YILDIRIM BEYAZIT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF FLEXIBLE MODEL BASED ADAPTIVE CONTROL ALGORITHM FOR REAL TIME SYSTEMS

M.Sc. Thesis by
Ufuk GÜNER

Department of Electronics and Communication Engineering

May, 2016

ANKARA

DESIGN AND IMPLEMENTATION OF FLEXIBLE MODEL BASED ADAPTIVE CONTROL ALGORITHM FOR REAL TIME SYSTEMS

A Thesis Submitted to the

Graduate School of Natural and Applied Sciences of Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Master of Science in

Electronics and Communication Engineering, Department of Electronics and Communication Engineering

by

Ufuk GÜNER

May, 2016

ANKARA

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"Design and Implementation of Flexible Model Based Adaptive Control Algorithm For Real Time Systems"** completed by **Ufuk GÜNER** under supervision of **Assoc. Prof. Dr. Hüseyin CANBOLAT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assoc. Prof. Dr. Hüseyin CANBOLAT

(Supervisor)

.....

Prof.Dr. M. Kemal LEBLEBİCİOĞLU

(Jury Member)

.....

Assoc. Prof. Dr. Arif ANKARALI

(Jury Member)

.....

Prof. Dr. Fatih V. ÇELEBİ

(Director)

Graduate School of Natural and Applied Sciences

ETHICAL DECLERATION

I have prepared this thesis in accordance with the Rules of Writing Thesis published by The Graduate School of Natural and Applied Science of Yıldırım Beyazıt University;

- Data I have presented in the thesis, information and documents are obtained in the framework of academic and ethical rules,
- All information, documentation, assessment and results are in accordance with scientific ethics and morals,
- I have given the references for all the works that I used for in this dissertation,
- I used the data obtained from the experiments without any change and modification,
- I declare that the work presented in this thesis is original,

I understand that all the rights and privileges earned through this thesis can be taken back in the event that any parts of the above decelerations are found to be untrue.

DESIGN AND IMPLEMENTATION OF FLEXIBLE MODEL BASED ADAPTIVE CONTROL ALGORITHM FOR REAL TIME SYSTEMS

ABSTRACT

This thesis proposes a flexible adaptive model based control algorithm for real time systems. The algorithm consists of a base controller and adaptation mechanisms. Parameters of base controller are updated with a flexible adaptation mechanism which tracks error of a plant and model output. The reference model is defined for a specific interval of the plant response and parameters of the reference model are updated for predefined interval trough identifications of plant input and output. In order to minimize the output error between model and the plant, Lyapunov based adaptation rules are designed. In addition, a fuzzy algorithm provides relation between the Lyapunov rules and base controller parameters. The proposed control algorithm does not need prior information of the dynamic system and has a flexible adaptation mechanism. Therefore, the algorithm can be easily implemented to different dynamic systems with minor modification. The algorithm is implemented for three different real time plants and experimental results are compared with base controller. Proportional–Integral-Derivative (PID) controller is used as base controller. The adaptive control algorithm needs simultaneous multi-processing for implementation, therefore some part of algorithm are run on a microcontroller and the others are run on a computer which has operation system. The experimental results show that the proposed algorithm tracks the system output in an acceptable error margin.

Keywords: adaptive control, reference model control, Lyapunov stability analysis, fuzzy control

GERÇEK ZAMANLI SİSTEMLER İÇİN ESNEK MODEL TABANLI UYARLAMALI KONTROL ALGORİTMASI TASARIMI VE UYGULANMASI

ÖZET

Bu çalışmada, gerçek zamanlı sistemler için, model tabanlı esnek bir uyarlamalı kontrol algoritması önerilmektedir. Önerilen algoritma temel kontrolcü ve uyarlama mekanizmalarından oluşmaktadır. Temel kontrolcüye ait parametreler uyarlama mekanizması tarafından model ve sistem arasındaki hatayı takip ederek güncellenmektedir. Referans model, belirli bir zaman aralığı için sistemin tepkisine göre tanımlanmış ve modele ait parametreler, sistem tanımlama kullanılarak sürekli güncellenebilmektedir. Model ve Sistem çıkışları arasındaki hata, Lyapunov tabanlı uyarlama kuralları ile azaltılmıştır. Bunlara ek olarak, temel kontrolcü ve uyarlama çıktıları bulanık mantık kullanılarak ilişkilendirilmiştir. Önerilen kontrol algoritması, sistemin karakteristiğine ait ön bilgilere ihtiyaç duymamaktadır ve esnek bir uyarlama mekanizmasına sahiptir. Dolayısıyla algoritma kolaylıkla farklı dinamik sistemlere uygulanabilmektedir. Algoritma üç farklı sisteme uygulanmış ve deneysel sonuçlar temel kontrol ile karşılaştırılmıştır. Temel kontrolcü olarak oransal-integral-türev (PID) kullanılmıştır. Önerilen kontrol algoritması, çoklu süreçlerin eşzamanlı çalışmasına ihtiyaç duymaktadır, bu amaçla algoritma iki işlemci üzerine eş zamanlı koşturulmuştur. Deneysel sonuçlar, önerilen kontrolcünün, sistem çıkışını kabul edilebilir bir hata seviyesi içinde tutuğunu göstermektedir.

Anahtar Kelimeler: uyarlamalı kontrol, referans model kontrol, Lyapunov kararlılık analizi, bulanık mantık

ACKNOWLEDGEMENTS

First of all, I want to thank my supervisor Assoc. Prof. Dr. Hüseyin CANBOLAT, for helping me to prepare this thesis. I want to thank my wife and daughter for their patience and supports.

2016, 24 May **Ufuk GÜNER**

CONTENTS

ABBREVIATIONS

ADC	Analog Digital Converter
ARM	Acorn RISC Machine
ANC	Adaptive Noise Canceller
CW	Clock Wise
CPU	Central Processor Unit
DC	Direct Current
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FIR	Finite Impulse Response
GSL	GNU Scientific Library
LMS	Least Mean Square
MEMS	Micro Electro-Mechanical System
MRAC	Model Reference Adaptive Control
MSE	Mean Square Error
NAND	Negative-AND
IMU	Inertial Measurement Unit
I2C	Integrated Circuit Communication
IIR	Infinite Impulse Response
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
RLS	Recursive Least Square
RAM	Random Access Memory
RPM	Revolution Per Unit
SBC	Single Board Computer
SISO	Single ınput Single Output
STR	Self-tuning Regulator
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
OS	Operating System

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Adaptive controllers are one of the most widely used systems in control engineering. The aim of the adaptive controllers is coping with system uncertainties and disturbance. In the last few decades, many researchers have worked on adaptive controllers and many different adaptive approaches have been developed. As result of these efforts, some adaptive controllers have been employed in many industrial fields. The applications of adaptive controller can be seen in many systems such as space crafts, satellites, flight controllers, servo drivers, etc. Although there are different approaches for adaptive controllers, many of them have adaptation mechanism which is based on modeling and/or identification. Model Reference Adaptive Controllers (MRAC) and Self Tuning Regulators (STR) are mainly focused adaptive controller schemes by researchers [1,2,3]. MRAC is based on modeling and adaptation mechanism which uses error between the model and plant output to update controller parameters. On the other hand, STR is based on identification which is used for estimating controller parameters. So, MRAC needs accurate mathematical modeling and STR needs accurate identification. Therefore, modeling and identifications have important for adaptive controllers. However, accurate modeling and identification are difficult: Generally, physical laws are used to develop the model of a system. In some cases, obtaining physical description of a plant is very difficult, because of complexity and unknown structure of plant dynamics [4]. Besides, in order to make accurate identifications, prior information of plant is needed and the complexity increases identification errors. So, MRAC and STR based controllers suffer from modeling errors and identifications with low accuracy for complex applications. These drawbacks decrease functionality and performance of adaptive controllers [5]. In addition, it is difficult to make generic model and/or identification based adaptive controller, since dynamic characteristics of systems vary considerably. Thus, specific adaptive controllers should be designed for a given system.

In the last decades, in order to improve the functionality and performance of model based adaptive controller, some methods have been developed. A modified MRAC and inverted pendulum application is proposed. The modified MRAC is combined with Proportional Integral Derivative (PID) controller in order to improve MRAC drawbacks [6]. A PID auto tuning method with MRAC approach is introduced [7]. Modified MRAC algorithm is implemented for vibration suppression of a piezoelectric smart structure. The modified algorithm uses integral term of control law in order to improve robustness of controller [8]. To solve closed loop instability of MRAC, a parameter projection algorithm is used for discrete-time systems [9]. A MIT rule based MRAC application is introduced for second order systems under variation of adaptation gain value [10]. A hybrid adaptive flight controller which consists of MRAC and STR features is investigated for reducing the effect of high-gain control [11]. In order to improving tracking performance of MRAC, a Lyapunov based PI and PID controller is designed and compared with gain scheduling methods for satellite launch vehicle systems [12]. Multiple model adaptive controller approach controller is proposed in order to improve transient response [13]. Variable structure MRAC is analyzed for transient and steady state performance [14]. A robot manipulator application of MRAC is introduced [15]. Fuzzy and MRAC combination is analyzed [16]. In addition, model free approaches are also investigated to make generic solution for adaptive controllers. Nowadays, model free and data-based design approaches are attractive alternative for researchers. These methods have an opportunity to make generic solution for adaptive controllers. Recently, many studies are made about adaptive free model and data-base controllers [17,18,19].

In this work, a model reference adaptive controller is designed with assumption of linear and time invariant system approximation. The adaptation process is based on tuning a base controller which is defined according to the actuator of plant. The adaptation is based on a low order linear stable reference model which is defined with dynamic behavior of system. In addition, the controller is supported with online system identification using experimental data of plant. The low order reference model allows using simple identification and adaptation laws. Thus, the proposed algorithm provides a flexible implementation.

