INTELLIGENT STABILIZATION CONTROL OF TURRET SUBSYSTEMS UNDER DISTURBANCES FROM UNSTRUCTURED TERRAIN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZDEMİR GÜMÜŞAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

NOVEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that I have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Prof. Dr. İsmet ERKMEN Co-Supervisor	Prof. Dr. Aydan M. ERKMEN Supervisor

Examining Committee Members

Prof. Dr. İsmet ERKMEN	(METU, EE)	_____
Prof. Dr. Aydan M. ERKMEN	(METU, EE)	_____
Prof. Dr. Aydın ERSAK	(METU, EE)	_____
Prof. Dr. Tuna BALKAN	(METU, ME)	_____
İlhan BAŞÇUHADAR (M.S.)	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Özdemir GÜMÜŞAY

Signature:

ABSTRACT

INTELLIGENT STABILIZATION CONTROL OF TURRET SUBSYSTEMS UNDER DISTURBANCES FROM UNSTRUCTURED TERRAIN

Gümüşay, Özdemir
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Aydan M. ERKMEN
Co-Supervisor: Prof. Dr. İsmet ERKMEN

November 2006, 147 Pages

In this thesis, an intelligent controller for gun and/or sight stabilization of turret subsystems is developed using artificial neural networks. A classical proportional, integral and derivative (PID) controller equipped with a non-linear unbalance compensation algorithm is used as the low-level controller. The gains of this PID controller are tuned using a multilayered back-propagation neural network. These gains are modeled as a function of the error between the command and feedback signals and this model is generated by the function fitting property of neural networks as an estimate. The network is called as the "Neural PID Tuner" and it takes the current and previous errors as inputs and outputs the PID gains of the controller.

Columb friction is the most important non-linearity in turret subsystems that heavily lower the efficiency of the controller. Another multilayered back-propagation neural network is used in order to increase the performance of the PID controller by identifying and compensating this Columb friction. This network utilizes the error between the output of the PID controller driving the physical

system with Columb friction and the output of the identical PID controller driving a virtual equivalent linear system without Columb friction. The linear dynamics of the physical system is identified using a single layer linear neural network with pure linear activation function and the equivalent virtual linear system is emulated using this identification.

The proposed methods are applied to both computer simulations and hardware experimental setup. In addition, sensitivity and performance analysis are performed both by using the mathematical model and hardware experimental setup.

Keywords: Intelligent Control, Neural Networks, Neural PID Tuner, Neural Friction Compensation, Stabilization Control.

ÖZ

HERHANGİ BİR ARAZİDEN GELEN BOZUCU ETKİLER ALTINDAKİ TARET ALT SİSTEMLERİNİN AKILLI STABİLİZASYON DENETİMİ

GÜMÜŞAY, Özdemir

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Ana Bilim Dalı

Tez Yöneticisi: Prof. Dr. Aydan M. ERKMEN

Ortak Tez Yöneticisi: Prof. Dr. İsmet ERKMEN

Kasım 2006, 147 Sayfa

Bu çalışmada taret alt sistemlerinin silah ve/veya görüş stabilizasyonu için yapay sinir ağları kullanılarak akıllı bir denetleç tasarlanmıştır. Alt seviye denetimde doğrusal olmayan bozuk balansın da giderildiği klasik bir oransal, integral ve türev denetleci (PID) kullanılmıştır. Bu denetlecin kazançları çok katmanlı bir geri iletimli sinir ağı ile eniyilenmiştir. Bu denetleç parametreleri istek ve geri besleme sinyalleri arasındaki hatanın bir fonksiyonu olarak modellenmiş ve bu fonksiyon yapay sinir ağlarının fonksiyon yakınsama özelliği kullanılarak tahmin edilmiştir. Bu ağ "Sinirsel PID Eniyileyici" olarak adlandırılmıştır ve giriş olarak denetim hatasının şu anki ve geçmiş değerlerini almakta ve denetleç kazançlarını çıkış olarak vermektedir.

Kuru sürtünme taret alt sistemlerinde denetim performansını düşüren doğrusal olmayan en önemli etmendir. PID denetlecinin performansını artırmak için başka birçok katmanlı geri iletimli sinir ağı kuru sürtünmenin öğrenilmesi ve giderilmesi için kullanılmıştır. Bu ağ sürtünmeyi öğrenmek için kuru sürtünme etkisinde olan fiziksel sistemi süren PID denetlecinin çıkışı ile kuru sürtünme etkisinde olmayan ve fiziksel sistemin doğrusal bir eşleniği olan yapay sistemi süren PID denetlecinin

çıkışı arasındaki hatadan yararlanmaktadır. Fiziksel sistemin doğrusal davranışı ise tek katmanlı doğrusal bir sinir ağı ile tanılanmaktadır ve yapay doğrusal sistem bu doğrusal tanılama sonucunu kullanmaktadır.

Son olarak, önerilen metot bilgisayar simülasyonlarına ve deneysel bir prototipe uygulanmış ve sinir ağlarının hassasiyet ve performans analizleri yapılmıştır.

Anahtar Kelimeler: Akıllı Denetim, Yapay Sinir Ağları, Sinirsel PID İyileyici, Sinirsel Sürtünme Giderici, Stabilizasyon Denetimi.

To My Wife…

ACKNOWLEDGEMENTS

I express my sincere appreciation to my thesis supervisor Prof. Dr. Aydan ERKMEN for her supervision, support and helpful critics throughout the progress of my thesis study.

I would like to thank to Mustafa Burak GÜRCAN, Ömer GÖKSU and Serkan GÜVEY for their grateful support and for their valuable commands.

The cooperation and friendly support of my colleagues in ASELSAN during my thesis study also deserves to be acknowledged.

Finally, many thanks to my wife for her endless support and for her immortal love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS & ABBREVIATIONS

η : Learning rate

α : Momentum term

PID : Proportional, integral and derivative

PI : Proportional and Integral

PIID : Proportional, double integral and derivative

APG : Aberdeen Proving Ground

BP : Back-propagation

NN : Neural network

ANN : Artificial neural network

u : Controller output

y : System output

e : Controller error

LMS : Least Mean Square

E : Performance Index

J : Performance Index

DRNN : Diagonal Recurrent Neural Network

SISO : Single Input Single Output

DOF : Degrees of Freedom

MEMS : Micro Electromechanical Systems

CHAPTER 1

INTRODUCTION

1.1 Intelligent Stabilization Control – Motivation

Battle tanks are technologically the starting point of today's modern land weapons both mechanically and electronically. For instance, with the invention of missiles, people started to use the knowledge of turret subsystems of battle tanks to design missile launcher platforms. These platforms are placed on both stationary and moving structures and are used to adjust the aiming of the missiles towards a target.

Today, these kinds of turret subsystems have many high technology capabilities. For instance, day and night vision enables high firing accuracy at any whether conditions and for any time of a day. Another capability of today's turret subsystems is automatic target tracking which enables high firing accuracy for the moving targets. This capability is especially critical for turret subsystems developed for air targets because of the high speed and high maneuvering capabilities of air targets. The distance between the target and the turret subsystem can be measured using laser or by other optical methods and this distance information is utilized in the ballistic calculations. Meteorically data can also be measured in today's turret subsystems and these are used again for the ballistic calculations with the help of digital computers. Guided missiles or trajectories are also launched from today's turret subsystems. One of the most important capabilities of today's turret subsystems is the fire on the move, which enables very high firing accuracy while moving on a rough terrain [2].

As mentioned before, one of the most important and the most critical capability needed to be acquired is the “fire on the move” ability of turret subsystems. This capability is obtained basically by gun stabilization, which means holding the orientation of the gun stationary relative to a reference on the ground even under random disturbances generated by the vehicle moving on an unconstrained terrain [3].

Previously, gun stabilization was realized based upon mechanical gyroscopes and analog/digital computers [4]. Up to now and even maybe in the future, the controller structure of these systems has been basically proportional + integral + derivative (PID) type classical controllers [3]. Although these kinds of classical controllers have many advantages; they cannot satisfy high performance criteria for complicated control tasks.

One of the main advantages of these classical controllers is their simplicity. They are simple to understand and well known. Many researches are conducted elaborately on these techniques, which have always been mathematically well stated. Another advantage is their ease of implementation. In other words, these techniques do not require high computing power because of their numeric simplicity. Besides, the parameters of these controllers are easy to tune and many tuning methodologies can be found in the literature. Most of the parameters of these controllers have a physical meaning and therefore the designer can predict the controller sensitivity to parameter changes [5].

However, these control techniques have a very critical primordial disadvantage. The problem is that these control strategies are not adaptive and robust against changes in the operation environment. In order to solve these problems, gain scheduling or switching type controllers are implemented using classical controllers. For instance, in order to solve the stiction problem caused by Columb friction, high controller gains are used or a high frequency signal is added to the output of the controller in low speed regions. However, both of these methods require high actuation power and high controller bandwidths [1]. Tuning these controllers have permitted to locally enhance performance thus may easily be temporary. For instance, a PID controller with a non-linear friction compensator

tuned in Ankara in the summer may not satisfy performance criterions at Erzurum in the winter. Like friction, there may be other uncertainties and non-linearities in the controlled system [49]. Because of the nature of uncertainty, a classical controller cannot compensate for these uncertainties without an estimator.

Today, most of the battle tanks, used all over the world, are designed for fighting outside urban areas. However, starting with the wars in the Iraq and Bosnia, the battlefield changed from rural to urban areas [6]. This change from rural to urban thus changed the terrain from earth to concrete, which totally transformed the disturbance characteristics in their frequency and amplitudes. Therefore, the controllers tuned for stabilization on sand terrain are not any more suitable for stabilization on concrete terrain.

As a summary, the control tasks needed for the aiming and stabilization of turret subsystems are very hard due to the challenges of today's high-tech battlefield. The uncertainties and non-linearities of the controlled system and changes in the operational environment also increase the complexity of the control task. Therefore, the main motivation of this thesis is to develop a controller for the stabilization control of turret subsystems, which will handle these changes in the operational conditions (i.e. terrain and environmental changes) and in the system dynamics in order to minimize the control error under random disturbances from the rough terrain.

1.2 Objective and Goals of the Study

The stabilization control task of turret subsystems is complicated due to the uncertainties and non-linearities of the controlled system and due to changes of the operation environment. The types of the terrain and meteorogical conditions affect the disturbance frequency and amplitude characteristic. In addition, the dynamics of the system changes with changing environmental conditions. Such as, the friction of the system changes with changing outside temperature.

The objective of this study is to design an intelligent controller, which will handle these unknown changes both in the controlled system and in the environment in order to have almost the same control performance under the influence of

disturbances from random terrain. Towards this objective, the goals of the proposed method are given below.

The control task is named as “Gun Stabilization Control” and it requires the minimization of the transient and steady state control errors between the user command and the feedback signals. Due to the unknown changes of the environment and the controlled system, adaptation of this controller to the new environment is needed, which requires learning of the new environment.

In addition, identification and compensation of system non-linearities will help the stabilization controller to increase the stabilization performance. Two main non-linearities namely the unbalance and the Column friction affects the controller performance a lot and compensation algorithms will be used for these non-linearities. These algorithms will also need learning of the new environment and identification of the controlled system dynamics.

Since the proposed controller is designed for turret subsystems, which are controlled with digital computers, the last and may be the most important point will be the real-time implementation of the proposed controller on these digital computers. Therefore, the controller will be mathematically simple and will require minimum memory and hardware.

1.3 Methodology

In order to perform the stabilization control task, a classical PID controller is used in the low-level controller because of its simplicity. Using this PID controller, the error between the speed command from the user and the feedback signal from a speed feedback device is minimized by producing a torque signal for the actuation system. This PID controller is equipped with an unbalance compensator, which utilizes a g-sensitive accelerometer placed in the center of rotation (COR) of the corresponding axis.

In order to handle the uncertainties in the controlled system and in the environmental conditions, the control gains of this PID controller is adjusted by a multilayered artificial neural network. This network is trained with back-

propagation learning algorithm in order to minimize the control error of the PID controller. The parameters of this network such as the number of inputs and number of hidden layers are optimized by performing computer simulations on the mathematical model of a battle tank.

The primary non-linearity of the system namely the Columb friction is identified and compensated in order to increase the controller performance. Again, a multilayered artificial neural network is used for identification of the Columb friction. This network is trained using back-propagation learning algorithm and it aims to minimize the error between the output of the PID controller driving the physical system and the output of the identical PID controller driving a virtual linear system, which is the linear estimate of the physical system. This linear system estimate is obtained by identifying the dynamics of the physical system with a single layer linear neural network in regions where the effect of Columb friction is minimized.

The sensitivity of these networks to the sensor noise and quantization errors in the D/A and A/D conversions are analyzed and the performance of the controller is examined with different terrain conditions, changing system dynamics and changing user commands.

1.4 Contributions of the Thesis

In this study, an intelligent stabilization controller is developed for turret subsystems. This proposed controller is implemented on the mathematical model of a battle tank and results are given for different operational conditions.

A unique Columb friction identification method is developed using two neural networks. The linear equivalent of the physical system is identified using a linear network and another multi-layered network utilizes this linear system estimate in order to identify the Columb friction characteristics of the system. Using this estimate, Columb friction is simply compensated by adding a correction signal to the output of the stabilization controller, which is the inverse of the estimated friction.

A hardware prototype is developed for the implementation of the proposed method. A scaled model of Challenger 2 tank is modified mechanically in order to represent real life applications. Feedback sensors and actuation electronics are added to the model. In addition to these, a sight camera for the visualization of stabilization performance and a wireless router for communication are added to the hardware prototype. The proposed controller is implemented using this prototype and the results are given in this report.

1.5 Outline of the Thesis

In the first chapter of this thesis, the motivation of this study, goals of the study and contributions to the literature is given. In the second chapter, a literature survey about stabilization and intelligent control techniques used in the study are represented. The third chapter is devoted to the proposed method giving the details of the proposed neural controller architecture. The simulation environment and the experimental setup are explained in detail in the fourth chapter. The simulation results on the mathematical model of a main battle tank and the test results on the experimental setup are discussed in the fifth chapter. The sensitivity analysis of the neural linear system identifier network and the performance analysis of the proposed neural controller are given in the sixth chapter. Finally, a conclusion is made and future works are stated in the seventh chapter.

CHAPTER 2

LITERATURE SURVEY

2.1 Battle Tanks and Turret Subsystems

Beginning of 1900s shows the rise of turret subsystems. People wanted to protect themselves from enemy attacks while firing. To this end, they kept themselves behind armors and fired through a hole on this armor while tracking the enemy from another hole. The gun was sometimes heavy cannon or sometimes a simple machine gun.

With time, the idea expanded to mobile armored gun structures. The pioneer was the German Army that introduced a tracked vehicle equipped, on top, with an armored stationary turret subsystem. British tanks named Mark I, II, III and IV were subsequently designed using the concept of this vehicle (Figure 2.1). After this one, larger vehicles were designed with many rotating or stationary turret subsystems. Guns on these vehicles were placed either at the sides or at the back or also at the front [7].

Figure 2.1 – Medium Mark B [7]

Turret subsystems were also used in fighting airplanes, especially in large airplanes, where there was more then one turret. In World War II (WWII), there were one turret on the tail and one on the tip of American airplanes [8].

Today, turret subsystems are used for many different purposes besides being still used on battle tanks. Turret subsystems can be found used in gunner and commander periscopes, commander's auxiliary machine gun and for active protection purposes on today's main battle tanks [9].

Turret subsystems can be found as launchers for missiles, platforms for electro-optical devices such as thermal imaging cameras and laser range finders, seeker heads of guided missiles, platforms for many different calibers of guns such as 0.50" caliber M2HB machine gun and 76mm caliber cannon for battle cruisers.

A mechanical structure with electrical or hydraulic actuation, which consists of at least one movable axis, which is independently actuated, best defines a turret subsystem [9]. These subsystems usually carry one or more guns with or without optical devices and/or other detection and/or destruction equipments. Most of the time, turret subsystems are used to orient a gun or an electro-optical device to a target and therefore two axes (traverse and elevation) are usually enough to perform this task.

Today's modern battle tanks have still a two axis turret in order to aim rotation axes usually named as traverse and elevation (sometimes called azimuth) and both of these axes are gyro stabilized. Different from the previous battle tanks, today there are at least two turret subsystems on top of the main gun turret. These turret subsystems are gunner and commander periscopes, which usually use inverted mirrors for vision. In addition to these, sometimes a stabilized turret is used to aim and stabilize the machine gun of the commander.

ASELSAN is one of the World leading companies dealing with these turret subsystems. Starting with the Turkish Pedestal Mounted Stinger Platform (TPMSP) project, which uses stinger missiles as main gun and 12.7mm machine gun as auxiliary gun, ASELSAN signed many contracts with Turkish Army and

foreign countries for systems, which have stabilized turret subsystems. Some of these systems are VOLKAN project for modernization of Leopard I tanks (Figure 2.2), Netherlands Pedestal Mounted Stinger Platform project (Figure 2.3) and Stabilized Machine Gun Platform (STAMP) for Turkish Navy. All of the turret subsystems used in these projects have stabilization capability with many other high technology capabilities such as day and night vision, automatic target tracking, target distance measurements and guided missiles [9].

Figure 2.2 – Leopard 1A1 Tank modernized by ASELSAN

Figure 2.3 – Pedestal Mounted Stinger System for Netherlands Army designed and produced by ASELSAN

2.2 Gun Stabilization

When developed in WWI, battle tanks were noisy, dangerous and difficult to operate. It was really a hard job to aim the gun to the target with certain accuracy. For a tank standing still, aiming the gun is relatively easy however, at stand still all points on a large size tank are open targets for the enemy and it is hard to camouflage such a big vehicle.

Thus, reducing the body target area achieved somewhat by keeping tanks in motion on the battlefield. In the primary versions of this idea, moving to a firing place and standing still here for a short enough time just for shooting was popular. After WWII, fighting vehicles such as British Centurion Mk 5 [11] were equipped with gun stabilization capability using gyroscopes for stabilization. This was an innovation for battle tanks because fixing the gun orientation towards the target became very easy. Actually, this was the first generation of gun stabilization systems.

With this very early version of gun stabilization, orientation of the main gun in both elevation and traverse axis was held stationary relative to a reference on the ground. The error between the target point and the aimed point became independent of the movements of the vehicle. Only translational movements whose effect is minor were left and corrected by the gunner.

The first generation gun stabilization is explained schematically in Figure 2.4 and Figure 2.5. In the first figure, the orientation of the gun is kept horizontal relative to the earth. In the upper view, the deviation from the target was only translational which is independent of the target distance. However, in the lower view, the orientation of the gun is kept steady relative to the vehicle and both angular and translational errors occurred. The rotational error results in very large aiming error, which increases with increasing target distance. The situation for the traverse axis is given in the second figure. For both of the figures, the errors obtained with the stabilized gun are only due to the translational movements of the vehicle and very low compared to the errors resulted in the non-stabilized gun case.

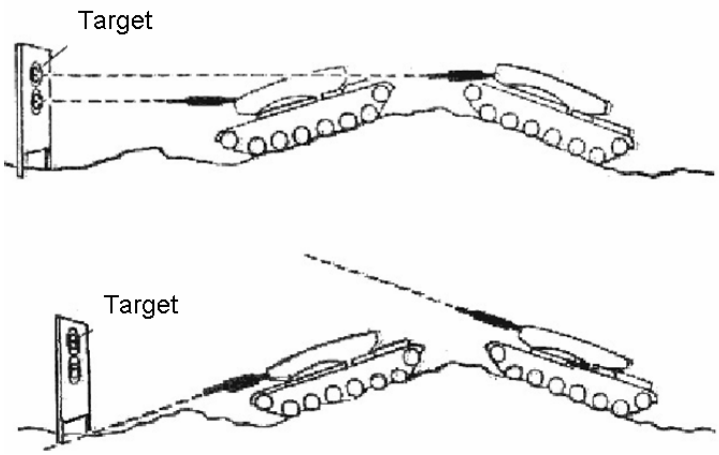

Figure 2.4 – Demonstration of gun stabilization on elevation axis

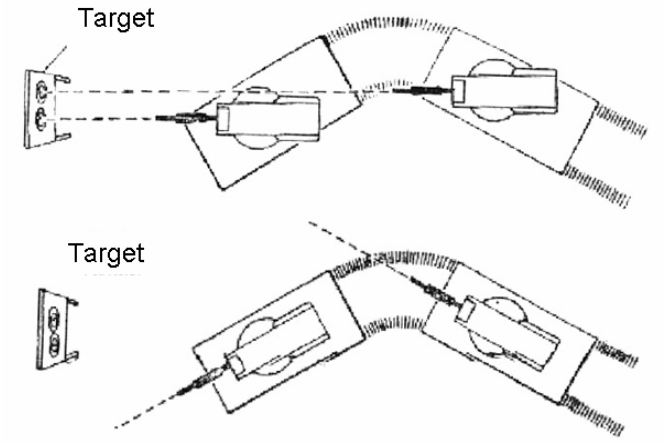

Figure 2.5 – Demonstration of gun stabilization on traverse axis

Although the first generation gun stabilization changed the role of the first battle tanks on battlefield, the performance was not enough resulting in low first round hit probability [12]. Before the invention of gun stabilization, the battle tanks were mobile weapons but capable of firing while standing still but after gun stabilization was introduced, the battle tanks became the main weapon of the battlefield. In the first generation gun stabilization systems, a rate gyroscope was used for each separate axis. These gyroscopes were used to sense the disturbances on the corresponding axis and this information was used to hold that axis stationary during the motion of the vehicle. The control structures of the first generation stabilization systems was using rate errors between the gyro feedback

and the command signal coming from the gunner's handle, therefore there was a need for error in order to apply an action (this is so called error based control).

For high first round hit probabilities, tolerance for error is usually very tight. Because of this requirement, second generation stabilization systems are introduced where disturbance gyroscopes measure disturbance rates on the moving vehicle platform [13]. This disturbance information is subsequently used to estimate turret motion as the reverse of the hull motion, and this estimated motion is applied to the control system in order to have the inverse motion of the disturbance. This technique is called disturbance feed-forward and has been found to increase the stabilization performance two ore three times more.

In the second-generation stabilization systems, the gunner uses a stabilized periscope in order to aim the gun to the target. In these systems, the control system of the turret also tracks the orientation of this gunner periscope [12]. Therefore, the control task is not only stabilization of the turret but at the same time tracking the orientation of the periscope with acceptable performance.

In today's stabilization systems, holding the turret orientation stationary on the target is not enough because most of today's targets do poses also "fire on the move" capability. The time for aiming and shooting has been greatly reduced [14]; therefore, there is a need for target tracking and fast reaction. Stabilization systems having target tracking capability are called third generation stabilization systems. The video trackers of gunner periscopes detect the target and bore-sight errors are used to aim the gun to the target. In this generation, unbalance compensation using accelerometers are used in addition to disturbance feed-forward technique because most of the turret subsystems using third generation stabilization systems have huge guns (120mm and 140mm guns for main battle tanks) resulting in high mechanical unbalances. This huge unbalance is compensated most of the system in order to increase the stabilization performance. For this compensation, an accelerometer which can measure both the gravity and the accelerations resulted in the vehicle motion is used. The same method is utilized in this study in order to compensate the non-linear unbalance effects. The detail of this method is given in chapter 3.

In today's modern fighting vehicles with turret subsystems, the general trend is to use sensors in order to minimize the stabilization, aiming errors and maximize the firing accuracy. In order to realize these goals, ballistic calculations are made using the initial speed of the trajectory, speed and the direction of the wind and target distance. Using these values, corrections are produced to the gun position both in elevation and in traverse axis.

However, there are many parameters which are not measurable but affecting the performance of the firing accuracy such as Columb friction, inertia changing with ammunition amount and payload, and disturbance characteristics. All of these unknown changes need information about the controlled system and the operation environment. However, these points are still not covered in today's modern controllers for turret subsystems.

In most of today's turret subsystems, the stabilization controllers are mainly based on classical PID controllers. The structure of the controller changes depending on the system and becomes PI controller neglecting the derivative term or PIID controller using double integrator. Other than the PID controller, lead compensators, lag compensators or lead-lag compensators are used in the controllers. In addition, low-pass, high-pass or notch filters are used to eliminate the sensor noise and avoid the mechanical resonances.

To improve the performance of the classical controller and to make it adaptive to the changes, gain scheduling or switching type controllers are also implemented for turret subsystems. In addition to these control techniques, modern control techniques like adaptive control or robust control are experimented on turret subsystems [15]. However, because of the complexity of the controlled system and uncertainties of the environment, none of these controllers handles the unknown changes both in the environment and in the physical system.

The effect of these unknown changes should only be minimized by learning the new environment and estimating the dynamics of the controlled system. That is why an intelligent controller is developed in this study to handle the uncertainties

and non-linearities of the system. The methods developed in order to estimate and compensate for one of these non-linearities is given in the chapter 3.

2.3 Stabilization Controllers

Up to this point, the stabilization concept is used only for turret subsystems but there are other application areas of stabilization concept and these are the thing that we are familiar in our daily lives. The applications are mainly in consumer electronics or in automotive. There are also other applications in aerospace and chemistry. The process stabilization in chemical plant is out of the scope of this study therefore will not be discussed.

The stabilization controllers are widely used in digital cameras in order to have high quality pictures and videos. Due to the disturbances coming from the users hand or from the platform on which the camera is fixed, the image quality decreases with the increasing disturbance amplitude. In order to eliminate the effect of these disturbances and to get sharp pictures, image stabilization is used.

There are two approaches used for the image stabilization. These are called as optical stabilization and digital image stabilization [50]. In optical stabilization approach, the lens is actuated with small motors and gyroscopes or accelerometers are used in order to sense the disturbances. The measured disturbance data is used to actuate the lens of the camera in the opposite direction of the disturbance [50].

The digital image stabilization is rather different from the optical stabilization. In this approach, there is no moving part for the image stabilization. The effect of the disturbance on the retrieved image is detected using special algorithms and image stabilization is performed by moving some part of the frame in the opposite direction of disturbance [50].

Optical image stabilization is very similar to the stabilization of turret subsystems conceptually but the controlled mechanical systems are completely different. The challenges discussed in first chapter for the stabilization control of turret subsystems are not problem for image stabilization. The mechanics of the lenses

are almost ideal which means almost no non-linearity and uncertainties. In addition, the performance specifications are completely different. One or two pixel errors are acceptable for image stabilization but almost ten times better stabilization requirements are usual for the stabilization of turret subsystems.

The stabilization controllers are also used for the stabilization of systems, which are naturally unstable. These systems can be motorcycles, unicycles and inverted pendulums. For instance, Dao and Lui [52] used a simple proportional controller for the stabilization of a unicycle but the proportional gain is scheduled using the non-linear mathematical model of the system.

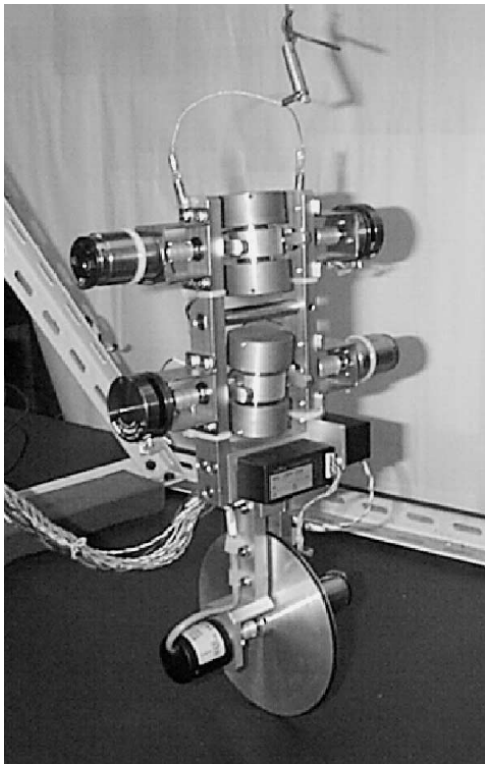

Figure 2.6 – Unicycle Robot in Motion

Similarly, Yi, Song, Levandowski and Jayasuriya [51] used a robust sliding mode controller for the stabilization of a motorcycle. A detailed mathematical model of the system is used in this study for the stabilization control of the system.

As mentioned before, stabilization controllers are also used in automobiles especially for safe drive under rainy or snowy whether conditions. Anti-skid

control is the most important property of today's automobiles and it is known as "Electronic Brake force Distribution (EBD) system". For this anti-skid control, accelerometers are utilized in order to sense the centrifugal forces on the vehicle especially while making turns. This measured acceleration data is used in order to calculate the brake forces for each wheel. For instance, while making a turn to the right, the brake force in the right wheels of the car is greater or less than the brake force in the left wheels depending on the angle of turn and road conditions [53].

Simple proportional controllers are widely used for the anti-skid control of today's commercial automobiles but there are some academic studies, which can be applied to the future's automobiles. One of these studies is performed by Fujimoto, Saito and Noguchi for the anti-skid control of an electrical vehicle on snowy conditions. They used observed based modern controllers for the stabilization of the electrical vehicle on the road [53].

Figure 2.7 – An Experimental Setup for Anti-Skid Control Studies

As a summary, stabilization controllers are widely used in today's technology and we have the benefit of stabilization even in our daily lives. Because of the performance requirements of the controller and environmental conditions of the plant under stabilization control, the challenges of turret stabilization is not valid for most of the other applications. In addition, there is limited computational power for most of the system needing stabilization control. Therefore, very simple stabilization controllers based on sensor measurement are used widely.

2.4 Friction Modeling and Compensation

The friction heavily affects the stabilization controller performance for turret subsystems. Turret subsystems experience frequently the non-linearity caused by friction while the motion is stabilized. For this reason, a literature survey is performed about friction models and compensation techniques.

Friction is a natural phenomenon, which affects almost all motion. It has been the subject of extensive study for centuries with the main objectives being the design of effective lubricating processes and the understanding of the mechanisms of wear. Whereas friction effects at moderate velocities are somewhat predictable, it is the effect of friction at low velocities, which is very difficult to model. The facts that;

i) friction changes sign with velocity

ii) asymmetric with respect to the velocity axis

iii) has evolutionary characteristics

iv) exhibits stick-slip phenomenon

etc., aggravates the problem. Although friction effects have been well understood qualitatively, researchers have often relied on experimental data to formulate various mathematical models [24].

From a control point of view, the most popular friction model has been the static + Coulomb friction model originally proposed by Leanardo da Vinci in 1519 [25]. Whereas this model is sufficient for most applications, it fails to capture the low-velocity effects such as the Stribeck effect, stick-slip motion, presliding displacement, etc., which play a significant role in high-precision position setpoint/tracking applications. To capture low-velocity friction effects, several researchers have proposed empirical models. For example, Tustin [26] proposed an exponential relationship between friction and velocity to fit the Stribeck curve. Hess and Soom [27] modeled the Stribeck curve using an inverse quadratic relationship between friction and velocity. In [28], Canudas de Wit et al.

approximated the exponential relationship proposed in [26] by a linearly parameterized model which fostered the design of an adaptive controller. Rabinowicz [29] integrated the stick-slip motion into a friction model, which takes into account the temporal phenomenons of dwell time and lag time. An alternative approach, known as the state-variable model, was proposed by researchers ([30],[31],[32]) to capture the effects of time delay associated with friction. Whereas the qualitative interpretations provided by researchers from the field of tribology led to a better understanding of the friction phenomenon, it is the control of low-velocity mechanical systems in the presence of friction, which attracted the attention of many researchers from the control community. Specifically, the fact that some of the proposed models can be utilized during control synthesis has resulted in several interesting papers, for example, in [33], Armstrong explored the implications of the Stribeck effect on feedback control using a reduced order model of friction. Armstrong also established experimental procedures to stabilize the performance of the controller at low velocities. In [28], Canudas de Wit et al. designed an adaptive controller for dc motor drives utilizing a friction model, which was asymmetric in angular velocity. In [34], Canudas de Wit addressed the problem of adaptive friction compensation for robots during low-velocity operations using a linear parameterizable model (i.e., the conventional exponential function which represents the Stribeck effect was replaced by a linear parameterizable model to facilitate the design of an adaptive feedforward controller). In [35], Armstrong applied dimensional and perturbation analysis to solve a nonlinear low-velocity friction control problem.

Almost in the half of the friction models and techniques for the compensation of the friction, a precise model of the physical system is used. In addition, in all of the compensation methods, a precise measurement of the relative speed is a must. For the methods without the system models, the friction is models are obtained using experimental data. Therefore, these models are valid only for the friction characteristics of the experimental conditions.

For turret subsystems, it is very hard to obtain a precise mathematical model. Besides, the relative velocity measurements are not so precise because of the

sensor noise. Performing experiments in order to model the friction is not a solution for changing friction characteristics.

In order to identify the friction characteristics of the physical systems, one approach is to estimate the dynamics of the physical system. The linear behavior of the physical system is identified in this study and therefore a literature survey is performed about system identification and virtual system emulation using system identification.

2.5 System Identification and System Emulation

The simulation of the system dynamics in order to mimic the system behaviors using system identification is called system emulation. There are plenty of system identification techniques in the literature. These techniques use the inputs to the system and the outputs of the system to estimate the system dynamics statistically.

Guojun [38] analyzed the complexity and undecidability in the modeling of speed control system of hydraulic elevator. The back-propagation neural networks model is used in this study and it is formed by the system identification.

Ping and Lihua [39] worked on the Diagonal Recurrent Neural Networks (DRNN) in order to examine the performance of DRNN with the dynamical system identification. They used these neural networks to identify the hydraulic servo system dynamical performance.

Purwar, Kar and Jha [40] worked on the computationally efficient artificial neural network (ANN) model for system identification of unknown dynamic nonlinear continuous and discrete time systems. A single layer functional link ANN is used for the model where the need of hidden layer is eliminated by expanding the input pattern by Chebyshev polynomials. These models are linear in their parameters. The recursive least squares method with forgetting factor is used as on-line learning algorithm for parameter updating in their work. The good behavior of the identification method is tested on two single input single output (SISO) continuous time plants and two discrete time plants. Stability of the identification scheme is also addressed in this study.

Constant, Lagarrigues, Dagues, Rivals and Personnaz [41] presented a new model of an induction machine based on neural network theory. They introduced a neural model architecture, which is based on the Park model. They then describe the training procedure of the neural model, and give evaluations of its performance, for example on startups with a speed vector control.

Jovanovic [42] presented a neural network approach for structural dynamic model identification. The neural network is trained and tested by using the responses recorded in a real frame during earthquakes. The results they obtained show the great potential of using neural networks in structural dynamic model identification.

As a result of the survey, it is obvious those neural networks are widely used for the identification of dynamic systems. Back-propagation learning is the mostly used technique for the training of these neural networks.

Before proceeding with the proposed controller architecture, a mathematical background is given covering the concepts about the intelligent control, artificial neural networks, back-propagation learning algorithm and least mean squares learning algorithm for the sake of completeness.

2.6 Mathematical Background

Mathematical details of learning techniques used in this study are stated in this section. In addition, general information about intelligent control and artificial neural networks covering the importance, needs and principles are given.

2.6.1 Intelligent Control

An intelligent system has the capability to act appropriately in an uncertain and unknown environment, where an appropriate action is the one that increase probability of success with success is the achievement of behavioral sub goals that support the system's ultimate goal [20].

An intelligent system ([16],[17],[18]) is based upon capabilities such as perception, learning, reasoning, and making inferences or decisions from incomplete and uncertain information. This requires a knowledge system in which

the representation and processing of knowledge are main functions. For instance, approximation is a “soft” concept, and the capability to approximate for the purposes of comparison, pattern recognition, reasoning, and decision-making is the challenge of intelligence. Soft computing is an important branch of computational intelligence where fuzzy logic, probability theory, neural networks, and genetic algorithms are used synergistically to mimic reasoning and decision making of a human [19].

Adaptation and Learning

The ability to adapt to changing conditions is a necessity in intelligent systems. Although adaptation does not necessarily require the ability to learn, it is almost impossible for systems to be able to adapt to a wide variety of unexpected changes without learning. Therefore, the capability called learning is an important aspect of (highly) intelligent systems. Learning is important for the stabilization control of turret subsystems. As stated many times before, there are always changes in the operation environment and the controller should adopt itself to this new environment in order to satisfy the controller goal for all operation conditions.

Autonomy and Intelligence

Autonomy in setting and achieving goals is an important aspect of intelligent control systems. When a system has the capability to act appropriately in an uncertain and unknown environment for long periods of time without external intervention it is said to be highly autonomous. There are degrees of autonomy; an adaptive control system can be expected as a system of higher autonomy than a control system with fixed controllers like classical PID controller, as it can cope with greater uncertainty than a fixed feedback controller can. Although for low autonomy no intelligence is necessary, for high degrees of autonomy, intelligence in the system is a must. For turret subsystems, the autonomy is very important. Many of the control problems are solved automatically without the action of the user. This is very logical for such fighting machines because a minor error may result in the loss of the crew in the turret subsystem.

Structures and Hierarchies

In order to cope with complexity of the controller structure, an intelligent system must have an well-structured functional architecture for efficient analysis and evaluation of control strategies. This structure should be "sparse" and it should provide a mechanism to build levels of abstraction (resolution, granularity) or at least some form of partial ordering so to reduce complexity. An approach to study intelligent machines involving entropy emphasizes such efficient computational structures. Hierarchies (that may be approximate, localized or combined in hierarchies) that are able to adapt, may serve as primary vehicles for such structures to cope with complexity. The term "hierarchies" refers to functional hierarchies, or hierarchies of range and resolution along spatial or temporal dimensions, and it does not necessarily imply hierarchical hardware. Some of these structures may be hardwired in part. To cope with changing circumstances the ability to learn is essential so these structures can adapt to significant, unanticipated changes [20].

This concept is important for an intelligent controller while implementing to the turret subsystems. The computational power is very limited in turret subsystems because of the limited space and limited power. In addition, the simplicity and the ease of understanding of the controller are the unavoidable needed for a stabilization controller for turret subsystems.

For these reasons, simple network structures, which are dedicated only for one purpose, are used and worked in parallel in the controller architecture proposed in this study. The controller divided into two parts one for the elevation and one for the traverse axis and distributed to two layers namely the low-level and high-level controllers.

2.6.2 Artificial Neural Networks

Neural networks are adaptive statistical models based on an analogy with the structure of the human brain. They are adaptive because they can learn to estimate the parameters of some population using a small number of exemplars (one or a few) at a time. Basically, neural networks are built from simple units, sometimes

called neurons or cells by analogy with the real thing. These units are linked by a set of weighted connections. Learning is usually accomplished by modification of the connection weights. Each unit codes or corresponds to a feature or a characteristic of a pattern that we want to analyze or that we want to use as a predictor. These networks usually organize their units into several layers. The first layer is called the input layer, the last one the output layer. The intermediate layers (if any) are called the hidden layers. The information to be analyzed is fed to the neurons of the first layer and then propagated to the neurons of the second layer for further processing. The result of this processing is then propagated to the next layer and so on until the last layer. Each unit receives some information from other units (or from the external world through some devices) and processes this information, which will be converted into the output of the unit. The goal of the network is to learn or to discover some association between input and output patterns, or to analyze, or to find the structure of the input patterns. The learning process is achieved through the modification of the connection weights between units [21].

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Figure 2.8. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are needed to train a network.

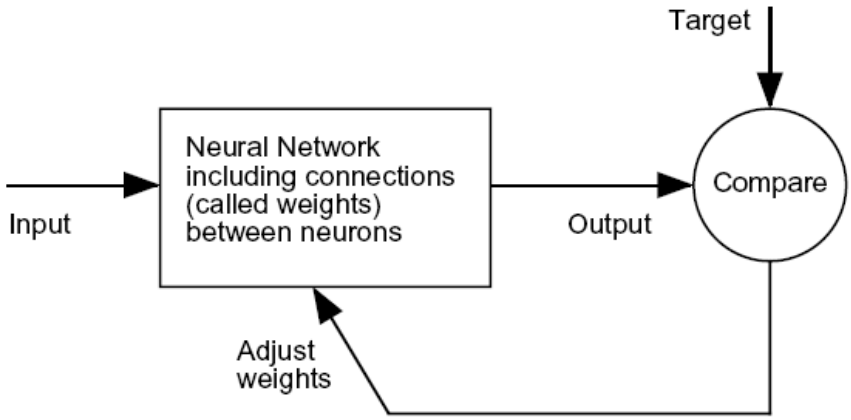

Figure 2.8 – General neural network training architecture

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems. Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings [22].

The learning algorithms used in this study is given in the following two sub-sections for the sake of completeness. These algorithms are the least mean squares learning (LMS) method and its expansion to the multilayered networks back-propagation learning algorithm. LMS learning algorithm is used for the training of the neural linear system identifier network and BP learning algorithms is used for the training of both the neural PID tuner and neural friction identifier.

2.6.2.1 Linear Networks and Least Mean Squares (LMS) Learning

The linear networks similar to the perceptrons, but their transfer function are linear rather than hard limiting. This allows their outputs to take on any value, whereas the perceptron output is limited to either 0 or 1. Linear networks, like the perceptron, can only solve linearly separable problems.

Like the perceptron, the linear network has a decision boundary shown in Figure 2.9 that is determined by the input vectors for which the net input n is zero. For $n = 0$ the equation $W_p + b = 0$ specifies such a decision boundary. Input vectors in the upper right gray area lead to an output greater than 0. Input vectors in the lower left white area lead to an output less than 0. Thus, the linear network can be used to classify objects into two categories. However, it can classify in this way only if the objects are linearly separable [22].

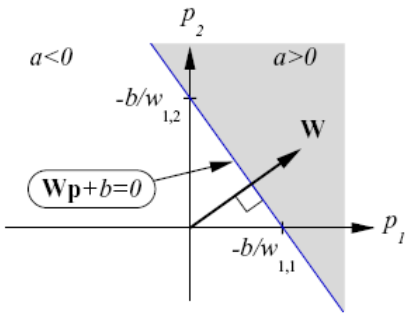

Figure 2.9 – Decision boundary representation for linear neural networks

The LMS algorithm, or Widrow-Hoff learning algorithm, is based on an approximate steepest descent procedure. Here again, linear networks are trained on examples of correct behavior. Widrow and Hoff had the insight that they could estimate the mean square error by using the squared error at each iteration [22]. Taking the partial derivative of the squared error with respect to the weights and biases at the kth iteration, we have,

$$\frac{\delta e^2(k)}{\delta w_{1,j}} = 2e(k)\frac{\delta e(k)}{\delta w_{1,j}} \tag{2.1}$$

for j = 1, 2, …, R and for the network bias

$$\frac{\delta e^2(k)}{\delta b} = 2e(k)\frac{\delta e(k)}{\delta b} \tag{2.2}$$

Next, the partial derivative of the error can be obtained as;

$$\frac{\delta e(k)}{\delta w_{1,j}} = \frac{\delta(t(k)-a(k))}{\delta w_{1,j}} = \frac{\delta}{\delta w_{1,j}}\left(t(k)-\left(W_p(k)+b\right)\right) \tag{2.3}$$

or

$$\frac{\delta e(k)}{\delta w_{1,j}} = \frac{\delta}{\delta w_{1,j}}\left(t(k)-\left(\sum_{i=1}^{n} w_{1,j}\,p_i(k)+b\right)\right) \tag{2.4}$$

Therefore;

$$\frac{\delta e(k)}{\delta w_{1,j}} = -p_j(k) \tag{2.5}$$

and

$$\frac{\delta e(k)}{\delta b} = -1 \tag{2.6}$$

Then, the Widrow-Hoff (LMS) learning algorithms can be formalized as;

$$W(k+1) = W(k) + 2\eta e(k) p^T(k) \tag{2.7}$$

and

$$b(k+1) = b(k) + 2\eta e(k) \tag{2.8}$$

Here e is the error vector and b is the bias vector, while η is a learning rate. If η is large, learning occurs quickly, but if it is too large it can lead to instability of the learning process and errors might even increase and oscillate [22].

