DEVELOPMENT OF NEW LEARNING ALGORITHMS FOR SPIKING NEURAL NETWORKS

by

Yeşim Öniz

B.S., Mechatronics Eng., Sabanci University, 2004

M. Sc., Electrical & Electronics Eng., Bogazici University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in
Boğaziçi University
2014

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to my advisor Prof. Okyay Kaynak for his invaluable support and continuous guidance. This Ph.D. thesis would not have been possible without his interest, encouragement and advice. I would also like to thank the members of the committee Prof. Kemal Cılız and Prof. Fikret Gürgen for their endeavor to review the progress of my thesis and to provide valuable comments and suggestions.

I would like to thank Assoc. Prof. Mehmet Akar for his careful evaluation of this study. His insightful comments and suggestions have been of immense help. I am indebted to Prof. Hakan Temeltaş for being a member of my jury. I am also thankful to Prof. Rahib Abiyev and Prof. Andon Topalov for their collaboration and scientific contributions.

My sincere thanks go to my friends Çisel Aras, Erdal Kayacan, Özkan Çiğdem, Telman Raoufiirani, Alireza Ahmadi Asl, Deniz Seviş and Hakan Karaoğuz for their friendship, support and contributions during the course of this study.

Finally, I want to express my deepest thanks and regards to my family for their never-ending motivation and positive attitude to my academic career. I would like to especially thank my mother-in-law Suzan Öniz, my spouse Çağıl and my mother Suheyla Değer. Knowing that they were behind me at every decision, inspired me to start my graduate work and kept me going during the hard days. And my little princess Defnoşka: Thank you so much for your understanding and patience. You gave me hope and strength to finish this thesis.

ABSTRACT

DEVELOPMENT OF NEW LEARNING ALGORITHMS FOR SPIKING NEURAL NETWORKS

The main shortcomings with the gradient descent-based learning algorithms used for neural networks are that the convergence speed is relatively slow and the algorithm can be easily trapped into a local optimum. The studies that demonstrate the robustness of variable structure control have motivated the use of the sliding mode control approach in the training of Artificial Neural Networks (ANNs). In this dissertation, the development of a Spiking Neural Network (SNN) with a novel variable structure systems (VSS)-based learning algorithm is considered for the identification and control of dynamic plants. The parameter update rules are derived based on the Lyapunov stability method. Stable online tuning of the neurocontroller parameters and a fast learning speed are the prominent characteristics of the proposed algorithm. Neural networks are very effective in implementing a mapping between the inputs and outputs of a system, but they have a *black box* nature. On the other hand, fuzzy logic systems are good at explaining how they reach their decisions but they require expert knowledge. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. In the proposed fuzzy SNN (FSNN) structure, SNN is utilized to automate this process and substantially reduce development time and cost while improving performance. The update rules have been derived based on the VSS theory to avoid global stability problems. The developed algorithm has been successfully implemented for the wheel slip regulation of an Antilock Braking System (ABS) and trajectory control of a two-dof SCARA manipulator.

ÖZET

DARBELİ YAPAY SİNİR AĞLARI İÇİN YENİ ÖĞRENME ALGORİTMALARININ GELİŞTİRİLMESİ

Yapay sinir ağlarında (YSA) sıklıkla kullanılan gradyan tabanlı öğrenme algoritmalarının temel eksiklikleri, yavaş yakınsama hızları ve hata fonksiyonunun sadece yerel minimuma indirilmesinin garanti edilmesi olarak sıralanabilir. Değişken Yapılı Sistemler kuramının gürbüzlüğünü vurgulayan çalışmalar, YSA'nın eğitiminde kayma kipli denetim kuramı tabanlı öğrenme algoritmalarının kullanımını desteklemiştir. Bu tez çalışmasında, dinamik sistemlerin tanımlanmasında ve denetiminde kullanılmak üzere Darbeli Yapay Sinir Ağları (DYSA) için değişken sistemler kuramı temelli öğrenme algoritmalarının geliştirilmesi ele alınmıştır. Parametrelerin güncelleme kurallarının saptanmasında Lyapunov kararlılık yöntemi kullanılmıştır. DYSA'nın parametrelerinin kararlı ve hızlı bir şekilde uyarlanması önerilen algoritmanın en önemli iki özelliğidir. YSA, sistemin girişleri ile çıkışları arasında bir ilişki kurmakta oldukça başarılı olmalarına karşın kara kutu olarak düşünülebilirler. Öte yandan, bulanık mantık sistemleri herhangi bir çıktıya hangi girişleri nasıl kullanarak ulaştıklarını açıklamak konusunda oldukça iyi olmakla beraber, uzman bilgisi gerektirmektedirler. Eğer bu bilgi eksik, yanlış veya çelişkili ise bulanık sistemlerin uyarlanması gerekir. Önerilen bulanık DYSA yapısında, DYSA bu uyarlama işlemini otomatikleştirmek ve başarımı artırırken geliştirme zamanını ve maliyetini önemli ölçüde azaltmak için kullanılmışlardır. Uyarlama kuralları, evrensel kararlılık problemini ortadan kaldırmak için değişken yapılı sistemler kuramı kullanılarak belirlenmiştir. Geliştirilen algoritmalar kilitlenmeyen fren sisteminde (KFS) kayma değerinin kontrolünde ve 2 eksenli SCARA tipi robotun pozisyon kontrolünde başarıyla uygulanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

C	Motor constant
C_m	Membrane capacitance
$c(p,q)$	Mean of the Gaussian membership function for the sensor reading p and associated membership function q
d_1	Viscous friction coefficient of the upper wheel in ABS
d_2	Viscous friction coefficient of the lower wheel in ABS
d^k	Delay for the k^{th} synapse
E	Induced electromotive force
$e(t)$	Error
E_K	Reversal potentials of the potassium channel
E_L	Reversal potentials of the leakage channel
E_m	Resting potential of the neuron in SRM
E_{Na}	Reversal potentials of the sodium channel
E_T	Cost function
F	Vector of Coulomb friction terms for the manipulator
$f(x(t))$	Nonlinear function of time and system's states
F_j	Vector of the firing times t_j of the neuron j
F_n	Total normal load
F_t	Road friction force
$g(x(t))$	Nonlinear function of time and system's states
g_L	Conductance of leakage channel
g_{Na}	Conductance of sodium channel
g_K	Conductance of potassium channel
h	Gating variables in the HH model
H	Set of neurons in the input layer for two layered SNN
I	Set of neurons in the hidden layer for two layered SNN
I_A	Armature current
$I(t)$	Current input

J	Set of neurons in the output layer for two layered SNN
J_1	Moment of inertia of the upper wheel in ABS
J_2	Moment of inertia of the lower wheel in ABS
J_m	Moment of inertia of the motor
K	Potassium
K_D	Derivative gain of PID controller
K_I	Integral gain of PID controller
K_P	Proportional gain of PID controller
L	Distance between the contact point of the wheels and the rotational axis of the balance lever in ABS
L_A	Armature winding inductance
m	Gating variable in the HH model
M	Torque produced by the motor
M_{10}	Static friction of the upper wheel in ABS
M_{20}	Static friction of the lower wheel in ABS
M_B	Acceleration torque
M_g	Moment of gravity acting on balance lever
M_L	Load torque
M_p	Mass of the payload
$M(\theta)$	State-varying inertia matrix of the manipulator
n	Gating variables in the HH model
Na	Sodium
R	Membrane resistance
r_1	Radius of the upper wheel in ABS
r_2	Radius of the lower wheel in ABS
R_A	Armature winding resistance
$s(x(t))$	Sliding surface
$sign$	Signum function
t	Time
T	Torque vector applied to the joints

T_A	Electrical time constant
T_B	Braking torque
t_{hit}	Hitting time in SMC
t_i	Firing time of the ith neuron
t_j^d	Desired firing times of the neuron j in the output layer
T_M	Mechanical time constant
t_{max}	Maximum allowable time window for a neuron to fire
t_{min}	Minimum time required for the neuron to be activated
T_s	Sampling time
$u(t)$	Membrane recovery variable in IH and LIF models
U_A	Armature terminal voltage
u_m	Membrane potentialof the neuron in HH model
u_{max}	Maximum value of the control signal
u_{min}	Minimum value of the control signal
u_{rest}	Resting potential of the neuron
$v(t)$	Membrane potential of the neuron in IH model
$V(x,t)$	Lyapunov function candidate
V_r	Relative velocity in LuGre tire/road friction model
V_s	Stribeck relative velocity in LuGre tire/road friction model
$V(\theta,\dot{\theta})$	Vector of centripetal and Coriolis forces for the manipulator
w_{ij}	Weight of the connection between presynaptic neuron i and postsynaptic neuron j
w_{ij}^k	Weight coefficient for the k^{th} synapse between the presynaptic neuron i and the postsynaptic neuron j
$x_j(t)$	Membrane potential of the jth neuron at time instant t
x_{max}	Maximum value of the input variable
x_{min}	Minimum value of the input variable
$y^d(t)$	Desired time-varying output state of the system
$y_i^k(t_j)$	Signal transmitted from neuron i to j through kth synaptic connection

$\alpha(u_m)$	Function determining the dynamics of the gating variables in the HH model
$\beta(u_m)$	Function determining the dynamics of the gating variables in the HH model
Γ_j	Set of presynaptic neurons of neuron j
δ_i	Delta error for neuron j
Δw_{ij}^k	Change of the weight for the k^{th} synapse between the presynaptic neuron i and the postsynaptic neuron j
$\varepsilon(t)$	Spike function in SRM
η	Learning rate
$\eta(t)$	Refractoriness period
θ	Threshold
θ_l	Joint angle of the shoulder link
θ_l	Joint angle of the elbow link
λ	Wheel slip
λ_R	Reference slip
λ_s	Strictly positive constant used in SMC
μ	Road adhesion coefficient
μ_c	Normalized Coulomb friction coefficient in LuGre tire/road friction model
μ_s	Normalized static friction coefficient in LuGre tire/road friction model
$\sigma(p,q)$	Variance of the Gaussian membership function for the sensor reading p and associated membership function q
σ_0	Rubber longitudinal stiffness in LuGre tire/road friction model
σ_1	Rubber longitudinal damping in LuGre tire/road friction model
σ_2	Viscous relative damping in LuGre tire/road friction model
τ	Decay time constant of the spike function
τ_j	Output of the neural network
τ_j^d	Desired output of the neural network

Φ	Magnetic excitation
φ	Angle between the normal in the contact point and the line L in ABS
ω	Speed of the rotor
ω_1	Angular velocity of the upper wheel in ABS
ω_2	Angular velocity of the lower wheel in ABS

LIST OF ACRONYMS/ABBREVIATIONS

ABS	Antilock Braking System
ANN	Artificial Neural Network
DC	Direct Current
DOF	Degrees of Freedom
DYSA	Darbeli Yapay Sinir Ağı
FSNN	Fuzzy Spiking Neural Network
HH	Hodgkin-Huxley Model
IM	Izhikevich Model
KFS	Kilitlenmeyen Fren Sistemi
LIF	Leaky Integrate-and-Fire Model
MLP	Multilayer Perceptron
PID	Proportional-Integral-Derivative
PSP	Postsynaptic Potential
RC	Resistor-Capacitor
RMSE	Root Mean Square Error
SMC	Sliding Mode Control
SN	Spiking Neuron
SNN	Spiking Neural Network
SNR	Signal-to-Noise ratio
SRM	Spike Response Model
STDP	Spike-Timing Dependent Plasticity
VSS	Variable Structure System
WTA	Winner-Takes-All
XOR	Exclusive OR
YSA	Yapay Sinir Ağı

1. INTRODUCTION

In most industrial applications, the engineers face the difficulty of incomplete or insufficient information about the system which hampers the development of an accurate dynamic model for use in the design of model-based control schemes. Furthermore, inevitably noisy and uncertain sensor measurements may pose significant hindrance in achieving the desired performance even if a sufficiently accurate model of the system is available. In such cases, model-free approaches are preferred to reduce the dependency of the controller performance on the exact modeling information.

Artificial neural networks (ANNs) are widely considered among the most common and effective approaches to model-free design. The first generation of artificial neural networks consists of McCulloch-Pitts neurons [2]. In this conceptually very simple model, a neuron sends a binary 'high' signal if the sum of its weighted incoming signals is above a threshold value. Neurons of the second generation use a continuous activation function to compute their output signals instead of a step or threshold function [3]. This feature makes them suitable for use in systems with analog inputs and outputs. The third generation of ANNs enhances the level of biological realism by utilizing the spiking neuron (SN) models [4]. Unlike the first two generations, in which the outputs of the network can be considered as the normalized firing rates of the neuron within a certain time interval, SN models make direct use of the temporal information of the individual spikes whereas the magnitude contains no information.

In one of the leading works on the artificial neural networks utilizing SN models, Maass [5] states that a network consisting of spiking neurons can be applied to any problem that is solvable by the first two generations and this network has at least the same computational power as the neural networks consisting of perceptrons or sigmoidal neurons. Furthermore, it has been shown that spiking neural networks (SNNs) are capable of processing a considerable amount of data with a relatively small number

of spikes [6]. The above-mentioned advantages and an increasing interest in temporal computation have encouraged the use of SNNs in a wide variety of applications, including classification [7–10], pattern recognition [11, 12] and control [13–16]. In this dissertation, SNNs are considered for the identification and the control of dynamic plants.

The supervised learning of SNNs has been considered in many research works [17] and most of these have concentrated on the development of gradient ascent/descent-based learning algorithms, as in the case of the traditional ANNs. In [7, 18], a gradient descent-based learning algorithm called *SpikeProp* is presented and received wide acceptance as the proposed method is analogous to the backpropagation algorithm [19] used in the learning of the second generation neural networks. To improve the performance of SpikeProp, a number of modifications are proposed to the basic algorithm such as inclusion of a momentum term [20], QuickProp [21], or enabling multiple firing [22]. Although the modified learning methods yield an increase in the convergence properties to some extent, as with most of the gradient-based learning algorithms, the convergence speed is still relatively slow compared to other approaches and the algorithm can be easily trapped in local optima [8]. Furthermore, a number of numerical robustness issues also need to be taken into account when recursive estimation algorithms are applied over long periods of time [23].

The variable structure systems (VSSs) approach provides the means for a robust and stable control system through a comprehensive stability analysis and its high performance in the control of systems confronted with uncertainties and imprecision promotes the research activities in developing new VSS-based training algorithms to be used in computationally intelligent structures [24–29]. In this study, the VSSs theory has been utilized in the derivation of parameter update rules for SNNs. Stable online tuning of the parameters and a fast learning speed are the prominent characteristics of the proposed algorithm. Unlike the gradient descent-based learning algorithms, which aim to minimize an error function, in the proposed approach a sliding surface is estab-

lished using the SNN controller parameters. The Lyapunov stability method is adopted in the derivations of the conditions to enforce the learning error toward zero in a stable manner.

Regarding the fact that every intelligent control technique has some advantages and disadvantages making them suitable for specific applications, hybrid systems combining fuzzy logic and neural networks have been widely used in a variety of classification and control applications. Neural networks are very effective in determining a mapping between the inputs and outputs of a system, but they have a "black box" nature. This gives rise to the lack of knowledge about the causal relationships between these inputs and outputs. The parameters that have the most impact on a particular output cannot be distinguished among others, and thus neural networks fail to provide an insight into the system dynamics. On the other hand, fuzzy logic systems are good at explaining how they reach their decisions but they require expert knowledge to form the rules and the membership functions that they use to make these decisions. However, if the related knowledge is deficient, erroneous or inexact fuzzy systems may require the adjustment of their membership functions in order to be able to provide the proper output. The tuning of the parameters is generally conducted in a heuristic way due to the lack of any formal approach for it. This heuristic approach might be very time consuming and error-prone. One of the most effective remedies to overcome this shortcoming is the use of a hybrid structure in which the learning ability of neural networks can be utilized to automate the parameter update process and reduce development time and cost substantially while improving performance.

The main goals of this thesis work can be stated as follows:

(i) *To develop learning algorithms for SNNs that will result in a high degree of accuracy with acceptable execution times for real-time applications*

This is considered to be still an open research topic in the literature. The approach followed in achieving this goal is basically the adaptation of the VSSs

theory-based learning algorithms for use in SNNs.

(ii) *To combine fuzzy logic and SNNs in a hybrid intelligent structure and to adapt the learning rules developed in (i) for this structure*

In the hybrid structure, a neural network consisting of spiking neurons is used to adapt the mean and variance values of the fuzzy membership functions. To achieve robust system response in handling the uncertainties and imprecision along with a faster convergence than the traditional learning techniques, the VSS-based approach has been followed in the derivation of update rules.

(iii) *To verify the performance of the proposed algorithms through simulated and experimental studies*

The effectiveness of the proposed approach has been illustrated with a set of experiments in which the developed algorithms are applied to various control tasks on different dynamic systems including highly nonlinear and uncertain environments such as the Antilock Braking System (ABS) and the two-dof direct drive manipulator.

1.1. Organization of the Thesis

This dissertation can be roughly divided into four parts. An introduction to the Spiking Neural Networks along with the basic concepts of the two former generations of Artificial Neural Networks have been presented in the first part (Chapter 2). In the second part (Chapter 3), gradient descent-based learning algorithms have been derived and tested for the identification and the control of the dynamic plants commonly used in the literature. In the third part (Chapter 4), the VSS-based learning approach for SNNs has been introduced and its properties are analyzed. In the last part (Chapter 5), a hybrid structure combining fuzzy reasoning and SNNs has been presented and VSS-based learning algorithms for this hybrid structure have been derived.

Chapter 2 introduces the Spiking Neural Networks along with a brief description of the available models of spiking neurons. The literature on the supervised and un-

supervised learning of SNNs has been reviewed. Sample real-life applications of SNNs have been presented.

In Chapter 3, the gradient descent-based learning method for SNNs has been introduced, the notational conventions and the method assumptions are presented.Through the simulation studies and real-time experiments conducted on a laboratory servo system, the potential of the SNNs with gradient-based learning algorithms has been illustrated.

In Chapter 4, the VSS theory-based approach for on-line learning as applied to SNNs has been introduced. The underlying idea of Sliding Mode Control theory and how this idea is reflected in the derivation of the learning rules have been presented. The proof of the learning process convergence is presented using Lyapunov stability method. The applications of this approach for identification and control tasks have been described.

In Chapter 5, the advantages as well as the disadvantages of artificial neural networks and fuzzy systems have been discussed. A hybrid structure combining fuzzy reasoning and SNNs has been presented and VSS-based learning algorithms for this hybrid structure have been derived.

2. SPIKING NEURAL NETWORKS

2.1. Biological Neurons

Artificial neurons and neural networks try to imitate the working mechanisms of their biological counterparts. Although the biological neurons, which are the actual processing units of the brain, may differ in their size and shape from each other, they all consist of four distinct parts called dendrite, soma, axon, and synapse (See Figure 2.1). The dendrites of a postsynaptic neuron collect electrical signals coming from the neighboring presynaptic neurons and transmit these signals to the soma of the postsynaptic neuron. If the weighted sum of the incoming signals is above a certain threshold value, then a spike is generated and propagated along the axon and its branches to other neurons. When this spike arrives at a synapse, which refers to the connection between the axon branch of a presynaptic neuron and the dendrite of a postsynaptic neuron, it triggers a complex chain of biochemical reactions. These biochemical processes give rise to a change in the postsynaptic membrane potential. This process is relatively slow and it is associated with a certain delay specific to that synapse.

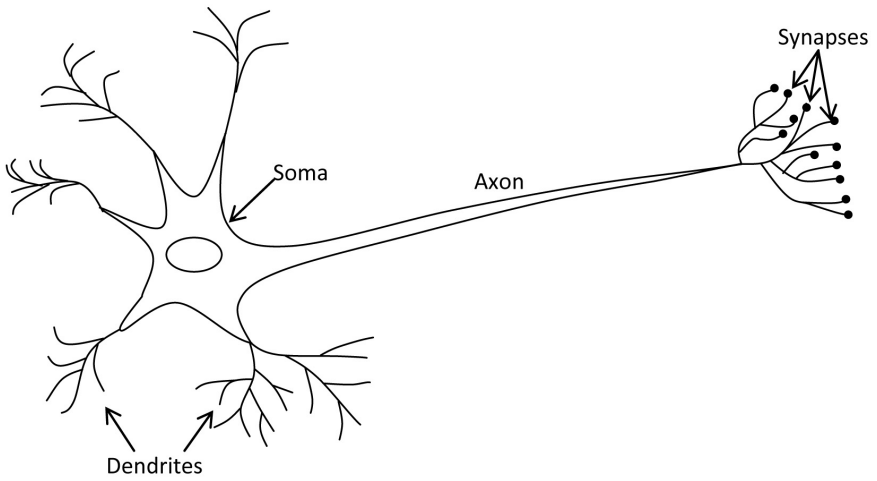


Figure 2.1. Simplified model of a biological neuron.

Depending on the type of synapse this change can be positive, raising the post-synaptic neuron's potential, or negative, lowering it. When it raises the potential and thereby can cause the neuron to fire, the synapse is called excitatory. Conversely, if the change is negative making it harder for that neuron to fire, the synapse is called inhibitory. The type of the synapse, inhibitory or excitatory, can never change, but the intensity of the potential change it causes can be altered. This effect called synaptic plasticity enables the network to learn from past experience. The effect of the potential change is only temporary and it fades away after a while. The time required for the neuron to be able to fire another spike is called the refractory period.

2.2. Generations of Artificial Neural Networks

The structure of an artificial neuron is analogous to its biological counterparts. Artificial neurons also have parts that receive, collect, process, and propagate signals. These are namely inputs, weights, summing junction, activation function and output. Based on the employed activation function and the information processing mechanism, ANNs are generally considered to have three generations [5].

In 1943, McCulloch and Pitts proposed a model based on simplified binary neurons, where a single neuron implements a simple thresholding function [2]. In this model, the neurons act as simple integrate-and-fire units which fire a spike if the weighted sum of the incoming signals exceeds some threshold value. The output of the postsynaptic neuron returns a value of 1 if the neuron fires and 0 otherwise. The model omits the temporal information of the incoming spikes; i.e. it is assumed that all the inputs of the postsynaptic neuron are synchronous.

Remarkably, networks of such simple computational elements can implement a range a mathematical functions relating input states to output states and can learn to classify a series of inputs if these inputs are linearly separable. Artificial neural networks consisting of McCulloch-Pitts threshold neurons are regarded as the first

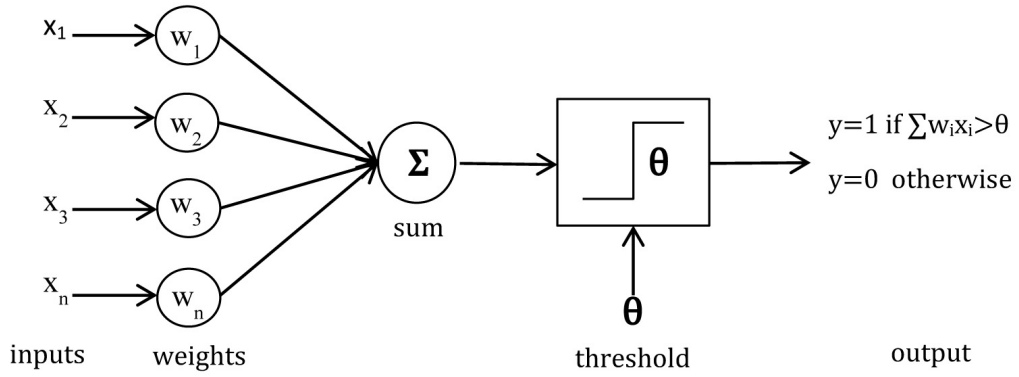


Figure 2.2. Artificial neuron model of the first generation ANN.

generation of artificial neural networks.

The limitations of these early artificial neural networks had been discovered quite quickly. As stated by Minsky and Papert in [30] the perceptron could not learn even simple non-linear examples such as XOR or parity. To alleviate these issues, in the second generation of artificial neural networks, the step threshold function has been replaced by a continuous activation function resulting in a graded response to changes in its inputs and therefore enabling the use of artificial neural networks in systems with analog inputs and outputs. Commonly used examples of activation functions are the sigmoid and hyperbolic tangent.

With the introduction of sigmoidal artificial neurons and learning rules for training networks consisting of multiple layers of neurons [3, 19], some of the shortcomings of the earlier neural networks were overcome. The most noticeable example was the ability of the network to learn XOR function where the inputs are not linearly separable.

2.3. Pulse Coding

The outputs of the neuron models of the first two generations typically lie between zero and one, which can be considered as the abstraction of the normalized firing rates or frequencies of the neuron within a certain period of time. This type of coding is

commonly referred to as rate coding. In this coding scheme a higher rate of firing corresponds to a higher output signal and the exact timing of the spikes are neglected.

The rate coding scheme found its roots in a number of research studies on the biological neurons. In one of the best known examples [31], it is shown that when the external mechanical pressure on the frog skin increased, the receptors on the frog skin respond with more spikes. It was generally believed that this was the only information that can be passed between two biological neurons [32].

Rate coding has been the dominant paradigm in neurophysiology for many years. However, recent studies suggest that neurons encode information in the precise timing of single spikes and not just only in their average firing frequency. By placing a fine electrode close to the soma or axon of a neuron the neuronal signals can be observed. A typical recording consists of a sequence of short pulses, called action potentials or spikes. The duration of an action potential is typically in the range of 1-2 ms with an amplitude of about 100mV. As all spikes of a neuron are similar to each other, the form of the action potential does not carry any information; rather, the information is encoded in the number and the timing of spikes [5].

One of the arguments supporting temporal coding over rate coding states that many behavioral responses are completed too quickly and there is not enough time for the average calculations which are essential for the rate coding [6]. This suggests that in neural systems where the processing speed is required to be high, the timing of individual spikes (especially the firing time of the first spike in most cases) comprises the means for transmitting information among neurons. Another argument states that the firing rate is not always sufficient to capture the entire information content. For instance, many auditory cortical neurons in anesthetized monkeys do not change their mean firing rates during the ongoing phase of continuous stimuli although the relative timing of their action potentials vary with the changing stimulus frequency [33].

Another advantage of pulse coding is that as the firing times of the neurons are known, the average firing rate can be computed easily. On the other hand, use of rate coding schemes leads to the loss of the temporal information about individual spikes and the ability to encode complex spike trains is reduced significantly.

2.4. Models of Neurons

Spiking neural networks are considered as the third generation of artificial neural networks and try to enhance the level of biological realism by utilizing the spiking neuron models [4]. Unlike the first two generations, in which the outputs of the network can be considered as the normalized firing rates of the neuron within a certain time interval, SN models make direct use of the temporal information of the individual spikes whereas the magnitude contains no information.

To formulate the information transmission between spiking neurons, various neuron models have been proposed throughout the literature.These neuron models differ from each other with regard to their degree of biological accuracy and computational efficiency. In the following subsections some commonly used spiking neuron models will be described.

2.4.1. Hodgkin-Huxley Model

This neuron model has been formed from the experimental results conducted on the squid giant axons and the authors Alan L. Hodgkin and Andrew F. Huxley received a Nobel Prize for this work, in which the electrochemical information transmission between the biological neurons has been modeled with electrical circuits consisting of capacitors and resistors. Based on the experiments with the giant axon of the squid, three types of ion channels have been identified: leakage channel, sodium (Na) channel and potassium (K) channel. The leakage channel is characterized by the voltage-independent conductance g_L whereas the conductance characteristics of the two other

channels (g_{Na} and g_K) depend on time and voltage.

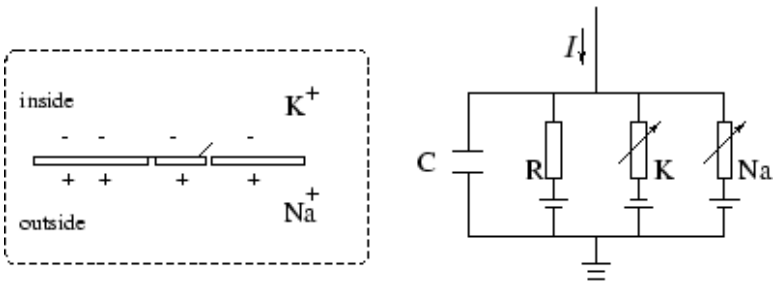


Figure 2.3. Hodgkin-Huxley neuron model [1].

Figure 2.3 shows the Hodgkin-Huxley Model represented by electrical RC circuit. The capacitor C_m refers to the cell membrane and each ion channel is demonstrated by a resistance. The resistances corresponding to the potassium and sodium channels are denoted by variable resistors. The dynamics of this model can be described by the following formula:

$$C_m \frac{du}{dt} = -g_{Na} m^3 h(u - E_{Na}) - g_K n^4 (u - E_K) - g_L(u - E_L) + I(t) \qquad (2.1)$$

where E_L, E_{Na} and E_K refer to the reversal potentials of the leakage channel, sodium channel and potassium channel, respectively. Furthermore, the variables $m(t)$, $h(t)$ and $n(t)$ are employed to describe the probability that the corresponding voltage dependent channel is open at the time instant t. These variables evolve in time with the following dynamics:

$$\frac{dp(t)}{dt} = \alpha_p(u_m)(1 - p(t)) - \beta_p(u_m)p(t), \qquad where \quad p = m, n, h \qquad (2.2)$$

If the parameters of the HH model can be calibrated appropriately, it can produce very realistic results. However, this neuron model is too complex to be used in the simulations of SNNs.

2.4.2. Izhikevich Model

This model can be considered as a simplification of the comprehensive Hodgkin-Huxley Model to a two-dimensional system described by the following differential equations:

$$\frac{dv(t)}{dt} = 0.04v^2 + 5v + 140 - u + I(t) \tag{2.3}$$

$$\frac{du(t)}{dt} = a(bv - u) \tag{2.4}$$

where $v(t)$ represents the membrane potential of the neuron and $u(t)$ stands for the membrane recovery variable providing negative feedback to v(t). The parameter a describes the time scale of $u(t)$ whereas b is used to denote the sensitivity of $u(t)$ to the fluctuations in $v(t)$ [34]. $I(t)$ is the constant current input.

When the membrane potential $v(t)$ reaches the peek value of $30mV$, a spike is emitted and the following function is used to reset $v(t)$ and $u(t)$:

$$If \;\; v(t) \geq 30mV, \;\; then \;\; v = c \;\; and \;\; u = u + d \tag{2.5}$$

The most important advantage of this model is that with the adjustment of only four parameters, results as realistic as those of the HH model can be obtained.

2.4.3. Leaky Integrate-and-Fire Model

The Leaky-integrate-and-fire (LIF) neuron can be considered as the most widely used and well-known spiking neuron model, in which the cell membrane is modeled by a simple RC electrical circuit. In this circuit, the capacitor C is placed in parallel with

a resistor R and it is driven by an input current I(t). Regarding Figure 2.4, a spike sent out by a presynaptic neuron is passed first through a low pass filter to generate current pulse $I(t-t_i)$, which will be fed to the integrate-and-fire circuit. The dynamics of the membrane potential $u(t)$ measured across the capacitor C can be described by the following formula:

$$C_m \frac{du}{dt} = -\frac{1}{R}\left(u(t) - u_{rest}\right) + I(t) \tag{2.6}$$

where u_{rest} refers to the resting potential of the neuron.