The adaptive controller has three layers. The first layer is a base controller. In this study, PID and PI controllers are chosen as base controllers. The second layer consists of reference model and model update mechanisms. The model update mechanism estimates the parameters of the reference model for a predefined interval of the system which is called term. The reference model is defined with a term based exponential plant input and plant output signals. The last layer is adaptation mechanism which tunes the parameter of base controller. Adaptation is based on Lyapunov stability analysis. However, adaptation rule is not applied directly to the base controller, Because of increased the mathematical complexity. So, a fuzzy logic interface is used to update parameter of base controller.

The algorithm is tested on a single input-single output (SISO) case. The performance of the controller is compared with classical PID controller. In order to make the controller test, three different test setups are built; a DC motor, basic servo system and a dual rotor tilt system. The controller shows better performance and flexible implementation.

1.1. History of Adaptive Controllers

The adaptive control concerns coping with uncertainties and environmental effects of a system. The main scope of the adaptive controller is decisions and error minimization in a specific stochastic class in order to reach optimum behavior of a process. Adaptive controller concept is based on ability of simultaneous system identification and updating the controller parameters to produce a proper system response.

The first adaptive controller decision is defined by Draper and Li in 1951 for improving the performance of combustion engines with uncertainties, and at the same year, Benner and Drenick designed an adaptive control system [20].

In 1958, Whitaker made an adaptive controller for flight control system by defining the model for the error between desired and actual system output. The work of Whitaker is assumed as the beginning of model reference adaptive system [1].

In 1960, Li and Van Der Velde studied on another adaptive controller for compensation of parameter uncertainties in closed loop systems [21]. This controller scheme is called self-tuning adaptive system.

In 1961, dual control theory was developed by Alexander Aronovich Feldbaum from Russia. His work shows principle of optimal solution using dynamic programing for a system which is initially unknown [22].

In variable structure systems, the control input is connected to a switching function in order to control the system output. Russian researchers proposed variable structure system studies in 1960's [23, 24].

In 1971, self-tuning adaptive controller was improved by Astrom and Wittenmark [25]. And Self–Tuning Regulator (STR) could be implemented on computers and microprocessors. As a result, many researchers have begun studying on STR systems.

In 1974, an augmented error approach was proposed by Monopoli for making stable model reference controller design [26].

In years following, many different works have been established for MRAC and STR systems. Nowadays, thanks to the developments on microprocessor and electronic technology, application of adaptive controllers has tremendously increased and many different approaches have been developed.

1.1.1. Self-Tuning Regulators

Self-tuning regulator is based on classical feedback control and parameter tuning mechanism [27]. The figure 1.1 shows the basic schema of the self-tuning regulators. Self-tuning regulator has three main parts, parameter estimation, controller design and controller. The controller parameter is updated with estimation of the plant input and output. The self-tuning regulator provides online parameter adjustment. So, controller parameters can be updated at each control cycle. Self-tuning regulator has a flexible design structure according to the choice of controller design and estimation. Unknown parameters can be estimated using recursive estimation

schemes, such as least square, extended and generalized least square, stochastic approximation, instrumental variable and maximum likelihood. In controller design, pole placement, model following, minimum variance and linear quadratic regulator can provide solution. The different combinations of estimation and controller design methods provide to self-tuning regulator different properties [27].

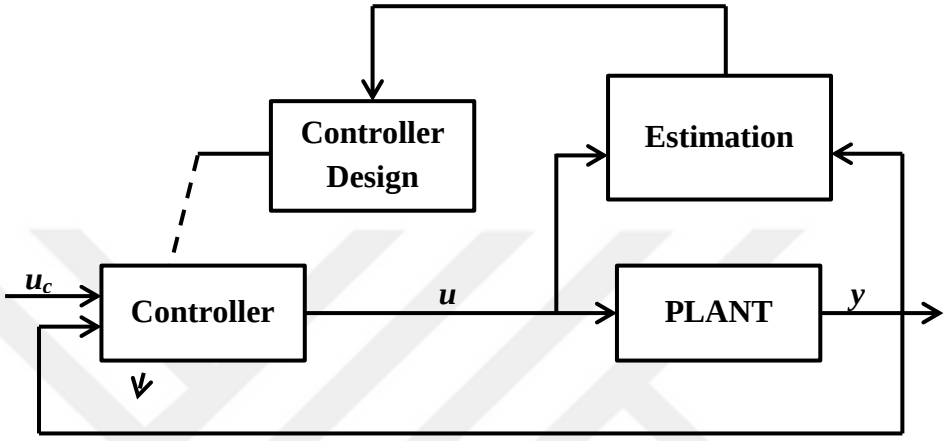


Figure 1.1 Self-Tuning Regulator (STR) scheme

1.1.2. Model Reference Adaptive Controller (MRAC)

Model reference adaptive control is based on comparison of real time system and a mathematical model. System input signal is applied on real time system and mathematical model. When error is occurred between model and real time system output, this error is used for adjusting parameters of controller. The model reference adaptive controllers are designed in order to make the output of the plant follow a prespecified reference model that provides desired behaviors of the plant. The figure 1.2 shows the essential MRAC scheme. The model corresponds to reference model of the plant. The adjustment mechanism is parameter update mechanism for controller. The general adjustment mechanisms are based on error minimization algorithm. The performance of the controller depends on the accuracy of reference model. For adjustment mechanism, generally gradient method and Lyapunov stability theory can be used [28]. For gradient method, MIT rule was developed and applied in Massachusetts Institute of Technology [29]. For linear system, Lyapunov theory also provides solution for many MRAC applications [30]. Lyapunov theory

can provide good solutions, because it is possible to make closed-loop system asymptotically stable. In this way, the controlled system tracks the reference model with minimum error, so transient response of system can be guaranteed to be bounded. [31]

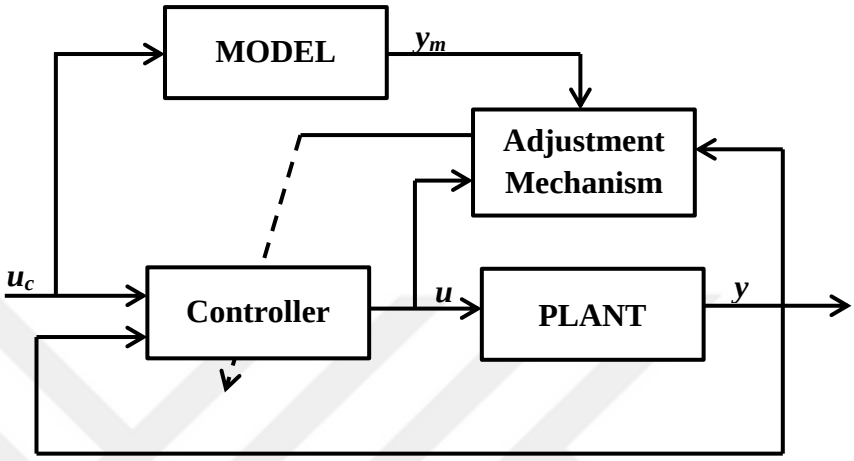


Figure 1.2 Model Reference Controller (MRAC) scheme

1.1.3. Lyapunov Stability

The theory of Lyapunov stability was introduced by Russian engineer and mathematician Aleksandr Lyapunov in 1892. Lyapunov's works were translated to many languages and published in the middle of 20th century. Lyapunov analysis deals with nonlinear systems. So, a nonlinear equation is given as

$$\frac{dy}{dt} = f(x) \qquad f(0) = 0 \qquad f : R^{+} \rightarrow R^{n} \tag{1.1}$$

In this condition, the system is assumed to be at equilibrium state at *f(0)* and the equation has the solution *x(t)=0* at the equilibrium. If *f(x)* is locally Lipschitz, there is a unique solution for the equation.

$$\| f(x) - f(y) \| \leq L \| x - y \| \qquad L > 0 \tag{1.2}$$

Where *L* is a positive real constant, and $\|.\|$ denotes Euclidean norm. The Lyapunov investigated that whether the solution is stable. The Lyapunov stability is defined by

that if there exist a $\delta(\varepsilon) > 0$ for every $\varepsilon > 0$, and $\| x(0) \| < \delta$ provides the condition $\| x(t) \| < \varepsilon$ for $0 \le t < \infty$, so the equation 1.1 is stable at equilibrium point $x = 0$. If the equation 1.1 is stable at equilibrium point and $\| x(t) \| < \delta$, $\| x(t) \| \to 0$ as $t \to \infty$. In this condition, the equilibrium point is asymptotical stabile. If the equation 1.1 is asymptotically stable for any initial value, then it is globally asymptotically stabile. Lyapunov method investigates the stability which is based on finding a positive definite function with special properties. This function is called Lyapunov function and there is no rule to define the Lyapunov function.

1.1.4. Lyapunov Direct Method

Lyapunov direct method deals with stability of nonlinear differential equations at equilibrium point. The method is based on a positive definite scalar function and investigating derivative of the function throughout all trajectory. The idea is based on the energy of a system which is always positive definite and dissipates with respect to the time.

Kalman and Bertram Lemma (1960): For a system, which depend on *x* and *t*, there exists a scalar function *V(x,t)* and the derivative of *V* is obtainable according to *x* and *t*. So, if *V* has the following property, system is said to be asymptotically stable:

i) $V(0,t) = 0$,

ii) $V(x,t)$ is positive definite and for continuous scalar functions *u* and *n*, $u(0) = 0$ and $n(0) = 0$, $u(\| x \|) \ge V(x,t) \ge n(\| x \|) > 0$, $\forall x \ne 0$,

iii) $\dot{V}(x,t)$ is negative definite and for a continuous scalar function v, $v(0) = 0$, $\dot{V}(x,t) \le -v(\| x \|) < 0$.