2.6.2.2 Back-Propagation Neural Networks

Back-propagation neural networks are a class of networks that propagates inputs forward with all outputs is computed using sigmoid thresholding of the inner product of the corresponding weight and input vectors and all outputs at stage *n* are connected to all the inputs at stage *n+1*. It propagates the errors backwards by apportioning them to each unit according to the amount of this error the unit is responsible for [23].

In the following part the derivation of the stochastic Backpropagation algorithm for the general case is given.

First, the variables of the network are defined. $\vec{x}_j$ is the input vector for unit j (xji = ith input to the jth unit), $\vec{w}_j$ is the weight vector for unit *j* ($w_{j,i}$ is the weight on $x_{j,i}$) , $z_j = \vec{w}_j . \vec{x}_j$ is the weighted sum of inputs for unit *j*, o_j is the output of unit *j* ($o_j = \sigma(z_j)$), t_j is the target for unit *j*, $Downstream(j)$ is the set of units whose immediate inputs include the output of *j*, *Outputs* is the set of output units in the final layer.

The notation can be simplified by imagining that the training set consists of exactly one example and so the error can simply be denoted by E.

It is needed to calculate $\frac{\delta E}{\delta w_{j,i}}$ for each input weight $w_{j,i}$ for each output unit *j*. Note first that since z_j is a function of $w_{j,i}$ regardless of where in the network unit *j* is located,

$$\frac{\delta E}{\delta w_{j,i}} = \frac{\delta E}{\delta z_j} \cdot \frac{\delta z_j}{\delta w_{j,i}} = \frac{\delta E}{\delta z_j} x_{j,i} \tag{2.9}$$

Furthermore, $\frac{\delta E}{\delta z_j}$ is the same regardless of which input weight of unit *j* we are trying to update. So this quantity is denote by ρ_i and called back propagation common term.

Consider the case when $j \in Outputs$. It is known that

$$E = \frac{1}{2}\left(\sum_{k \in Outputs} (t_k - \sigma(z_k))^2 \right) \tag{2.10}$$

Since the outputs of all units $k \neq j$ are independent of $w_{j,i}$, we can drop the summation and consider just the contribution to E by *j*.

$$\rho_j = \frac{\delta E}{\delta z_j} = \frac{\delta}{\delta z_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j) \frac{\delta o_j}{\delta z_j} = -(t_j - o_j) \frac{\delta}{\delta z_j} \sigma(z_j) \tag{2.11}$$

$$\rho_j = -(t_j - o_j)(1 - \sigma(z_j))\sigma(z_j) = -(t_j - o_j)(1 - o_j) o_j \tag{2.12}$$

Thus, the network weight update term can be stated as;

$$\Delta w_{j,i} = -\eta \frac{\delta E}{\delta w_{j,i}} = \eta \rho_j x_{j,i} \tag{2.13}$$

Now consider the case when *j* is a hidden unit. Like before, we make the following two important observations.

- For each unit k downstream from *j*, z_k is a function of z_j
- The contribution to error by all units $l \neq j$ in the same layer as *j* is independent of $w_{j,i}$

We want to calculate $\frac{\delta E}{\delta w_{j,i}}$ for each input weight $w_{j,i}$ for each hidden unit *j*. Note that $w_{j,i}$ influences just z_j which influences o_j which influences $z_k \forall k \in Downstream(j)$ each of which influence E. So we can write;

$$\frac{\delta E}{\delta w_{j,i}} = \sum_{k \in Downstream(j)} \frac{\delta E}{\delta z_k} \frac{\delta z_k}{\delta o_j} \frac{\delta o_j}{\delta z_j} \frac{\delta z_j}{\delta w_{j,i}} \tag{2.14}$$

$$\frac{\delta E}{\delta w_{j,i}} = \sum_{k \in Downstream(j)} \frac{\delta E}{\delta z_k} \frac{\delta z_k}{\delta o_j} \frac{\delta o_j}{\delta z_j} x_{j,i} \tag{2.15}$$

Again, note that all the terms except $x_{j,i}$ in the above product are the same regardless of which input weight of unit *j* we are trying to update. Like before, we denote this common quantity by ρ_j. Also note that $\frac{\delta E}{\delta z_k} = \rho_k$, $\frac{\delta z_k}{\delta o_j} = w_{k,j}$ and $\frac{\delta o_j}{\delta z_j} = o_j(1 - o_j)$. Substituting,

$$\rho_j = \sum_{k \in Downstream(j)} \frac{\delta E}{\delta z_k} \cdot \frac{\delta z_k}{\delta o_j} \cdot \frac{\delta o_j}{\delta z_j} = \sum_{k \in Downstream(j)} \rho_k w_{k,j} o_j (1 - o_j) \tag{2.16}$$

Thus,

$$\rho_k = o_j (1 - o_j) \sum_{k \in Downstream(j)} \rho_k w_{k,j} \tag{2.17}$$

Finally, the network weights are updated using the following equation;

$$w_{j,i}(k+1) = w_{j,i}(k) + \Delta w_{j,i} \tag{2.18}$$

where

$$\Delta w_{j,i} = \eta \rho_j x_{j,i} \tag{2.19}$$

As a summary, a literature survey is performed for the stabilization control of turret subsystems covering different methods for stabilization control, friction models and compensation techniques, intelligent control, neural networks and learning algorithms for the training of these networks.

The stabilization control for turret subsystems are usually performed using classical control techniques. Modern control techniques and little work on intelligent control for turret subsystems can also be found. Compensation of system non-linearities shows considerable improvement on the controller performance according to the survey. In addition, identification of the system dynamics is used for these compensation algorithms.

Using the information obtained from literature survey, an intelligent controller architecture is proposed using artificial neural networks and classical linear controller in the next chapter.

CHAPTER 3

PROPOSED CONTROLLER ARCHITECTURE

An intelligent stabilization controller architecture for turret subsystems is proposed in this chapter using a classical discrete PID controller, a disturbance compensator and three neural networks. The proposed intelligent controller architecture is given in Figure 3.1.

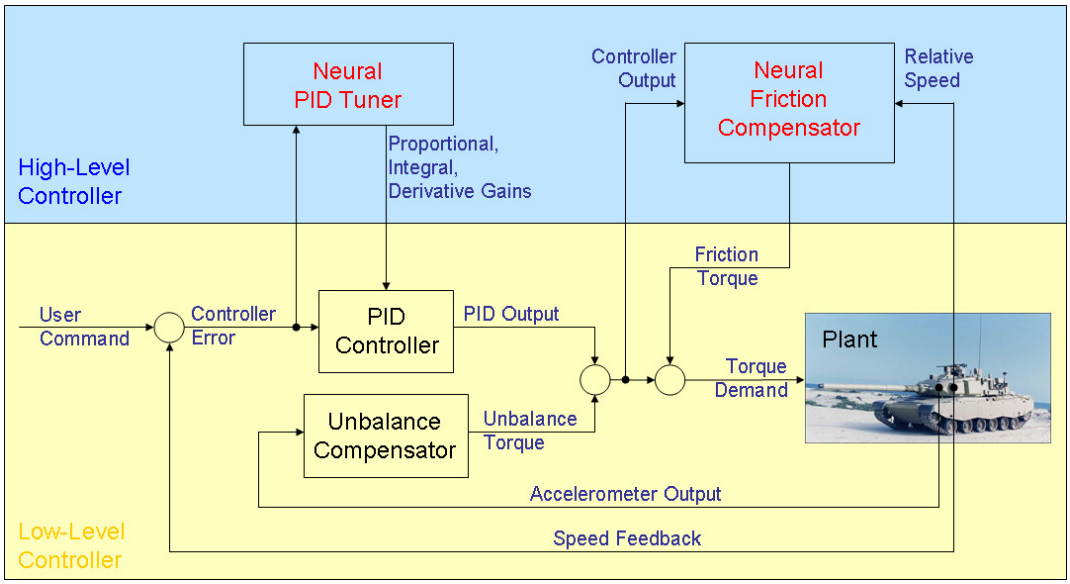

Figure 3.1 – Proposed Controller Architecture

The controller has two layers. The layer called “Low-Level Controller” is responsible from the stabilization of the corresponding axis of the turret subsystem. There is a classical discrete PID controller at this level and the performance of this controller is enhanced using an unbalance compensator. This unbalance compensator uses a gravity sensitive accelerometer in order to sense the unbalance. The details of the low-level controller are given in the next chapter.

In the second layer called as “High-Level Controller”, there are neural networks in which there is learning and adaptation of the low-level controller. The controller parameters of the classical PID controller are tuned in order to minimize the controller error using a multilayered neural network and this network is trained using back-propagation learning algorithm. In addition, there is a compensation algorithm for Columb friction using a multilayered neural network. A linear neural network is used for the linear identification of the physical system and the output of this identification is used for the training of the Columb friction compensation network. Details of these networks will be given in section 3.3.

The details of the proposed intelligent stabilization controller architecture start with the low-level controller. The mathematical details of this low-level controller will be given in the next section. Then, mathematical derivations for the neural PID tuner network and design of this network will be stated. Finally, this chapter ends with the mathematical derivation of the networks used for the compensation of Columb friction and design of these networks.

3.1 Low-Level Stabilization Controller

In the proposed control architecture, the low-level controller is based on a well-known, easy structured discrete PID controller. That can be stated as;

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i e(k) + K_d\left(e(k) - 2e(k-1) + e(k-2)\right) \quad (3.1)$$

where $u(k)$: controller output at time k

$e(k)$: error between the speed command and speed feedback at time k

K_p: proportional gain of the PID controller

K_i: integral gain of the PID controller

K_p: derivative gain of the PID controller

The proportional gain of this controller mainly works against high amplitude transient errors. When it senses an error, it produces an output proportional to this error. Due to this fact, proportional controllers suffer from output saturation and

constant steady state error may occur. The integral controller is introduced in order to overcome this constant error but high integral gain may cause oscillations in steady state. Therefore, in order to suppress these oscillations at steady state and to reduce the fluctuations while reaching this steady state, derivative controller is used [37]. Although derivative controller improves the controller performance especially in position control loops, derivative controller is usually either used with very low gains or is not used at all due to noise on the feedback signals. If derivative component of a PID controller is not used then the controller is called proportional + integral controller or PI controller.

The schematic view of this control structure is shown in Figure 3.2 below. As seen from the figure, there are two possible speed feedback signals. If speed feedbacks are obtained using the resolver of the driving motor or any other relative speed feedback device, the axis in concern tracks the desired speed trajectory relative the structure on which it stands. For example, if the axis in concern is the elevation axis of a battle tank and speed feedback is obtained from resolver signal then elevation axis tracks the speed command relative to the traverse axis of the tank. This operation mode is often called as nonstab-mode or normal-mode.

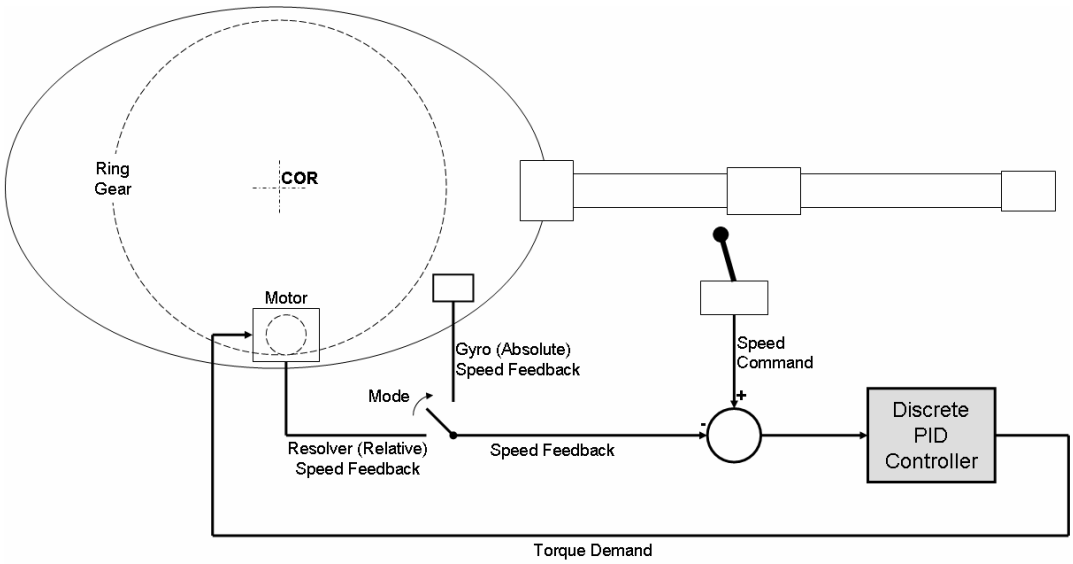

Figure 3.2 – Controller structure of a turret with discrete PID controller

On the other hand, if the speed feedback is obtained using the gyroscope signal, the axis in concern tracks the desired trajectory relative to a reference on the ground. Similarly, if the axis in concern is the elevation axis of a battle tank, the elevation axis tracks the speed commands relative to a reference on the ground. This operation mode is often called as stab-mode.

Usually, in the mechanical design stage of turret subsystems, designers try to minimize the non-linear behavior of the system in order to simplify the controller design. However, one of these non-linear behaviors called unbalance cannot be overlooked because of the limitations of the mechanical structure that will result in unbalance that will directly affect controller performance. This non-linear effect should be then compensated for increased efficiency in performance.

More specifically, unbalance compensation is required if maximum unbalance torque is comparable in magnitude with maximum torque of the driving motor. In this context, we should note that battle tanks with 105 mm main gun have unbalance values much less than the maximum torque of the elevation axis motor and in these systems unbalance is not compensated. However, tanks with 120 mm and higher caliber main guns have considerable unbalance values therefore in most of these systems unbalance compensation algorithms are used.

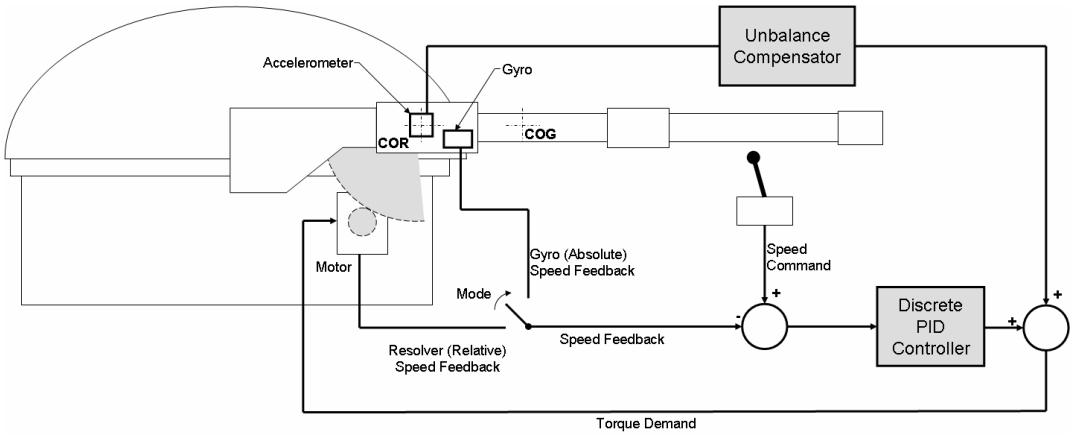

Figure 3.3 – Controller structure of a turret with discrete PID controller and unbalance compensation

Since unbalance is a mechanical concept and its value does not change with time or with other effects, deterministic algorithms can be used with satisfactory performance. One of these algorithms used in the elevation axis of main battle tanks is based on linear accelerations in the direction perpendicular to the line of sight of the gun for the calculation of the unbalance torque. In this algorithm, a gravity sensitive accelerometer is placed at the center of rotation of the elevation axis. The measurement axis of this accelerometer is aligned with the direction, which is perpendicular to the line of sight.

The measured accelerations are multiplied with the distance along the line of sight between the center of rotation (COR) and center of gravity (COG) with mass of the gun. The whole equation is given as;

$$T_u = a_m \times l \times m \tag{3.2}$$

where a_m : Measured acceleration

l : Distance along the line of sight between COR and COG

m : Mass of the gun

Using this equation, the unbalance torque can be calculated and corresponding control signal can be added to the output of the PID controller to compensate the unbalance. The schematic view of this control structure is shown in Figure 3.3.

3.2 Neural PID Tuner

Although there are, some methods to tune the parameters of a PID controller in time or frequency domain or in continuous or discrete time, but the controller performance cannot be met because of the complexity of the physical system. Therefore, designers apply offline methods to tune the controller up to some level and afterwards they tune online the controller manually by trial-end-error to reach the desired performance.

In offline tuning processes, the dynamics of the physical system are usually time dependent and may also change with environmental effects resulting in a controller performance that may no longer be acceptable or even lead to

unstability. To solve these problems, adaptive controllers are developed but most of these controllers are model dependent and therefore the designer should obtain a closed form mathematical model of the physical system.

These difficulties are the well-known challenges of the controller design process confronting control engineers. In order to minimize the design effort and obtain an adaptive type controller, which is not model dependent, a PID tuner neural network is used for battle tank turrets in this study. The parameters of this network are optimized by performing computer simulations with different parameters in order to have the best stabilization performance.

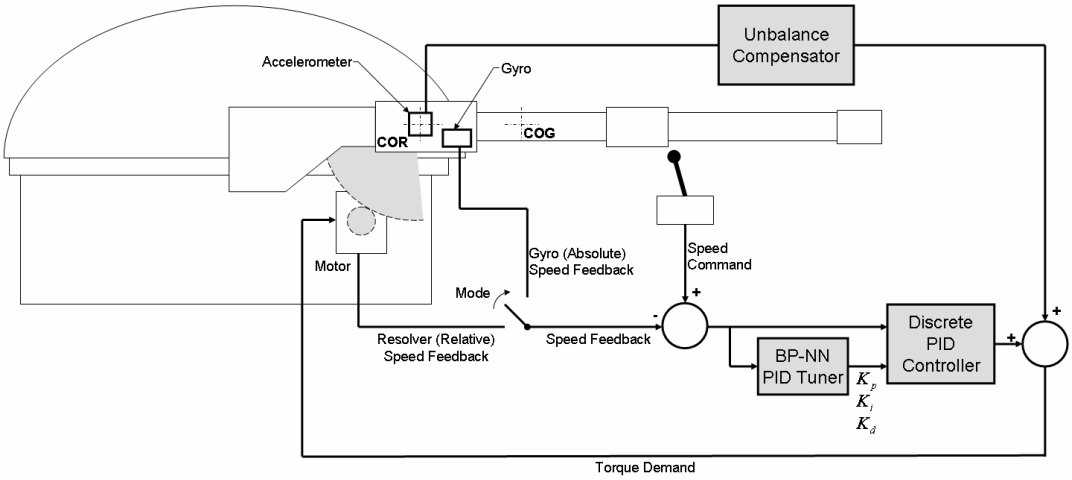

Figure 3.4 - Controller structure of a turret with unbalance compensation and PID controller tuned by a BP-NN PID tuner.

The place of this PID tuner network is in Figure 3.4 for a battle tank turret. As seen from this figure, PID tuner network inputs current and previous values of controller error signal between the command and feedback signals in order to tune the proportional (P), integral (I) and derivative (D) components of a PID controller based on the back propagation learning algorithm in order to minimize the controller error.

The mathematical derivations of this PID tuner network will be given in the next subsection in detail. After this mathematical interpretation, the design of the

network parameters for the best stabilization performance will be given by performing computer simulations on a main battle tank mathematical model.

3.2.1 Mathematical Derivations for Neural PID Tuner Network

The back propagation neural network used for the tuning of the gains of the PID controller consists of an input, an output and M hidden layers. Current and previous values of error signal between the commanded signal and feedback signal are the inputs to the network. The outputs of the network are the proportional (P), integral (I) and derivative (D) gains of the PID controller. The block diagram of the neural PID tuner is shown in Figure 3.5 below.

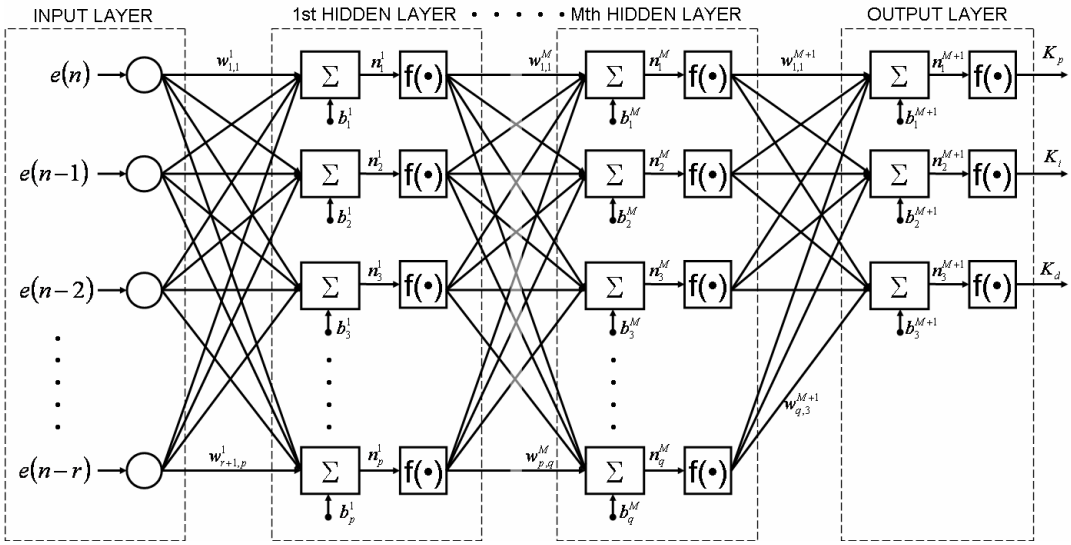

Figure 3.5 – Architecture of the neural PID tuner

In Figure 3.5, $w_{j,i}^{n}$ represents the network weight between the neuron j at layer $n-1$ and neuron i at layer n. Similarly b_i^n represents the neuron bias of neuron i at layer n. The activation functions of the neurons are shown as **f(•)**. The activation functions used in back-propagation neural networks are usually either logarithmic sigmoid (Figure 3.6) or hyperbolic sigmoid (Figure 3.7) functions. The type of the activation function used in this network and the reasons for this selection will be given in the design stage of this network.

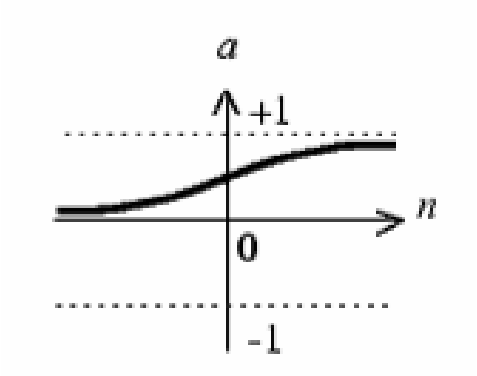

Figure 3.6 – Logarithmic sigmoid function

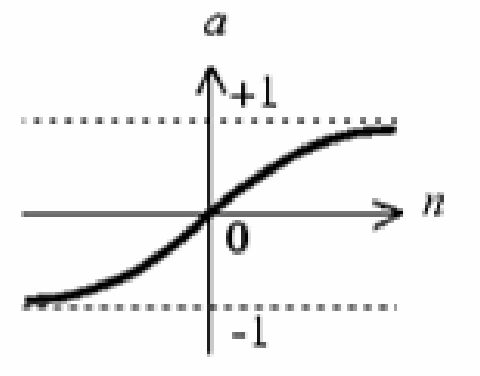

Figure 3.7 – Hyperbolic sigmoid function

The PID tuner back propagation neural network aims to minimize the error between the commanded signal and the feedback signal at the next sampling period. In order to perform this, the network used the following cost function.

$$J = \frac{1}{2}(r(k+1) - y(k+1))^2 = \frac{1}{2}\big(e(k+1)\big)^2 \tag{3.3}$$

where, $r(k+1)$: Commanded signal

$y(k+1)$: Feedback signal

$e(k+1)$: Error signal

Using this cost function, the network weights are updated using the following update equation.

$$w_{j,i}^{n}(k+1) = w_{j,i}^{n}(k) + \Delta w_{j,i}^{n}(k+1) \text{ where } \Delta w_{j,i}^{n}(k+1) = -\eta \frac{\delta J}{\delta w_{j,i}^{n}(k)} + \alpha \Delta w_{j,i}^{n}(k) \qquad (3.4)$$

In this update equation, the network weight ($w_{j,i}^{n}$) is updated using the current network weight, current and previous update terms ($\Delta w_{j,i}^{n}$) , learning rate η and momentum term α .

Learning rate determines the speed of the learning process. Usage of high learning rate values increases the speed of the learning process but this may lead to unstability whereas using low values may lead very slow learning speed, which is practically unacceptable. Therefore the learning rate (η) is a design parameter which should be determined carefully.

In order to suppress oscillations of network weight updates in the learning process, the momentum term α is introduced into the update equation, which behaves like a damper on the network weight adding inertia to the learning process. The importance of this momentum term in learning process and optimum values obtained by computer simulations will be given in the next section for both momentum term and learning rate.

The update term $\Delta w_{j,i}^{n}$ is obtained using the following partial differential equation.

$$\frac{\delta J}{\delta w_{j,i}^{n}(k)} = \frac{1}{2} \frac{\delta \left[\left(r(k+1) - y(k+1) \right)^2 \right]}{\delta w_{j,i}^{n}(k)} = -e(k+1) \frac{\delta y(k+1)}{\delta w_{j,i}^{n}(k)} \qquad (3.5)$$

This equation can be expanded using the chain rule and the following equation can be obtained. In this equation $u(k)$ denotes the control input to the plant.

$$\frac{\delta J}{\delta w_{j,i}^{n}(k)} = -e(k+1) \frac{\delta y(k+1)}{\delta u(k)} \frac{\delta u(k)}{\delta out_{i}^{n}(k)} \frac{\delta out_{i}^{n}(k)}{\delta n_{i}^{n}(k)} \frac{\delta n_{i}^{n}(k)}{\delta w_{j,i}^{n}(k)} \qquad (3.6)$$

The term out_i^n represents the neuron activation function output of neuron i at layer n and it is given as $out_i^n = f_n\left(n_i^n\right)$.

The update term can be further simplified using the following equalities.

$$\frac{\delta n_i^n(k)}{\delta w_{j,i}^n(k)} = \frac{\delta\left(w_{1,1}^n(k)out_1^{n-1}(k)+\ldots+w_{j,i}^n(k)out_j^{n-1}(k)+\ldots+b_i^n(k)\right)}{\delta w_{j,i}^n(k)} = out_j^{n-1}(k) \quad (3.7)$$

and

$$\frac{\delta out_i^n(k)}{\delta n_i^n(k)} = f_n'\left(n_i^n(k)\right) \quad (3.8)$$

Using these equalities, following equality called back propagation common term ρ_i^n can be defined as follows.

$$\rho_i^n(k) = \frac{\delta u(k)}{\delta out_i^n(k)}\frac{\delta out_i^n(k)}{\delta n_i^n(k)} = \frac{\delta u(k)}{\delta out_i^n(k)} f_n'\left(n_i^n(k)\right) \quad (3.9)$$

This common term can be obtained for previous network layers using the following equalities,

$$\rho_j^{n-1}(k) = \frac{\delta u(k)}{\delta out_j^{n-1}(k)}\frac{\delta out_j^{n-1}(k)}{\delta n_j^{n-1}(k)} \quad (3.10)$$

$$\rho_j^{n-1}(k) = \left(\sum_i \frac{\delta u(k)}{\delta out_i^n(k)}\frac{\delta out_i^n(k)}{\delta n_i^n(k)}\frac{\delta n_i^n(k)}{\delta out_j^{n-1}(k)}\right) f_{n-1}'\left(n_j^{n-1}(k)\right) \quad (3.11)$$

$$\rho_j^{n-1}(k) = \left(\sum_i \frac{\delta u(k)}{\delta out_i^n(k)}\frac{\delta out_i^n(k)}{\delta n_i^n(k)}\omega_{j,i}^n(k)\right) f_{n-1}'\left(n_j^{n-1}(k)\right) \quad (3.12)$$

Further simplification of this equation results in a recursive equation given below,

$$\rho_j^{n-1}(k) = \left(\sum_i \rho_i^n(k) w_{j,i}^n(k)\right) f_{n-1}'\left(n_j^{n-1}(k)\right) \quad (3.13)$$

Therefore, the network weight update equation can be rewritten as,

$$w_{j,i}^{n}(k+1) = w_{j,i}^{n}(k) - \eta\left(-e(k+1)\frac{\delta y(k+1)}{\delta u(k)}\rho_i^n(k) out_j^{n-1}(k)\right) + \alpha\Delta w_{j,i}^{n}(k) \qquad (3.14)$$

Up to now, only one unknown is left which is the change of plant output with respect to plant input $\frac{\delta y(k+1)}{\delta u(k)}$. Since the proposed method do not use the mathematical model of the system, the best and easiest way is to use the following equality,

$$\frac{\delta y(k+1)}{\delta u(k)} = sign\left(\frac{\delta y(k+1)}{\delta u(k)}\right).\left|\frac{\delta y(k+1)}{\delta u(k)}\right| \qquad (3.15)$$

Therefore, if we know the sign of this term, we can merge the amplitude of the term with the learning term η and simplify the network weight update equation as follows,

$$w_{j,i}^{n}(k+1) = w_{j,i}^{n}(k) + \eta e(k+1) sign\left(\frac{\delta y(k+1)}{\delta u(k)}\right)\rho_i^n(k) out_j^{n-1}(k) + \alpha\Delta w_{j,i}^{n}(k) \qquad (3.16)$$

Since the common term ρ_i^n is obtained recursively, its values at the output layer should be obtained. The following equations can be used to obtain the values of these common terms.

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i e(k) + K_d\left(e(k) - 2e(k-1) + e(k-2)\right) \qquad (3.17)$$

where $K_p = out_1^n(k)$, $K_i = out_2^n(k)$, $K_d = out_3^n(k)$

$$\rho_1^n(k) = \frac{\delta u(k)}{\delta out_1^n(k)} f_n'\left(n_1^n(k)\right) = \frac{\delta u(k)}{\delta K_p} f_n'\left(n_1^n(k)\right) = \left(e(k) - e(k-1)\right) f_n'\left(n_1^n(k)\right) \qquad (3.18)$$

$$\rho_2^n(k) = \frac{\delta u(k)}{\delta out_2^n(k)} f_n'\left(n_2^n(k)\right) = \frac{\delta u(k)}{\delta K_i} f_n'\left(n_2^n(k)\right) = e(k) f_n'\left(n_2^n(k)\right) \qquad (3.19)$$

$$\rho_3^n(k) = \frac{\delta u(k)}{\delta out_3^n(k)} f_n'\left(n_3^n(k)\right) = \frac{\delta u(k)}{\delta K_d} f_n'\left(n_3^n(k)\right) \qquad (3.20)$$

$$\rho_3^n(k) = (e(k) - 2e(k-1) + e(k-2)) f_n'(n_3^n(k)) \tag{3.21}$$

The update equation for the network biases is very similar to the update equation for the network weights. The equation is given below;

$$b_i^n(k+1) = b_i^n(k) - \eta \Delta b_i^n(k+1) + \alpha \Delta b_i^n(k) \text{ where } \Delta b_i^n(k+1) = \frac{\delta J}{\delta b_i^n(k)} \tag{3.22}$$

The expansion of the term $\Delta b_i^n(k+1)$ is very similar to those used for weight update equation; therefore, it is not given here. The bias update equation obtained following the same approach is given as;

$$b_i^n(k+1) = b_i^n(k) + \eta e(k+1) sign\left(\frac{\delta y(k+1)}{\delta u(k)}\right) \rho_i^n(k) + \alpha \Delta b_i^n(k) \tag{3.23}$$

In this equation, different than the one obtained for network weights, the term $out_j^{n-1}(k)$ is not present. This is due to the difference in the term $\frac{\delta n_i^n(k)}{\delta b_i^n(k)}$ given below;

$$\frac{\delta n_i^n(k)}{\delta b_i^n(k)} = \frac{\delta\left(w_{1,1}^n(k) out_1^{n-1}(k) + \ldots + w_{j,i}^n(k) out_j^{n-1}(k) + \ldots + b_i^n(k)\right)}{\delta b_i^n(k)} = 1 \tag{3.24}$$

The mathematical details of the neural PID tuner network are given under this title. In the next section, design process for the optimization of network parameters for the best stabilization performance will be given in detail.

3.2.2 Design of the Neural PID Tuner Network

The neural PID tuner is a multilayer neural network and uses two parameters called learning rate (η) which determines how fast the learning occurs and momentum term (α) which damps the learning process in order to reduce the fast changes in the network gains. In this chapter, the optimum values for these network parameters will be determined.

The neural PID tuner uses an input layer with n-many inputs which are the present and previous control errors and an output layer which are the controller parameters

K_p, K_i and K_d. It also uses m-many hidden layers, which may also have more than one neuron.

The inputs of the network are normalized using the maximum allowable error limit values (3 deg/sec, 0.05 rad/sec). This normalization is done in order to obtain uniform inputs to the sigmoid functions. Due to the shape of the sigmoid functions, they produce almost the same output if the input is larger than 1 or less than -1. Therefore, the inputs of the network is divided by the maximum allowable error value and by two in order to obtain inputs between -0.5 to 0.5.

The parameters of the neural PID tuner, which are needed to be optimized, can be listed as the learning rate, momentum term, and number of inputs, number of hidden layers and number of neurons in these hidden layers. There is also need for the selection of the activation functions for each neuron. The details of the optimization of these parameters will be given in the following few pages.

The learning process does not occur through the whole simulation. The training of the network stops after the controller reaches to a satisfactory performance level. This is done by utilizing the forgetting sum of the controller errors. In other words, the controller errors are accumulated with a forgetting term and if this sum becomes lower than a limit, which means that the controller reached a satisfactory performance, the training process of the network stops and network continues to run without training. The forgetting sum of the controller errors are calculated by the following equation.

$$err(n) = \lambda.err(n-1) + \left(cmd(n) - fbk(n)\right) \tag{3.25}$$

Here in this equation, $err(n)$ is the sum of the errors at time step n, λ is the forgetting term which is a positive number between 0 and 1, and this term determines the decay of the effects of the previous errors in the present sum of errors. The term $cmd(n)$ is the speed command from the user and $fbk(n)$ is the speed feedback from the rate gyroscope of the corresponding axis.

In this study, forgetting term is used as $\lambda = 0.95$ and this value leads to the loss of an error value within approximately 1 second. Selection of low forgetting term values leads the learning process dependent mostly on the current controller errors and usage of high values leads to a learning process dependent mostly on the pervious controller errors. For low values, learning process slows down but for high values learning occurs fast. The value used in this study is determined by trial-and-error and it gives satisfactory performance.

As stated before, the neural PID tuner has input and output layers and multi hidden layers. The number of hidden layers, number of neurons, which will be used in these hidden layers and type of the activation functions, are needed to be determined.

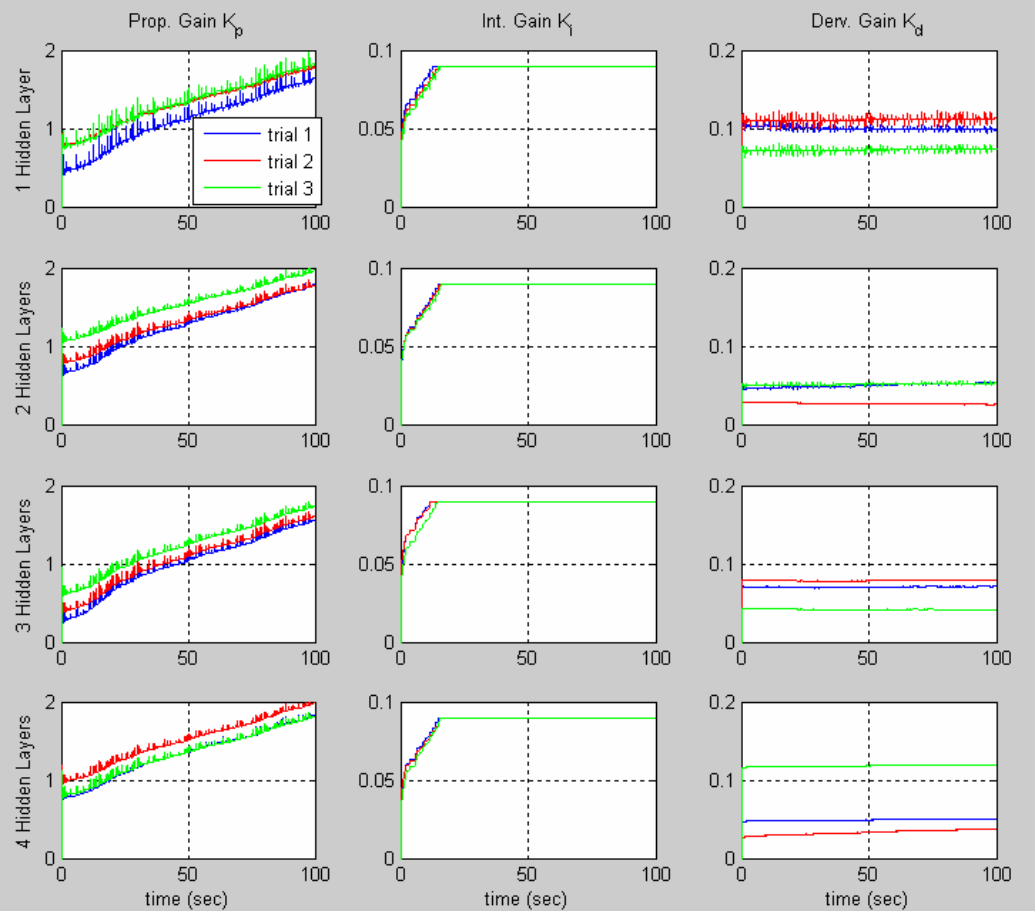

Figure 3.8 – Controller Gains Obtained for Different Number of Hidden Layers for the Tank Moving on APG Track with a Speed of 10 km/h

Since the proposed neural network is aimed to be used in real-time applications for the control of turret subsystems, minimum number of hidden layers and minimum number of neurons in these layers are the goals of design. Addition of hidden layers to the network increases considerably the computational complexity of the algorithm therefore; simulations are performed using different numbers of hidden layers in order to determine the optimum number of hidden layers.

The PID tuner network is simulated over the computer simulations for the tank moving on APG track with a speed of 10 km/h in stab-mode for elevation axis. In other words, the neural PID tuner performs the stabilization task for elevation axis while moving over the APG track. The simulation results are given only for elevation axis but the results obtained for traverse axis are very similar.

The controller gains obtained in these simulations are given in Figure 3.8 as a figure matrix. In this figure, first column shows the proportional gain of the controller with respect to time, second column shows the integral gain of the controller with respect to time and the last column shows the derivative gain of the controller with respect to time. In all of the subfigures, there are three different data, which are obtained from other trials. The gain limits used in this simulations are 0 as minimum and 10 as maximum for K_p, 0 as minimum and 0.3 as maximum for K_i and 0 as minimum and 1 as maximum for K_d.

For all simulations, learning rate is used as 0.01 and momentum term is used as 0.5. These values are not the optimum values but these values give satisfactory performance for all cases. The optimization of these values will be discussed later. In addition, the initial network weights and initial biases are random numbers between -0.1 and 0.1. These values are not too critical for the network performance assuring that the network does not start with high controller gains. In order to satisfy this constraint, the initial weights and biases are chosen as non-zero and small enough numbers. Choosing too small initial weights and biases increases the training time a little but large numbers may result in high initial control gains. Also starting with positive weights and biases forces the network for

positive weights and biases thought the simulation; therefore, use of both negative and positive numbers assures the uniformity of network weights and biases.

The rows in the figure correspond to different number of hidden layers. First raw shows the results obtained for one hidden layer and the last one shows the results for four hidden layers. As can be seen from the figure, the controller gains do not change too much with different number of hidden layers. In other words, both the shape of the controller gain curves and gain values are similar for each number of hidden layers. On the other hand, the performance criterion which will determine performance increase or decrease of change in the number of hidden layers is not the controller gains since controller gain do not mirror the performance of the controller. The performance criterion of stabilization controller is surely the stabilization accuracy of the tank on APG track [48] for elevation axis and stabilization accuracy of the tank on sinus track.

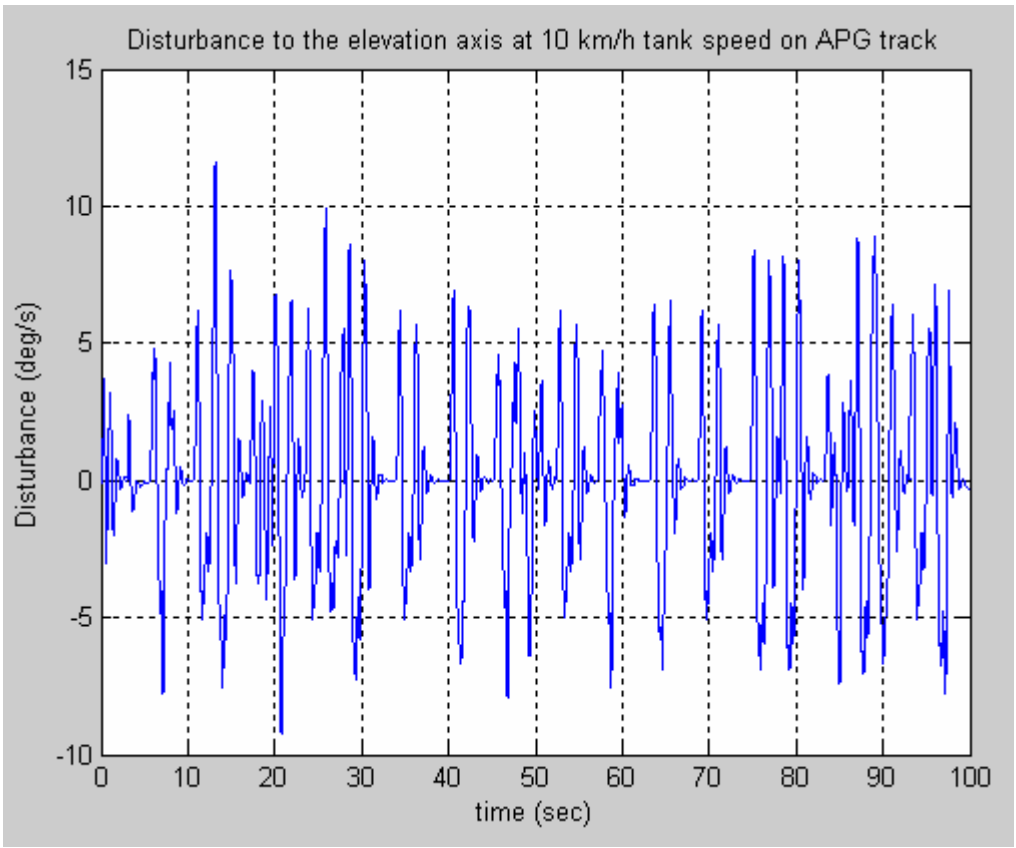

Figure 3.9 – Disturbance Speed for Elevation Axis on APG Track at 10 km/h

The disturbance speed affecting the elevation axis while the tank moves along the APG track with a speed of 10 km/h is given in Figure 3.9. With this disturbance effect, the deviations of the gun orientation from the ideal stabilization line for different number of hidden layers are given in Figure 3.10 as a figure matrix. Here the stabilization accuracy is given in mrads and is calculated as the standard deviation of the integral of the controller errors (Eqn-3.26) [48].

$$Stab.Acc = std\left(\int_0^t (Cmd - Fbk)dt\right) \tag{3.26}$$

In this figure, rows represent different number of hidden layers and columns represent the results obtained from different trials for each number of hidden layer. As can be seen from these results, increasing the number of hidden layers does not make any difference on the network performance.