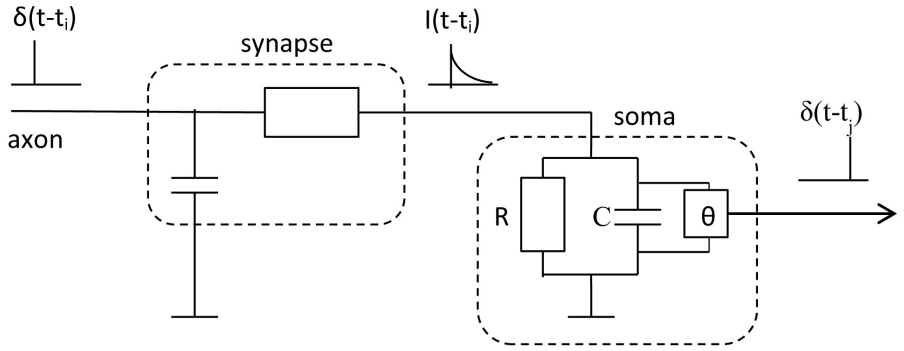


Figure 2.4. Leaky Integrate-and-fire neuron model.

When the voltage $u(t)$ reaches the threshold value θ, the neuron fires a spike and the corresponding time instant t_j is recoded as the firing time of the postsynaptic neuron. The word "leaky" refers to the case where the value of the membrane potential is immediately set back to u_{rest} after firing and the integration starts from this point again. In this model, the shape of the action potentials is neglected and each spike is defined only by the time of firing.

2.4.4. Spike Response Model

The Spike Response Model differs from the aforementioned voltage dependent models in the sense that it utilizes the time elapsed since the last action potential to formulate the membrane potential. The model is widely exploited because of its

mathematical simplicity and its ability to approximate the realistic Hodgkin-Huxley model by selecting an appropriate spike response function [1].

In this model, the resting potential of the neuron in the absence of any external stimuli is denoted by E_m. The spike t_i emitted by the presynaptic neuron i causes a variation in the membrane potential $x_j(t)$ of the postsynaptic neuron j. (To emphasize that the membrane potential is computed, based on the elapsed time and not on the voltage value, $x_j(t)$ is preferred instead of $u_j(t)$ to denote the membrane potential) Despite the fact that there are many different mathematical formulations available for the spike functions $\varepsilon(s)$, where s refers to $s = t - t_i$, the general shape of the spike response function consists of a short rising part and a long decaying part.

The postsynaptic neuron emits a spike when its membrane potential $x_j(t)$ crosses the predetermined threshold value θ for the first time (that is $x_j(t) \geq \theta$). Thus, the firing time t_j is a nonlinear function of the state variable x_j: $t_j = t_j(x_j)$. After firing a spike, the membrane potential $x_j(t)$ is set to a very low value to avoid an immediate second spike. This refractoriness period is modeled by $\eta(t - t_j)$.

The dynamics of the postsynaptic membrane potential $x_j(t)$ can be described by the following formula:

$$x_j(t) = \Sigma_{t_j \in F_j} \eta(t - t_j) + \Sigma_{i \in \Gamma_j} w_{ij} \Sigma_{t_i \in F_i} \varepsilon(s) \tag{2.7}$$

where F_j is the vector consisting of the firing times t_j of postsynaptic neuron j, Γ_j is the set of presynaptic neurons of neuron j, w_{ij} is the weight of the connection between presynaptic neuron i and postsynaptic neuron j and F_i is the vector consisting of the firing times t_i of presynaptic neuron i

2.5. Spiking Neural Networks

The general structure of a Spiking Neural Network is presented in Figure 2.5. The presented structure resembles the Multilayer Perceptron Model of the second generation. Similar to that generation, the network consisting of input, output and hidden layers is considered to be fully connected. The number of hidden layers can be altered at will. The information flow is unidirectional (from the input neurons to the output units) and there is no feedback among the connections.

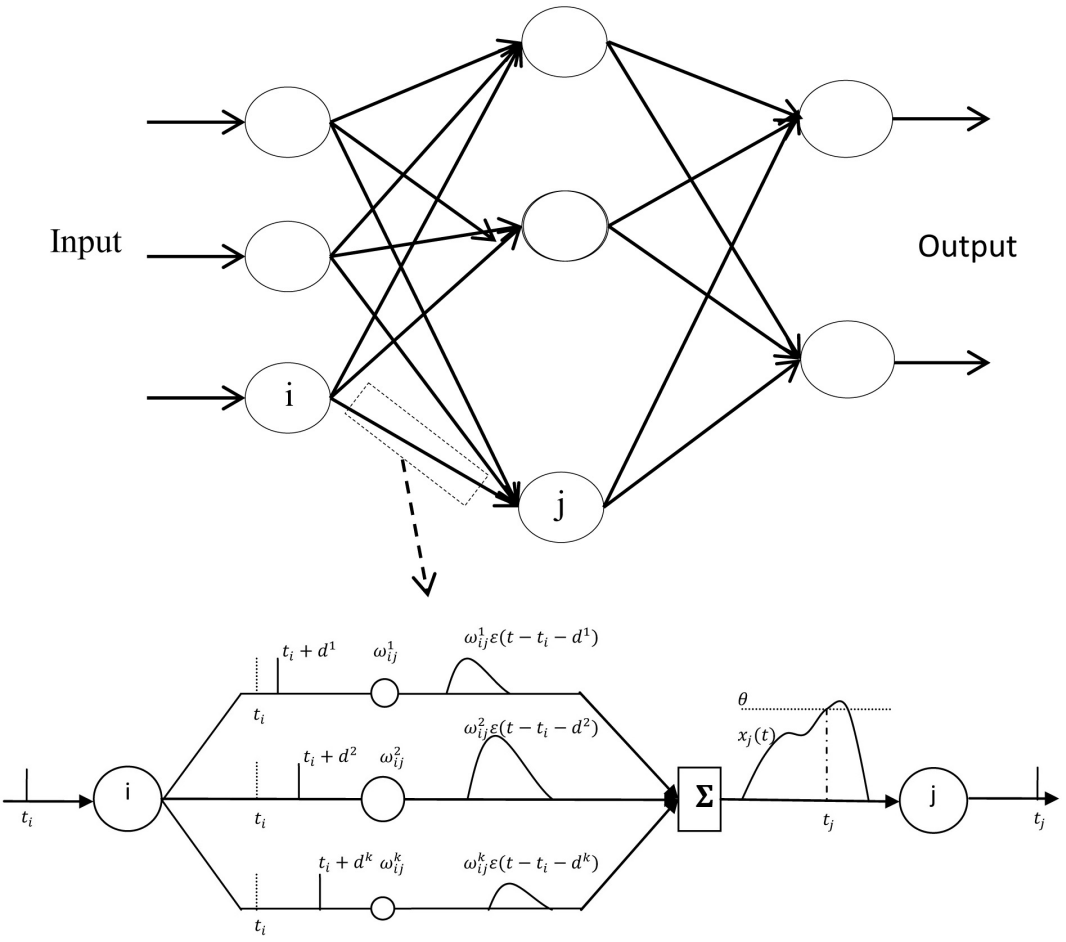


Figure 2.5. The structure of a SNN.

Apart from these similarities, there are some fundamental differences between the networks of these two generations. The first and most important difference is that the inputs and outputs of a SNN are the temporal information of the spikes. Furthermore,

as mentioned earlier, studies on the biological neurons [35,36] show that the spikes of a neuron have similar shapes and the information is encoded in the number and precise timing of the spikes rather than in their form. Neural networks consisting of spiking neurons attempt to imitate this phenomenon by using the timing of the spikes as the means of communication and neural computation instead of utilizing their magnitudes, i.e. the inputs of a SNN are the temporal information of the spikes whose magnitude information is omitted.

Unlike the network structures of the second generation, in SNN multiple synapses are involved in each connection between a presynaptic and a postsynaptic neuron. In this model each synapse is associated with a different delay and adjustable weight. The different delay times are introduced so that the spike emitted by a presynaptic neuron will have an effect on the postsynaptic neuron potential for a longer period of time.

The membrane potential of a spiking neuron is modeled by a dynamic variable and works as a leaky integrator of the incoming spikes: newer spikes contribute more to the potential than older spikes. Suppose that a presynaptic neuron i fires a spike at time t_i. This spike will be transmitted to the postsynaptic neuron j by the k^{th} synapse at time t_i+d^k, where d^k denotes the delay for the k^{th} synapse. To describe the resulting membrane potential of the postsynaptic neuron j at time t, the SRM model is used owing to its low computational cost and comparable level of biological plausibility. Throughout this study, it is assumed that each neuron is capable of generating a single spike during the simulation interval when its membrane potential reaches a predefined threshold value θ for the first time. This assumption eliminates the need for the refractoriness term. The postsynaptic membrane potential corresponding to the altered SRM can be expressed as follows:

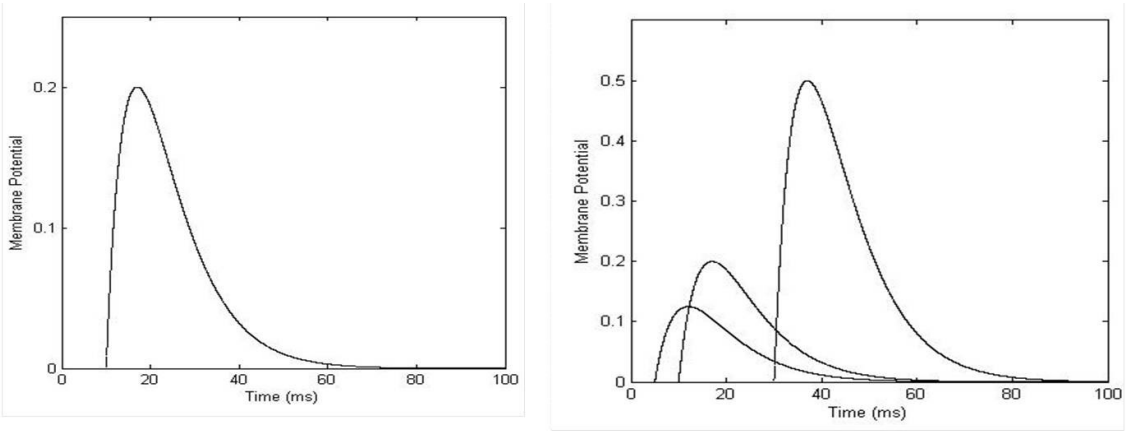
$$\begin{aligned} x_j(t) &= \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ij}^k \varepsilon\left(t - t_i - d^k\right) \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ij}^k y_i^k(t) \end{aligned} \tag{2.8}$$

In this formulation N corresponds to the number of presynaptic neurons, whereas K denotes the number of synapses. d^k and w_{ij}^k are the delay and the weight coefficient for the k^{th} synapse between the presynaptic neuron i and the postsynaptic neuron j, respectively.

The spike response function $\varepsilon(t)$ describes the effect of incoming spikes on the internal state of the postsynaptic neuron. Despite the fact that there are many different mathematical formulations available, the general shape of the spike response function consists of a short rising part and a long decaying part. The formulation given in [7] has been adopted in this study.

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}} & t > 0 \\ 0 & t \leq 0 \end{cases} \tag{2.9}$$

The decay time constant τ models the rise and decay time of the membrane potential.



(a) (b)

Figure 2.6. Postsynaptic membrane potential. (a) for one synapse, (b) for three synapses.

2.6. Encoding Continuous Inputs into Spike Times

As mentioned earlier, the inputs and the outputs of SNNs are described by the spike times. To be able to use SNNs in engineering applications, some methods should

be developed for the transformation between analog signals and spike times. The three main approaches to encode information into spikes are rate coding, population coding, and delay coding [37].

In the rate coding, the sensor analog value is mapped to the firing rate of the neuron. In this coding scheme, a higher value of the sensor signal corresponds to a higher frequency or rate of spike firing, whereas smaller signal values are associated with low frequencies of firing [38]. The precise timing of the spikes is not distinctive (e.g. two neurons with different firing frequencies can both fire a spike at the time instant $t = t_a$.); rather the number of spikes within a certain period of time is determined. In the proposed algorithm, the information of the exact timing of the spikes was needed; therefore this encoding method has not been used in this study.

In population coding, the sensor signal is distributed over a population of neurons instead of a single neuron. In this method, the entire input space is covered with overlapping activation functions, which are usually Gaussian receptive fields [18]. For a sensor signal n, m neurons will provide a non-zero value. Neurons with the highest value fire earlier, whereas smaller values result in late firing times. In this study, this approach was not utilized as it is computationally much more costly than the other methods. (If there are N inputs and m neurons to encode each input, then (N x m) firing times - the inputs of the SNN - should be calculated at each control cycle.)

In delay coding, the strength of the sensor value is encoded in the firing delay of the neuron. Thus, a spiking neuron receiving weaker input signals tends to fire a spike later than the same neuron receiving higher input signals. This method has been used in many studies, including this one, owing to its simplicity and reduced computational burden [39–41]. It arises from the fact that a spiking neuron receiving weaker input signals tends to fire a spike later than the same neuron receiving higher input signals.

To convert input values into the spike times to be entered to the SNN, the fol-

lowing formula has been used:

$$t_j(x) = t_{max} - round\left(t_{min} + \frac{(x - x_{min})(t_{max} - t_{min})}{x_{max} - x_{min}}\right) \tag{2.10}$$

where $t_i(x)$, t_{min} and t_{max} are the current, the minimum and the maximum spike times, and x, x_{min} and x_{max} are the current, the minimum and the maximum values of the input variables, respectively. Regarding the mapping stated in Equation 2.10, first the maximum allowable time window is determined after which a neuron can't fire a spike. The minimum spike time describes the minimum time required for the neuron to be activated. Thus if a sensor signal has its rated value, then it will fire at $t = t_{min}$. Similarly, if the sensor signal has a minimum value, it will fire at $t = t_{max}$. Similarly, other signal values have been mapped linearly to the corresponding delay times.

2.7. Learning in Spiking Neural Networks

2.7.1. Unsupervised Learning

In the unsupervised learning of SNNs, the network discovers patterns, clusters, and regularities from the characteristics of the given data in the absence of any feedback from the environment to dictate what the output firing times should be for a given input spike train. In general, the complexity of the unsupervised learning algorithms and their computational cost are lower than the supervised algorithms since the need for multiple iterations through the training dataset has been eliminated.

The unsupervised learning algorithms for SNNs are mostly based on Hebbian learning, which states that the weight of the connection between two neurons should be strengthened if the neurons fire simultaneously. In one of the earliest work, the stimuli were presented by the precise timing of spikes and the spike pattern was encoded in the synaptic delays. In this network, a neuron fires if the input spike pattern is similar to the spike pattern encoded as the center of the RBFs. In [42], a Winner-Takes-All

(WTA) rule based on Hebbian learning is utilized to modify the weights between the presynaptic neuron and the neuron with the earliest firing time in the subsequent layer.

Another commonly used approach, Spike-timing dependent plasticity (STDP), can be considered as a variation of the Hebbian learning rule with a spike-based formulation. The underlying idea of this approach is that if a presynaptic neuron fires a spike just before the postsynaptic neuron, then the corresponding weight value of the synapse will be increased. Otherwise, if the presynaptic spike has been observed after the postsynaptic spike then this synapse will be weakened. In [43], STDP learning is used in an image recognition application. In that study, at each step, one of the images is presented to the network which triggers a single spike in each neuron of the input layer. These spikes are propagated along the network and the synaptic weight of the first firing neuron is increased. As a result, each neuron in the output layer learns a single category.

Although many successful applications for the Hebbian learning rule have been reported, the key problem of Hebbian learning is that for a dominant signal, the synaptic weights will increase exponentially leading to instability.

2.7.2. Supervised Learning

The main goal of supervised learning in SNNs can be stated as determining the weight values of the synaptic connections which will result in an output spike train that is close to the desired spike train for the given input spike train. If we recall the supervised learning of the second generation ANNs, we can state that the gradient descent-based learning algorithms utilizing error backpropagation is the dominant method for training. Adopting this approach to SNNs was not possible until recently because of the discontinuous-in-time nature of spiking neurons. In [7] this problem has been overcome by assuming that each neuron can emit a single spike during a simulation cycle. The membrane potential of the postsynaptic neuron after this time

instant has been neglected. This way the discontinuity of the membrane potential has been avoided. The proposed algorithm called SpikeProp is widely used among the researchers due to its analogy to the well known backpropagation algorithm.

In [44], it is shown that the performance of the SpikeProp algorithm highly depends on the initial values of the synaptic weight. Furthermore, it is shown that it is possible to obtain good results even if large values of leaning rates and/or negative weight values are used. These results contradict the findings of [7]. To improve the performance of SpikeProp, a number of modifications have been proposed to the basic algorithm such as inclusion of a momentum term [20], QuickProp [21], or enabling multiple firing [22]. In [45], besides the synaptic weights, the delays associated with each synapse, time constants, and threshold value have also been adapted. Although the modified learning methods yield an increase in the convergence properties to some extent, as with most of the gradient-based learning algorithms, the convergence speed is still relatively slow compared to other approaches and the algorithm can be easily trapped in local optima [8]. Furthermore, a number of numerical robustness issues also need to be taken into account when recursive estimation algorithms are applied over long periods of time [23].

Besides Gradient-based learning algorithms, evolutionary approaches have also been utilized for the adaptation of the network parameters. The advantage of these methods is that they do not require derivatives, which is advantageous in SNNs because of the discontinuous nature of membrane potential. In [9], evolutionary strategies have been used to update the synaptic weights and the algorithm has been tested on the benchmark XOR and Iris classification problems. In [46], the parallel differential evolution algorithm has been tested on the benchmark classification problems. In [40], the genetic algorithm has been employed to evolve the weights of the SNNs online using real robots. Although the results presented in these studies are promising, as with all evolutionary algorithms the evolutionary process is very time-consuming and this makes these algorithms unsuitable for online learning.

3. GRADIENT-BASED LEARNING ALGORITHMS FOR SPIKING NEURAL NETWORKS

Before the derivation of the sliding mode control (SMC)-based learning algorithms, it will be beneficial to have a look at the gradient based learning algorithm SpikeProp developed by Bohte [7]. The importance of this algorithm relies on the fact that it enabled the application of the backpropagation algorithm for SNNs. The features, capacity, and capability of the backpropagation algorithm have been exhaustively investigated, thus the adaptation of this algorithm for the new emerging SNNs has been considered as a valuable asset and accelerates the research activities in the field. In this study, to assess the capability of the Spikeprop, simulated and experimental studies were conducted on different plants for identification and control problems.

3.1. Derivation of the Learning Rules

Consider a two-layered fully connected feedforward spiking neural network with the set of neurons in the input layer denoted by H, in the hidden layer by I and in the output layer by J. The neurons are modeled using SRM to reduce the computational burden and it is considered that each neuron can fire a single spike during a simulation cycle. As mentioned earlier, the goal of the supervised learning algorithm is to teach the network to generate the desired output spike train for a given input spike train. Similar to the traditional neural networks, the first step was to develop the definition of a cost function to be minimized :

$$E_T = \frac{1}{2}\sum_{j\in J}(t_j - t_j^d)^2 \tag{3.1}$$

where t_j and t_j^d stands for the actual and desired firing times of the output layer neurons, respectively.

To make use of the backpropagation algorithm, we need to calculate

$$\Delta w_{ij}^{k} = -\eta \frac{\partial E_T}{\partial w_{ij}^{k}} \tag{3.2}$$

where w_{ij}^{k} denotes the synaptic weight between the hidden layer neuron i and the output layer neuron j for the kth synapse. The gradient of the cost function with respect to the weight values can be written using the chain rule as follows:

$$\frac{\partial E_T}{\partial w_{ij}^{k}} = \frac{\partial E_T}{\partial t_j} \frac{\partial t_j}{\partial x_j(t_j)} \frac{\partial x_j(t_j)}{\partial w_{ij}^{k}} \tag{3.3}$$

For the detailed derivation of the partial derivative terms, readers are encouraged to refer to [7]. Equation 3.3 can be written as:

$$\begin{aligned} \frac{\partial E_T}{\partial w_{ij}^{k}} &= -y_i^{k}(t_j) \frac{(t_j - t_j^{d})}{\sum_{i \in \Gamma_j} \sum_{k} w_{ij}^{k} \frac{\partial y_i^{k}(t_j)}{\partial t_j}} \\ &= y_i^{k} \delta_j \end{aligned} \tag{3.4}$$

Thus, the weight update rule can be stated as:

$$\Delta w_{ij}^{k} = -\eta y_i^{k}(t_j) \delta_j \tag{3.5}$$

which is analogous to the weight update rule of the Multilayer Perceptron (MLP) model.

Next comes the derivation of the weight update rules for the preceding layer. The generalized delta error for the hidden layer has been defined as follows:

$$\begin{aligned}\delta_i &= \frac{\partial E_T}{\partial t_i}\frac{\partial t_i}{\partial x_i(t_i)} = \frac{\partial t_i}{\partial x_i(t_i)}\frac{\partial E_T}{\partial t_i} \\ &= \frac{\partial t_i}{\partial x_i(t_i)}\sum_{j\in J}\frac{\partial E_T}{\partial (t_j)}\frac{\partial (t_j)}{\partial x_j(t_j)}\frac{\partial x_j(t_j)}{\partial t_i} \\ &= \frac{\partial t_i}{\partial x_i(t_i)}\sum_{j\in J}\delta_j\frac{\partial x_j(t_j)}{\partial t_i}\end{aligned} \tag{3.6}$$

which can be expressed as:

$$\delta_i = \frac{\sum_{j\in J}\delta_j\left\{\sum_{k=1}^{m} w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_i}\right\}}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{3.7}$$

The weight adaptation rule for the hidden layer can be stated as in a similar manner to the traditional backpropagation algorithm.

$$\Delta w_{hi}^k = -\eta y_h^k(t_i)\delta_i \tag{3.8}$$

3.2. Application of SpikeProp for Identification and Control Tasks

In this study, it was intended to verify the performance of the proposed algorithm through simulated studies and real-time experiments. For the simulations, the computational time of an algorithm might be not very restrictive, but it plays a crucial role in the real time experiments. Real time applications require the algorithm to be computed within strict time constraints, i.e. the algorithm should be completed before the deadline. Under this restriction, the computational cost and complexity of an algorithm are of great importance. It's worth mentioning that the partial derivatives of the preferred spike response function with respect to the firing times possess some interesting properties which enhance their applications in real-time systems. For ex-

ample, if we consider the partial derivative of $\varepsilon(t_j - t_i - d^k)$ with respect to t_j, then this partial derivative simply results in:

$$\frac{\partial \varepsilon(t_j - t_i - d^k)}{\partial t_j} = \varepsilon(t_j - t_i - d^k)\frac{\tau - (t_j - t_i - d^k)}{\tau(t_j - t_i - d^k)} \tag{3.9}$$

As the values of $\varepsilon(t_j - t_i - d^k)$ and $(t_j - t_i - d^k)$ have been already computed, the computational cost to calculate its partial derivative is considerably low. Similarly, the partial derivative of $\varepsilon(t_j - t_i - d^k)$ with respect to t_i yields:

$$\frac{\partial \varepsilon(t_j - t_i - d^k)}{\partial t_i} = -\varepsilon(t_j - t_i - d^k)\frac{\tau - (t_j - t_i - d^k)}{\tau(t_j - t_i - d^k)} = -\frac{\partial \varepsilon(t_j - t_i - d^k)}{\partial t_j} \tag{3.10}$$

which facilitates the derivation of the parameter update rules considerably.

3.2.1. Identification Performance Studies

The identification problem involves the determination of the relation between the inputs and the outputs of a system. In Figure 3.1, the structure of the identification scheme has been shown. The inputs to the SNN identifier are the external input signal, its one-, two-, ..., d_i - step delayed values and the one-, two-, ..., d_o - step delayed outputs of the plant. Here the problem is to determine the values of the SNN parameters for which the difference between plant output y and network output y_n will be minimum for all input values of u.

3.2.1.1. Example 1. The following nonlinear plant given by the difference equation is considered as an example for the identification problem [47]:

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

where $y(t), y(t-1), y(t-2)$ are the current, the one- and the two-step delayed outputs respectively, and $u(t-1), u(t-2)$ and $u(t-3)$ are the inputs of the plant.

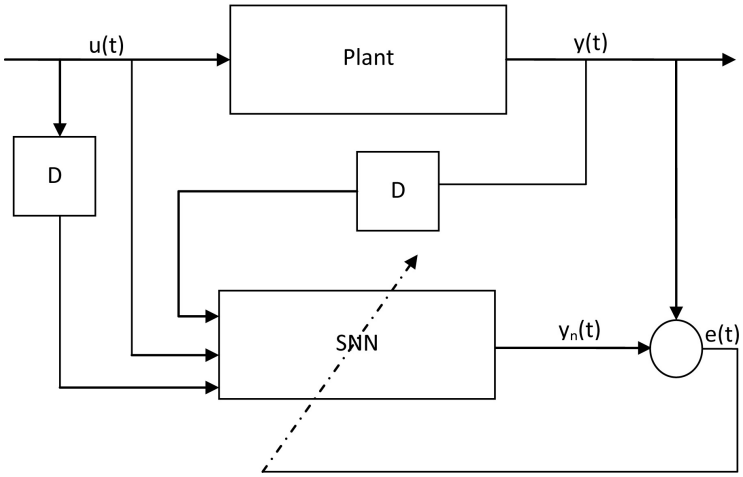


Figure 3.1. Identification scheme.

The one layered SNN is used for identification, where the current output of the plant depends on the previous input and the output signals. The initial weight values of the SNN are set randomly within the interval $[0, 1]$. Using the parameter update rules stated in Section 3.1, these weights are updated for a given excitation signal of $u(t) = sin(\pi * t/10)$. The training of the SNN is carried out for 200 epochs. For the simulation studies, the one-step delayed state of system $y(t-1)$ and the one-step delayed control signal $u(t-1)$ are fed into the SNN as inputs, thus, the identification is performed with 2 inputs and 8 synapses in each connection. As a performance criterion the Root Mean Squared Error (RMSE) is used. The initial value of the learning rate η is taken as 0.0001.

After training, the following test signal is used to see the identified results:

$$u(t) = \begin{cases} sin(\frac{\pi t}{25}) & t < 250 \\ 1 & 250 \leq t < 500 \\ -1 & 500 \leq t < 750 \\ 0.3sin(\frac{\pi t}{25}) + 0.1sin(\frac{\pi t}{32}) + 0.6sin(\frac{\pi t}{10}) & 750 \leq t < 1000 \end{cases} \tag{3.12}$$

The RMSE value with 8 synapses for training is 0.064 and for testing 0.068.

Figure 3.2 shows the on-line identification performance of the SNN. Here the solid line is the plant output and the dashed line is the SNN identifier output.

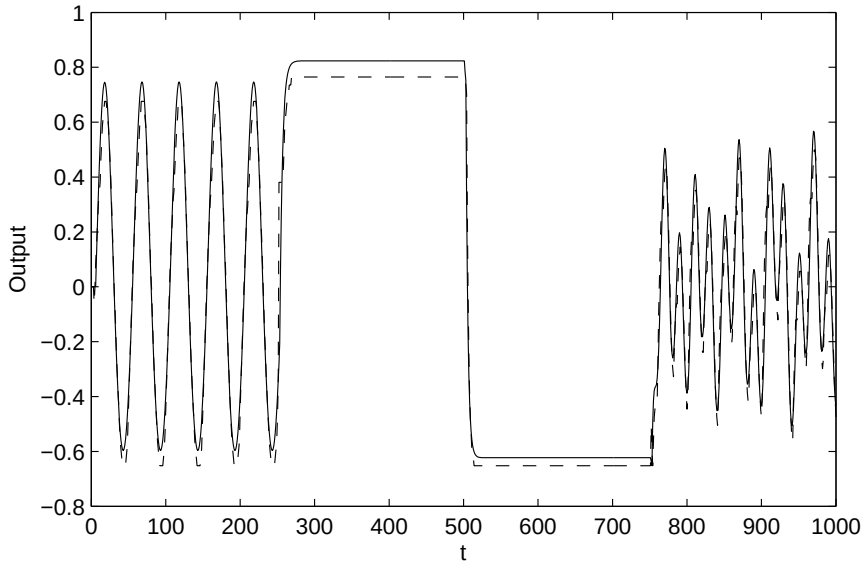


Figure 3.2. Identification results for Example 1 (solid line: plant output, dashed line: SNN identifier output).

The simulation is also performed for the case when the number of synapses for each input signal is 16. For this case the RMSE value for the training part is obtained as 0.052, and 0.057 for the testing part.

Table 3.1. Identification simulation results for Example 1.

Models	Network Structure and Parameters	Train	Test
SNN	layers=1;inputs=2; synapses=8; Parameters =16	0.064	0.068
	layers=1; inputs=2; synapses=16; Parameters =16	0.052	0.057
ERNN	Parameters=54	0.036	0.078

3.2.1.2. Example 2. This example considers the second order nonlinear plant described by the following difference equation:

$$y(t) = f(y(t-1), y(t-2), y(t-3), u(t), u(t-1)) \tag{3.13}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2} \tag{3.14}$$

A one layered SNN with 8 synapses and another with 16 synapses for each connection is used for identification purposes. The excitation signal given in Example 1 is used for the training of the identifier. As before, the weight values of the SNN are initialized within the interval [0,1]. The RMSE value for training with 8 synapses is 0.0962, and for testing it is 0.0982. With 16 synapses, the average RMSE value for training is 0.0585, and for testing 0.0741. Figure 3.3 depicts the performance of the SNN.

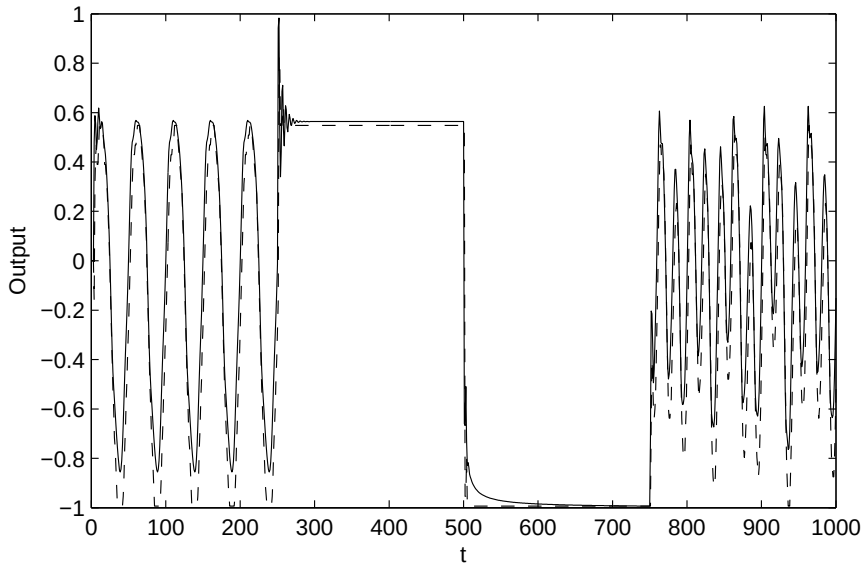


Figure 3.3. Identification results for Example 2 (solid line: plant output, dashed line: SNN identifier output).