The lemma is proofed by Narendra and Annaswamy (1989).

1.1.5. Positive Definite Function

A differentiable function $V : R^n \to R$ is called a positive definite function, when $V(0) = 0$ and $V(x) > 0$, $x \in R^n, x \ne 0$. If V is a Lyapunov function and the time derivative of V is negative definite, states of the system close to equilibrium point

and stay in a neighborhood around equilibrium point. Therefore, the system is asymptotically stable.

1.1.6. Fuzzy Logic

The fuzzy logic is introduced by Lotfi A. Zadeh in 1965[32]. After that, fuzzy logic has been investigated extensively and applied to various field. Nowadays, fuzzy logic based controller can be found in electronic home applications and industrial devices. The basic fuzzy logic system deals with uncertain conditions and determines a response for those conditions.

The fuzzy logic evaluates a situation with uncertain variable, instead of absolute variable. Therefore, fuzzy logic provides better description of an uncertain condition. It is also possible to describe a situation with a linguistic variable like as little, a lot, middle, short, tall, fast, slow normal etc. For this reason, fuzzy logic provides determination which is close to human thinking system. Fuzzy based system can provide solution for some systems which cannot be modeled or have complex mathematical model.

On the other hand, fuzzy logic has some disadvantages. For example, it is not possible to make stability and controllability analysis of fuzzy system and experience is needed to define the situation and relations. The structure of fuzzy logic is based on fuzzy set theory and consists of three stages, fuzzification, inference engine and defuzzification. The basic fuzzy logic process can be seen in figure 1.3.

The fuzzification prepares the input information for inference engine using a membership function. The inference engine produces a decision according to the rule base. The defuzzification converts the decision to output. In fuzzification process, fuzzy sets are used to define membership degree of physical data. So, the physical data is represented by a linguistic word, such as big, middle, small and zero. After that, those words are converted to numerical variables in a predefined interval.

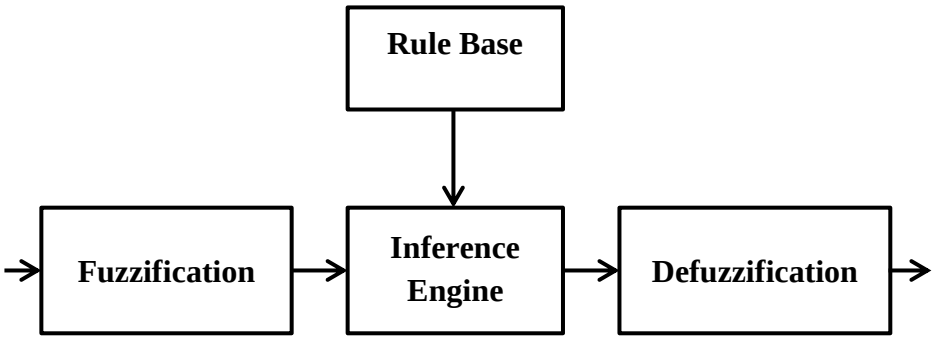


Figure 1.3 Fuzzy logic process

The fuzzy membership functions are often described with geometric shapes and formulation. Triangle and trapezoid are widely used as fuzzy membership function. For one fuzzy set, triangle and trapezoid membership functions can be seen in figure1.4.

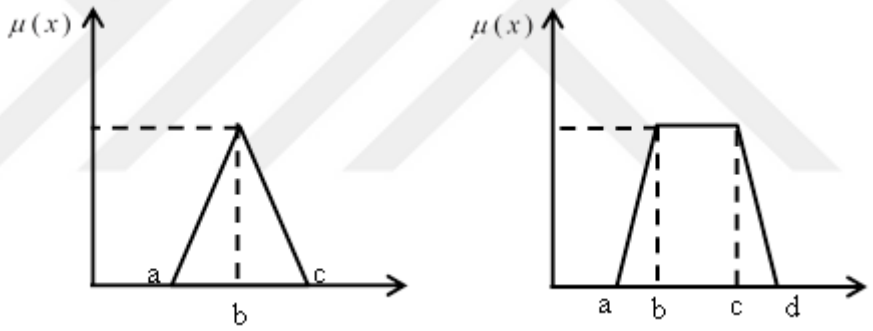


Figure 1.4 Triangle and trapezoid membership functions

In figure 1.4, *μ(x)* represent membership function and mathematical representation of a triangle membership function can be expressed as

$$\mu(x) = traingle(x;a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & x > c \end{cases} \quad \textbf{(1.3)}$$

The mathematical representation of trapezoid membership function can be expressed as:

$$\mu(x) = trapezoid(x:a,b,c,d) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x < b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & b \le x \le c \\ 0, & x \ge d \end{cases} \quad \textbf{(1.4)}$$

If all input states are defined with Negative Big (NB), Negative Small (NS), Zero (Z), Positive Small (PS) and Positive Big (PB), the resultant triangle membership function can be expressed with figure 1.5.

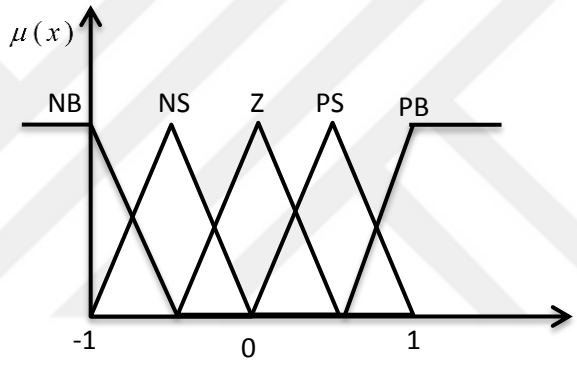


Figure 1.5 Five state membership function

The rule base is important part of the fuzzy systems. It works together with inference process. The input membership degree is determined with respect to rule base and a decision is produced for the output degree. So, rule base should cover decisions and solutions for the desired control system.

The inference unit produced fuzzy output according to the rule base. There are different inference methods in literature. The min-max inference methods are used widely for fuzzy logic applications. The min-max methods are first introduced by Mamdani [33], so this method is also called Mandani inference method. In Mamdani methods, “AND” connected variables represent the minimum and “OR” connected variables represent the maximum. The figure 1.6 shows basic Mamdani method.

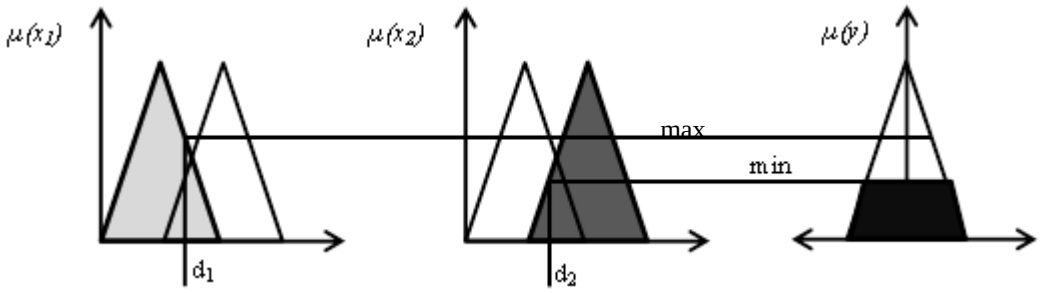


Figure 1.6 The Mamdani inference method

For mathematical representation, the equation 1.5 is used.

$$\mu_k(y) = \max_k \left\{ \min \left[\mu_k(d_1), \mu_k(d_2) \right] \right\} \quad k = 1, 2, ... p \tag{1.5}$$

Where p represents the number of inputs, d_1 and d_2 are values of x_1 and x_2. There are various methods can be used for defuzzification. In the maximum membership method, the biggest membership degree is used for defuzzification. It is defined as

$$\mu_c(z^*) = \max(\mu_c(z)) \quad for\ all\ z \in Z \tag{1.6}$$

Where z is set of fuzzification and z^* is set of defuzzification. The center of area method is used the most frequently for defuzzification. It can be expressed as

$$z^* = \frac{\int \mu_c(z).z dz}{\int \mu_c(z) dz} \tag{1.7}$$

The algebraic expression of the weighted average is given in equation 1.8.

$$z^* = \frac{\sum \mu_c(\overline{z})\overline{z}}{\sum \mu_c(\overline{z})} \tag{1.8}$$

Where, $\overline{z}$ represents the mean of the output membership degree. The center of sum method is based on the algebraic sum of individual output fuzzy sets. It is defined as

$$z^* = \frac{\int z \sum_{k=1}^{n} \mu_{c_k}(z) dz}{\int \sum_{k=1}^{n} \mu_{c_k}(z) dz} \tag{1.9}$$

CHAPTER 2

CONTROLLER DESIGN

2.1. Adaptive Controller Design

The proposed adaptive controller is built on a MRAC with an adaptation mechanism and a base controller. The base controller can be a classical controller such as PI, PD or PID. The adaptation mechanism is based on a reference model which tunes parameters of the base controller. The reference model is defined with input-output behavior of the plant. Moreover, the reference model has an update mechanism which provides a self-tuning feature to the controller. The update mechanism uses identification of plant input-output signals to estimate parameter of the reference model. The identification process is repeated for every certain number of samples of input-output signal. From now on, this certain number of samples is called a cycle or an update cycle. The reference and plant model are defined through a cycle model. In this way, the controller does not need exact plant model. In the initial cycle, the controller needs data from the plant to start the adaptation. Therefore, the plant is controlled with only base controller during the initial cycle of the system. The initial response of the system provides information for the controller, in order to introduce initial parameters of the reference model.