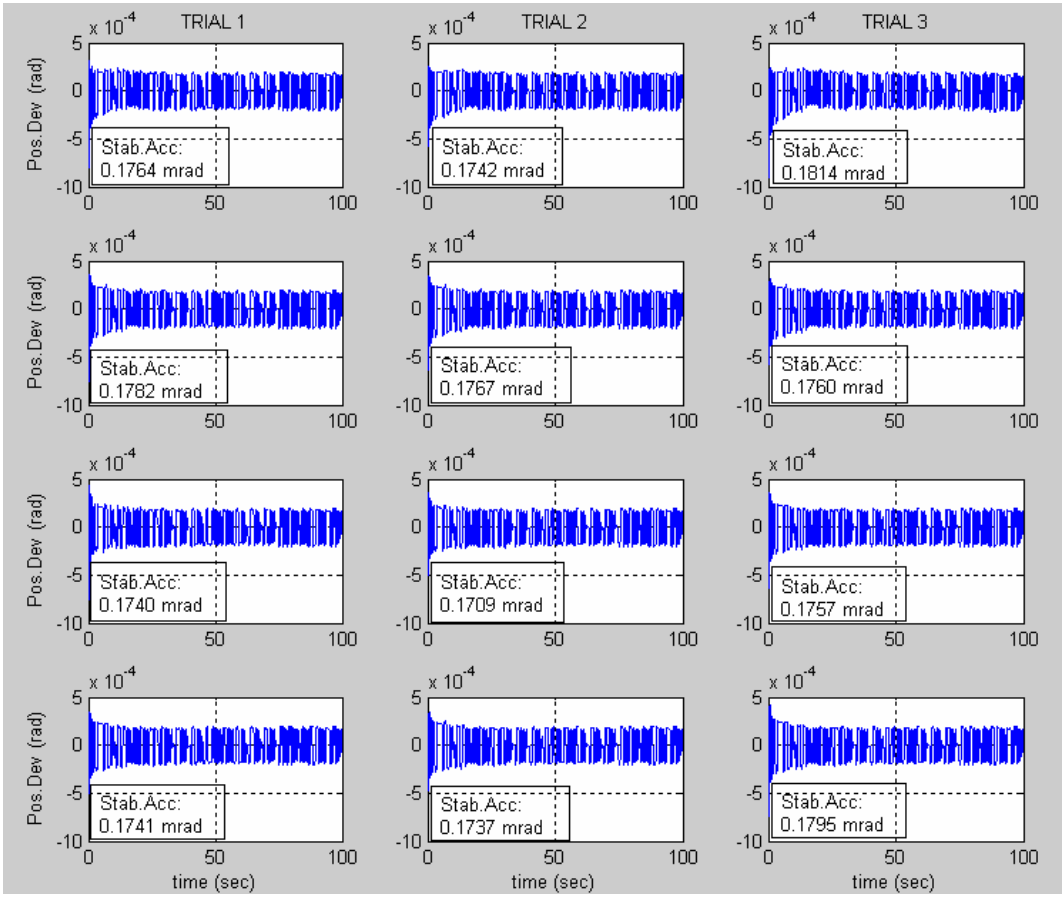

Figure 3.10 – Stabilization Accuracy of Elevation Axis on APG Track with Different Number of Hidden Layers

The stabilization accuracy values are more or less the same for all simulations. In addition, the controller gain trajectories are very similar for all cases. Therefore, due to the computational cost of using high number of hidden layers, it is decided to use only one hidden layer in the neural PID tuner in this study.

In all of the simulations discussed above, tangent hyperbolic activation functions are used for all the neurons. Logarithmic sigmoid function is not used because its output for zero input is 0.5 and it gives always a positive output. In other words, if logarithmic sigmoid function is used in the network, the network always starts with the middle of the controller gain limits; this also causes the loss of sign information of the inputs and always outputs positive gains. Because of these reasons, tangent hyperbolic sigmoid function will be used in all of the neurons for the neural PID tuner.

For the results discussed above, there were five neurons in the input layer and five neurons in all of the hidden layers. There are always three neurons in the output layer because there are three PID parameters, which are the outputs of the network. Up to this point, it is decided to use one hidden layer as a result of computer simulations and tangent hyperbolic activation functions for all neurons. After this point, the optimum number of neurons in the input layer and in the hidden layer will be searched.

For this purpose, computer simulations are performed for different number of neurons in the input layer and in the hidden layer. The results obtained for learning rate $\eta = 0.01$ and momentum term $\alpha = 0.5$ is given in

Table **1**. In these simulations, the initial values of the network weights and biases are random numbers between -0.1 and 0.1.

As can be seen from the results shown in the table, for low number of neurons in the input layer, the stabilization accuracy becomes better with the increase of the number of neurons in the hidden layer. However, increase of the number of neurons in the input layer, makes the stabilization accuracy worse up to 10 neurons. After this point, the stabilization accuracy becomes better with increasing

number of neurons in the input layer. In addition, it is obvious that for large number of neurons in the input layer, increase of neurons in the hidden layer makes the performance better up to 0.1674 mrad and further increase in the number of neurons in the hidden layer makes the stabilization accuracy worse.

Table 1 – Stabilization Accuracy Values for Different Number of Neurons for the Input and Hidden Layers for $\eta = 0.01$ and $\alpha = 0.5$

Stabilization Accuracy in mrads		**Number of Neurons in the Hidden Layer**			
		5	**10**	**20**	**50**
Number of Neurons in the Input Layer	**2**	0.1823	0.1643	0.1577	0.1538
	5	0.2088	0.1916	0.1698	0.1576
	10	0.2144	0.1900	0.1697	0.1696
	20	0.2121	0.1893	0.1800	0.1907
	50	0.2003	0.1787	0.1674	0.1752

Table 2 – Stabilization Accuracy Values for Different Number of Neurons for the Input and Hidden Layers for $\eta = 0.02$ and $\alpha = 0.5$

Stabilization Accuracy in mrads		**Number of Neurons in the Hidden Layer**			
		5	**10**	**20**	**50**
Number of Neurons in the Input Layer	**2**	0.1522	0.1531	0.1520	0.1514
	5	0.1602	0.1536	0.1525	0.1523
	10	0.1765	0.1663	0.1530	0.1514
	20	0.1539	0.1591	0.1519	0.1810
	50	1.4504	1.3848	0.1514	0.2301

These conclusions are valid for the given learning rate and momentum term. In order to generalize these conclusions, other simulations are performed for different

learning rate values. In , the results are summarized for learning rate $\eta = 0.02$ and momentum term $\alpha = 0.5$ and in , the results for learning rate $\eta = 0.005$ and momentum term $\alpha = 0.5$ are given.

The results obtained for learning rate values $\eta = 0.02$ and $\eta = 0.005$ are very similar to the results obtained for $\eta = 0.01$ although there are little exceptions. For learning rate $\eta = 0.02$, there are large stabilization accuracy values. These are due to the high oscillations in the system response due to the high controller gains obtained. Disregarding these two cases, the conclusions given above is still valid for these two learning rate values.

Table 3 – Stabilization Accuracy Values for Different Number of Neurons for the Input and Hidden Layers for $\eta = 0.005$ and $\alpha = 0.5$

Stabilization Accuracy in mrads		**Number of Neurons in the Hidden Layer**			
		5	**10**	**20**	**50**
Number of Neurons in the Input Layer	**2**	0.2810	0.1966	0.1718	0.1540
	5	0.2116	0.1965	0.1740	0.1570
	10	0.2634	0.2410	0.1631	0.1558
	20	0.2282	0.1580	0.1541	0.1533
	50	0.3180	0.1593	0.1743	0.1528

The results showed that, using large number of neurons in the input and hidden layers makes the stabilization accuracy better. However, the system response data in time domain shows that oscillations in the system response increases with increasing number of neurons in the input layer. This is mainly due to the increasing error history of the network. With large number of neurons in the input layer, the network continues to increase the controller gains even if the controller error is nearly zero. On the other hand, increase of the number of neurons in the hidden layer makes the network too sensitive to controller errors. In other words,

This effect is very similar to the case with high learning rate and affects the controller stability in a bad manner.

The neural PID tuner is developed mainly for real-time applications. Therefore, the computational complexity is a very important issue. In order to implement the network on real-time computers, the number of neurons in the input layers and hidden layers must be minimized. In Figure 3.11 the stabilization accuracy values are represented as surfaces for different number of neuron values of input and hidden layers for three different learning rate values. From these figures, it is obvious that increasing the number of neurons in the input layer may decrease the stabilization performance and increasing the number of neurons in the hidden layer increases the stabilization performance almost for every case. However, after 20 number of neurons in the hidden layer, further increase does not increase the stabilization performance. Therefore, an optimum value for the number of neurons in the hidden layer can be concluded as 20 and for the number of neurons in the input layer as 10 using these figures.

Until now, the number of hidden layers, number of neurons in the input and hidden layers and activation functions for these neurons are determined. Only the optimum values of learning rate and momentum term values remain to be analyzed. After this point, the optimum values of learning rate and momentum term will be determined by performing computer simulations for different values of learning rate and momentum term and the values found in this optimization will be used in the final computer simulations in order to show the performance of the neural PID tuner network on the mathematical model of the main battle tank.

In order to determine the optimum values of the learning rate and momentum term, computer simulations are performed using different values of the learning rate and momentum term. The simulations are performed two times for each combination. The values used for learning rate are 0.001, 0.003, 0.005, 0.007, 0.01, 0.02, 0.03, 0.04 and 0.05. These values are determined by pre-simulations and 0.001 is the minimum performance limit and 0.05 is the maximum stability limit. The momentum term can only have values between 0 and 1 therefore the

numbers 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9 are used in the simulations.

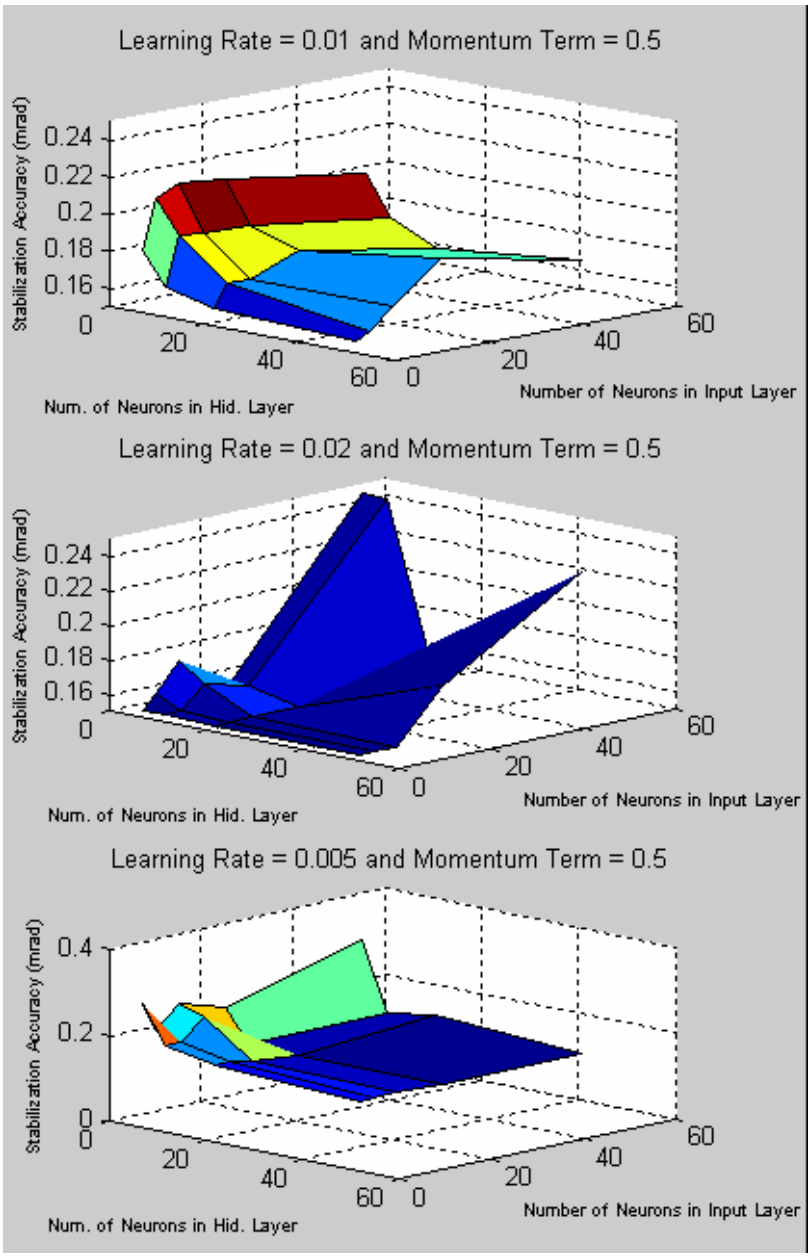

Figure 3.11 – Stabilization Accuracy Surfaces for Different Number of Neurons in the Input and Hidden Layers and for Different Learning Rate Values

The results are represented as stabilization accuracy surfaces in Figure 3.12 for the first trial and in Figure 3.13 for the second trial. Some the learning rate and momentum term values which makes the system unstable, the stabilization accuracy values are indicated with 0.3 mrad for easy visualization.

The results shows that for low values (between 0.001 and 0.01) of the learning rate, the stabilization performance is relatively low for almost all values of momentum term. Stabilization accuracy values vary between 0.15 mrad and 0.25 mrad for these value of learning rate and for all momentum term values. Also results show that, usage of high learning rate with low momentum term causes unstability. On the other hand, using high learning rate values with high momentum term values also make the system unstable.

For values of learning rate between 0.01 and 0.04 and momentum term values between 0.2 and 0.6, the stabilization performance is always good and the system is always stable. However, stabilization accuracy values are nearly the same (0.15 mrad) in this region. Therefore, the values of learning rate and momentum term should be selected form this region considering the stability of the system. The safest learning rate and momentum term values, which will give the best performance, are selected as the middle point of the bottom of the surface. The numerical values are $\eta = 0.025$ for the learning rate and $\alpha = 0.4$ for the momentum term.

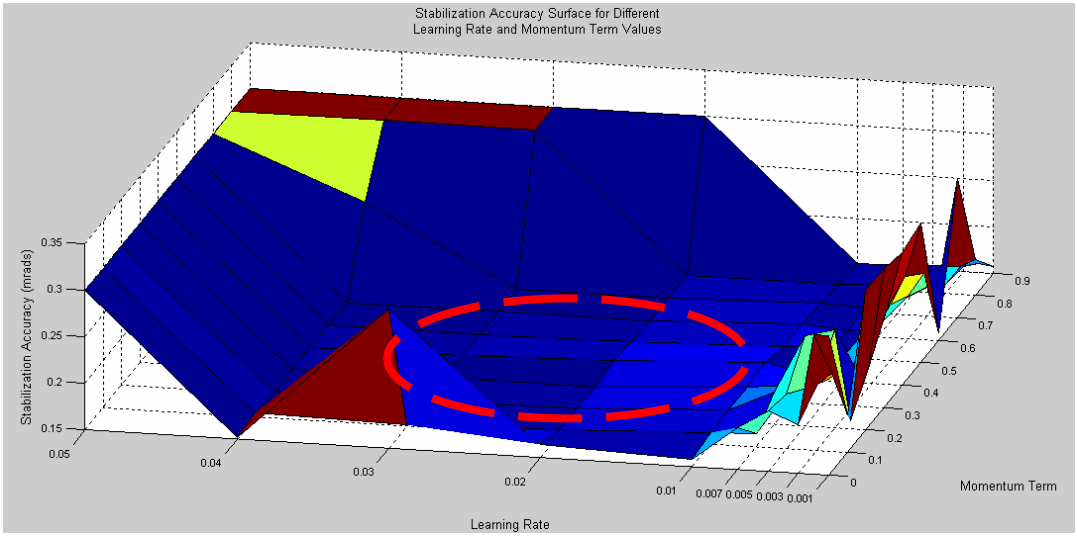

Figure 3.12 – Stabilization Accuracy Values Obtained for Different Learning Rate and Momentum Term Values, Trail 1 (0.3 mrad Shows Unstable Cases)

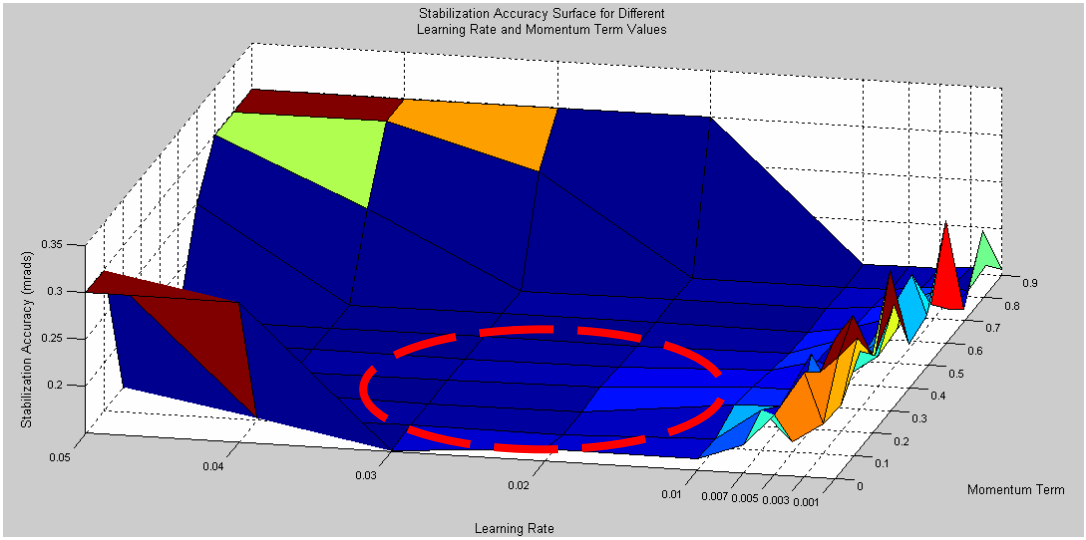

Figure 3.13 - Stabilization Accuracy Values Obtained for Different Learning Rate and Momentum Term Values, Trail 2 (0.3 mrad Shows Unstable Cases)

Now, all of the design parameters for the neural PID tuner are determined by performing computer simulations. These are;

- The network has one input, one hidden and one output layer
- All the neurons use tangent hyperbolic activation functions
- There are 10 neurons in the input layer and 20 neurons in the hidden layer
- The learning rate is $\eta = 0.025$ and momentum term is $\alpha = 0.4$
- Initial network weights and biases are random and between -0.1 and 0.1.

The results showing the performance of the neural PID tuner network with these network parameters on both traverse and elevation axes are given in chapter 5. After the design of the neural PID tuner network, a friction compensation algorithm will be proposed in the next chapter. With the compensation of the Columb friction, it is expected to increase the stabilization performance considerably.

3.3 Neural Friction Compensator

Columb friction is a natural phenomenon that always exists in mechanical systems. The aim of neural friction compensation network proposed in this section is to estimate this non-linear Columb friction characteristic and produce a correction signal, which is added to the controller output in order to eliminate the effect of Columb friction. If the compensator can apply a signal, which is the inverse of the Columb friction, the linear PID controller deals only with the linear dynamics of the physical system assuming that there is no other non-linearity in the system other than the Columb friction or other non-linearities in the system are also compensated by other methods such as by unbalance compensation given in section 3.1.

The proposed neural friction compensator with the unbalance compensation algorithm mentioned before prepares a linearized turret subsystem for the classical PID controller used for stabilization control and this PID controller gives best performance with this linearized system. This is because of the fact that linear controllers give best performance with linear systems.

In our case, turret subsystems have non-linear Columb friction, unbalance for both elevation and traverse axis. Other non-linearity may be due to the kinematics of the actuators but this non-linearity can be compensated by mathematical inverse kinematics. Therefore, the linear system assumption holds if both the non-linear unbalance and non-linear Columb friction are compensated. Of course, there will be some compensation error, which will affect the linearity of the system, but the neural PID tuner will compensate these errors.

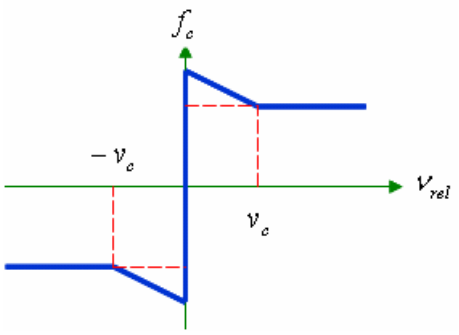

Figure 3.14 – Columb Friction Dependency on Relative Speed

The dependency of Columb friction on relative speed of the corresponding axis is shown in Figure 3.14. As seen from this figure, Columb friction is a function of relative velocity. It depends on the sign of the velocity and it opposes the motion by applying a force in the opposite direction of relative speed. Because of these facts, the proposed neural friction compensator uses the relative velocity of the physical system to estimate the Columb friction on the system using the function fitting property of neural networks.

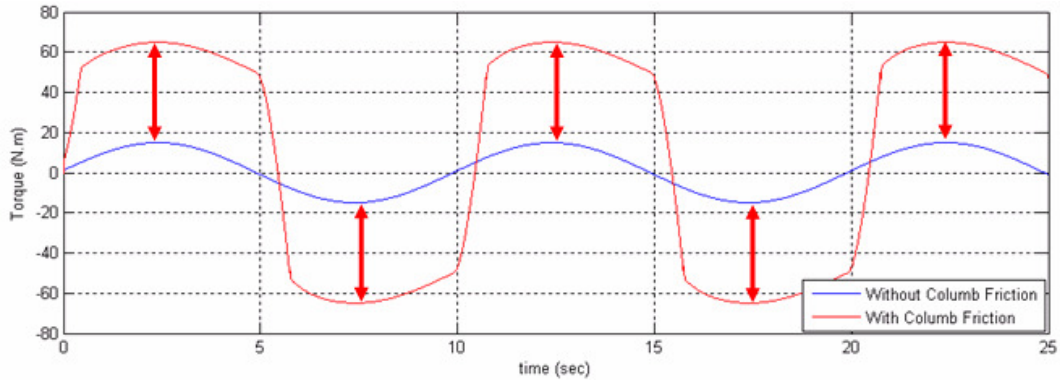

Figure 3.15 – Outputs of a PID Controller for Systems with and without Columb Friction

The torque demand output of a classical PID controller for systems with and without Columb friction is given in Figure 3.15. As seen from the figure, the output of the controller is a pure sinus for a sinus command for the system without Columb friction but sudden jumps occurs in the controller output for the system with Columb friction. The amplitude of these jumps is actually the magnitude of the Columb friction, which exists on the physical system.

The approach used in this study is to estimate the output of the PID controller, which drives a linearized virtual system emulation of the real system in order to train the neural network, which estimates the function relationship between the relative speed and Columb friction. For this reason, the linear dynamics of the physical system is identified in the regions where the Columb friction effects are minimized and this identification is used for the emulation of a virtual linear system. This emulated system is driven by a PID controller, which has the same

initial conditions and controller gains with the PID controller driving the real physical system.

Due to the nature of the Columb friction, it is a constant torque opposing the motion of the system for large relative velocities. In this region, the non-linear effect of the Column friction is minimum since it is only a constant torque. Therefore, the linear system dynamics is identified in this region with high relative velocities. On the other hand, the friction non-linearity is most active for low relative velocities because in this region relative speed changes its sign resulting in the change of friction torque. Therefore, the network for the estimation of the Columb friction is trained in this region with low relative velocities. The flow chart of the proposed friction compensation algorithm is given in Figure 3.16. As seen from this figure, when the relative velocity is low, friction compensation network is trained using the output of the PID controller and linear system emulator. For the large velocities, the training of this friction compensation network stops and training of the linear system identifier starts.

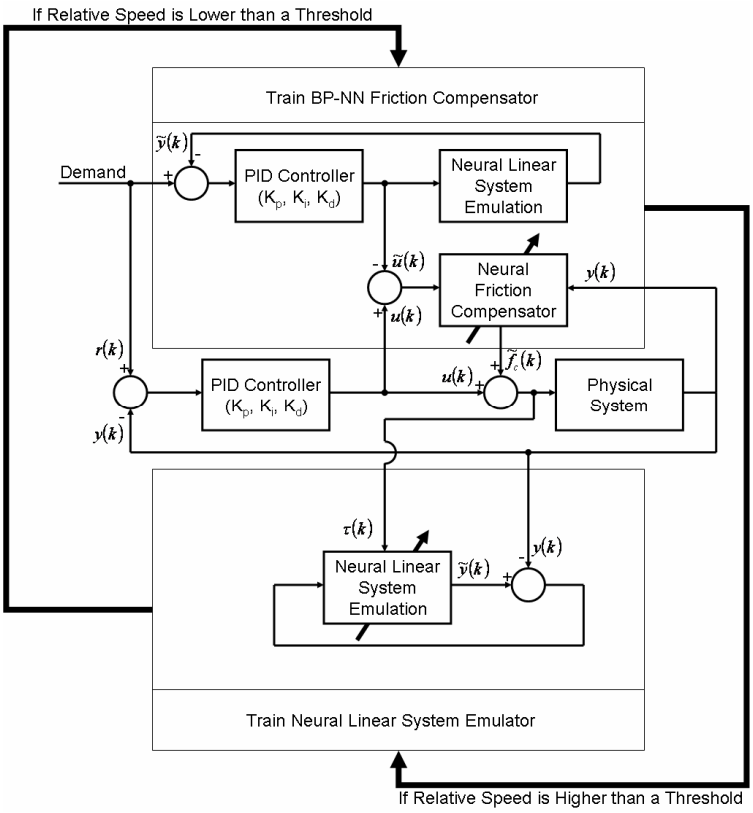

Figure 3.16 – Flowchart of the proposed neural friction identifier and compensator

After this point, the mathematical details of the neural linear system identifier network will be given in the next section. After this mathematical interpretation, the mathematical derivations of the neural friction compensation network will be given. Finally, the design of the parameters of these networks will be given using the mathematical model of a battle tank.

3.3.1 Neural Linear System Identifier Network and System Emulator

The proposed neural system emulator uses the linear system identification of the physical system in order to approximate this system with a linearized virtual system. The block diagram of this algorithm is shown in Figure 3.17. As seen from the figure, the estimated system linear system parameters are used by the linear system emulator for the virtual linear system and this virtual system is controller using a linear PID controller in order to estimate the PID output of the system if there were no Columb friction in the system.

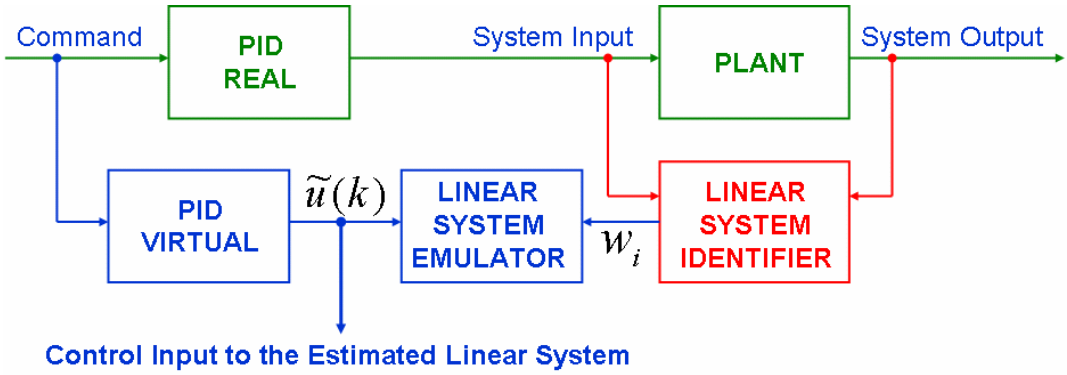

Figure 3.17 – Block Diagram for the Estimation of the Linear Dynamics of the System and Emulation of the Virtual Linear System

For the estimation of the linear dynamics of the real physical system, the discrete time linear system representation given as $y(k) = \dfrac{a_0 + a_1 z^{-1} + \cdots + a_r z^{-r}}{1 + b_1 z^{-1} + \cdots + b_r z^{-r}} u(n)$ is used where $u(k)$ is the input to the physical system, $y(k)$ is the system response, z is the time delay operator and n is the time index. The given equality can be rewritten as;

$$y(k)=a_0u(k)+a_1u(k-1)+\cdots+a_ru(k-r)-b_1y(k-1)-b_2y(k-2)-\cdots-b_ry(k-r) \quad (3.27)$$

Using this equality and the function fitting property of neural networks, the system response at the current time $y(k)$ can be estimated by $\tilde{y}(k)$ with the following linear neural network.

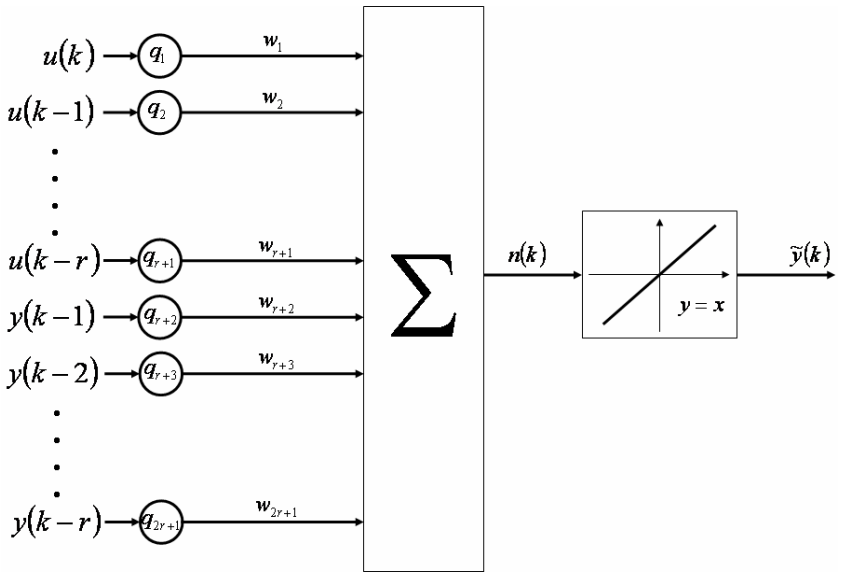

Figure 3.18 - Architecture of the neural linear system emulator

The network weights w_1, w_2, …, w_{r+1} correspond to the weights a_0, a_1, …, a_r in the same order and the network weights w_{r+2}, w_{r+3}, …, w_{2r+1} correspond to the remaining weights b_1, b_2, …, b_r in the same order.

The network weights are updated using the gradient decent technique. Using this technique, the network is trained using the following weight update equation;

$$w_i(k+1)=w_i(k)-\eta\frac{\delta J(k)}{\delta w_i(k)} \quad \text{where} \quad J(k)=\left(y(k)-\tilde{y}(k)\right)^2 \quad (3.28)$$

The learning rate η determines the speed of learning but using high values for this parameter may cause instability in the process. Further expanding the weight update equation, the following equality is obtained;

$$w_i(k+1)=w_i(k)-2\eta\left(\frac{\delta y(k)}{\delta w_i(k)}-\frac{\delta \tilde{y}(k)}{\delta w_i(k)}\right)\left(y(k)-\tilde{y}(k)\right) \quad (3.29)$$

Since the present physical system output $y(k)$ does not depend on the network weights, the term $\frac{\delta y(k)}{\delta w_i(k)}$ is equal to zero. Therefore, the network weight update equation can be rewritten as;

$$w_i(k+1) = w_i(k) + 2\eta \frac{\delta \tilde{y}(k)}{\delta w_i(k)} (y(k) - \tilde{y}(k)) \tag{3.30}$$

Using equation 3.27, the term $\frac{\delta \tilde{y}(k)}{\delta w_i(k)}$ is equal to the corresponding input to the node i which is denoted as $q_i(k)$. Therefore, the network weight update equation is finally written as;

$$w_i(k+1) = w_i(k) + 2\eta q_i(k)(y(k) - \tilde{y}(k)) \tag{3.31}$$

The network weights are the estimated system parameters $a_1, a_2, \ldots, a_r$ and $b_1, b_2, \ldots, b_r$ and these parameters are used for the emulation of a virtual linear system.

The mathematical details for the linear system identification network and linear virtual linear system emulation are given here. The mathematical details of the friction compensation network are given in the next section using the results obtained in this section.

3.3.2 Mathematical Derivations for the Friction Compensator Network

The friction compensation network used for the estimation of the Columb friction is a multilayered neural network trained using back-propagation learning algorithm. The schematic view of the neural friction compensator is given below in Figure 3.19. The output of this network is the estimated Columb friction and inverse sign of this output is added to the output of the stabilization controller in order to compensate the Columb friction existing on the physical system.

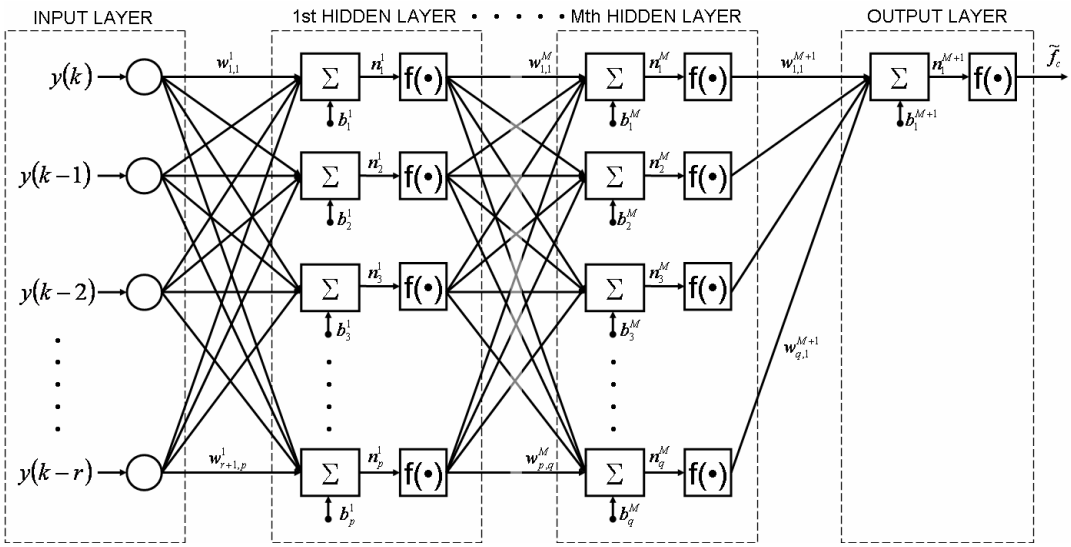

Figure 3.19 - Architecture of the neural friction identifier and compensator

Similar to the neural PID tuner network, the proposed neural friction compensator aims to minimize an error signal. This time, this error signal is the difference between the output of the PID controller that controls the real physical system and the output of the PID controller that controls the virtual linear system emulation.

The network searches optimum compensation minimizing the cost function given below as a function of the aforementioned error:

$$J = \frac{1}{2}(u(k+1) - \tilde{u}(k+1))^2 = \frac{1}{2}(\tilde{e}(k+1))^2 \tag{3.32}$$

where, $u(k+1)$: Output of the PID controller driving the physical system

$\tilde{u}(k+1)$: Output of the PID controller driving the virtual system

$\tilde{e}(k+1)$: Emulation error

The network weights are updated using the following network weight update equation that incorporates the cost function given in eqn.3.32. This update equation is the same as the weight update equation of the neural PID tuner and therefore the remaining approach is very similar to the one derived for the neural PID tuner.

$$w_{j,i}^{n}(k+1) = w_{j,i}^{n}(k) + \Delta w_{j,i}^{n}(k+1) \text{ where } \Delta w_{j,i}^{n}(k+1) = -\eta \frac{\delta J}{\delta w_{j,i}^{n}(k)} + \alpha \Delta w_{j,i}^{n}(k) \quad (3.33)$$

Here again η represents the learning rate and α represents the momentum term and determination of these parameters will be given in the next section. The term $\Delta w_{j,i}^{n}(k+1)$ is expanded to have the following equality.

$$\frac{\delta J}{\delta w_{j,i}^{n}(k)} = \frac{1}{2} \frac{\delta \left[\left(u(k+1) - \tilde{u}(k+1) \right)^2 \right]}{\delta w_{j,i}^{n}(k)} = -\tilde{e}(k+1) \frac{\delta u(k+1)}{\delta w_{j,i}^{n}(k)} \quad (3.34)$$

In this expansion, the partial derivative of $\tilde{u}(k+1)$ with respect to the network weight $w_{j,i}^{n}(k)$ is zero because the output of the PID controller driving the linear system model is not dependent on the friction compensation term and therefore it is not a function of the network weights of the friction compensation network.

In order to relate this partial derivative term to the network parameters, the variable $u(k+1)$ should be further expanded. The output of the PID controller was introduced in the chapter 3 as;

$$u(k) = u(k-1) + K_p \left(e(k) - e(k-1) \right) + K_i e(k) + K_d \left(e(k) - 2e(k-1) + e(k-2) \right) \quad (3.35)$$

From this equation, it is clear that the variable u is a function of its previous values, proportional, integral and derivative gains of the PID controller, error signal between the command and feedback signals and its previous values. Therefore, there is a need to examine the relationships between the terms given above and the network weights of the compensator.

First, the previous value of the output of the PID controller is examined. Since the PID controller output at the last time step can only be affected by the outputs of the friction compensator before the last step, the term $u_{real}(k)$ is not a function of $w_{j,i}^{n}(k)$. Therefore, the term $\frac{\delta u(k)}{\delta w_{j,i}^{n}(k)}$ is equal to zero.

Secondly, the error terms are examined. The error between the command and the feedback signals at the last time step is affected by the friction compensator outputs at time steps before this last time step. Therefore, the terms $\frac{\delta e(k)}{\delta w_{j,i}^{n}(k)}$ and $\frac{\delta e(k-1)}{\delta w_{j,i}^{n}(k)}$ are identically zero but the term $\frac{\delta e(k+1)}{\delta w_{j,i}^{n}(k)}$ may be not be zero.

Knowing these facts, the partial derivative term $\frac{\delta u(k+1)}{\delta w_{j,i}^{n}(k)}$ can be rewritten as;

$$\frac{\delta u(k+1)}{\delta w_{j,i}^{n}(k)} = \left(K_p + K_i + K_d\right)\frac{\delta e(k+1)}{\delta w_{j,i}^{n}(k)} = \left(K_p + K_i + K_d\right)\frac{\delta\left(r(k+1) - y(k+1)\right)}{\delta w_{j,i}^{n}(k)} \tag{3.36}$$

Finally, using the above equation and the fact that the user command signal is not dependent on the network weights of the friction compensator, the following equality is obtained;

$$\frac{\delta u(k+1)}{\delta w_{j,i}^{n}(k)} = -\left(K_p + K_i + K_d\right)\frac{\delta y(k+1)}{\delta w_{j,i}^{n}(k)} \tag{3.37}$$

Using the equation given above, the partial derivative of the performance index relative to the network weights can be written as;

$$\frac{\delta J}{\delta w_{j,i}^{n}(k)} = \left(K_p + K_i + K_d\right)\frac{\delta y(k+1)}{\delta w_{j,i}^{n}(k)}\tilde{e}(k+1) \tag{3.38}$$

The output of the physical system y is a function of the input denoted by τ which is the sum of the friction compensator output and that of the PID controller driving the physical system. Therefore, the following equality can be written.

$$y(k+1) = f(\tau) = f\left(u(k) + \tilde{f}_c(k)\right) \tag{3.39}$$

Similar to the approach used in the derivations of the PID tuner algorithm, the partial derivative of the system output $y(k+1)$ with respect to the friction compensator output $\tilde{f}_c(k)$ can be written as;

$$\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)} = \left|\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right| .sign\left(\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right) \tag{3.40}$$

Using the above equation and chain rule expansion of equation 3.37, the following equality can be obtained;

$$\frac{\delta J}{\delta w_{j,i}^n(k)} = \left(K_p + K_i + K_d\right)\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\frac{\delta \tilde{f}_c(k)}{\delta n_i^n(k)}\frac{\delta n_i^n(k)}{\delta w_{j,i}^n(k)}\tilde{e}(k+1) \tag{3.41}$$

$$\frac{\delta J}{\delta w_{j,i}^n(k)} = \left(K_p + K_i + K_d\right)\left|\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right| .sign\left(\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right)\frac{\delta \tilde{f}_c(k)}{\delta n_i^n(k)}\frac{\delta n_i^n(k)}{\delta w_{j,i}^n(k)}\tilde{e}(k+1) \tag{3.42}$$

In this equation, because of the structure of the PID controller, the controller gains K_p, K_i and K_d are all positive. The term $\left|\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right|$ is obviously also a positive quantity. Therefore, we can join the summation of these controller gains with the absolute value term to obtain a positive quantity denoted as β. After these simplifications, the equation can be rewritten as;

$$\frac{\delta J}{\delta w_{j,i}^n(k)} = \beta .sign\left(\frac{\delta y(k+1)}{\delta \tilde{f}_c(k)}\right)\frac{\delta \tilde{f}_c(k)}{\delta n_i^n(k)}\frac{\delta n_i^n(k)}{\delta w_{j,i}^n(k)}\tilde{e}(k+1) \tag{3.43}$$

The back propagation common term can be defined in this case as follows;

$$\rho_i^n(k) = \frac{\delta \tilde{f}_c(k)}{\delta out_i^n(k)}\frac{\delta out_i^n(k)}{\delta n_i^n(k)} = \mathrm{f}_n'\left(n_i^n(k)\right) \tag{3.44}$$

This common term can be obtained for previous network layers using the following equalities,

$$\rho_j^{n-1}(k) = \frac{\delta \tilde{f}_c(k)}{\delta out_j^{n-1}(k)}\frac{\delta out_j^{n-1}(k)}{\delta n_j^{n-1}(k)} \tag{3.45}$$

$$\rho_j^{n-1}(k) = \left(\sum_i \frac{\delta \tilde{f}_c(k)}{\delta out_i^n(k)}\frac{\delta out_i^n(k)}{\delta n_i^n(k)}\frac{\delta n_i^n(k)}{\delta out_j^{n-1}(k)}\right)\frac{\delta out_j^{n-1}(k)}{\delta n_j^{n-1}(k)}$$

Further simplification of this equation results in a recursive equation given below,

$$\rho_j^{n-1}(k) = \left(\sum_i \rho_i^n(k) w_{j,i}^n(k) \right) \mathrm{f}'_{n-1}\left(n_j^{n-1}(k)\right) \tag{3.46}$$

Therefore, the network weight update equation can be rewritten as,

$$w_{j,i}^n(k+1) = w_{j,i}^n(k) - \eta \left(\beta.\tilde{e}(k+1).sign\left(\frac{\delta y(k+1)}{\delta u(k)} \right).\rho_i^n(k).out_j^{n-1}(k) \right) + \alpha.\Delta w_{j,i}^n(k) \tag{3.47}$$

Since the combined term β is a positive term, we can join β with the learning rate η in order to have a single learning rate term η'. With this little modification, the network weight update equation is finally written as;

$$w_{j,i}^n(k+1) = w_{j,i}^n(k) - \eta' \left(\tilde{e}(k+1).sign\left(\frac{\delta y(k+1)}{\delta u(k)} \right).\rho_i^n(k).out_j^{n-1}(k) \right) + \alpha.\Delta w_{j,i}^n(k) \tag{3.48}$$

After giving the mathematical details of the friction compensation network, the design of the parameters of friction compensation network will be given in the next section.