3.2.2. Control Performance Studies

3.2.2.1. Control of Antilock Braking System. Antilock Braking System (ABS) is an electronically controlled system that prevents the wheels from locking by limiting the pressure delivered to each slave cylinder of the vehicle. By preventing lock up, ABS

gives the driver the opportunity to maintain the steering control of the vehicle during emergency braking. Another significant feature of ABS is that by keeping brake pressure just below the point of causing a wheel to lock, ABS enables the application of the maximum braking power, which may give rise to shorter stopping distances on slippery or snowy surfaces.

In most control engineering problems, the performance of a controller is directly related to the accuracy of the mathematical model of the system to be controlled. However, an accurate mathematical model of ABS has not been obtained yet. One of the reasons for that is that the controller should operate at an unstable equilibrium point to achieve the optimal performance available. Thus, small changes in the input may give rise to drastic variations in the output. The parameters of ABS highly depend on the road conditions and they can vary over a wide range, but easily available and affordable sensors are still not capable of identifying road condition which will be utilized by ABS to determine the optimal value of the brake pressure, resulting in a performance degradation. Another shortcoming is that the sensor signals usually have highly uncertain and noisy characteristics.

Figure 3.4 demonstrates the free body diagram of the quarter vehicle model under braking. The model of ABS consists of two rolling wheels. The lower wheel imitates the relative road motion, whereas the upper wheel mounted to the balance lever animates the wheel of the vehicle. The velocity of the car is defined to be equivalent to the angular velocity of the lower wheel multiplied by the radius of this wheel, and the angular velocity of the wheel is equivalent to the angular velocity of the upper wheel. Despite the fact that the model is quite simple it attains all the essential characteristics of an actual braking system. In deriving the dynamic equations for the longitudinal motion of the vehicle and angular motion of the wheel, it is assumed that there is no interaction between the four wheels of the vehicle. Additionally, the lateral and vertical motions of the vehicle are neglected.

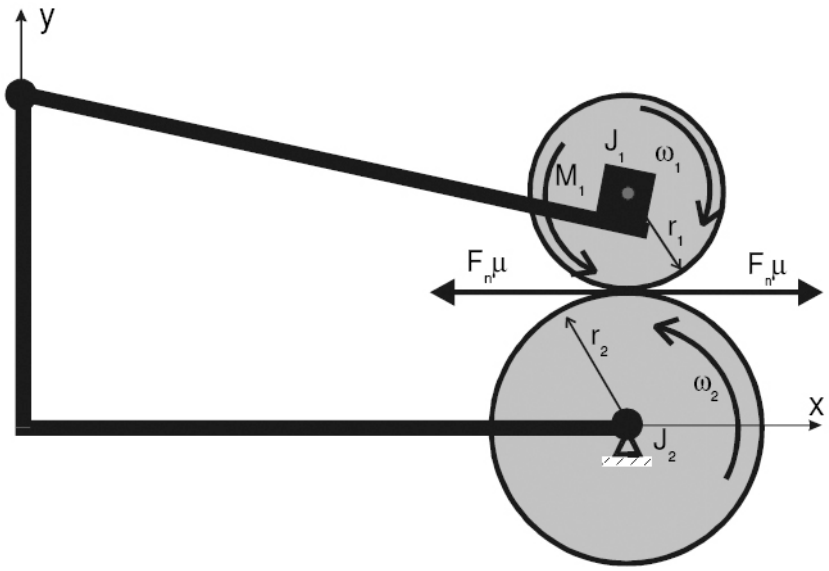


Figure 3.4. Schematic view of the ABS experimental setup.

Before the application of the brakes, the vehicle is moving with an initial longitudinal velocity $V = \omega_2 r_2$ and the angular velocity of the wheel is ω_1. Due to the friction between the tyre and the road surface, a tractive force F_t is generated. When the brakes are applied, there are three torques acting on the upper wheel. These are namely the braking torque, friction torque in the upper bearing and the friction torque among the wheels. Similarly, there are two torques acting on the lower wheel: The friction torque in the lower bearing and the friction torque between these wheels.

According to Newton's second law, the equation of the motion of the system can be written as:

$$J_1\dot{\omega}_1 = F_t r_1 - (d_1\omega_1 + M_{10} + T_B) \tag{3.15}$$

$$J_2\dot{\omega}_2 = -(F_t r_2 + d_2\omega_2 + M_{20}) \tag{3.16}$$

The values of F_t in 3.15 and 3.16 stands for the road friction force which is given by

Table 3.2. Parameters of Antilock Braking System.

Symbol	**Description**	**Value**	**Unit**
ω_1	Angular velocity of the upper wheel		rad/s
ω_2	Angular velocity of the lower wheel		rad/s
T_B	Braking torque		Nm
r_1	Radius of the upper wheel	0.0995	m
r_2	Radius of the lower wheel	0.099	m
J_1	Moment of inertia of the upper wheel	$7.53\,10^{-3}$	kgm^2
J_2	Moment of inertia of the lower wheel	$25.6\,10^{-3}$	kgm^2
d_1	Viscous friction coefficient of the upper wheel	$1.187\,10^{-4}$	kgm^2/s
d_2	Viscous friction coefficient of the lower wheel	$2.1468\,10^{-4}$	kgm^2/s
F_n	Total normal load	58.214	N
μ	Road adhesion coefficient		
λ	Wheel slip		
λ_R	Reference slip		
F_t	Road friction force		N
M_{10}	Static friction of the upper wheel	0.0032	Nm
M_{20}	Static friction of the lower wheel	0.0925	Nm
M_g	Moment of gravity acting on balance lever	19.62	Nm
L	Distance between the contact point of the wheels and the rotational axis of the balance lever	0.37	m
φ	Angle between the normal in the contact point and the line L	65.61	^{o}
u	Brake control input		

Coulomb Law:

$$F_t = \mu(\lambda) F_n \tag{3.17}$$

To derive the normal force in Equation 3.17 we should write the sum of torques corresponding to the point A in the Figure 3.5 In this figure, L is the distance between the contact point of the wheels and the rotational axis of the balance lever and ϕ is the angle between the normal in the contact point and the line L.

$$F_n L(\sin\varphi - \mu(\lambda)\cos\varphi) = d_1\omega_1 + M_{10} + T_B + M_g \tag{3.18}$$

$$F_n = \frac{d_1\omega_1 + M_{10} + T_B + M_g}{L(\sin\varphi - \mu(\lambda)\cos\varphi)} \tag{3.19}$$

Under normal operating conditions, the rotational velocity of the wheel would match the forward velocity of the car. When the brakes are applied, braking torque is generated which causes the wheel speed to decrease. The braking torque depends on the value of brake control input u, and it can be approximated by the following formula.

$$\dot{T}_B = c_{31}(b(u) - T_B) \tag{3.20}$$

and

$$b(u) = \begin{cases} b_1 u + b_2 & \text{if } u \geq u_0 \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

where $c_{31} = 20.37$, $u_0 = 0.415$, $b_1 = 15.24$, and $b_2 = -6.21$. The relationship between braking torque and brake control input is presented in Figure 3.6.

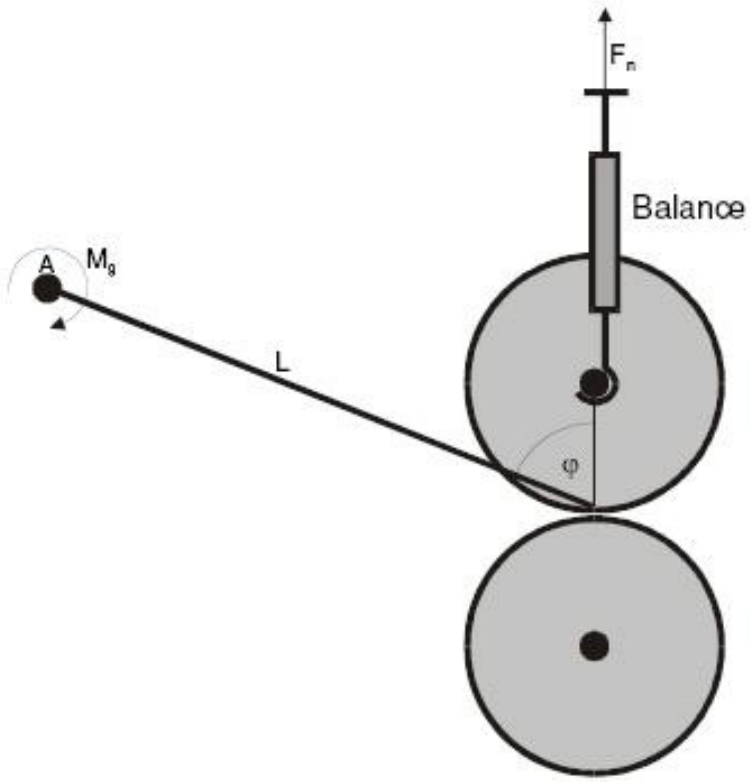


Figure 3.5. Auxiliary diagram of the ABS to develop the model.

With the application of braking torque, braking forces are generated at the interface between the wheel and road surface. As the force at the wheel increases, slippage will occur between the tire and the road surface and the wheel speed will tend to be lower than vehicle speed. The parameter used to specify this difference in these velocities is called wheel slip (λ), and it is defined as:

$$\lambda = \frac{r_2\omega_2 - r_1\omega_1}{r_2\omega_2} \tag{3.22}$$

While a wheel slip of *zero* indicates that the wheel velocity and the vehicle velocity are the same, a ratio of *one* indicates that the tire is not rotating and the wheels are skidding on the road surface, i.e., the vehicle is no longer steerable.

The road adhesion coefficient (or coefficient of friction) is the proportion of road friction force to the normal load of the vehicle and it is a nonlinear function of some

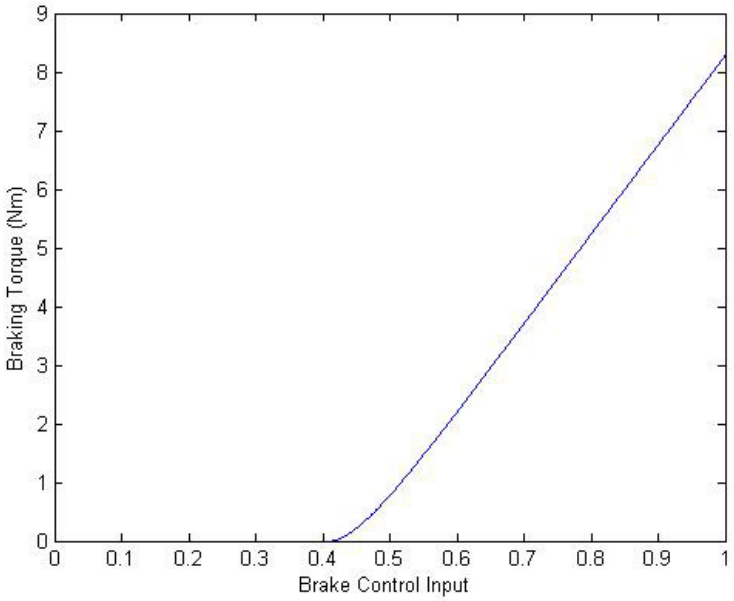


Figure 3.6. Braking torque vs. brake control input.

physical variables including wheel slip. There are different approaches to find the value of slip which will maximize the road adhesion coefficient and friction force. In this study, only two of them will be discussed and employed in the designed controllers.

In the first method, it is assumed that the road adhesion coefficient is a single-variable unimodal function of wheel slip. The main issue in this approach is to develop a model to relate wheel slip and coefficient of friction to each other in such a way that the resulting graph would match with the experimental test results. Figure 3.7 shows the dependence of the road adhesion coefficient to the wheel slip based on the experimental results. Furthermore, based on these data the following formula is derived in [48] to find the road adhesion coefficient corresponding to a wheel slip value.

$$\mu(\lambda) = \frac{c_4 \lambda^p}{a + \lambda^p} + c_3 \lambda^3 + c_2 \lambda^2 + c_1 \lambda \tag{3.23}$$

The resulting road adhesion coefficient vs. wheel slip curve is presented in Figure 3.7.

Another approach followed in this study is based on the LuGre dynamic tire/road

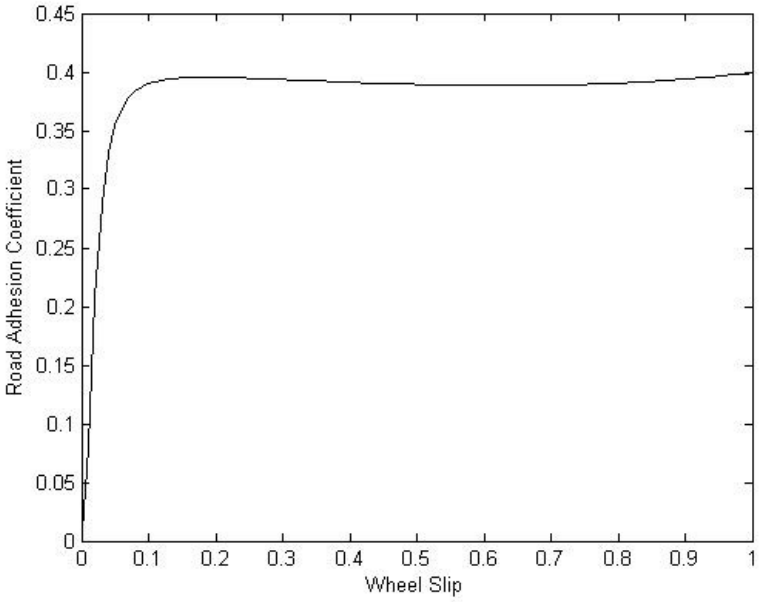


Figure 3.7. Road adhesion coefficient vs. wheel slip.

friction model developed by the Department of Automatic Control in Lund (Sweden) and in Grenoble (France) [49]. This pseudo-static model deals with the dependence of friction on velocity. The following relationship between λ and μ is obtained solving the distributed LuGre tire/road friction model.

$$\mu(\eta,\nu) = -h(V_r)\left[1 + 2\gamma\frac{h(V_r)}{\sigma_0 L|\eta|}\left(e^{-\frac{\sigma_0 L|\eta|}{2h(V_r)}} - 1\right)\right] - \sigma_2 V_r \tag{3.24}$$

where $\eta = \frac{\lambda}{\lambda-1}$, $\quad h(V_r) = \mu_c + (\mu_s - \mu_c)e^{-|\frac{V_r}{V_s}|^{1/2}}$, and $\quad \gamma = 1 - \frac{\sigma_1|\eta|}{Rwh(V_r)}$.

As can be seen from Equation 3.24, if the velocity of the vehicle changes, the curve will change as well. However, the change in the curve will happen faster than a change in the vehicle velocity. Hence, it is possible to calculate the approximated peak value of braking force produced by tire/road friction for each time step. The description and numerical values or parameters used in this model are presented in Table 3.3.

Figure 3.8 shows the relationship between μ and λ for different velocities from

Table 3.3. System parameters for LuGre tire/road friction model.

Symbol	Description	Value
L	Length of the tire/road contact patch	0.25 (m)
V_s	Stribeck relative velocity	10 (m/s)
V_r	Relative velocity $(V - R\omega)$	
μ_s	Normalized static friction coefficient	0.5
μ_c	Normalized Coulomb friction coefficient	0.35
σ_0	Rubber longitudinal stiffness	100 (1/m)
σ_1	Rubber longitudinal damping	0.7 (s/m)
σ_2	Viscous relative damping	0.011 (s/m)

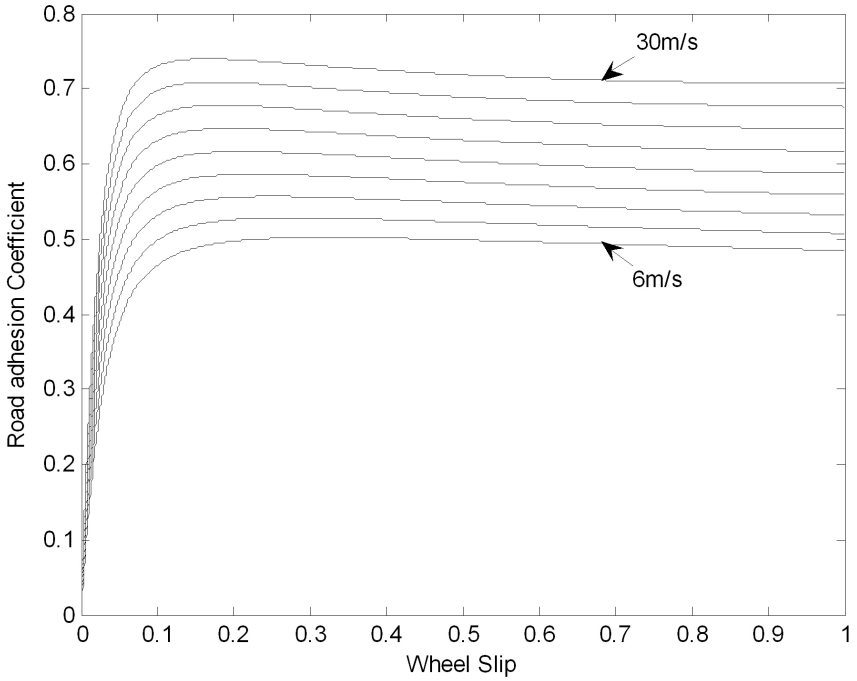


Figure 3.8. μ-λ relation for several velocity values for dry road conditions.

$6m/s$ to $30m/s$ for every $3m/s$ increment. The results demonstrate the significant influence of velocity on both the road adhesion coefficient and corresponding slip value.

To investigate the performance of the SNN controller, number of computer simulated dynamic responses have been obtained. The initial longitudinal velocity of the

vehicle is taken as $V = 30m/s$. It is assumed that the vehicle is moving on a straight line. The sampling time T_s is set to $1ms$ for all simulations. Two different approaches are followed to determine the reference value of the wheel slip. In the first approach the reference wheel slip is considered to be a constant value of 0.2. In the latter approach, the pseudo-static curve mentioned in Equation 3.24 is used for determining the reference value of the wheel slip. A look up table is composed to relate the vehicle velocity to the peak value of μ-λ curve. At each time step of the simulation, the reference wheel slip, which corresponds to the maximum value of the road adhesion coefficient specified for the current vehicle velocity, is made available to the proposed controller. The error, which is defined as the discrepancy between the desired and actual values of wheel slip, and its time derivative are fed to the controller inputs. These input values are converted to the input spike times by using the encoding scheme given in Equation 2.10. The output of the SNN is specified by a firing time. To convert this firing time to the control signal $u(t)$, which will be utilized in Equation 3.21 and Equation 3.20 for the generation of the required braking torque T_B, the following decoding formula is used:

$$u = u_{min} + \frac{(t_{max} - t_{min} - t_i)\,(u_{max} - u_{min})}{(t_{max} - t_{min})} \tag{3.25}$$

where u_i, u_{min} and u_{max} are the current, minimum and the maximum values of the control signal, respectively.

The SNN controller consists of three layers: an input layer with 2 neurons, a hidden layer with 4 neurons, and an output layer with a single spiking neuron. There are 10 synapses between each presynaptic and postsynaptic neuron. The time delays for synaptic connections vary between $1ms$ and $10ms$ and the weights of the network are randomly initialized between 0 and 1. The decay time constant is selected as 5 and the threshold value is 4. The learning constant is determined as 0.0005. For the transition among real numbers and spike times the coding scheme described in Equation 2.10 is utilized. Figure 3.9 shows the wheel slip values for SNN and PI controllers.

The PI controller used in this study has been assumed to have the following structure:

$$U(s) = \frac{K_p s + K_i}{s} E(s) \tag{3.26}$$

where $E(s)$ and $U(s)$ are the error and controller output signals, respectively. To tune the PI controller gains, the mathematical model of the ABS has been simplified and linearized around the optimum value of the wheel slip. Based on the linearized model, the relationship between the braking torque T_B and the controller output $U(s)$ can be rewritten as:

$$T_B(s) = 14U(s) \tag{3.27}$$

Using Equation 3.27 and the numerical values of the ABS parameters, the simplified dynamics of the upper wheel can be stated as follows:

$$7.5310^{-3} s\Omega_1(s) = -1.18710^{-4}\Omega_1(s) + 14U(s) \tag{3.28}$$

The simplified transfer function of the ABS can be thus written as:

$$G(s) = \frac{\Omega_1(s)}{U(s)} \simeq \frac{14}{0.007(s + 0.016)} = \frac{2000}{s + 0.016} \tag{3.29}$$

Using Equation 3.29 and Equation 3.26, the closed loop transfer function of the system can be stated as:

$$G_c(s) = \frac{2000(K_p s + K_i)}{s^2 + (0.016 + 2000K_p)s + 2000K_i} \tag{3.30}$$

Let us define $K_P = 2000K_p$ and $K_I = 2000K_i$. Then the closed loop transfer function

of the overall system yields

$$G_c(s) = \frac{K_P s + K_I}{s^2 + (0.016 + K_P)s + K_I} \tag{3.31}$$

Using Equation 3.31, the parameters of the PD controller can be chosen such that the desired transient response characteristics are met. To obtain the fastest response available without any overshoot or ringing, the parameters have been selected as $K_P = 15.984$ and $K_I = 64$, which corresponds to a critically damped response with a natural frequency of $8 rad/s$.

As expected, the response of SNN controller is relatively slow compared to the response of the PD controller because the SNN controller requires some time for the adaptation of its weights. Both controllers are capable of stopping the car within 9 seconds and regulate the wheel slip to target value of 0.2 to maximize the friction coefficient. Figure 3.10 shows the vehicle and wheel speed for the SNN controller. As the response characteristics of the PD controller are similar to those of the SNN controller, the figure for the vehicle and wheel speed for PD controller is not given. It can be inferred from Figure 3.11 that both controllers yield almost the same stopping distance.

As mentioned earlier, the signals coming from the sensors are usually noisy, which contribute to performance degradation of the controllers. To imitate the effect of the noise on the controllers, band-limited white noise with a noise power of $5x10^{-6}$ is added to the slip measurements in the simulations. Figure 3.12 demonstrates that the SNN controller is capable of overcoming the effect of noise in the measurements, whereas the response of the PD controller is quite oscillatory. This result support the idea that the performance of model based approached drastically reduces when uncertainties are involved, and model free approaches should be preferred in such cases. Figure 3.13 shows the dynamic responses of the controllers when the wheel slip is not constant, but vary over time with regard to the velocity of the car. Noise is also included in the

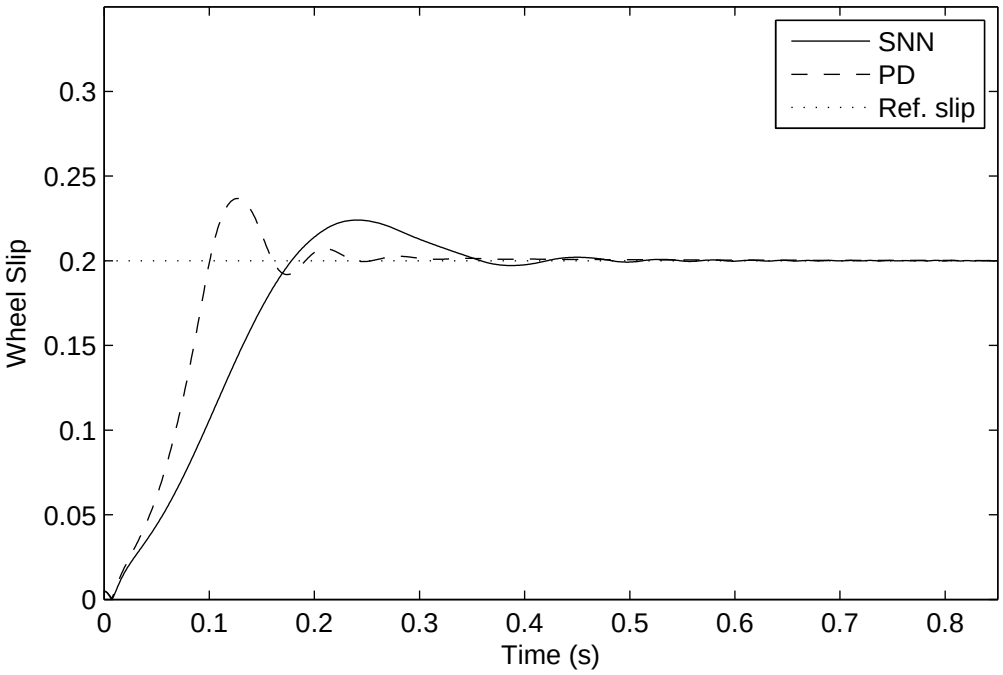


Figure 3.9. Wheel slip for SNN and PD controllers for a constant reference wheel slip value.

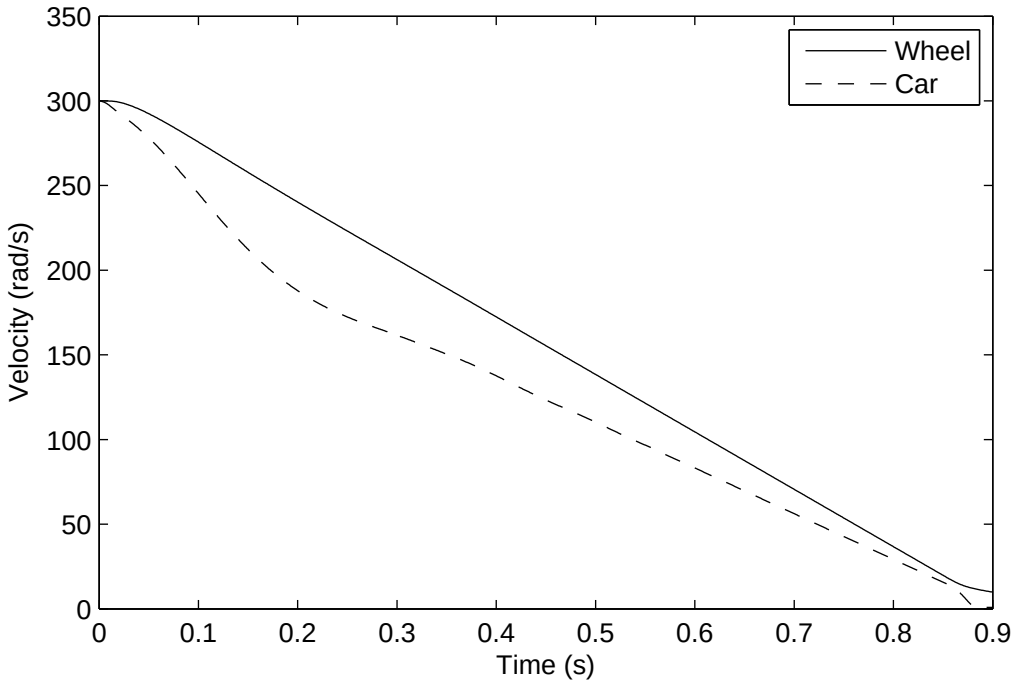


Figure 3.10. Vehicle and wheel speed for SNN controller with a constant reference wheel slip value.

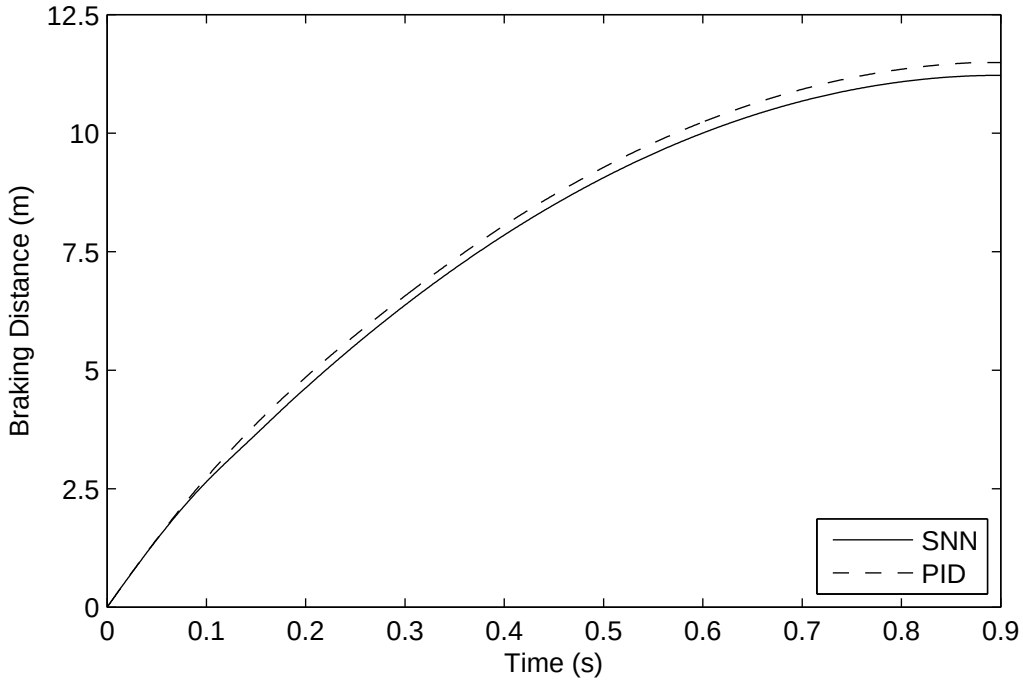


Figure 3.11. Stopping distance for SNN and PD controllers with a constant reference wheel slip value.

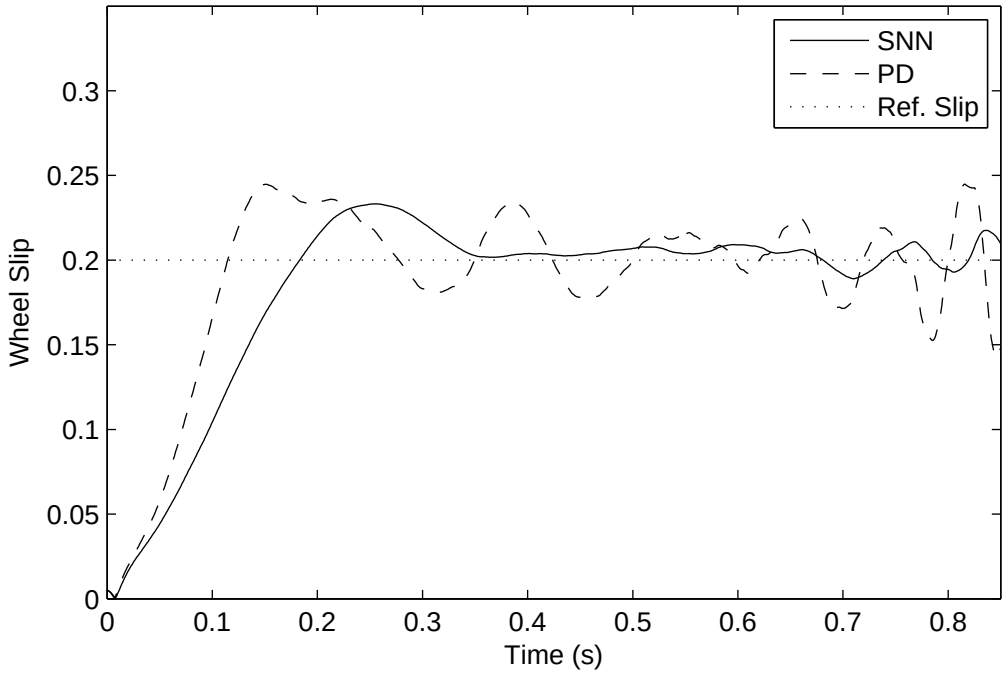


Figure 3.12. Wheel slip for SNN and PD controllers for noisy sensor measurements.

