having the state space representation, the next step is to decide a sliding manifold and obtain the weight update algorithm. Using (2.3), for second order systems, the sliding manifold is selected as

$$\sigma = \dot{e} + Ce \tag{2.44}$$

where $e = (e_m)_d - e_m$ and C is a positive constant. The neural network's job is to minimize the error function $E = (1/2)(\dot{\sigma} + D\sigma)^2$. For the second order system, the network structure is shown in Figure 2.7.

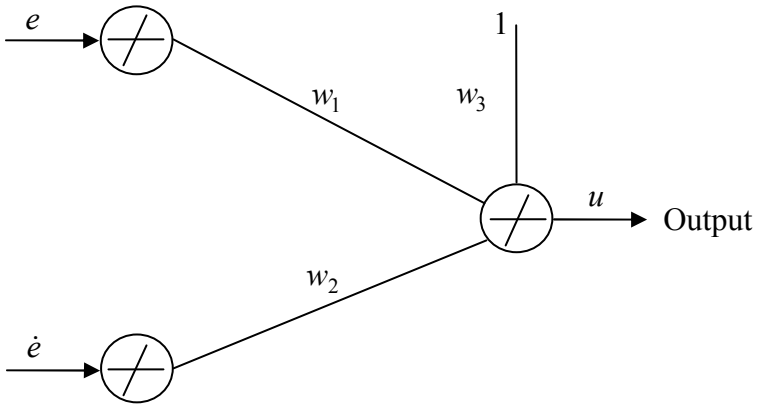

Figure 2.7. Controller structure for second order system

The control input, u , then becomes

$$u = w_1 e + w_2 \dot{e} + w_3 \tag{2.45}$$

Following the procedure presented at (2.13)-(2.19), the weight updates are found as

$$w_1(kh+h) = w_1(kh) + \eta h \frac{K_t r}{J}\left(\dot{\sigma}(kh) + D\sigma(kh)\right)e(kh) \tag{2.46}$$

$$w_2(kh+h) = w_2(kh) + \eta h \frac{K_t r}{J}\left(\dot{\sigma}(kh) + D\sigma(kh)\right)\dot{e}(kh) \tag{2.47}$$

$$w_3(kh+h) = w_3(kh) + \eta \frac{K_t r}{J}\left(\dot{\sigma}(kh) + D\sigma(kh)\right) \tag{2.48}$$

Applying these weight updates, the controller becomes stable and robust to external disturbances and parameter variations. In the following section this is verified by the simulation results.

2.3.3 Simulation Results

In the simulations, the controller parameters are set as in the following.

$$C = 40\,,\ D = 90\,,\ \eta = 0.00005 \tag{2.49}$$

Figures (2.8) - (2.13) show the response of the system to a position reference input which is in the form of a smooth s-shaped curve realized as a linear segment preceded and followed by parabolic segments. This corresponds to a trapezoidal velocity reference curve. In Fig. 2.8, the reference position and the actual motor response is presented. As it is seen from the graph, the motor tracks its reference quite well, that it is hard to distinguish the two lines. The error in this tracking is shown in Fig. 2.9. The transient error is exceptionally small and steady state error is zero. In Fig. 2.10 the control input is presented. As seen from the figure, control input is smooth and has acceptable bounds. Figures 2.11 - 2.13 show the time evolution of the network weights. The initial values are taken as zero for all the three parameters. After updating

themselves, they converge to a finite value, which indicates that the weight update algorithm is stable.

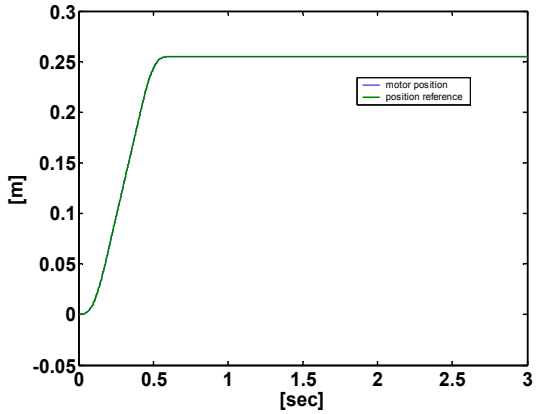

Figure 2.8. Position tracking

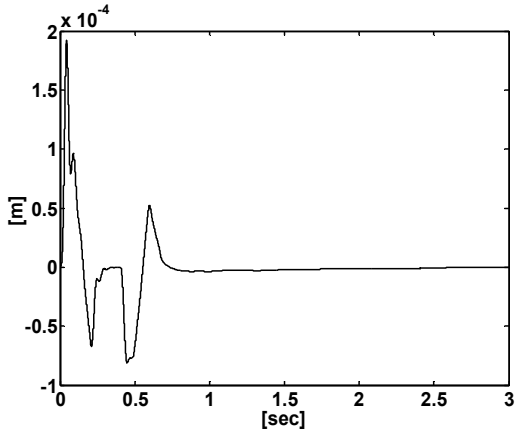

Figure 2.9. Position tracking error

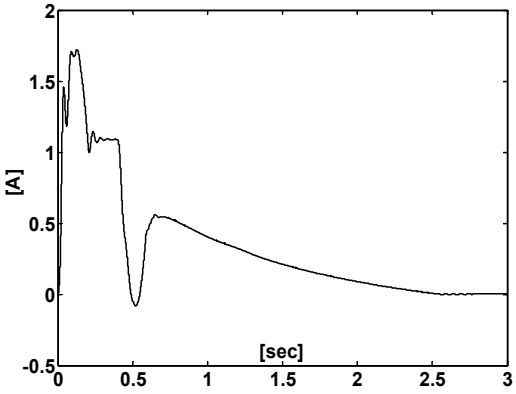

Figure 2.10. The control input

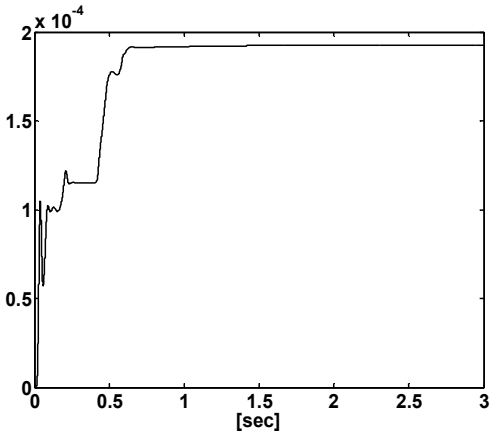

Figure 2.11. Time evolution of w1

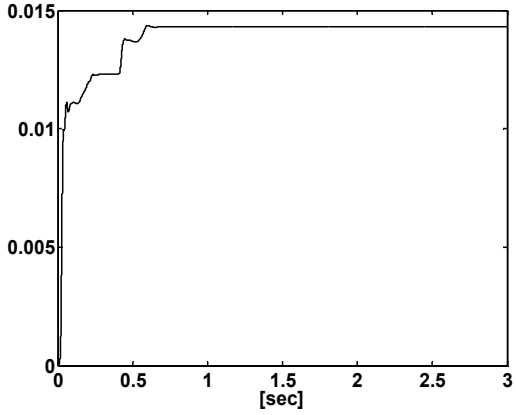

Figure 2.12. Time evolution of w2

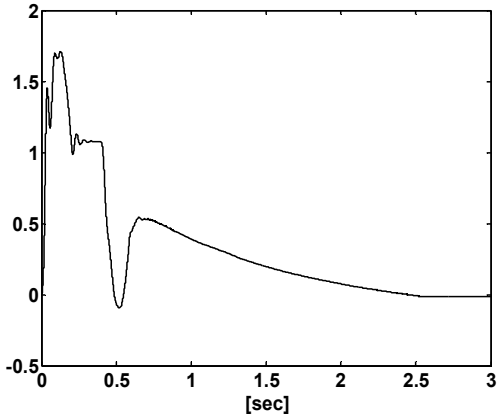

Figure 2.13. Time evolution of w3

When the torque applied to the motor is converted to force, it is found that the force created by the motor changes between -5 and 55 Newtons during this operation. To verify that this neurocontroller is robust to external disturbances, an external disturbance of 30 Newtons step input is applied to the motor at time t = 0.5 seconds. Fig. 2.14 shows the force created by the motor before applying this step disturbance and Fig. 2.15 shows the total force created by the motor and the step disturbance. Figures reveal that although a step disturbance that is approximately equal to half the force created by the motor is applied, the motor force is adjusted such that, after a very small transient, the net force converges to its previous value. This means that the position of the motor is not affected dramatically due to this huge disturbance and continues to follow its reference. This is shown by figures 2.16 and 2.17. Fig. 2.16 shows the position tracking of the motor. When compared with the position tracking curve Fig. 2.8, no difference can be recognized. To see the difference more clearly, Fig. 2.17 showing the position error after step disturbance is applied and Fig. 2.9, the position error before step disturbance is applied, should be compared. As it is seen, after a small deviation from the original one, the error converges to its original value in a short time. Overall, it can be said that the controller adapted its output in such a way that the system is not affected dramatically after a large step disturbance. This shows that controller is robust enough.

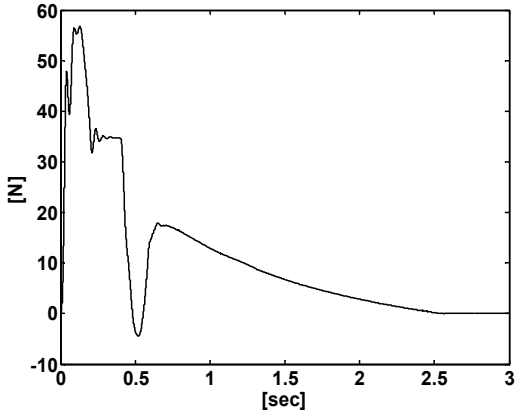

Figure 2.14. Force Created by the Motor, Before Step Disturbance is Applied

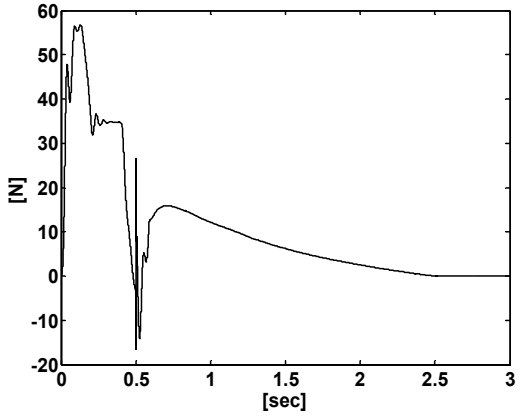

Figure 2.15. Net Force Acting on the Motor Shaft After Disturbance is Applied

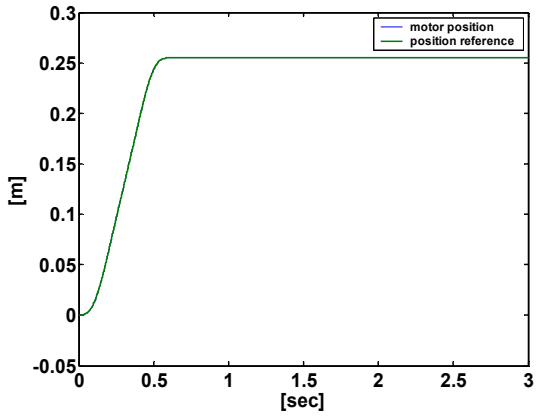

Figure 2.16. Position Tracking After Step Disturbance is Applied

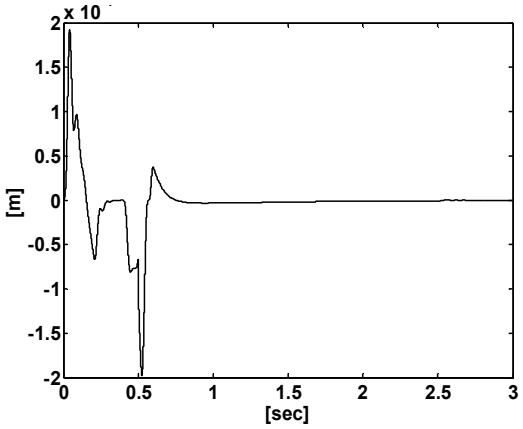

Figure 2.17. Position Tracking Error After Step Disturbance is Applied

Because the reference is smooth, the error and the derivative of the error start from zero and move in the vicinity of zero, hence, the reaching phase to the sliding manifold is hard to distinguish. Thus, giving a step position reference solves this problem and makes the states catching the manifold visible. Fig. 2.18 shows the position tracking for a step input and Fig. 2.19 shows the phase plane, when this step reference is applied to the system. In this case, the controller parameters are set as, $C = 10$, $D = 50$, and $\eta = 0.0001$. It is seen that a quick reaching phase without significant overshoot over the sliding line is achieved. When the line is reached, the sliding behavior is observed and the position error decays to zero with the error dynamics dictated by the parameter C of the controller (sliding line slope is $-10 = -C$). The simulation outputs are in harmony with the theoretical results. In the following section, the experimental results on the same linear drive system are presented.