3.3.3 Design of the Neural Linear System Identifier and Neural Friction Compensation Networks

The first step in the design of the neural friction compensator is that of the neural linear system identifier. This network identifies the linear behavior of the physical system in order to generate a virtual system emulator. This linear system emulator is controlled by the same PID controller driving the real system and the comparison of these outputs creates the errors between these two PID controllers that are used to train the neural friction compensator.

The neural linear system identifier is a single layer network. Therefore, the only things, which need an optimization, are the number of neurons in this layer, which is actually the order of the system, the learning rate, and momentum term values of the network. For this optimization, the performance criteria are the speed and the stability of the learning of the system dynamics.

For fast learning, the learning rate should be as high as possible but after some value of the learning rate, the learning process may become unstable. In addition, the momentum term damps the learning process but using large values for this parameter may lead too damped learning of the network. Therefore, the optimum values for these parameters should be determined.

For the system order value, the value should be large enough to cover all the dynamics of the system but using very large values does not increase the performance of the network. In addition, large number of neurons in the network makes the real-time implementation of the network harder. Therefore, for the best performance, the optimum system order value should be determined.

For this purpose, simulations are performed for different values of system order. In these simulations, the errors between the output of the real system and the output of the system identifier are compared. This comparison is given in Figure 3.20. As seen from the figure, the performance of the neural system identifier increases with the increase of the system order. After the value 10, the decrease in the final error value stops but learning speed of the network continues to increase. However, large system order values increase the identification error while increasing the learning speed. Therefore, the system order value 10 is selected as the optimum value for the elevation axis simulations. The results for the traverse axis are very similar due to the mechanical similarity of the axis with the elevation axis and the system order 10 will be used in the traverse axis simulations.

The neural system identifier will emulate a virtual system, which is the linear approximation of the real system. For this virtual system emulator to start, the neural linear system identifier should reach to a satisfactory estimation level. In order to sense the performance level of the network, forgetting sum of the absolute values of the identification error is used. In Figure 3.21, the forgetting sum of the identification errors for different system order values are given. In these calculations, the forgetting term is used as 0.99 and which leads the sum to forget a value approximately after 5 seconds.

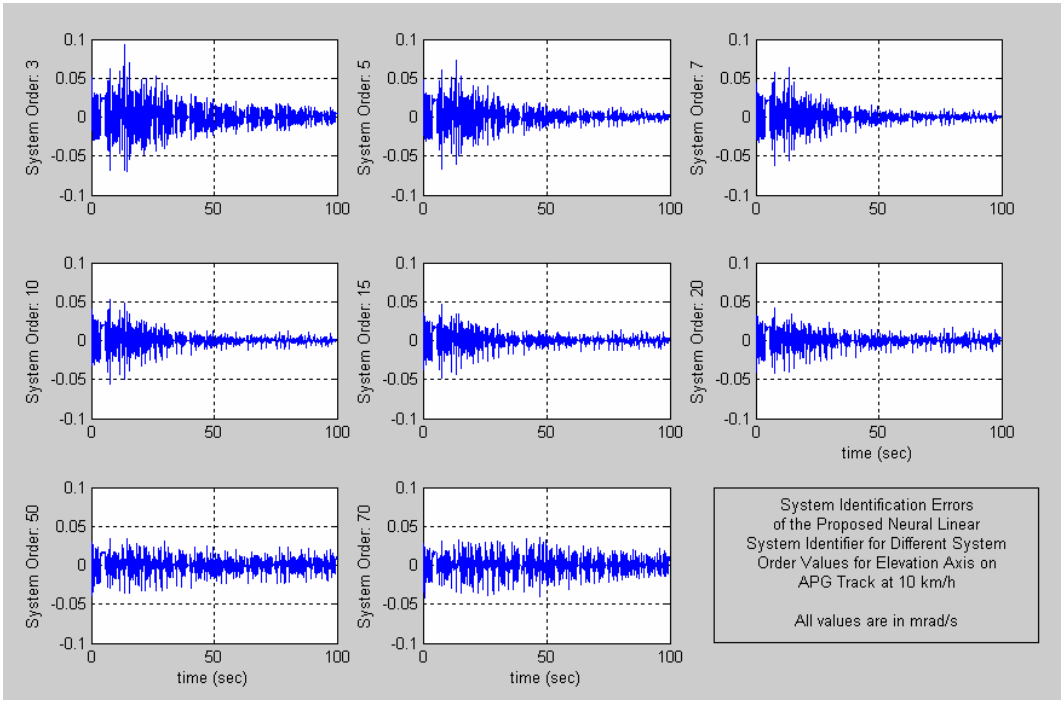

Figure 3.20 – System Identification Errors of the Proposed Neural Linear System Identifier for Different System Order Values for the Elevation Axis in APG Track at 10 km/h Speed

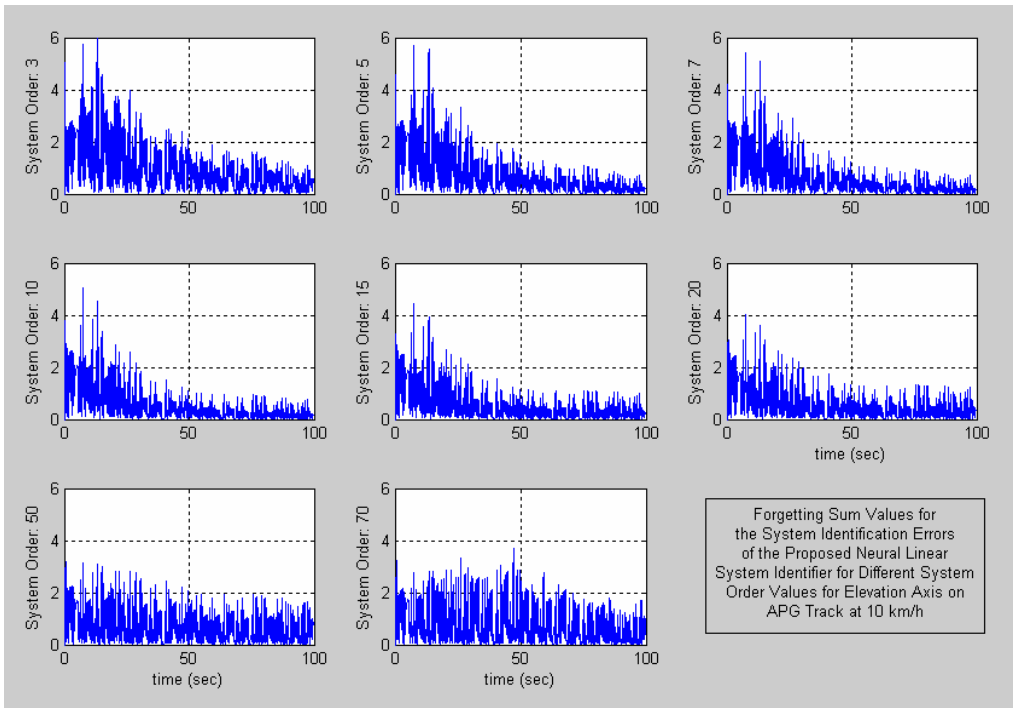

Figure 3.21 – Forgetting Sum Values for the System Identification Errors of the Proposed Neural System Identifier for Different System Order Values for the Elevation Axis on APG Track at 10 km/h Speed

Again, the selected system order 10 gives the best results in this figure. In the simulations with the neural friction compensator, the neural system emulator will be activated after the forgetting sum of the identification errors falls below 1. This approximately takes 50 seconds. Therefore, the neural system emulator will be activated after 50 seconds of the simulation start.

As mentioned many times before, the system identifier aims to identify the linear behavior of the real system. For the system in concern, the only non-linearity left is the Columb friction. In order to avoid this non-linearity, the neural identifier is trained only for relative speed of the axis greater than a threshold. Since the Columb friction shows the non-linearity in the low speed region, the network is not trained in low velocity region. In these simulations and the simulations after this point, this speed threshold is used as 1 deg/s for both axes. This value is selected as small as possible for good linear identifier training and as large as possible for good neural friction compensator training since the friction compensator is trained in this low velocity region.

The results given above is obtained for the learning rate value $\eta = 0.01$ and momentum term value $\alpha = 0.5$. What happens to the identification errors with different values of η and α? The answer to this question is given with simulations given in Figure 3.22, which shows that increasing the learning rate, increase the learning speed but again the identifier gives the best performance with the system order value 10. This is simply because of the system dynamics. The system dynamics or the system order does not change with the learning rate of the identifier.

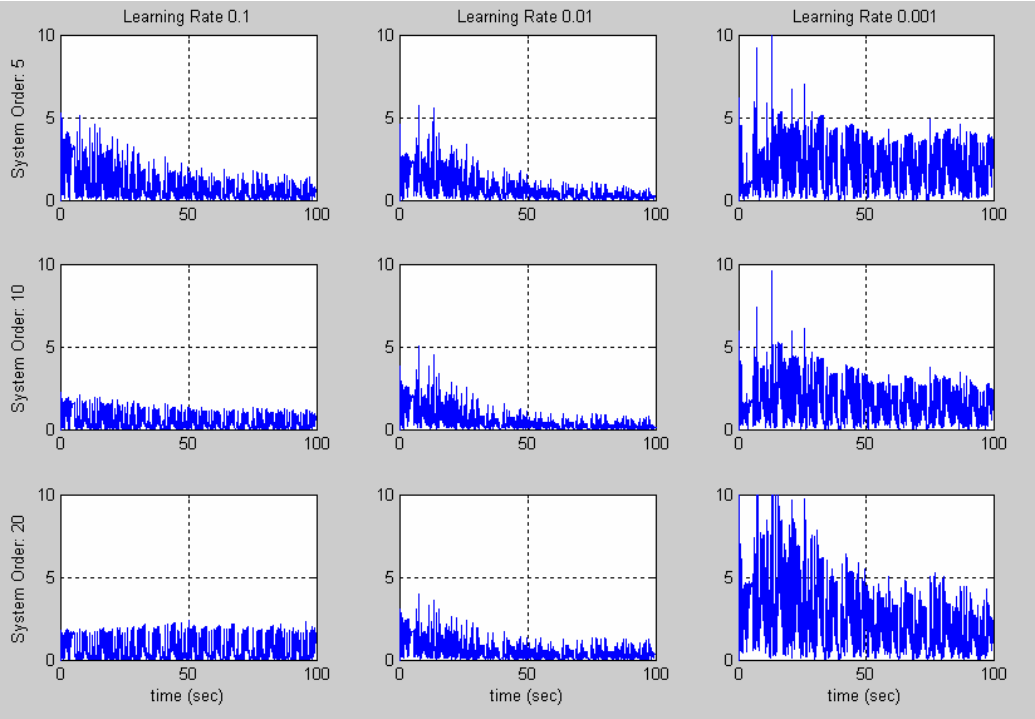

Figure 3.23 – Forgetting Sum of Identification Errors for Different System Orders and Different Learning Rates for the Elevation Axis on APG Track at 10 km/h

The last parameters remaining to be optimized are the learning rate and momentum term values. These values are optimized by computer simulations by trying different values and comparing the forgetting sum of the resulting identification errors.

The final forgetting sum values of the identification errors are given in

Table **4** for different values of the learning rate and momentum term for 100 seconds long simulations. The results shown by 'U' means unstable learning of the system dynamics. In this case, the identification starts with low values but increases with time.

The results show that, low values of the learning rate results in high identification error. In these cases, increasing the momentum term increases the identification performance. On the other hand, high values of learning rate with small momentum term values, starts with relatively high identification errors and learning becomes unstable for high values of momentum term.

Table 4 – Final Forgetting Sum Values of the System Identification Errors for Different Values of the Learning Rate and Momentum Term (100 seconds Simulation)

η \ α	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
0.001	4	4	4	3.5	3.3	3.0	2.5	2.2	1.6	1.0
0.003	2.5	2.5	2.0	1.8	1.6	1.4	1.2	1.0	0.9	0.8
0.006	1.4	1.3	1.2	1.1	1.0	1.0	1.0	0.9	0.9	1.0
0.01	1.0	1.0	0.9	0.9	0.9	0.8	0.8	0.8	0.9	1.3
0.03	0.8	0.8	0.8	0.8	0.8	0.9	1.0	1.1	1.3	U
0.05	0.9	0.9	1.0	1.0	1.1	1.2	1.2	1.4	1.6	U
0.07	1.0	1.0	1.0	1.1	1.1	1.2	1.4	1.5	U	U
0.1	1.1	1.1	1.1	1.2	1.3	1.3	1.4	1.8	U	U
0.2	1.2	1.3	1.3	1.4	1.6	1.8	U	U	U	U
0.4	1.6	1.8	1.8	1.8	U	U	U	U	U	U

For medium learning rate values, increasing the momentum term increases the identification performance up to some point. Further increase of the momentum term for these cases results in a decrease in the identification performance.

The best value obtained in all of the simulations is 0.8 and this value is obtained only for learning rate values $\eta = 0.01$ and $\eta = 0.03$. For these two learning rate values, although the final values of the forgetting term is the same for some values of the momentum term, the identification error decreases faster for $\eta = 0.03$ and for momentum term value $\alpha = 0.4$. Therefore these values will be used in the computer simulations after this point.

The neural system identifier, which will be used for the neural friction compensator design, is completed and system order, learning rate and momentum

term values are determined. After this point, the design of the neural friction compensator will be given.

The proposed neural friction compensator is a multi-layered neural network and back propagation is used to train the network. The network uses the relative velocity and its previous values of the corresponding axes as the inputs and outputs the estimated Columb friction of the system. The network is trained based on the error between the PID controller output of the real system and the PID controller output of the virtual system emulation.

Since the network is multilayered, the first thing is the decision of the number of layers. The system has an input and output layer and n-many hidden layers. For different number of hidden layers and other network parameters, there are lots of combinations and there may be multiple optimum points. Therefore, simulations will be performed with the network with only one hidden layer and after the completed design, the effect of the number of hidden layers will be analyzed.

The network has only one neuron in the output layer because the network only outputs the estimated Columb friction. In the input layer, increasing the number of neurons gives damping to the network and the estimated friction value is applied to the system with a considerable delay. On the other hand, small number of neurons in the input layer makes the network too sensitive to the changes in the resolver speed inputs. Therefore, the number of neurons in the input layer should be carefully selected.

Since there are many design parameters of the network and there are many local optimum points of these parameters, the design approach is to fix the number of neurons in the input layers and in the hidden layer. Keeping in mind the points discussed above, the number of neurons in the input layer is selected as 10 and number of neurons in the hidden layer is selected as 5. These values are selected as small as possible in order to decrease the computational complexity of the network.

Therefore, there remains two design parameters to be determined, namely the learning rate and momentum term. The optimum values of these parameters are searched for the network with;

- An input layer with 10 neurons
- An output layer with 1 neuron
- One hidden layer with 5 neurons
- Hyperbolic tangent activation function for all neurons

For this purpose, computer simulations are performed for different values of the learning rate and momentum term and the estimation performance of the proposed neural friction compensator is compared.

Table 5 – Percentage Error Values of Estimated Columb Friction Value for the Elevation Axis (U: Unstable Learning, N: Insufficient Learning)

α \ η	0.0001	0.0003	0.0007	0.001	0.003	0.007	0.01
0.0	N	N	N	%17.3	%14.0	%18.7	U
0.2	N	N	%14.7	%16.7	%16.7	%15.3	U
0.4	N	N	%17.3	%17.3	%14.0	%62.0	U
0.6	N	N	%16.0	%12.7	%15.3	U	U
0.8	N	%17.3	%12.7	%17.3	%14.7	U	U

The results obtained for different values of the learning rate and momentum term values showing the different percentage error of the estimated Columb friction value for the elevation axis is given in

Table **5**. For low values of the learning rate, the learning is so slow that the network cannot learn the friction characteristics in 250 seconds of simulation. On the other hand, for large values of learning rate, the learning process is so fast that the learning process becomes unstable and the network converges to wrong values.

From this table, the learning rate value $\eta = 0.003$ gives the best performance. Although the lowest error is obtained using learning rate value as $\eta = 0.0007$ and momentum term value $\alpha = 0.8$, the network learns the friction characteristics with $\eta = 0.003$ and $\alpha = 0.4$ faster than the parameters with lowest error. Therefore, the network parameters $\eta = 0.003$ and $\alpha = 0.4$ are selected as the optimum parameters for the neural friction compensator.

For the previous simulations, the network has only one hidden layer. In order to see the effect of the number of hidden layers on the network performance, simulations are performed for two and three number of hidden layers. However, the simulation results show that addition of hidden layers to the network slows down the learning process. In order to increase the speed of learning, learning rate must be high for networks for multiple hidden layers. Again for the sake of simplicity and for minimum computational cost, one hidden layer is the optimum number for the neural friction compensator.

The last thing to investigate is the effect of number of neurons in the input and hidden layer on the system performance. In order to see this effect, again simulations are performed for different number of neurons in the input and hidden layer. The results of these simulations are given in Table 6.

According to these results, the best performance is obtained with 5 neurons in the input layer and 5 neurons in the hidden layer considering the maximum error value. However, if rise time is considered, these parameters are not the optimum ones. The network with 15 neurons in the input layer and 3 neurons in the hidden layer gives the best performance in both settling time and steady state error. In addition, the rise time and maximum error values are not so high for these values. Therefore these values can be selected as the optimum ones. By doing this, we changed the number of neurons in the input and hidden layers and obtained better performance. In order to seek for parameters, which will result with the best performance, simulations can be performed with these number of neurons with different learning rates, momentum terms and with different number of layers but the performance obtained as a result of the current simulations are expectable.

Table 6 – Simulation Results for Different Numbers of Neurons in the Input and Hidden Layer for $\eta = 0.003$ and $\alpha = 0.4$ (U: Unstable Learning)

# In. Neurons	**5**	**10**	**15**	**5**	**10**	**15**	**5**	**10**	**15**
# Hid. Neurons	**3**	**3**	**3**	**5**	**5**	**5**	**10**	**10**	**10**
Max.Error	%15.3	%12.0	%10.7	%8.0	%16.7	%18	%9.3	U	U
Rise Time	28sec	45sec	32sec	32sec	28sec	38sec	32sec	U	U
Settling Time	125sec	76sec	52sec	58sec	55sec	100sec	50sec	U	U
Steady State Error	%9.3	%8.0	%6.7	%8.0	%8.0	%10.0	%8.0	U	U

As a consequence from all the simulations obtained up to now for the neural friction compensator, the optimum network parameters are determined as;

- One input layer with 15 neurons
- One hidden layer with 3 neurons
- Learning rate $\eta = 0.003$
- Momentum term $\alpha = 0.4$
- Tangent hyperbolic activation function for all neurons
- Random initial weights between -0.0001 and 0.0001

The Columb friction, which is applied to the system, and the estimated friction characteristic are given in Figure 3.24. As seen from the figure, the estimated friction is slightly greater than the actual one. This is simply because of the very small delay in the estimated friction. The estimated friction changes its sign a little time after the actual friction direction change. Therefore, in order to reduce this

error, the neural friction compensator applies a slightly higher friction torque to the system.

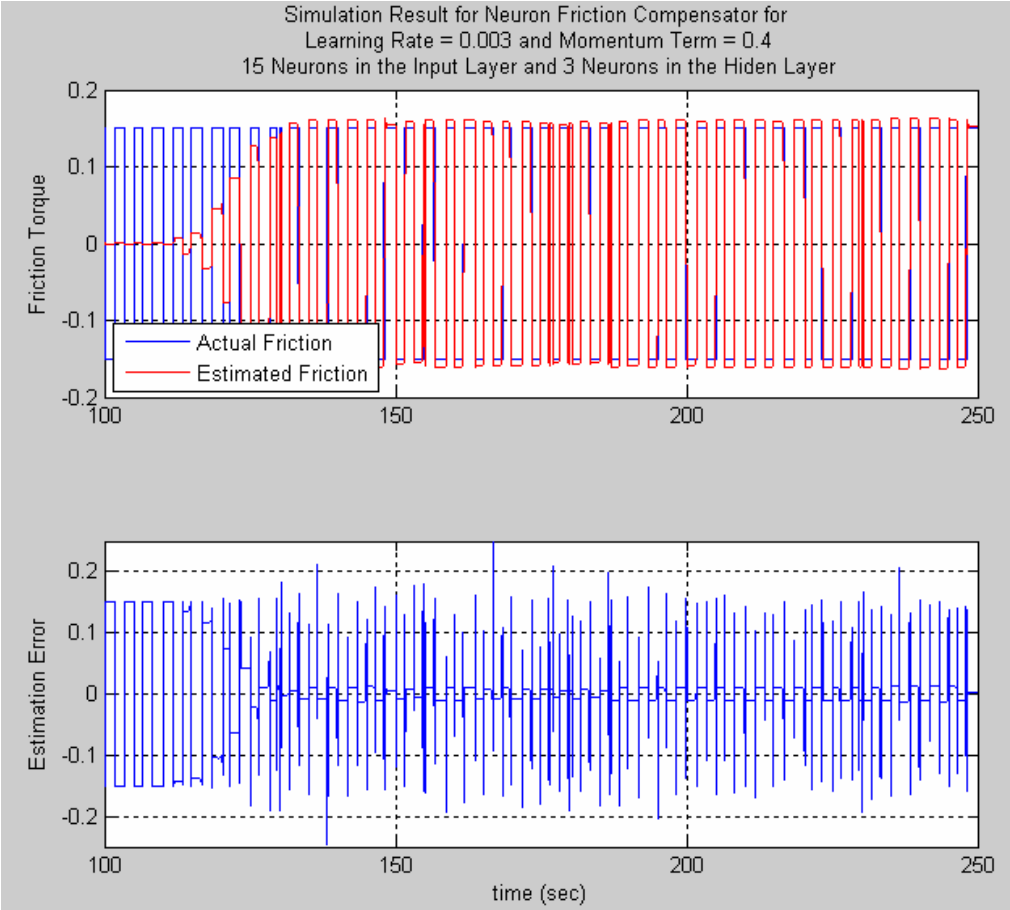

Figure 3.24 – Simulation Results for Neural Friction Compensator for Learning Rate $\eta = 0.003$, momentum term $\alpha = 0.4$, 15 Neurons in the Input Layer and 3 Neurons in the Hidden Layer for the Elevation Axis

The results for the traverse axis are very similar because of the similarity of the mechanical structure. Therefore, the same network parameters can be used for both elevation and traverse axis.

Now, all the parameters are determined for all the networks in the proposed controller architecture. Therefore, the simulations results for elevation and traverse axis using the mathematical model of the main battle tank will be given in chapter 5 using the networks parameters determined for all networks used in this study.

CHAPTER 4

SIMULATION ENVIRONMENT AND HARDWARE EXPERIMENTAL SETUP

The proposed control architecture is applied both to a computer simulation and also a real physical system. The computer simulation is performed using the mathematical model of Leopard 1A1 main battle tank. The mathematical model is realized running MATLAB® [43], Simulink® [44] and SimMechanics® [45] in an integrated manner.

SimMechanics® is a module of Simulink® and it is used to model physical systems with functional blocks. The details of this computer program and the mathematical models will be given in the following pages.

On the other hand, hardware implementation is done on a 1/6 scale model main battle tank which is a scaled version of the British Challenger II tank and the actuation system is designed to in order to be very similar to the one in the original tank. The controller is implemented on Diamond System's Athena CPU board, which is an industrial computer with PC/104 format. Using MATLAB® and xPC® Target [47] computer programs the controller is implemented on this computer running real-time.

4.1 The Mathematical Model of a Main Battle Tank

In this study, the proposed neuro controller architecture is applied to the mathematical model of Leopard 1 main battle tank which aims being as close as to the dynamics of the real system as possible. In order to represent the whole dynamics of the system, which is excited by the proposed controller, the turret of the main battle tank, including the traverse and elevation axis, is modeled in many details. In addition, the vehicle of the tank and the ground such as APG track are modeled in order to simulate the disturbance characteristics of the real tank.

The visualization of this computer simulation is performed using the Virtual Reality Toolbox® of MATLAB®. A screen shot of the visual environment is given in Figure 4.1 below.

Most of the dynamic and kinematic parameters such as inertias and link lengths are obtained using the solid modeling program I-DEAS® and the rest of the parameters, which are stiffness of torsional springs, damping functions, activation angles of hard stops, are either estimated or found from the literature or from documents of the Leopard 1 tank.

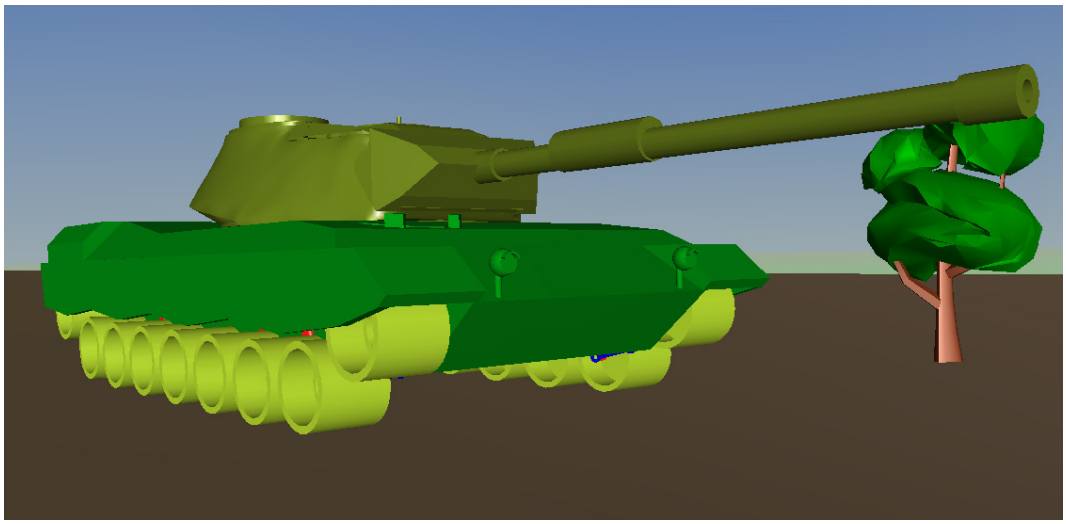

Figure 4.1 – Visualization Environment of Computer Simulation

The physical system is modeled using SimMechanics® which is a part of Simulink® by using functional blocks. In order to explain the usage of this

computer program, an example is given for the mathematical model of a four bar mechanism. The mechanism could be modeled using ground connection block to fix two link of the mechanism to a frame, revolute joint block in order to define the axis of rotations and a body blocks defining the mass, COR, COG and inertia matrix of the links. A screen shot is given for the model of a four bar mechanism in Figure 4.2.

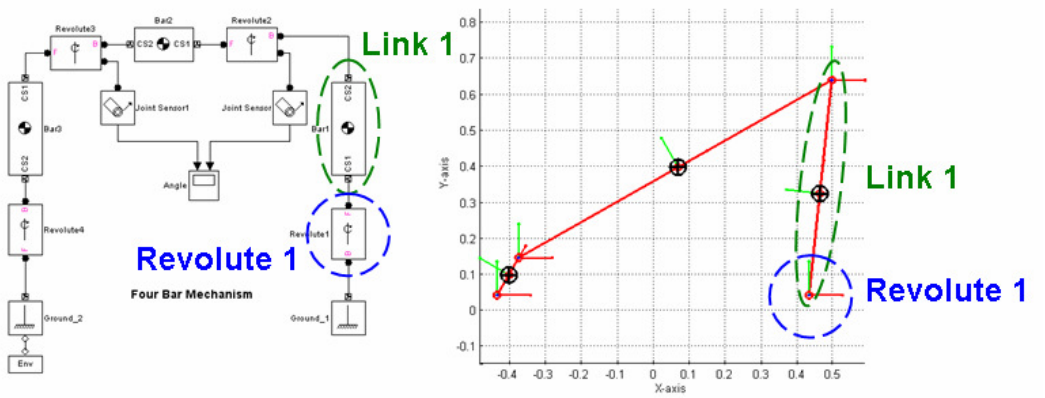

Figure 4.2 – Mathematical Modeling of a Four Bar Mechanism with SimMechanics

The mathematical model of the main battle tank is obtained using similar approaches and the details of this mathematical model will be given in the following pages. First, the mathematical model of the hull, which is the vehicle part of the main battle tank, will be represented and then the mathematical model of the turret will be explained.

The hull model consists of the following functional blocks:

- Body of the hull
- Suspension system
- Tracks
- Tire-ground interaction

The body of the hull is modeled as a lumped mass on the center of gravity (COG) of the hull body (Figure 4.3). This assumption is considered to be valid because of the high rigidity of the hull body and this approach also simplifies the

mathematical model considerably. In order to model the body of the hull, a standard “body” block of SimMechanics is used. In this block, mass, inertia matrix and joint coordinates of the hull body is defined (Figure 4.4).

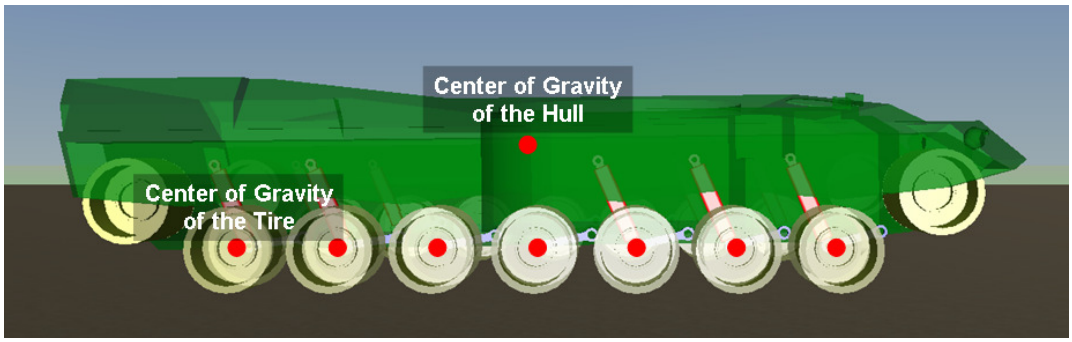

Figure 4.3 – Center of Gravity Representation of the Hull and the Wheels

The suspension system of the tank consists of wheels, torsional bars, dampers and hard stops. The wheels are modeled as lumped masses on their COGs. Again standard “Body” blocks are used to model the wheels.

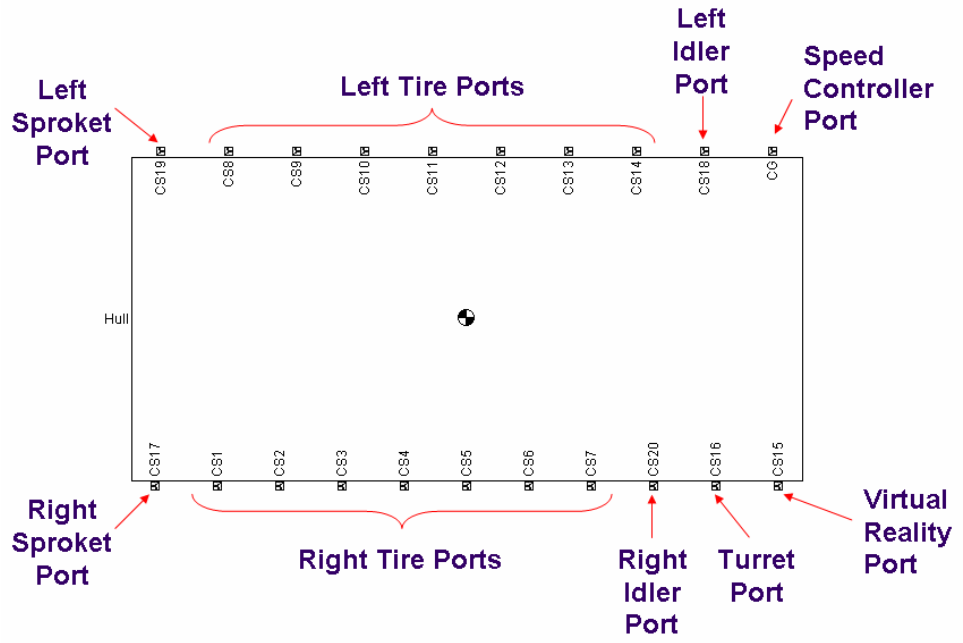

Figure 4.4 – Definition of The Joint Connections for the Hull of the Tank

For every wheel, there is always a torsional bar and a hard stop but dampers are not used for wheels, which are in the middle of the body. The schematic view of the suspension system is given in Figure 4.5.

The damper and torsional bar is assumed to be linear and therefore they could be modeled using "Spring&Damper" block of SimMechanics. The damping and spring coefficients are obtained from the maintenance documents of the real tank. The hard stop is modeled as a linear spring, which is active only for some values of the angle of the torsional bar. In other words, the hard stops apply forces to the wheels only if the angle of the torsional bar exceeds the critical angle and the applied force is only in the direction pushing the tires towards the ground.

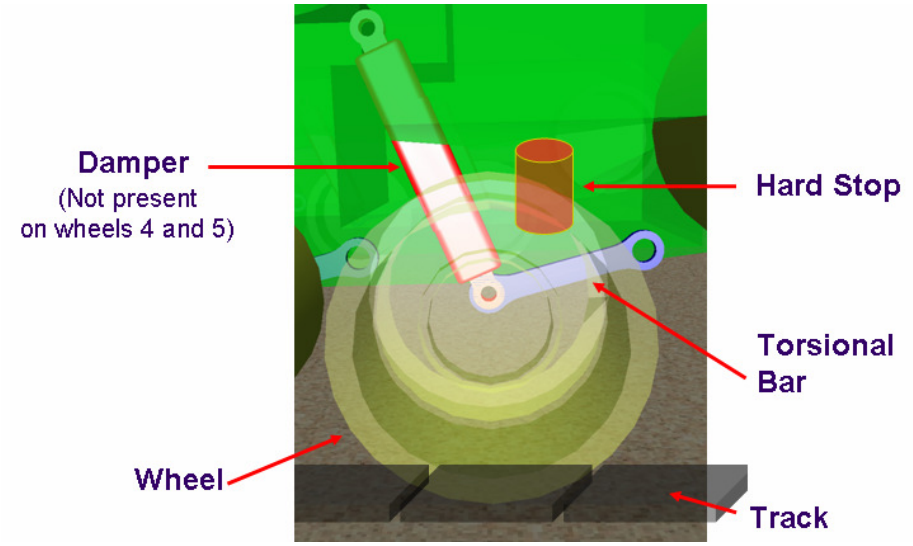

Figure 4.5 – Suspension System of Leopard 1 Tank

Tracks are modeled using the simplest track model in the literature called rubber band model. In this model, track is modeled as a rubber band with a total length of L and stiffness k encircling the wheels. If the length of the rubber band deviates from L, a force proportional to this deviation and stiffness k is applied to the tires in order to keep the length of the tracks at length L. The model is explained schematically in Figure 4.6 given below. The summation of all individual lengths l_i is kept constant by applying a force to the tires in the direction of the line l_i

shown in the figure. These direction of these forces are given for the interaction of two tires in the figure.

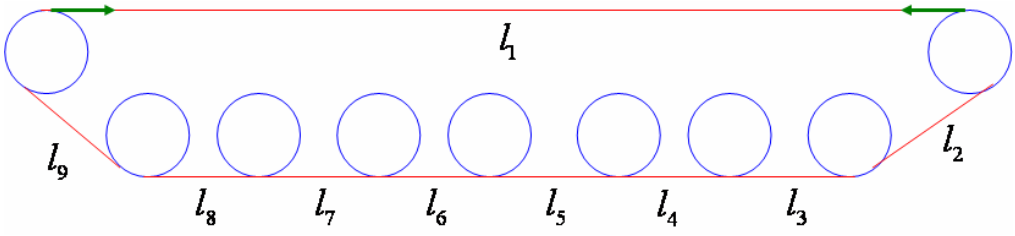

Figure 4.6 – Track Model of the Main Battle Tank (Rubber Band Model)

The tire-ground interaction is modeled as if there is a spring-damper pair between the tires and the ground. This is done by calculating the interference area of the wheel and the ground and a force is applied to the tires proportional to this calculated interference area, the stiffness of the ground, which is the spring coefficient, and the damping of the ground. The applied force to the tires are given in equation 4.1 and k is the coefficient of stiffness, b is the coefficient of damping and A is the interference are of the wheel and the ground.

$$F = k.A + b.\dot{A} \tag{4.1}$$

The turret including both the traverse and the elevation axis is modeled using a similar approach and again SimMechanics is used as the modeling tool. The mechanical structure of the traverse axis is given in Figure 4.7.

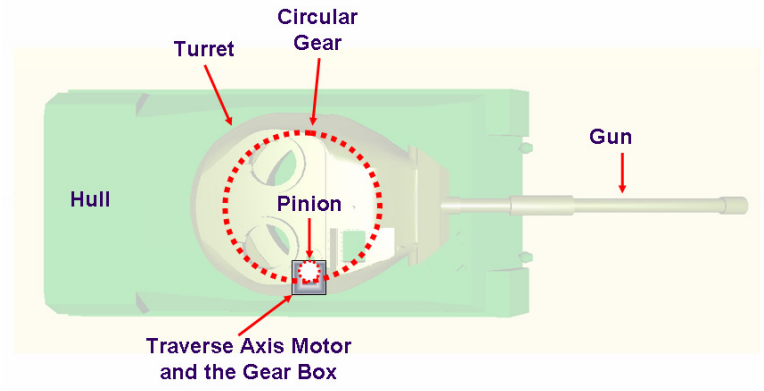

Figure 4.7 – Mechanical Structure of the Traverse Axis

As can be seen from the figure, the traverse axis is actuated by a servomotor through a gearbox. Since there is a mechanical anti-backlash mechanism in the actuation system, the backlash effects are minimized. Because of this minimization, backlash is not included in the mathematical model.

The body of the traverse axes called the turret is modeled as a lumped mass at the center of gravity of the geometry. The mass, inertia matrix and joint connection locations are obtained using the solid modeling program I-DEAS. The Columb friction between the turret and the hull of the tank is included in the mathematical model and its effect on the dynamics of the system is considerably important.

The elevation axis called the gun of the system is modeled very similar to the traverse axis because of the mechanical similarities. The mechanical structure of the gun is shown in Figure 4.8. There is a sector gear which is fixed on the gun is driven by a gear motor which is fixed on the turret of the system. There is again an anti-backlash mechanism to prevent backlash effects in the system therefore backlash is not included in the mathematical model. The dynamic and kinematic parameters of the gun are obtained from solid modeling program I-DEAS.

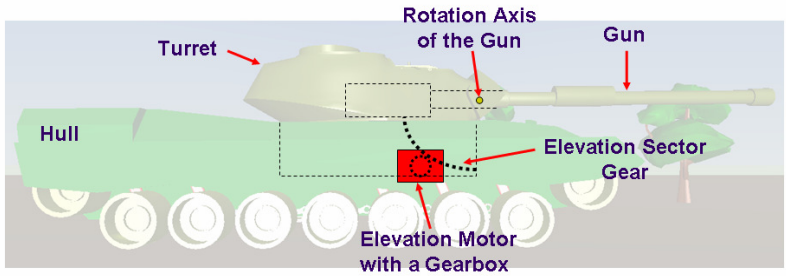

Figure 4.8 – Mechanical Structure of the Elevation Axis

The obtained mathematical model of the main battle tank can be used to simulate the real physical system in all kinds of terrains. In this study, in order to test the proposed neuro controller, APG track is used for elevation axis and sinus track is used for the traverse axis. A screen shot of the visualization environment is given in Figure 4.9 while the system passes over the APG track.

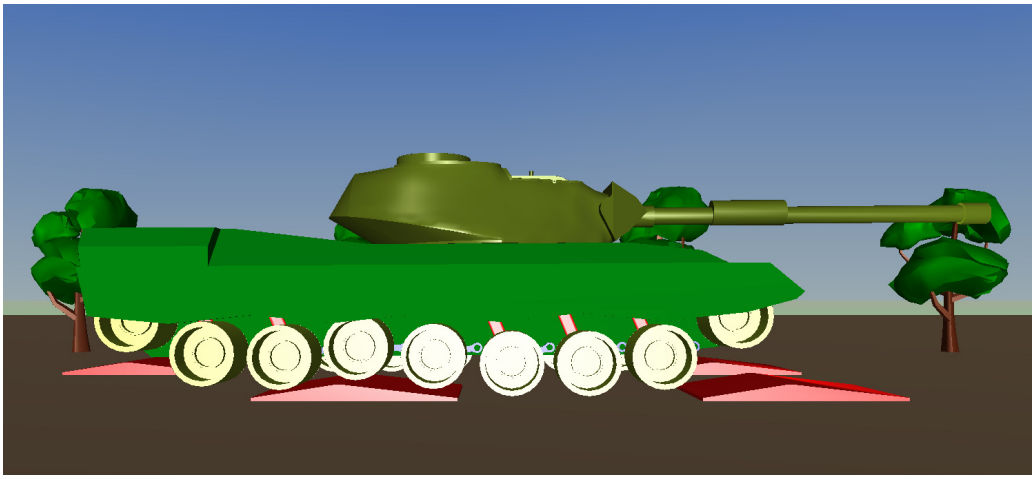

Figure 4.9 – The Main Battle Tank Passes Over the APG Track in the Model

The mathematical model obtained in this study is verified by using the data obtained from the real system but this verification is not included in this report. The details of this verification process and details of the mathematical model can be found in the reference **Error! Reference source not found.**.

4.2 Hardware Experimental Setup

The hardware experimental setup developed for this study uses a 1/6 scale Challenger 2 main battle tank prototype. The model is almost the true copy of the original tank. The prototype is tracked vehicle and the vehicle suspension system with springs and metal-plastic tracks. The turret can rotate in both traverse and elevation axis. The picture of the tank prototype is given in Figure 4.10.