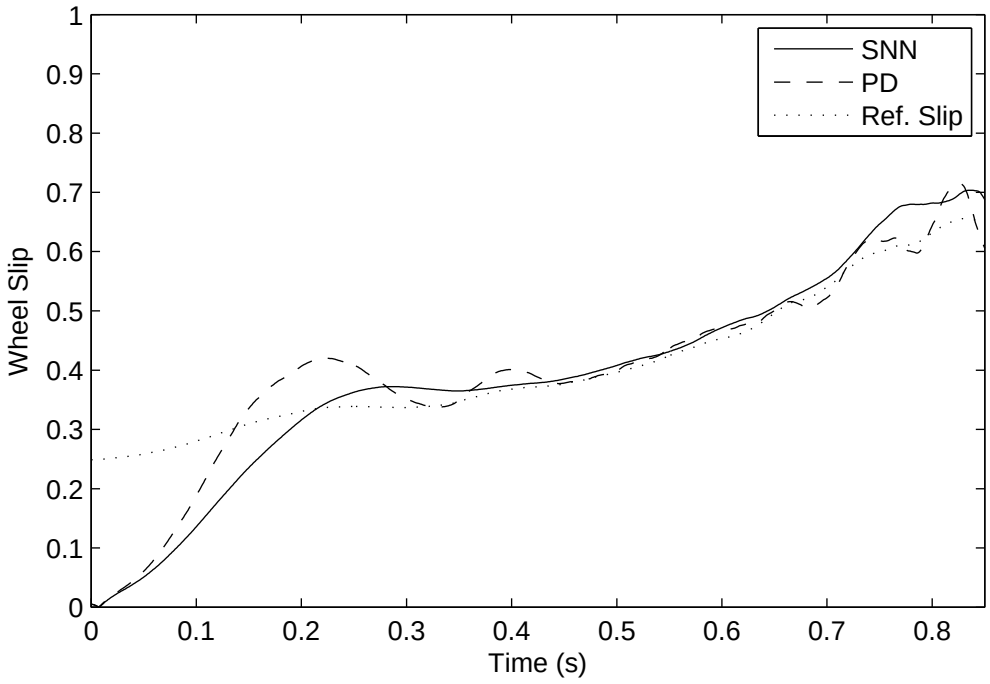


Figure 3.13. Wheel slip for SNN and PD controllers for velocity dependent reference value.

slip measurements to imitate real life conditions. The simulation result indicates that the SNN controller can track the reference value with reduced oscillations.

3.2.2.2. DC motor speed control. The experimental setup used in this study consists of two DC motors, which are placed facing each other and their drive shafts are rigidly coupled by a mechanical clutch (See Figure 3.14). The Humusoft MF624 data acquisition card is used for the connection between computer and the DR300 Amira servo system. All functions of the board can be accessed from the Real-Time toolbox which operates directly in the MATLAB/Simulink environment. The input signal for the permanently exited DC motor M1 is the armature current and it is provided by a cascaded current control loop. Available sensors for the output signal (rotation) are an analog tacho generator and a digital incremental encoder. The second motor M2 operates in generator mode to apply a load torque to the first motor. Its output current is adjustable by another current controller.The essential feature of this setup is that it

enables the generation of nonlinear load conditions to provide the means for comparing the performance of various controllers in the presence of nonlinear load disturbances.

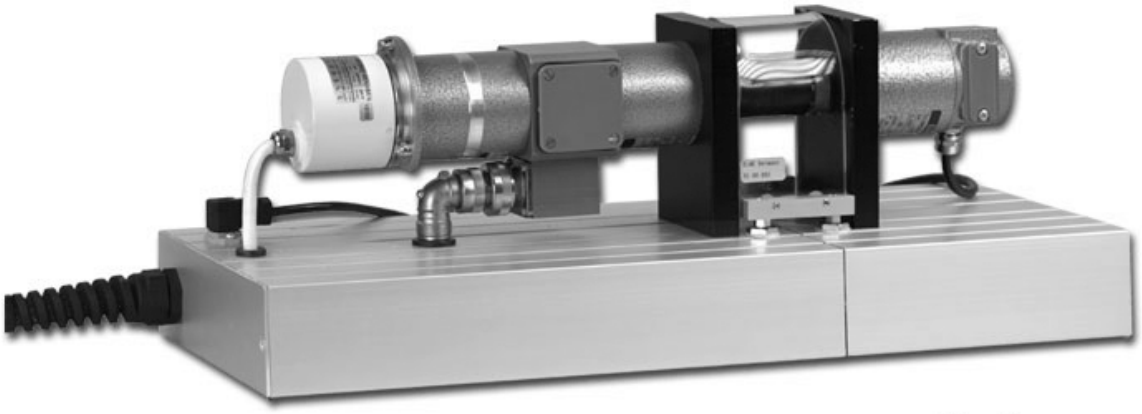


Figure 3.14. DR300 Amira servo system.

The block diagram of the servo control system is shown in Figure 3.15, the nomenclature of the symbols used being given in Table 3.4. From the block diagram of the system depicted in Figure 3.15, the transfer function of the overall system can readily be derived as follows:

$$\begin{aligned} \omega(s) &= \frac{1}{C\Phi}\frac{1}{1+T_M s+T_M T_A s^2}U_A(s) \\ &- \frac{R_A}{K_M C\Phi}\frac{1+T_A s}{1+T_M s+T_M T_A s^2}M_L(s) \end{aligned} \tag{3.32}$$

where

$$T_M = \frac{J_m R_A}{K_M C\Phi} \quad \text{and} \quad T_A = \frac{L_A}{R_A}$$

The motor M1 is controlled by a current controller. The range for the input voltage U_A is [-10, 10] volts. The amplification constant of the current controller, which behaves like a first order lag with a time constant of 5 ms, is 0.4 A/V. The ratio between the generated torque M and the armature current I_A is referred to as the torque constant K_M. This torque is balanced with the load torque M_L and the acceleration torque M_B. The second motor M2, which is used for the generation of the

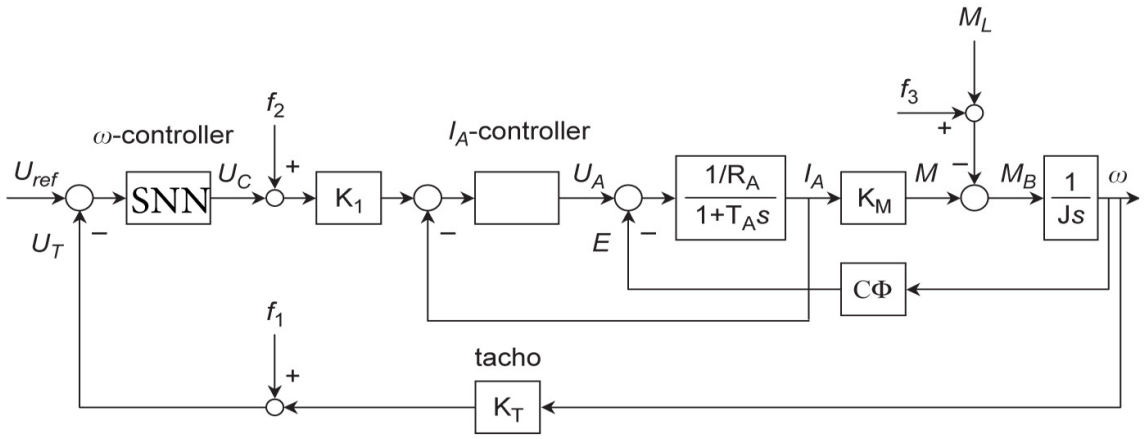


Figure 3.15. Block diagram of the motor with load.

load torque M_L is also controlled by a current controller with an input voltage range of [-10, 10] volts. The corresponding amplification constant is 0.237 A/V. Using Newton's second law, the acceleration torque M_B can be stated as follows:

$$M_B = J_m \frac{d\omega}{dt} \tag{3.33}$$

where J_m refers to the moment of the inertia of the complete system.

The tachogenerator is used for the measurement of the rotational speed of the motor shaft relying on the fact that the voltage U_T is proportional to the rotational speed ω. The output signal of the tachogenerator ranges from -10 V to $+10$ V with 2.5 mV/Rpm.

The one layered SNN is composed of three input neurons and one output neuron. Two of the inputs are the error and the change of error. A third bias input neuron that always fires at $t = 0$ is included to allocate the reference start time. 6 synapses are used for each connection resulting in 18 parameters to be updated. The connections have a delay interval of 5 ms; hence the available synaptic delays are from 1 to 6 ms. The sampling time is set to 10 ms for all the experiments. The speed of the motor and the load torque are scaled to [-10,10]. The decay time constant is selected as $\tau = 7ms$

Table 3.4. Nomenclature.

Name	**Description**
U_A	Armature terminal voltage
E	Induced electromotive force
I_A	Armature current
R_A	Armature winding resistance
L_A	Armature winding inductance
C	Motor constant
Φ	Magnetic excitation
ω	Speed of the rotor
M_L	Load torque
M_B	Acceleration torque
M	Torque produced by the motor
J	Moment of inertia of the motor
T_A	Electrical time constant
T_M	Mechanical time constant

and the threshold value is set to $\theta = 3$.

In order to determine the efficiency and the accuracy of the proposed controllers, three different types of load conditions are considered. In order to initialize w_{ij}^k parameters of the SNN, first the simulation model of the setup is run for 100 epochs. The final values obtained from the training operation are used as the initial values in the real time experiments. The initial value for the learning rate is taken as 0.05. During the synthesis of the input signals of the SNN control system, the error and the change of error are converted into spike times. The output signal of the SNN is also spike characterized with a spike time. This spike time is converted into a real number that is entered in the plant input.

Figure 4.6 presents the speed responses of the motor with the SNN controller for

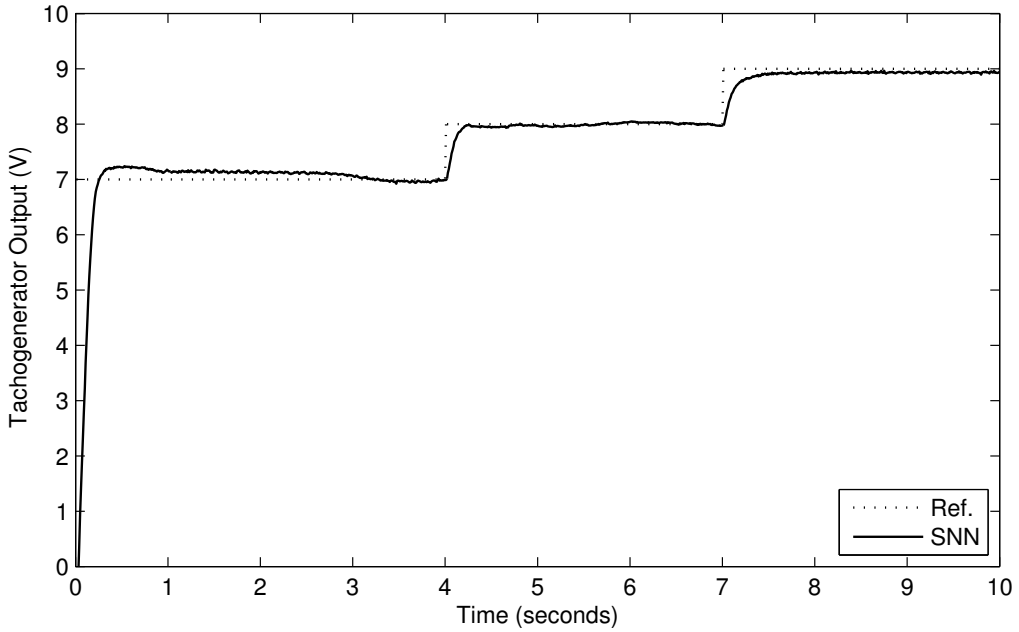


Figure 3.16. Speed response characteristic of the SNN controller for a changing reference value.

different set-point signals. The set point signal has been suddenly changed from 7 to 8 at the 4th second and then suddenly becomes 9 at the 7th second. It can be inferred that the SNN controller is capable of adapting its weights to regulate the system to changing set points.

The speed responses of the motor with the SNN controller have been given in Figure 3.17. The corresponding load condition is shown in Figure 3.18, which indicates that the load torque starts with a value of $0.015Nm$ and increases suddenly to $0.03Nm$ on the 4th second, and then comes back to $0.015Nm$ on 7th second.

Figure 3.19 shows the speed response of the motor under the sinusoidal load condition, i.e. $M_L(t) = 0.03 + 0.006sin(0.0225t)$. On the other hand, Figure 3.20 shows the speed response of the motor under load condition which is proportional to the square of the speed, i.e. $M_L(t) = 0.1125x10^{-3}(TachogeneratorOutput)^2$. This type of load corresponds to the load-torque characteristics of centrifugal fans and pumps.

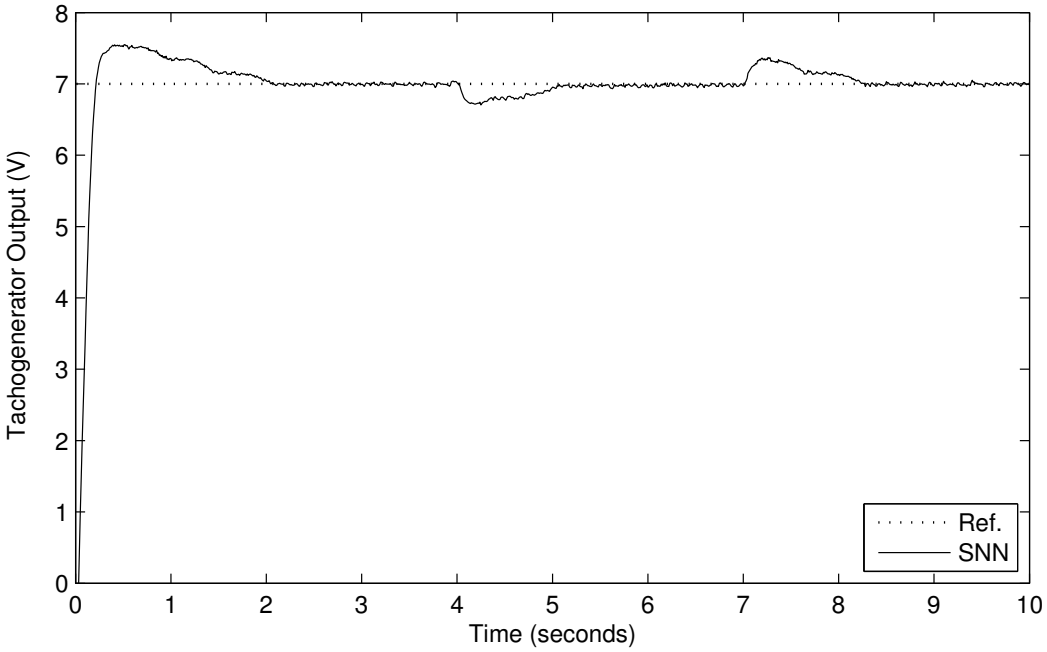


Figure 3.17. Speed response of the motor for step changing load.

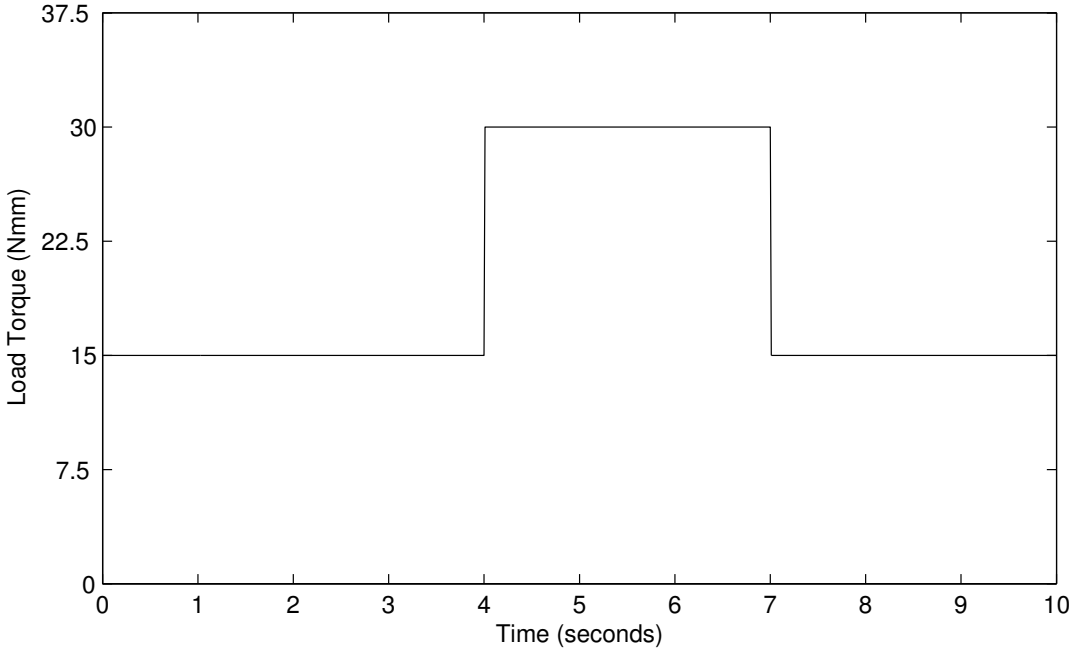


Figure 3.18. Step changing load condition.

3.3. Analysis and Discussion

In this study, the gradient descent-based SpikeProp algorithm has been tested on different identification and control tasks. For the identification part, the plants

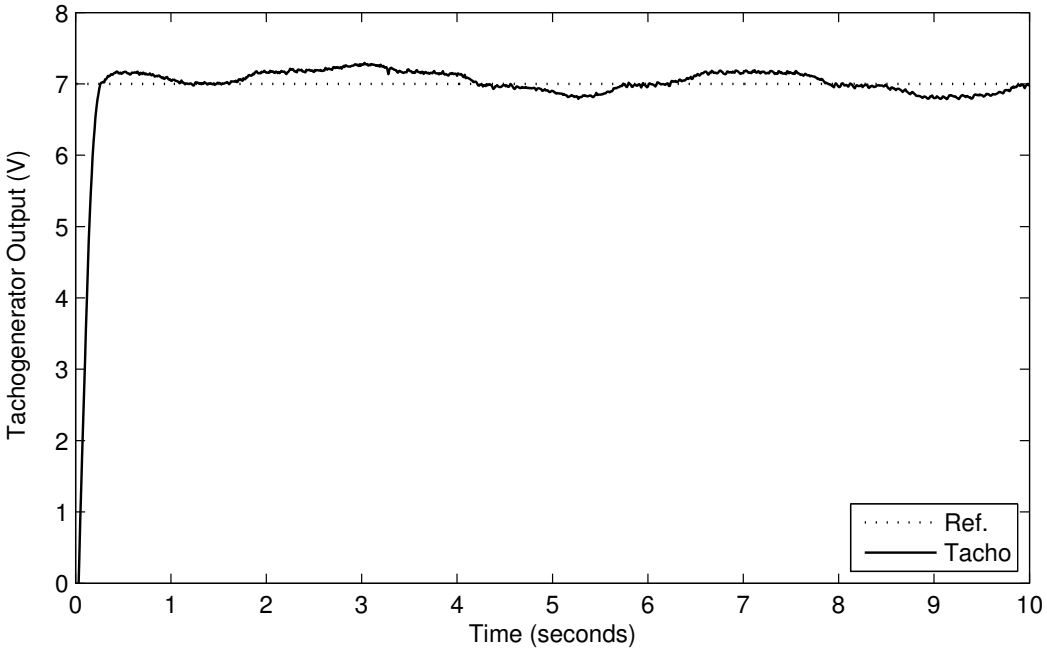


Figure 3.19. Speed response of the motor for sinusoidal load condition.

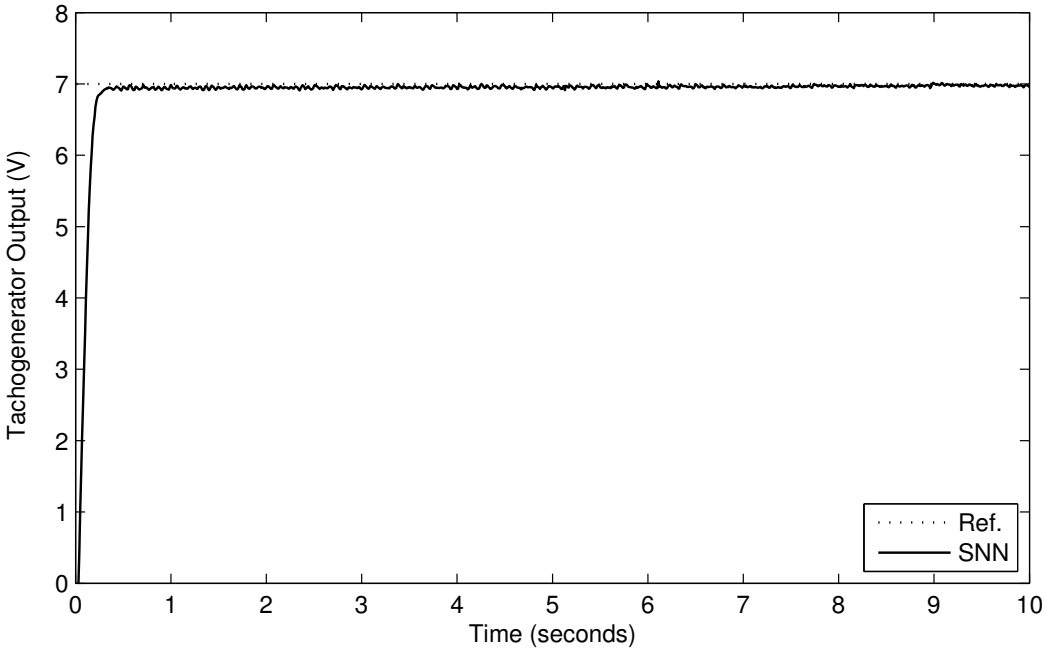


Figure 3.20. Speed response of the motor for nonlinear load condition.

are selected from literature. The delay coding and decoding is applied for converting real numbers into spikes. To gain insight into the capabilities of spiking neurons, a

Table 3.5. RMSE values for different load conditions.

SNN	Step Changing Load Condition	Sinusoidal Load Condition	Nonlinear Load Condition
RMSE	0.6331 V	0.6372 V	0.6008 V

single layered neural network structure has been considered. This way the advantage of having multi layers, and thus more computational power and complexity, has been minimized. It is seen that the SNN can converge quickly. Despite the smaller number of parameters, the small RMSE values indicate that the performance of the SNN is comparable to other approaches commonly used in the literature.

In this study, a SNN controller is considered for the regulation of the wheel slip value at its optimum value in an ABS. The control of ABS is a challenging task due to its strongly nonlinear structure and uncertainties involved. The control algorithm is applied to a quarter vehicle model, and it is verified through simulations indicating fast convergence and good performance of the designed controller.

Designing a controller and simulating it in the computer environment can give a general understanding about how the system may behave on an actual system; however, this information is generally neither sufficient nor precise. Because of unpredictable factors such as internal and external disturbances, it is possible that the system may move in an unexpected way, which may make the performance of the proposed control algorithm unacceptable. The SNN structure is also tested on a real time DC motor system under different nonlinear load conditions. The experimental results obtained indicate the potential of this new generation of ANNs. The control structure proposed has the ability to regulate the servo system with reduced oscillations around the set point signal in the presence of load disturbances.

Although the results obtained from the simulation and experimental studies are promising, like in the case of MLP, the SNN has a layered structure of nonlinear

functions at each node and it is not possible to know the shape of the cost function in the weight space in advance. It can have many local minimum points apart from an absolute local minimum. Hence the convergence of the update rule is not guaranteed. The error may get stuck at a local minimum. Although better convergence characteristics can be achieved by a proper choice of the initial weights, threshold value, and learning rate, there is no predefined procedures for this selection. Next chapter describes the steps taken to overcome the global convergence problem which involved adopting the sliding mode theory in the derivation of the parameter update rules.

4. SLIDING MODE CONTROL-BASED PARAMETER UPDATE RULES FOR SNN

Gradient-based learning methods have been frequently used in NN-based control applications to determine the proper values of the synaptic weights for the generation of the desired output signals [50–53]. Despite the fact that numerous successful applications have been reported throughout literature, these learning algorithms can very often lead to suboptimal performances in terms of the convergence speed, robustness, and computational burden. Furthermore, the stability of the learning process is not guaranteed as the error surface in the weight space includes many local minima and the tuning process may easily get stuck in one of these [54]. The use of evolutionary approaches has been suggested to address the problems mentioned [55–59], but the stability of such approaches is questionable and the optimal values for the stochastic operators are difficult to derive. Additionally, the computational burden can be very high. The shortcomings with the evolutionary approaches give rise to the development of sliding mode control (SMC) theory-based algorithms for the adaptation of the neural networks' weights [24–27, 29, 60–62], by means of which the robust system response in handling the uncertainties and imprecision and faster convergence than the traditional learning techniques in online tuning of ANNs can be ensured.

SMC has attracted the attention of many researchers in the field of computational intelligence owing to its versatile characteristics such as robustness to external disturbances, parameter variations and uncertainties [63, 64]. The imprecision in nonlinear system models may result either from the actual uncertainty about the system such as unknown plant parameters, or from the purposeful choice of a simplified representation of the system dynamics. The vast majority of industrial applications are subjected to disturbances and parameter fluctuations because of the noise in the measurements, mechanical stresses and friction, which can not be computed or estimated beforehand. Moreover, in the derivation of the mathematical model of the plant, sev-

eral assumptions and linearization can be made to ease the calculations process. These uncertainties can have strong adverse effects on the control systems. SMC is an attractive option to overcome these shortcomings, as it guarantees the robustness of the system for changing working conditions and modeling imprecision.

The underlying idea of the SMC approach is to restrict the motion of the system in a plane referred to as the sliding surface, where the predefined function of the error is zero, and to keep them moving along this surface for succeeding times. [65].

Let us consider the single input dynamic system:

$$y^{(n)} = f(x(t)) + g(x(t))u(t) \tag{4.1}$$

In this formulation $x(t)$ refers to the state vector, $u(t)$ to the control input, and y to the output state of the system. The order of the differentiation is denoted by (n). We further assume that $f(x(t))$ and $g(x(t))$ are some nonlinear functions of time and the states of the system. The discrepancy between the desired and the actual output states of the system and its derivatives are widely used to construct a sliding surface as follows [66]:

$$s(x(t)) = \left(\frac{d}{dt} + \lambda_s\right)^{(n-1)} (y - y^d) = 0 \tag{4.2}$$

where λ_s is a strictly positive constant and y^d refers to the desired time-varying output state of the system. As can be seen, sliding mode control enables to replace an n-dimensional tracking problem by a first order stabilization problem in s.

The sliding surface $s(x)$ will be attractive if [67]:

- Any trajectory starting on the surface remains there
- Any trajectory starting outside the surface tends to the sliding surface at least

asymptotically

The first condition requires if $s(x(t)) = 0$, then its derivative should be $\dot{s}(x(t)) = 0$ as well. The second condition can be satisfied if

$$\lim_{s \to 0^+} \dot{s} < 0 \quad \text{and} \quad \lim_{s \to 0^-} \dot{s} > 0 \tag{4.3}$$

This is referred to as the reachability condition, and to enforce it Lyapunov stability method is applied [68].

Consider the positive definite Lyapunov function candidate

$$V(x,t) = \frac{1}{2}s^2 \geq 0 \tag{4.4}$$

This equality holds if and only if $s = 0$. If we recall the Lyapunov Stability Theorem, the derivative of this function should be negative definite to guarantee the reachability condition, the equality sign being applicable if and only if $s = 0$.

$$\dot{V}(x,t,s) = s\frac{\partial s}{\partial t} \leq 0 \tag{4.5}$$

Thus, if we define a Lyapunov candidate function which satisfies the reachability condition then it is guaranteed that the state trajectories converge to the sliding surface and move along it for all succeeding time.

The proposed incremental learning algorithm makes direct use of the variable structure systems theory and establishes a sliding motion in terms of the neural network's parameters. A sliding surface is constructed by letting s be equal to the discrepancy between the actual and the desired firing times/values of the output neuron. This selection of the sliding surface corresponds to $n = 1$ in Equation 4.2. Next, the

weight update rules are derived in such a way that the multiplication of the sliding surface and its derivative, i.e. error times the derivative of error; always result in a negative value, which is consistent with the reachability condition. Therefore, unlike the gradient descent-based learning algorithms, the convergence of this approach is always guaranteed. In the following subsection, SMC-based learning algorithms will be derived for SNNs with two layers.

4.1. Single Layered SNN

Consider the one-layered feedforward spiking neural network. It is assumed that there are n neurons in the input layer, and only one neuron in the output layer. The following definitions will be used:

- $t_i(t) = [\ t_{i1}(t) \quad t_{i2}(t) \quad \dots \quad t_{in}(t)\ ]^T$ - vector of the firing times of the input neurons
- $t_j(t)$ - firing time of the output neuron
- $w_{ij}^k(t)$ - time-varying connections' weight between the neuron i in the input layer and the output node j for the synaptic terminal k

It is assumed that the input vector $t_i(t) = [\ t_{i1}(t) \quad t_{i2}(t) \quad \dots \quad t_{in}(t)\ ]^T$ and its time derivative $\dot{t}_i(t) = [\ \dot{t}_{i1}(t) \quad \dot{t}_{i2}(t) \quad \dots \quad \dot{t}_{in}(t)\ ]^T$ are bounded. $t_j^d(t)$ represents the time-varying desired spike time of the output neuron. It will be assumed that $t_j^d(t)$ and $\dot{t_j^d}(t)$ are also bounded, i.e. $|t_j^d(t)| \leq B_{t_j^d}$ and $|\dot{t}_j^d(t)| \leq B_{\dot{t}_j^d}$, where $B_{t_j^d}$ and $B_{\dot{t}_j^d}$ are positive constants.

We define the learning error $e(t)$ as the scalar quantity obtained from $e(t) = t_j(t) - t_j^d(t)$. Using the SMC approach, we define the zero value of the learning error coordinate $e(t)$ as a time-varying sliding surface, i.e.,

$$S(e(t)) = e(t) = t_j(t) - t_j^d(t) = 0 \tag{4.6}$$

which is the condition that guarantees that the neural network output $t_j(t)$ coincides with the desired output signal $t_j^d(t)$ for all time $t > t_{hit}$, where t_{hit} is the hitting time of $e(t) = 0$.

Definition 4.1. *A* sliding motion *will take place on a sliding manifold* $S(e(t)) = e(t) = 0$*, after time* t_{hit}*, if the condition* $S(t)\dot{S}(t) = e(t)\dot{e}(t) < 0$ *is true for all* t *in some nontrivial semi-open subinterval of time of the form* $[t, t_{hit}) \subset (-\infty, t_{hit})$.