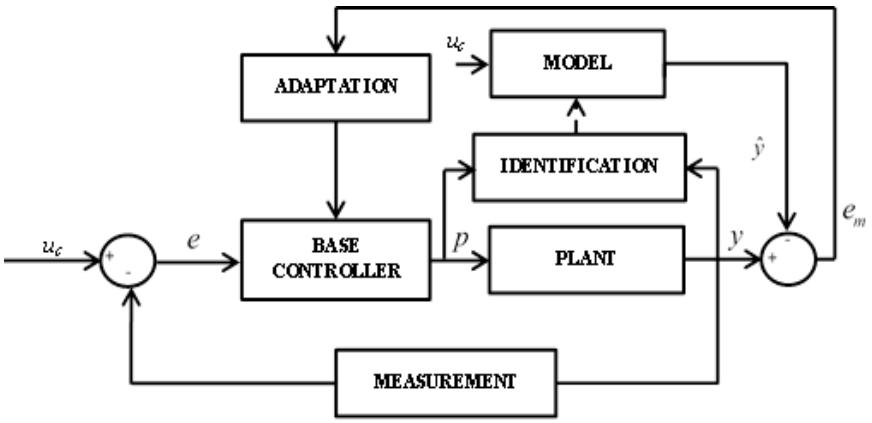


Figure 2.1 Basic schema of the proposed adaptive controller

The basic schema of the proposed adaptive controller can be seen in figure 2.1. In the figure, *y* and *p* represents plant output and input signals respectively, u_c is the control signal which defines the desired level of the plant. The error between plant output and control signal is represented with *e* which drives base controller. The error between plant output and model output is represented by e_m which is used for adaptation.

The Controller has two adaptation mechanisms. The first mechanism is applied to the update cycle model. The identification process updates parameter of the update cycle model for the next cycle. The second mechanism is Lyapunov direct method which is used for decreasing the error between plant and model outputs. A Lyapunov adaptation rules are used for tuning the parameters of the base controller. A fuzzy algorithm provides the connection between Lyapunov adaptive rules and the base controller.

The fuzzy algorithm consists of fuzzification, rule base and defuzzification blocks. The rule bases provide relation between the Lyapunov adaptation and base controller.

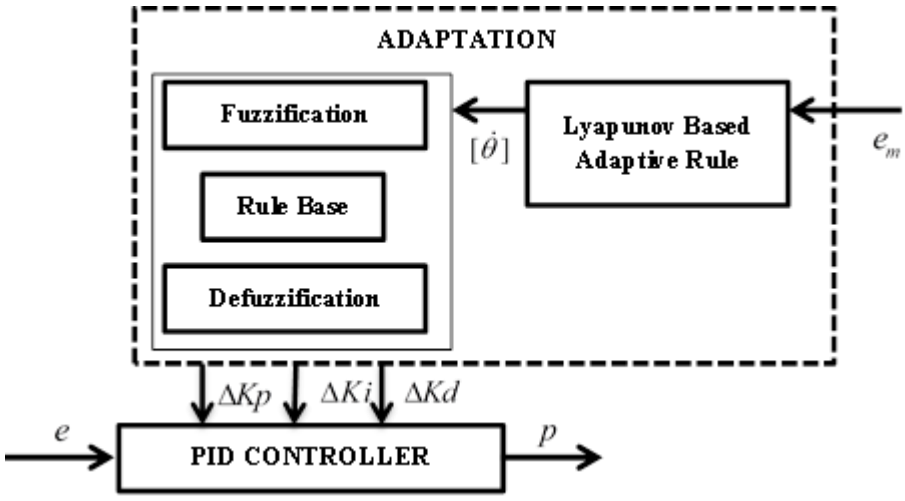


Figure 2.2 Adaptation mechanism

Therefore, the fuzzy rule base has an important role in the adaptation. The fuzzy rule base is to be arranged for desired outputs of plant. The figure 2.2 shows details of the adaptation mechanism.

PID controller is chosen as base controller. Therefore, adaptation mechanism has three outputs which are represented with ΔKp, ΔKi and ΔKd. The vector $[\dot{\theta}]$ represents the Lyapunov adaptation rules output vector.

The aspect of the adaptation process is the minimization of e_m, so plant output is expected to become close the model, and adaptive controller forces the plant to behave as the model.

The actual plant model is assumed unknown, so a low order cycle model is designed for system. The cycle model is assumed linear time invariant. So, the Lyapunov adaptive rule extraction method can easily be implemented for error minimization.

2.1.1. Modeling

It is assumed that there is an actuator based SISO system whose closed loop transfer function is stable. The aim of the adaptive controller is to determine proper input signal for reaching the desired output level. So, *p* is defined as the input function of the actuator and *y* is defined as the output of the plant. The system output is expected to reach the desired level exponentially for ideal case and the input of the plant is also assumed an exponential function.

Therefore, an update cycle function can be defined for *i*'th cycle as follows:

$$p_i(t) = a_i(1-e^{-\gamma_i t})u(t) \quad \gamma_i > 0, a_i \neq 0 \quad i \in Z^+ \tag{2.1}$$

Where, *u(t)* is unit step function, *a* is the saturation coefficient and γ is the time constant. The variables *a* and γ are defined for the update cycle *i*.

A short term function can be defined for predefined interval of output signal.

$$y_i(t) = b_i(1-e^{-\lambda t})u(t) \quad \lambda_i > 0 \; b_i \neq 0 \quad i \in Z^+ \tag{2.2}$$

Where, u*(t)* is the unit step function, b is the saturation coefficient and λ is the time constant. The variables b and λ are defined for the update cycle *i*.

In order to define the transfer function of the plant for the cycle *i*, Laplace transform of the p_i(t) and y_i(t) can be used:

$$P_i(s) = \frac{a_i}{s} - \frac{a_i}{s+\gamma_i} \tag{2.3}$$

$$Y_i(s) = \frac{b_i}{s} - \frac{b_i}{s+\lambda_i} \tag{2.4}$$

Therefore, transfer function for cycle *i* is

$$G(s) = \frac{Y_i(s)}{P_i(s)} = \frac{b_i\lambda_i}{a_i\gamma_i}\left(\frac{s+\gamma_i}{s+\lambda_i}\right) = m_i\left(\frac{s+\gamma_i}{s+\lambda_i}\right) \tag{2.5}$$

where m_i is

$$m_i = \frac{b_i\lambda_i}{a_i\gamma_i} \tag{2.6}$$

G(s) is defined as first order biproper transfer function according to the input-output behavior of the plant. *G(s)* is used both model and plant. *G(s)* has a parameter vector which has to be determined by identification. So, the parameter vector is defined as:

$$\Lambda = [\lambda_i, \gamma_i, a_i, b_i] \tag{2.7}$$

These parameters are updated online at end of each cycle. However, they are constant during the cycle.

2.1.2. Adaptive Rules

The Lyapunov adaptive rule extraction method is used for error minimization. The actual plant model is assumed unknown, *G(s)* can be used for model and plant differential equations. The plant differential equation at each update cycle is

$$\dot{y} = m\dot{p} + m\gamma p - \lambda y \tag{2.8}$$

and the model equation can be defined as:

$$\dot{y}_m = m\dot{\hat{p}} + m\gamma\hat{p} - \lambda y_m \tag{2.9}$$

where $\hat{p}$ represents the virtual input and has parametric relation to actual plant input, as follows:

$$\begin{aligned} &\hat{p} \cong p(t,\theta) \\ &\hat{p}_i(t) = a_i\theta_1(u(t) - e^{-\theta_2\gamma_i t}) \quad \gamma_i > 0 \quad a_i \in R \end{aligned} \tag{2.10}$$

where θ_1 and θ_2 are adaptation parameters. In order to parameterize the model and plant equations, k_p and θ variables are used and the nominal value of θ is expressed as $\theta^*=1/k_p$ for $k_p > 0$. The plant and model equations become

$$\dot{y} = k_{p1}\theta_1 m_i \dot{p}_i + k_{p2}\theta_2 m_i \gamma_i p_i - \lambda_i y \tag{2.11}$$

$$\dot{y}_m = k_{p1}\theta_1^* m_i \dot{\hat{p}} + k_{p2}\theta_2^* m_i \gamma_i \hat{p} - \lambda_i y_m \tag{2.12}$$

Model and plant output error can be expressed as:

$$e_m = y_m - y \tag{2.13}$$

If the error is made go to zero in time, the plant output is expected to become close to the reference output.

$$\lim_{t\to 0} e_m \to 0 \Rightarrow y \to \hat{y} \tag{2.14}$$

The time derivative of (2.13) is defined as:

$$\dot{e}_m = \dot{y} - \dot{y}_m \tag{2.15}$$

substituting from (2.11) and (2.12) for $\dot{y}$ and $\dot{y}_m$ yields:

$$\begin{aligned} \dot{e}_m = {} & k_{p1}\theta_1 m_i \dot{p} + k_{p2}\theta_2 m_i \gamma_i p - \lambda_i y \\ & - k_{p1}\theta_1^* m_i \dot{p} - k_{p2}\theta_2^* m_i \gamma_i p + \lambda_i y_m \end{aligned} \tag{2.16}$$

rearranging (2.16), error equation becomes

$$\dot{e}_m = -\lambda_i e_m + k_{p1} m_i \dot{p}(\theta_1 - \theta_1^*) + k_{p2} m_i \gamma_i p(\theta_2 - \theta_2^*) \tag{2.17}$$

at that point, ϕ_1 and ϕ_2 are defined as :

$$\begin{aligned} \phi_1 &= \theta_1 - \theta_1^* \\ \phi_2 &= \theta_2 - \theta_2^* \end{aligned} \tag{2.18}$$