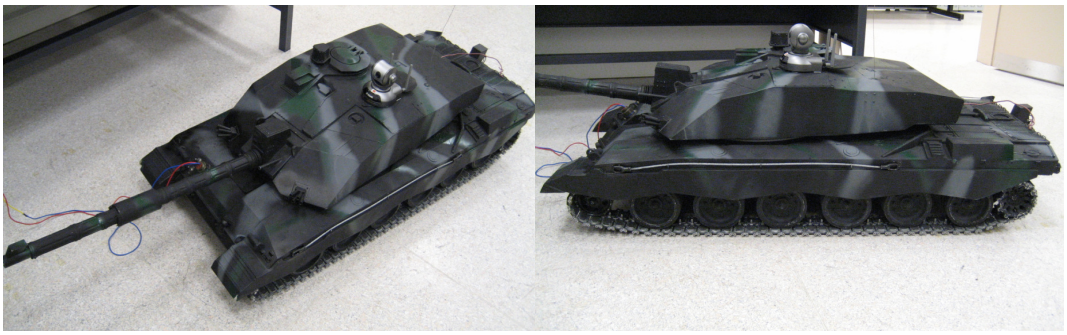

Figure 4.10 – The Experimental Setup (Challenger 2 1/6 Model Tank)

For this study, the mechanics of the traverse axis is modified in order to simulate the traverse axis actuation system of the Leopard 1 tank. A ring gear is added to the traverse axis and a gear motor is used to drive the axis. A motor controller is added to the system in order to drive the traverse axis motor. The picture of the modified traverse axis actuation system is given in Figure 4.11.

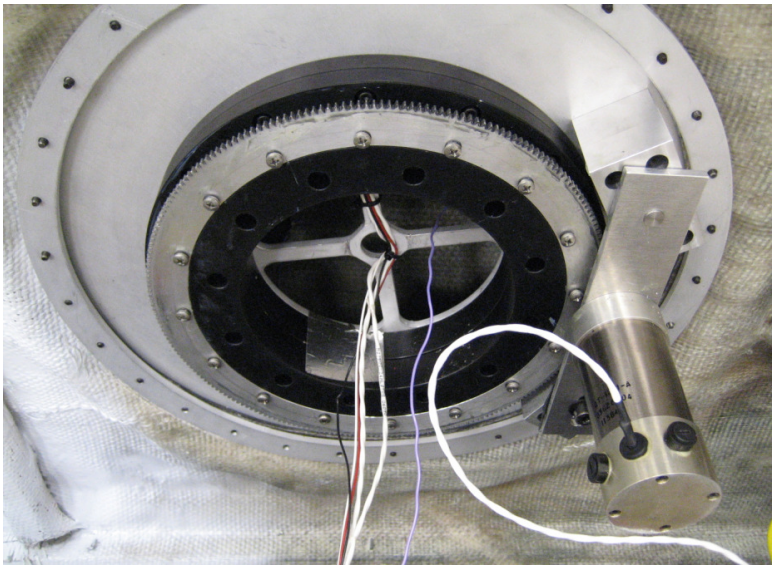

Figure 4.11 – Traverse Axis Actuation System of the Model Tank

Two rate gyroscopes are added to the system. One is fixed on the traverse axis and used as a feedback gyroscope. The other gyroscope is fixed onto the vehicle and it is used as a disturbance gyroscope. The pictures of these gyroscopes are given in Figure 4.12.

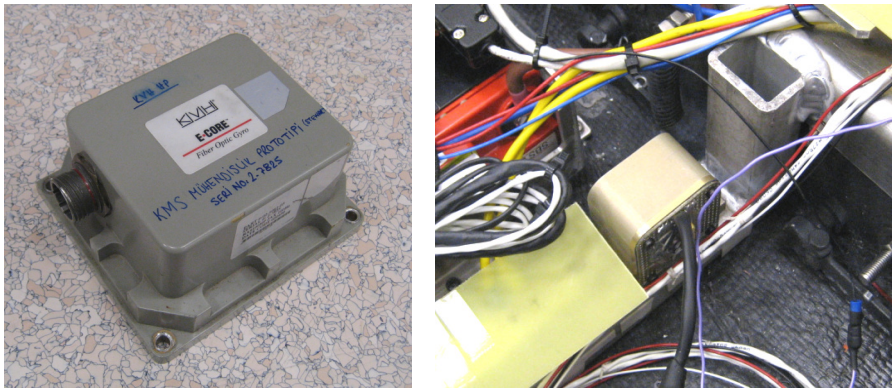

Figure 4.12 – Feedback Gyroscope (On the Left, Fiberoptic) and Disturbance Gyroscope (On the Right, MEMS)

There is also PC/104 form factor computer, which has built-in analog and digital I/O channels in the system. The producer of this computer is Diamond Systems Inc. and the type number is the ATHENA-660MHz. This computer is used as the target computer of xPC target application. The computer reads the analog inputs from the feedback and disturbance gyroscopes and produces an analog output to the motor driver. The controller is implemented on this computer and the computer can run the algorithms at 1000 Hz sampling frequency. The picture of the control computer is given in Figure 4.13.

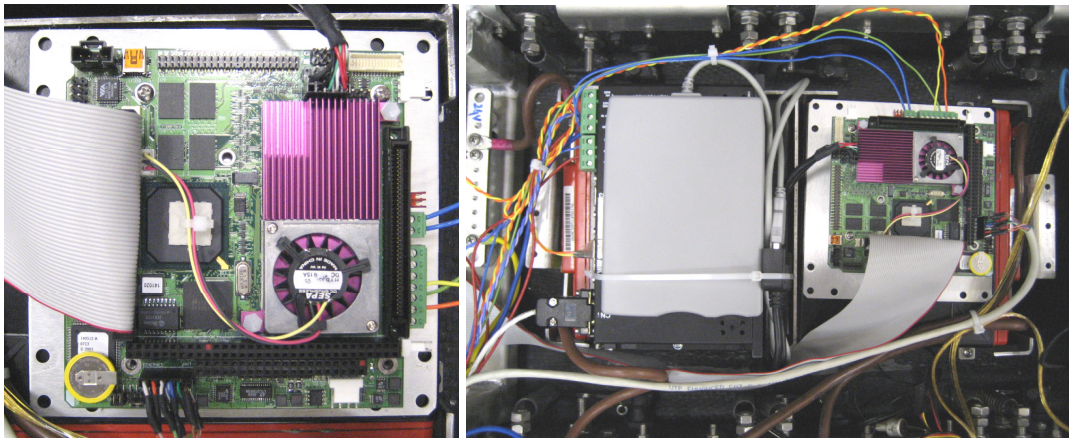

Figure 4.13 – Control Computer Used in the Experimental Setup

The hardware implementation medium of the proposed architecture is explained schematically in Figure 4.14. As can be seen from this figure, the controller model is designed in the host computer using MATLAB and Simulink as block diagrams and C codes. Then this controller model is compiled and emdebbed into the target computer. Since the target computer has its own kernel and there is no overhead other than the control software, this computer can run the control code on real-time.

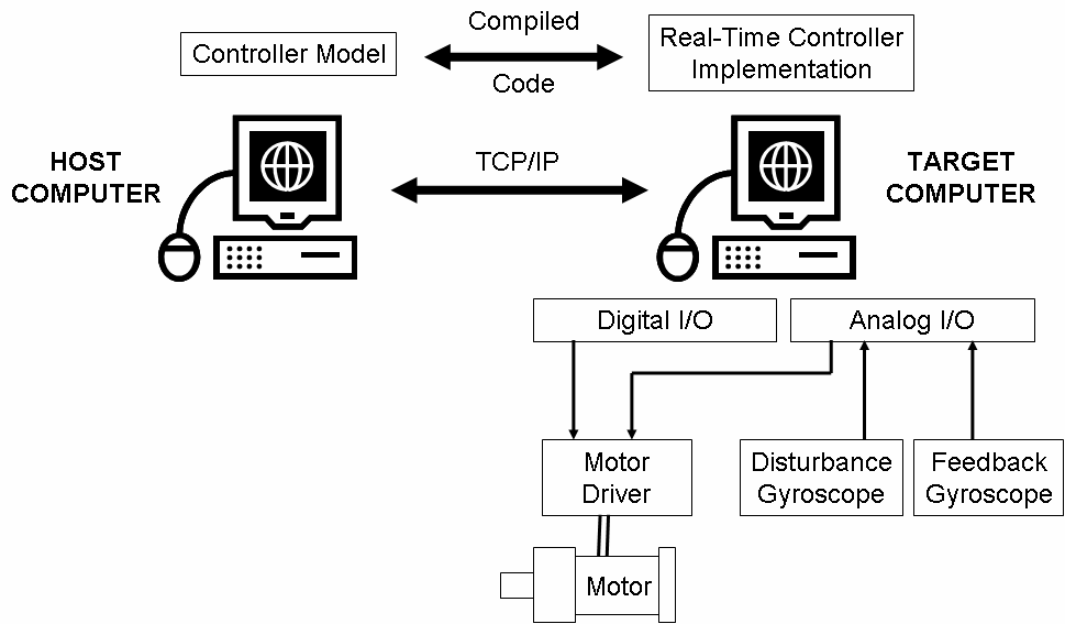

Figure 4.14 – xPC Target Implementation of the Proposed Neural Controller on the Model Main Battle Tank

The target computer waits for a start message from the host computer. After it receives the start message, the target computer starts to run the controller. With wireless TCP/IP connection, any data can be obtained from the target computer real-time and any command signal or any parameter change can be sent using this connection.

In the experimental setup, the backlash between the traverse axis ring gear and motor pinion is minimized and therefore can be ignored. On the other hand, considerable Columb friction exists in the system.

The details of the mathematical model and hardware experimental setup are given in this chapter. The results obtained as a result of the implementation of the proposed controller architecture will be given in the next chapter.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Proposed Neuro Controller Applied to the Computer Simulations

The proposed neuro controller is applied to the mathematical model of the Leopard 1 tank for the stabilization control of both traverse and elevation axis. The controller architecture is shown here again in Figure 5.1 for the sake of ease in interpretation of the results.

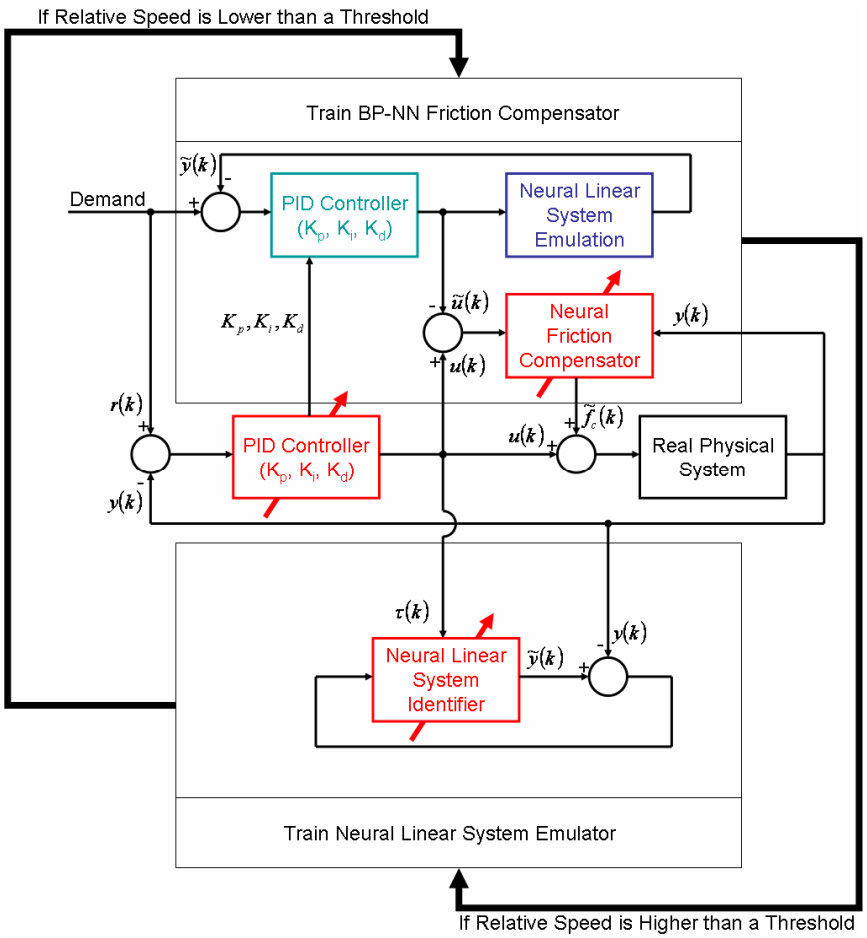

Figure 5.1 – Proposed Neuro Controller Architecture

In the following sections, the proposed control modules namely the neural PID tuner, neural linear system identifier and neural friction compensator are applied to the system model in the following order. First the neural PID tuner is applied to the system without friction compensation (neural linear system identifier + neural friction compensator). The results are obtained for the network parameters designed in chapter 3, which are the learning rate, momentum term, activation function type, number of inputs, number of hidden layers and number of neurons in these hidden layers. Then, friction compensation is applied to the system at the same time with the neural PID tuner and results showing the effect of friction compensation on the control performance are given in detail.

Before passing to the simulation results, the software prepared for these simulations will be discussed. Since the simulation are performed on Simulink environment, the three controller module (neural PID tuner, neural system identifier and neural friction compensator) are coded in C language in order to obtain C-MEX S-functions for simulink.

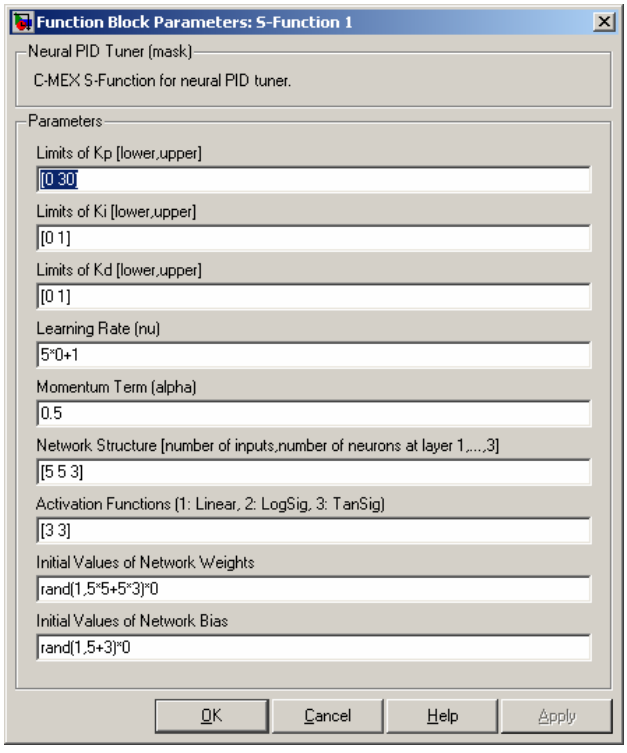

Figure 5.2 – Graphical User Interface for Neural PID Tuner

The graphical user interface developed for neural PID tuner is given in Figure 5.2. Using the interface, the limits are entered for each controller gain. In addition, the learning rate and momentum term are defined from here. The network structure, which determines the number of layers and number of neurons in these layers and activation function types for each layer are entered using this interface. Initial values for network weights and biases are also entered from this interface.

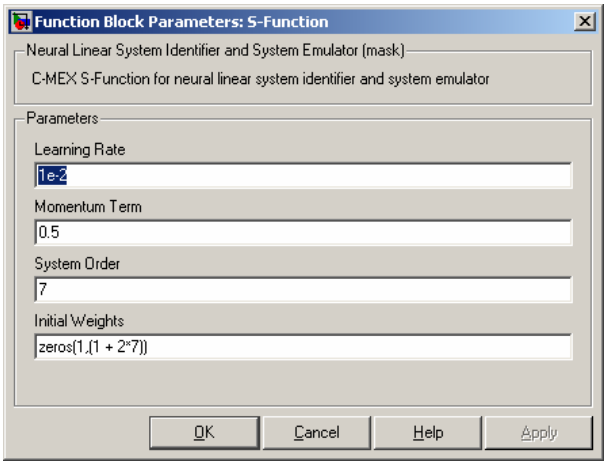

Figure 5.3 - Graphical User Interface for Neural Linear System Identifier and System Emulator

A similar user interface is prepared for the neural linear system identifier and system emulator and learning rate, momentum term, system order and initial weights are entered using this interface. The screen shot is given in Figure 5.3.

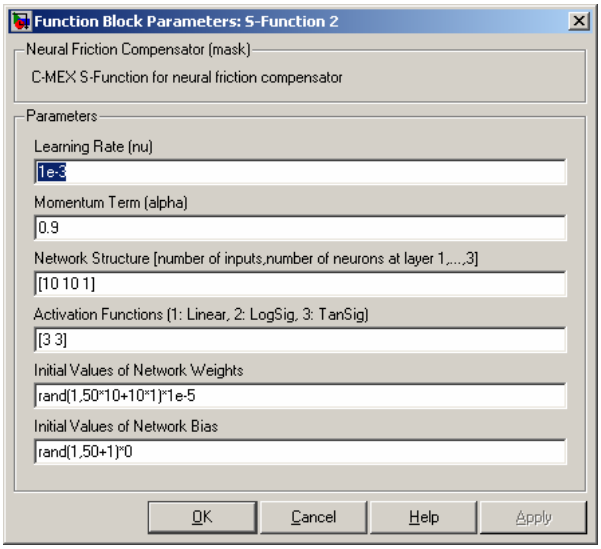

Figure 5.4 – Graphical User Interface for Neural Friction Compensator

The user interface prepared for the neural friction compensator is given in Figure 5.4. From this interface, the learning rate and momentum term are entered. Similarly, network structure showing the number of layers and number of neurons in these layers is entered here. Activation function types and initial values of network weights and biases are also entered using this interface.

Before giving the results of the simulations, the parameters used in these simulations will be listed in Table 7 for the neural PID tuner, in Table 8 for the neural linear system identifier and in Table 9 for the neural friction compensator for the sake of completeness.

Table 7 – Network Parameters Used in the Computer Simulations for the Proposed Neural PID Tuner

	Elevation Axis	**Traverse Axis**
Number of Hidden Layers	1	1
Number of Neurons in the Input Layer	10	10
Number of Neurons in the Hidden Layer	20	20
Activation Function Type	Tangent Hyperbolic Sigmoid Function	Tangent Hyperbolic Sigmoid Function
Learning Rate	0.025	0.025
Momentum Term	0.4	0.4
Proportional Gain Limits	0 (min) , 5 (max)	0 (min) , 10 (max)
Integral Gain Limits	0 (min) , 0.5 (max)	0 (min) , 0.5 (max)
Derivative Gain Limits	0 (min) , 1 (max)	0 (min) , 1 (max)
Initial Network Weights	Random Between -0.001 and 0.001	Random Between -0.001 and 0.001
Initial Network Biases	Random Between -0.001 and 0.001	Random Between -0.001 and 0.001

Table 8 – Network Parameters Used in the Computer Simulations for the Proposed Neural Linear System Identifier

	Elevation Axis	**Traverse Axis**
System Order	10	10
Learning Rate	0.03	0.03
Momentum Term	0.4	0.4
Initial Network Weights	0	0
Activation Speed	>1 deg/sec	>1 deg/sec

Table 9 – Network Parameters Used in the Computer Simulations for the Proposed Neural Friction Compensator

	Elevation Axis	**Traverse Axis**
Number of Hidden Layers	1	1
Number of Neurons in the Input Layer	15	15
Number of Neurons in the Hidden Layer	3	3
Activation Function Type	Tangent Hyperbolic Sigmoid Function	Tangent Hyperbolic Sigmoid Function
Learning Rate	0.003	0.003
Momentum Term	0.4	0.4
Output Gain	0.4	0.6
Initial Network Weights	Random Between -0.0001 and 0.0001	Random Between -0.0001 and 0.0001
Activation Speed	<1 deg/sec	<1 deg/sec

5.1.1 Neural PID Tuner Applied to the Computer Simulations

The proposed neural PID tuner is applied to the mathematical model of the Leopard 1 tank in both traverse and elevation axis. The task of the PID controller is the stabilization of the corresponding axis and this is done by using a feedback gyroscope. The controller uses the error between the aiming command signal from the joystick and the speed feedback from the gyroscope. The controller uses this error signal in order to generate an output torque at the corresponding actuator (servo motor) using a discrete PID controller given as;

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i e(k) + K_d\left(e(k) - 2e(k-1) + e(k-2)\right) \quad (5.1)$$

Here in this equation, K_p is the proportional gain, K_i is the integral gain and K_d is the derivative gain of the controller and these controller gains are tuned using the proposed neural PID tuner.

For the simulations performed in this chapter, the mathematical model of the main battle tank goes over the APG track with different speeds for 350 seconds for the elevation axis simulations and the model goes over the sinus track for 350 seconds for the traverse axis simulations.

The controller architecture for the elevation axis is given in Figure 5.5. The same figure is shown before and given here as a repetition. For the traverse axis, the controller structure is shown in Figure 5.6. In both of the two axes, the joystick command is zero therefore the speed of the gun relative to a reference on the ground will remain as zero.

The controller gain limits used in the simulations are 0 as minimum and 5 as maximum for the proportional gain, 0 as minimum and 1 as maximum for the integral gain and 0 as minimum and 1 as maximum for the derivative gain. It is worth to note that with the combination of the controller gains within these limits, the system may become unstable.

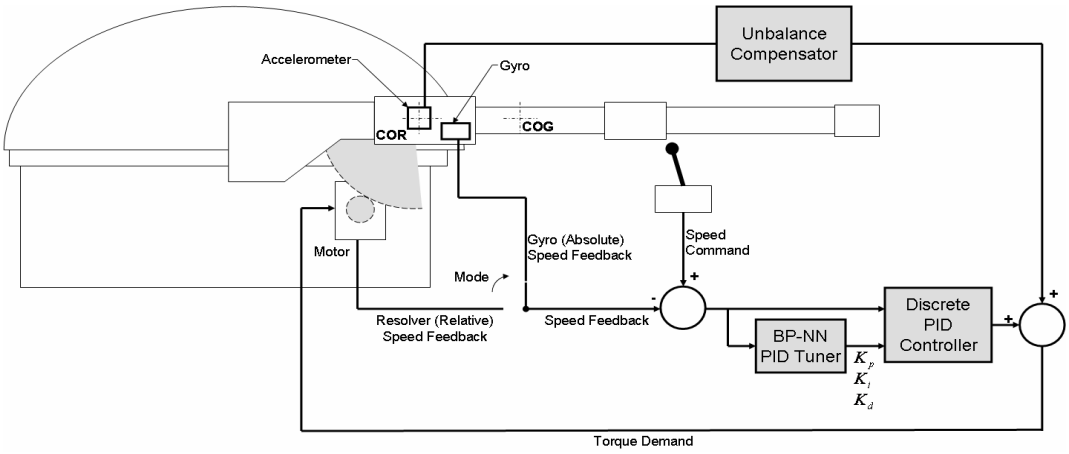

Figure 5.5 – Controller Architecture for Neural PID Tuner Simulations for Elevation Axis

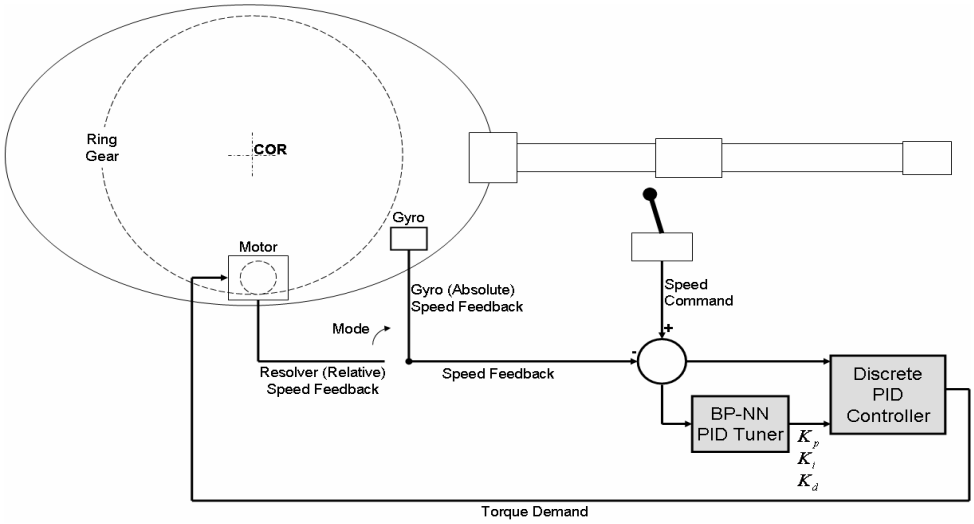

Figure 5.6 - Controller Architecture for Neural PID Tuner Simulations for Traverse Axis

The training of the neural PID controller is allowed only when the forgetting sum of the controller error exceeds 0.015 rad/s (1 deg/s). This value may be decreased but leads to fast increase in the controller gains, which decreases the safety margin. Instead of fast increase in the controller parameters, we desire to increase considerably the stabilization performance by the neural friction compensator. This will be demonstrated a few pages later.

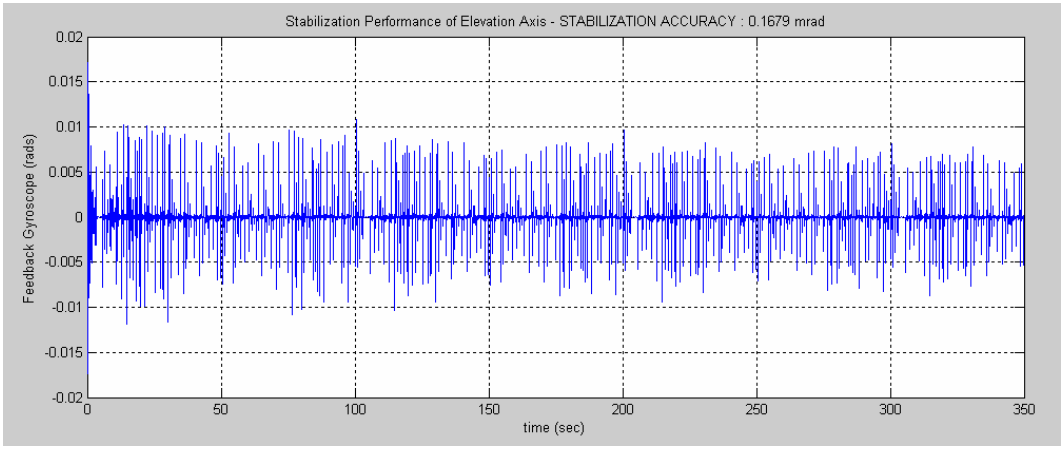

Figure 5.7 – Stabilization Performance of the PID Controller with Neural PID Tuner for the Elevation Axis

The stabilization performance of the proposed PID controller with the neural PID tuner is given in Figure 5.7 for the elevation axis. The figure shows the feedback gyroscope signal. The stabilization accuracy value obtained in this simulation is 0.1679 mrads.

The gyroscope speed, which is actually the controller error because the speed command is zero, decreases gradually during the training of the network. This decrease in error is shown in the following figures.

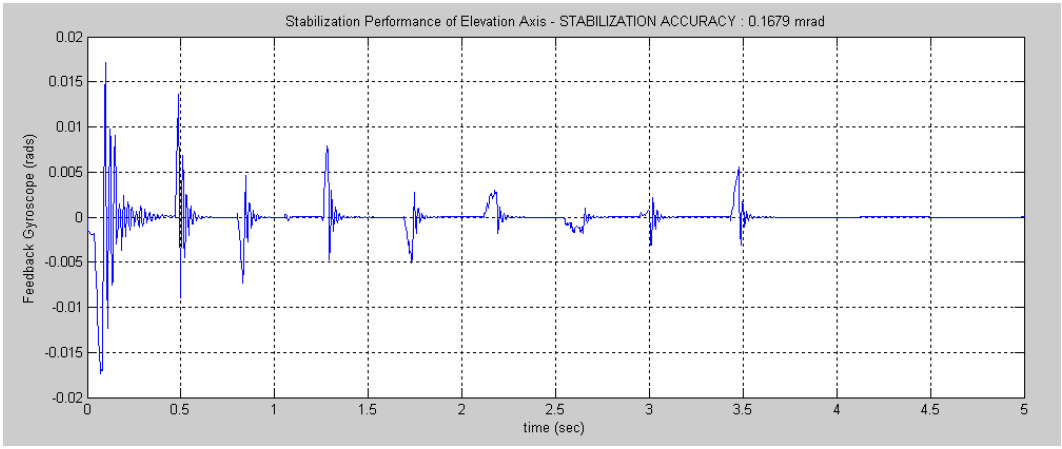

Figure 5.8 – Feedback Gyroscope Signal for Time Range 0 and 5 seconds

In Figure 5.8, the gyroscope speed signal is given for time range 0 and 5 seconds. In this time range, the controller starts with zero gains and training of the network starts. After 300 seconds where the same disturbance is applied to the system, the controller error or the gyroscope speed feedback is given in Figure 5.9. Both the oscillations in the system response and error values are decreased considerably.

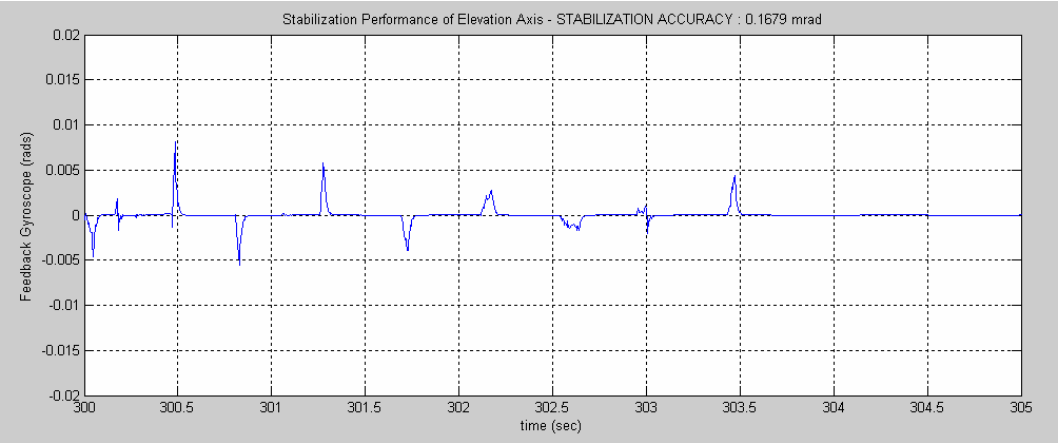

Figure 5.9 – Feedback Gyroscope Signal for Time Range 300 and 305 second

The controller parameters namely the proportional gain K_p, integral gain K_i and derivative gain K_d are given in Figure 5.10. In this figure, the controller parameters are given for the full simulation time range and for the very first of the simulation. As seen from these figures, the integral gain reaches its final value in a very short time. The derivative gain is nonzero for a short time and settles to zero for the remaining of the simulation. The proportional gain increases gradually in order to decrease the controller errors. Due to the high friction in the elevation axis, the proportional gain continues to increase but this increase stops after the addition of neural friction compensator to the controller architecture.

The figure shows that the controller parameters are learned by the proposed neural PID tuner in a very short time. For the elevation axis, the final values of integral and derivative gains are reached almost in 3 seconds.

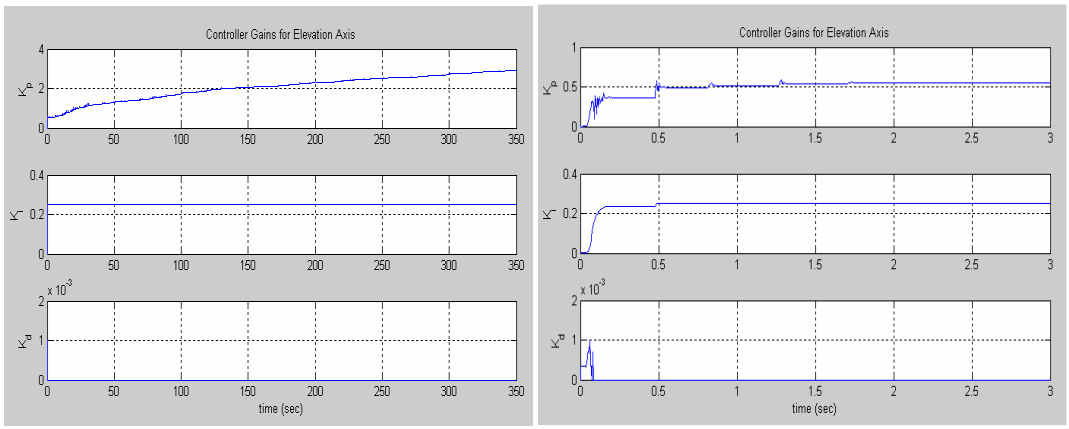

Figure 5.10 – Controller Gains for the Elevation Axis Simulations

Simulations are also performed using the proposed PID controller tuned with the neural PID tuner for the traverse axis. The feedback gyroscope signal is given in Figure 5.11. Since the speed command in this simulation is zero, this gyroscope signal is equal to the controller error. Since the disturbance profile is a sinus for the traverse axis, the error decrease with time is not so obvious from the figure but a closer look may help see this error decrease. The stabilization accuracy value obtained for this simulation for the traverse axis is 0.2441 mrads. This value is a little high compared to the elevation axis and this difference is due to the high inertia of the traverse axis compared to the inertia of the elevation axis.

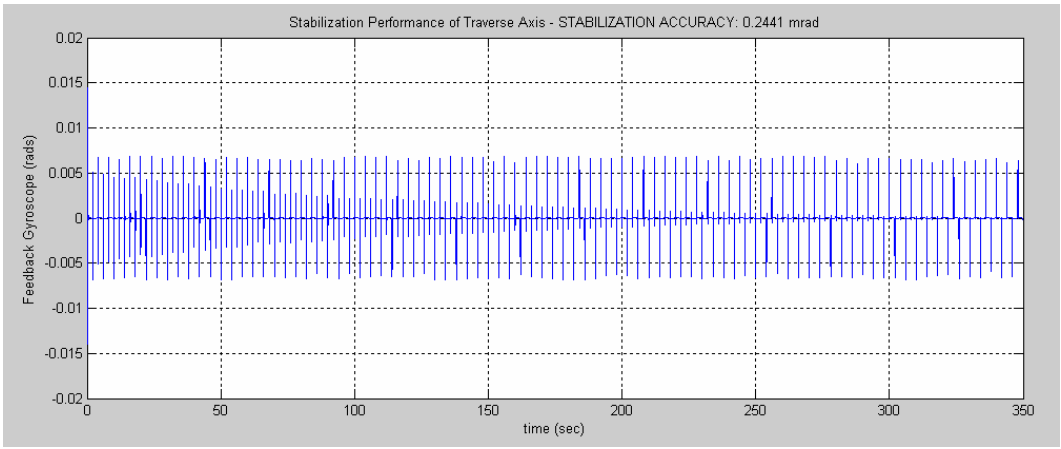

Figure 5.11 - Stabilization Performance of the PID Controller with Neural PID Tuner for the Traverse Axis

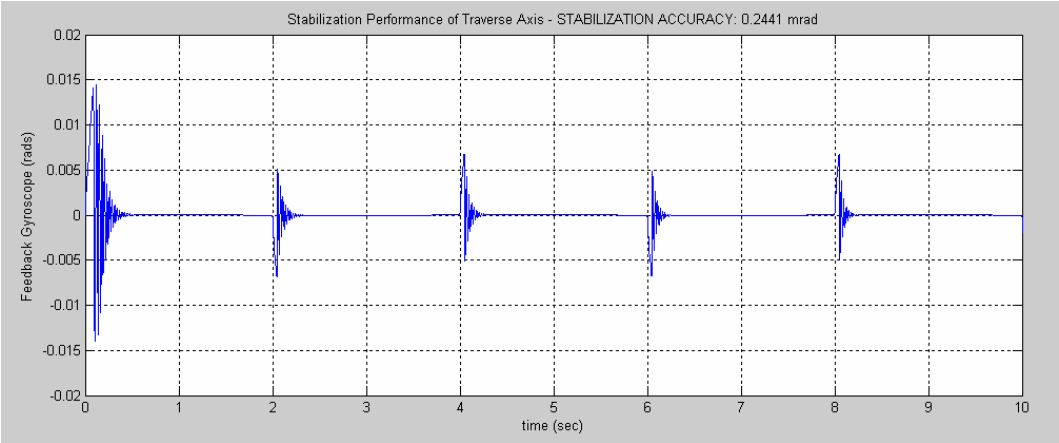

Figure 5.12 - Feedback Gyroscope Signal for Time Range 0 and 10 seconds

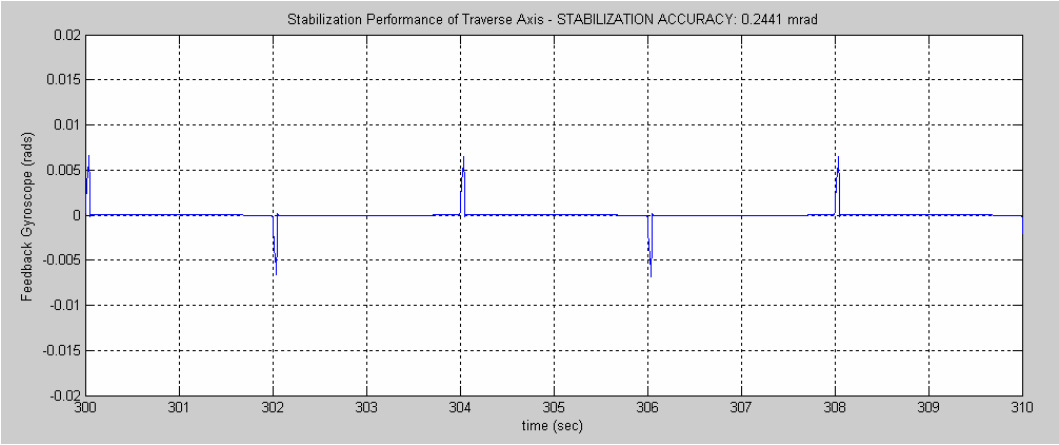

Figure 5.13 – Feedback Gyroscope Signal for Time Range 300 and 310 seconds

For the first figure (Figure 5.12), first 10 seconds of the simulation is given. The oscillations in the system response and error values are notable.

For the second figure (Figure 5.13), the time range between 300 and 310 seconds is given. The oscillations in the system response vanished and the error magnitudes decrease considerable in this time range.

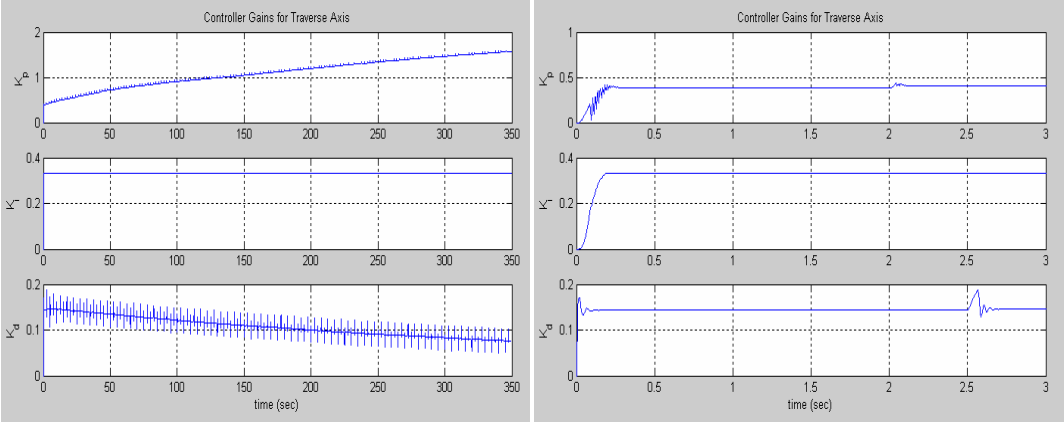

Figure 5.14 - Controller Gains for the Traverse Axis Simulations

The change of the controller gains during the simulation performed for traverse axis is given in Figure 5.14. Again a closer look for the beginning of the simulation is given in this figure. As seen from the figure, the integral gain settles to its final value within very short time. In addition, the derivative gain reaches its final value in this short time period. The proportional gain again increases gradually due to high friction in the traverse axis.

After this point, the proposed neural friction compensator with the neural linear system identifier will be added to the controller architecture. The details of the neural friction compensator design and simulation results will be given in section 5.1.2.

5.1.2 Neural PID Tuner + Neural Friction Compensator Applied to the Computer Simulations

In the previous section, the results of the neural PID tuner on the mathematical model are provided. In this section, however, the proposed neural friction compensator will be added to the neural PID tuner in order to analyze the stabilization performance under the influence of high friction effects.

The controller gains of the proposed neural PID tuner are given in Figure 5.15 for the elevation axis on APG track with 10 km/h speed. As can be seen from the figure, the integral gain reaches its final value in a very short time. The proportional gain continues to increase with time due to the Columb friction in the

system. However, this increase stops after the neural friction compensator learns the friction characteristics of the system and compensates the friction. The stabilization accuracy obtained in this simulation is mrad for elevation axis.

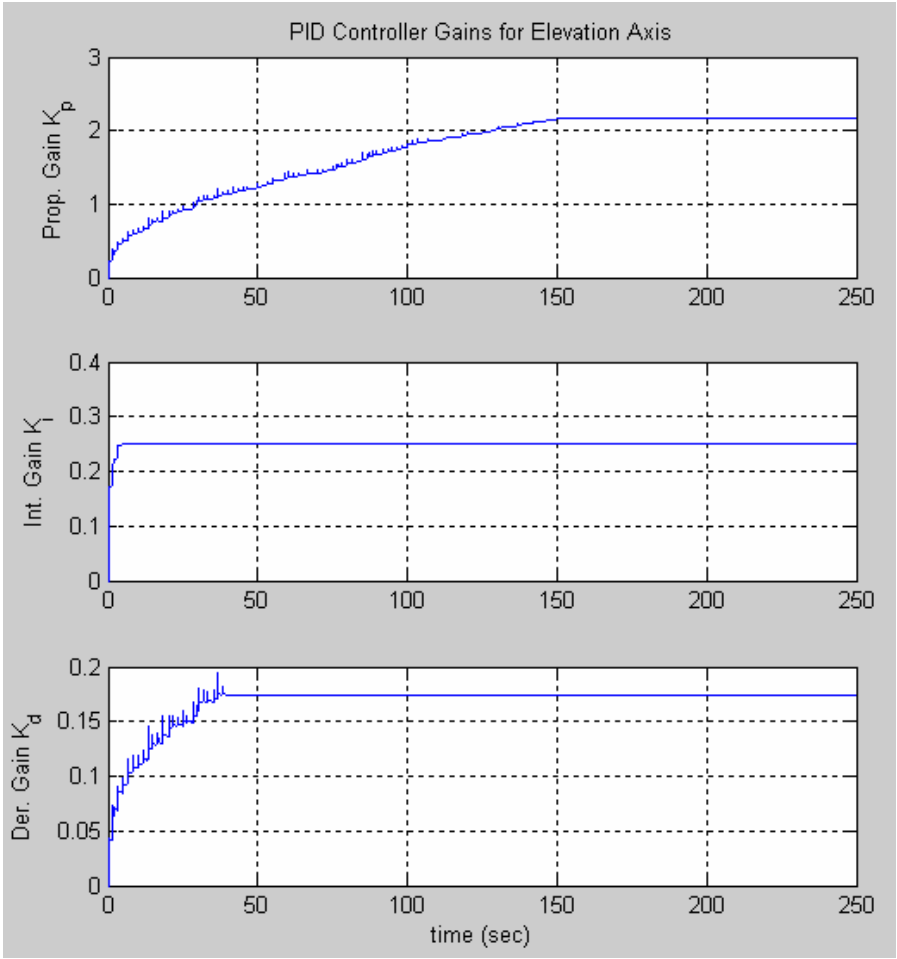

Figure 5.15 – Simulation Result for the Elevation Axis – PID Controller Parameters Learned Through the Simulation on APG Track with 10 km/h Speed

The effect of the friction compensation on the stabilization accuracy value is really considerable. Before the friction compensation is activated, the stabilization accuracy obtained is 0.3 mrad. After the activation of the friction compensator, this value decreases to 0.13 mrad. This means 57% improvement in the stabilization performance for the elevation axis. The results of the simulations related to the stabilization performance of the elevation axis for the final computer simulation are given in Figure 5.16.