The learning algorithm for the neural network weights $w_{ij}^k(t)$ should be derived in such a way that the sliding mode condition of Definition 4.1 will be enforced. The Lyapunov function candidate has been chosen as follows:

$$V(e(t)) = \frac{1}{2}e^2(t) = \frac{1}{2}\left(t_j(t) - t_j^d(t)\right)^2 \tag{4.7}$$

Then differentiating $V(e(t))$ yields:

$$\dot{V}(e(t)) = e\dot{e} = e(\dot{t}_j - \dot{t}_j^d) \tag{4.8}$$

To find the time derivative of $\dot{t}_j$:

$$\frac{\partial t_j}{\partial t} = \frac{\partial t_j}{\partial x_j(t_j)} \frac{\partial x_j(t_j)}{\partial w_{ij}^k(t_j)} \frac{\partial w_{ij}^k(t_j)}{\partial t} \tag{4.9}$$

Since t_j is not continuous, it is not possible to calculate the derivative term $\frac{\partial t_j}{\partial x_j(t_j)}$. To overcome this problem, it is assumed that for a small enough region around $t = t_j$, the function x_j is approximated by a linear function of t. For such a small region, we approximate the threshold function $\delta t_j(x_j) = -\delta x_j(t_j)/\alpha$, where α equals the local derivative of $x_j(t)$ with respect to t. Thus, the first term on the right hand side of

Equation 4.9 evaluates to:

$$\frac{\partial t_j}{\partial x_j(t_j)} = \left[\frac{\partial t_j(x_j)}{\partial x_j(t)}\right]_{x_j=\theta} = \frac{-1}{\frac{\partial x_j(t_j)}{\partial t}} = \frac{-1}{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}} \tag{4.10}$$

The second term in Equation 4.9 yields

$$\frac{\partial x_j(t_j)}{\partial w_{ij}^k} = \frac{\partial\left\{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k y_i^k(t_j)\right\}}{\partial w_{ij}^k} = y_i^k(t_j) \tag{4.11}$$

Thus, the time derivative of t_j can be written as:

$$\dot{t}_j = -\frac{y_i^k(t_j)}{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\dot{w}_{ij}^k \tag{4.12}$$

Using Equation 4.12, Equation 4.8 can be written as:

$$\dot{V} = e\left[-\frac{y_i^k(t_j)}{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\dot{w}_{ij}^k - \dot{t}_j^d\right] \tag{4.13}$$

If the learning algorithm for the weights w_{ij}^k is chosen as

$$\dot{w}_{ij}^k = \alpha y_i^k(t_j)\left(\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}\right) sign(e) \tag{4.14}$$

where α is a strictly positive constant, then the derivative of t_j will be

$$\begin{aligned}\dot{t}_j &= -\frac{y_i^k(t_j)}{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\alpha y_i^k(t_j)\left(\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}\right) sign(e) \\ &= -\alpha\left(y_i^k(t_j)\right)^2 sign(e)\end{aligned} \tag{4.15}$$

If we consider that the desired spike time of the output neuron is constant, i.e. $\dot{t}_j^d = 0$, then for the selected weight update rule, the derivative of the Lyapunov candidate function will be as follows:

$$\begin{aligned}\dot{V} &= e\left[-\frac{y_i^k(t_j)}{\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}}\alpha y_i^k(t_j)\left(\sum_{i\in\Gamma_j}\sum_k w_{ij}^k \frac{\partial y_i^k(t_j)}{\partial t_j}\right) sign(e)\right] \\ &= -\alpha\left(y_i^k(t_j)\right)^2 |e| \end{aligned} \tag{4.16}$$

This implies

$$\dot{V} < 0, \quad \forall e \neq 0 \tag{4.17}$$

The design steps of the SMC-based learning algorithm for a single layered SNN have been given in Figure 4.1.

Setup the network;
Initialize weights w_{ij}^k randomly within the interval $[0, 1]$;
for each input neuron i **do**
  Convert the sensor signal to the spike time t_i using Equation 2.10;
end for
Set simulation time $t = 0$;
for each neuron j in the output layer **do**
  while $t < max\_steps$ **do**
    Calculate $x_j(t)$ using Equation 2.8 and Equation 2.9;
    if $x_j(t) < \theta$ **then**
      Set $t = t + time\_step$;
    else
      Set the time instance t, where the membrane potential $x_j(t)$ crosses the threshold value, as the firing time of neuron j: $t_j = t$;
    end if
    if all neurons in the output layer fire **then**
      Set $t = max\_steps$;
    end if
  end while
end for
Set error $e = \sum_{j \in J}(t_j - t_j^d)$;
for each neuron i in the input layer **do**
  for each neuron j in the output layer **do**
    for each synapse k **do**
      Calculate $\dot{w}_{ij}^k$ using Equation 4.14;
      Set the new weights $w_{ij}^k = w_{ij}^k + \dot{w}_{ij}^k$;
    end for
  end for
end for
Save parameters of SNN;

Figure 4.1. SMC-based learning algorithm for a single layered SNN.

4.2. Two Layered SNN

Consider a two-layered SNN structure where the neuron in the output layer is considered with a linear activation function. Such a structure will eliminate the need for decoding and thus reduce the computational burden. In this structure $w_{hi}^k(t)$ is the weights of the time-varying connections between the neuron h in the input layer and the node i in the hidden layer for the synaptic terminal k, whereas $w_{ij}(t)$ corresponds to the weights of the time-varying connections between the neuron i in the hidden layer and the node j in the output layer.

The vector $t_h(t) = [\ t_{h1}(t) \quad t_{h2}(t) \quad \dots \quad t_{hn}(t)\ ]^T$ consisting of the firing times of the spiking neurons in the input layer and the vector consisting of the time derivatives of these firing times $\dot{t_h}(t) = [\ \dot{t_h}1(t) \quad \dot{t_h}2(t) \quad \dots \quad \dot{t_h}n(t)\ ]^T$are assumed to be bounded. The scalar signal $\tau_j^d(t)$ represents the time-varying desired output of the neural network. It will be assumed that $\tau_j^d(t)$ and $\dot{\tau}_j^d(t)$ are also bounded signals, i.e. $|\tau^d(t)| \leq B_{\tau_j^d}$ and $|\dot{\tau}_j^d(t)| \leq B_{\dot{\tau}_j^d}$, where $B_{\tau_j^d}$ and $B_{\dot{\tau}_j^d}$ are positive constants.

Similar to the one layered structure, the learning error $e(t)$ has been described as the discrepancy between the actual output of the neural network τ_j and the desired output τ_j^d. The zero value of the learning error coordinate $e(t)$ has been defined as a time-varying sliding surface:

$$S(e(t)) = e(t) = \tau_j(t) - \tau_j^d(t) = 0 \tag{4.18}$$

which is the condition that guarantees that the neural network output $\tau_j(t)$ coincides with the desired output signal $\tau_j^d(t)$ for all time $t > t_{hit}$, where t_{hit} is the hitting time of $e(t) = 0$.

The learning algorithm for the neural network weights $w_{hi}^k(t)$ and $w_{ij}(t)$ should be derived in such a way that the sliding mode condition of Definition 4.1 will be enforced.

We choose the Lyapunov function candidate as follows:

$$V(e(t)) = \frac{1}{2}e^2(t) = \frac{1}{2}\left(\tau_j(t) - \tau_j^d(t)\right)^2 = 0 \tag{4.19}$$

Then differentiating $V(e(t))$ yields:

$$\dot{V}(e(t)) = e\dot{e} = e(\dot{\tau}_j - \dot{\tau}_j^d) \tag{4.20}$$

To find the time derivative of $\dot{\tau}_j$:

$$\begin{aligned}\frac{d\tau_j}{dt} &= \frac{\partial \tau_j}{\partial w_{ij}}\frac{dw_{ij}}{dt} + \frac{\partial \tau_j}{\partial w_{hi}^k}\frac{dw_{hi}^k}{dt} \\ &= \frac{\partial \tau_j}{\partial w_{ij}}\dot{w}_{ij} + \frac{\partial \tau_j}{\partial t_i}\frac{\partial t_i}{\partial x_i(t_i)}\frac{\partial x_i(t_i)}{\partial w_{hi}^k}\dot{w}_{hi}^k\end{aligned} \tag{4.21}$$

The first term in Equation 4.21 can be computed as

$$\frac{\partial \tau_j}{\partial w_{ij}} = \frac{\partial\left\{\sum_{i\in\Gamma_j} w_{ij}t_i\right\}}{\partial w_{ij}} = t_i \tag{4.22}$$

The derivative of the neural network output with respect to the firing times of the neurons in the hidden layer yields

$$\frac{\partial \tau_j}{\partial t_i} = \frac{\partial\left\{\sum_{i\in\Gamma_j} w_{ij}t_i\right\}}{\partial t_i} = w_{ij} \tag{4.23}$$

The terms $\frac{\partial t_i}{\partial x_i(t_i)}$ and $\frac{\partial x_i(t_i)}{\partial w_{hi}^k}$ can be computed in a similar manner to Equation 4.10 and Equation 4.11

$$\frac{\partial t_i}{\partial x_i(t_i)} = \frac{-1}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{4.24}$$

and

$$\frac{\partial x_i(t_i)}{\partial w_{hi}^k} = \frac{\partial \left\{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k y_h^k(t_i)\right\}}{\partial w_{hi}^k} = y_h^k(t_i) \tag{4.25}$$

Thus, the time derivative of τ_j can be written as:

$$\dot{\tau}_j = t_i\dot{w}_{ij} - \frac{w_{ij}y_h^k(t_i)}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}}\dot{w}_{hi}^k \tag{4.26}$$

Using Equation 4.26, Equation 4.20 can be written as:

$$\dot{V} = e\left[t_i\dot{w}_{ij} - \frac{w_{ij}y_h^k(t_i)}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}}\dot{w}_{hi}^k - \dot{\tau}_j^d\right] \tag{4.27}$$

If the learning algorithm for the weights w_{ij} and w_{hi}^k is chosen as

$$\dot{w}_{ij} = -\alpha t_i sign(e) \tag{4.28}$$

$$\dot{w}_{hi}^k = \alpha w_{ij}y_h^k(t_i)\left\{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k\frac{\partial y_h^k(t_i)}{\partial t_i}\right\}sign(e) \tag{4.29}$$

then the derivative of τ_j will be

$$\begin{aligned}\dot{\tau}_j &= -\alpha\left(t_i\right)^2 sign(e) - \alpha\frac{w_{ij}^2 y_h^k(t_i)^2}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k\frac{\partial y_h^k(t_i)}{\partial t_i}} \\ &\quad\left\{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k\frac{\partial y_h^k(t_i)}{\partial t_i}\right\}sign(e) \\ &= -\alpha\left(t_i\right)^2 sign(e) - \alpha w_{ij}^2 y_h^k(t_i)^2 sign(e)\end{aligned} \tag{4.30}$$

We consider that the desired spike time of the output neuron is constant, i.e.

$\dot{\tau}_j^d = 0$. For the selected weight update rule, the derivative of the Lyapunov candidate function yields

$$\begin{aligned}\dot{V} &= e\left[-\alpha\left(t_i\right)^2 sign(e) - \alpha w_{ij}^2 y_h^k(t_i)^2 sign(e)\right] \\ &= -\alpha\left(t_i\right)^2 |e| - \alpha w_{ij}^2 y_h^k(t_i)^2 |e| \\ &= -\alpha|e|\left[(t_i)^2 + w_{ij}^2 y_h^k(t_i)^2\right] \end{aligned} \tag{4.31}$$

This implies

$$\dot{V} < 0, \quad \forall e \neq 0 \tag{4.32}$$

The design steps of the two layered SNN have been given in 4.2.

Setup the network;
Initialize weights w_{hi}^k and w_{ij}randomly within the interval $[0, 1]$;
for each input neuron h **do**
  Convert the sensor signal to the spike time t_h using Equation 2.10;
end for
Set simulation time $t = 0$;
for each neuron i in the hidden layer **do**
  while $t < max\_steps$ **do**
    Calculate $x_i(t)$ using Equation 2.8 and Equation 2.9;
    if $x_i(t) < \theta$ **then**
      Set $t = t + time\_step$;
    else
      Set the time instance t, where the membrane potential $x_i(t)$ crosses the threshold value, as the firing time of neuron i: $t_i = t$;
    end if
    if all neurons in the hidden layer fire **then**
      Set $t = max\_steps$;
    end if
  end while
end for
for each neuron j in the output layer **do**
  Compute output $\tau_j \sum_{i \in \Gamma_j} w_{ij} t_i$;
end for
Set error $e = \sum_{j \in J} (\tau_j - \tau_j^d)$;
for each neuron h in the input layer **do**
  for each neuron i in the hidden layer **do**
    for each synapse k **do**
      Calculate $\dot{w}_{hi}^k$ using Equation 4.29;
      Set the new weights $w_{hi}^k = w_{hi}^k + \dot{w}_{hi}^k$;
    end for
  end for
end for
for each neuron i in the hidden layer **do**
  for each neuron j in the output layer **do**
    Calculate $\dot{w}_{ij}$ using Equation 4.28;
    Set the new weights $w_{ij} = w_{ij} + \dot{w}_{ij}$;
  end for
end for
Save parameters of SNN;

Figure 4.2. The SMC-based learning algorithm for a two layered SNN.

4.3. Application of SMC-based Learning Algorithms for Identification and Control Tasks

To assess the performance of the SNN for the proposed SMC-based learning algorithm, experimental studies carried out on the real time laboratory servo system for identification and control problems. The details of this test bed has been presented in the previous chapter. ABS, which constitutes a challenging control task owing to its strong nonlinear characteristics, has been also used in the control performance studies.

4.3.1. System Identification Experiments on the Servo System Amira DR300

The identifier used in this study has two inputs. These are the one-step delayed value of the input signal along with the one-step delayed system output. The initial weights of the synapses are set randomly within the interval $[0, 1]$. Using the developed sliding mode based learning algorithm, these weights are adapted for a chirp excitation signal shown in Figure 4.3. The amplitude of the chirp signal is 3 Volts and its initial frequency is 0.1 Hz whereas its frequency becomes 1 Hz at the end of the 20^{th} second. With the use of a chirp signal as the excitation signal, the notion of richness in excitation is satisfied for better identification results and it becomes possible to have an output signal of varying amplitude and frequency, which yields a more challenging identification problem compared to one with fixed magnitude or frequency. The applied signal has both negative and positive values, which requires the motor to rotate in both directions.

The SNN used as the identifier has a single hidden layer with seven spiking neurons on it. It is assumed that there are four synapses for each connection among the neurons. The time decay constant and the threshold value are selected as 8 ms and 2, respectively. The sampling time is 10 ms. The results presented in Figure 4.4 indicate that the identifier can approach the plant output very fast, indicating correct modeling. As a performance criterion the root-mean-square-error (RMSE) defined in

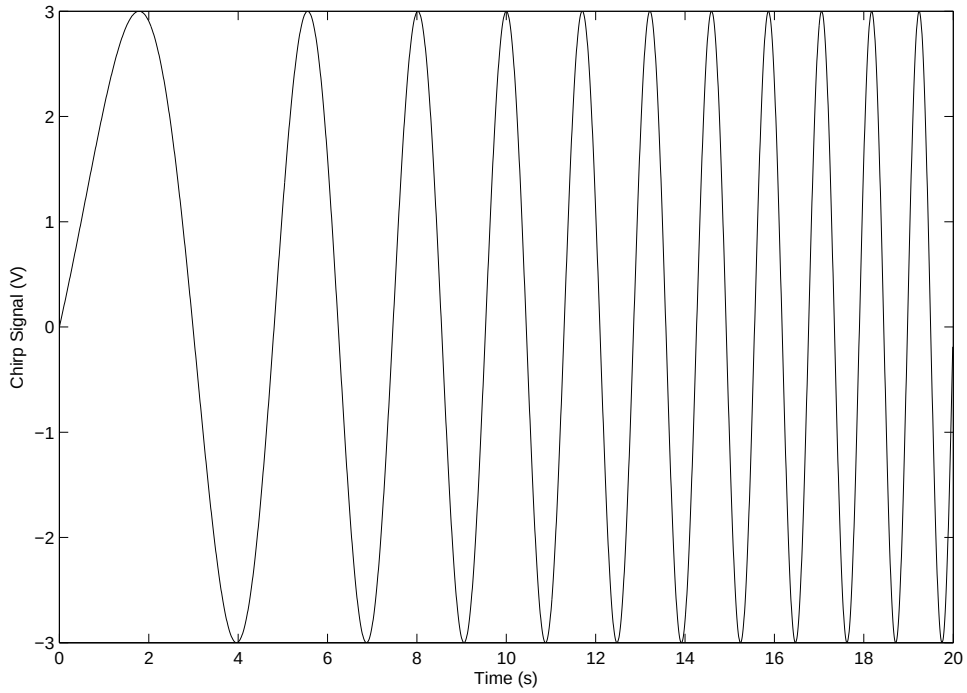


Figure 4.3. Chirp excitation signal.

Equation 4.33 is used.

$$RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(x_i^d - x_i)^2} \tag{4.33}$$

where K represents the total number of the samples, which is in this case 2000. The corresponding RMSE value for the training part is $0.35V$.

After the learning of the parameters, in the testing part, the signal $u(t) = 2 + sin(2\pi t)$ is applied as an input to the system. The experiment is run for 10 seconds resulting in 1000 data points. The corresponding RMSE value is $0.22V$.

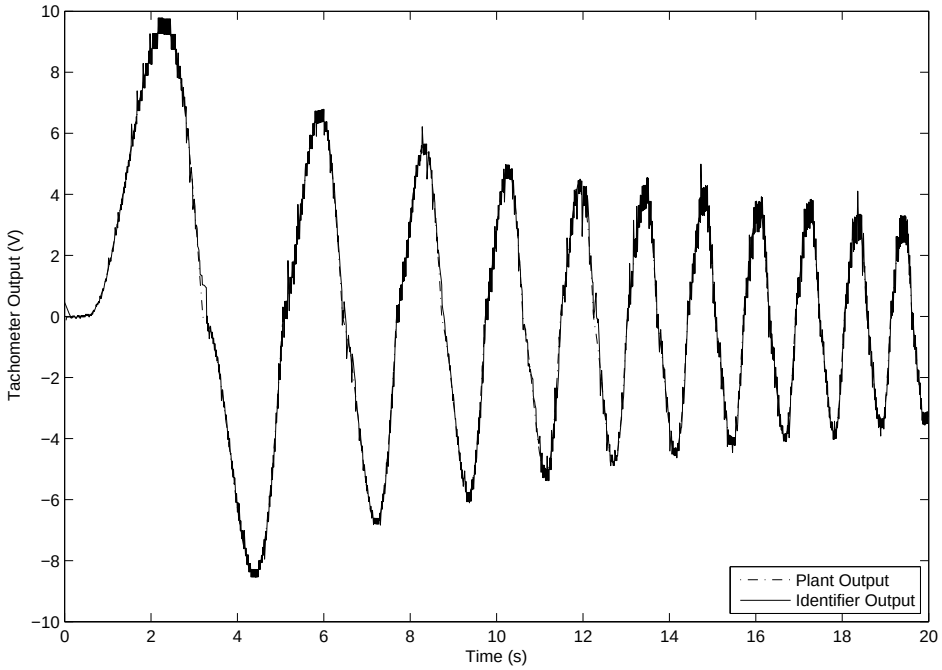


Figure 4.4. SNN identifier output vs. plant output.

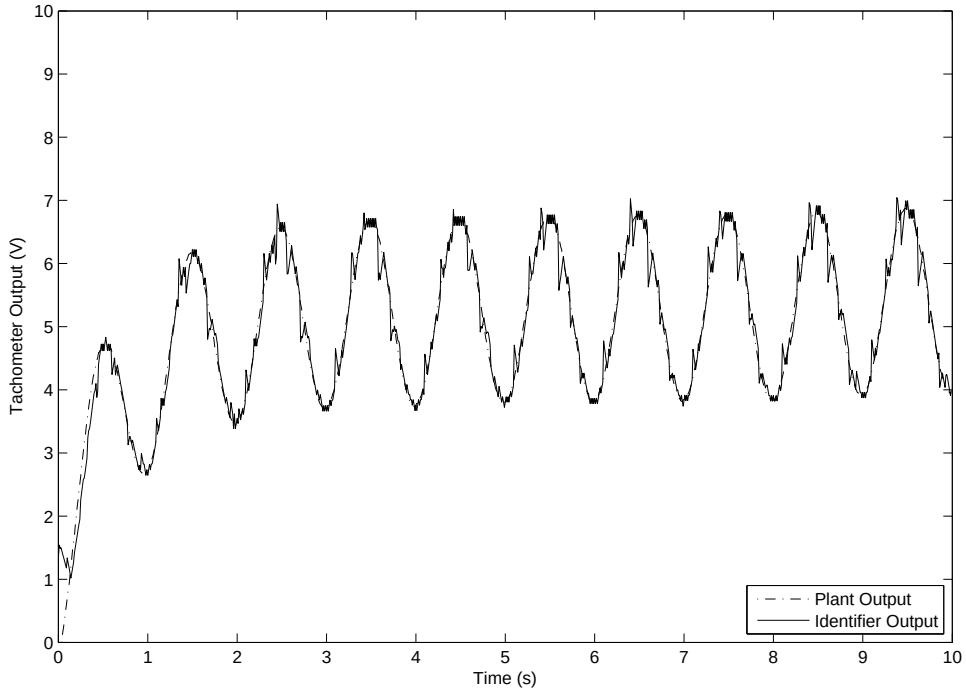


Figure 4.5. SNN identifier output vs. plant output.

4.3.2. Control Performance Studies

In order to make an in-depth comparison of the effectiveness of the gradient-based and VSS-based learning algorithms for SNNs, the parameter update rules for the former are also derived. The resulting update rules are as follows:

$$\Delta w_{ij} = -\eta(\tau_j - \tau_j^d)t_i \tag{4.34}$$

$$\Delta w_{hi}^k = \frac{\eta(\tau_j - \tau_j^d)\{\sum_{i\in\Gamma_j} w_{ij}\}y_h^k(t_i)}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{4.35}$$

where η is the learning constant.

4.3.2.1. DC motor speed control. The two layered SNN is composed of two input neurons, seven hidden layer neurons and one output neuron. The inputs are the error and the change of error. It is considered that the output neuron has a linear activation function. 4 synapses are used for each connection between the input layer and the hidden layer resulting in 63 parameters to be updated. The connections have a delay interval of 3 ms; hence the available synaptic delays are from 1 to 4 ms. The sampling time is set to 10 ms for all the experiments. The decay time constant is selected as $\tau = 6ms$ and the threshold value is set to $\theta = 3$. The initial value for the learning rate is selected as 0.0005 by the trial-and-error method considering the general facts that a small value of the learning rate results in a slow convergence whereas a higher value of the learning rate may result in oscillations.

Figure 4.6 presents the speed responses of the motor with the SNN controller for different set-point signals. The set point signal has been suddenly changed from 7 to 8 at the 4th second and then suddenly becomes 9 at the 7th second. It can be inferred that the SNN controller is capable of adapting its weights to regulate the system to changing set points.

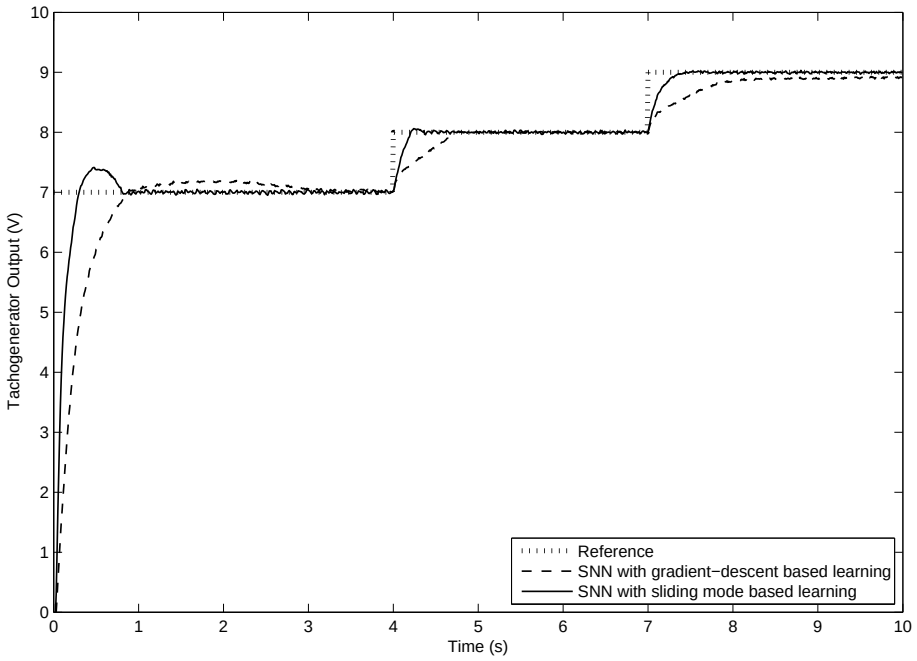


Figure 4.6. Speed response characteristic of SNN controller with SMC-based learning algorithm for different set-point signals.

In order to determine the efficiency and the accuracy of the proposed controllers, three different types of load conditions are considered. The speed responses of the motor with the SNN controller for step changing load condition is given in Figure 4.7. The corresponding load condition starts with a value of $0.015Nm$, and increases suddenly to $0.03Nm$ on 4ths, and then comes back to $0.015Nm$ on 7ths.

Figure 4.8 shows the speed response of the motor under the sinusoidal load condition, i.e. $M_L(t) = 0.03 + 0.006sin(0.0225t)\ \ Nm$. The corresponding load condition is presented in Figure 4.9. On the other hand, Figure 4.10 shows the speed response of the motor under load condition which is proportional to the square of the speed, i.e. $M_L(t) = 0.1125x10^{-3}(TachogeneratorOutput)^2\ \ Nm$.

In Table 1, experimental results obtained using multilayer feedforward neural networks with sigmoidal and hyperbolic tangent activation functions are also included. Similar to experimental studies carried on using Spiking Neural Networks, it is consid-

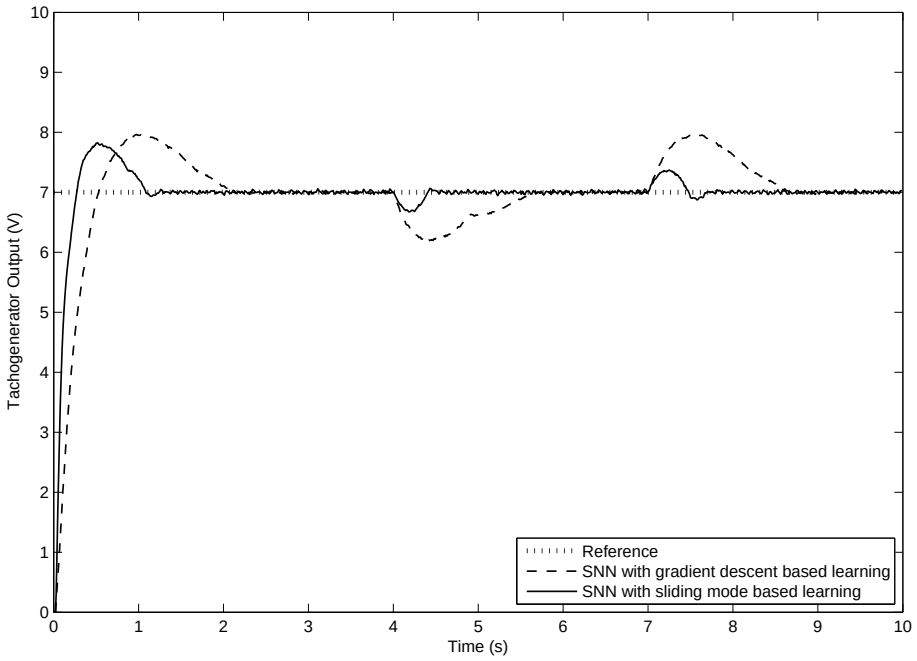


Figure 4.7. Speed response of the motor for step changing load.

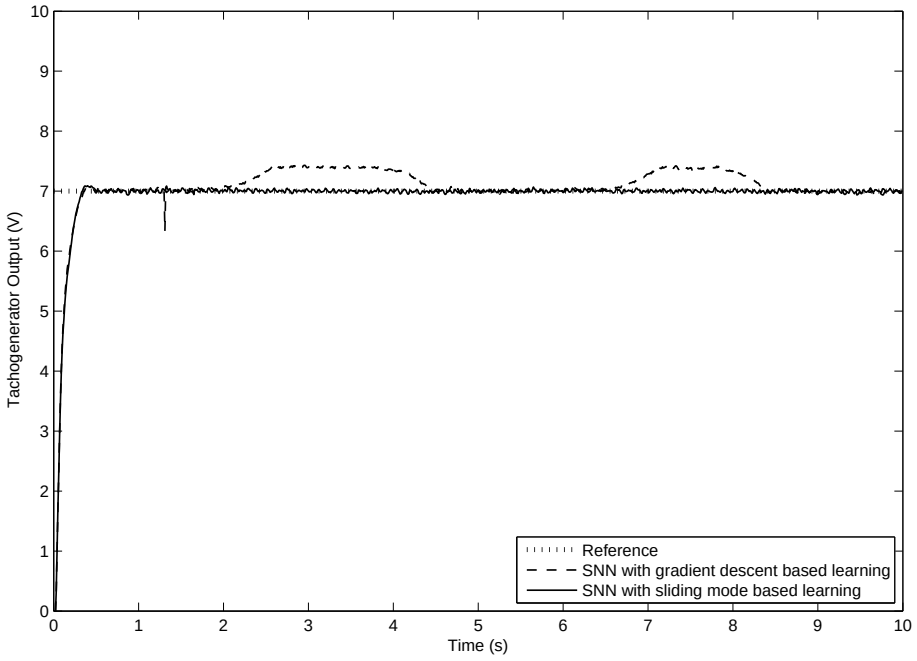


Figure 4.8. Speed response of the SNN controller with SMC-based learning algorithm for sinusoidal load condition.

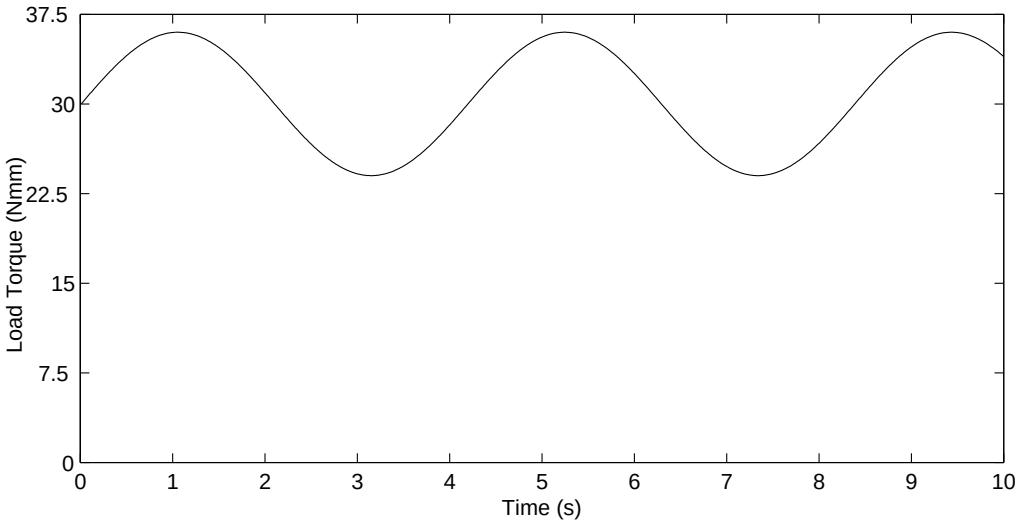


Figure 4.9. Sinusoidal changing load condition.