Substituting (2.18) into (2.17), the error dynamic is found as follows:

$$\dot{e}_m = -\lambda_i e_m + \phi_1 k_{p1} m_i \dot{p} + \phi_2 k_{p2} m_i \gamma_i p \tag{2.19}$$

Let the Lyapunov function candidate be defined as :

$$V(e_m, \phi_1, \phi_1) = \frac{1}{2}(e_m^2 + k_{p1}\phi_1^2 + k_{p2}\phi_2^2) \tag{2.20}$$

Clearly, *V* is positive definite. The time derivative of *V* is

$$\begin{aligned} \dot{V}(e_m, \phi_1, \phi_2) = &-\lambda_i e_m^2 + e_m k_{p1} m_i \dot{p}\phi_1 + e_m k_{p2} m_i \gamma_i p\phi_2 \\ &+ k_{p1}\phi_1\dot{\phi}_1 + k_{p2}\phi_2\dot{\phi}_2 \end{aligned} \tag{2.21}$$

Rearranged (2.21) yields the following:

$$\dot{V}(e_m, \phi_1, \phi_2) = -\lambda_i e_m^2 + k_{p1}\phi_1(e_m m_i \dot{p} + \dot{\phi}_1) + k_{p2}\phi_2(e_m m_i \gamma_i p + \dot{\phi}_2) \tag{2.22}$$

At this point, one can introduce the adaptive rules as:

$$\begin{aligned} \frac{d\theta_1}{dt} &= \dot{\phi}_1 = -m_i \dot{p} e_m \\ \frac{d\theta_2}{dt} &= \dot{\phi}_2 = -m_i \gamma_i p e_m \end{aligned} \tag{2.23}$$

substituting (2.23) into (2.22) gives the following:

$$\dot{V} = -\lambda_i e_m^2 \qquad \textbf{(2.24)}$$

From (2.24), it is clear that the derivative of V is negative semidefinite.

There are two adaptation rules for parameters θ_1, θ_2. If these parameters are directly applied to the base controller, the control complexity is increased. Therefore, adaptive controller needs a relation algorithm between adaptation and base controller. A fuzzy approximation is used for relation algorithm. The input function has two features according to (2.10), one defines maximum amplitude level and other defines time constant. The fuzzy rule base can be derived from input function behavior. The adaptation mechanism is given in figure 2.3.

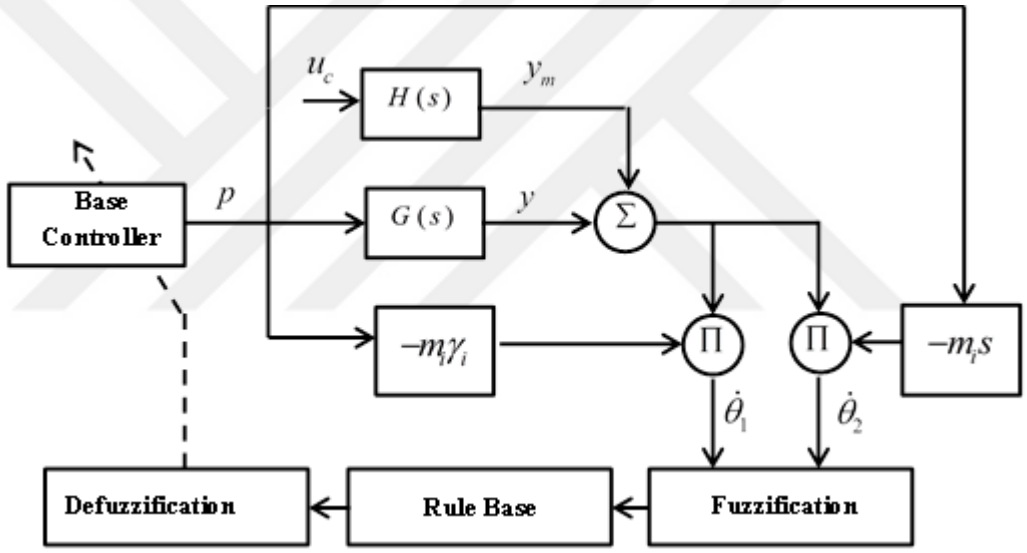


Figure 2.3 Lyapunov adaptation

In the figure 2.3, *H(s)* is reference model and *G(s)* is plant transfer function. *H(s)* is introduced from model differential equation (see equation 2.9). *G(s)* represents actual plant model which is unknown. However, *G(s)* is related with equation 2.8 for update cycle. The adaptive rule results are passed to the fuzzification block as derivative. However, they are normalized according to pre-defined maximum and minimum values.

The maximum and minimum values of adaptive rules are determined experimentally. After that point, the fuzzy rule base is defined. The rule base is arranged according to response of plant. In order to define the relation between the plant response and adaptation parameter, a basic servo setup is used. The figure 2.4 shows the basic servo setup. This system provides more robust response, because of the servo motor which has a position controller inside. The detail of the basic servo setup can be found in section 3.2.

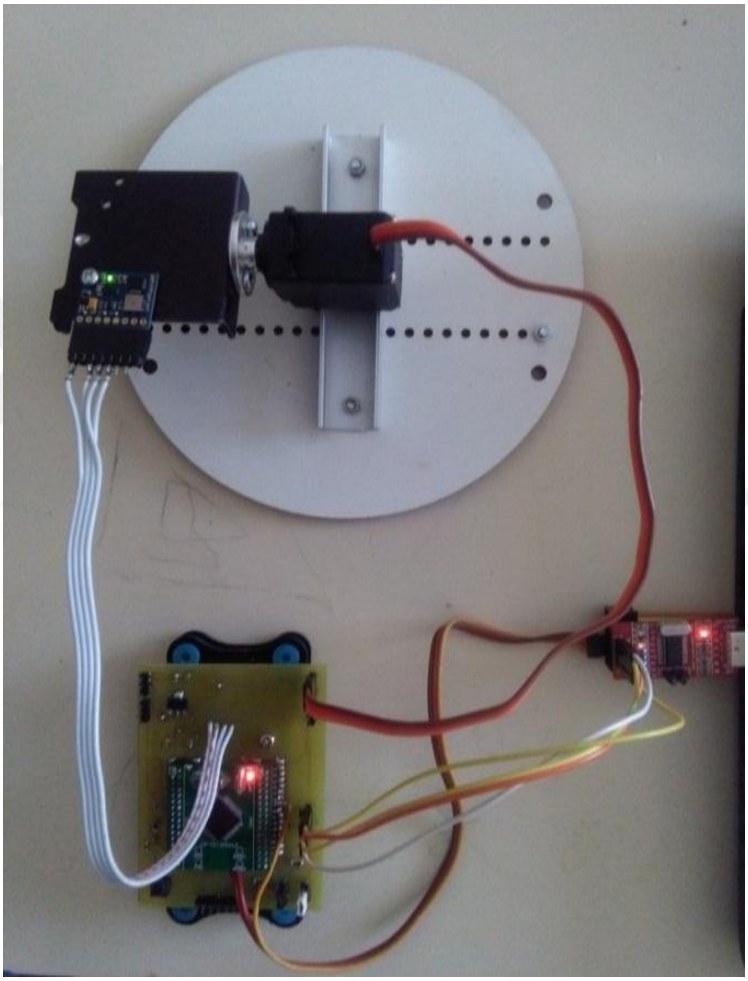


Figure 2.4 The basic servo setup

The relation between the plant response and adaptation parameter can be seen in table 2.1. This table is prepared experimentally with investigating real system response with using the basic servo setup.

For experiment, system response is observed for only PID controller and ideal system response is defined. After that, system is controlled by adaptive controller and effects of adaptive rules are observed.

Table 2.1. Adaptation parameter and plant response relation

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error
$\dot{\theta}_1 >> 0$	Not enough	Not expected	Longer	Higher
$\dot{\theta}_1 > 0$	Not enough	Not expected	Long	High
$\dot{\theta}_1 \cong 0$	Enough	Low	Short	Not defined
$\dot{\theta}_1 < 0$	High	Expected	Not defined	Not defined
$\dot{\theta}_1 << 0$	Higher	High	Not defined	Not defined
$\dot{\theta}_2 >> 0$	Not enough	Not expected	Longer	Higher
$\dot{\theta}_2 > 0$	Not enough	Not expected	Long	High
$\dot{\theta}_2 \cong 0$	Enough	Low	Short	Low
$\dot{\theta}_2 < 0$	High	High	Short	High
$\dot{\theta}_2 << 0$	Higher	Higher	Short	Higher

In the table 2.1, $\dot{\theta}_1$ and $\dot{\theta}_2$ are Lyapunov adaptation outputs and the plant response is represented with four state; Rise Time, Overshoot, Settling Time and Steady-State Error. The behavior of the states is defined by linguistic words. These states come from closed loop system response. The magnitude of $\dot{\theta}_1$ and $\dot{\theta}_2$ is investigated for five condition with respect to positive and negative side of zero; much larger than zero ($\dot{\theta} \gg 0$), larger than zero ($\dot{\theta} > 0$), close to zero ($\dot{\theta} \cong 0$), smaller than zero ($\dot{\theta} < 0$), much smaller than zero ($\dot{\theta} \ll 0$). The desired system response is realized when $\dot{\theta}_1$, $\dot{\theta}_2$ are close to zero. The states are used to define the fuzzy rule base.

The fuzzy rule base defines the change of PID parameters *Kp, Ki* and *Kd.* The table 2.2 shows the rule base for changing of ΔKp, ΔKi and ΔKd. The rule base is argued from table 2.1, in order to force the system response for minimizing overshoot, minimizing settling error and fast settling time. The rule base can be also used for PI and PD base controllers. For PI controller, ΔKp and ΔKi are used. For PD controller, ΔKp and ΔKd can be used.