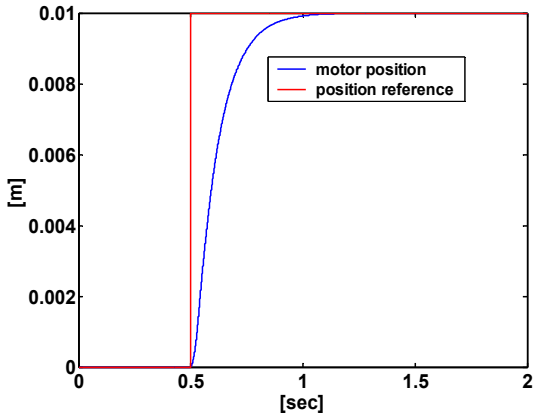

Figure 2.18. Position Tracking

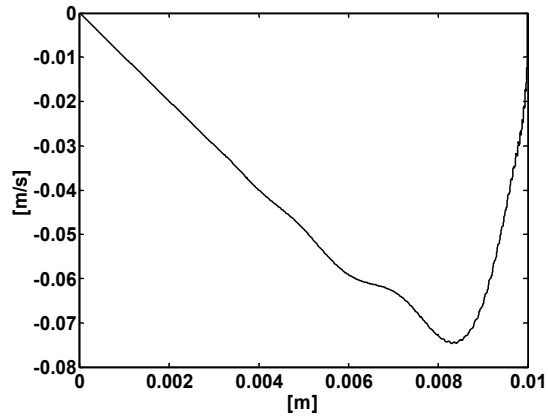

Figure 2.19. Phase Plane

2.4. Experimental Results on the Linear Drive

In this section, the experimental verification of the theory presented in the previous sections is presented using the same plant the model of which was introduced in section 2.3.1, that is the linear drive. The controller performance is tested by several reference inputs such as sigmoid, step, repeated pulse and sine. For the experiments the controller parameters used are $D = 200$, $C = 10$, $\eta = 0.00001$.

Fig. 2.20 – 2.26 show the response of the system to a smooth sigmoid position reference. From Fig. 2.20, which shows the position tracking of the motor, it is very hard to distinguish the reference and actual motor positions. Fig. 2.21 shows the error in this tracking. As it is seen, the transient error makes a jump in the very beginning of the motion and then decreases dramatically during the tracking. In the end, steady state error reaches its theoretical limit, which is set by the position measurement device. In Fig. 2.22, the velocity tracking of the motor is presented. This figure also shows that after a deviation from the reference, the velocity catches its reference and tracks it. The initial deviation can be explained by the weights of the network starting from zero. After they reach certain values in a short time, the system behaves as desired. In Fig. 2.23, the control signal produced is shown. It is seen that the control signal is sufficiently smooth. Fig. 2.24 – 2.26 indicate that after having a transient period, the weights are converging to a finite value, hence the update algorithm is stable.

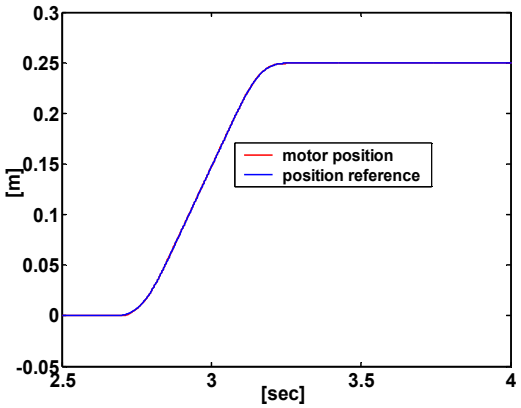

Figure 2.20. Position tracking

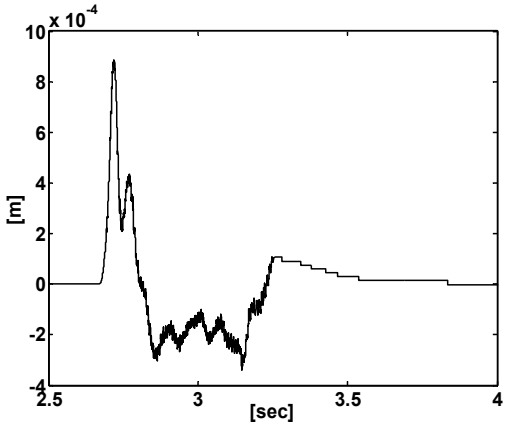

Figure 2.21. Position tracking error

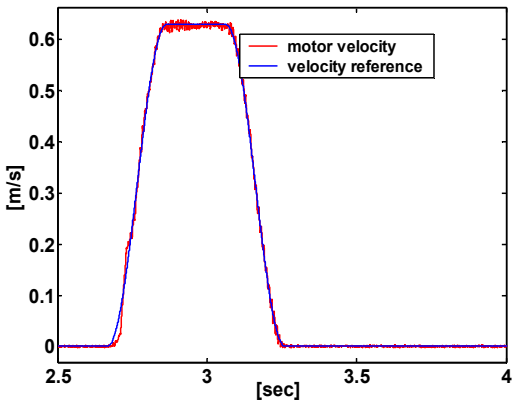

Figure 2.22. Velocity tracking

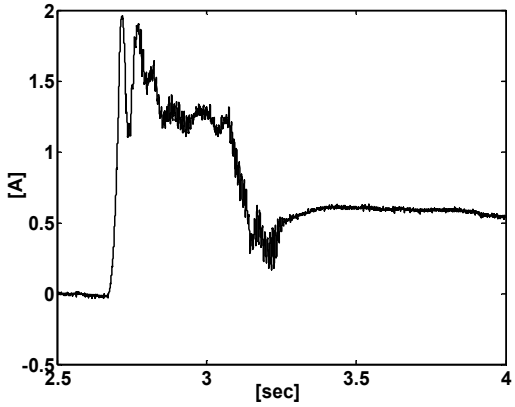

Figure 2.23. The control input

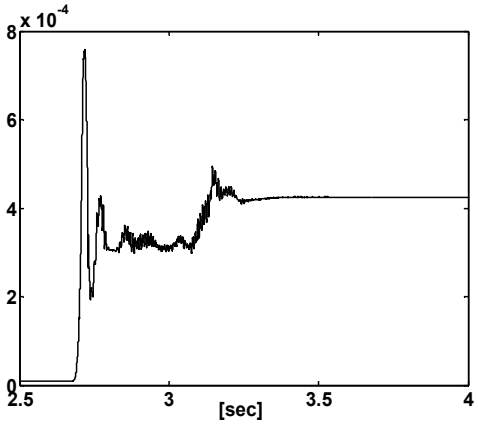

Figure 2.24. Time evolution of w1

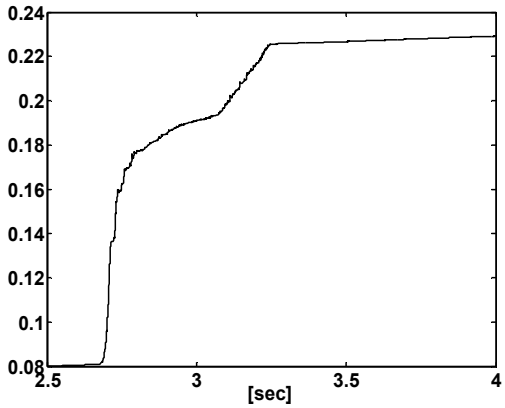

Figure 2.25. Time evolution of w2

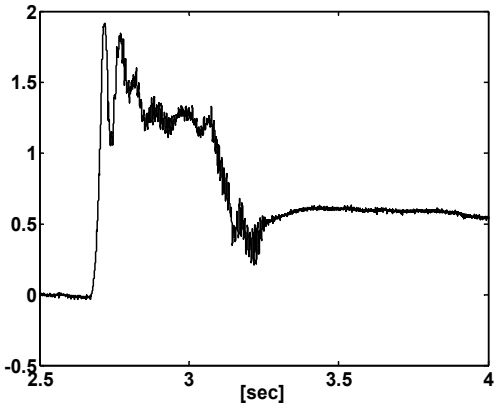

Figure 2.26. Time evolution of w3

As explained in the simulation section, to see the phase graph more clearly, a step reference input should be given to the system. Fig. 2.27- 2.28 show the position tracking

and phase plane for such an input. It is seen that a relatively quick reaching phase over the sliding line is achieved. When the line is reached, the sliding behavior is observed and the position error decays to zero with the error dynamics dictated by the parameter C of the controller (sliding line slope is $-10 = -C$).

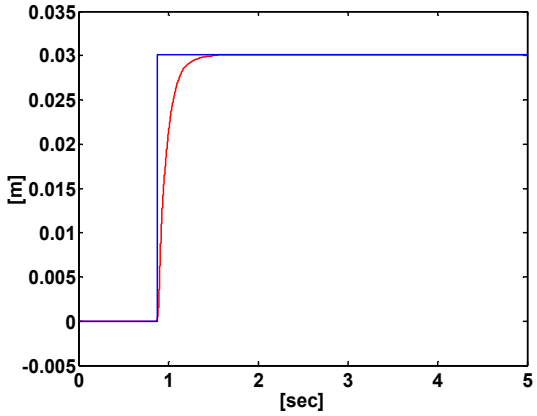

Figure 2.27. Position tracking

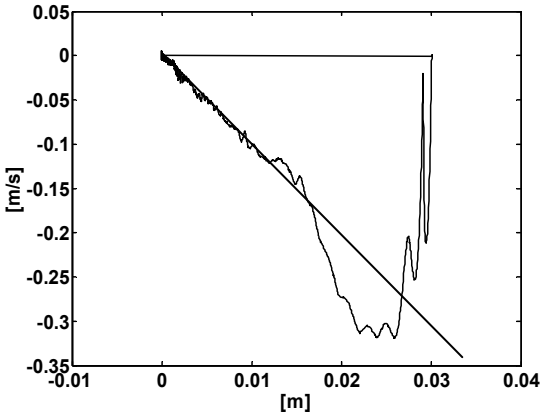

Figure 2.28. The phase plane

To see the tracking capabilities of the system, a repeated pulse position reference is given to the system and the response of the system is presented in Fig. 2.29 – 2.34. When the figures are examined, it is seen that the system tracks its reference successfully and the control input is smooth. However, Fig. 2.32 and 2.33 show that the weights w_1 and w_2 grow while the system is following its trajectory. This may create dangerous outcomes on the system such as deterioration of the stability due to control input built up over time.

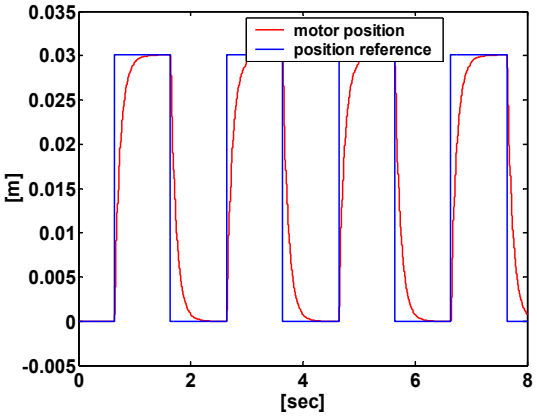

Figure 2.29. Position tracking

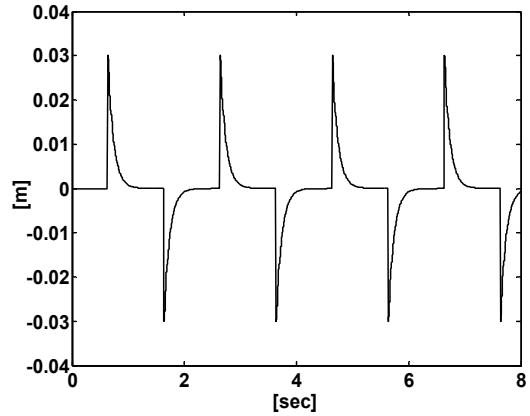

Figure 2.30. Position tracking error

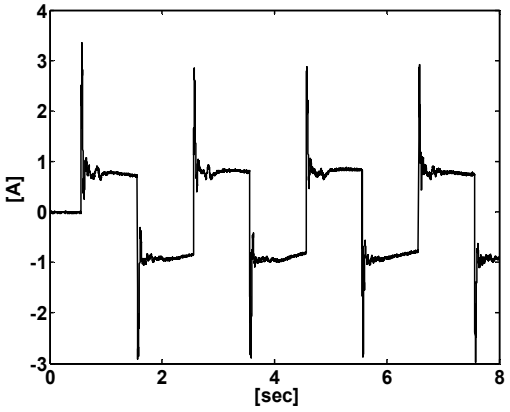

Figure 2.31. The control input

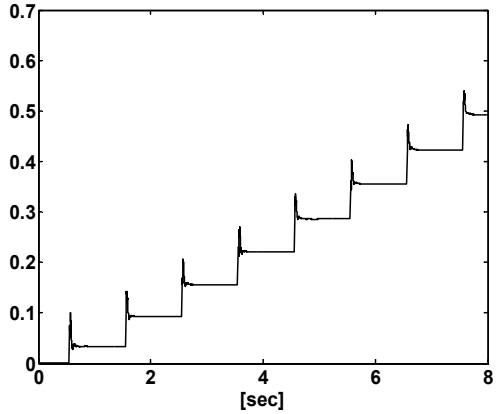

Figure 2.32. Time evolution of w1

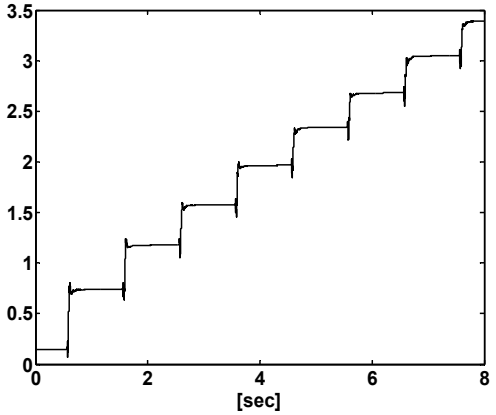

Figure 2.33. Time evolution of w2

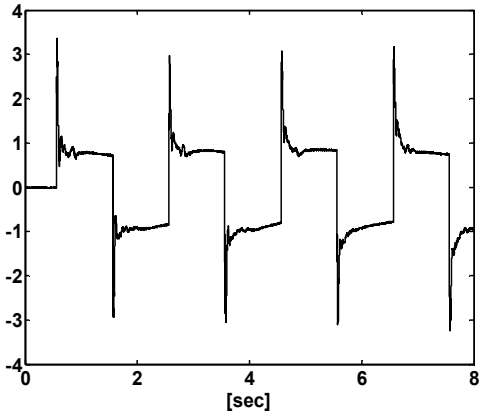

Figure 2.34. Time evolution of w3

The weights w_1 and w_2 are increasing for each pulse because the terms $(\dot{\sigma} + D\sigma)e$ and $(\dot{\sigma} + D\sigma)\dot{e}$ in (2.46) and (2.47) always make them bigger than what they were initially after a pulse period of updating, no matter the direction of reference is. However, when w_3 is examined, it is seen that it does not have such a problem. It behaves just the opposite for the pulse coming after the pulse it just has been exposed to. In the end it has an average of zero during the whole action. It shows us that if the sign of the weight update is determined by the term $(\dot{\sigma} + D\sigma)$, as in the w_3 case, weights does not built up but instead adjust themselves according to the direction of the reference – increasing or decreasing reference- so that in the end that have an average value during the tracking of a cyclic trajectory. It is logical then, to make the sign of the weight update for w_1 and w_2 same as the sign of the term $(\dot{\sigma} + D\sigma)$. To achieve this, the weight updates are changed as shown in the following.