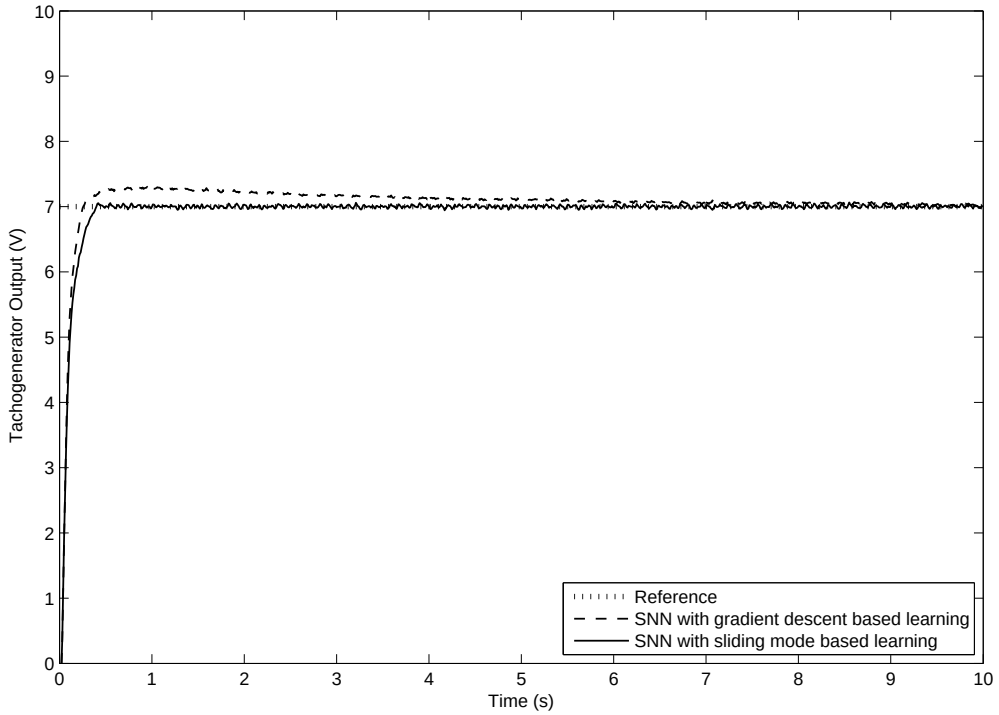


Figure 4.10. Speed response of the SNN controller with SMC-based learning algorithm for nonlinear load condition.

ered that these networks have seven neurons in their hidden layer and the neurons in the output layer employ a linear activation function. Regarding the results presented

Table 4.1. RMSE values for different load conditions.

RMSE	Step Changing Reference	Step Changing Load Condition	Sinusoidal Load Condition	Nonlinear Load Condition
MLP with sigmoid activation function	1.0166 V	0.9420 V	0.6985 V	0.6170 V
MLP with hyperbolic tangent activation function	0.9274 V	0.8928 V	0.6315 V	0.5884 V
SNN with gradient descent based learning	0.8953 V	0.9264 V	0.6293 V	0.5802 V
SNN with sliding mode based learning	0.6093 V	0.6149 V	0.5959 V	0.5784 V

in Table 1, it can be inferred that Spiking Neural Networks are capable of providing similar or smaller RMSE values compared to the second generation neural networks. These results are consistent with the statement [5] that a network consisting of spiking neurons has at least the same computational power as the neural networks of the first two generations. The experimental results also indicate that the SNN controller with a sliding mode-based learning algorithm can outperform SNN and MLP controllers with gradient-descent based learning algorithms for all types of load conditions. The proposed controller structure possesses shorter settling time and less perturbation form the reference value. Another advantage is that it exhibits reduced oscillations in the presence of load disturbances.

4.3.2.2. Control of the Antilock Braking System. In this study, the proposed SMC-based learning algorithm has been applied to the wheel slip regulation problem of an ABS, whose dynamics have been derived in the preceding Chapter. A two-layered SNN with 4 neurons in the hidden layer and a single spiking neuron with linear activation function in the output layer has been considered. The inputs of the SNN controller are the error, which is defined as the discrepancy between the current and desired value of the wheel slip, and its time derivative. Similar to the study on the SNN with

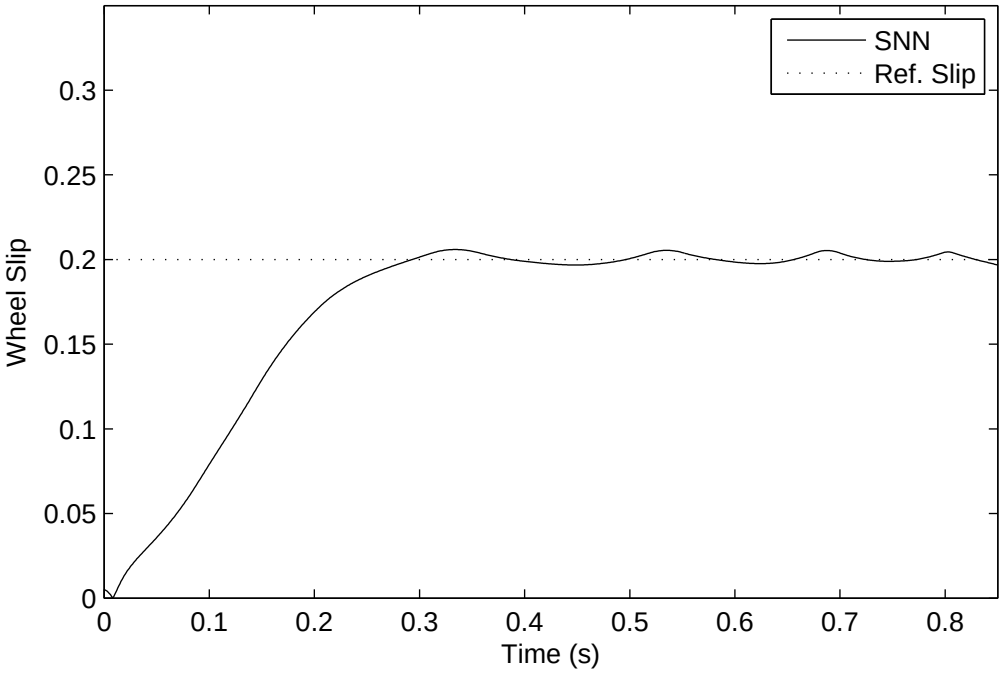


Figure 4.11. Wheel slip for SNN controller with SMC-based learning algorithm.

the gradient-based learning algorithm, the initial longitudinal velocity of the vehicle is taken as $V = 30m/s$ and the sampling time T_s is set to $1ms$ for all simulations. Two different approaches are followed to determine the reference value of the wheel slip. In the first approach the reference wheel slip is considered to be a constant value of 0.2. In the latter approach, the pseudo-static curve mentioned in Equation 3.24 is used for determining the reference value of the wheel slip. These input values are converted to the input spike times by using the encoding scheme given in Equation 2.10. As a linear activation function is used in the output layer, the need for decoding has been eliminated. For each connection between a presynaptic and postsynaptic neuron 5 synapses are assigned. The time delays for synaptic connections vary between $1ms$ and $5ms$ and the weights of the network are randomly initialized between 0 and 1. The decay time constant is selected as 7 and the threshold value is 2. The learning constant is determined by the trial-and-error method as 0.00025.

Figure 4.11 presents the resulting wheel slip values corresponding to the SNN controller for the proposed algorithm. The oscillations around the desired trajectory

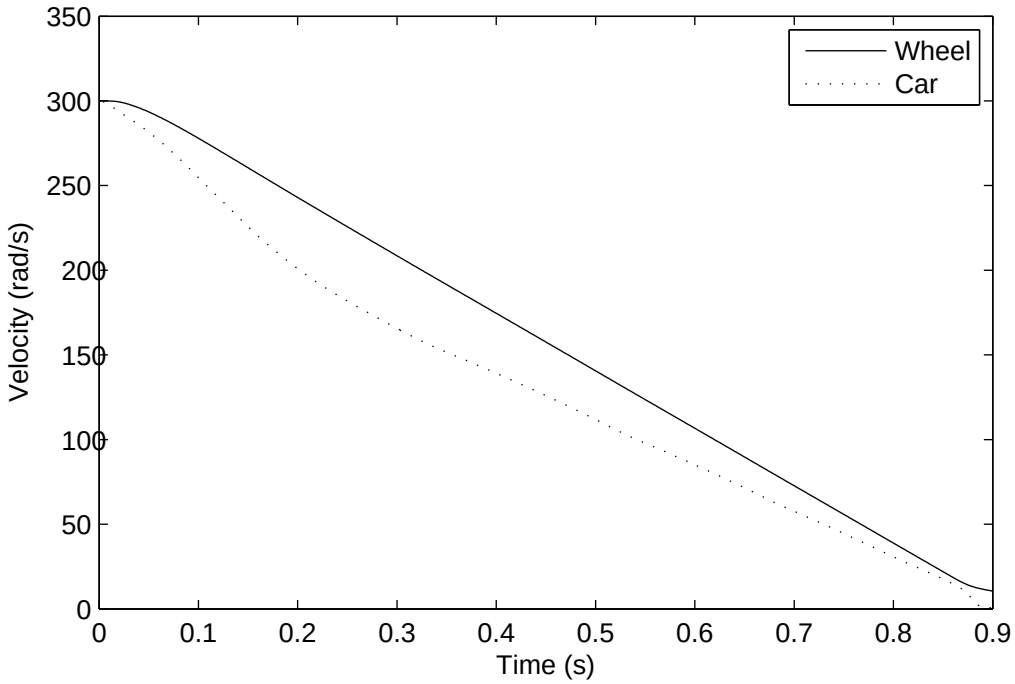


Figure 4.12. Vehicle and wheel speed for SNN controller with SMC-based learning algorithm.

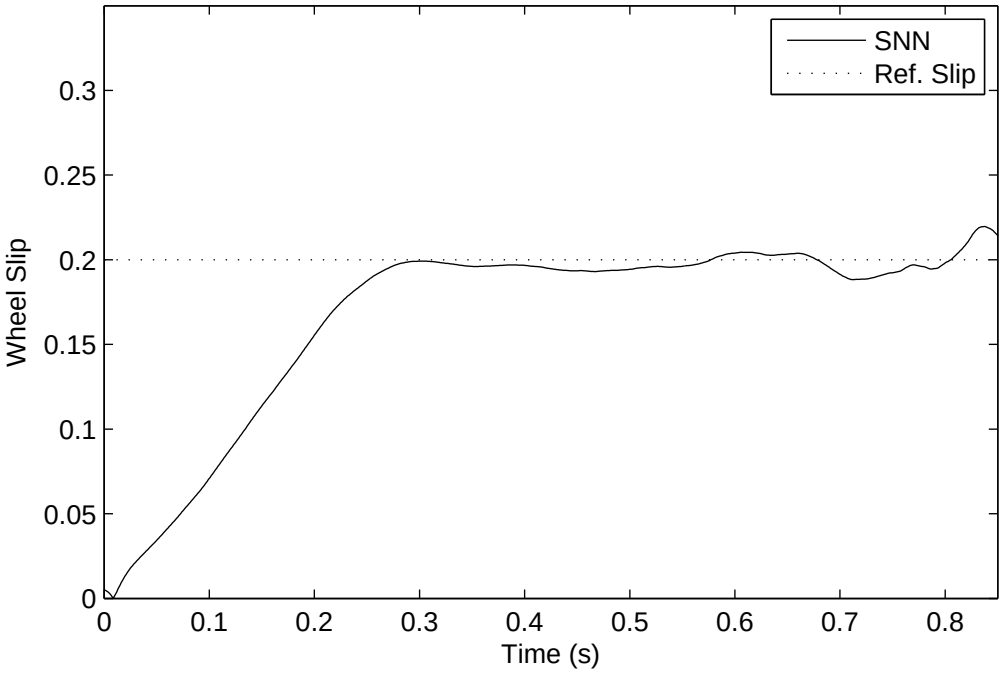


Figure 4.13. Wheel slip for SNN controller with SMC-based learning algorithm under noisy sensor measurements.

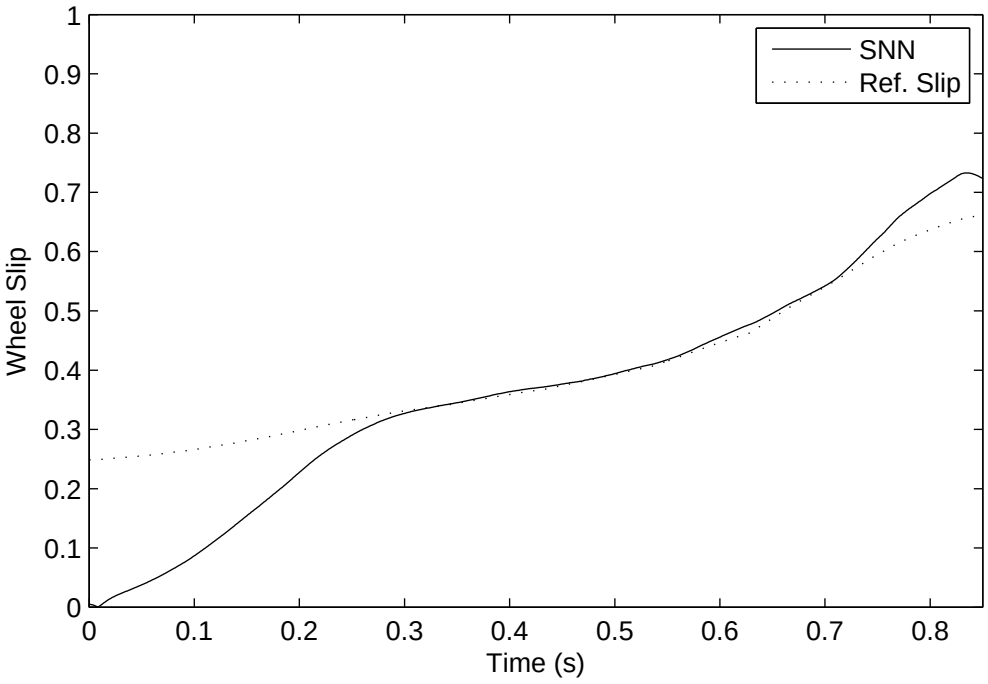


Figure 4.14. Wheel slip for SNN controller with SMC-based learning algorithm for velocity dependent reference value.

arise from the nature of the sliding mode control. As mentioned earlier in this Chapter, the main idea behind the sliding mode control is stated as:

- To direct the states of the controlled plant to a sliding surface $s(x,t)$
- To keep the states satisfying the condition $s(x,t) = 0$ once the sliding surface has been reached

To ensure the first item, it should be satisfied that $s\dot{s} < 0$ for any $s \neq 0$. This may require infinitely fast switching. However, in real life applications the switching is limited to finite frequency because of the imperfections in the switched controller [68]. In the case of finite frequency switching, the control is constant within a sampling interval, which may give rise to finite frequency, finite amplitude oscillations around the sliding surface. Although there are some research work on the chattering reduction problem of SMC, these are not investigated in the scope of this study.

To imitate the effect of the noise in the sensor measurements, band-limited white noise with a noise power of $5x10^{-6}$ is added to the slip measurements. The signal to noise ratio is 17 dB. Figure 4.13 demonstrates the robustness property of the SMC-based learning algorithm, as the noise doesn't cause any significant deviations from the reference value.

Figure 4.13 shows the dynamic response of the neurocontroller for the case when the reference value of the wheel slip varies over time with regard to the velocity of the car. Noise is also included to the slip measurements to imitate the real life conditions. The result shows that SNN controller can track the reference value with reduced oscillations.

4.4. Analysis and Discussion

The studies that demonstrate the robustness of variable structure control have motivated the use of the SMC approach in the training of ANNs. In this study, the development of a SNN with a VSS-based learning algorithm is considered for identification and control of dynamic plants. The parameter update rules are derived based on the Lyapunov stability method to achieve the stable online tuning of the neurocontroller parameters. The structure of the network is modified by replacing the spiking neurons in the output layer with the neurons having a linear activation function. Thus, the need for decoding has been eliminated resulting in a reduction in the computational time. The proposed algorithm is tested on both simulation studies with the ABS and real time experiments with the laboratory servo system for different load conditions. The results indicate that the SNNs with VSS-based learning algorithm can yield better transient and steady-state characteristics. Considering that in any practical application, the training data obtained are always subjected to noise (or maybe outliers), the proposed algorithm utilizing a variable-structure-systems based approach for online learning of SNNs can be effectively applied to a variety of engineering problems. The possible applications include but are not limited to robot control, industrial process

control, detection of various medical phenomena, and time series prediction.

5. FUZZY SPIKING NEURAL NETWORKS

Regarding the fact that all intelligent control techniques have some advantages and disadvantages making them suitable for specific applications, hybrid systems combining fuzzy logic and neural networks have been widely used in a variety of classification and control applications. Neural networks are very effective in determining a mapping between the inputs and outputs of a system, but they have a "black box" nature. This gives rise to the lack of knowledge about the causal relationships between these inputs and outputs. The parameters that have the most impact on a particular output cannot be distinguished among others, and thus neural networks fail to provide an insight into the system dynamics. On the other hand, fuzzy logic systems are good at explaining how they reach their decisions but they require expert knowledge to form the rules and the membership functions that they use to make these decisions. However, if the related knowledge is deficient, erroneous, or inexact fuzzy systems may require the adjustment of their membership functions in order to be able to provide the proper output. The tuning of the parameters is generally conducted in a heuristic way due to the lack of any formal approach for it. This heuristic approach might be very time consuming and error-prone. One of the most effective remedies to overcome this shortcoming is the use of a hybrid structure in which the learning ability of neural networks can be utilized to automate the parameter update process and reduce development time and cost substantially while improving performance.

The common learning algorithms employed in the neuro-fuzzy systems can be considered in two groups: Those relying on the methods based on gradient evaluation and those using evolutionary methods [69]. The gradient descent-based algorithms of the first group require the computation of partial derivatives of some cost function with respect to the parameters to be updated, which may give rise to the entrapment of the algorithm into a local minimum of the nonlinear objective function. Some other concerns about these learning algorithms can be stated as their sluggish learning

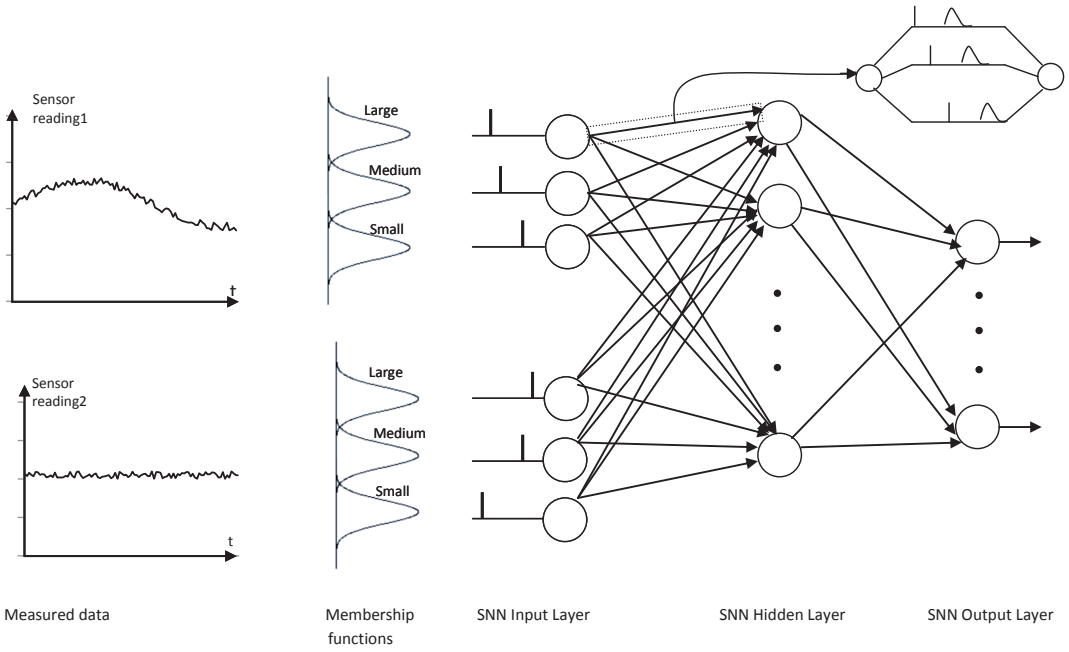


Figure 5.1. The structure of FSNN.

speed and the difficulty to obtain analytical results concerning the convergence and stability of the learning schemes [70]. On the other hand, evolutionary algorithms are capable of finding a global optimal solution in most cases, as they do not employ any derivatives. Nevertheless, the stability of such approaches is still questionable and the optimal values for the stochastic operators are difficult to derive. Additionally, the computational burden can be very high.

In this study, a sliding mode theory-based supervised training algorithm that implements fuzzy reasoning on a spiking neural network is developed and tested for the trajectory control problem of a robotic manipulator and wheel slip regulation problem of ABS. The stable online tuning of the parameters and a fast learning speed are the prominent characteristics of the proposed algorithm. The Lyapunov stability method has been adopted in the derivations of the conditions to enforce the learning error

toward zero in a stable manner.

5.1. Derivation of the Learning Rules

Consider the two-layered FSNN structure as shown in Figure 5.1. Once the sensor data are available, these signals are converted into the inputs of the SNN by fuzzy membership functions. It is assumed that there are p inputs/sensor reading and q membership functions assigned for each of these p inputs. Furthermore, it is considered that there are m neurons in the hidden layer and only one neuron with a linear activation function in the output layer.The following definitions for the derivation of the parameter update rules have been used:

- $X(t) = [\ x_1(t) \quad x_2(t) \quad \dots \quad x_p(t)\ ]^T$ - vector of the sensor readings
- $t_h(t) = [\ t_{h1}(t) \quad t_{h2}(t) \quad \dots \quad t_{hn}(t)\ ]^T$ - vector of the firing times of the input neurons
- $t_i(t) = [\ t_{i1}(t) \quad t_{i2}(t) \quad \dots \quad t_{im}(t)\ ]^T$ - vector of the firing times of the hidden neurons
- Γ_i - set of presynaptic neurons of the i^{th} neuron in the hidden layer
- Γ_j - set of presynaptic neurons of the j^{th} neuron in the output layer
- $\tau_j(t)$ - scalar output of the network
- $c(p,q)$ - the mean of the Gaussian membership function for the sensor reading p and associated membership function q
- $\sigma(p,q)$ - the variance of the Gaussian membership function for the sensor reading p and associated membership function q
- $w_{hi}^k(t)$ - time-varying weight of the connections between the neuron h in the input layer and the node i in the hidden layer for the synaptic terminal k
- $w_{ij}(t)$ - time-varying weight of the connections between the neuron i in the hidden layer and the node j in the output layer

In this study, Gaussian membership functions have been used to compute the firing times of the neurons in the input layer. To calculate the firing time of the neuron h in the input layer associated with the input p and membership function q, the following formula has been employed:

$$t_h\left((p-1)*q+q\right) = exp\left(-\frac{(x_p - c_{p,q})^2}{2\sigma_{p,q}^2}\right) \tag{5.1}$$

Once the firing times of the neurons in the input layer are computed, the next step is to compute the firing times of hidden layer neurons using Equation 2.8 and Equation 2.9. The neurons in the output layer are considered with linear activation functions to eliminate the need for decoding and reduce the computational burden.

The vector $t_h(t) = [\ t_{h1}(t) \quad t_{h2}(t) \quad \dots \quad t_{hn}(t)\ ]^T$ of the firing times of the spiking neurons in the input layer and its time derivative $\dot{t_h}(t) = [\ \dot{t_h}1(t) \quad \dot{t_h}2(t) \quad \dots \quad \dot{t_h}n(t)\ ]^T$ as well as the input vector $X(t) = [\ x_1(t) \quad x_2(t) \quad \dots \quad x_p(t)\ ]^T$ are assumed to be bounded. The scalar signal $\tau_j^d(t)$ represents the time-varying desired output of the neural network. It will be assumed that $\tau_j^d(t)$ and $\dot{\tau}_j^d(t)$ are also bounded signals, i.e. $|\tau^d(t)| \leq B_{\tau_j^d}$ and $|\dot{\tau}_j^d(t)| \leq B_{\dot{\tau}_j^d}$, where $B_{\tau_j^d}$ and $B_{\dot{\tau}_j^d}$ are positive constants.

The learning error $e(t)$ has been defined as the discrepancy between the actual output of the neural network τ_j and the desired output τ_j^d. The zero value of the learning error coordinate $e(t)$ can be expressed as a time-varying sliding surface:

$$S(e(t)) = e(t) = \tau_j(t) - \tau_j^d(t) = 0 \tag{5.2}$$

which is the condition that guarantees that the neural network output $\tau_j(t)$ coincides with the desired output signal $\tau_j^d(t)$ for all time $t > t_{hit}$, where t_{hit} is the hitting time of $e(t) = 0$.

The learning algorithm for the neural network weights $w_{hi}^k(t)$ and $w_{ij}(t)$ and for the membership function parameters $c(p,q)$ and $\sigma(p,q)$ should be derived in such a way that the sliding mode condition of the Definition 4.1 will be enforced. The Lyapunov function candidate has been chosen as follows:

$$V(e(t)) = \frac{1}{2}e^2(t) = \frac{1}{2}\left(\tau_j(t) - \tau_j^d(t)\right)^2 = 0 \tag{5.3}$$

Then differentiating $V(e(t))$ yields:

$$\dot{V}(e(t)) = e\dot{e} = e(\dot{\tau}_j - \dot{\tau}_j^d) \tag{5.4}$$

To find the time derivative of τ_j, the chain rule has been applied:

$$\begin{aligned} \frac{d\tau_j}{dt} &= \frac{\partial\tau_j}{\partial w_{ij}}\frac{dw_{ij}}{dt} + \frac{\partial\tau_j}{\partial w_{hi}^k}\frac{dw_{hi}^k}{dt} + \frac{\partial\tau_j}{\partial c_{p,q}}\frac{dc_{p,q}}{dt} + \frac{\partial\tau_j}{\partial\sigma_{p,q}}\frac{d\sigma_{p,q}}{dt} \\ &= \frac{\partial\tau_j}{\partial w_{ij}}\dot{w}_{ij} + \frac{\partial\tau_j}{\partial w_{hi}^k}\dot{w}_{hi}^k + \frac{\partial\tau_j}{\partial c_{p,q}}\dot{c}_{p,q} + \frac{\partial\tau_j}{\partial\sigma_{p,q}}\dot{\sigma}_{p,q} \end{aligned} \tag{5.5}$$

The first and second terms on the right hand side of Equation 5.5 have already been computed in the previous chapter:

$$\frac{\partial\tau_j}{\partial w_{ij}} = \frac{\partial\left\{\sum_{i\in\Gamma_j} w_{ij}t_i\right\}}{\partial w_{ij}} = t_i \tag{5.6}$$

$$\frac{\partial\tau_j}{\partial w_{hi}^k} = -\frac{y_h^k(t_i)w_{ij}}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k\frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{5.7}$$

As mentioned earlier, the fuzzy logic systems require expert knowledge. If the knowledge is incomplete and/or uncertain, there is a need to tune the membership functions in order to obtain the desired performance. In this part of the study, the variable structured systems-based learning algorithm has been expanded to include the

update rules for the mean and variance of the membership functions that have been used to encode the sensor signals into the inputs for SNN. For this purpose, the term $\frac{\partial \tau_j}{\partial c_{p,q}}$ can be expanded as:

$$\frac{\partial \tau_j}{\partial c_{p,q}} = \frac{\partial \tau_j}{\partial t_h}\frac{\partial t_h}{\partial c_{p,q}} = \left(\sum_{i\in\Gamma_h}\frac{\partial \tau_j}{\partial t_i}\frac{\partial t_i}{\partial x_i(t_i)}\frac{\partial x_i(t_i)}{\partial t_h}\right)\frac{\partial t_h}{\partial c_{p,q}} \tag{5.8}$$

From Equation 4.23 and Equation 4.24, the term $\frac{\partial \tau_j}{\partial x_i(t_i)}$ yields:

$$\frac{\partial \tau_j}{\partial x_i(t_i)} = \frac{-w_{ij}}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{5.9}$$

The next term can be computed as:

$$\frac{\partial x_i(t_i)}{\partial t_h} = \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_h} = -\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \tag{5.10}$$

The derivative of t_h with respect to the center $c_{p,q}$ can be derived as:

$$\frac{\partial t_h}{\partial c_{p,q}} = t_h \frac{x_p - c_{p,q}}{\sigma_{p,q}^2} \tag{5.11}$$

If the terms from Equation 5.9 to (5.11) are combined, the following statement can be obtained for $\frac{\partial \tau_j}{\partial c_{p,q}}$:

$$\frac{\partial \tau_j}{\partial c_{p,q}} = \sum_{i\in\Gamma_h}\left(\frac{-w_{ij}}{\sum_{h\in\Gamma_i}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}}\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_h}\right) t_h \frac{x_p - c_{p,q}}{\sigma_{p,q}^2} \tag{5.12}$$

Similarly the term $\frac{\partial \tau_j}{\partial \sigma_{p,q}}$ can be expanded as

$$\frac{\partial \tau_j}{\partial \sigma_{p,q}} = \frac{\partial \tau_j}{\partial t_h}\frac{\partial t_h}{\partial c_{p,q}} = \left(\sum_{i\in\Gamma_h}\frac{\partial \tau_j}{\partial t_i}\frac{\partial t_i}{\partial x_i(t_i)}\frac{\partial x_i(t_i)}{\partial t_h}\right)\frac{\partial t_h}{\partial \sigma_{p,q}} \tag{5.13}$$

The derivative of t_h with respect to $\sigma_{p,q}$ can be derived as:

$$\frac{\partial t_h}{\partial \sigma_{p,q}} = t_h \frac{(x_p - c_{p,q})^2}{\sigma_{p,q}^3} \tag{5.14}$$

Using Equation 5.14:

$$\frac{\partial \tau_j}{\partial \sigma_{p,q}} = \sum_{i \in \Gamma_h} \left(\frac{-w_{ij}}{\sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_h} \right) t_h \frac{(x_p - c_{p,q})^2}{\sigma_{p,q}^3} \tag{5.15}$$