Table 2.2. Fuzzy rule base

	$\dot{\theta}_2$ \ $\dot{\theta}_1$	NB	NS	Z	PS	PB
ΔK_p	**NB**	NB	NB	Z	PS	PS
	NS	NB	NS	Z	PS	PS
	Z	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	NS	Z	PS	PB	PB
ΔK_i	**NB**	NB	NB	NS	PS	Z
	NS	NB	NS	NS	PS	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	PS	PS	PS	PB
	PB	Z	PS	PS	PB	PB
ΔK_d	**NB**	NB	NB	Z	PS	NB
	NS	Z	NS	Z	Z	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	Z	PS	PS	Z
	PB	PS	PS	PS	PB	PB

For fuzzification, the member degrees of $\dot{\theta}_1$ and $\dot{\theta}_2$ are defined with using the triangle membership function. The figure 2.5 shows the triangle membership function of normalized $\dot{\theta}_1$ and $\dot{\theta}_2$. The $\mu(z)$ represents membership value for $0 \le \mu(z) \le 1$

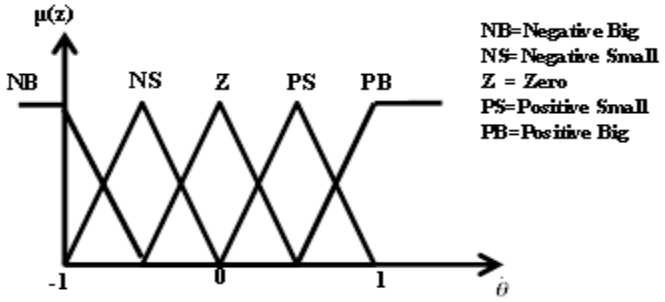


Figure 2.5 Membership function of fuzzy algorithm

The relation among the membership functions is provided with the Mamdani inference methods. The fuzzy inference process provides a resultant fuzzy set. The resultant fuzzy set is converted into parameter change value through defuzzfication. The center of area method is used for defuzzification.

$$z^* = \frac{\sum \mu_c(z) z}{\sum \mu_c(z)} \qquad \textbf{(2.25)}$$

The results of the defuzzification process are applied to change of *Kp, Ki* and *Kd* parameters.

2.1.3. Algorithm

The algorithm is based on sampling of plant data. There are two simultaneous processes for the algorithm. The base controller and adaptive controller have their own algorithm. There is a communication protocol between base and adaptive controller. The communication protocol is responsible for bidirectional data transmission and updating parameter of the base controller. The algorithm of adaptive controller consists of three blocks; adaptation, identification and model update. The adaptation and fuzzy algorithms run together. The identification and model update algorithm run parallel with adaptation and fuzzy algorithm. The identification is based on linear fitting algorithm and applied to both input and output samples of plant. The model update algorithm uses identified parameter to rearrange model output. The main algorithm stores the plant input and output data in buffers. Therefore, the necessary buffers are created and fuzzy member functions and fuzzy rule base are initialized before system starts. The pseudo code of adaptation and fuzzy is outlined in algorithm 1.

Algorithm1:

1 **set** plant_index ← 0
2 **set** model_block_index ← 0
3 **set** block_cnt ←0
4 **set** block_size ←n
5 **set** par_index ←0
6 **set** y_i ← {0,0,….}
7 **set** model_y_i ← {0,0,….}
8 **set** p_i ← {0,0,….}
9 **set** fuzzy_member
10 **set** fuzzy_rule_table
11 **while**
12 p_i ← **new plant input data**

```
13 y_i ← new plant measured output data
14          if block_cnt is bigger than zero
15                  e_m ← y_i – model_y_i
16                   ṗ ← (p_i-p_i-1)/Δ
17                   θ̇_1 ← -m_k ṗ e_m
18                   θ̇_2 ← -m_k γ_k p e_m
19                  normalize (θ̇_1)
20                  normalize (θ̇_2)
21                  member_degrees ← fuzzification(θ̇_1, θ̇_2)
22                  ΔKp ,ΔKi, ΔKd ← defuzzification(member_degrees)
23                  Kp ← Kp+ ΔKp
24                  Ki ← Ki + ΔKi
25                  Kd ← Kd+ ΔKp
26          end if
27          i ← i+1
28          block_cnt ← block_cnt+1
29          if block_cnt is equal to block_size
30                  linearfit(p,y, block_size)
31                   idendify a,b,λ,γ
32                   m ← b λ/aγ
33                  model_y ← updatemodel()
34                  block_cnt ← 0
35          end if
36 continue
```

The controller is designed for PID base controller. Therefore the algorithm needs PID output and plant output. The PID output is the plant input. In the algorithm, PID output is defined as "*new plant input data*" and the plant output is entered to the algorithm as "*new plant measurement output*". The *block_size*, whose value is *n*, defines the update cycle interval for the algorithm. The algorithm waits for gathering the first *n* samples. In this period, only PID controller is in charge. After that, adaptation process is started. For each sample, model and plant output error is

calculated. This error and identified variable are used for computing Lyapunov adaptive rules. After normalization of these rules, fuzzy process alters the PID parameters. When every *n* samples are gathered, identification process runs and updates the model parameters. The identification process is based on finding best fitting function of plant output and input. For identification, the linear fitting algorithm of GNU Scientific Library (GSL) is used [34]. The weighted least squares algorithm of GSL is used for linear fitting. The model output is revised by model update algorithm. The pseudo code of model update algorithm is outlined in algorithm 2.

Algorithm2:

```
1  set kl ← 0
2  set base← 0
3  set i←plant_index
4  set ds←0
5  kl←set_point-model_yi
6  if |kl|< deadband then kl ← 0
7  end if
8  base ← model_yi
9  while  i<max_size_of_output
10       if kl is not equal to zero
11            ds←base+kl*(1-exp(-|λ/kl|*i))
12       else
13            ds←base
14       end if
15            mode_yi←ds
16 continue
```

In model update algorithm, there is a *deadband* for difference between reference level and last point of model output. The "*base*" represents the last point of the model output. Each end of the update cycle, model update algorithm rearranges the whole reference model output array.

CHAPTER 3

ACTUAL SYSTEM INTEGRATION

For real time integration, three different test platforms were built. First, a DC motor setup is constructed to set the desired speed. The second platform is a basic servo system whose position is settled with Inertial Measurement Unit (IMU). The third system is a mechanical tilt platform controlled by IMU. Moreover, in order to implement the algorithm, software was developed for microcontroller, computer and embedded Linux board. As a base controller, PI controller is used for the mechanical tilt platform and basic servo system. In DC motor setup, PID controllers are used.

In in order to improve the IMU measurements, a quaternion based extended Kalman filter is designed. In mechanical tilt platform, a software based adaptive noise canceller is designed for IMU, to suppress high vibration.

3.1. DC Motor Setup

The DC motor setup consists of a geared DC motor, motor driver and microcontroller. The DC motor has a Hall Effect sensor encoder which provides 64 pulses for one rotation and the gear box has 30:1 ratio. Therefore, the 1920 pules are produced by the encoder for one rotation of gear box. The motor is driven with PWM signal which is provided by the microcontroller. The encoder output is read by microcontroller and is converted to Revolution Per Minute (RPM). The basic block of DC motor setup is given in figure 3.1.

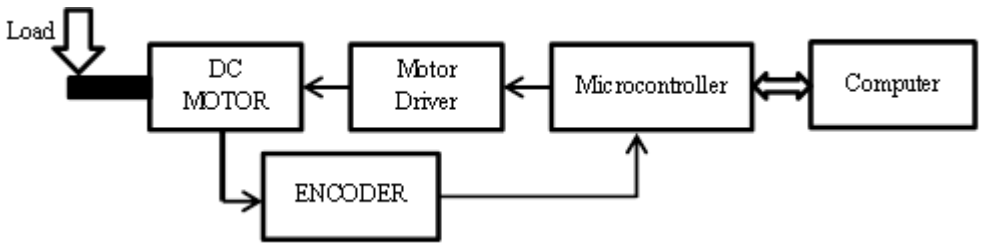


Figure 3.1 DC motor setup block diagram

In the microcontroller, there is PID controller to set desired RPM for the motor. The parameters of controller and encoder outputs are sent to computer. In computer side, the adaptation process updates parameters of the PID controllers. For load test, a servo system, which applies friction on gearbox shaft, is used. The friction level is adjusted with the PWM of servo motor. The figure 3.2 shows the DC motor setup.