$$w_1(k+1) = w_1(k) + \eta \frac{K_t r}{J} \left(\dot{\sigma}(k) + D\sigma(k)\right) \left\| e(k) \right\| \tag{2.50}$$

$$w_2(k+1) = w_2(k) + \eta \frac{K_t r}{J} \left(\dot{\sigma}(k) + D\sigma(k)\right) \left\| \dot{e}(k) \right\| \tag{2.51}$$

Using the absolute values of the e and $\dot{e}$, the sign of the weight updates are determined by the term $\left(\dot{\sigma} + D\sigma\right)$. The results of the experiment made using these upgraded weight updates are given in Fig. 2.35 – 2.40. It is seen that the upgraded update algorithm performs as well as the previous one without causing a weight built up.

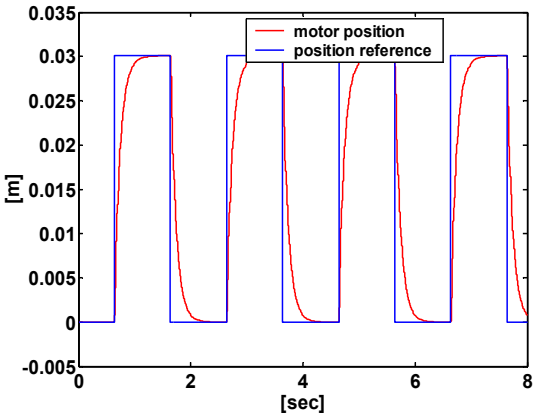

Figure 2.35. Position tracking

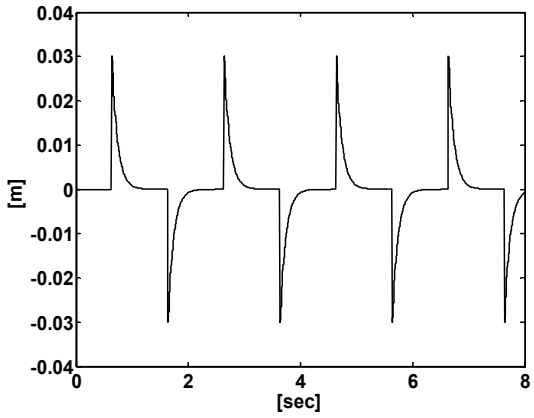

Figure 2.36. Position tracking error

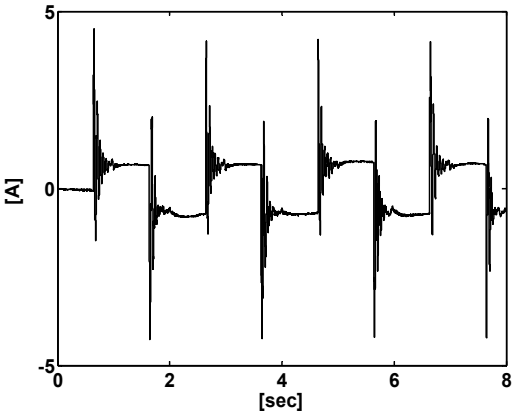

Figure 2.37. The control input

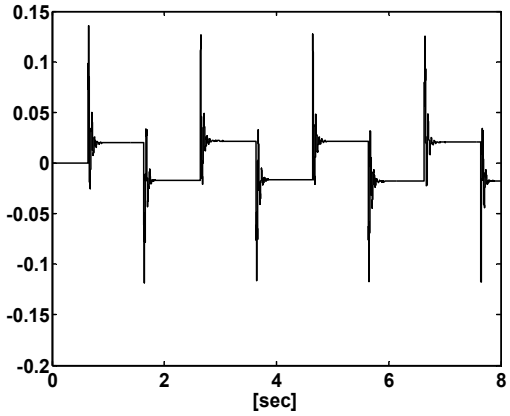

Figure 2.38. Time evolution of w1

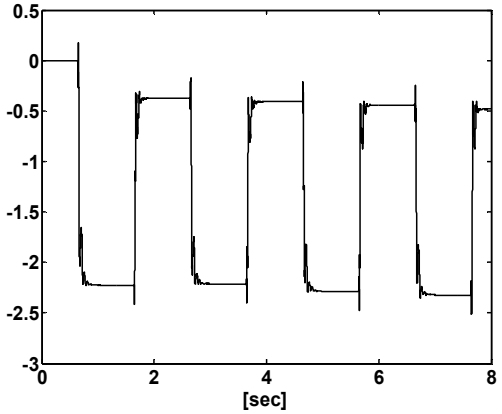

Figure 2.39. Time evolution of w2

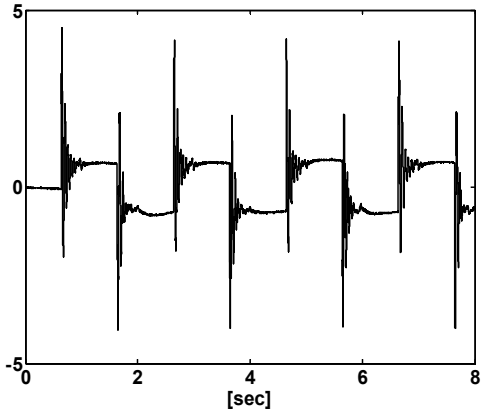

Figure 2.40. Time evolution of w3

2.5. Experimental Results on a Piezoelectric Actuator

Micromanipulator applications require controllers that can provide accurate position tracking performance in addition to robustness. These objectives are significantly compromised by the presence of backlash and Coulomb friction in the control plant, the effects of which are exaggerated in small scales. Since PZT stack actuators are monolithic and have no sliding or rolling parts, they exhibit no significant mechanical stiction or backlash. Additionally, a typical PZT stack actuator can perform step movements in nanometer resolutions with bandwidths on the order of a kilohertz. Consequently, PZT actuators are well suited for use as precision micro actuators for micro positioning devices.

An inherent non-linearity in piezo-ceramic actuators is hysteresis. This hysteresis non-linearity is usually 15-20% of the output thereby greatly reducing the performance of the actuators. Additionally, many attempts of modeling this behavior have been fruitless due to its peculiarities. In [35] and [36] attempts were made to model the voltage-to-displacement behavior of PZT actuators using Bond-Graph and Priesach models. These models proved effective, however, these models failed to explain the physical behavior of the actuators. In [37] and [38], models were made based on the physics of the actuators and these models proved to be effective in modeling the behavior of these actuators under different excitations. Additionally, they claim that the hysteresis behavior exists in the electrical domain of the actuator and is between voltage and charge. In Fig. 2.41, the actuators used in the experiments are shown.

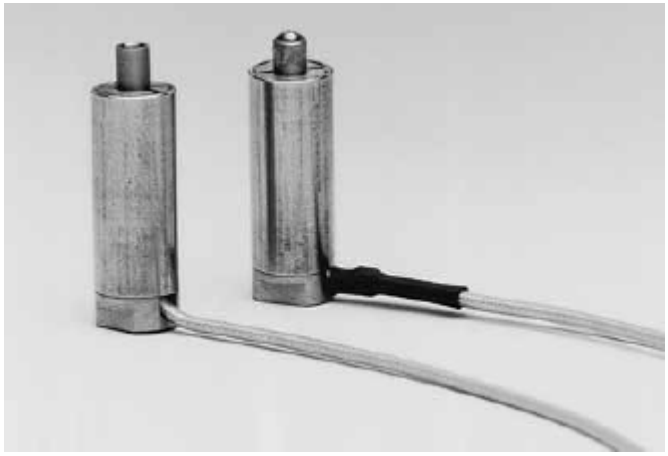

Fig. 2.41. Stack actuators used in the experiments

2.5.1. Structure of the piezoelectric actuator

Dielectric materials are insulators, thus, there is an electrical relation between electrical voltage and electrical charge. Piezoelectric materials are a special type of dielectric in the sense that, in piezoelectric materials, an externally applied force induces an electrical charge. Conversely, an applied electrical charge induces a force. The former effect is known as the piezoelectric effect and was discovered in 1880 by the Curies. The latter effect is the inverse piezoelectric effect. The word "piezo" derives from the Greek word "piezen," which means "to push." The effect was discovered when a pushing force or, in other words pressure, was applied to the material. In the beginning, both pressure electricity and piezoelectricity were used to describe the same phenomenon. Besides the piezoelectric and inverse piezoelectric effect, we have the already mentioned electrical relation between voltage and charge, and a mechanical relation between force and elongation.

In naturally occurring piezoelectric materials, such as quartz, the (inverse) piezoelectric effect is too small to be of practical use. Man-made piezoelectric polycrystalline ceramics are much more suitable for actuator purposes because the useful properties, such as maximum elongation, can be influenced by the proper mixture of ingredients. A disadvantage of man-made piezoelectric ceramics is that a hysteresis effect is encountered between electrical voltage and electrical charge. The piezoelectric effect (or the piezo effect for short) and the hysteresis effect play an important role in the dynamical behavior of these actuators.

The fundamental component of a PZT stack actuator is a wafer of piezoelectric material sandwiched between two electrodes. Prior to fabrication, the wafer is polarized uniaxially along its thickness, and thus exhibits significant piezoelectric effect in this direction only. A typical PZT stack actuator is formed by assembling several of the wafer elements in series mechanically and connecting the electrodes so that the wafers are parallel electrically, as illustrated in Fig. 2.42. The nominal quasi-static behavior of a PZT stack actuator is a steady-state output displacement that is monotonically related to the voltage input.

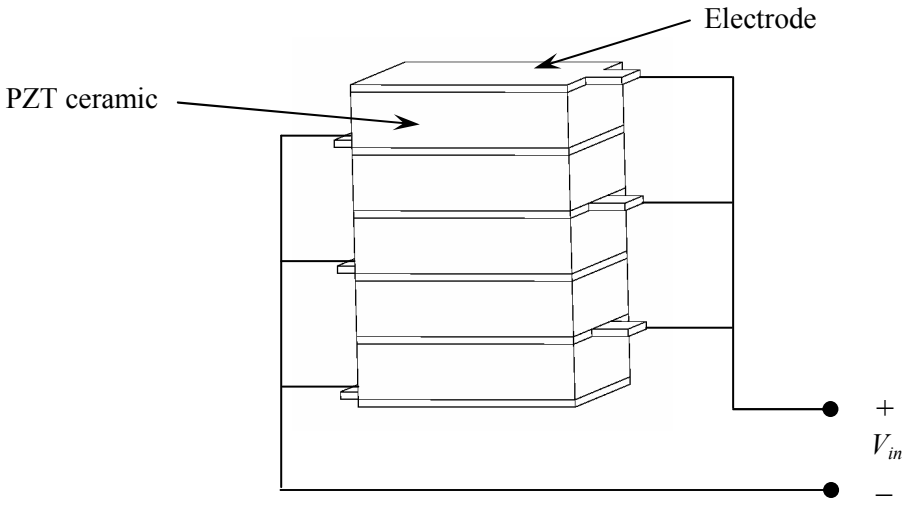

Figure 2.42. Illustration of a PZT stack actuator

2.5.2. Model of the piezoelectric actuator

In this section, a few models that exist for PZT actuators is presented.

2.5.2.1 Earlier Formulations

The most widely recognized description of piezoelectric ceramic behavior published by a standards committee of the IEEE Ultrasonics, Ferroelectrics and Frequency Control Society originally in 1966 and most recently revised in 1987 [38]. This committee formulated linearized constitutive relations describing piezoelectric continua which form the basis of piezoelectric behavior that is presently in general use.

The linearized constitutive relations are typically represented in a compressed matrix notation as shown in the following equations,

$$S_p = s_{pq}^E \boldsymbol{T}_q + \boldsymbol{d}_{kp} \boldsymbol{E}_k \tag{2.52}$$

$$\boldsymbol{D}_i = \boldsymbol{d}_{iq} \boldsymbol{T}_q + \varepsilon_{ik}^T \boldsymbol{E}_k \tag{2.53}$$

where S represents the strain tensor, s^E is the elastic compliance matrix when subjected to constant electric field, $\boldsymbol{T}$ represents the stress tensor, $\boldsymbol{d}$ is a matrix of piezoelectric material constants, $\boldsymbol{E}$ is the electric field vector, $\boldsymbol{D}$ is the electric displacement vector, and ε^T the permittivity measured at a constant stress. Aside from the awkward notation and the obvious difficulty in implementing these in real-time applications these relations fail to explicitly describe the nonlinearities that are present in all piezoelectric ceramics.