Thus, the time derivative of τ_j can be written as:

$$\begin{aligned} \dot{\tau}_j = {} & t_i \dot{w}_{ij} - \frac{y_h^k(t_i) w_{ij}}{\sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \dot{w}_{hi}^k \\ & + \sum_{i \in \Gamma_h} \left(\frac{w_{ij} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}}{\sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \right) t_h \frac{x_p - c_{p,q}}{\sigma_{p,q}^2} \left(\dot{c}_{p,q} + \frac{(x_p - c_{p,q})}{\sigma_{p,q}} \dot{\sigma}_{p,q} \right) \end{aligned} \tag{5.16}$$

If the update rules are selected as:

$$\frac{dw_{ij}}{dt} = -\alpha t_i sgn(e) \tag{5.17}$$

$$\frac{dw_{hi}^k}{dt} = \alpha w_{ij} \sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} sgn(e) \tag{5.18}$$

$$\frac{dc_{p,q}}{dt} = -\alpha \sum_{i \in \Gamma_h} w_{ij} \left(\left\{ \sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \right\} \left\{ \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \right\} \right) (x_p - c_{p,q}) sgn(e) \tag{5.19}$$

$$\frac{d\sigma_{p,q}}{dt} = -\alpha \sum_{i \in \Gamma_h} w_{ij} \left(\left\{ \sum_{h \in \Gamma_i} \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \right\} \left\{ \sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \right\} \right) \sigma_{p,q} sgn(e) \tag{5.20}$$

then the derivative of τ_j will be

$$\begin{aligned}\dot{\tau}_j = &\left[-\alpha\left(t_i\right)^2 sgn(e) - \alpha y_h^k(t_i) w_{ij}^2 sgn(e)\right] \\ &+ e\left[-2\alpha \sum_{i\in\Gamma_h} w_{ij}^2 \left\{\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}\right\}^2\right] t_h \frac{(x_p - c_{p,q})^2}{\sigma_{p,q}^2} sgn(e)\end{aligned} \tag{5.21}$$

We consider that the desired value of the output neuron is constant, i.e. $\dot{\tau}_j^d = 0$. For the selected parameter update rules, the derivative of the Lyapunov candidate function yields

$$\begin{aligned}\dot{V} = &e\left[-\alpha\left(t_i\right)^2 sgn(e) - \alpha y_h^k(t_i) w_{ij}^2 sgn(e)\right] \\ &+ e\left[-2\alpha \sum_{i\in\Gamma_h} w_{ij}^2 \left\{\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}\right\}^2\right] t_h \frac{(x_p - c_{p,q})^2}{\sigma_{p,q}^2} sgn(e) \\ &= -\alpha|e|\left[\left(t_i\right)^2 + y_h^k(t_i) w_{ij}^2 + 2\left(\sum_{i\in\Gamma_h} w_{ij}\left\{\sum_k w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}\right\}\right)^2 t_h \frac{(x_p - c_{p,q})^2}{\sigma_{p,q}^2}\right]\end{aligned} \tag{5.22}$$

This implies

$$\dot{V} < 0, \quad \forall e \neq 0 \tag{5.23}$$

5.2. Control Performance Studies

5.2.1. Control of Antilock Braking System

In this part of the study, a number of computer simulated dynamic responses are obtained to investigate the performance of the proposed control algorithm. The sampling time used in the simulations is 1 ms. All figures below show simulation results

Setup the network;
Initialize weights w_{hi}^k and w_{ij} randomly within the interval $[0, 1]$;
Initialize the mean $c_{p,q}$ and variance $\sigma_{p,q}$ of the Gaussian membership functions;
for each sensor reading x_p **do**
  for each membership function q **do**
    Convert the sensor signal to the spike time t_h using Equation 5.1;
  end for
end for
Set simulation time $t = 0$;
for each neuron i in the hidden layer **do**
  while $t < max\_steps$ **do**
    Calculate $x_i(t)$ using Equation 2.8 and Equation 2.9;
    if $x_i(t) < \theta$ **then**
      Set $t = t + time\_step$;
    else
      Set the time instance t, where the membrane potential $x_i(t)$ crosses the threshold value, as the firing time of neuron i: $t_i = t$;
    end if
    if all neurons in the hidden layer fire **then**
      Set $t = max\_steps$;
    end if
  end while
end for
for each neuron j in the output layer **do**
  Compute output $\tau_j \sum_{i \in \Gamma_j} w_{ij} t_i$;
end for
Set error $e = \sum_{j \in J} (\tau_j - \tau_j^d)$;
for each neuron h in the input layer **do**
  for each neuron i in the hidden layer **do**
    for each synapse k **do**
      Calculate $\dot{w}_{hi}^k$ using Equation 5.18 and set the new weights $w_{hi}^k = w_{hi}^k + \dot{w}_{hi}^k$;
    end for
  end for
  for each sensor reading x_p **do**
    for each membership function q **do**
      Calculate $\dot{c}_{p,q}$ using Equation 5.19 and set the new mean value $c_{p,q} = c_{p,q} + \dot{c}_{p,q}$;
      Calculate $\dot{\sigma}_{p,q}$ using Equation 5.19 and set the new variance value $\sigma_{p,q} = \sigma_{p,q} + \dot{\sigma}_{p,q}$;
    end for
  end for
end for
for each neuron i in the hidden layer **do**
  for each neuron j in the output layer **do**
    Calculate $\dot{w}_{ij}$ using Equation 5.17 and set the new weights $w_{ij} = w_{ij} + \dot{w}_{ij}$;
  end for
end for
Save parameters of SNN;

Figure 5.2. SMC-based learning algorithm for a two layered FSNN.

for the quarter vehicle model of Section with initial longitudinal velocity of V = 30 m/s maneuvering on a straight line. The difference between the desired and actual values of wheel slip is fed to the FSNN controller as the input signal. Five membership functions have been assigned for this input. The neural network has 5 neurons in the hidden layer and one neuron in the output layer. There are 5 synapses between each presynaptic and postsynaptic neuron. The weights of the network are initialized randomly between 0 and 1. The decay time constant is selected as 5 and the threshold value is 2. The learning rate is set to 0.00025 by the trial-and-error method.

The first set of simulations have been conducted for the constant reference wheel slip value $\lambda_R = 0.2$. The actual value of the wheel slip can vary within the interval $[0, 1]$. Thus, the initial membership functions for the wheel slip error (error=wheel slip measured form the system-reference wheel slip) are distributed uniformly over the range $[-0.2, 0.8]$ with a variance of 0.1 (See Figure 5.3). In order to obtain more realistic results, band-limited noise with a power of $5x10^{-6}$ is added to the system at slip measurements.

The performance of the controller in regulating the wheel slip around its set value is illustrated in Fig 5.4. The simulation results obtained for the SNN structure with the SMC-based learning algorithm have been also included in the Figure to enable a proper comparison of both controllers. Although both controllers are capable of tracking the reference value successfully for noisy measurements, the FSNN with a SMC-based learning algorithm exhibits better transient and steady-state characteristics compared to the SNN structure. The resulting membership functions used to convert the wheel slip measurements to firing times can be seen in Figure 5.5.

Another set of simulations has been performed for the case when the reference wheel slip was considered as a function of vehicle velocity. Figure 5.7 show simulation results of both controllers for velocity dependent reference wheel slip. As the reference value of the wheel slip can have a value within the range $[0.2, 0.7]$, the initial member-

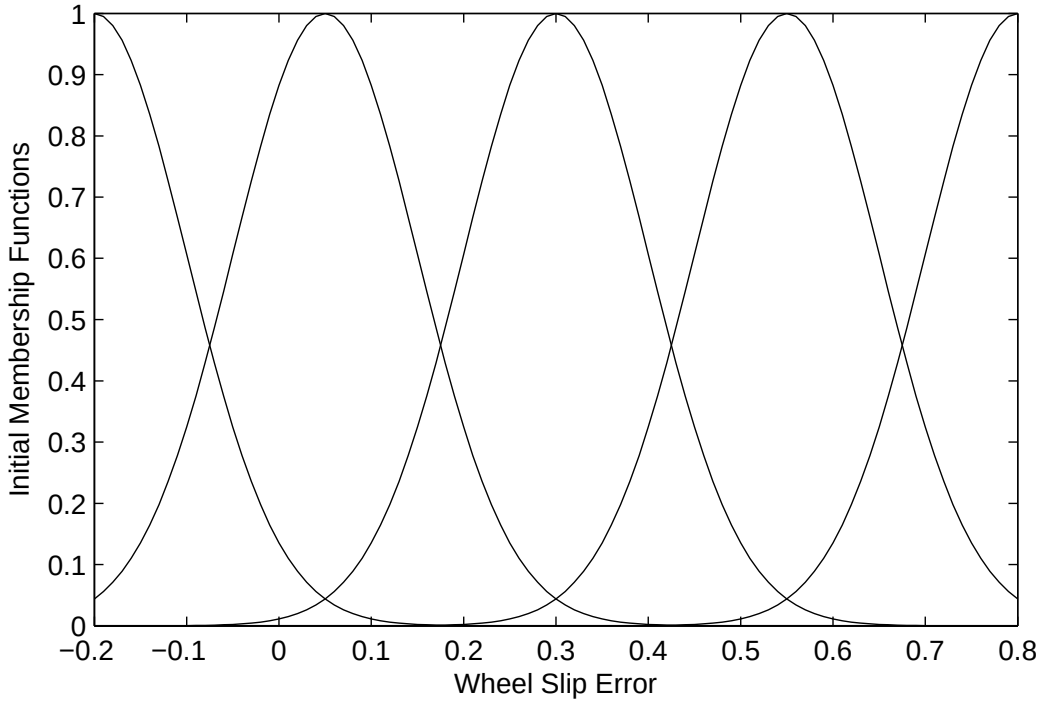


Figure 5.3. Initial membership functions for the wheel slip error.

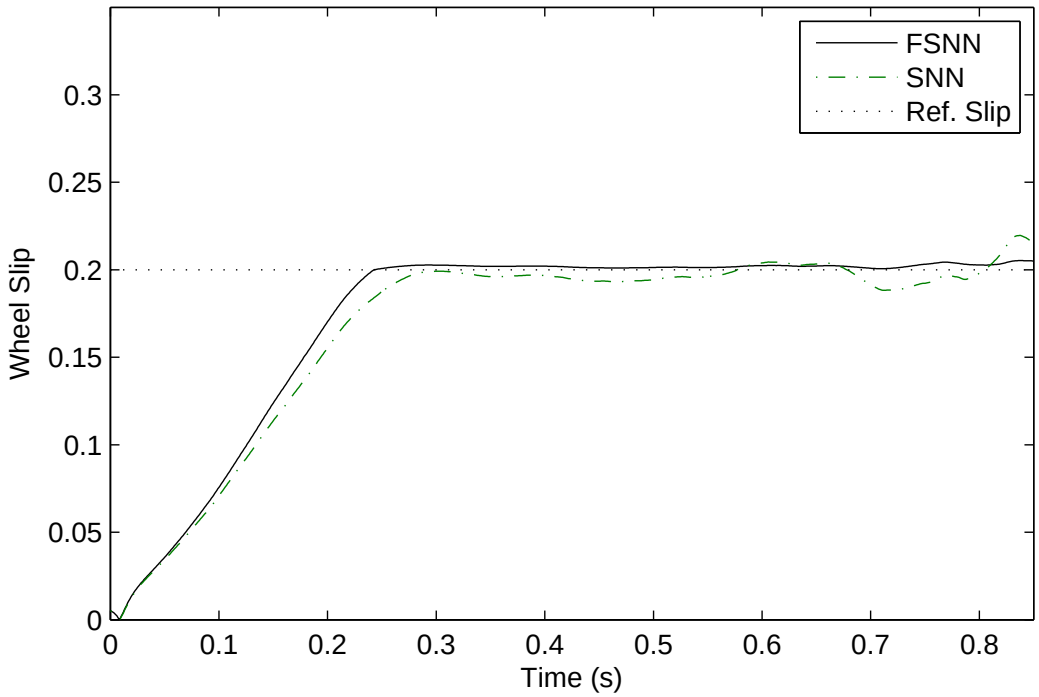


Figure 5.4. Wheel slip for FSNN and SNN for constant reference value.

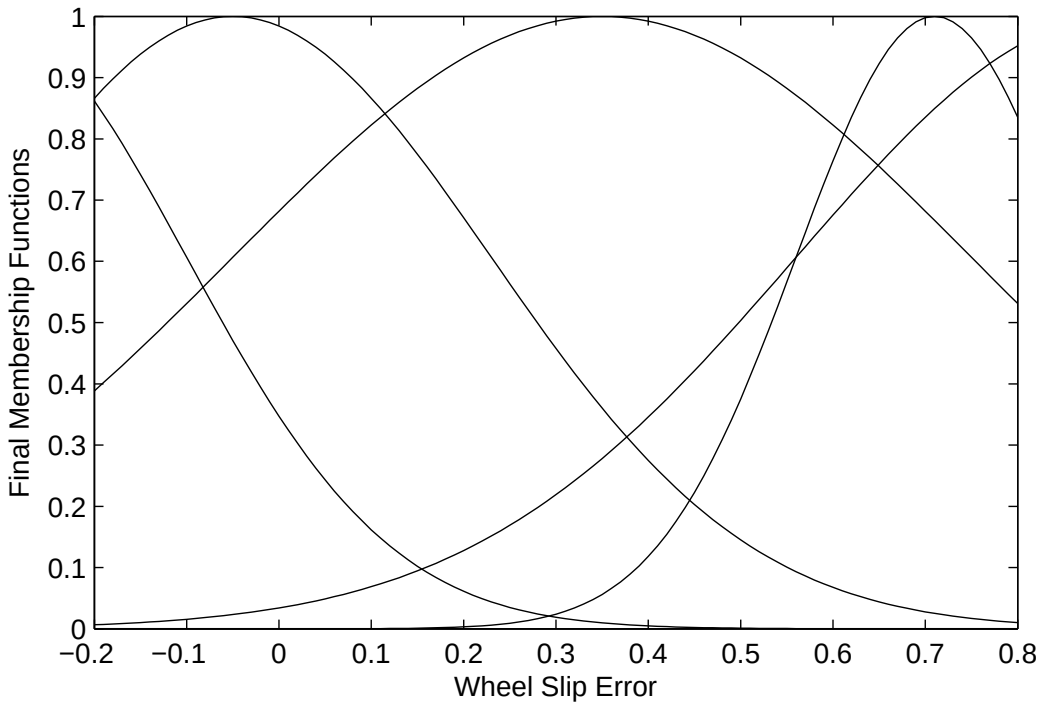


Figure 5.5. Final membership functions for the wheel slip error.

ship functions for the wheel slip are distributed uniformly over the range $[-0.7, 0.8]$ with a variance of 0.2 as shown in Figure 5.6. To simulate real life working conditions, noise is added to the slip measurements. The noise power is $5x10^{-6}$ and the SNR is 17 dB.

The performance of the FSNN and SNN controllers with SMC-based learning algorithms for velocity dependent reference value is presented in Fig 5.7. The simulation results indicate that both controllers can accurately track the reference wheel slip and they have matching settling times. For the FSNN controller, less deviations from the set point have been observed. The final mean and variance values of the membership functions can be seen in Figure 5.8.

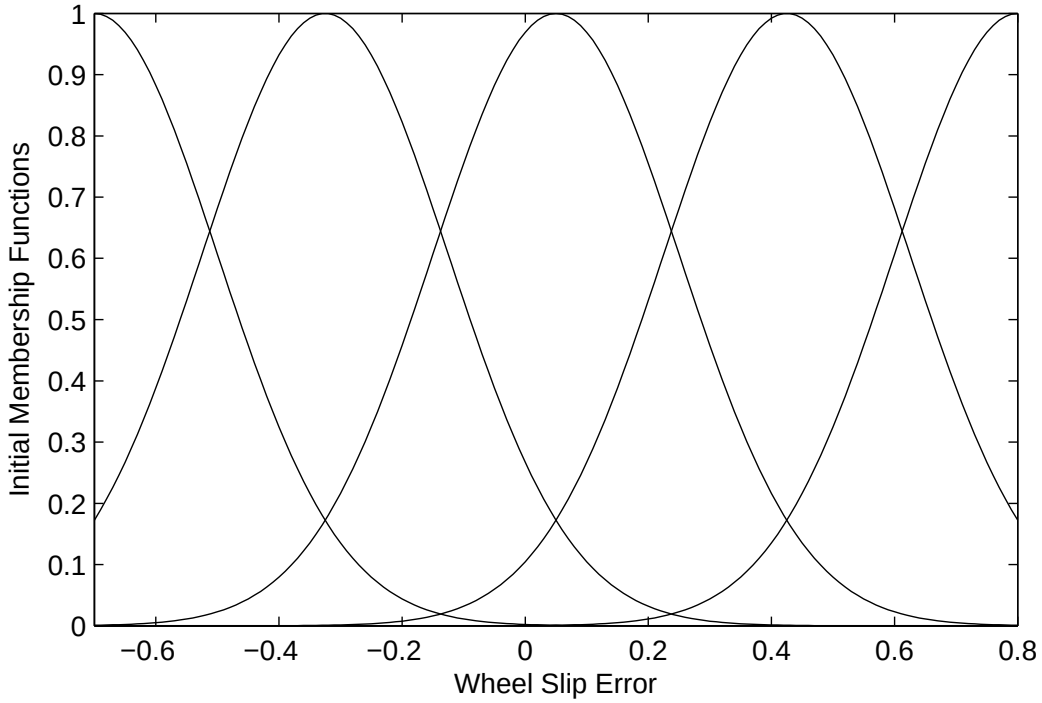


Figure 5.6. Initial membership functions for the wheel slip error for velocity dependent reference value.

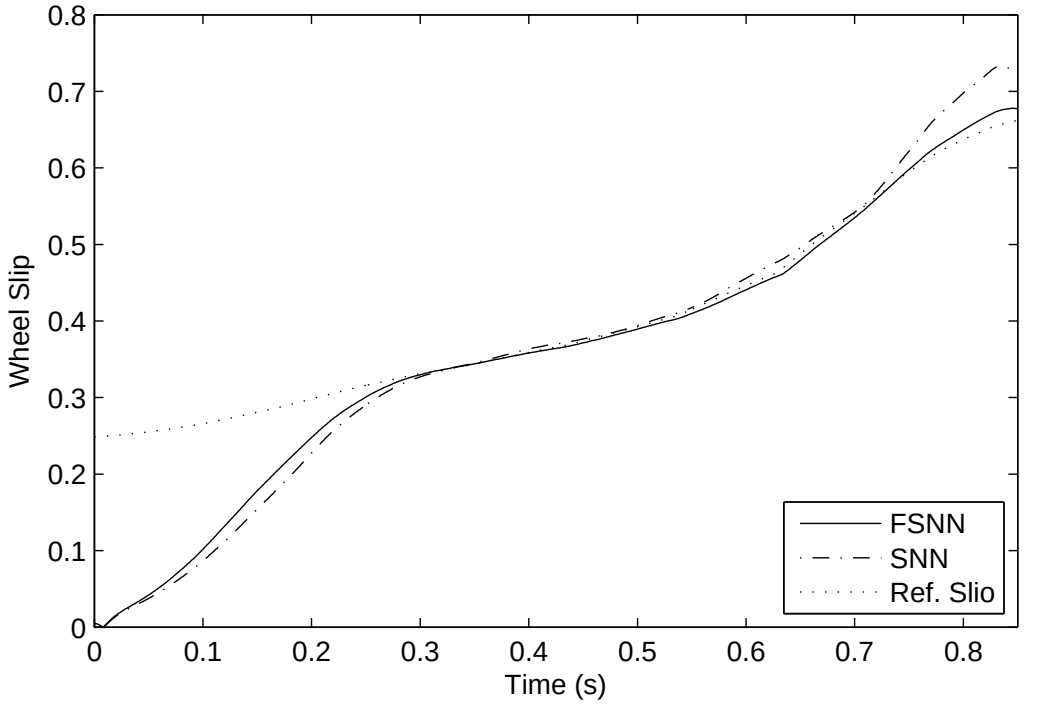


Figure 5.7. Wheel slip for FSNN and SNN for velocity dependent reference value.

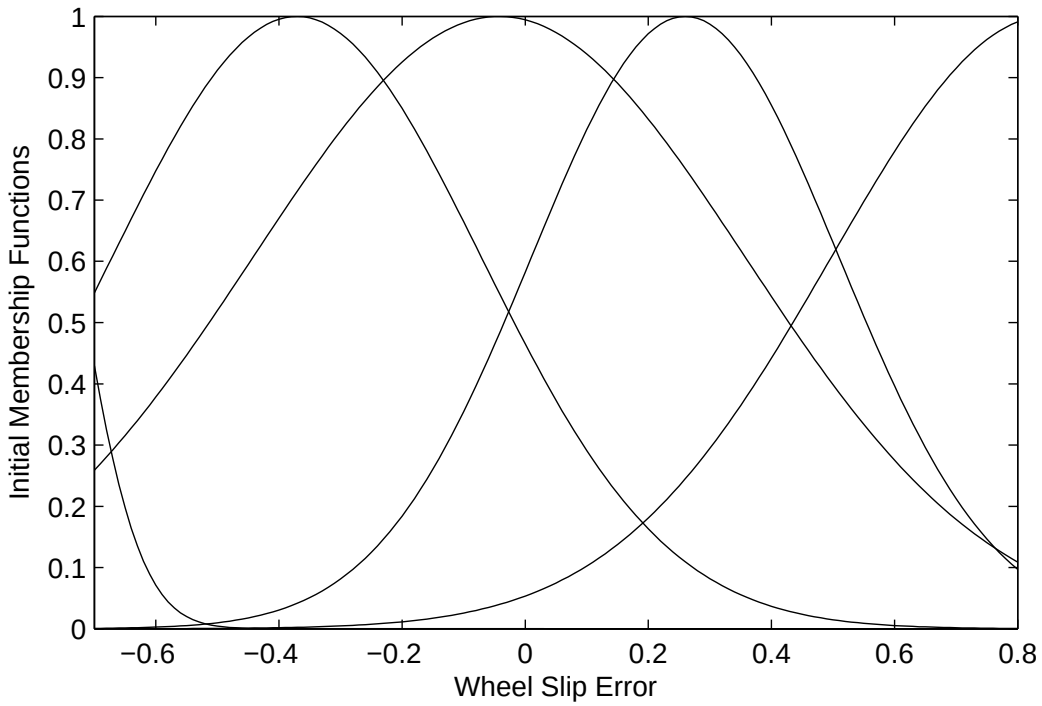


Figure 5.8. Final membership functions for the wheel slip error for velocity dependent reference value.

5.2.2. Position Control of a Two-dof Manipulator

To assess the performance of the proposed SMC-based learning algorithm, experimental studies have been carried out on the two-dof direct drive Scara type manipulator shown in Figure 5.9. The use of a direct drive system enables the user to evade the negative effects caused by the mechanical backlash and gear train compliance. A further advantage of these systems is that they are capable of delivering very high torques with very low levels of friction. On the other end of the spectrum, direct drive systems pose a challenging control problem since the design of the controller is optimized mainly for a particular set of parameters and in the absence of gear trains, payload and configuration variations may lead to an increased adverse effect on the performance of the system. Furthermore, the variations in the system parameters with time hamper the development of an accurate model and there are very often uncertainties associated with the load that the gripper carries. Model-free control methodologies based on com-

putational intelligence techniques have been widely used in the trajectory control of manipulators throughout the literature to overcome the shortcomings stemming from the lack of modeling information and inaccuracies in the measurements.

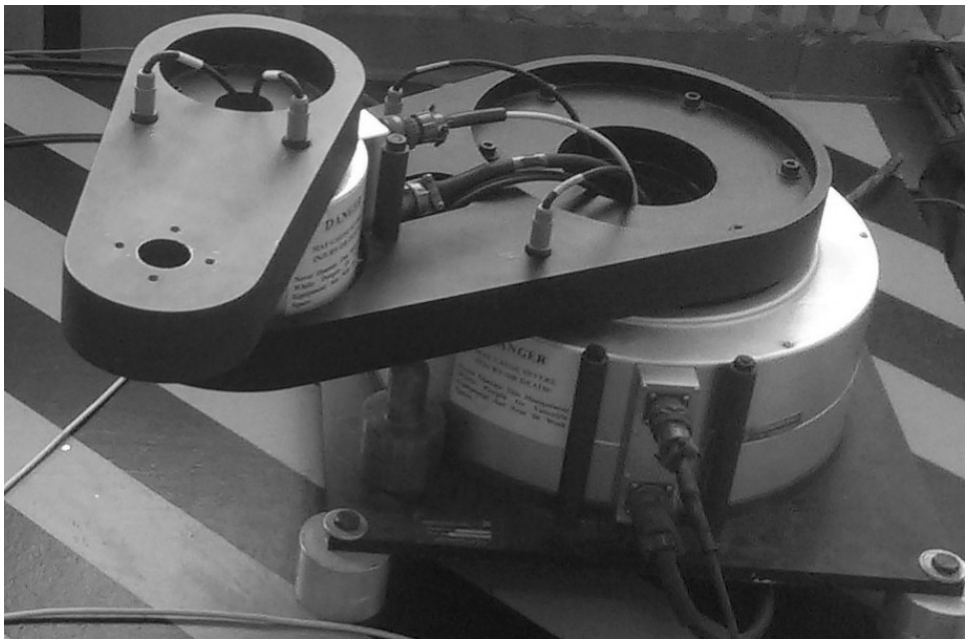


Figure 5.9. The robot arm.

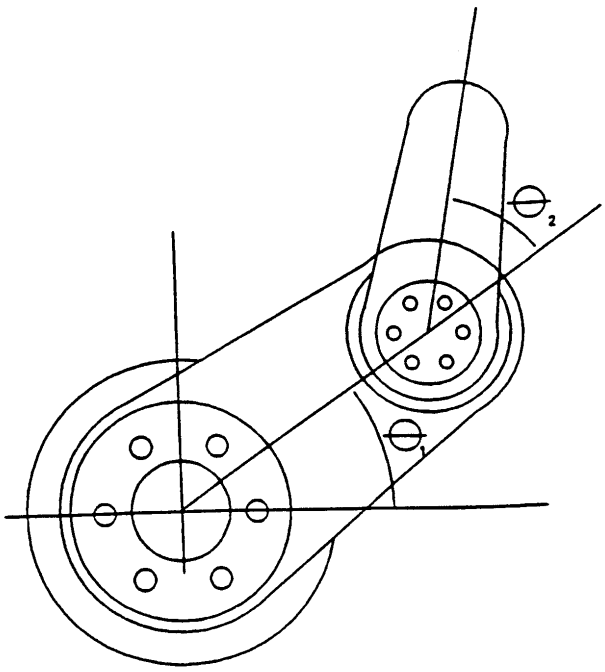


Figure 5.10. The assignment of joint angles.

The highly nonlinear and coupled dynamic equations of the manipulator can be

stated by the following equation:

$$M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + F = T \tag{5.24}$$

In this formulation $M(\theta)$ is the 2x2 state-varying inertia matrix of the manipulator, $V(\theta, \dot{\theta})$ is the two dimensional vector of centripetal and Coriolis forces, $\theta = [\ \theta_l \quad \theta_2\ ]^T$ is the vector of joint angles shown in Figure 5.10, T is the torque vector applied to the joints and F is two dimensional vector of Coulomb friction terms. M and V can be written explicitly as:

$$M(\theta) = \begin{pmatrix} p_l + 2p_3 cos(\theta_2) & p_2 + p_3 cos(\theta_2) \\ p_2 + p_3 cos(\theta_2) & p_2 \end{pmatrix} \tag{5.25}$$

$$V(\theta, \dot{\theta}) = \begin{pmatrix} -\dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2)p_3 sin(\theta_2) \\ \dot{\theta}_1^2 p_3 sin(\theta_2) \end{pmatrix} \tag{5.26}$$

where $p_1 = 2.0857 + 0.0576M_p$, $p_2 = 0.1168 + 0.0576M_p$ and $p_3 = 0.1630 + 0.0862M_p$. M_p denotes the mass of the payload. A detailed derivation of the dynamic model and the computation of the parameters p_l, p_2 and p_3 can be found in [71].

The work station consists of the two-dof direct drive Scara robot, a PC computer with Matlab and DS Control Desk software and DS1104 digital signal processing board. The control technique used in this study assumes a decentralized model where a controller is associated with each joint and a separate neural network is used to adjust the parameters of the controllers. Each FSNN controller, which is used to calculate the torque vector to be applied to the joints in order to obtain the desired position trajectory, consists of three layers. In the first layer, discrepancies between the encoder outputs of the elbow and shoulder links and the desired positions for these links are computed and fed to the system as the inputs of the control system. Three membership functions denoted by Negative, Zero, and Positive are used for each of these input

Table 5.1. System Parameters.

Motor 1 rotor inertia	0.2670	kgm^2
Shoulder link inertia	0.3340	kgm^2
Motor 2 rotor inertia	0.0075	kgm^2
Motor 2 stator inertia	0.0400	kgm^2
Elbow link inertia	0.0630	kgm^2
Motor 1 mass	73.0000	kg
Shoulder link mass	9.7800	kg
Motor 2 mass	3.0000	kg
Elbow link mass	4.4500	kg
Shoulder link length	0.3590	m
Elbow link length	0.2400	m
Shoulder link CG distance	0.1360	m
Elbow link CG distance	0.1020	m
Coulomb friction torque of shoulder link	4.900	Nm
Coulomb friction torque of elbow link	1.6700	Nm
Torque limit for Shoulder link	245.0000	Nm
Torque limit for Elbow link	39.2000	Nm

signals. Consequently, there are 2 different neural networks each having 3 neurons in the input layer of the FSNN. Using the initial Gaussian membership functions shown in Figure 5.11, the position error signals are converted into the firing times of the input layer. The initial values of the mean and variance of the membership functions have been selected such that the entire input space is covered with overlapping membership functions.

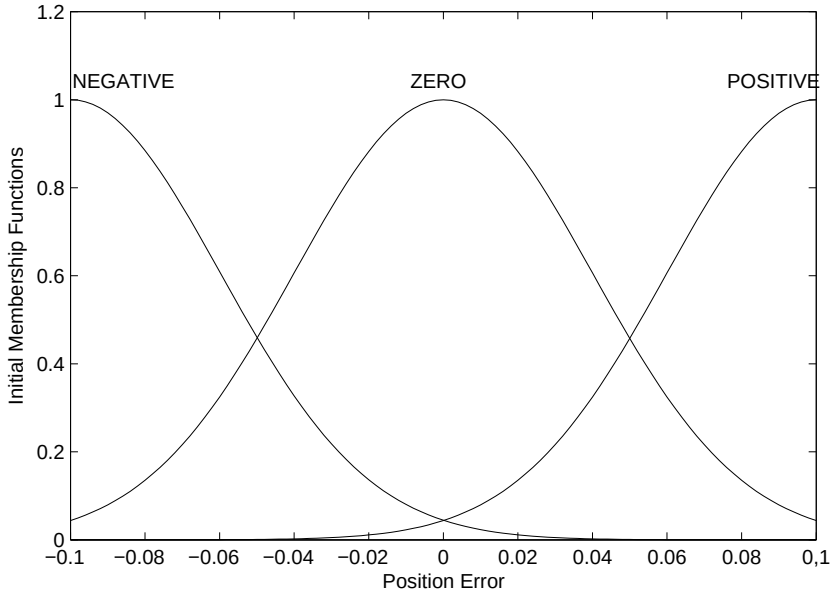


Figure 5.11. Initial membership functions for the position errors of the shoulder link and elbow link.