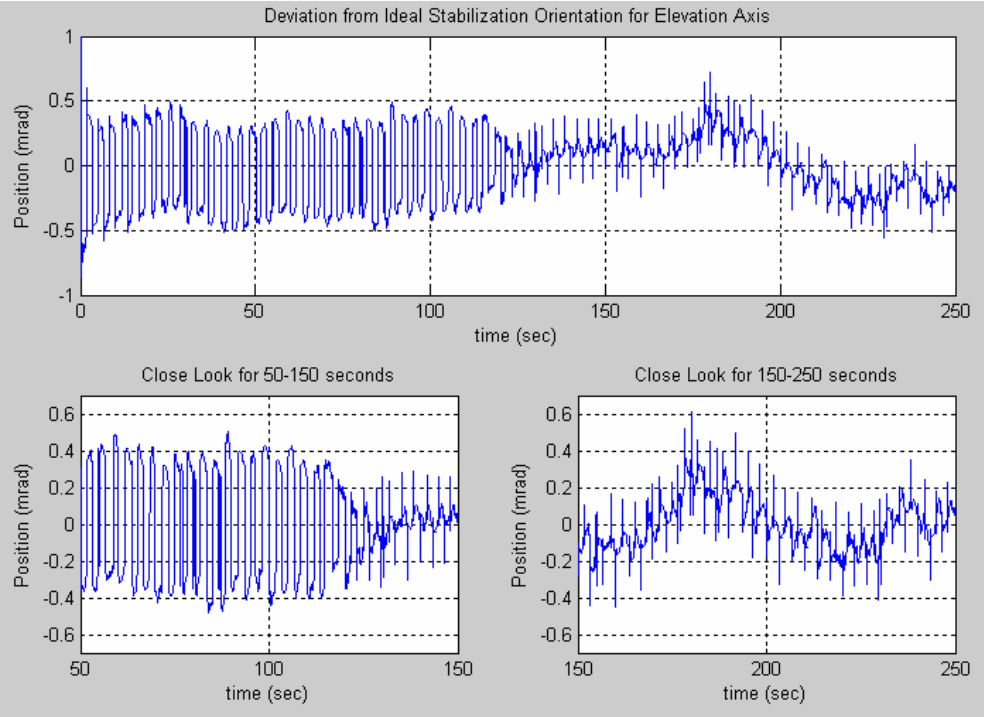

Figure 5.16 – Stabilization Performance of the Elevation Axis on APG Track with 10 km/h Speed for the Final Simulation

The controller gains obtained from the computer simulations learned by the proposed neural PID tuner for the traverse axis is given in Figure 5.17. As seen from the figure, proportional, integral and derivative gains increase gradually with time. This increase is large in the beginning of the simulation and parameter increase slows down at the end due to the decrease in the controller error.

The stabilization accuracy of the traverse axis is given in Figure 5.18. At the beginning of the simulation, the stabilization performance is not satisfactory. However, after the training of the neural PID tuner, the stabilization accuracy becomes 0.39 mrad. This value is larger than the stabilization accuracy of the elevation axis and this is mainly due to high inertia of the traverse axis. This stabilization performance increases a lot with the activation of the neural friction compensator. The stabilization accuracy value becomes 0.04 mrad and this means %90 increase in the stabilization performance.

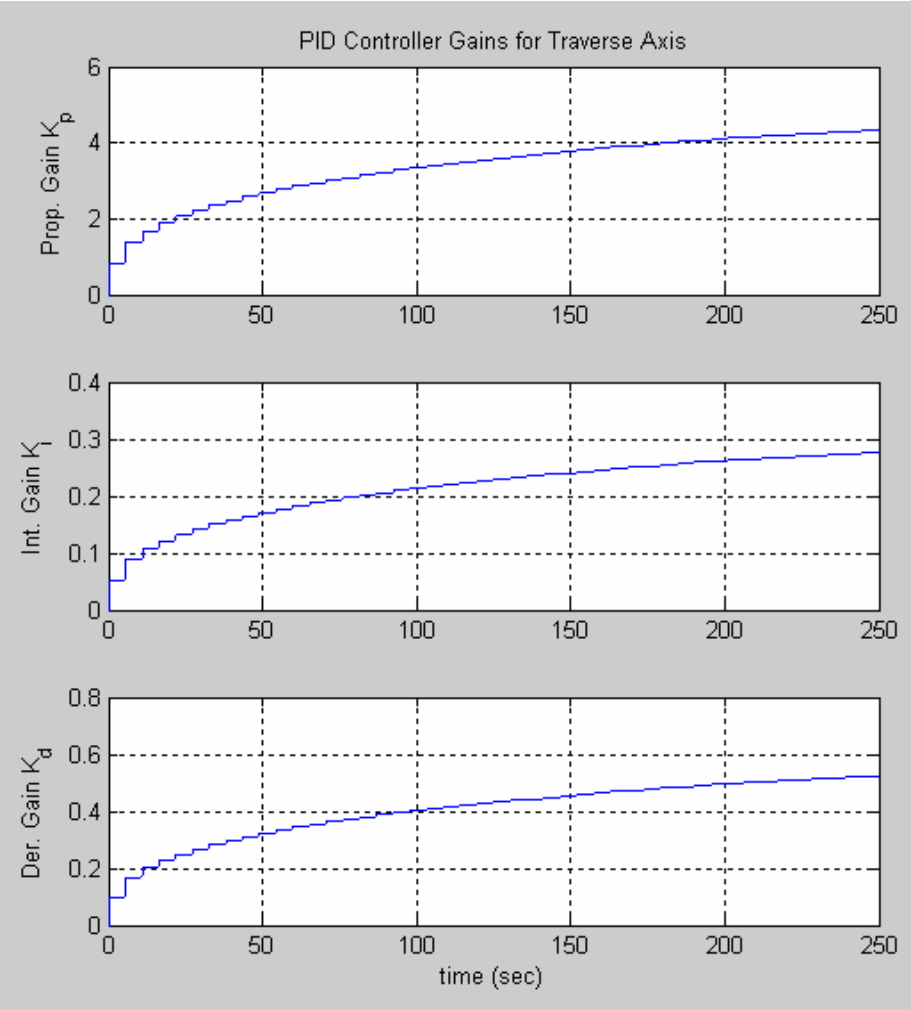

Figure 5.17 - Simulation Result for the Traverse Axis – PID Controller Parameters Learned Through the Simulation on Sinus Track with 10 km/h Speed

Considering the neural linear system identifier, the estimation error of the network for the elevation axis is given in Figure 5.19. The error of the network decreases to very low values (approximately 0.01 rad/sec or 0.6 deg/sec) in the first 50 seconds. This error value increases a little with the activation of the neural friction compensator and this is due to the change in the system dynamics with the addition of friction compensator. Error in the estimation of the system response decreases again slowly to 0.6 deg/sec at the end of the simulation.

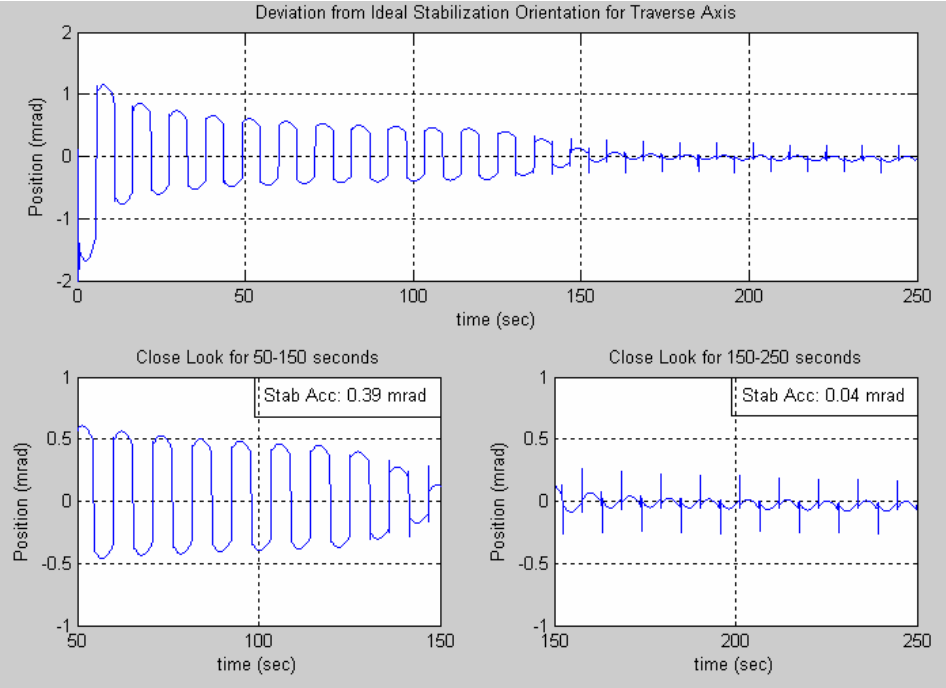

Figure 5.18 - Stabilization Performance of the Traverse Axis on Sinus Track with 10 km/h Speed for the Final Simulation

The error of the neural linear system identifier for the traverse axis is given in Figure 5.20. As seen from the figure, the network starts with almost zero estimation error and decreases slowly to approximately 1 deg/sec value. Again this error increases with the activation of the neural friction compensator but decreases considerable at the end of the simulation.

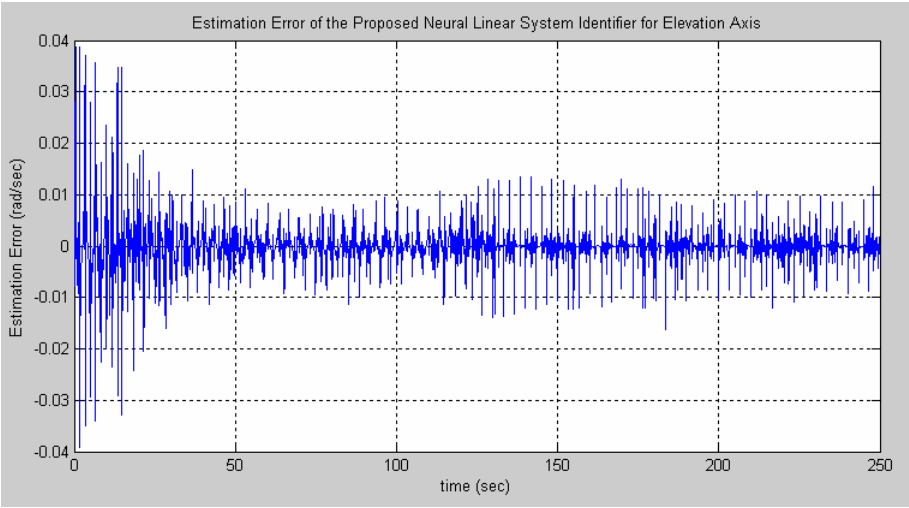

Figure 5.19 – Estimation Error of the Neural Linear System Identifier for the Elevation Axis on APG Track with 10 km/h for the Final Simulation

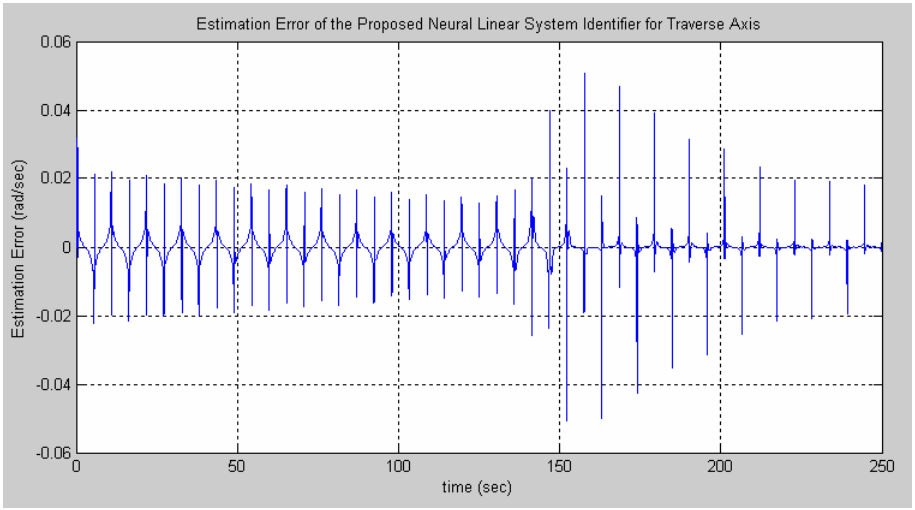

Figure 5.20 - Estimation Error of the Neural Linear System Identifier for the Traverse Axis on Sinus Track with 10 km/h for the Final Simulation

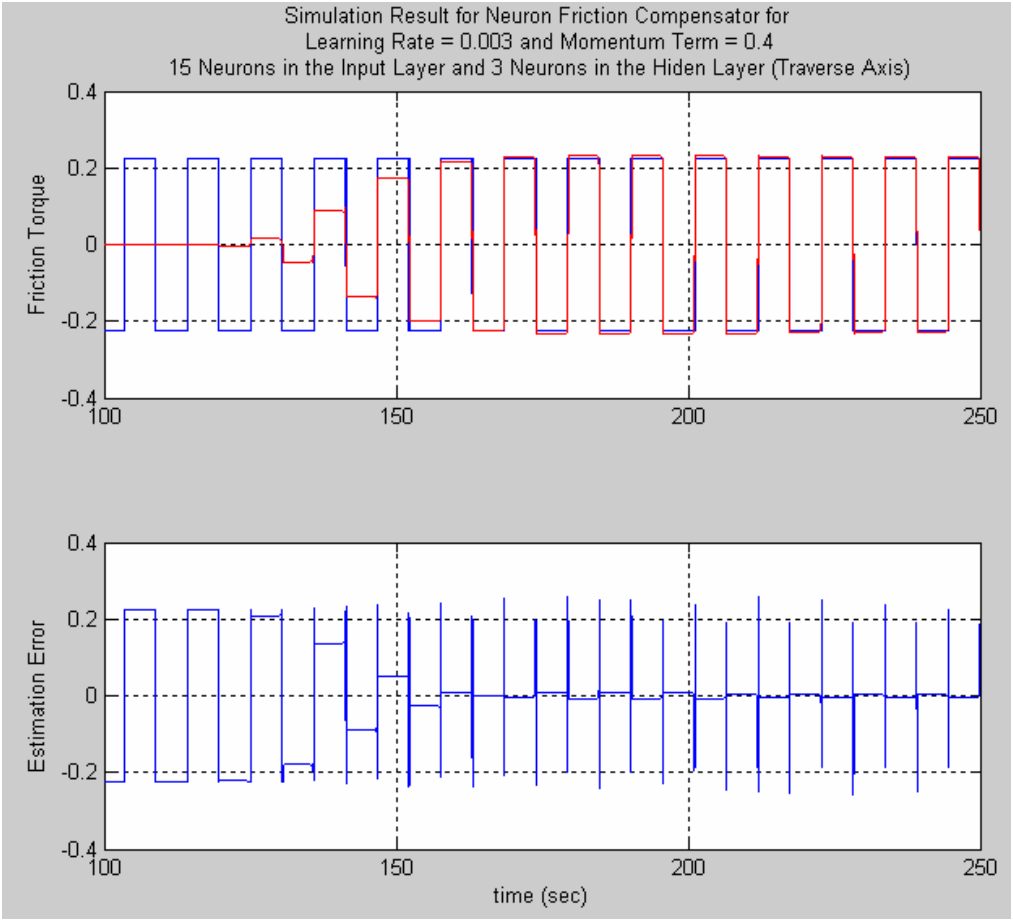

Figure 5.21 - Simulation Results for Neural Friction Compensator for Learning Rate $\eta = 0.003$, momentum term $\alpha = 0.4$, 15 Neurons in the Input Layer and 3 Neurons in the Hidden Layer for the Traverse Axis

The result of the neural friction compensator for the elevation axis is not repeated here but is given in Figure 3.24. As seen from this figure, the Columb friction present in the system is learned within a very short time (approximately in 50 seconds) after the activation of the training of the network. The figure showing the result of the friction learning process for the traverse axis is given Figure 5.21. Again the Columb friction present in the traverse axis is learned in a very short time by the proposed network and with a very low error.

The results of the computer simulations for both elevation and traverse axes are given above. In these simulations, the results of the three networks are obtained and represented here. After this point, the proposed controller architecture will be applied to the experimental setup and the results obtained in this study will be discussed.

5.2 Proposed Neural Controller Applied on the Experimental Setup

The proposed neural controller architecture consisting of the neural PID tuner, neural linear system identifier and the neural friction compensator is also applied on the experimental setup developed for this study.

Before passing to the simulation results for the experimental setup, the control model used in these simulations will be given in detail. As mentioned before, Simulink and xPC Target is used for the real-time implementation of the proposed neural controller. The block diagram of the neural controller used in the experimental setup is shown in Figure 5.22.

As seen from the figure, the controller consists of a plenty of sub-blocks. The blocks named ‘SIGNALS FROM THE PLANT’ and ‘SIGNALS TO THE PLANT’ are the interfaces with the real physical plant used to read the feedback signals and send the actuation signal to the actuation system.

The block named ‘SYSTEM MODE’ is used to switch between the system modes STAB MODE and NON-STAB MODE. In stab mode, the feedback sensor is the feedback gyroscope, which is on the turret of the model tank. In non-stab mode, the difference between the feedback gyroscope and the disturbance gyroscope

placed on the vehicle of the model tank is used as the feedback signal since there is no resolver or other sensor, which is, can measure the relative motion of the turret.

The block named 'FAULT CHECK' is used to protect the system from the cases, which may damage the system. This block captures the case in which the actuation signal is above a threshold for a time period in order to protect the cables from damage by twisting because holding the actuation signal high for a period of time causes the traverse axis to turn to the same direction with many turns.

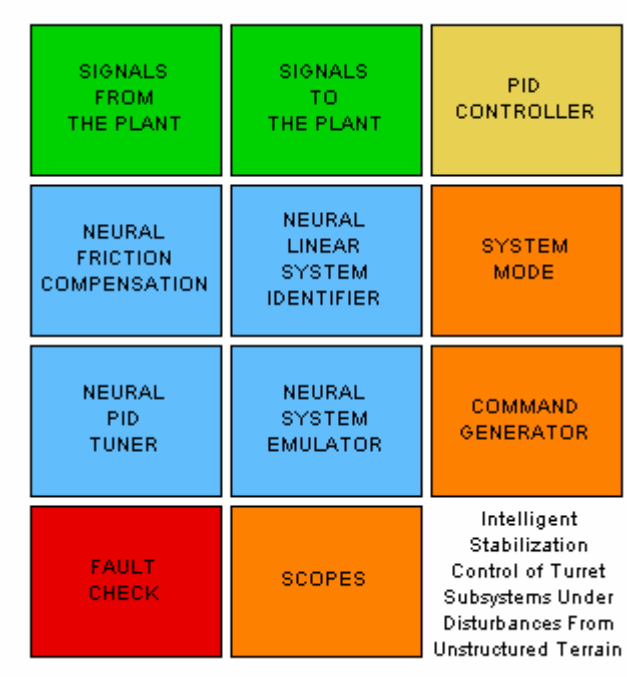

Figure 5.22 – Simulink Block Diagram of the Neural Controller for the Experimental Setup

The speed command signals are generated in the block named 'COMMAND GENERATOR'. In this block, the signal shapes like sinus, block and saw tooth can be generated. In addition, speed offset can be added to these signals by using this block. The speed commands generated by this block are applied to a rate limiter in order not to give speed commands, which exceeds the acceleration limits (± 240 deg/s) of the turret of the model tank.

Some of the signals like feedback signals and controller gains are plotted and recorded using the block named 'SCOPES'. In this block, many signals can be displayed and saved instantaneously.

The proposed neural PID tuner algorithm is implemented in the block named 'NEURAL PID TUNER'. This block uses the controller errors as the input and outputs the proportional (K_p), integral (K_i) and derivative (K_d) gains for the PID controller which is implemented in the block named 'PID CONTROLLER'.

The linear system identification is performed in the block 'NEURAL LINEAR SYSTEM IDENTIFIER'. This block uses the response of the real physical system, which is the relative speed of the turret, and actuation signal, which is the output of the PID controller of the system as the inputs and outputs the estimated system response. The same block also emulated the virtual linear system.

Finally the block named 'NEURAL SYSTEM EMULATOR' is used to generate the actuation signal for the linear virtual emulated system. The block uses the speed command as the input and controls the virtual linear emulated system.

For the design of these neural components of the proposed controller architecture, a similar optimization procedure can be applied for the controller developed for the experimental setup. However, the parameters of the neural components are tuned by trial and error for the experimental setup for the sake of simplicity. The controller parameters and the structure of the networks are summarized in Table 10 for the neural PID tuner, in Table 11 for the neural linear system identifier and in Table 12 for the neural friction compensator.

In this section, the simulation results of the tests performed on the experimental setup will be given for different test conditions. First, the results of the test with only the neural PID tuner will be given and discussed. Then, the results of the neural linear system identifier and the linear system emulation will be given. Finally, the estimated frictional characteristics of the system by the proposed neural friction compensator will be given and discussed.

Table 10 – Network Parameters Used in the Experimental Setup for the Proposed Neural PID Tuner

	Traverse Axis
Number of Hidden Layers	1
Number of Neurons in the Input Layer	5
Number of Neurons in the Hidden Layer	5
Activation Function Type	Tangent Hyperbolic Sigmoid Fun.
Learning Rate	0.00005
Momentum Term	0.4
Proportional Gain Limits	0 (min) , 8 (max)
Integral Gain Limits	0 (min) , 40 (max)
Derivative Gain Limits	0 (min) , 0.001 (max)
Initial Network Weights	Random Between -0.01 and 0.01
Initial Network Biases	Random Between -0.01 and 0.01

Table 11 – Network Parameters Used in the Experimental Setup for the Proposed Neural Linear System Identifier

	Traverse Axis
System Order	5
Learning Rate	0.00001
Momentum Term	0.9
Initial Network Weights	0
Activation Speed	>5 deg/sec

Table 12 – Network Parameters Used in the Experimental Setup for the Proposed Neural Friction Compensator

	Traverse Axis
Number of Hidden Layers	1
Number of Neurons in the Input Layer	15
Number of Neurons in the Hidden Layer	5
Activation Function Type	Tangent Hyperbolic Sigmoid Fun.
Learning Rate	0.0005
Momentum Term	0.4
Output Gain	4 Volt
Initial Network Weights	Random Between -0.001 and 0.001
Activation Speed	<5 deg/sec

First result for the neural PID tuner on the experimental setup is shown for the case for which the vehicle is not moving but a speed command is applied to the controller. The tracking result of the system is given in Figure 5.23 for the first 20 seconds.

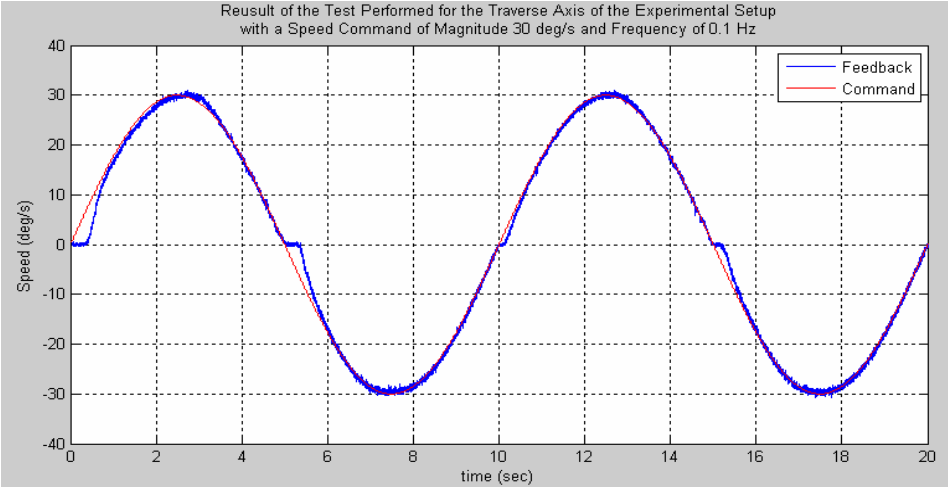

Figure 5.23 – Tracking Performance of the Traverse Axis for a Sinus Speed Command with 30 deg/s Amplitude and 0.1 Hz Frequency

In this test, a sinus speed command with 30 deg/s amplitude and a frequency of 0.1 Hz is applied to the controller. The proposed neural PID tuner starts with zero gains and is trained in order to track the speed command with minimum error.

The figure showing the tracking error at the beginning and the end of the test is given in Figure 5.24. As seen from this figure, the peak tracking error in the beginning of the test is approximately 7 deg/s and this error becomes approximately 1.5 deg/s at the end of the test.

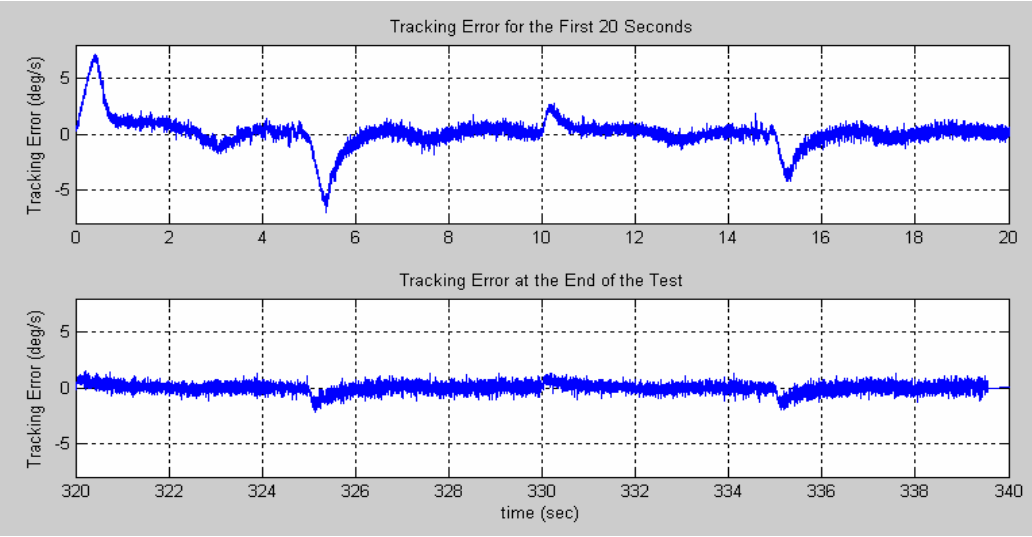

Figure 5.24 – Tracking Errors for the Test with Sinus Speed Command with 30 deg/s Amplitude and 0.1 Hz Frequency

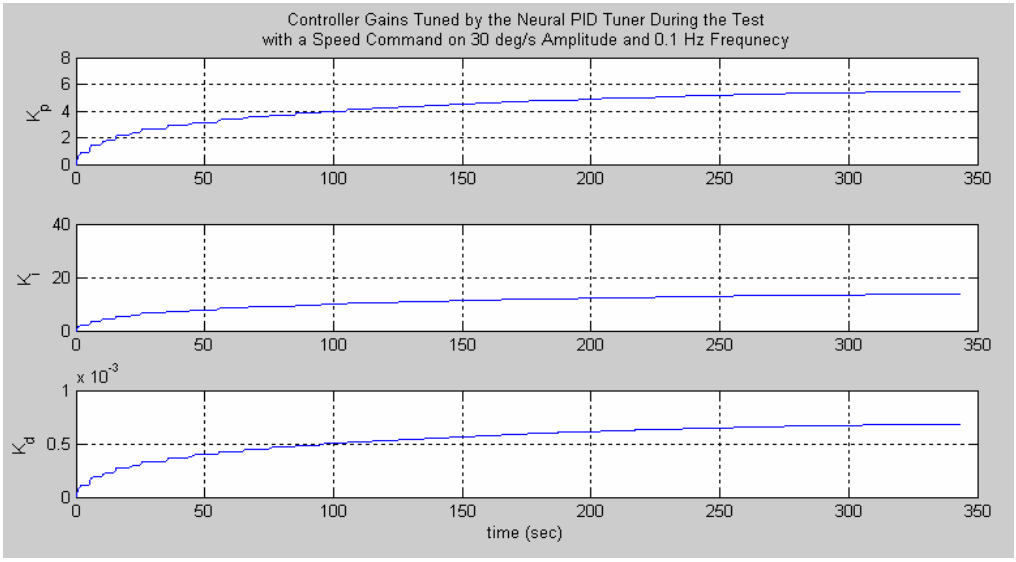

Figure 5.25 – Controller Gains Tuned by the Neural PID Tuner for the Test with Sinus Speed Command of 30 deg/s Amplitude and 0.1 Hz Frequency

The controller parameters tuned by the neural PID tuner during the test is given in Figure 5.25. As seen from the figure, the controller gains are increased gradually in order to minimize the tracking error.

After this test, a stabilization test is performed using only the PID tuner on the experimental setup. The speed command is zero in this test and the vehicle makes randomly turns in the CW and CCW directions.

The speed errors occurred during the test is given in Figure 5.26. As seen from the figure, the speed error reaches to approximately to 7 deg/s in the beginning of the test. After the training of the PID parameters by the neural PID tuner, the stabilization speed error decreases to approximately to 2 deg/s.

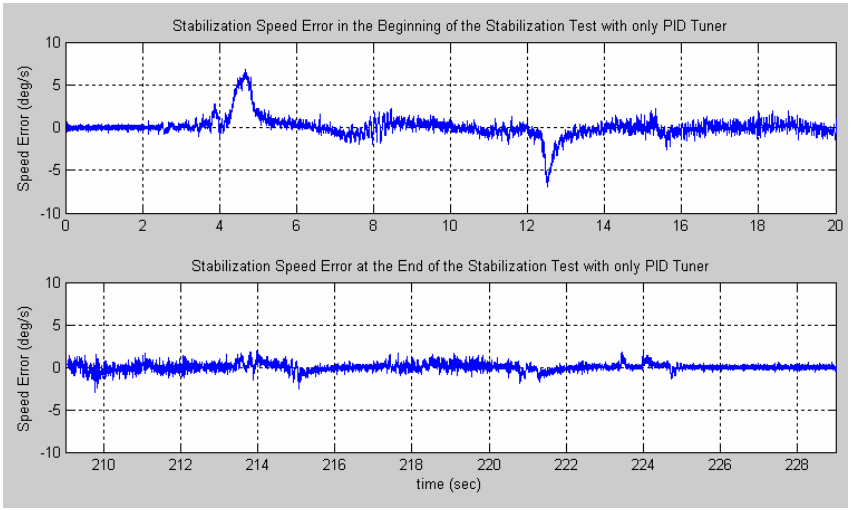

Figure 5.26 – Stabilization Speed Error for the Stabilization Test with only PID Tuner

The stabilization performance of the traverse axis of the model tank is given in Figure 5.27. The stabilization performance in the beginning of the test is calculated as 12.1 mrad. This value is really high considering real life examples. Towards the end of the test, the stabilization performance increases to 4.9 mrad. This value is still high but with time, the stabilization performance increases with the increase of the controller parameters by the neural PID tuner.

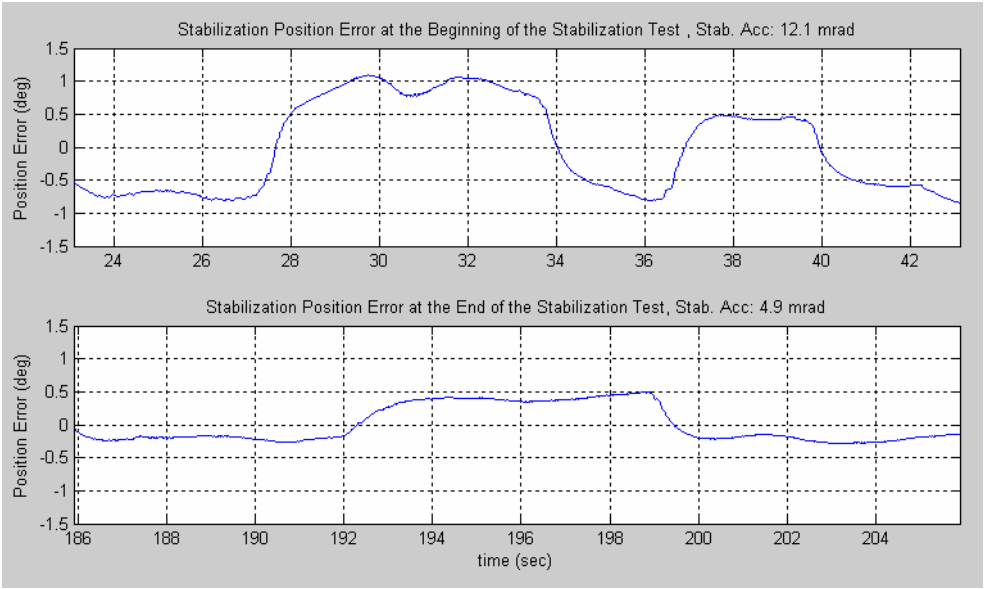

Figure 5.27 – Stabilization Position Error for the Stabilization Test with only PID Tuner

The controller gains learned by the neural PID tuner are given in Figure 5.28. As seen from the figure, the controller parameters increase gradually with time in order to minimize the controller error. For the given time testing period, controller parameters continuously increases which ends when the performance of the controller reaches a satisfactory value.

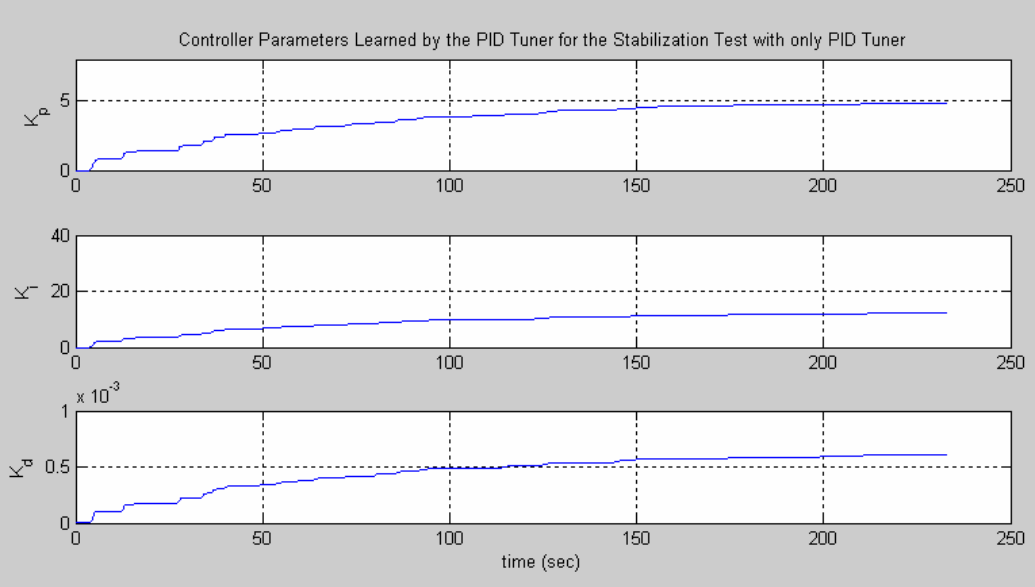

Figure 5.28 – Controller Gains Tuned by the Neural PID Tuner for the Stabilization Test with only the PID Tuner

After giving the results of the neural PID tuner for the case with only the PID tuner, the results of the neural linear system identifier and linear system emulator will be discussed. In this part of the report, we will compare and discuss the outputs of the PID controllers that are driving the real system and the other is driving the linear virtual system emulator.

In order to show the results of the neural linear system identifier and the linear system emulator, a test is performed on the experimental setup with stationary vehicle and a sinus speed command of 10 deg/s amplitude and 0.1 Hz frequency.

One of the outputs of the PID controllers is driving the real physical system and the other is driving the linear system emulator is given in Figure 5.29. As seen

from the figure, the output of the PID controller driving the real physical system has sudden jumps due to the Columb friction. In these regions, the relative speed of the traverse axis changes its direction and therefore the Columb friction changes its direction.

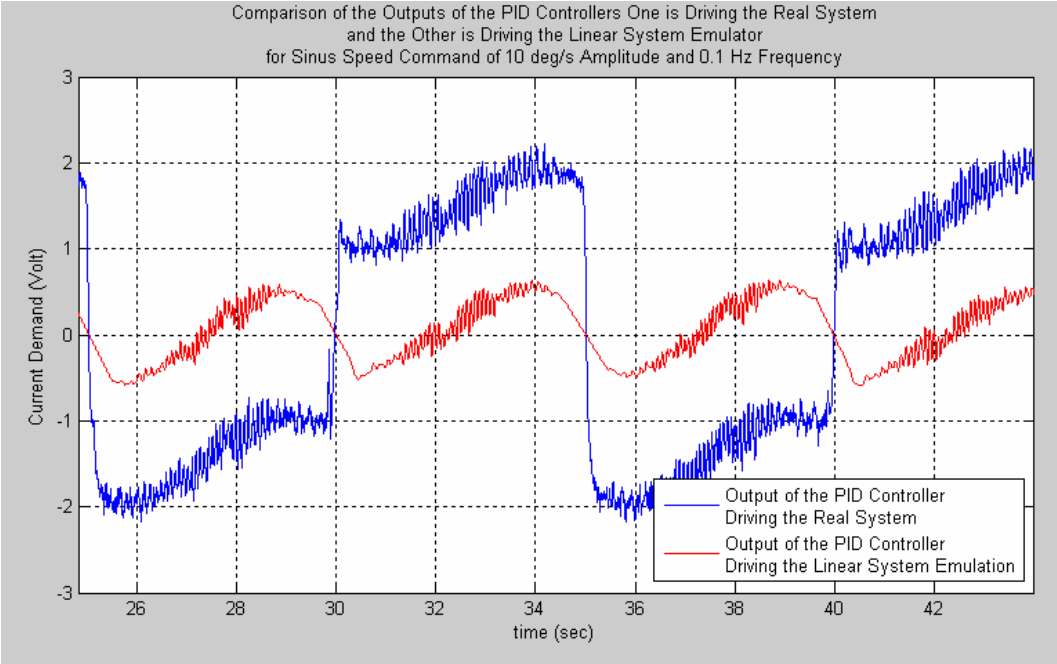

Figure 5.29 – Outputs of the PID Controllers One is Driving the Real System and the Other is Driving the Linear System Emulator for a Sinus Speed Command of 10 deg/s Amplitude and 0.1 Hz Frequency

The output of the PID controller driving the linear system emulator is very similar to the output of the PID controller driving the real physical system if the jumps are eliminated. The proposed neural friction compensator performs this elimination of the jumps. It used the error between these two PID controller outputs in order to learn the friction characteristics and tries to make the output of the PID controller driving the real physical system as close to the output of the PID controller driving the linear system emulation as possible.

A similar test is performed for a sinus speed command of 30 deg/s amplitude and 0.1 Hz frequency. The results are very similar. Two figures show that the emulated system identified by proposed neural linear system identifier represents the characteristics of the real physical system if there were no Columb friction.

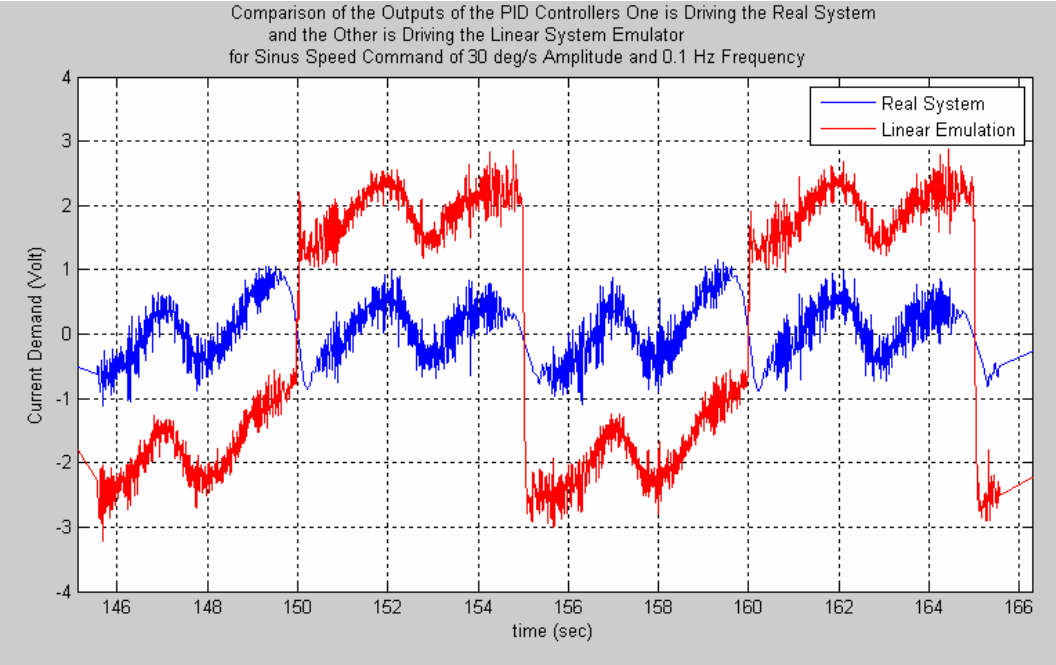

Figure 5.30 - Outputs of the PID Controllers One is Driving the Real System and the Other is Driving the Linear System Emulator for a Sinus Speed Command of 30 deg/s Amplitude and 0.1 Hz Frequency

The neural friction compensator network is analyzed using the same test. At the beginning of the test, both the neural PID tuner and the neural friction compensator start from zero. As time passes, both the PID tuner and the neural friction compensator are trained in order to reach a satisfactory control performance.

The friction characteristics estimated by the neural friction compensator network is shown in Figure 5.31. As seen from the figure, at the beginning of the test, because of the initial conditions of the network, the estimated friction is not true. However, the estimated friction starts to catch the real Columb friction, which exists on the system after the time stamp 50 sec.