2.5.2.2 A More Accurate Model

A fairly accurate overall electromechanical model of a PZT actuator is given in [38]. It is reproduced in figure 2.43. H represents the hysteresis effect and u_h is the voltage due to this effect. The piezoelectric effect is represented by T_{em}, which is an electromechanical transducer with transformer ratio T_{em}. The capacitance C represents the sum of the capacitances of the individual PZT wafers. The total current flowing through the circuit is $\dot{q}$. The charge q_p is the transduced charge from the mechanical side. The voltage u_p is due to the piezo effect. The total voltage over the PZT actuator is u_{in}, F_p is the transduced force from the electrical side, F_{ext} is the externally applied force, and the resulting elongation of the PZT actuator is denoted by x. The mechanical relation between F_p and x is denoted by M.

The piezoelectric ceramic has elasticity modulus E, viscosity η, and mass density ρ. Furthermore, the geometrical properties of the PZT actuator are length L and cross-sectional area A. Mass m_p, stiffness k_p, and damping coefficient c_p can be calculated from the material and geometrical properties as follows:

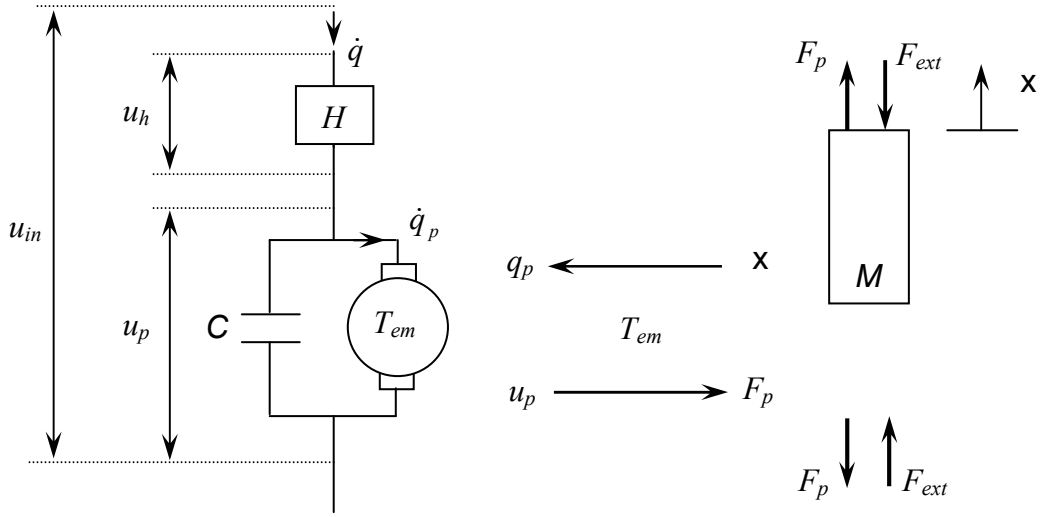

Figure 2.43. Electromechanical model of the PZT actuator

$$m_p = \rho AL \tag{2.54}$$

$$k_p = \frac{EA}{L} \tag{2.55}$$

$$c_p = \frac{\eta A}{L} \tag{2.56}$$

The complete electromechanical equations are defined by (2.57) thru (2.62). The model can also be described by the block diagram given in Fig. 2.44.

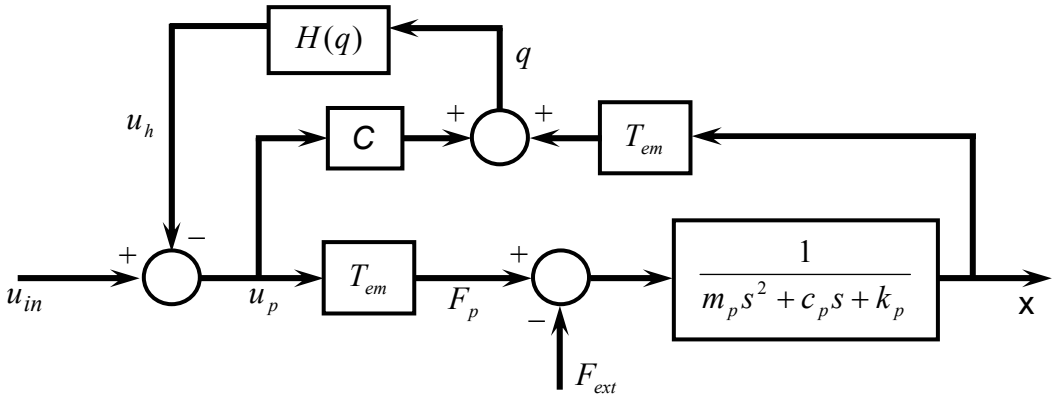

Fig. 2.44. Block-Diagram representation of the electromechanical model

$$u_p = u_{in} - u_h \tag{2.57}$$

$$u_h = H(q) \tag{2.58}$$

$$q = \mathbf{C}u_p + q_p \tag{2.59}$$

$$q_p = T_{em}\mathbf{x} \tag{2.60}$$

$$F_p = T_{em}u_p \tag{2.61}$$

$$m_p\ddot{\mathbf{x}} + c_p\dot{\mathbf{x}} + k_p\mathbf{x} = F_p - F_{ext} \tag{2.62}$$

2.5.2.3 Hysteresis Model

By definition, a hysteretic effect is dynamic, rate-independent and nonlinear. In [38], this effect is modeled by a combination of elements that were called elasto-slide elements. The accuracy of the model was improved by using larger numbers of those elements, and hence the number of parameters involved was large.

A hysteresis loop is defined as the stationary loop in the input-output plane for a quasi-static monotone oscillating input such as a low-frequency sinusoid. The equation under consideration is a first-order differential equation that is proposed in [37]. The model between the hysteresis effect between u_h and q is given by

$$\dot{q} = \alpha|\dot{u}_h|\big(f(u_h) - q\big) + \dot{u}_h g(u_h) \tag{2.63}$$

where $f(u_h)$ and $g(u_h)$ are functions which are used to "shape" the hysteresis loop.

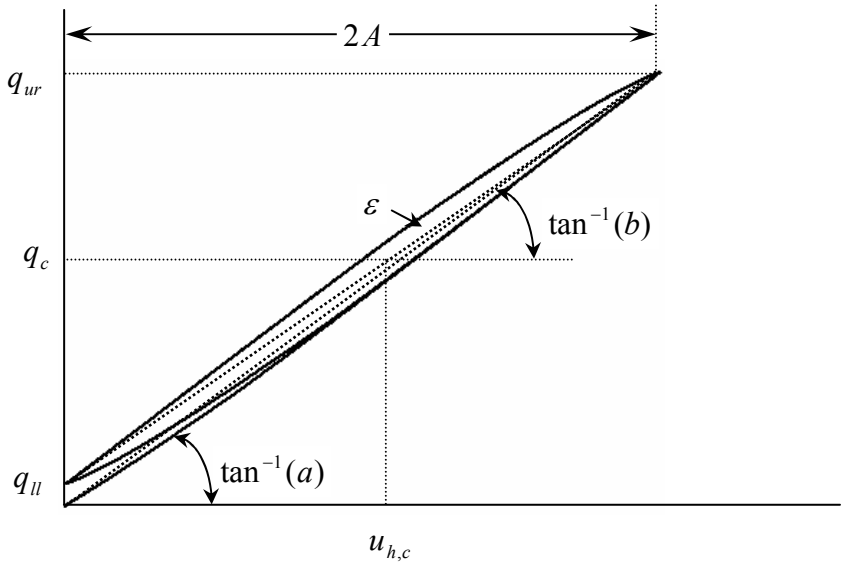

Figure 2.45. A hysteresis loop

In theory, PZT actuators show lengthening saturation. In practice, however, saturation is avoided, i.e., hysteresis loops that are similar in shape to the one in Fig. 2.45 are dealt with. Therefore, the functions $f(u_h)$ and $g(u_h)$ may be chosen as

$$f(u_h) = au_h \tag{2.64}$$

$$g(u_h) = b \tag{2.65}$$

where a and b are constants.

Using the previously mentioned results, the equations for the center point and the average slope of a hysteresis loop are given by

$$q_c = au_{h,c} \tag{2.66}$$

$$q_{ur} - q_{ll} = b \cdot 2\boldsymbol{A} \tag{2.67}$$

where q_c and q_{ll} are the upper right- and lower left-hand-side points of a hysteresis loop, respectively, and $\boldsymbol{A}$ is the input amplitude.

In (2.63), (2.64) and (2.65) there are three independent parameters, namely α, a and b. This means that there should be three independent characteristic quantities in a hysteresis loop. Besides the center point and the average slope, in [37], a relation has been derived for the hysteresis area for relatively small amplitudes of the sinusoidal input (Figure 2.45), as shown in the following.

$$\varepsilon = \frac{4}{3}(a-b)\alpha A^3 \tag{2.68}$$

Having experimentally determined a and b from center points and average slopes, the parameter α can then be experimentally determined from hysteresis areas.

As it seen clearly from the above discussion, forming a mathematical model of a PZT actuator is a very complicated task. Furthermore, the resulting formulas may not perform well for different experimental setups due to the nature of the formulas. Thus, the neurocontroller proposed in this thesis is a very good candidate for the control of this plant. In the following sections, the controller used is described briefly and the experimental results are presented.

2.5.3. The Controller Structure

In (2.62) the term m_p representing the mass of the plant is approximately one million times smaller than the damping and elasticity terms for low frequencies. Thus, the system can be treated as a first order system, to simply the structure of the controller. The proposed controller is shown in Fig. 2.46.

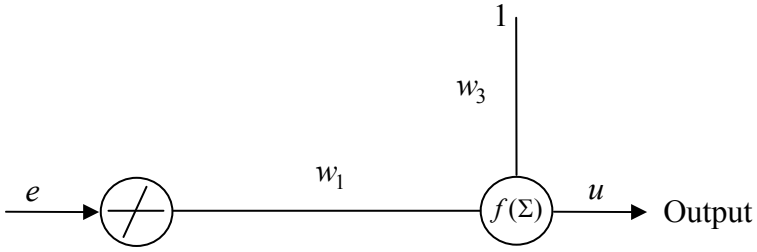

Figure 2.46. The proposed controller for the PZT Actuator

The weight updates are shown in the following equations.

$$w_1(k+1) = w_1(k) + \overline{\eta}\left(\dot{\sigma}(k) + D\sigma(k)\right)\left\|e(k)\right\| \tag{2.69}$$

$$w_3(k+1) = w_3(k) + \overline{\eta}\left(\dot{\sigma}(k) + D\sigma(k)\right) \tag{2.70}$$

The controller parameters are selected as $D = 7000$, $\overline{\eta} = 800$. The system response is presented in Fig. 2.47 – 2.51. Fig. 2.47 shows that the tracking is excellent. The error in this tracking is presented at Fig. 2.48. As it is seen, the error bound is around 0.15 nanometers, which indicates the high performance of the controller. Fig. 2.50 shows the time evolution of w_1 and as it seen there is no built up problem.

To show that the controller can also handle very small step references, a 0,5 nanometer step reference is given to the system and the response of the system is presented in Fig. 2.52. From the figure it can be concluded that the controller performs well even for such a small step reference input.