Initially the two FSNN controllers used for each link have the same structure and parameters except for their randomly generated weight values. In each network, there are 4 neurons in the hidden layer and the output neuron is assumed to have a linear activation function. 4 synapses are used for each connection between the input layer and the hidden layer resulting in 58 parameters to be updated for each FSNN. The connections have a delay interval of 3 ms; hence the available synaptic delays are from 1 to 4 ms. The sampling time is set to 10 ms for all the experiments. The decay time constant is selected as $\tau = 6ms$ and the threshold value is set to $\theta = 2$. The initial value for the learning rate is selected as 0.0005 by the trial-and-error method considering the general facts that a small value of the learning rate results in a slow

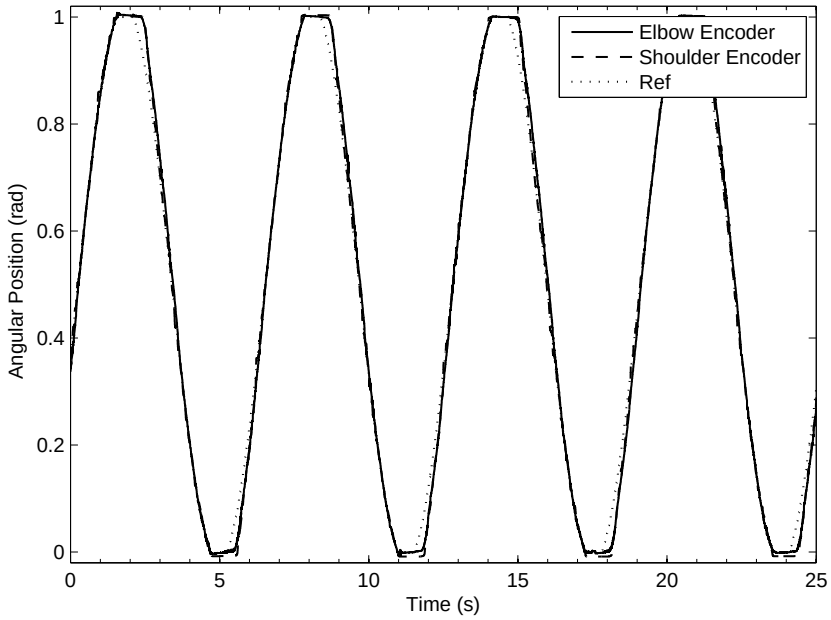


Figure 5.12. Reference position trajectory and angular positions of the shoulder link and elbow link.

convergence whereas a higher value of the learning rate may result in oscillations.

The reference angular positions and the angular positions of links which are measured by encoders with 153 600 counts per actuator revolution are shown in Figure 5.12. The reference signal with a period of 10 s can have values within the interval $[0, 1]$ rad. The highest amount of deviations from the reference trajectory occur on the edges of the reference signal. It can be inferred that the FSNN controller is capable of adapting its weights and membership function parameters to regulate the system for changing set point values.

The state tracking errors are depicted in Figure 5.13. The maximum value of the error signal is less than 0.08 rad, which supports the idea that the proposed algorithm is capable of accurate state tracking. The torque signals which are generated by FSNN can be seen in Figure 5.14. As these control signals are below their corresponding torque limits, they are directly applied to the manipulator links without requiring a saturation operation.

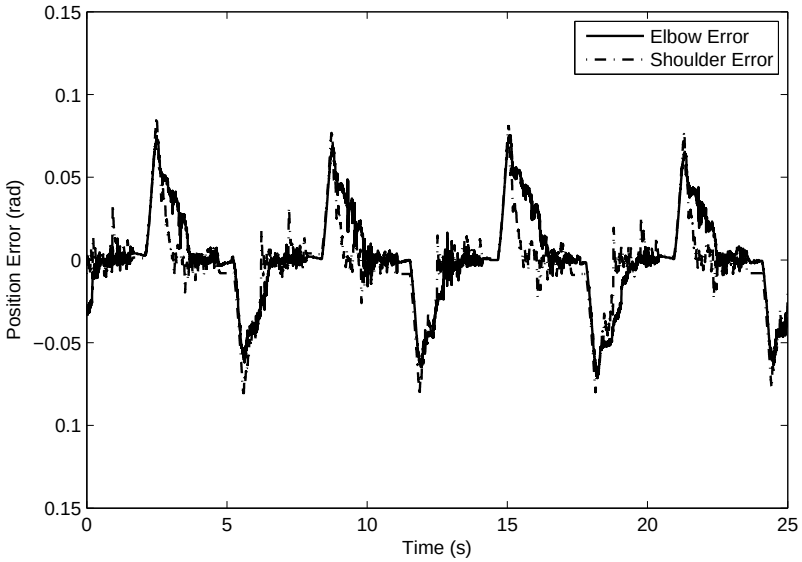


Figure 5.13. Tracking errors for the shoulder and elbow links.

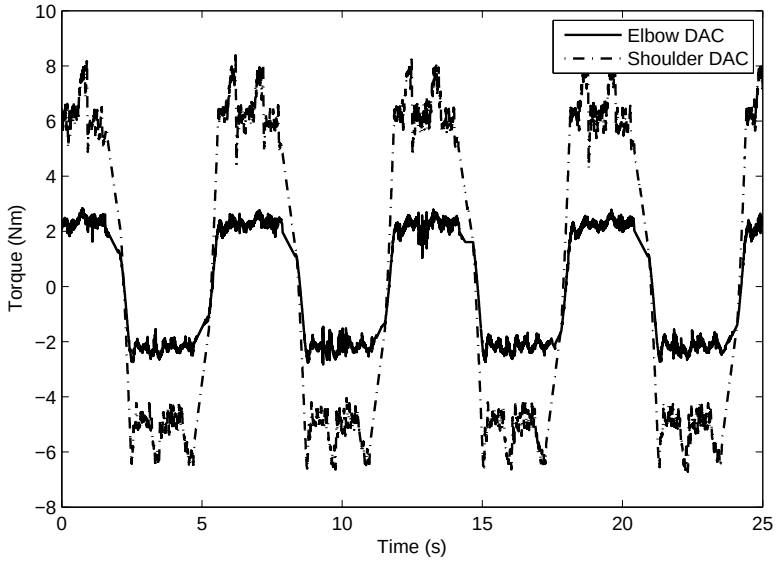


Figure 5.14. Applied torque inputs.

The variations in the mean and variance of the membership functions to generate the desired torque values can be observed in Figure 5.15 and Figure 5.16. Regarding the membership functions for the position error of the elbow link, the highest amount of variation has occurred in the mean and variance of the membership function Zero, whereas there is no significant change in the mean values for the membership functions Negative and Positive. For the shoulder link, the mean of the membership function

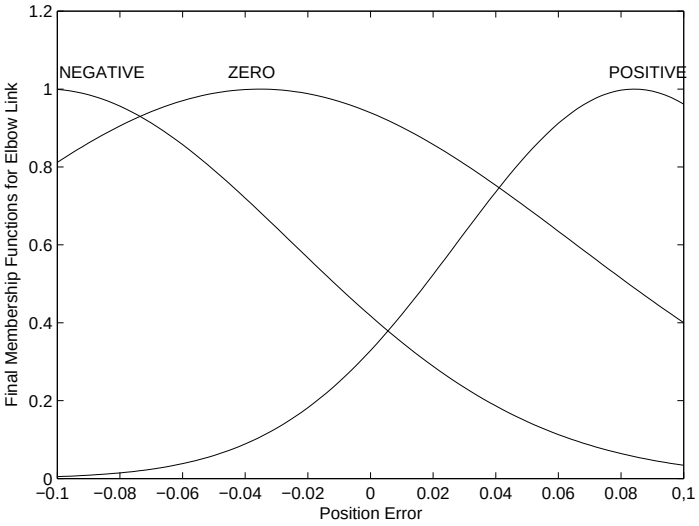


Figure 5.15. Final membership functions for the position error of the elbow link.

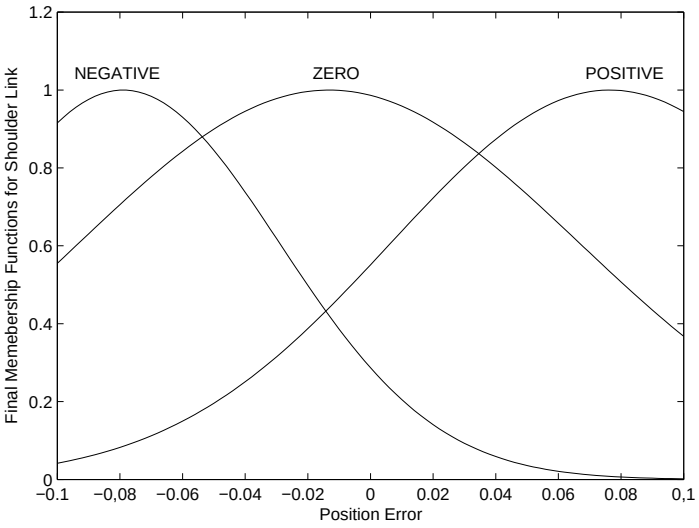


Figure 5.16. Final membership functions for the position error of the shoulder link.

Zero remains almost at the same place, whereas an increase in the variance value can be observed.

In order to make an in-depth comparison of the effectiveness of the gradient-based and SMC-based learning algorithms developed for FSNNs, the parameter update rules for the former have also been derived assuming linear activation function in the output layer. The resulting angular positions of the links are presented in Figure 5.17, which

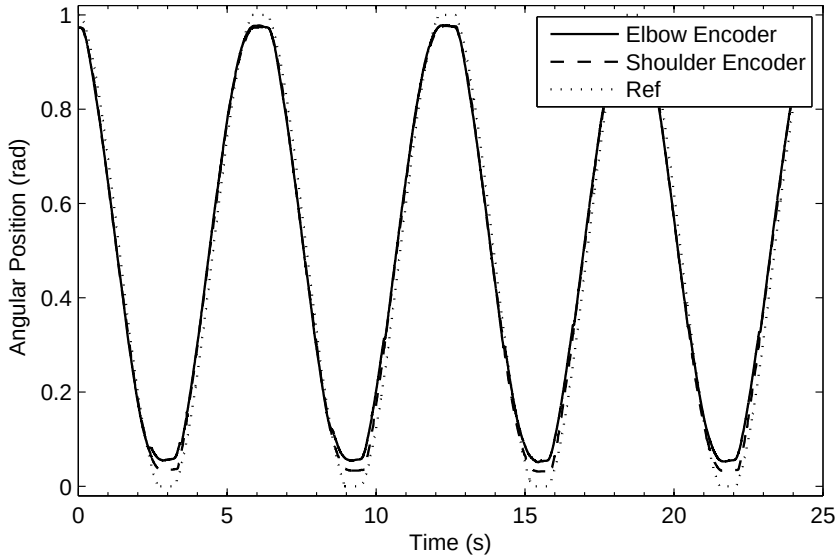


Figure 5.17. Reference position trajectory and angular positions of the links for FSNN with gradient-based learning algorithm.

implies that the proposed gradient descent-based algorithm fails to achieve zero steady-state error.

5.3. Analysis and Discussion

In this chapter, the development of a FSNN with a novel variable structure systems-based learning algorithm has been considered.The parameter update rules have been derived based on the Lyapunov stability method to achieve the stable online tuning of the neurocontroller parameters. The structure of the network is modified by replacing the spiking neurons in the output layer with the neurons having a linear activation function. Thus, the need for decoding has been eliminated resulting in a reduction in the computational time.

The proposed algorithm has been tested both on ABS for wheel slip regulation and on a two-link robot manipulator for angular position control of joints. The simulation studies on the ABS demonstrate that the FSNN controller can yield better transient and steady state characteristics than the SNN structure for constant reference value.

The same tests are repeated for the velocity dependent reference value and FSNN shows less deviations from the reference value.

In the second part, the control of the joint angular positions of a two-dof direct drive manipulator has been considered. Because of the highly nonlinear and coupled dynamics involved, the trajectory control of robotic manipulators is usually considered a challenging engineering problem. The results indicate that the FSNNs with a SMC-based learning algorithm can yield better transient and steady-state characteristics compared to the gradient-based algorithms.

6. CONCLUSION

In this dissertation, the problem of supervised learning in spiking neural networks has been considered. A novel learning algorithm based on the sliding mode control approach has been introduced and tested on different applications. The motivation behind the proposed learning scheme was to develop a feasible algorithm that can be applied to real tasks in real time applications. Therefore, in the derivation of the learning algorithms applicability, computational cost and effectiveness have been taken into account.

The vast majority of the supervised learning algorithms developed for SNNs relies on the gradient descent principle. Although studies using this approach show promising results, these kinds of learning algorithms inherently suffer from the convergence problem. The convergence properties of the algorithm have been generally defined for a specific set of network parameters. This is much more evident in the case of SNNs compared to traditional second generation networks, because the literature on these systems are comparatively new emerging and rule of thumb is not available yet. Under these circumstances, deviations in the system states due to the disturbances and uncertainties may have significant adverse effects on the performance.

The studies that demonstrate the robustness of variable structure control have motivated the use of the sliding mode control approach in the training of ANNs. In this study, the development of a SNN with variable structure systems-based learning algorithm is considered for identification and control of dynamic plants. The parameter update rules have been derived based on the Lyapunov stability method to achieve the stable online tuning of the neurocontroller parameters. It is shown that the developed learning algorithms can achieve a successful mapping between the input and output spike trains and can process the spikes effectively. At each step of the simulation or real time experiments, the synaptic weights are updated incrementally based on the

desired output spike train (or desired output value) and the given input signal. The results of the applications on different systems stress the fast convergence and online processing abilities of the network.

The main shortcoming associated with the SNN applications for control problems is that these algorithms require high computational time because of the increased number of parameters to be adapted, which stems from the network structure in which multiple synapses are used for each connection between a presynaptic and postsynaptic neuron. To lessen the computational burden, the neurons in the output layer have been considered with a linear activation function. This way the need for decoding is eliminated and for the connections between the last hidden layer and output layer the use of multiple synapses with adjustable weights have been omitted.

The use of the Spike Response Model was another attempt to reduce the required computational time. As the spike response function utilized in this time-dependent model includes exponential terms, its derivative with respect to the firing times of the presynaptic or postsynaptic neuron can be computed with ease and it will include terms that have been already computed.

This dissertation can be considered to consist of three parts. In the first part, the gradient descent-based learning algorithm and its application for identification and control tasks have been demonstrated. The identification studies are carried on the plant models from the literature. A single layered SNN structure with only two input neurons and a single output neuron has been used to gain a better insight into the computational power of the spiking neurons. Despite the small number of parameters, the small RMSE values indicate that the performance of the SNN is comparable to other approaches commonly used in the literature for similar applications. For the control problems, the network structure has been varied to have one more layer. The antilock braking system and DC motor system with nonlinear load conditions have been used as the test bed. The performance of the SNN controller has been compared with

a model-based PID controller. The results support the idea that model free approaches are preferable in dynamic systems in the presence of the uncertainties.

In the second part, SMC-based learning algorithms for one and two-layered SNNs have been developed. The proposed algorithm is tested on a laboratory servo system in real time for different load conditions including nonlinear and time-varying ones. The proposed algorithm has been also applied for the wheel slip regulation of an ABS. The results indicate that the SNNs with a VSS-based learning algorithm can exhibit a highly robust behavior against disturbances and sudden changes in the command signal and they can yield better transient and steady-state characteristics compared to a gradient-based learning method.

In the last part, the studies have been expanded to include fuzzy reasoning on the spiking neural network structure. New update rules based on the Lyapunov stability method to achieve the stable online tuning have been derived for the synaptic weights of the neural network and the parameters of the fuzzy membership functions. In this structure, sensor readings are converted into spike times using fuzzy membership functions. To achieve the desired output, not only the synaptic weights of the neural network have been altered, but also the mean and variance values of the membership functions have been adapted to facilitate the convergence characteristics. The performance of this approach has been compared with the performance of the SNN with the SMC-based learning algorithm and the FSNN with the gradient-based learning algorithm for different applications. The results indicate that the FSNN with the SMC-based learning algorithm can possess better transient and steady-state characteristics for both cases.

Inspite of the large volume of work published, spiking neural networks are still in their development stages and therefore is still a significant virgin area that is waiting to be explored by researchers. The studies presented in this dissertation can be further extended by relaxing the restrictive assumptions made in the derivation of the param-

eter update rules. One such improvement might be the development of new learning algorithms that can cope with neurons that fire multiple spikes such that full advantage of the capabilities of spiking neurons can be taken. Throughout this dissertation, to overcome the discontinuity of the membrane potential it is considered that each neuron can emit a single spike during a simulation cycle and the value of the membrane potential after this point has been neglected. In the case of multiple firing, some new approaches should be derived for the calculation of partial derivatives to overcome the discontinuous nature of the membrane potential.

REFERENCES

1. Gerstner, W. and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, Cambridge, 2002.

2. McCulloch, W. and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, Vol. 5, No. 1, pp. 115–133, 1943.

3. Werbos, P. J., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D.Thesis, Harvard University, 1974.

4. Ghosh-Dastidar, S. and H. Adeli, "Spiking Neural Networks", *International Journal of Neural Systems*, Vol. 19, No. 4, pp. 295–308, 2009.

5. Maass, W., "Fast Sigmoidal Networks via Spiking Neurons", *Neural Computation*, Vol. 9, No. 1, pp. 279–304, 1997.

6. VanRullen, R., R. Guyonneau and S. J. Thorpe, "Spike Times Make Sense", *Trends in Neurosciences*, Vol. 28, No. 1, pp. 1–4, 2005.

7. Bohte, S., J. Kok and J. L. Poutre, "Errorbackpropagation in Temporally Encoded Networks of Spiking Neurons", *Neurocomputing*, Vol. 48, No. 1, pp. 17–37, 2002.

8. Ghosh-Dastidara, S. and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection", *Integrated Computer-Aided Engineering*, Vol. 14, No. 1, pp. 187–212, 2007.

9. Belatreche, A., L. Maguire and M. McGinnity, "Advances in Design and Application of Spiking Neural Networks", *Soft Computing*, Vol. 11, No. 1, pp. 239–248, 2006.

10. Chandra, B. and K. V. N. Babu, "Classification of Gene Expression Data Using

Spiking Wavelet Radial Basis Neural Network", *Expert Systems with Applications*, Vol. 41, No. 4, pp. 1326–1330, 2013.

11. Kasabov, N., *Evolving Spiking Neural Networks and Neurogenetic Systems for Spatio-and Spectro-Temporal Data Modelling and Pattern Recognition*, Springer, Berlin, 2012.

12. Shin, J., D. Smith, W. Swiercz, K. S. K, J. Rickard, J. Montero, L. Kurgan and K. Cios, "Recognition of Partially Occluded and Rotated Images with a Network of Spiking Neurons", *IEEE Transactions on Neural Networks*, Vol. 21, No. 1, pp. 1697–1709, 2010.

13. Batllori, R., C. Laramee, W. Land and J. Schaffer, "Evolving Spiking Neural Networks for Robot Control", *Procedia Computer Science*, Vol. 6, No. 1, pp. 329–334, 2011.

14. Bouganis, A. and M. Shanahan, "Training a Spiking Neural Network to Control a 4-Dof Robotic Arm Based on Spike Timing-Dependent Plasticity", *Proceedings of The 2010 International Joint Conference on Neural Networks*, pp. 1–8, 2010.

15. Abiyev, R., O.Kaynak and Y. Oniz, "Supervised Learning with Spiking Neural Networks", *Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1030–1035, 2012.

16. Gamez, D., A. K. Fidjeland and E. Lazdins, "iSpike: A Spiking Neural Interface for the iCub Robot", *Bioinspiration & Biomimetics*, Vol. 7, No. 2, p. 025008, 2012.

17. Kasinski, A. and F. Ponulak, "Comparison of Supervised Learning Methods for Spike Time Coding in Spiking Neural Networks", *International Journal of Applied Mathematics and Computer Science*, Vol. 16, No. 1, pp. 101–113, 2006.

18. Bohte, S., *Spiking Neural Networks*, Ph.D.Thesis, Universiteit Leiden, 2003.

19. Rumelhart, D., G. Hinton and R. Williams, "Learning Representations by Back-Propagating Errors", *Nature*, Vol. 323, No. 1, pp. 533–536, 1986.

20. Xin, J. and M. Embrechts, "Supervised Learning with Spiking Neural Networks", *Proceedings of the International Joint Conference on Neural Networks*, pp. 1772–1777, 2001.

21. McKennoch, S., D. Liu and L. Bushnell, "Fast Modifications of the SpikeProp Algorithm", *Proceedings of the International Joint Conference on Neural Networks*, pp. 3970–3977, 2006.

22. Sporea, I. and A. Gruning, "Supervised Learning in Multilayer Spiking Neural Networks", *Neural Computation*, Vol. 25, No. 1, pp. 473–509, 2013.

23. Astrom, K. J. and B. Wittenmark, *Adaptive Control*, Addison-Wesley, Massachusetts, 1995.

24. Sadati, N. and R. Ghadami, "Adaptive Multi-Model Sliding Mode Control of Robotic Manipulators Using Soft Computing", *Neurocomputing*, Vol. 71, No. 13, pp. 2702–2710, 2008.

25. Topalov, A. and O. Kaynak, "Neural Network Modeling and Control of Cement Mills Using a Variable Structure Systems Theory Based Online Learning Mechanism", *Journal of Process Control*, Vol. 14, No. 1, pp. 581–589, 2004.

26. Kayacan, E., O. Cigdem and O. Kaynak, "Sliding Mode Control Approach for Online Learning as Applied to Type-2 Fuzzy Neural Networks and its Experimental Evaluation", *IEEE Transactions on Industrial Electronics*, Vol. 59, No. 1, pp. 3510–3520, 2012.

27. Ahmed, S., N. Shakev, A. Topalov, K. Shiev and O. Kaynak, "Sliding Mode Incremental Learning Algorithm for Interval Type-2 Takagi-Sugeno-Kang Fuzzy Neural

Networks", *Evolving Systems*, Vol. 3, No. 3, pp. 179–188, 2012.

28. Lian, R., "Design of an Enhanced Adaptive Self-Organizing Fuzzy Sliding-Mode Controller for Robotic Systems", *Expert Systems with Applications*, Vol. 39, No. 1, pp. 1545–1554, 2012.

29. Lin, D. and X. Wang, "Observer-Based Decentralized Fuzzy Neural Sliding Mode Control for Interconnected Unknown Chaotic Systems via Network Structure Adaptation", *Fuzzy Sets and Systems*, Vol. 161, No. 15, pp. 2066 – 2080, 2010.

30. Minsky, M. and P. Seymour, *Perceptrons*, MIT Press, Oxford, 1969.

31. Adrian, E. D. and Y. Zotterman, "The Impulses Produced by Sensory Nerve-Endings Part II. The Response of a Single End-Organ", *The Journal of Physiology*, Vol. 61, No. 2, pp. 151–171, 1926.

32. Rieke, F., *Spikes: Exploring the Neural Code*, MIT Press, Massachusetts, 1999.

33. Christopher deCharms, R., "Information Doding in the Cortex by Independent or Coordinated Populations", *Proceedings of the National Academy of Sciences*, Vol. 95, No. 26, pp. 15166–15168, 1998.

34. Izhikevich, E. M., "Simple Model of Spiking Neurons", *IEEE Transactions on Neural Networks*, Vol. 14, No. 6, pp. 1569–1572, 2003.

35. Wehr, M. and G. Laurent, "Odour Encoding by Temporal Sequences of Firing in Oscillating Neural Assemblies", *Nature*, Vol. 384, No. 6605, pp. 162–166, 1996.

36. Johansson, R. and I. Birznieks, "First Spikes in Ensembles of Human Tactile Afferents Code Complex Spatial Fingertip Events", *Nature Neuroscience*, Vol. 7, No. 2, pp. 170–177, 2004.

37. Ruf, B., *Computing and Learning with Spiking Neurons-Theory and Simulations*, Ph.D. Thesis, Technische Universitat Graz, 1998.

38. Wang, X., Z.-G. Hou, A. Zou, M. Tan and L. Cheng, "A Behavior Controller Based on Spiking Neural Networks for Mobile Robots", *Neurocomputing*, Vol. 71, No. 4, pp. 655 – 666, 2008.

39. Yu, Q., H. Tang, K. C. Tan and H. Li, "Rapid Feedforward Computation by Temporal Encoding and Learning with Spiking Neurons", *Neural Networks and Learning Systems*, Vol. 24, No. 10, pp. 1539–1552, 2013.

40. Hagras, H., A. Pounds-Cornish, M. Colley, V. Callaghan and G. Clarke, "Evolving Spiking Neural Network Controllers for Autonomous Robots", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, Vol. 5, pp. 4620–4626, 2004.

41. Fontaine, B., D. F. Goodman, V. Benichoux and R. Brette, "Brian Hears: Online Auditory Processing Using Vectorization over Channels", *Frontiers in Neuroinformatics*, Vol. 5, No. 1, pp. 1–9, 2011.

42. Natschlager, T. and B. Ruf, "Spatial and Temporal Pattern Analysis via Spiking Neurons", *Network: Computation in Neural Systems*, Vol. 9, No. 3, pp. 319–332, 1998.

43. Thorpe, S., A. Delorme and R. Van Rullen, "Spike-Based Strategies for Rapid Processing", *Neural Networks*, Vol. 14, No. 6, pp. 715–725, 2001.

44. Moore, S. C., *Back-Propagation in Spiking Neural Networks*, M.Sc. Thesis, University of Bath, 2002.

45. Schrauwen, B. and J. Van Campenhout, "BSA: A Fast and Accurate Spike Train Encoding Scheme", *Proceedings of the International Joint Conference on Neural*

Networks, Vol. 4, pp. 2825–2830, 2003.

46. Pavlidis, N., D. Tasoulis, V. P. Plagianakos, G. Nikiforidis and M. Vrahatis, "Spiking Neural Network Training Using Evolutionary Algorithms", *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 4, pp. 2190–2194, 2005.

47. Elman, J. L., "Finding Structure in Time", *Cognitive Science*, Vol. 14, No. 2, pp. 179–211, 1990.

48. Ltd., I., *The Laboratory Antilock Braking System Controlled from PC*, Tech. rep., Inteco, 2006.

49. Olsson, H., *Control Systems with Friction*, Ph.D. Thesis, Lund University, 1996.

50. Ko, C.-N., "WSVR-Based Fuzzy Neural Network with Annealing Robust Algorithm for System Identification", *Journal of the Franklin Institute*, Vol. 349, No. 1, pp. 1758 – 1780, 2012.

51. Chou, P.-H., C.-S. Chen and F.-J. Lin, "DSP-Based Synchronous Control of Dual Linear Motors via Sugeno Type Fuzzy Neural Network Compensator", *Journal of the Franklin Institute*, Vol. 349, No. 1, pp. 792 – 812, 2012.

52. Abiyev, R., O.Kaynak and E. Kayacan, "A Type-2 Fuzzy Wavelet Neural Network for System Identification and Control", *Journal of the Franklin Institute*, Vol. 350, No. 1, pp. 1658–1685, 2013.

53. McFall, K., "Automated Design Parameter Selection for Neural Networks Solving Coupled Partial Differential Equations with Discontinuities", *Journal of the Franklin Institute*, Vol. 350, No. 1, pp. 300 – 317, 2013.

54. Topalov, A. and O. Kaynak, "Online Learning in Adaptive Neurocontrol Schemes

with a Sliding Mode Algorithm", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 31, No. 1, pp. 445–450, 2001.

55. Topalov, A., K.-C. Kim, J.-H. Kim and B.-K. Lee, "Fast Genetic Online Learning Algorithm for Neural Network and its Application to Temperature Control", *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 649–654, 1996.

56. Niska, H., T. Hiltunen, A. Karppinen, J. Ruuskanen and M. Kolehmainen, "Evolving the Neural Network Model for Forecasting Air Pollution Time Series", *Engineering Applications of Artificial Intelligence*, Vol. 17, No. 2, pp. 159–167, 2004.

57. Chen, H. and Z. Zeng, "Deformation Prediction of Landslide Based on Improved Backpropagation Neural Network", *Cognitive Computation*, Vol. 5, No. 1, pp. 56–62, 2013.

58. Khorsand, A. R., T. Akbarzadeh and R. Mohammad, "Multi-Objective Meta Level Soft Computing-Based Evolutionary Structural Design", *Journal of the Franklin Institute*, Vol. 344, No. 5, pp. 595–612, 2007.

59. Howard, G., E. Gale, L. Bull, B. Costello and A. Adamatzky, "Evolution of Plastic Learning in Spiking Networks via Memristive Connections", *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 5, pp. 711–729, 2012.

60. Parma, G., B. Menezes and A. Braga, "Sliding Mode Algorithm for Training Multilayer Artificial Neural Networks", *Electronics Letters*, Vol. 34, No. 1, pp. 97–98, 1998.

61. Shuanghe, Y., Y. Xinghuo and M. Zhihong, "A Fuzzy Neural Network Approximator with Fast Terminal Sliding Mode and its Applications", *Fuzzy Sets and Systems*, Vol. 148, No. 1, pp. 469–486, 2004.

62. Chen, C. W., P. C. Chen and W. L. Chiang, "Modified Intelligent Genetic Algorithm-Based Adaptive Neural Network Control for Uncertain Structural Systems", *Journal of Vibration and Control*, Vol. 19, No. 9, pp. 1333–1347, 2013.

63. Yu, X. and O. Kaynak, "Sliding-Mode Control With Soft Computing: A Survey", *IEEE Transactions on Industrial Electronics*, Vol. 56, No. 1, pp. 3275–3285, 2009.

64. Kaynak, O., K. Erbatur and M. Ertugrul, "The Fusion of Computationally Intelligent Methodologies and Sliding-Mode Control - A Survey", *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 1, pp. 4–17, 2001.

65. Oniz, Y., E. Kayacan and O. Kaynak, "A Dynamic Method to Forecast the Wheel Slip for Antilock Braking System and Its Experimental Evaluation", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 39, No. 1, pp. 551 –560, 2009.

66. Slotine, J. J. E. and W. Li, *Applied Nonlinear Control*, Vol. 199, Prentice Hall, New Jersey, 1991.

67. Zak, S. H., *Systems and Control*, Vol. 388, Oxford University Press, New York, 2003.

68. Utkin, V. I., J. Guldner and J. Shi, *Sliding Mode Control in Electromechanical Systems*, Vol. 9, CRC Press, London, 1999.

69. Juang, C., "A TSK-Type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithm", *IEEE Transactions on Fuzzy Systems*, Vol. 10, No. 1, pp. 155–170, 2002.

70. Lin, C., "A Neural Fuzzy Control System with Structure and Parameter Learning", *Fuzzy Sets and Systems*, Vol. 70, No. 1, pp. 183–212, 1995.

71. *Direct Drive Manipulator Research and Development Package User Guide*, Integrated Motions Inc., Berkeley, CA, 1992.