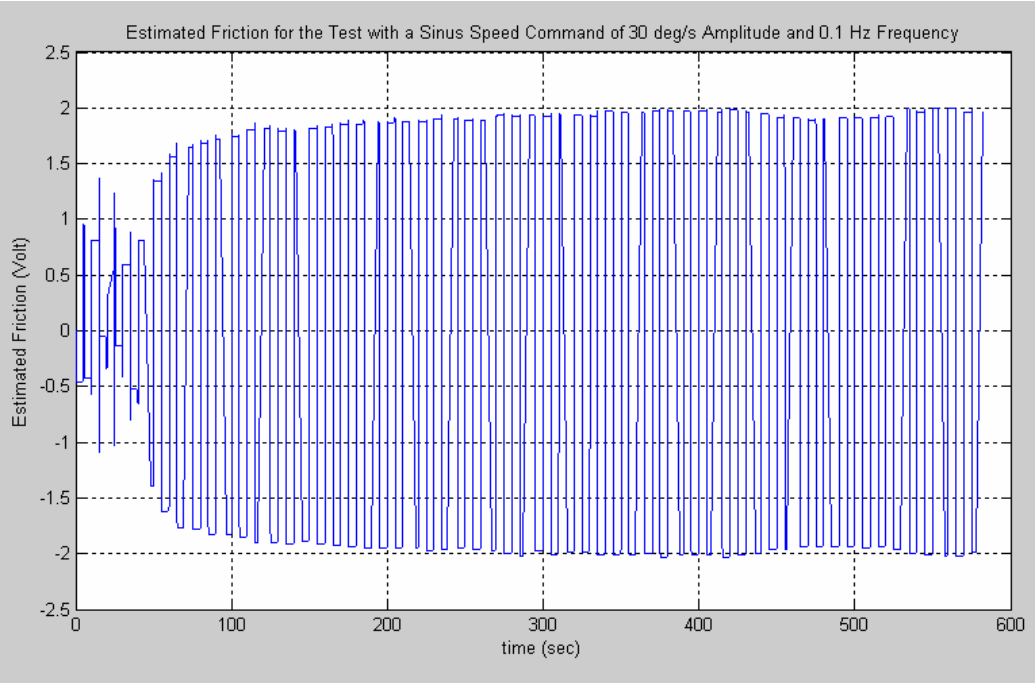

Figure 5.31 – Estimated Friction of the Traverse Axis of the Experimental Setup for a Sinus Speed Command of 30 deg/s Amplitude and 0.1 Hz Frequency

The network almost learns the Columb friction in 200 seconds. This time period needed to learn the true value of the friction is considered to be satisfactory.

The test conditions are the same with the test performed for the case with only PID tuner. Therefore, it is possible to compare the results of these two tests. This comparison will show the performance improvement of the addition of the neural friction compensator to the controller.

The results are given in Figure 5.32. As seen from the figure, the tracking errors are very similar in the beginning of the test since the friction compensator is not trained yet. After the training of the friction compensator, the tracking error is less for the case with friction compensation compared to the case with only PID tuner. The result is not so clear due to the high noise in the feedback signal but the improvement of the addition of the neural friction compensator to the controller will be obvious in the stabilization test.

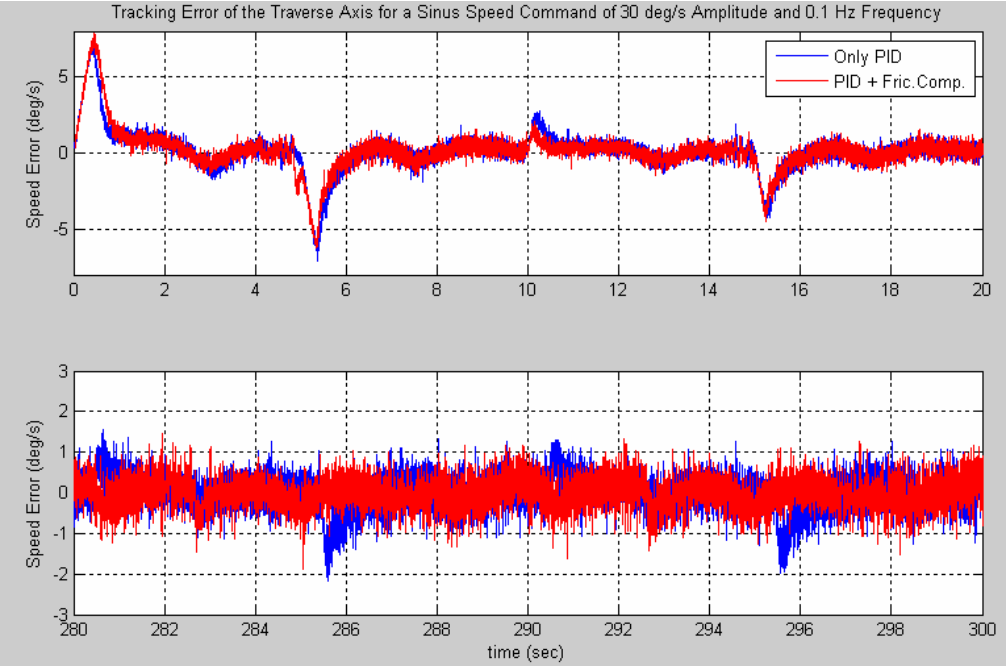

Figure 5.32 – Tracking Errors of the Traverse Axis of the Experimental Setup for Cases with PID Controller and PID Controller + Friction Compensator for a Sinus Speed Command of 30 deg/s Amplitude and 0.1 Hz Frequency

Before passing to the results of the stabilization test with the addition of the proposed neural friction compensator to the controller, the controller parameters tuned by the neural PID tuner is given in Figure 5.33. As seen from the figure, the controller gains are lower for the case with friction compensation then the case without the friction compensator. Therefore, a better tracking result is obtained with lower PID controller gains with the addition of the neural friction compensator to the controller.

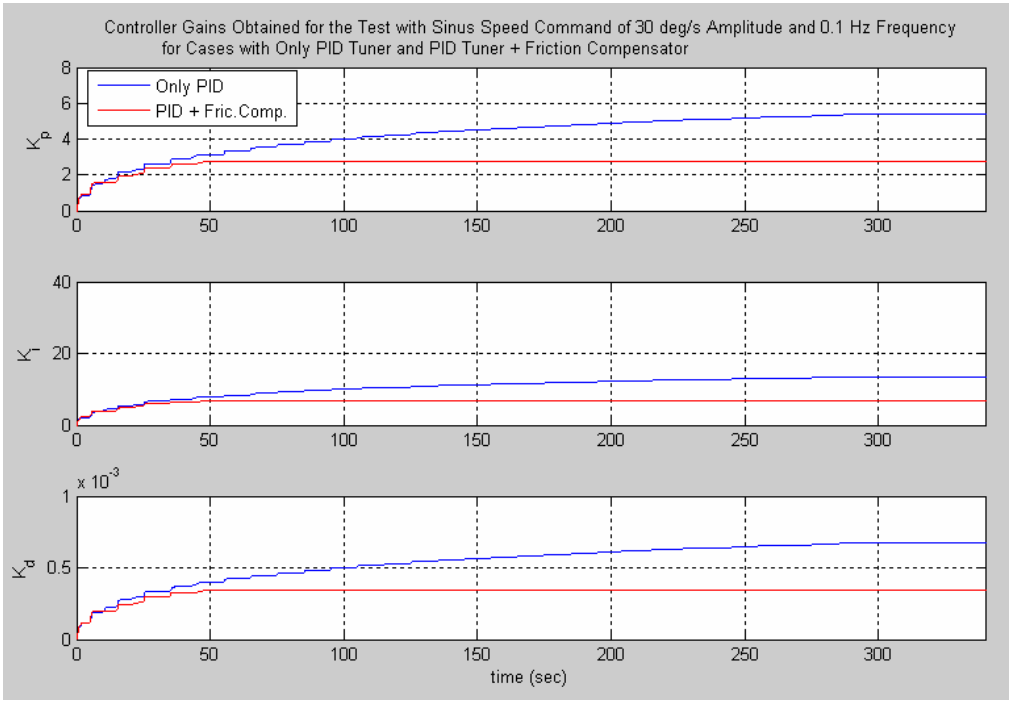

Figure 5.33 – Controller Gains Obtained for Cases with PID Tuner and PID Tuner + Friction Compensator

The neural controller including the proposed neural friction compensator is tested on the experimental setup while speed command is zero and the vehicle moves on a random terrain making turns to the left and to the right. The disturbance applied to the traverse axis is given in Figure 5.34.

The PID controller parameters learned by the neural PID tuner is given in Figure 5.35. As seen from the figure, the controller gains increases gradually in order to minimize the controller error. Compared to the results with only the PID tuner given in Figure 5.28, the controller gains are almost the half at the end of the test. This mainly due to low controller error occurred during the test as a result of the addition of the neural friction compensator.

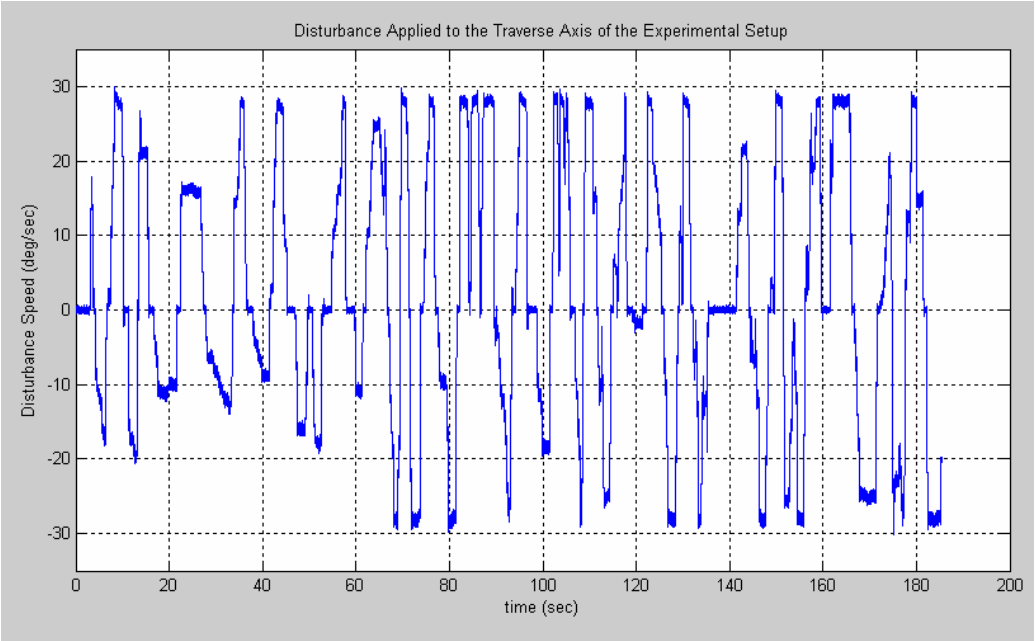

Figure 5.34 – Disturbance Speed Applied to the Traverse Axis of the Experimental Setup for the test of the Neural Friction Compensator

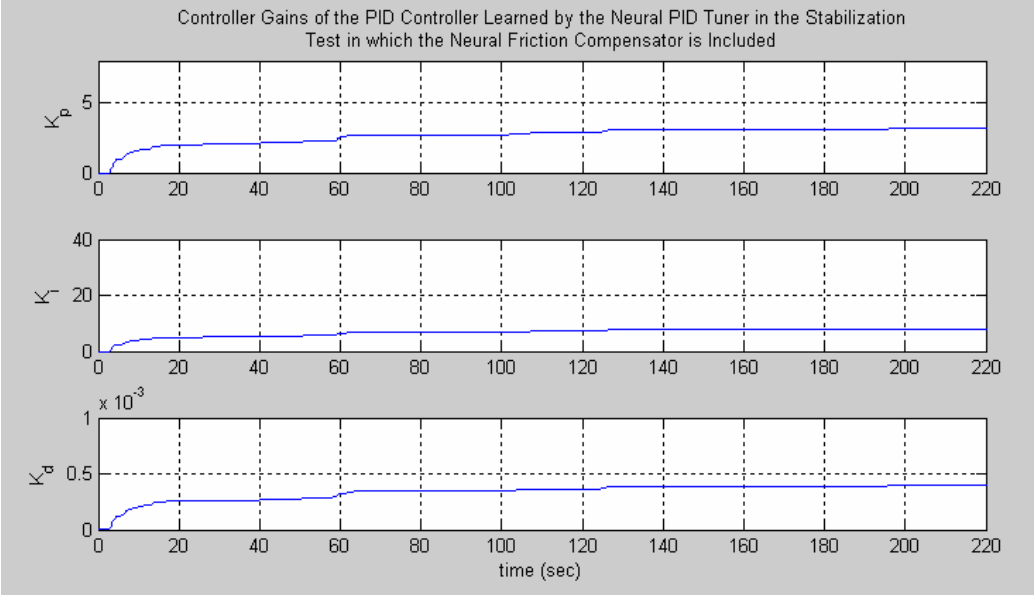

Figure 5.35 – Controller Gains of the PID Controller Tuned during the Stabilization Tests in which the Neural Friction Compensator is Included

The stabilization performance obtained as a result of this test is given in Figure 5.36. As seen from the figure, the stabilization performance at the beginning of the test is approximately 5.2 mrad and this value decreases to 1.0 mrad at the end of

the test. This increase in the stabilization performance is due to the friction compensator used with the neural PID tuner.

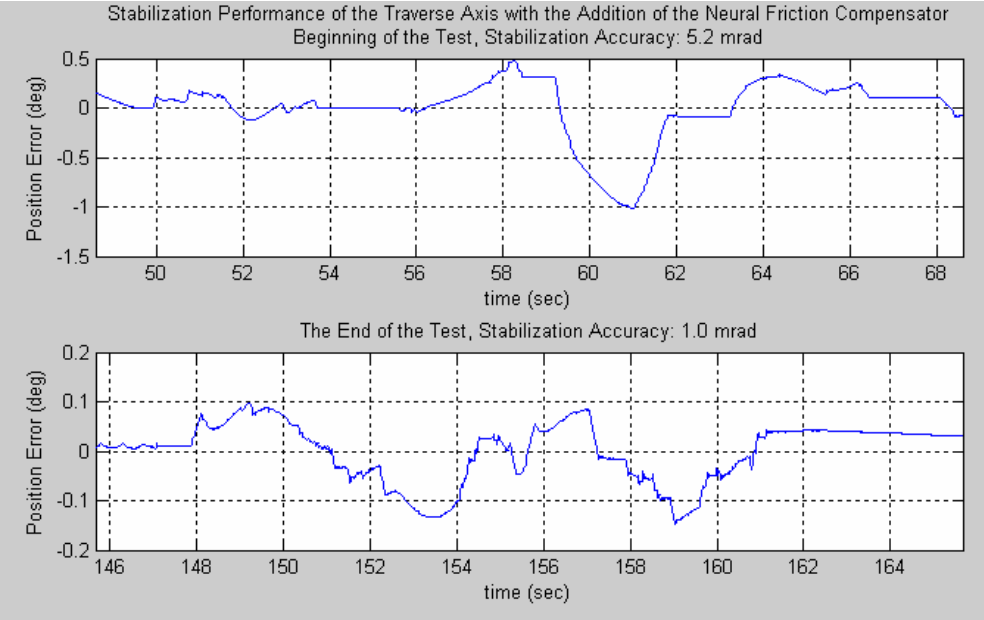

Figure 5.36 – Stabilization Performance of the Traverse Axis Including the Proposed Neural Friction Compensator

Considering the results of the stabilization test performed for the case with only PID tuner, the stabilization performance at the end of the test is 5 times better with the addition of the neural friction compensator to the controller. In addition, much better stabilization accuracy values are obtained with almost half of the PID controller gains.

The estimated Columb friction in this test is given in Figure 5.37. As seen from the figure, the friction compensation torque applied to the system has approximately 1.7 volts and the instantaneous values changes according to the turret position at which the friction changes its sign and the acceleration of the system at this point.

As a result of the tests using the experimental setup, the stabilization accuracy becomes approximately 1.0 mrad with the training of the neural PID tuner and the neural friction compensator. If the neural friction compensator is not used, the stabilization performance is 5 times worse and the PID controller gains at the end

of the test are approximately 2 times greater considering the use of the neural friction compensator. Results are given in Figure 5.33.

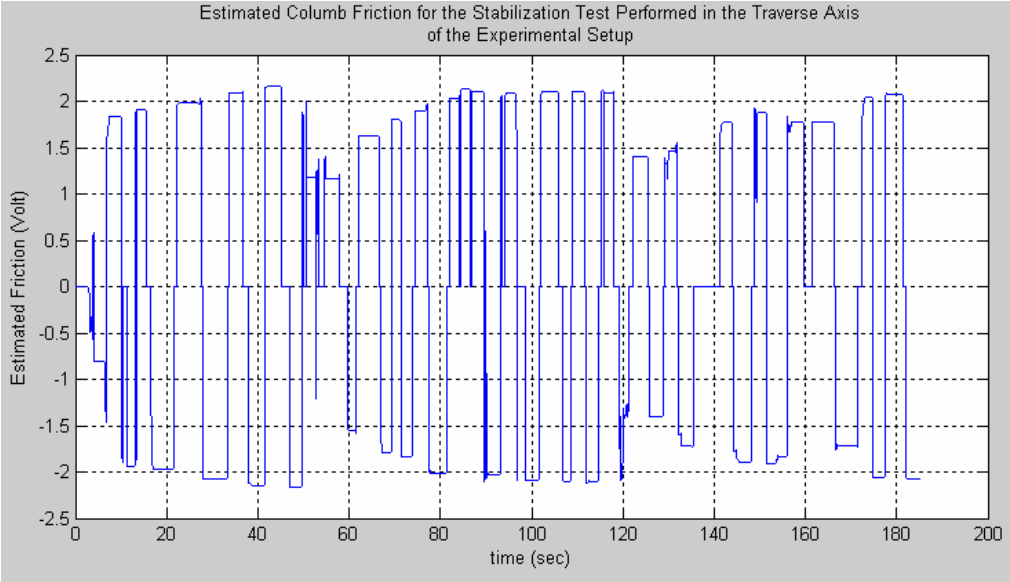

Figure 5.37 – Estimated Columb Friction in the Stabilization Test Performed for the Traverse Axis of the Experimental Setup

The estimated Columb friction value is not constant because of the changing frictional characteristics of the system with the turret position and acceleration of the motion.

In this chapter, the results of the computer simulations on the mathematical model of a battle tank with the network parameters designed in chapter 3 and the results of the test performed on the hardware experimental setup are given and discussed in detail. In the next section, the sensitivity of the linear system identifier network to the sensor noise and quantization errors and performance analysis of the proposed controller will be given.

CHAPTER 6

SENSITIVITY AND PERFORMANCE ANALYSIS

The proposed controller architecture is both applied to the computer simulations and applied on the experimental setup. In the computer simulations, sensors and actuators were ideal. On the other hand, there were sensor noise and quantization errors in the experimental setup.

In this section, the sensitivity of the proposed networks to sensor noises will be examined. Then, the performance of the proposed controller architecture will be explored.

In sensitivity analysis part of this section, the sensitivity of the estimated system response, and the output of the neural linear system identifier to sensor noises and quantization errors will be analyzed. This analysis will be given only for the neural linear system identifier network because of the simplicity of the network. Other two networks (neural PID tuner and neural friction compensator) are very similar to the neural linear system identifier network and the same approach can be used to analyze these networks. Therefore the sensitivity analysis of these networks is left as a future work.

The performance of the proposed controller architecture will be analyzed using the mathematical model of the main battle tank. The critical parameters of the system, which can be changed in practice, namely the Columb friction, disturbance characteristics and inertia, are changed slightly during simulations and the adaptation of the controller is analyzed using the results of these simulations.

6.1 Sensitivity Analysis of the Neural Linear System Identifier Network for Sensor Noise and Quantization Errors

In order to estimate the system dynamics of the controlled system, a linear artificial neural network is proposed and the details of this network are given in the section 3. In this part of the report, the sensitivity of the estimated system response output of the network to the sensor noise and quantization errors in digitization will be analyzed.

In computer simulations reported in chapter 5, the sensors and actuators were ideal. In other words, there were no noise on the sensors and the control signals are ideally transformed to actuator motions. In this chapter, the effect of these sensor noise and quantization errors on the estimated system response will be determined.

In practice, sensor manufacturers specify an upper bound for the measurement errors of the corresponding sensor. In this analysis, the upper bound given for the feedback gyroscope used in the experimental setup will be used.

For the actuation of the system, an analog signal is generated for the servo motor driver. The quantization error of the experimental setup in the digital-to-analog conversion will be used in these analyses.

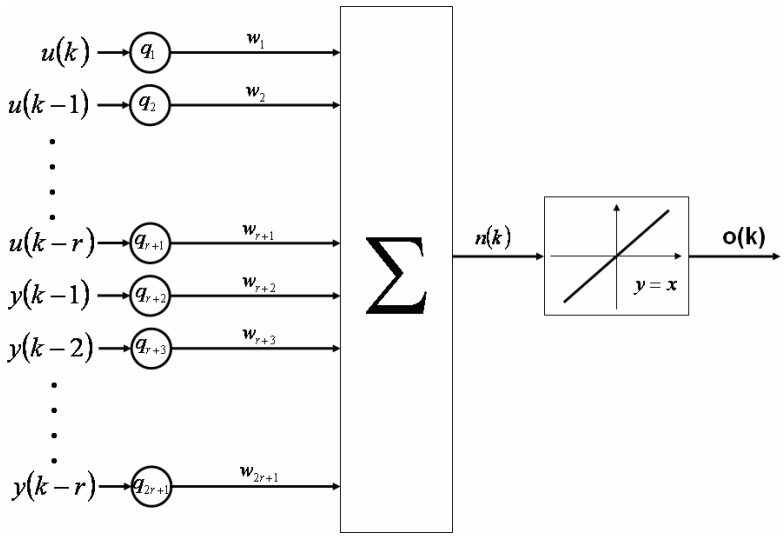

Figure 6.1 – Structure of the Proposed Neural Linear System Identifier

The structure of the proposed neural linear system identifier is given in Figure 6.1. In this figure, the inputs of the network are the present and previous values of the system inputs ($u(k)$, $u(k-1)$, …, $u(k-r)$) and the previous system responses ($y(k-1)$, …, $y(k-r)$). The network outputs the estimated system response ($o(k)$) of the present time.

Let us define sensor noise on the system response as Δy and quantization error on the actuation signal as Δu. Therefore, the actual values of the network inputs can be defined as $\bar{u}(k) = u(k) + \Delta u(k)$, …, $\bar{u}(k-r) = u(k-r) + \Delta u(k-r)$ and $\bar{y}(k-1) = y(k-1) + \Delta y(k-1)$, …, $\bar{y}(k-r) = y(k-r) + \Delta y(k-r)$ and the desired output of the network can be defined as $\bar{o}(k) = o(k) + \Delta o(k)$ where $\Delta o(k)$ is the error due to the sensor noise and quantization error.

The output error of the network $\Delta o(k)$ can be rewritten as;

$$\Delta o(k) = \bar{o}(k) - o(k) \tag{6.1}$$

this can be expanded as;

$$\begin{aligned} \Delta o(k) = {} & f(\bar{u}(k), \bar{u}(k-1), ..., \bar{u}(k-r), \bar{y}(k-1), ..., \bar{y}(k-r)) \\ & - f(u(k), u(k-1), ..., u(k-r), y(k-1), ..., y(k-r)) \end{aligned} \tag{6.2}$$

Here in this equation, $f(\bullet)$ represents the activation function of the network, which is a pure linear $y = x$ function.

The Taylor expansion of the first term in equation 6.2 around $u(k)$, …, $u(k-r)$, $y(k-1)$, …, $y(k-r)$ is given as;

$$\begin{aligned} f(\bar{u}(k), ..., \bar{u}(k-r), \bar{y}(k), ..., \bar{y}(k-r)) = {} & f(u(k) ..., u(k-r), y(k), ..., y(k-r)) \\ & + \frac{\partial f}{\partial u(k)} (\bar{u}(k) - u(k)) + ... \\ & + \frac{\partial f}{\partial y(k-1)} (\bar{y}(k-1) - y(k-1)) + ... \\ & + H.O.T. \end{aligned} \tag{6.3}$$

Neglecting the higher order terms (H.O.T.) of the expansion and putting the Taylor expansion of the first term in its place in equation 6.2 gives the following equality;

$$\begin{aligned}\Delta o(k) = &\frac{\partial f}{\partial u(k)}\Delta u(k) + ... + \frac{\partial f}{\partial u(k-r)}\Delta u(k-r) \\ &+ \frac{\partial f}{\partial y(k-1)}\Delta y(k-1) + ... + \frac{\partial f}{\partial y(k-r)}\Delta y(k-r)\end{aligned} \tag{6.4}$$

As mentioned before, the activation function is actually a pure linear function of the multiplication of network weights and network inputs.

$$\begin{aligned}f(u(k),...,u(k-r), y(k),..., y(k-r)) = &w_1 u(k) + ... + w_{r+1} u(k-r) \\ &+ w_{r+2} y(k-1) + ... + w_{2r+1} y(k-r)\end{aligned} \tag{6.5}$$

Therefore, the partial derivative terms given in equation 6.4 are the network weights of the neural linear system identifier. Rewriting equation 6.4 gives the following equation;

$$\Delta o(k) = w_1 \Delta \hat{u}(k) + ... + w_{r+1} \Delta u(k-r) + w_{r+2} \Delta y(k-1) + ... + w_{2r+1} \Delta y(k-r) \tag{6.6}$$

After obtaining this equation, two different approaches are used in the analysis. First approach is the maximum error approach and second one is a statistical approach.

In the maximum error approach, the errors in the inputs due to sensor noise and quantization errors are assumed as $|\Delta u(k)| \le \Delta_1$, $|\Delta u(k-1)| \le \Delta_2$, …, $|\Delta u(k-r)| \le \Delta_{r+1}$, $|\Delta y(k-1)| \le \Delta_{r+2}$, …, $|\Delta y(k-r)| \le \Delta_{2r+1}$. Using this assumption, the maximum error due to the errors in the input can be found as;

$$\Delta = |w_1|\Delta_1 + |w_2|\Delta_2 + ... + |w_{r+1}|\Delta_{r+1} + |w_{r+1}|\Delta_{r+2} + ... + |w_{2r+1}|\Delta_{2r+1} \tag{6.7}$$

Other approach is based on the statistical characteristics of the input errors. In this approach, the errors in the network inputs are assumed as normally distributed with 0 average and known standard deviation. In this approach, the errors in the

inputs are assumed independent of each other. Using these assumptions, the variance of the error in the network output can be found as;

$$V[o(k)] = (w_1)^2 \sigma_1^2 + ... + (w_{2r+1})^2 \sigma_{2r+1}^2 \qquad (6.8)$$

Here in this equation, the parameters σ_1, …, σ_{2r+1} are the standard deviations of the errors on $u(k)$, …, $u(k-r)$, $y(k-1)$, …, $y(k-r)$ respectively.

After obtaining these equations, the numerical values will be put into the equations in order to obtain the output error of the network and compare these error amplitudes with the network output amplitudes.

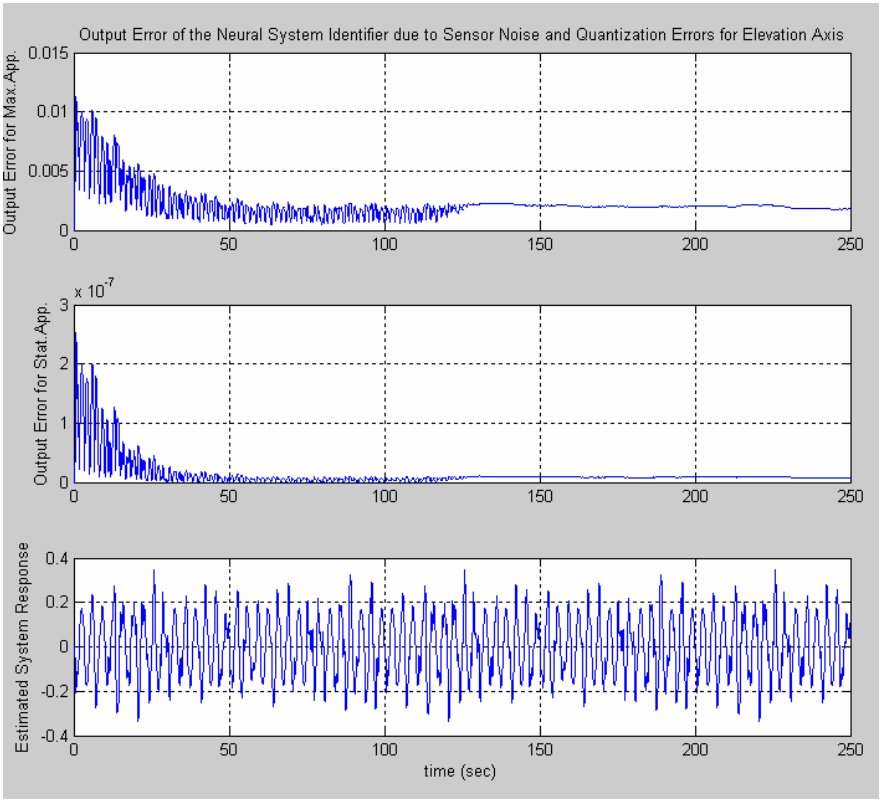

Figure 6.2 – Output Error of the Proposed Neural Linear System Identifier due to the Sensor Noise and Quantization Errors in the D/A Conversion for the Elevation Axis

The maximum error of the feedback gyroscope used in the experimental setup is 0.95 degrees/second and the standard deviation of the sensor noise is approximately 0.14 degrees/second. The control computer analog output ports

have 12-bit digital-to-analog converters therefore the maximum quantization error at these D/A conversion is $\frac{20}{2^{13}} = 0.0025$ volts since the analog output of the computer can produce voltages between $-10\,\mathrm{V}$ and $10\,\mathrm{V}$. Since the quantization error distribution is uniform and have zero mean, the standard deviation of the quantization error is 0.

Putting these numeric values into the equations 6.7 and 6.8, the maximum output error and standard deviation of the output error are given in Figure 6.2. In this figure, the output of the network is also given. Comparing the network output and the output error values given in the figure, the errors on the network output due to the sensor noise and quantization errors are less than 1% of the network output. This percentage is given for the maximum error approach case. The percentage of the output error with respect to the network output for the statistical approach case is really very low, even less than 0.00005% of the network output.

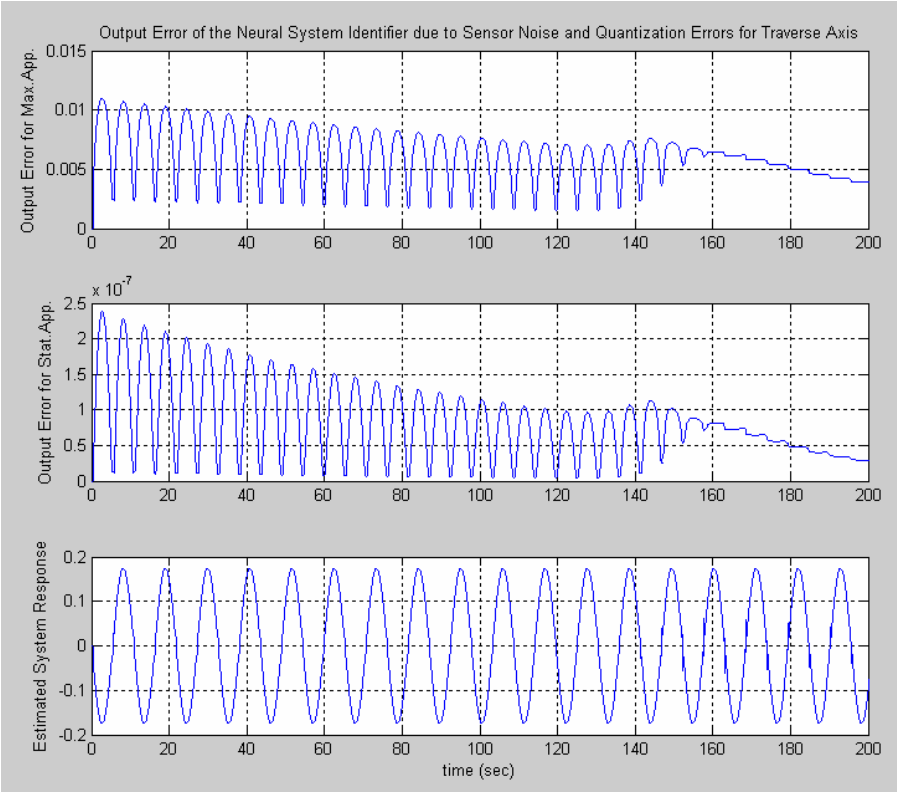

Figure 6.3 - Output Error of the Proposed Neural Linear System Identifier due to the Sensor Noise and Quantization Errors in the D/A Conversion for the Traverse Axis

The situation is not very different than the elevation axis for the traverse axis. The output error of the network for the traverse axis is given in Figure 6.3. Again the error in the network output due to the sensor noise and quantization errors are very low compared to the estimated system response output of the network. The percentage error on the network output is approximately 0.75% for the maximum error approach case and this value decreases to almost zero for the statistical approach case.

6.2 Performance Analysis of the Proposed Neural Controller

In this section, the performance of the proposed neural controller architecture will be analyzed by performing computer simulations with the mathematical model and performing tests on the hardware experimental setup.

The performance analysis of the proposed controller will be conducted in three stages. In the first stage, the performance of the proposed neural PID tuner will be analyzed using the mathematical setup with varying disturbance characteristics and varying plant dynamics. In the second stage, the performance of the proposed neural friction compensator will be analyzed using the mathematical model with varying frictional characteristics. In the last stage, the performance of the whole proposed neural controller will be analyzed using the hardware experimental setup.

6.2.1 Performance of the Proposed Neural PID Tuner on the Mathematical Model

The performance of the proposed neural PID tuner is analyzed using the mathematical model of the main battle tank. In these analyses, the disturbance characteristics of the system are changed with time by increasing the speed of the tank over the APG track. The same analysis is repeated by changing inertia of the turret by either increasing or decreasing it.

In the first part, the tank goes over the APG track with 10 km/h speed. After some time period, the tank speeds up to 30 km/h and finally the tank speeds up to 40 km/h speed. The disturbance characteristics on the system during the simulations

for the elevation axis are given in Figure 6.4. As seen from the figure, the disturbance speed on the elevation axis increases considerably with the increasing tank speed.

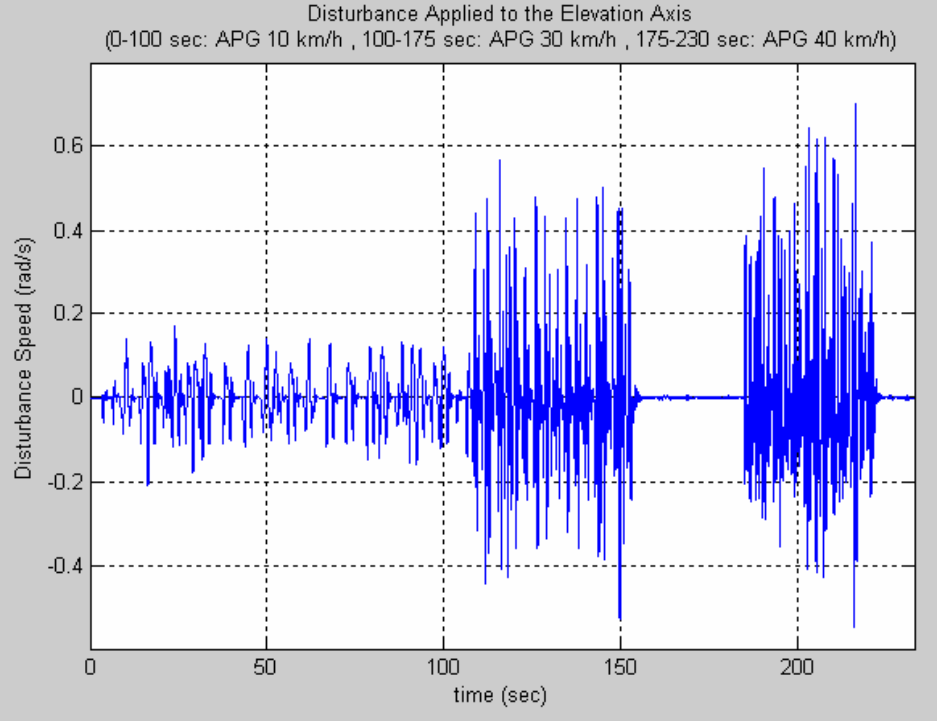

Figure 6.4 – The Disturbance Speed Excerpted on the Elevation Axis for the Performance Analysis of the Neural PID Tuner

With this disturbance applied to the elevation axis, the stabilization performance of the elevation axis is given in Figure 6.5. As seen from the figure, the stabilization performance decreases a little with the increasing tank speed. The disturbance accuracy is 0.30 mrad for the tank speed 10 km/h and becomes 0.32 mrad for the tank speeds 30 km/h and 40 km/h. The performance decrease for this simulation is 6.7% for both 30 km/h and 40 km/h tank speeds.

The same simulation is performed by fixing the PID parameters after a training period on the APG track with 10 km/h tank speed. The stabilization performance of the elevation axis is given in Figure 6.6. As seen from the figure, the stabilization performance decreases sharply with the increasing tank speed. For the tank speed 30 km/h, the stabilization accuracy is obtained as 0.78 mrad and for the 40 km/h speed, the value is obtained as 1.43 mrad. In other words, the stabilization

performance decreases approximately 250% for the tank speed 30 km/h and %500 for the tank speed 40 km/h. As a result, the neural PID tuner configures the controller parameters so that the stabilization performance is kept satisfactory for all tank speeds.

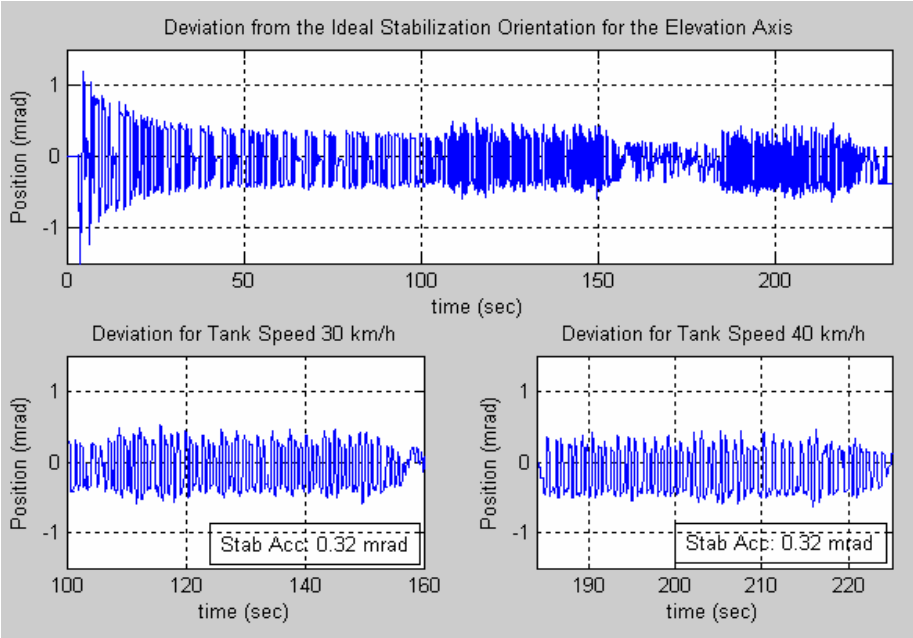

Figure 6.5 – Stabilization Performance of the Elevation Axis for Different Tank Speed on the APG Track

After analyzing the neural PID tuner performance for different tank speed on APG track, the performance of the network is analyzed with varying the inertia of the plant. In this analysis, the inertia of the system increases incrementally approximately 300% after a time period. In the first 100 seconds, the PID tuner is trained with the original plant inertia and the inertia is increases 300% between time range 100 and 150 seconds. The stabilization performance of the elevation axis is given in Figure 6.7. As seen from the figure, the stabilization accuracy value becomes 3 times greater than the stabilization accuracy value obtained for the original plant inertia. In other words, the stabilization accuracy for the original plant inertia was 0.30 mrad and increased 200% for the 300% increased inertia and become 0.88 mrad.

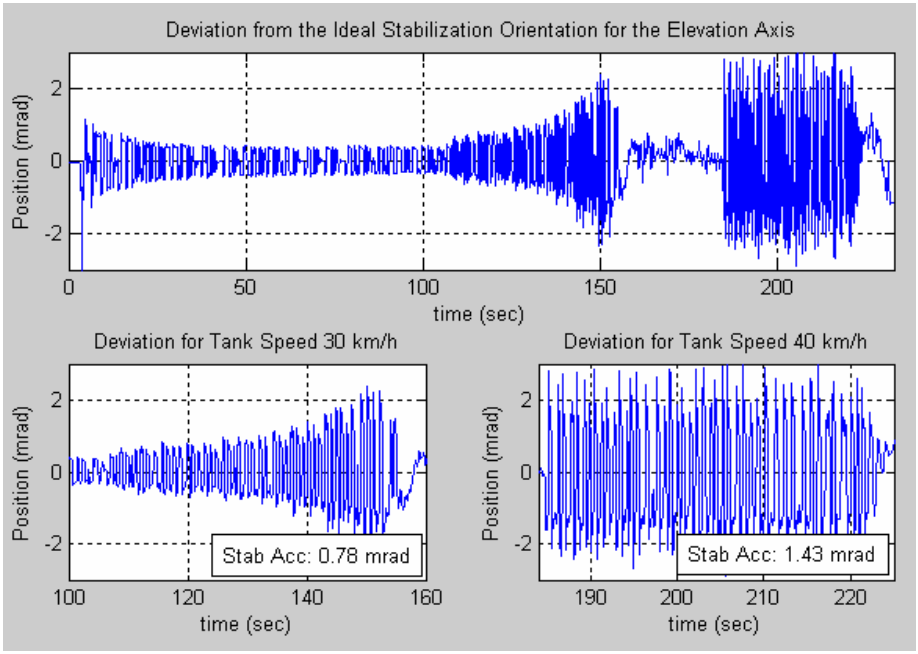

Figure 6.6 - Stabilization Performance of the Elevation Axis for Different Tank Speed on the APG Track (Fixed PID Parameters)

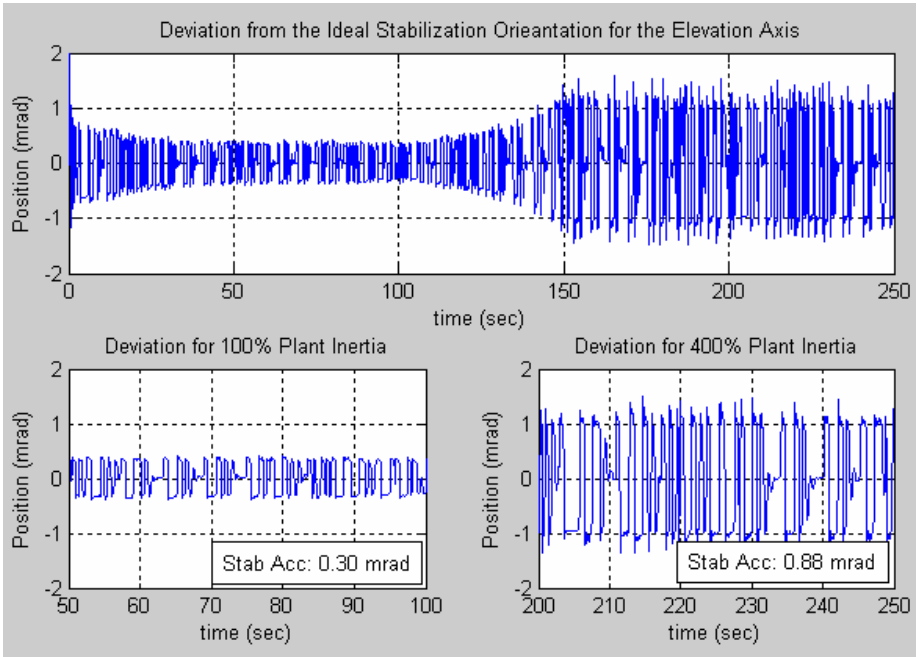

Figure 6.7 - Stabilization Performance of the Elevation Axis for Varying Elevation Axis Inertia on the APG Track

The same simulation is repeated for the fixed PID parameters. The PID parameters are tuned during the first 100 seconds and fixed after this time step. With the

increasing plant inertia, the stabilization performance is obtained as shown in Figure 6.8. As seen from the figure, the stabilization performance decreased approximately 370% for the fixed PID parameters.