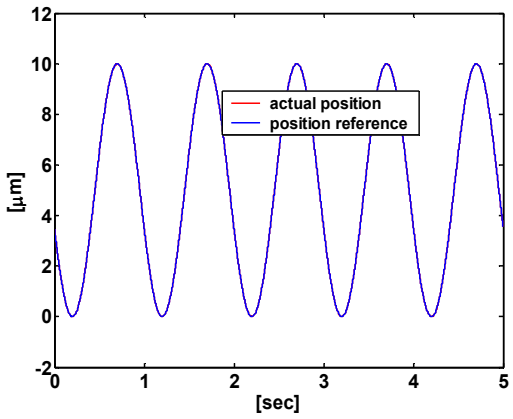

Figure 2.47. Position tracking

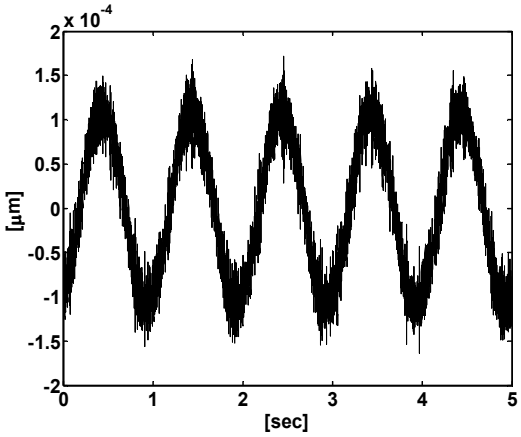

Figure 2.48. Position tracking Error

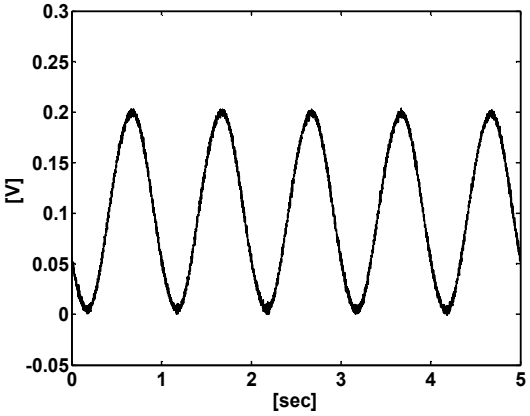

Figure 2.49. The control input

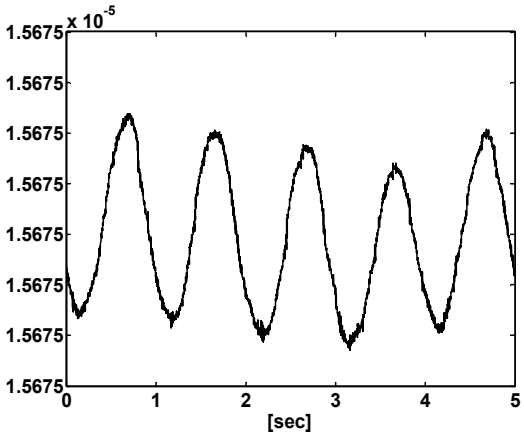

Figure 2.50. Time evolution of w1

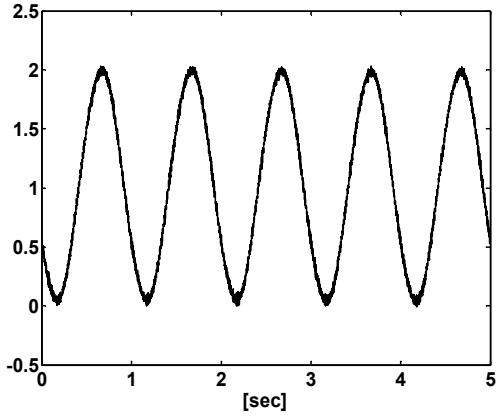

Figure 2.51. Time evolution of w3

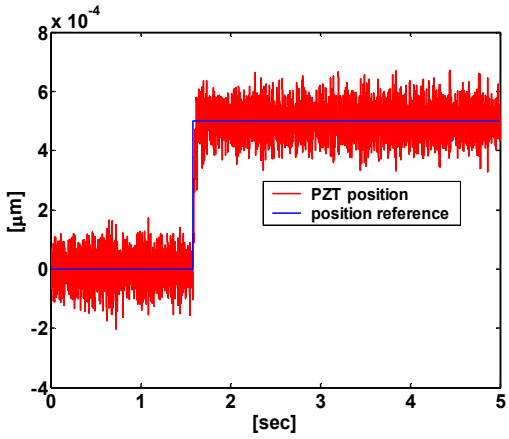

Figure 2.52. Position tracking of the PZT

2.6. Discussion

In this chapter a control algorithm by merging sliding mode control and neural network technology is presented for SISO systems with partially known dynamics. The reasons of the necessity for such a controller are given and the formulation of the controller is shown. The simulation and experimental results are given for different plants to verify the performance of the controller.

The first plant used to test the controller is a linear, single axis, servo drive, driven by an electric motor. It is shown that although the friction and the load on the system is not known, the proposed controller is able to make the system to follow a desired trajectory with a control effort that is bounded by acceptable values.

The second plant is a PZT actuator, modeling of which is still a hot research topic. Again it is shown by experiments that the controller proposed can control this system with 0.15 nanometers error bound, without the information of the model. Also, it is shown that the system can be pushed to follow a 0.5 nanometer step input.

As a result, it is shown that the proposed controller can be used for systems with partially unknown dynamics and for systems that are exposed to uncertain external disturbances. In the next chapter, the generalization of this control algorithm for MIMO systems is presented and results are supported by simulations and experiments.

3. NEURO-SLIDING MODE CONTROLLER DESIGN FOR MIMO SYSTEMS

3.1. Problem Definition

Consider the class of MIMO systems described by the following differential equation:

$$\dot{x} = f(x) + B(x)u + d \tag{3.1}$$

where $x = \left[y_1,...,y_1^{(n_1-1)},...,y_m,...y_m^{(n_m-1)}\right]^T \in \Re^n$ is the state vector, $y = \left[y_1,..,y_m\right]^T \in \Re^m$ is the output vector, $u \in \Re^m$ is the control vector, $f(x) \in \Re^n$ is an unknown, continuous and bounded nonlinear function, $B(x) \in \Re^{nxm}$ is a known input matrix whose elements are continuous and bounded and $d \in \Re^n$ is an unknown, bounded external disturbance. Also, $y_i^{(n_i)} = d^{n_i} y_i / dt^{n_i}$. It is assumed that the system is controllable. The goal is to compute the control action $u = \left[u_1,...,u_m\right]^T$, such that the outputs of the system $y_1,...y_m$ track the desired trajectories $y_{d_1},...y_{d_m}$. To achieve this goal sliding mode control is used as the control scheme. The tracking error is defined as $e_t = \left[e_1,...,e_1^{(n_1-1)},...,e_m,...e_m^{(n_m-1)}\right] \in \Re^n$, where, $e_i = y_{d_i} - y_i$.

3.2. Controller Design

To design a sliding mode controller for the defined problem, firstly, a sliding manifold should be chosen. Secondly, to ensure the sliding mode existence, a Lyapunov function should be determined in terms of the sliding function and the necessary control input should be computed that will fulfill the requirements of the Lyapunov stability criteria.

In the following sections, these design steps is described in detail and the reason for the necessity of a neural network controller is given.

3.2.1. Selection of the sliding surfaces

The sliding surfaces are defined as:

$$\sigma = Ge_t = 0 \tag{3.2}$$

where, $\sigma = [\sigma_i, ...\sigma_m] \in \Re^m$, $\boldsymbol{G} \in \Re^{mxn}$. Matrix $\boldsymbol{G}$ is selected such that each sliding surface takes the form: $\sigma_i = \left(\frac{d}{dt} + C\right)^{n-1} e_i = 0$, where C is a positive, real constant.

3.2.2. Computing the Necessary Control Input

A Lyapunov Function candidate can be selected as:

$$V = \frac{1}{2}\sigma^T\sigma \tag{3.3}$$

where, $V \in \Re$. This function can also be stated as: $V = (1/2)\|\sigma\|_2^2$, where $\|\cdot\|_2$ indicates Euclidian norm. Equating the time derivative of this function to a negative definite function, the necessary control input can be computed:

$$\dot{V} = -\sigma^T \boldsymbol{D}\sigma \tag{3.4}$$

where, D is a positive definite, symmetric matrix. Substituting (3.3) into (3.4) and making the necessary arrangements, the following requirement is found.

$$\sigma^T\left(\dot{\sigma} + \boldsymbol{D}\sigma\right) = 0 \tag{3.5}$$

Therefore, for $\sigma \neq 0$, the control law can be calculated by satisfying the following equation.

$$\left(\dot{\sigma} + \boldsymbol{D}\sigma\right) = 0 \tag{3.6}$$

Finally, using (3.1) and (3.2), the control law is calculated as:

$$u = -\boldsymbol{G}B(x)^{-1}\left(\boldsymbol{G}f(x) + \boldsymbol{G}d(x) - \boldsymbol{G}\dot{x}_{d_i} - \boldsymbol{D}\sigma\right) \tag{3.7}$$

where, $x_d = \left[y_{d_1},...,y_{d_1}^{(n_1-1)},...,y_{d_m},...,y_{d_m}^{(n_m-1)}\right]$. To realize this control input, exact information about the plant dynamics and external disturbances are needed. In the literature some solutions approximating the actual control input (3.7) exist to overcome this difficulty (see [30]) but here, instead of approximation, the output of a neural network, which is computed by minimizing the function $\dot{\sigma} + \boldsymbol{D}\sigma$, is used as the control input. The neural network consists of only one layer of weights and uses linear activation function in the output layer.

3.2.3 The Structure and the Working Principles of the Neural Network

The structure of the NN is presented in Fig. 3.1, where, e_{t_i} is the i^{th} row of e_t.

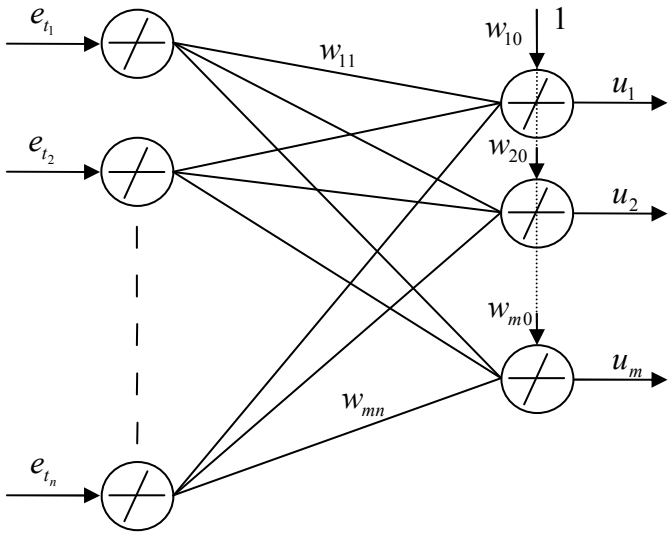

Figure 3.1 Structure of the NN

The control input can be defined as $u = \left[u_1,...,u_m\right]$, where each component is computed as:

$$u_i = \sum_{j=1}^{n} e_{t_j} w_{ij} + 1 w_{i0},\ i = 1,...,m \tag{3.8}$$

The weight update algorithm is shaped by least square minimization, i.e. introducing the following error function to the network for minimization.

$$E = \frac{1}{2}\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \left(\dot{\sigma} + \boldsymbol{D}\sigma\right) \tag{3.9}$$

Equivalently, this error function can be stated as $E = (1/2)\left\|\dot{\sigma} + \boldsymbol{D}\sigma\right\|_2^2$. In the following sections, it is shown that the weights, hence the outputs of the NN, is updated successfully and (3.9) is forced to go to zero, resulting in a sliding motion.

3.2.3.1 Weight Update Algorithm

Weights are updated by using backpropagation:

$$\dot{w}_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{3.10}$$

where, η is the learning constant, generally chosen between 0 and 1. To compute the weight updates, the derivative of the error function E w.r.t. w_{ij} should be found. Using the chain rule, the derivative can be written as:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_i}\frac{\partial u_i}{\partial w_{ij}} \tag{3.11}$$

Substituting (3.9) into (3.11) and taking the necessary derivatives, the following equation is obtained.

$$\frac{\partial E}{\partial w_{ij}} = \left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \frac{\partial\left(\dot{\sigma}\right)}{\partial u_i} e_{t_j} \tag{3.12}$$

Substituting (3.2) into (3.12), the following equation is obtained.

$$\frac{\partial E}{\partial w_{ij}} = \left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \frac{\partial\left(\boldsymbol{G}\dot{x}_d - \boldsymbol{G}\dot{x}\right)}{du_i} e_{t_j} \tag{3.13}$$

Rewriting (3.1),

$$\dot{x} = f(x) + \left[B_1(x)\vdots\ldots\vdots B_m(x)\right]\begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} + d \tag{3.14}$$

and substituting (3.14) into (3.13) and taking the derivative gives the following result.

$$\frac{\partial E}{\partial w_{ij}} = -\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \boldsymbol{G}B_i(x)e_{t_j} \tag{3.15}$$

where $B_i(x)$ is the i^{th} column of the matrix $B(x)$. Therefore, the weight update is given as:

$$\dot{w}_{ij} = \eta\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \boldsymbol{G}B_i(x)e_{t_j} \tag{3.16}$$

For the bias terms, the weight update is given as:

$$\dot{w}_{i0} = \eta\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T \boldsymbol{G}B_i(x) \tag{3.17}$$

For computer applications, the weights are updated in a discrete fashion, as shown in the following.

$$w_{ij}\left[(k+1)h\right] = w_{ij}\left[(kh)\right] + \eta h\left(\dot{\sigma}\left[(kh)\right] + \boldsymbol{D}\sigma\left[(kh)\right]\right)^T \boldsymbol{G}B_i(x,t)e_{t_j}\left[(kh)\right] \tag{3.18}$$

where h is the sampling interval. The derivative of the sliding function is calculated as $\dot{\sigma}(kh) = \left(\sigma(kh) - \sigma(kh-h)\right)/h$. The update for the bias term is done in the same way.

3.2.3.2 Stability Proof

It is a known fact that one of the biggest problems in backpropagation weight update algorithm is sticking to local minima. In this section, it is shown that, using the

proposed control algorithm, the error function (3.9) is pushed to zero without sticking to a local minimum.