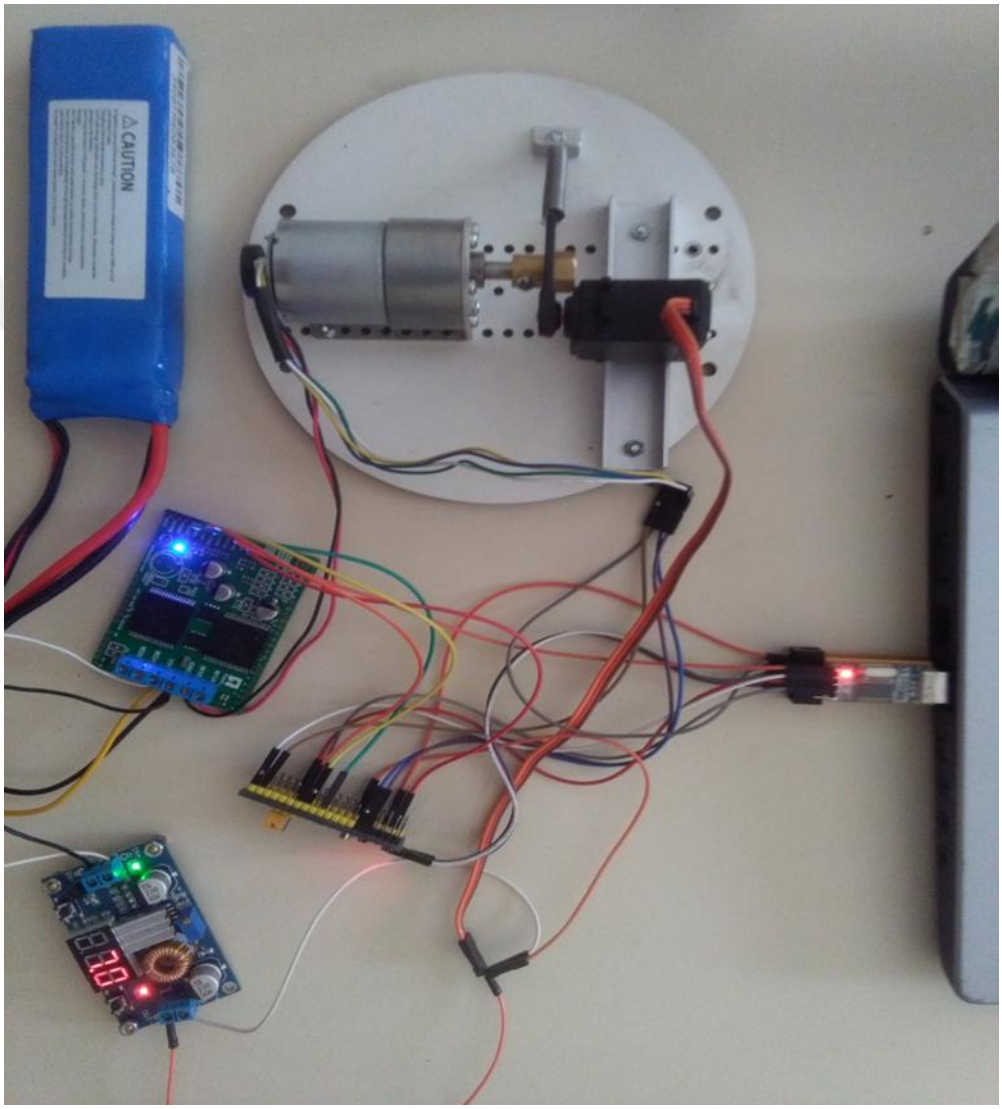


Figure 3.2 DC motor setup

The communication between computer and microcontroller is provided over serial port. The model update, adaptation and identification process run on computer sides. In the setup, STM32F103C8T6 Cortex M3 microcontroller is used.

The microcontroller is responsible for PID controller and encoder reading. The process chart can be seen in figure 3.3.

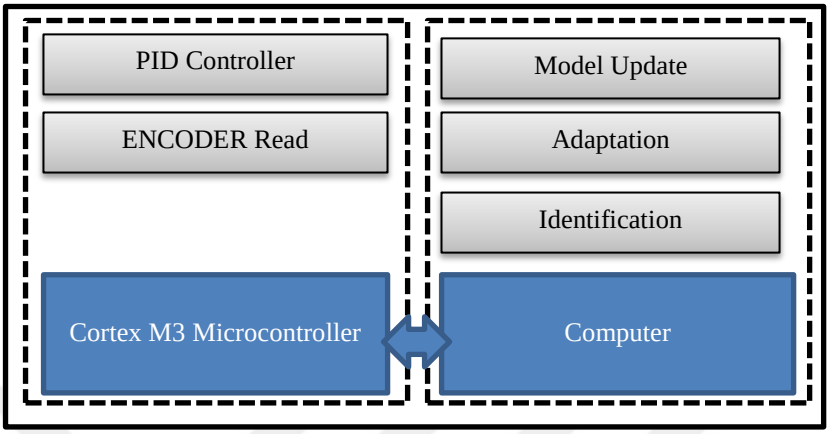


Figure 3.3 DC motor setup process chart

For experiment, fuzzy rule base can be seen in table 3.1. The rule table is introduced from adaptation parameter and plant response table (see table 2.1). The fuzzy rule table is arranged PID controller for fast settling to reference speed without overshoot.

Table 3.1. Fuzzy rule base for DC motor setup

	$\dot{\theta}_2$ \ $\dot{\theta}_1$	**NB**	**NS**	**Z**	**PS**	**PB**
ΔK_p	**NB**	NB	NB	Z	PS	PS
	NS	NB	NS	Z	PS	PS
	Z	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	NS	Z	PS	PB	PB
ΔK_i	**NB**	NB	NB	NS	PS	Z
	NS	NB	NS	NS	PS	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	PS	PS	PS	PB
	PB	Z	PS	PS	PB	PB
ΔK_d	**NB**	NB	NB	Z	PS	NB
	NS	Z	NS	Z	Z	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	Z	PS	PS	Z
	PB	PS	PS	PS	PB	PB

The table 3.1 is used for defuzzification of the adaptive control. In the experiment, all parameters of PID controller are updated using ΔKp, ΔK , ΔKd part of the table. Each

process, in the adaptive controller, has particular refresh frequency and number of samples for DC motor setup. The table 3.2 shows frequency and samples of the processes.

Table 3.2. Refresh frequency and sample for DC motor setup

Process	*Frequency*	*Process Samples*
Model Update	2.5 Hz	4
Adaptation	10 Hz	1
Identifications	2.5 Hz	4
PID Control	10 Hz	1
Encoder Read	10 Hz	1

The model output is updated for every 4 samples. The identification process is also run for every 4 samples. PID control and encoder reading processes are made for every sample. The PID control refresh time is 100 ms. Therefore, the lowest refresh frequency is 10 Hz for the system.

3.2. Basic Servo System

The Basic servo system consists of a servo motor and IMU. The servo motor is used for settling desired roll angle. In order to drive the servo motor, a PWM signal is produced by microcontroller. The algorithms of adaptive controller run on computer. There is serial communication protocol between microcontroller and computer. In this system, PI controller is used as base controller. The block diagram of the basic servo system can be seen in figure 3.4.

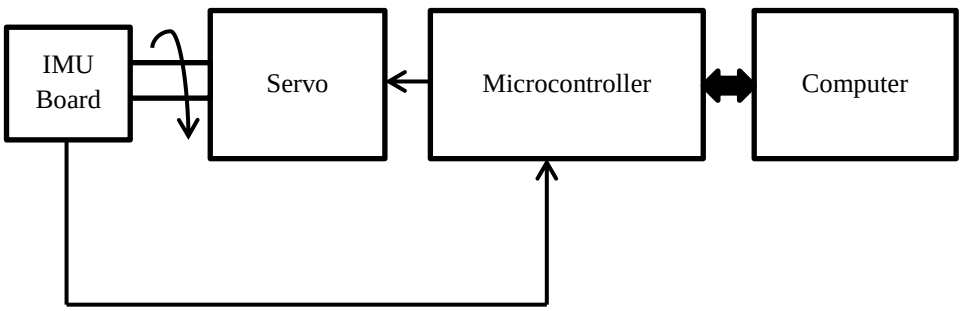


Figure 3.4 Basic servo system block diagram

The IMU sensor board is connected to servo motor shaft and measures the roll angle change. However, IMU board is capable to measure three axes orientation changes. The figure 3.5 shows the basic servo system.

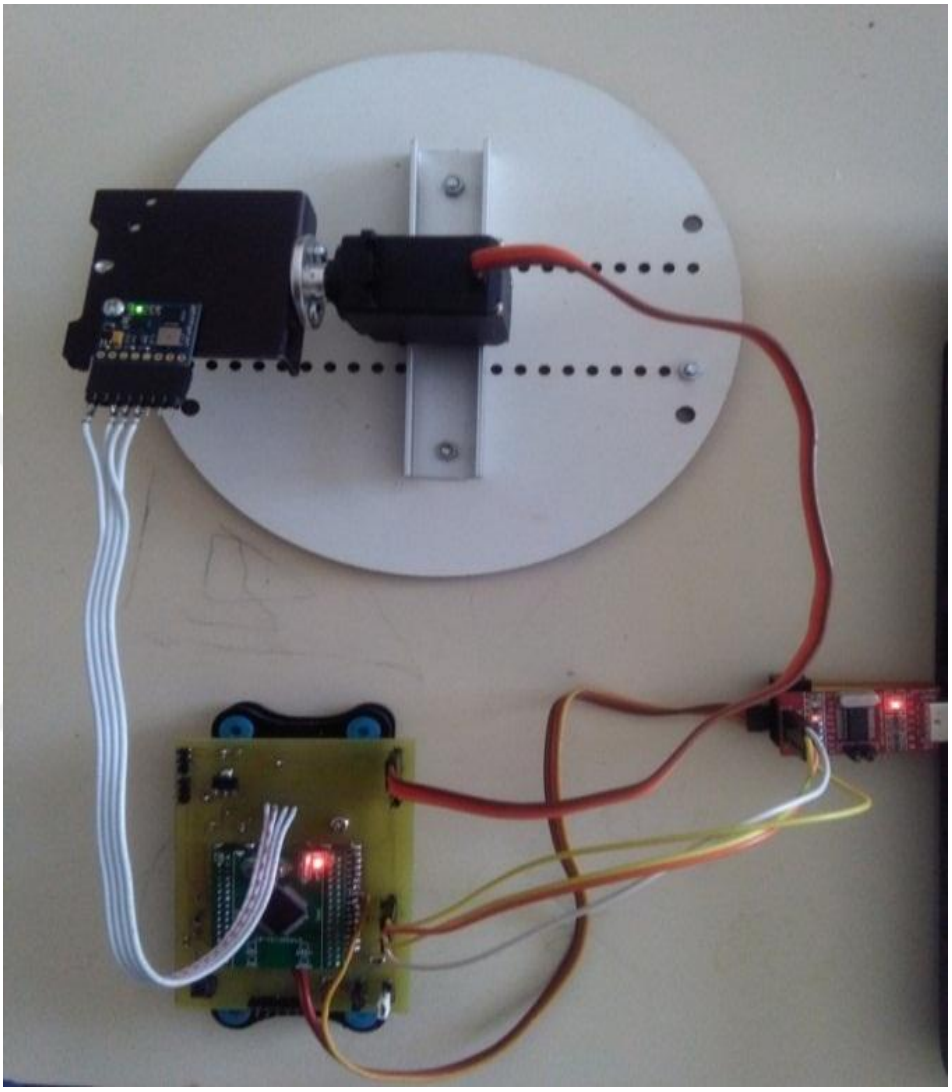


Figure 3.5 The basic servo system

In the IMU, STM32f103RBT6 microcontroller is used and a sensor board is connected to the microcontroller. For communication, UART port of the microcontroller is connected to the computer with a Serial-USB converter. The detail of the IMU board is outlined in the next section.