To sum up, the stabilization performance increases 70% with the proposed neural PID tuner considering the fixed PID parameter case. This performance increase is relatively small considering the previous analysis for different tank speeds and this is mainly due to the high inertia change and limited actuation power of the plant.

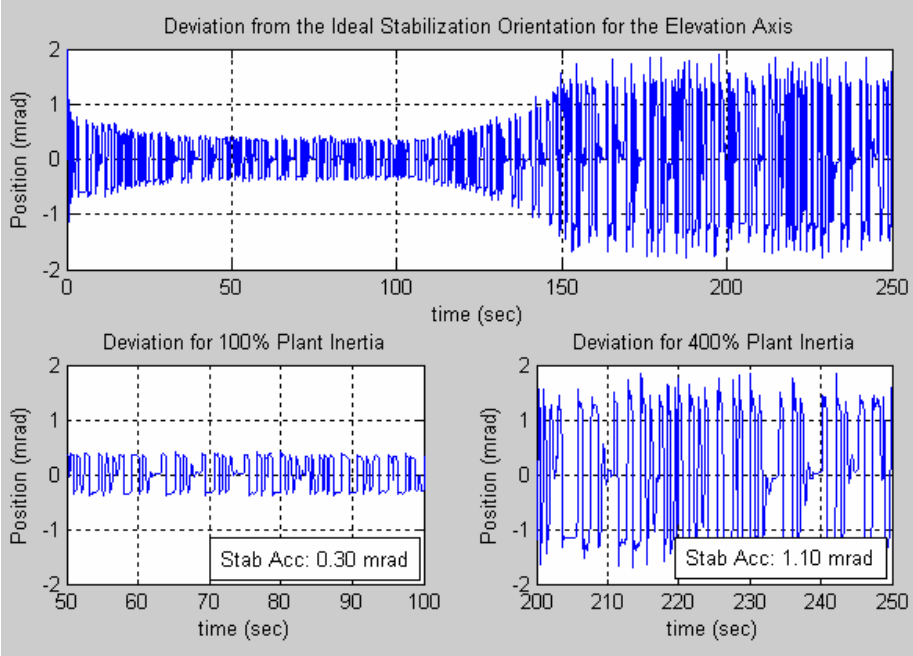

Figure 6.8 - Stabilization Performance of the Elevation Axis for Varying Elevation Axis Inertia on the APG Track (Fixed PID Parameters)

The inertia of the system decreased in computer simulations but the results obtained for fixed PID parameter case and usage of PID tuner case is not so different because the PID parameters freeze with the decrease of the plant inertia. Therefore, the proposed neural PID tuner does not increase the stabilization performance of the system but the stabilization performance increases as a result of the decrease in the plant inertia.

6.2.2 Performance Analysis of the Proposed Neural Friction Compensator on the Mathematical Model

The performance of the proposed neural friction compensator is analyzed on the mathematical model of the main battle tank by changing suddenly the Columb friction of the corresponding axis. Because of the similarity of the elevation and traverse axis considering the actuation mechanisms and controller architectures, the results are shown only for the elevation axis.

In the simulations performed for the elevation axis in order to analyze the performance of the friction compensation algorithm, the tank goes over the APG track with a speed of 10 km/h. While the tank goes over the track, the friction value changes suddenly. After this sudden change, an error occurs between the emulated virtual linear system response and the real system with friction. This error leads the neural friction compensator to learn the new friction characteristics.

In the first part of the analysis, the friction existing on the elevation axis is decreased by 25% after the neural friction compensator reaches the steady state for the original friction characteristics. After this decrease, the friction value is decreased by 25% more (totally 50%). The results of this simulation are given in Figure 6.9.

As seen from the figure, the neural friction compensator adapts itself to the new friction characteristics in a short time. For the first decrease of the friction value, the estimated friction value decreases gradually to the new friction value and the estimated friction value reaches to the steady state in approximately 20 seconds. For the second decrease of the friction value, the response is faster and the estimated friction value reaches to steady state approximately in 5 seconds. Considering the real life applications, these time periods needed to learn the new frictional characteristics are considered to be very short.

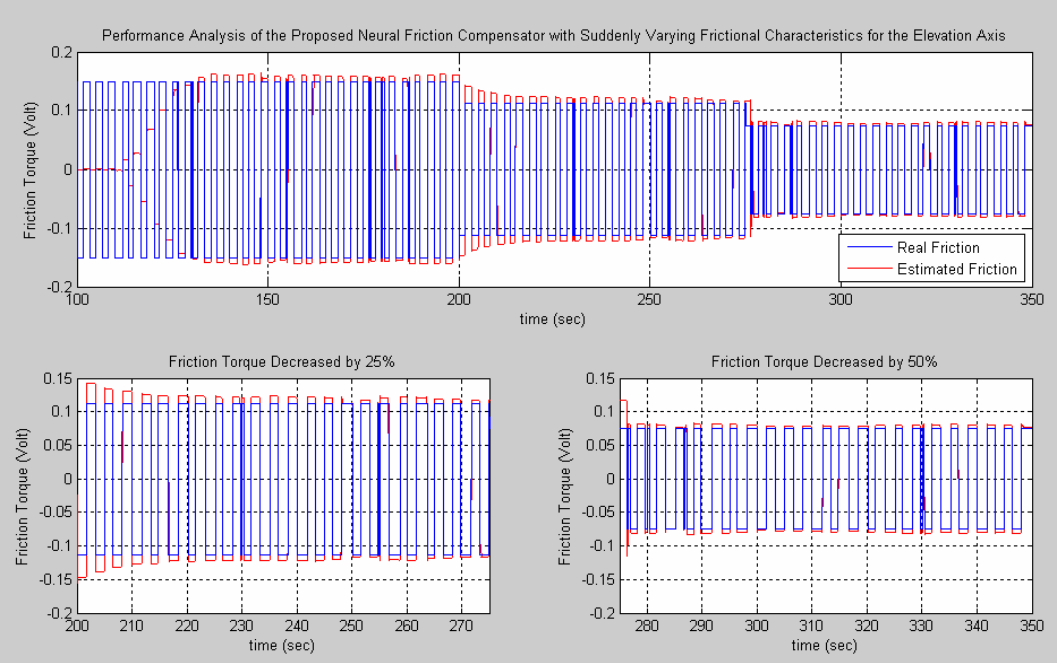

Figure 6.9 – Performance Analysis of the Proposed Neural Friction Compensator for Decreasing Frictional Effects

In the second part of the analysis, the friction value is increased suddenly by 50% after the estimated friction value reached to steady state for the original friction value. The friction value is increased by 50% more (totally 100%) after the adaptation of the friction compensator to the new frictional characteristics. The results of the simulation are given in Figure 6.10.

The results are very similar to the case with decreasing friction values. Again the proposed network learns the new frictional characteristics in a very short time. For the first increase, the time needed to learn the new friction value is approximately 35 seconds and it becomes 30 seconds for the seconds increase.

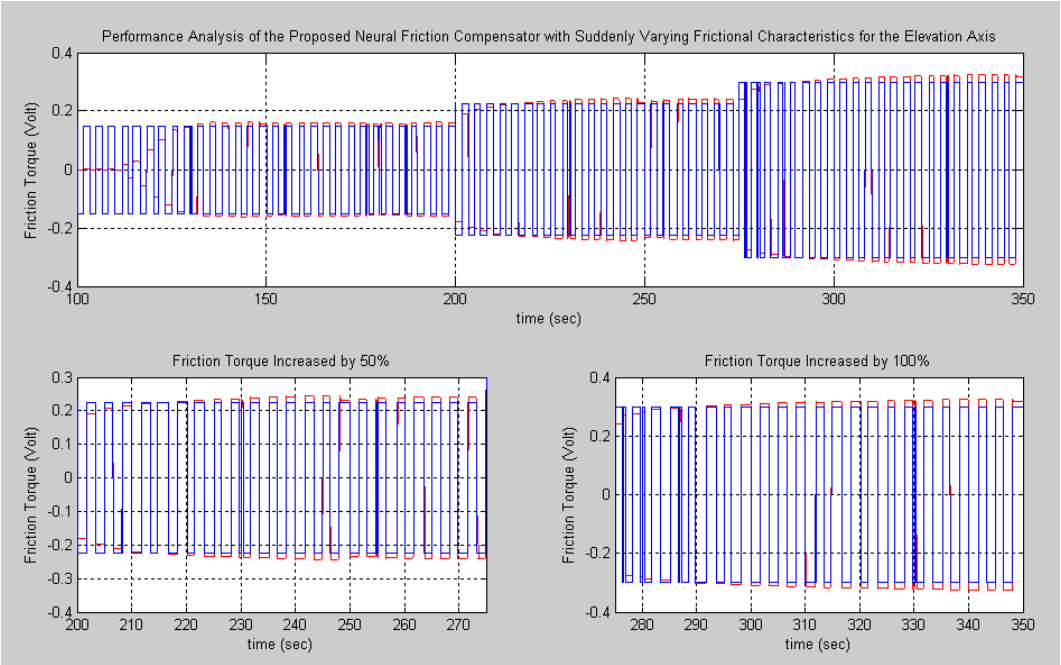

Figure 6.10 - Performance Analysis of the Proposed Neural Friction Compensator for Increasing Frictional Effects (Blue: Real Friction, Red: Estimated Friction)

As a result of these analyses, the proposed neural friction compensator adapts itself to new frictional characteristics even for sudden changes in the frictional effects. The time periods needed for this adaptation are really promising considering the amplitude of the increase in the friction values. With this adaptation, the PID controller is not affected by the changing friction in the system and the stabilization performance of the system is kept steady for changing frictional characteristics.

6.2.3 Performance of the Proposed Neural Controller Architecture on the Experimental Setup

The results of the proposed neural controller architecture consisting of the neural PID tuner, neural linear system identifier and the neural friction compensator are given in the previous section. In this section, the performance of the proposed neural controller will be analyzed for different speed command signal shapes and signal frequencies. In these tests, the vehicle will remain stationary and the turret will move. The feedback gyroscope will be used as the feedback sensor.

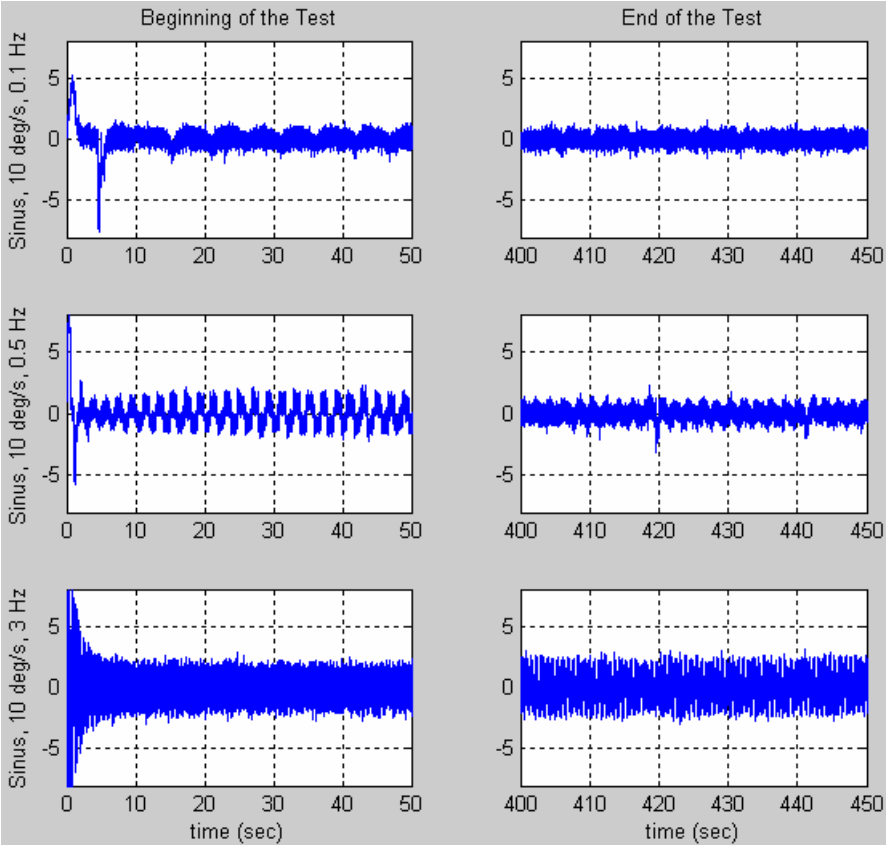

Figure 6.11 – Tracking Speed Errors of the Test for a Sinus Speed Command with 10 deg/s Amplitude for the Traverse Axis of the Experimental Setup

In the first part of the analysis, a sinus speed command is applied to the controller with amplitude of 10 deg/s but the frequency of the signal increased. The tracking speed errors obtained in these tests are given in Figure 6.11. As seen from the figure, for frequency of 0.1 Hz, the tracking error becomes almost zero at the end of the test. This is not so clear in the figure but after the elimination of the sensor noise, the maximum tracking error becomes approximately 0.4 deg/s for 0.1 Hz frequency. This maximum tracking error at the end of the test is kept as 0.4 deg/s for the 0.5 Hz frequency but it becomes approximately 2 deg/s for the frequency of 3 Hz.

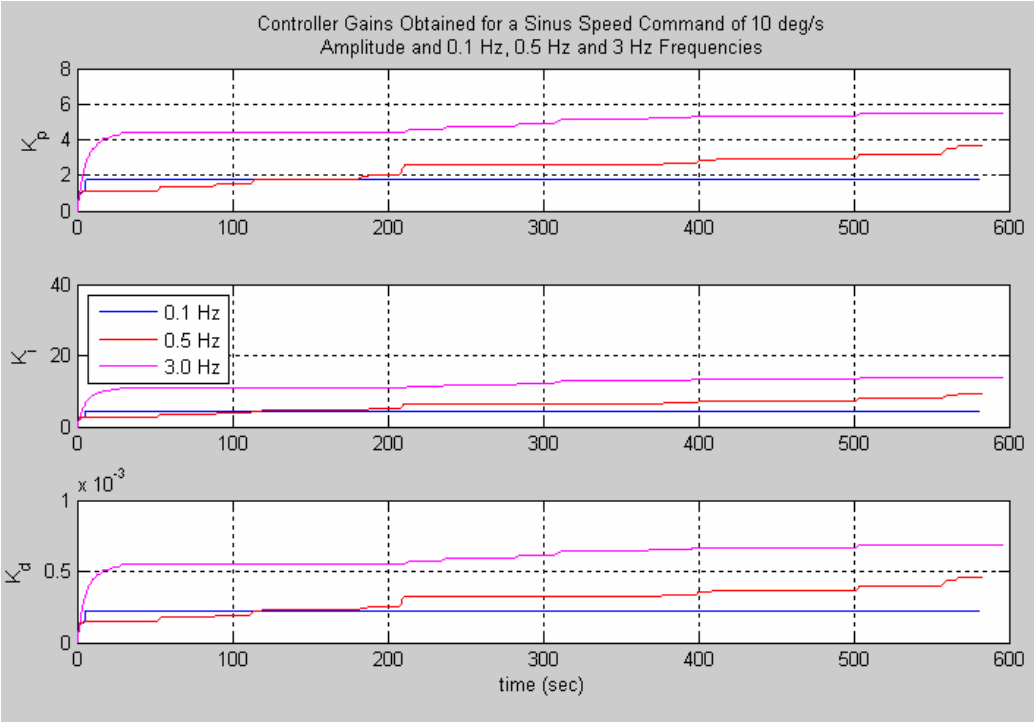

Figure 6.12 – Controller Gains Tuned by the Neural PID Tuner for a Sinus Speed Command of 10 deg/s Amplitude and Different Frequencies for the Traverse Axis of the Experimental Setup

The controller gains obtained in these tests are given in Figure 6.12. As seen from the figure, the controller gains for the lowest frequency 0.1 Hz are relatively low and reached the steady state values just at the beginning of the test. For the frequency value of 0.5 Hz, the controller gains are higher. On the other hand, the controller gains are increased fast and almost reach their steady state values in the beginning of the test for the frequency value of 3 Hz. For all of the frequency values, the controller gains do not reach to their limit values. For the frequency values of 0.5 Hz and 3 Hz, the controller gains do not reach to their steady state values in 600 seconds.

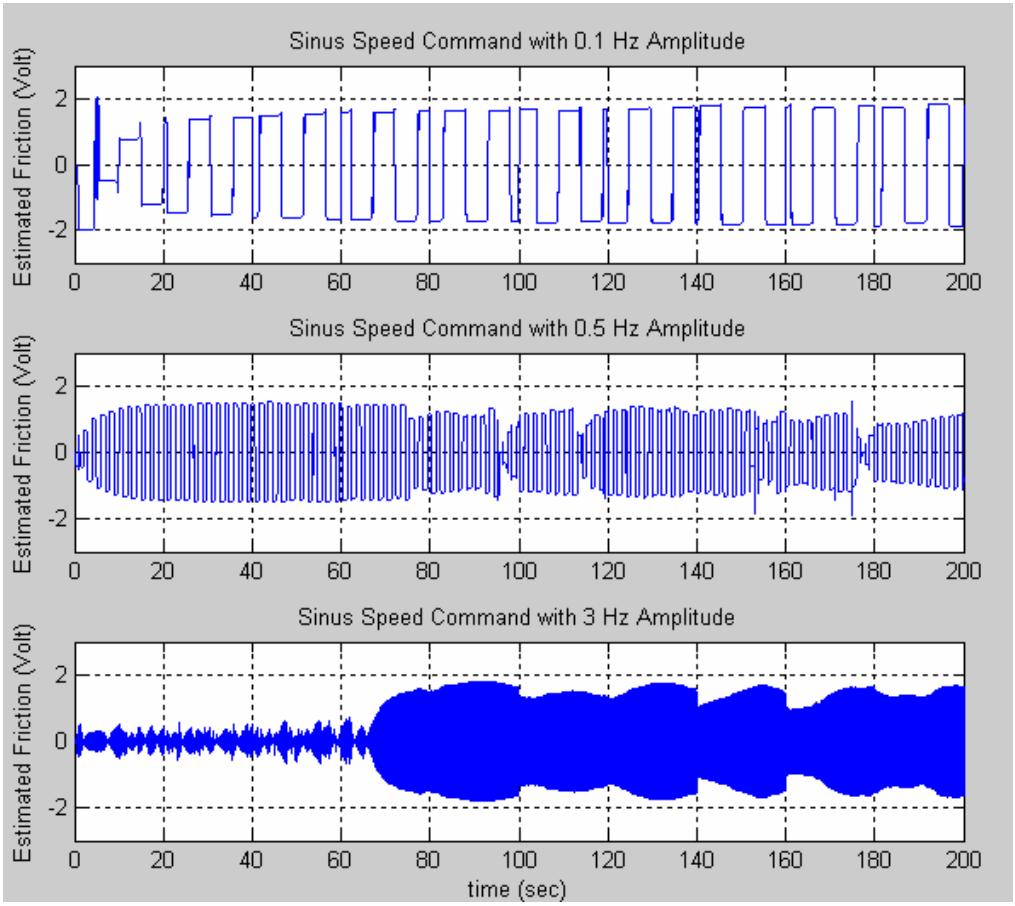

Figure 6.13 – Estimated Columb Friction for the Sinus Speed Command of 10 deg/s Amplitude and Different Frequencies for the Traverse Axis of the Experimental Setup

The estimated friction obtained from these tests is given in Figure 6.13. For the frequency value of 0.1 Hz, the friction characteristic is learned approximately in 60 seconds. Although the learning time is less for the frequency of 0.5 Hz, it shows fluctuations after some point. These fluctuations may be due to the change in the position of the turret where the friction changed its sign. On the other hand, the time period needed to learn the friction characteristics of the system is larger for the frequency value of 3 Hz. In addition, the estimated friction value fluctuates around the steady state estimation value. The increase of the learning time is mainly due to short training time periods created by the fast changing speed of the signal.

As a summary of these tests on the experimental setup, increasing the frequency of the signal decreases the tracking performance and leads to higher controller gains. In addition, the estimated friction is affected by the frequency of the motion and fluctuations occurred in the estimated friction value. In addition, the time needed to learn the friction value increased due to the decrease in the training time periods of the neural friction compensator.

After analyzing the effect of the frequency of the speed command signal on the performance of the proposed neural controller architecture, the effect of the signal shape will be analyzed. For this purpose, tests are performed with block signals with amplitude of 10 deg/s and frequencies of 0.1 Hz and 3 Hz.

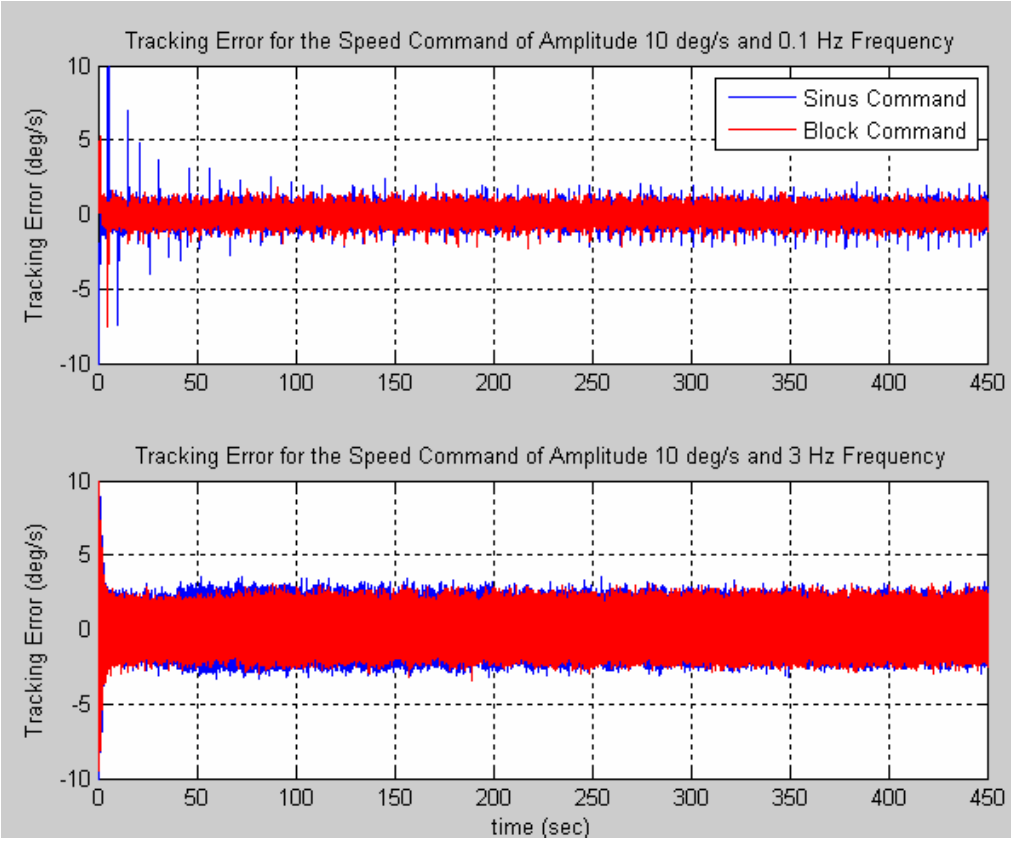

Figure 6.14 – Tracking Speed Errors for the Speed Command of 10 deg/s Amplitude and Different Frequencies for Different Signal Shapes for the Traverse Axis of the Experimental Setup

The results for the tracking error both for sinus and block speed commands for frequencies of 0.1 Hz and 3 Hz are given in Figure 6.14. As seen from the figure,

for the frequency value of 0.1 Hz, the tracking error is larger in the beginning of the test but the tracking error becomes almost the same at the end of the test. For the frequency value of 3 Hz, the tracking errors obtained for both sinus and block speed commands are almost the same throughout the whole test.

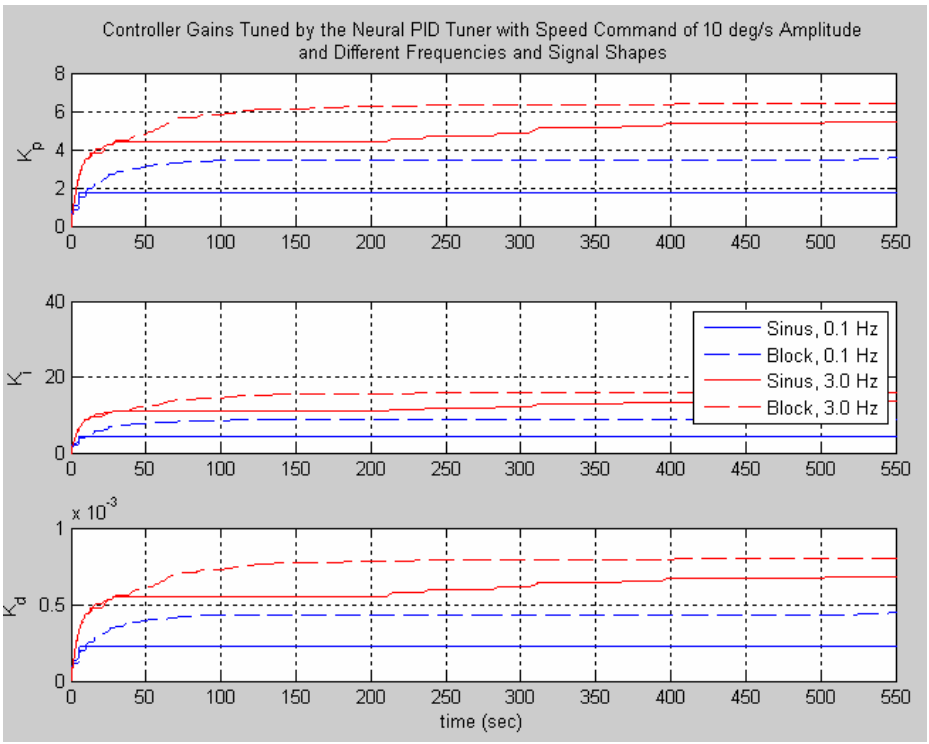

Figure 6.15 – Controller Parameters Tuned by the Neural PID Tuner with Speed Command of 10 deg/s Amplitude and Different Frequencies and Signal Shapes

The controller parameters obtained as a result of these tests are given in Figure 6.15. As seen from the figure, the controller gains increases from sinus signal to the block signal shape. For all of the speed command cases, the PID controller gains reaches almost their steady state values in 550 seconds. In addition, the controller gains do not reach their limit values for all of the cases therefore there is still some margin for the increase of the PID controller gains.

The results showing the estimated friction of the traverse axis in these tests are given in Figure 6.16. As seen from the figure, for the frequency of 0.1 Hz, the time period needed to learn the friction value increases from 60 seconds to 200 seconds as a result of the decrease in the training time period. Since the speed command is

a block signal, the neural friction compensator is trained for a very short time since the neural friction compensator is trained only when the relative speed of the turret is below a threshold.

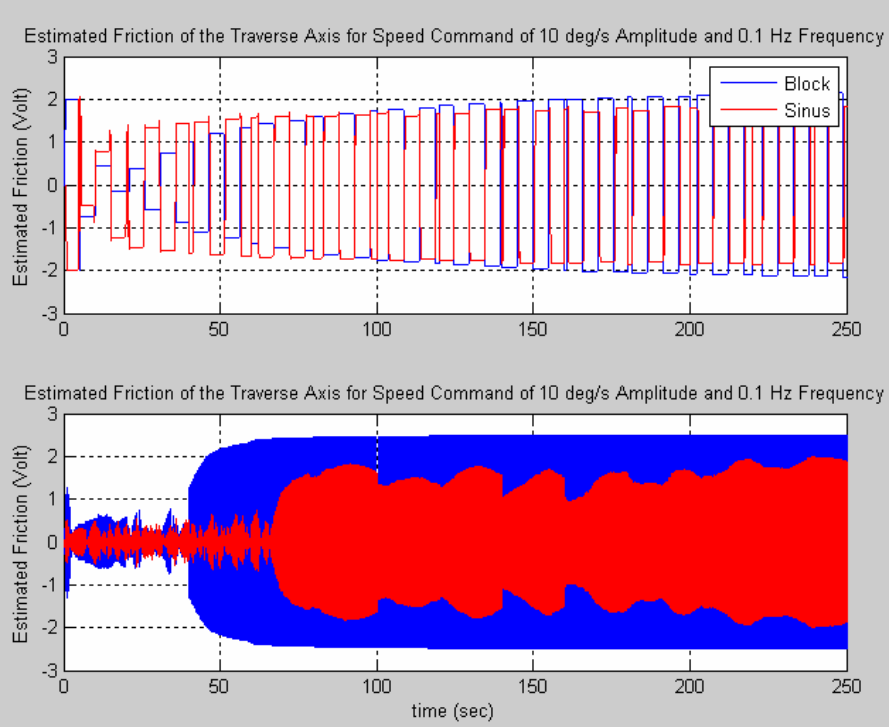

Figure 6.16 – Estimated Friction Values of the Traverse Axis of the Experimental Setup for Different Signal Shapes and Different Signal Frequencies with Amplitude of 10 deg/s

Considering the case with frequency value of 3 Hz, the time period needed to learn the friction value for the block speed command decreases relative to the case with sinus speed command. However, the steady state value of the estimated friction value is a little higher than the Columb friction that exists on the real physical system.

As a result of the tests performed with different signal shapes, the performance of the neural friction compensator is affected by the signal shape. The time needed to train the neural friction compensator is dependent on the relative speed threshold and therefore longer time periods in this training region makes the learning process faster.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The objective of this study is stated to develop an intelligent controller for the stabilization of turret subsystems under disturbances from unstructured terrain as the environmental conditions and the dynamics of the mechanical system changes. The main motivation is to handle the varying disturbance characteristics from terrain to terrain and the dynamical changes of the mechanical system for instance the Columb friction from hot environment to cold environment.

A very important criterion of this study is to keep the delevoped controller architecture simple in order to ensure the real-time implementation of the controller. For this purpose, it is decided to base the controller to a classical PID controller but further equipping it with intelligent algorithms are used to tune its controller gains.

As a result of a literature survey, a backpropagation network is used to tune the controller gains of this PID controller. This network mainly aims to minimize the error between the commanded signal and the feedback signal and it can be thought as an optimal controller, which tracks a given signal with minimum error. This PID based neural controller did not provide enough satisfaction in its stabilization performance therefore additional algorithms have been developed in order to enhance the performance of the controller.

The main problem of the controller was the system non-linearities, one of the non-linearities of the system being the unbalance in the elevation axis. This unbalance is compensated using a simple algorithm with the help of a gravity sensitive accelerometer that is placed at the center of rotation of the elevation axis.

The second non-linearity of the system was the Columb friction. This non-linearity was rather complicated therefore and intelligent algorithm is used to compensate for this non-linearity. A backpropagation neural network is decided to use for the compensation of the Columb friction. This network was very similar to the network used for the tuning of the PID controller.

This network aims to compensate the Columb friction by adding a correction signal to the output of the PID controller. This is done simply by identifying the system dynamics and finding the PID controller output if the system was an ideal linear system. In order to find the output of the PID controller for the linear system, a virtual linear system is emulated using the identified system. The neural friction compensator aims to make the output of the PID controller closer to the output of the PID controller for the linearized system emulation.

The proposed neural controller is applied to both a mathematical simulation model of a main battle tank namely the Leopard 1 tank with detailed vehicle, suspension and turret models and to hardware experimental setup. The hardware is a 1/6 true scale model of Challenger 2 tank and the mechanics and the actuation mechanism of the model tank is modified for this study.

The simulation results with the mathematical model show that the proposed neural controller is suitable for the stabilization of the turret subsystem. The controller gains of the PID controller are tuned in order to satisfy the performance criterions in a very short time. In addition, the Columb friction, which exists on the mechanical system, is identified and compensated in a very short time with high accuracy. The addition of the neural friction compensator resulted in a 10 times higher stabilization performance.

Many tests are performed on the hardware with the proposed neural controller. The results obtained in these tests are also satisfactory. Although the stabilization accuracy values obtained in the experimental setup is larger than the stabilization accuracy values for the mathematical simulation model, this is mainly due to the ideal behavior of the model in the simulation and power limitations of actuation in

the hardware setup. The addition of the neural friction compensation algorithm to the controller increased the stabilization performance up to 5 times better values.

In addition, sensitivity analysis is performed for the neural linear system identifier network because of its simplicity. In this analysis, the sensor noise in the feedback signals and the quantization errors in the actuation system are considered. Two methods are used in this analysis. In the first method, the maximum values of the errors are used but in the second method, the statistical behavior of the errors are used. The results showed that the sensitivity of the network to sensor noise and quantization errors is almost negligible.

Performance analysis of the proposed neural controller is performed both using the mathematical model of the main battle tank and the experimental setup. In these analyses, the dynamics of the system is changed in the mathematical model in order to examine the performance of the controller. The dynamics changed in these tests are the turret inertia and the Columb friction values. Also performance analysis is performed with changing disturbance characteristics. This is done by changing the tank speed on the APG track.

The performance of the proposed neural controller is analyzed on the experimental setup with different speed command signal shapes, amplitudes and frequencies. As a result of these analyses, the performance of the proposed neural controller is considered to be satisfactory. The controller adapted itself to new signal shapes, signal amplitudes and signal frequencies. One problem was the very short time for the training of the neural friction compensator caused by the high rate of change of the command signals.

The sensitivity analysis of the neural PID tuner and the neural friction compensation networks are left for future works. A very similar procedure, which is used in the analysis of the linear neural system identifier, will be followed for these networks and the only two differences are the activation function type and the number of layers of the networks.

To sum up, an intelligent controller for the stabilization control of turret subsystems under disturbances from unstructured terrain is developed in this study. The proposed neural controller is both analyzed in the mathematical model of a main battle tank and on the experimental setup developed for this study. The results showed that the performance of the controller is satisfactory considering the stabilization and the tracking performance of the controller.

REFERENCES

[1] Carl Knospe, “PID Control”, IEEE Control Systems Magazine, pp. 30-32, February 2006.

[2] Wikipedia Organization, “Specialist Tanks”, “en.wikipedia.org”, July 2006

[3] Gourley, R., Scott, “Turret Gun Stabilisation or the Art of Moving to Keep Steady” Armada Magazine, Reader Service 004, 2004, pp 48-56.

[4] Ogarkiewicz, R.M., “Stabilized Tank Gun Controls”, International Defense Review, 1978, Vol. 5.

[5] Pease, Bob, “What’s All This P-I-D Stuff, Anyhow?”, Electronic Design, June 26, p.122., 1995.

[6] Wikipedia Organization, “Asymmetric Warfare, Strategic Basis”, “en.wikipedia.org”, July 2006

[7] Wikipedia Organization, “Medium Mark B”, “en.wikipedia.org”, July 2006

[8] Vern Nordstrand, “Boeing B-17 Tail Gun Turret: a Story from the War Years”, “www.historylik.org”, November 20, 2004.

[9] “Turret Controls”, Textron Systems, Cadillac Gauge, USA, 2003.

[10] “Savunma ve Silah Sistemleri”, “www.aselsan.com.tr”, ASELSAN INC., July 2006.

[11] India Defence, “Centurion Tank”, “www.india-defence.com”, July 2006.

[12] Kölbl, E., Wilhelm, “Fire Control Systems for Main Battle Tanks”, Military Technology, Vol. 5, No. 21, pp. 63-68, Feb. 1981.

[13] Purdy, J., David, “Main Battle Tank Stabilization Ratio Enhancement Using Hull Rate Feedforward”, Journal of Battlefield Technology, Vol. 1, No. 2, pp. 5-9, July 1998.

[14] Feinstein, Samuel, “The Evaluation of the Tank Fire Control System”, Armor Magazine, pp. 21-24, Feb. 1973

[15] J., Gregory, C., Jerry, I., Jeffrey, S., Rickie, W., Thomas, “MRAAS Weapon Stabilization Assessment”, Proceedings of NATO RTO Symposium on Functional and Mechanical Integration of Weapons and Land and Air Vehicles, Williamsburg, USA, pp. 38.1-38.16, June 2004.

[16] G. N. Saridis, "Foundations of the Theory of Intelligent Controls," Proceedings of the IEEE Workshop on Intelligent Control, Eds. G. Saridis, A. Meystel, Troy, NY 1985, pp. 23-28.

[17] A. Meystel, "K.-S. Fu: Life Devoted to the New Frontier of Science," Proc. of the IEEE Workshop on Intelligent Control, Eds. G. Saridis, A. Meystel, Troy, NY 1985, pp. iii-vi

[18] J. Albus, C. R. McLean, A. J. Barbera, M. L. Fitzgerald, "Hierarchical Control for Robots and Teleoperators," Proc. of the IEEE Workshop on Intelligent Control, Eds. G. Saridis, A. Meystel, Troy, NY 1985, pp. 39-49.

[19] Karray, F. O. and de Silva, C. W., “Soft Computing and Intelligent Systems Design—Theory, Tools, and Applications” Addison Wesley, 2004, ISBN 0-321-11617-8.

[20] Antsaklis, Panos, “Defining Intelligent Control”, Report of the Task Force on Intelligent Control, IEEE Control Systems Society, Dec. 1993.

[21] Abdi, Herve, “Neural Networks”, The University of Texas, Dallas, USA, 2003.

[22] Matlab Neural Networks Toolbox User Manual, Mathworks Corp., USA, 2006.

[23] Venkataraman, Anand, “The Back Propagation Algorithm”, “www.speech.sri.com”, 1999.

[24] Vedagarbha, Praveen, Dawson, M., Darren, Feemster, Matthew “Tracking Control of Mechanical Systems in the Precense of Nonlinear Dynamic Friction Effects”, IEEE Transactions on Control Systems Technology, Vol. 7, No. 4, July 1999.

[25] L. Da Vinci, The Notebooks, New York, 1519.

[26] A. Tustin, “The Effects of Backlash and of Speed-Dependent Friction on the Stability of Closed-Cycle Control Systems,” Proceedings of Inst. Elect. Eng., vol. 94, No. 2A, pp. 143–151, 1947.

[27] D. P. Hess and A. Soom, “Friction at a Lubricated Line Contact Operating at Oscillating Sliding Velocities,” J. Tribology, vol. 112, pp. 147–152, Jan. 1990.

[28] C. Canudas de Wit, K. J. Åström, and K. Braun, “Adaptive Friction Compensation in DC-motor Drives,” IEEE J. Robot. Automat., Vol. RA-3, pp. 681–685, Dec. 1987.

[29] E. Rabinowicz, “The Intrinsic Variables Affecting the Stick-Slip Process,” Proc. Phys. Soc. London, Vol. 71, No. 4, pp. 668–675.

[30] A. L. Ruina, “Friction Laws and Instabilities: A Quasistatic Analysis of Some Dry Frictional Behavior,” Ph.D. Dissertation, Division of Engineering, Brown Univ., Providence, RI.

[31] J. R. Rice and A. L. Ruina, “Stability of Steady Frictional Slipping,” J. Appl. Mechan., Vol. 50, pp. 343–349, 1983.

[32] P. E. Dupont and E. P. Dunlap, “Friction Modeling and Control in Boundary Lubrication,” Proceedings of Amer. Contr. Conf., AACC, San Francisco, CA, pp. 1910–1914, 1993.

[33] “Stick-Slip Arising from Stribeck Friction,” in Proc. IEEE Int. Conf. Robot. Automat., Cincinnati, OH, May 1990, pp. 1377–1382.

[34] C. Canudas de Wit, C. P. Noel, A. Aubin, and B. Brogliato, “Adaptive Friction Compensation in Robot Manipulators: Low Velocities,” Int. J. Robot. Res., Vol. 10, No. 3, pp. 189–199, 1991.

[35] “Stick-Slip and Control in Low-Speed Motion,” IEEE Trans. Automat. Contr., Vol. 38, pp. 1483–1496, Oct. 1993.

[36] A., Saiful and O., Sigeru, “Neuromorphic Self-Tuning PID Controller”, Dept. of Information Science and Intelligent Systems, University of Tokushima, Japan, 1993.

[37] M., Shigenori, O., Sigeru, H., Hiromasa, “Improvement of Speed Control Performance Using PID Type Neurocontroller in an Electric Vehicle System”, Shikoku Research Institute Inc., 1994.

[38] L., Xu, G., Zhao, “Application of an Identification Method of BP Neural Networks in the Speed Control System of Hyrdaulic Elevator”, Zhejiang University of Technology, 1998.

[39] P., Chen, L., Qiu, “System Identification in Hydraulic Servo System with Diagonal Recurrunt Neural Networks”, Beijing University of Aeronautics and Astronautics, 2000.

[40] S. Purwar, I.N. Kar, A.N. Jha, “On-Line System Identification Using Chebyshev Neural Networks”, Indian Institute of Technology, 2003.

[41] L. Constant, P. Lagarrigues, B. Dagues, I. Rivals, L. Peronnaz, "Modeling of Electromechanical Systems Using Neural Networks", LEEI, ESPCI, Paris, 2000.

[42] Jovanovic Olivera, "Identification of Dynamic System Using Neural Network", Department of Mechanical Engineering, University of Montenegro, 1998.

[43] Mathworks, MATLAB®, "The Language of Technical Computing, Using MATLAB®", 2002.

[44] Mathworks, Simulink®, "Dynamic System Simulation for MATLAB®; Using Simulink®", 2000.

[45] Mathworks, "Simmechanics® for Use with Simulink®; User's Guide", 2004.

[46] Mathworks, "Virtual Reality Toolbox® Reality Toolbox, for Use with MATLAB® and Simulink®; User's Guide", 2004.

[47] Mathworks, "Getting Started with xPC Target", 2002.

[48] International Test Operations Procedure, Stabilization Accuracy, ITOP 3-2-836, 29 June 1995.

[49] Gene Slover, "Roll, Pitch and Yaw Fire Control Problems and Mark 1/1A Solutions", "http://www.navweaps.com/index_tech/tech-074.htm", July 2000.

[50] Wen-Chung Kao, Lien-Yang Chen, Shou-Hung Chen, "Real-Time Image Stabilization For Digital Video Cameras", Proc. Of 2006 CACS Automatic Control Conf., pp. 451-454, November 2006.

[51] Jinyang Yi, Dezhen Song, Anthony Levendowski, Suhada Jayasuria, "Trajectory Tracking and Balance Stabilization Control of Autonomous Motorcycles", DARPA Grand Challenge 2005.

[52] Minh-Quan Dao, Kang-Zhi Liu, "Gain-Scheduled Stabilization Control of Unicycle Robot", JSME International Journal, Series C., Vol. 48, No. 4, pp. 649-659, 2005.

[53] Hiroshi Fujimoto, Takeo Saito, Tashihiko Naguchi, "Motion Stabilization Control of Electric Vehicle under Snowy Conditions Based on Yaw-Moment Observer", Department of Electrical Engineering, Nagaoka University of Technology

[54] M.B. Gürcan, Ö. Gümüşay, U. Batu, "Ana Muharebe Tankı Namlu, Kule, Gövde ve Süspansiyonunun Modellenmesi, Simülasyonu ve Namlunun Stabilizasyon Denetimi", USMOS 2005