3.2.3.2.1 The Shape of the Error Surface

If a function's second derivative does not change sign with respect to a function variable, then the function at hand does not have a change in the curvature sign through that variable, which means that the function does not have a local minimum through that variable. Taking the second derivative of the error function (3.9), w.r.t the weight w_{ij} gives the following result.

$$\frac{\partial^2 E}{\partial w_{ij}^2} = -\eta \left(\frac{\partial \left(\dot{\sigma} + \boldsymbol{D}\sigma \right)^T}{\partial w_{ij}} \right) \boldsymbol{G} B_i(x) e_{t_j} \tag{3.19}$$

Using the chain rule,

$$\frac{\partial^2 E}{\partial w_{ij}^2} = -\eta \left(\frac{\partial \left(\dot{\sigma} + \boldsymbol{D}\sigma \right)^T}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}} \right) \boldsymbol{G} B_i(x) e_{t_j} \tag{3.20}$$

and substituting (3.8), the following equation is obtained.

$$\frac{\partial^2 E}{\partial w_{ij}^2} = -\eta \left(\frac{\partial \left(\dot{\sigma} + \boldsymbol{D}\sigma \right)^T}{\partial u_i} \right) \boldsymbol{G} B_i(x) e_{t_j}^2 \tag{3.21}$$

The term $D\sigma$ does not depend on u. Knowing this and using (3.2), (3.21) can be simplified as:

$$\frac{\partial^2 E}{\partial w_{ij}^2} = -\eta \left(\frac{\partial \left(\boldsymbol{G}\dot{x}_d - \boldsymbol{G}\dot{x} \right)^T}{\partial u_i} \right) \boldsymbol{G} B_i(x) e_{t_j}^2 \tag{3.22}$$

The term x_d does not depend on u. Knowing this and substituting (1) into (3.22), gives the following equation.

$$\frac{\partial^2 E}{\partial w_{ij}^2} = \eta\left(\frac{\partial\left(f(x)^T \boldsymbol{G}^T + u^T B(x)^T \boldsymbol{G}^T + d(x)^T \boldsymbol{G}^T\right)}{\partial u_i}\right)\boldsymbol{G}B_i(x)e_{t_j}^2 \tag{3.23}$$

Taking the derivative,

$$\frac{\partial^2 E}{\partial w_{ij}^2} = \eta B_i(x)^T \boldsymbol{G}^T \boldsymbol{G}B_i(x)e_{t_j}^2 = \eta\left\|\boldsymbol{G}B_i(x)\right\|_2^2 e_{t_j}^2 \tag{3.24}$$

Using the same procedure, second derivative of the error function (3.9) w.r.t the bias weights are calculated as:

$$\frac{\partial^2 E}{\partial w_{i0}^2} = \eta B_i(x)^T \boldsymbol{G}^T \boldsymbol{G}B_i(x) = \eta\left\|\boldsymbol{G}B_i(x)\right\|_2^2 \tag{3.25}$$

These results (3.24) & (3.25) show that the sign of the curvature of the error surface (3.9) is always positive, meaning that there are no local minima, thus there is no danger of sticking to a local minimum. All these discussions indicate that, with a proper selection of the learning rate, the proposed network is capable of minimizing the function (3.9) up to its global minimum, which is nothing but zero, which forces the system to approach to the intersection of the sliding surfaces and eventually converges to its reference. Also, since η is a constant scalar, $\boldsymbol{G}$ is a constant matrix and $B_i(x)$ is bounded, weight update algorithms (3.16) & (3.17) show that weights converge to a finite value in steady state. A finite value for the weights in steady state results in a bounded control input (3.8). As a result, all the signals in the control system are bounded.

3.2.3.2.2 Stability Proof Based on Lyapunov

Let the Lyapunov function candidate be

$$V = \frac{1}{2}\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T\left(\dot{\sigma} + \boldsymbol{D}\sigma\right) \tag{3.26}$$

This is exactly the same function as the error function (3.9) introduced to the NN. It is seen that $V > 0$ for $\dot{\sigma} + D\sigma \neq 0$. Taking the time derivative of V, one obtains the following equation.

$$\dot{V} = \sum_{i=1}^{m} \sum_{j=0}^{n} \frac{\partial V}{\partial w_{ij}} \frac{dw_{ij}}{dt} \tag{3.27}$$

Since $E = V$, using (3.10), the following expression is obtained.

$$\dot{V} = -\eta \sum_{i=1}^{m} \sum_{j=0}^{n} \left(\frac{\partial V}{\partial w_{ij}} \right)^2 \tag{3.28}$$

In this expression, (3.28), η is a positive scalar, thus $\dot{V} \geq 0$. However, since it is proven that the error surface – hence, the Lyapunov function – does not have any local minima, the expression $\partial V / \partial w_{ij}$ becomes zero only at the global minimum, which is zero. So, $\dot{V} < 0$ for $\dot{\sigma} + \boldsymbol{D}\sigma \neq 0$. This proves that Lyapunov function converges to zero and the requirement (3.6) is satisfied, resulting a stable system.

3.3. Simulation Results

To check the performance of the controller, the problem of parking a nonholonomic mobile robot to a given desired position and orientation in plane, is attacked.

3.3.1 Kinematic Model of the System

Fig. 3.2 demonstrates the kinematics of the mobile robot. According to this model, the kinematic equations can be written as

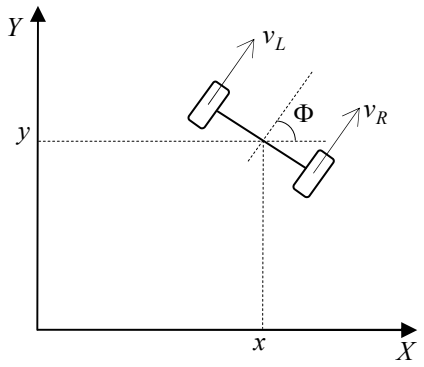

Figure 3.2 Kinematic model of the mobile robot

$$\begin{aligned} \dot{x} &= \frac{u}{2}\cos(\Phi) \\ \dot{y} &= \frac{u}{2}\sin(\Phi) \\ \dot{\Phi} &= \frac{v}{L} \end{aligned} \tag{3.29}$$

where, u is the average of the left and right wheel velocities and v is half of the difference between the wheel velocities.

After these definitions, the problem is stated as: "What should be the control inputs u and v such that the robot will go to the origin and park itself horizontally ($\Phi = 0$) from an arbitrary initial position and orientation?" To solve this problem, the error function

$$E = \frac{1}{2}\left(\dot{\sigma} + \boldsymbol{D}\sigma\right)^T\left(\dot{\sigma} + \boldsymbol{D}\sigma\right) \tag{3.30}$$

where,

$$\sigma = \begin{bmatrix} \dot{x} + Cx \\ \dot{y} + Cy \\ \dot{\Phi} + C\Phi \end{bmatrix} \tag{3.31}$$

is introduced to the controller and the weights are updated according to the procedure explained above.

3.3.2 Simulation Results

Controller parameters used are,

$$\begin{aligned} C &= 10 \\ \boldsymbol{D} &= 10I \\ \eta &= 0.01 \end{aligned} \tag{3.32}$$

Where I is the identity matrix. The trajectory that the mobile robot follows and time evolution of robot coordinates are shown in Fig. 3.3 – 3.6. As it is seen from the graphs, the initial coordinate of the mobile robot is (10, 10) and the orientation is π radians. Applying the control, the robot goes to the origin of the plane with a final orientation of zero radians successfully.

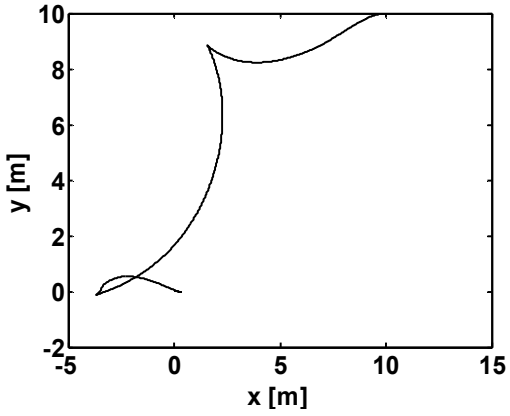

Figure 3.3. Trajectory of the mobile robot

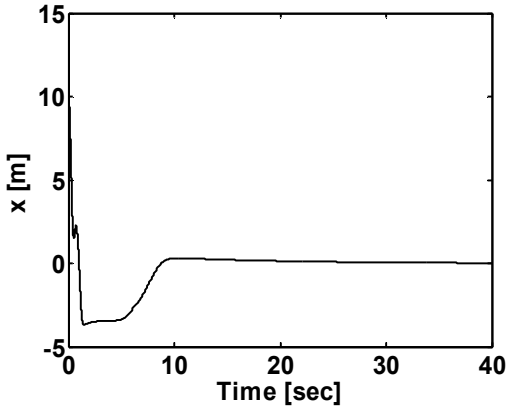

Figure 3.4. Time evolution of x

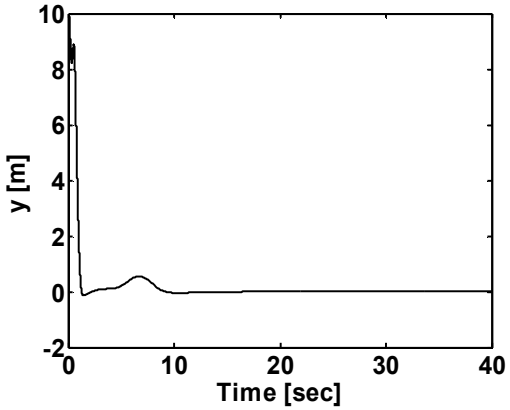

Figure 3.5 Time Evolution of y

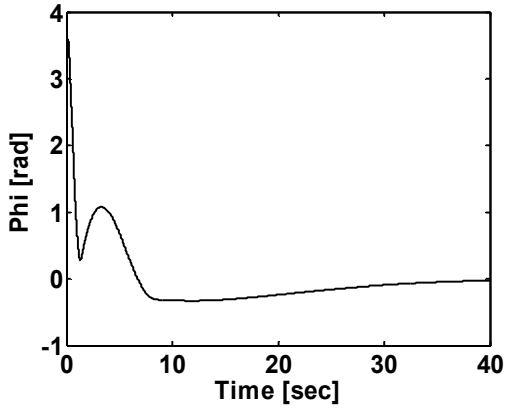

Figure 3.6 Time Evolution of Φ

3.4. Experimental Results

To verify the performance of the proposed controller for MIMO systems, the experimental setup used for SISO case – piezoelectric actuator setup – is modified to represent a MIMO system. The structure of the setup is presented in Fig. 3.7. In this setup, two piezo-drives (PD) are attached to each other via a load cell that is used for force measurement. The aim is to control the position of one actuator while controlling the force that is created due to the reaction of the load cell. Force control is achieved by moving the other actuator. Thus, there are two outputs of the system, position of one actuator and the force created. Also there are two inputs: the voltage input to the actuator whose position is controlled and the voltage input to the other actuator by the help of which, the force is controlled.

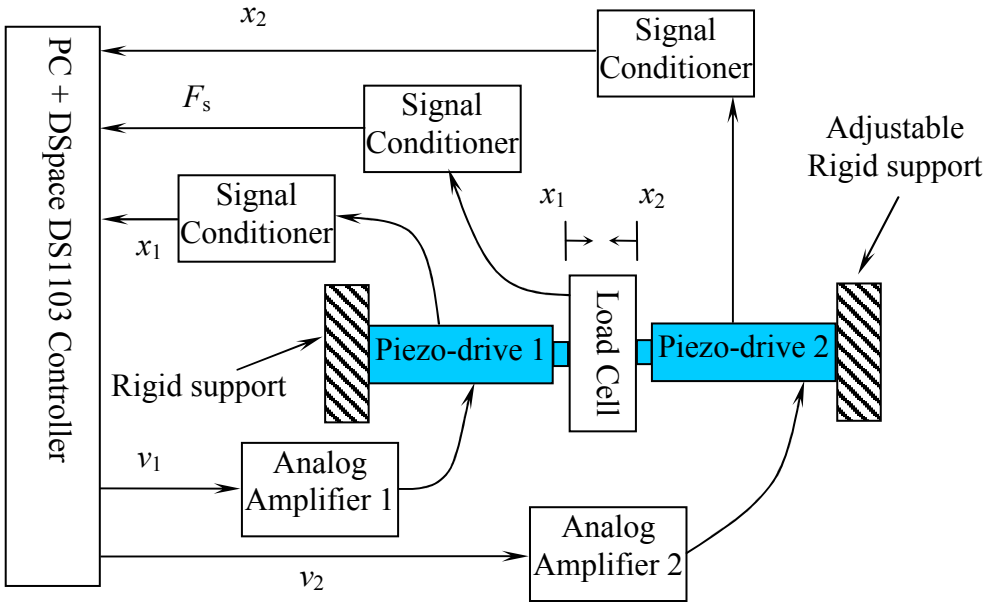

Figure 3.7 Structure of the experimental setup

For all the experiments, the controller parameters are $\boldsymbol{D} = 400I, \eta = 1.5$, and sampling time is selected as 0.0001 seconds.

Fig. 3.8 – 3.14 represent the response of the system for a sigmoid reference for each of the actuators. The references are applied at the same time and PD-1 is able to

track a sigmoid position reference while simultaneously PD-2 moves in such a way that the force created also tracks a sigmoid reference. Fig. 3.14 presents the funny looking trajectory that PD-2 follows to maintain the sigmoid force reference. Also, figures 3.12 and 3.13 shows that both control inputs are bounded and well-behaving.

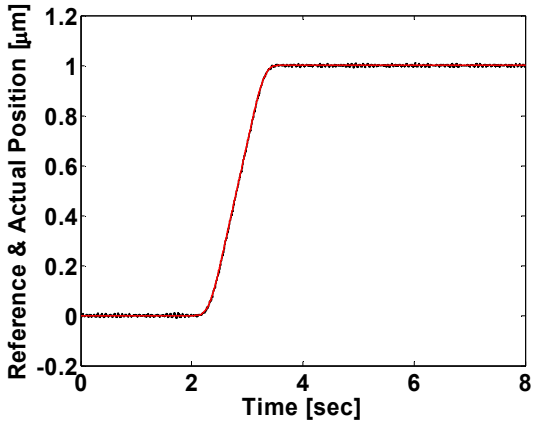

Figure 3.8 Position tracking of PD-1

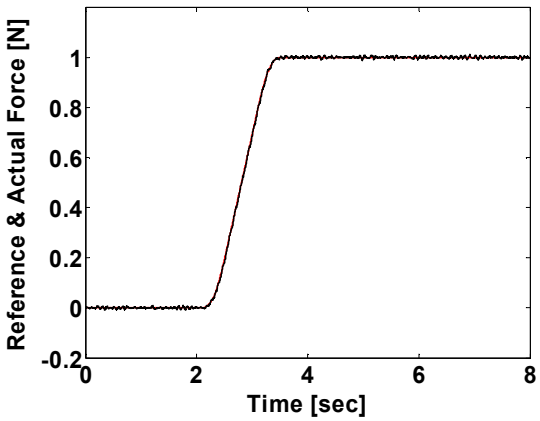

Figure 3.9 Force tracking of PD-2

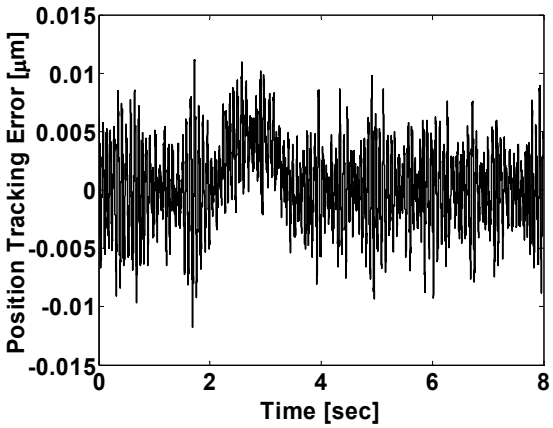

Figure 3.10 Tracking error of PD-1

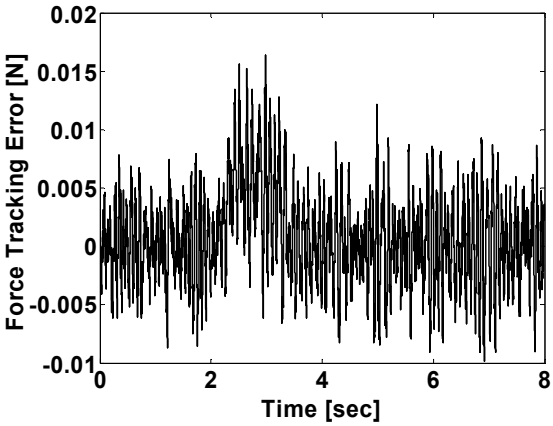

Figure 3.11 Tracking error of PD-2

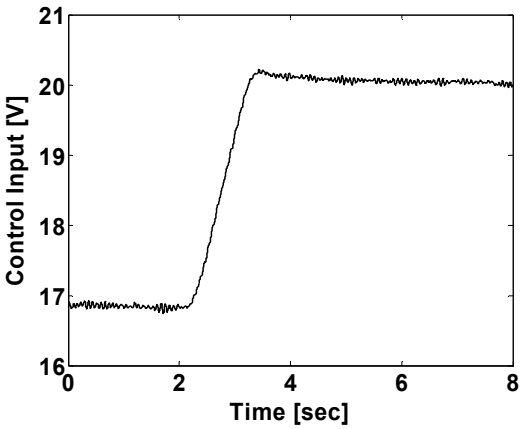

Figure 3.12 Control Input for PD-1

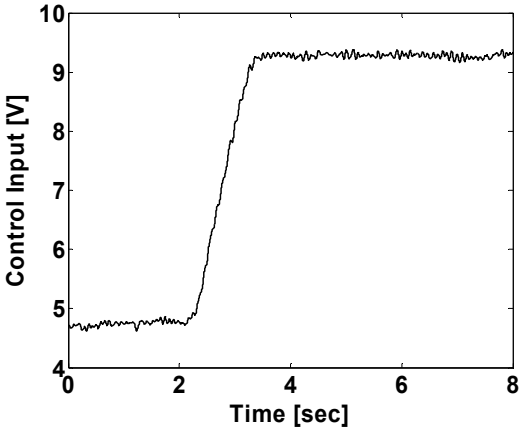

Figure 3.13 Control Input for PD-2

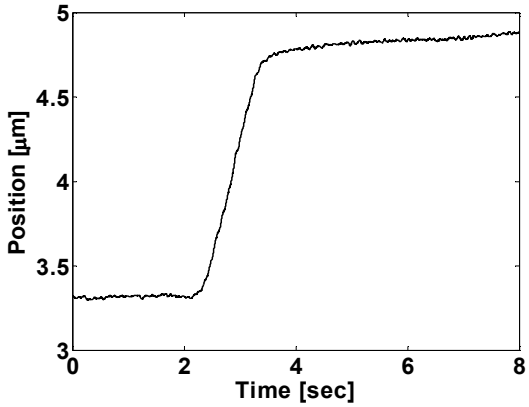

Figure 3.14 Position of PD-2

Another experiment is held to check the performance of the controller for a sine reference for position the force. Fig. 3.15 – 3.21 represent the response of the system for such a reference. Again, the position and the force references are applied at the same time and the result is successful: PD-1 is able to track the sine position reference while simultaneously PD-2 moves in such a way that the force created also tracks the sine reference. Again, PD-2 has to make a strange looking move, which is presented in Fig. 21, to maintain the desired force in the system. Also in this experiment, figures 3.19 and 3.20 shows that both control inputs are bounded and well-behaving.

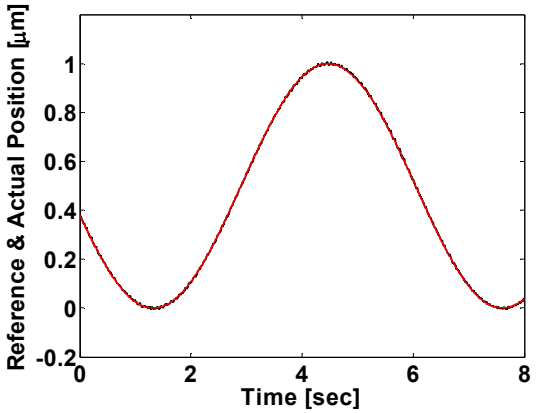

Figure 3.15 Position tracking of PD-1

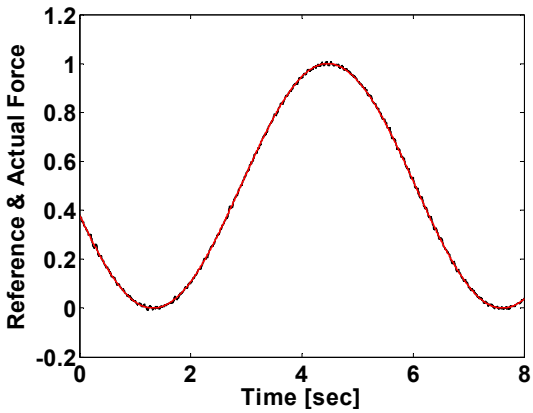

Figure 3.16 Force tracking of PD-2

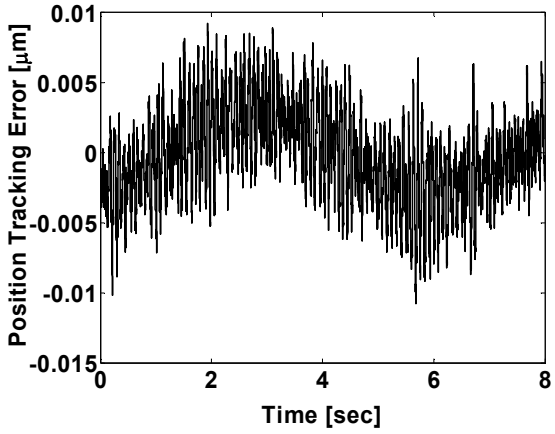

Figure 3.17 Tracking Error of PD-1

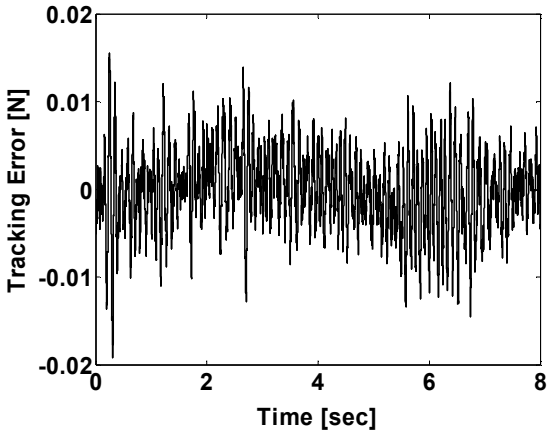

Figure 3.18 Tracking Error of PD-2

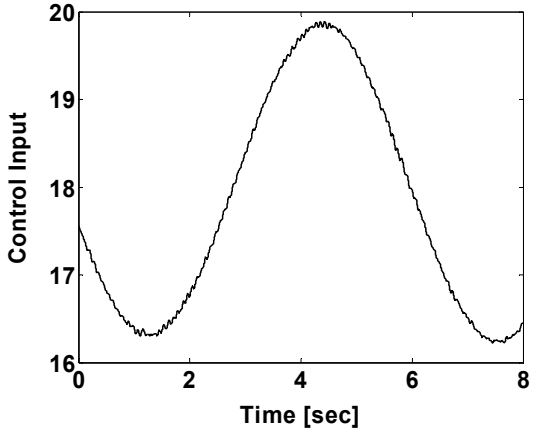

Figure 3.19 Control Input for PD-1

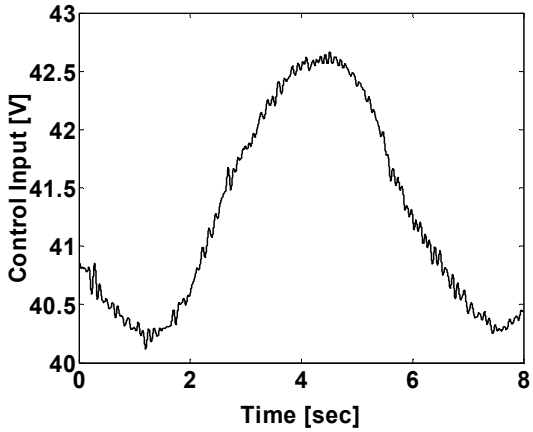

Figure 3.20 Control Input for PD-2

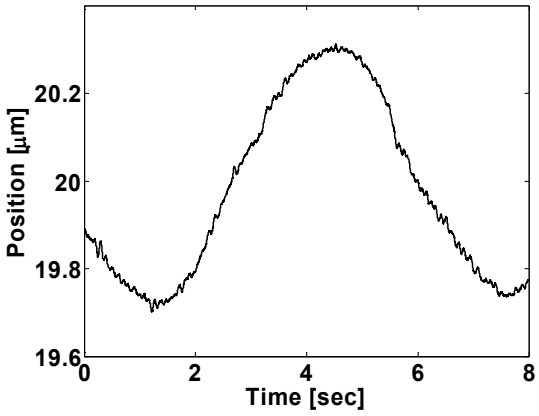

Figure 3.21 Position of PD-2

3.5. Discussion

In this chapter, the theory presented for SISO systems in the previous chapter is generalized for MIMO systems. It is shown that error surface characteristics and stability proofs hold also for MIMO case. To verify the theory simulation and experimental results are given.

A nonholonomic mobile robot model is used for the simulations. It is shown that the proposed algorithm can solve the parking problem, i.e starting from an arbitrary point and orientation on the x-y plane and parking the mobile robot to the origin with a predefined orientation.

Two piezo actuators attached together via a load cell that is used to measure the force is used as the experimental setup. The system has two inputs, two voltages that feed the piezo actuators, and two outputs, the position of one of the actuators and the force created in the system due to the reaction of the load cell. Experimental results show that using the proposed control scheme it is possible to control the force and the position at the same time.

To conclude, other than satisfying Lyapunov stability criteria, the proposed control scheme for MIMO systems prove itself after performing well in the simulations and experiments.

4. CONCLUSION AND DISCUSSION

In this work, a controller scheme that merges sliding mode control and neural networks is presented. The controller benefits from the well-established theory of the sliding mode control and uncertainty dealing potential of the neural networks. The idea is simple: Find the necessary condition for sliding mode to exist and satisfy it by minimizing a cost function via backpropagation. One of the most important results of this controller is that, unlike many similar attempts in the literature to use sliding mode and neural networks together, its stability could be proven. The biggest reason that the stability could be proven comes from the structure of the neural network used. The neural network is a one-layer net with linear activation functions in all the neurons. Since the activation functions are linear, the outputs, that are the control signals, are linear combinations of the inputs. This fact leads to a cost function that has no local minima, thus guaranteeing the convergence to the global minimum.

Although the neural network structure used gives the opportunity to prove the stability, it brings on the following question: Is this really a *Neural Network* ? To answer this question, first one needs a proper and exact definition of neural networks. Let's have a look at one of the definitions:

According to the DARPA Neural Network Study (1988, AFCEA International Press, p. 60):

> *"... a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes."*

According to this explanation, it can be easily claimed that yes, the network used in this work is really a neural network. Unfortunately, this is not the only definition for the neural networks. Here is another one:

According to Haykin, S. (1994), Neural Networks: A Comprehensive Foundation, NY: Macmillan, p. 2:

> *"A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
>
> *1.Knowledge is acquired by the network through a learning process.*
> *2.Interneuron connection strengths known as synaptic weights are used to store the knowledge."*

Now, there is a problem. Does the network used in this work store experiential knowledge and makes it available for use? The answer is most probably no, because this network is updating its variables from zero whenever one operation finishes and the other starts. However, this problem can be solved by the idea of on-line learning which means that the network uses its experiential knowledge about one operation while the operation is being executed. Hence, using the information at the previous sampling intervals, it decides what to do for the sampling interval it is in.

As it is seen, both claims are reasonable. The network used can be named as neural network or without pronouncing the word *neural* , the whole controller can be named as *adaptive sliding mode control* . Anyhow, whatever the name is, as soon as it works, it counts.

Simulations and experiments show that the controller performs well under uncertainties. However, while tuning the controller parameters, there is no rigorous method, it's still trial and error. A deeper mathematical analysis may give some bounds on the controller parameters, thus facilitate the tuning process.

As a conclusion, this controller is promising for the control of uncertain systems and is a good candidate for the industry applications that needs intelligent controllers.

REFERENCES

[1] Bishop, J.M., Mitchell, R.J., "Neural Networks – An Introduction," *Neural Networks for Systems: Principles and Applications, IEE Colloquium on*, 25 Jan. 1991.

[2] Alexander, I and Morton, H., "An Introduction to Neural Computing" Charpman and Hall, 1990.

[3] Minsky, M. and Papert, S. "Perceptrons" MIT Press, 1969.

[4] Rumelhart, D.E., McClelland, J.L. "Parallel Distributed Processing" MIT Press, vol. 1 and 2, 1986.

[5] Narendra. K.S., "Neural networks for control theory and practice," *Proceedings of the IEEE* , vol. 84 , Issue: 10, pp. 1385 – 1406, Oct. 1996.

[6] K. Funashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183-192, 1989.

[7] G. Cybenko, "Approximation by superposition of a Sigmoidal function," *Mathematics Contr., Signals, and Syst*., vol. 2, no. 4, pp. 303-314, 1989.

[8] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, pp. 359-366, 1989.

[9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares algorithm for radial basis function networks," *IEEE Trans. Neur. Networks*, vol. 2, pp. 302-309, Mar. 1991.

[10] K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neur. Networks*, vol. 1, pp. 4-27, Mar. 1990.

[11] A. R. Barron, "Neural Net Approximation," *Proc. 7th Yale Workshop on Adaptive and Learning Syst.*, 1992, pp. 68-72; "Universal approximation bounds for superpositions of a Sigmoidal function," *IEEE Trans. Inform. Theory*, vol. 39, pp. 930-945, Mar. 1993.

[12] M. Kawato, "Computational schemes and neural network models for formation and control of multi-joint arm trajectory," *Neural Networks for Control*, pp.197-228, 1990.

[13] A.Y. Zomoya and T.M. Nabhan, "Centralized and decentralized neuro-adaptive controllers," *Neural Networks*, vol. 6, no. 2, pp. 223-244, 1993.

[14] M.B. Leahy, M. A. Johnson, and S. K. Rogers, "Neural network payload estimation for adaptive robot control," *IEEE Trans. Neur. Networks*, vol. 2, pp. 93-100, Jan. 1991.

[15] M. Kuperstein, "Neural network model for adaptive hand-eye coordination," *Sci.*, vol. 239, pp. 1308-1311, 1998.

[16] D.A. Handelman, S. H. Lane, and J.J. Gelfand, "Integrating neural networks and knowledge-based systems for intelligent robotic control," *IEEE Contr. Syst. Mag*., vol. 10, no. 3, pp. 77-87, 1990.

[17] S. Liu and H. Asada, " Transferring manipulative skills to robots – Representation and acquisition of tool manipulative skills using a process dynamics model," *ASME J. Dynamic Syst., Measure, and Contr*., vol. 114, no.2, pp. 220-228, 1992.

[18] S. T. Venkataraman, S. Gulati, J. Barhen, and N. Toomarian, "Compliance control with neuromorphic identification of environmental dynamics," *Proc. IEEE Conf. on Decision and Contr.*, Tucson, AZ, 1992, pp. 3475-3480.

[19] Y. Y. Hwang and I. Todo, "Cooperative control of 2 direct-drive robots using neural networks (grasping and movement of an object)," *JSMEC*, vol. 37, no. 2, pp. 335-341, 1994.

[20] M. K. Hanes, S. C. Ahalt, K. Mirza, and D. E. Orin "Power grasp force distribution control using neural networks," *J. Robotic Syst*., vol .9, no. 5, pp. 635-661, 1992.

[21] W. T. Miller, "Real-time neural network control of a biped walking robot," *IEEE Cont. Systm. Mag.*, vol. 14, no. I, pp.41-48, 1994.

[22] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neur. Computation*, vol. 3, pp. 88 – 97, 1991.

[23] R. D. Beer, "Intelligence as Adaptive Behavious: An Experiment in Computational Neuroehology," *San Diego: Academic*, 1990.

[24] R. D. Beer et al., "A distributed neural network architecture for hexapod robot locomotion," *Neur. Computation*, vol. 4, pp. 356-365, 1992.

[25] W. E. Faller and S. J. Schreck, "Neural networks: Applications and opportunities in aeronautics," *Progress in Aerospace Sciences*, 1996.

[26] H. E. Rauch and D. B. Schaecter, "Neural networks for control, identification, and diagnosis," *Proc. World Space Congress*, pp. F4.4-M1.06, 1992.

[27] V. I. Utkin, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic Control.*, vol. AC-22, pp. 212-222, 1997.

[28] J. Y. Hung, W. B. Gao, and J. C. Hung, "Variable structure control: A survey," *IEEE Transactions on Industrial Electronics*, vol. 40, pp. 2-22, Feb. 1993.

[29] J. J. E. Slotine and W. Li, "Applied Nonlinear Control," Upper Saddle River, NJ: Prentice-Hall, 1991.

[30] Erbatur. K., Kaynak. M.O., Sabanovic A., "A study on robustness property of sliding-mode controllers: a novel design and experimental investigations" *Industrial Electronics, IEEE Transactions on*, vol. 46, Issue: 5, pp. 1012 – 1018, Oct. 1999.

[31] O. Kaynak, K. Erbatur, M. Ertugrul, "The fusion of computationally intelligent methodologies and sliding-mode control – A survey," *IEEE Trans. Indust.* Electr. , vol. 48, no. 1, Feb. 2001.

[32] K. Jezernik, M. Rodic, R. Safaric, B. Curk, "Neural network sliding mode robot control", Robotica, vol. 15, pp 23-30, 1997.

[33] M. Rodic, K. Jezernic, A. Sabanovic, R. Safaric, "Sliding Mode Based Neural Network Learning Procedure", *Proceedings, 5th International Workshop on Robotics in Alpe-Adria-Danube region*, pp. 547-552, June 10-13, 1996.

[34] Fang. Y., Chow. T.W.S., Li. X.D., "Use of a recurrent neural network in discrete sliding-mode control," *Control Theory and Applications, IEE Proceedings*, vol. 146, Issue: 1, pp. 84-90, Jan. 1999.

[35] R. Ben Mrad, H. Hu, " A Model for Voltage to Displacement Dynamics in piezoceramic Actuators Subject to Dynamic – Voltage Excitations," IEEE/ASME Transactions on Mechatronics, vol. 7, no. 4, pp. 479-489, 2002.

[36] Hewong Jung, Jong Yuop Shim and DaeGab Gweon, "Tracking control of piezoelectric actuators," Institute of Physics Publishing, Nano-technology, Vol. 12, pp. 14-20, 2001.

[37] R. Banning, W.L. de Koning and J. M. T. A. Adriaens, "Modeling Piezoelectric Actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 5, pp. 331-341, No. 4, 2000.

[38] Michael Goldfarb and Nikola Celanovic, "Modeling Piezolectric Stack Actuators for Control of Micromanipulation," *IEEE Contr. Sys. Mag*., vol. 17, pp. 69-79, 1997.

REFERENCES

[1] Bishop, J.M., Mitchell, R.J., "Neural Networks – An Introduction," *Neural Networks for Systems: Principles and Applications, IEE Colloquium on*, 25 Jan. 1991.

[2] Alexander, I and Morton, H., "An Introduction to Neural Computing" Charpman and Hall, 1990.

[3] Minsky, M. and Papert, S. "Perceptrons" MIT Press, 1969.

[4] Rumelhart, D.E., McClelland, J.L. "Parallel Distributed Processing" MIT Press, vol. 1 and 2, 1986.

[5] Narendra. K.S., "Neural networks for control theory and practice," *Proceedings of the IEEE* , vol. 84 , Issue: 10, pp. 1385 – 1406, Oct. 1996.

[6] K. Funashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183-192, 1989.

[7] G. Cybenko, "Approximation by superposition of a Sigmoidal function," *Mathematics Contr., Signals, and Syst*., vol. 2, no. 4, pp. 303-314, 1989.

[8] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, pp. 359-366, 1989.

[9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares algorithm for radial basis function networks," *IEEE Trans. Neur. Networks*, vol. 2, pp. 302-309, Mar. 1991.

[10] K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neur. Networks*, vol. 1, pp. 4-27, Mar. 1990.

[11] A. R. Barron, "Neural Net Approximation," *Proc. 7th Yale Workshop on Adaptive and Learning Syst.*, 1992, pp. 68-72; "Universal approximation bounds for superpositions of a Sigmoidal function," *IEEE Trans. Inform. Theory*, vol. 39, pp. 930-945, Mar. 1993.

[12] M. Kawato, "Computational schemes and neural network models for formation and control of multi-joint arm trajectory," *Neural Networks for Control*, pp.197-228, 1990.

[13] A.Y. Zomoya and T.M. Nabhan, "Centralized and decentralized neuro-adaptive controllers," *Neural Networks*, vol. 6, no. 2, pp. 223-244, 1993.

[14] M.B. Leahy, M. A. Johnson, and S. K. Rogers, "Neural network payload estimation for adaptive robot control," *IEEE Trans. Neur. Networks*, vol. 2, pp. 93-100, Jan. 1991.

[15] M. Kuperstein, "Neural network model for adaptive hand-eye coordination," *Sci.*, vol. 239, pp. 1308-1311, 1998.

[16] D.A. Handelman, S. H. Lane, and J.J. Gelfand, "Integrating neural networks and knowledge-based systems for intelligent robotic control," *IEEE Contr. Syst. Mag*., vol. 10, no. 3, pp. 77-87, 1990.

[17] S. Liu and H. Asada, " Transferring manipulative skills to robots – Representation and acquisition of tool manipulative skills using a process dynamics model," *ASME J. Dynamic Syst., Measure, and Contr*., vol. 114, no.2, pp. 220-228, 1992.

[18] S. T. Venkataraman, S. Gulati, J. Barhen, and N. Toomarian, "Compliance control with neuromorphic identification of environmental dynamics," *Proc. IEEE Conf. on Decision and Contr.*, Tucson, AZ, 1992, pp. 3475-3480.

[19] Y. Y. Hwang and I. Todo, "Cooperative control of 2 direct-drive robots using neural networks (grasping and movement of an object)," *JSMEC*, vol. 37, no. 2, pp. 335-341, 1994.

[20] M. K. Hanes, S. C. Ahalt, K. Mirza, and D. E. Orin "Power grasp force distribution control using neural networks," *J. Robotic Syst*., vol .9, no. 5, pp. 635-661, 1992.

[21] W. T. Miller, "Real-time neural network control of a biped walking robot," *IEEE Cont. Systm. Mag.*, vol. 14, no. I, pp.41-48, 1994.

[22] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neur. Computation*, vol. 3, pp. 88 – 97, 1991.

[23] R. D. Beer, "Intelligence as Adaptive Behavious: An Experiment in Computational Neuroehology," *San Diego: Academic*, 1990.

[24] R. D. Beer et al., "A distributed neural network architecture for hexapod robot locomotion," *Neur. Computation*, vol. 4, pp. 356-365, 1992.

[25] W. E. Faller and S. J. Schreck, "Neural networks: Applications and opportunities in aeronautics," *Progress in Aerospace Sciences*, 1996.

[26] H. E. Rauch and D. B. Schaecter, "Neural networks for control, identification, and diagnosis," *Proc. World Space Congress*, pp. F4.4-M1.06, 1992.

[27] V. I. Utkin, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic Control.*, vol. AC-22, pp. 212-222, 1997.

[28] J. Y. Hung, W. B. Gao, and J. C. Hung, "Variable structure control: A survey," *IEEE Transactions on Industrial Electronics*, vol. 40, pp. 2-22, Feb. 1993.

[29] J. J. E. Slotine and W. Li, "Applied Nonlinear Control," Upper Saddle River, NJ: Prentice-Hall, 1991.

[30] Erbatur. K., Kaynak. M.O., Sabanovic A., "A study on robustness property of sliding-mode controllers: a novel design and experimental investigations" *Industrial Electronics, IEEE Transactions on*, vol. 46, Issue: 5, pp. 1012 – 1018, Oct. 1999.

[31] O. Kaynak, K. Erbatur, M. Ertugrul, "The fusion of computationally intelligent methodologies and sliding-mode control – A survey," *IEEE Trans. Indust.* Electr. , vol. 48, no. 1, Feb. 2001.

[32] K. Jezernik, M. Rodic, R. Safaric, B. Curk, "Neural network sliding mode robot control", Robotica, vol. 15, pp 23-30, 1997.

[33] M. Rodic, K. Jezernic, A. Sabanovic, R. Safaric, "Sliding Mode Based Neural Network Learning Procedure", *Proceedings, 5th International Workshop on Robotics in Alpe-Adria-Danube region*, pp. 547-552, June 10-13, 1996.

[34] Fang. Y., Chow. T.W.S., Li. X.D., "Use of a recurrent neural network in discrete sliding-mode control," *Control Theory and Applications, IEE Proceedings*, vol. 146, Issue: 1, pp. 84-90, Jan. 1999.

[35] R. Ben Mrad, H. Hu, " A Model for Voltage to Displacement Dynamics in piezoceramic Actuators Subject to Dynamic – Voltage Excitations," IEEE/ASME Transactions on Mechatronics, vol. 7, no. 4, pp. 479-489, 2002.

[36] Hewong Jung, Jong Yuop Shim and DaeGab Gweon, "Tracking control of piezoelectric actuators," Institute of Physics Publishing, Nano-technology, Vol. 12, pp. 14-20, 2001.

[37] R. Banning, W.L. de Koning and J. M. T. A. Adriaens, "Modeling Piezoelectric Actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 5, pp. 331-341, No. 4, 2000.

[38] Michael Goldfarb and Nikola Celanovic, "Modeling Piezolectric Stack Actuators for Control of Micromanipulation," *IEEE Contr. Sys. Mag*., vol. 17, pp. 69-79, 1997.