The PI algorithm is implemented to IMU board to control the servo motor. For PI controller, fuzzy rule base can be seen in table 3.3.

Table 3.3. The fuzzy rule base for servo system

	$\dot{\theta}_2$ \ $\dot{\theta}_1$	**NB**	**NS**	**Z**	**PS**	**PB**
K_p	**NB**	NB	NB	Z	PS	PS
	NS	NB	NS	Z	PS	PS
	Z	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	NS	Z	PS	PB	PB
K_i	**NB**	NB	NB	NS	PS	Z
	NS	NB	NS	NS	PS	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	PS	PS	PS	PB
	PB	Z	PS	PS	PB	PB

The table 3.3 is used for defuzzification part of the adaptive control. The table is the same as the table 2.2 and because of the PI controller, only *Kp* and *Ki* parts are used. Each process, in the servo system, has particular refresh frequency and number of samples. The table 3.4 shows distribution of frequency and samples for servo system.

Table 3.4. The refresh frequency and samples for servo system

Process	***Frequency***	***Process Samples***
Model Update	6.25 Hz	16
Adaptation	100 Hz	1
Identifications	6.25 Hz	16
PI Control	100 Hz	1
Sensor Fusion	100 Hz	1
Measurement	100 Hz	1

The model output is updated for every 16 samples. The identification processes also run for every 16 samples. The adaptation processes run for every sample. The PI control, sensor fusion and measurement processes are made for every sample. So, the essential refresh time for the system is 10 ms and the base refresh frequency is defined as 100 Hz.

3.2.1. MEMS Based Inertial Measurement Unit

Accurate orientation measurements are necessary to control many systems. Inertial Measurement Unit is an essential device which is used for orientation measurement. An IMU can be designed with different types of sensors. Sensor based electronics, mechanics, optics are commonly used for IMU design. In last decade, Micro Electro-Mechanical System (MEMS) types of sensors are widely used, because of its inexpensiveness and fast fabrication. Nowadays, in almost every smart phones, many navigation and robotic systems, MEMS sensors are used to measure the orientation. Despite the wide usage of MEMS sensors, their measurements are corrupted from various noise sources

MEMS based IMU needs three different sensors to measure three axes orientation angle changes. Gyroscope, accelerometer and magnetometer are MEMS based sensor which are commonly used for IMU design.

Accelerometer sensors are able to measure applied acceleration on its axis. Therefore, accelerometer can measure orientation change using gravitational force vector. However, these sensors are capable of only measuring two axes orientation angles change since the gravitational vector intersects only two planes. Thus, accelerometer sensors are generally applied to roll and pitch measurements. However, MEMS based accelerometer is very sensitive to environmental noise. Therefore, this sensor does not provide reliable orientation measurements.

Gyroscope is another sensor which can measure angular rate change on its axis. This sensor is capable of measuring three axes orientation angle changes. However, MEMS based gyroscope does not provide reliable orientation measurement because of bias drift problem. Thus, fusion of accelerometer and gyroscope makes it available to avoid noise and bias drift error. But this sensor group can only measure roll and pitch change. So, another sensor is needed to combine with gyroscope for measuring yaw angle. Magnetometer is a general yaw sensor which works like compass and it can measure magnetic field of earth. Fusion of magnetometer and gyroscope provides reliable yaw measurement.

The designed IMU has four MEMS based sensor. MPU6050 is used for gyroscope and accelerometer. HMC5883L, another sensor, is an electronic compass. The last sensor in the IMU is BMP085, a barometric pressure sensor. Gyroscope and accelerometer sensors are in the same package in MPU6050. Both sensors, inside the MPU6050, have 16 bit Analog Digital Converter (ADC). Therefore, the MPU6050 can give digital output directly.

Gyroscope is used for three axes angular velocity measurement and accelerometer is used for gravitational force vector measurement in two planes. HMC5883L is a three axes magnetometer sensor. This sensor can measure the strength and direction of the local magnetic field. The magnetometer sensor can define the yaw angle with respect to magnetic North Pole of the Earth. BMP085 is a one axis barometric pressure sensor, which is capable of measuring altitude using air pressure and temperature change. The designed IMU system is shown in figure 3.6.

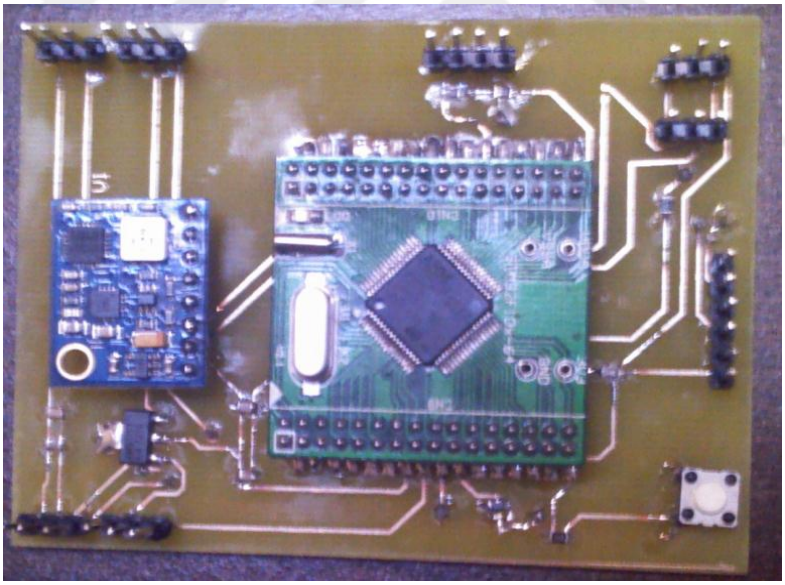


Figure 3.6 Inertial Measurement Unit (IMU)

Hardware of the IMU is based on Cortex M3 based microcontroller and sensor board. All sensors are in one board and connected to the microcontroller over Inner Integrated Circuit Communication (I2C) bus.

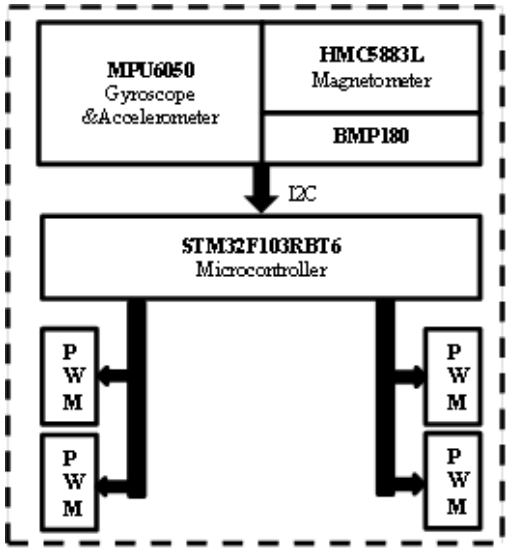


Figure 3.7 Hardware of the IMU

The figure 3.7 shows the hardware of the IMU. The microcontroller reads all sensors and makes necessary calculations in real time. Serial interface of the microcontroller is used for communication. The IMU board has four PWM output port to control motors.

3.2.1.1. Sensor Model and Calibration

The designed IMU system has ten degrees of freedom. Therefore, system can take ten measurements. Measurement axes and symbol of the IMU are shown in figure

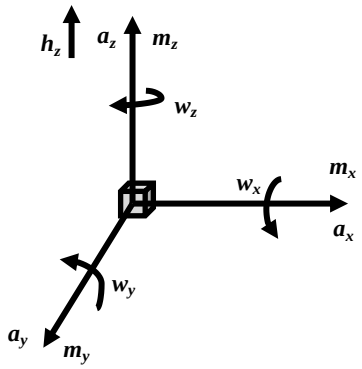


Figure 3.8 Measurement axes of the IMU

Tree axes gyroscope sensor measurements are represented by w_x, w_y, w_z. Three axes accelerometer measurements are represented by a_z, a_y, a_z. The measurements of the magnetometer are represented by m_x, m_y, m_z. The barometric pressure sensor is represented by only h_z symbol.

MEMS based gyroscope and accelerometer sensor have similar structural error source. To use these sensors, error source should be defined and subtracted from measurement. Offset error, misalignment and scale factor error are main structural error sources for MEMS based gyroscope and accelerometer. Gyroscope and accelerometer are expected to have no signal output when sensor is in stationary condition. However, MEMS based gyroscope and accelerometer give signal outputs in stationary condition, because of vibration of micro-mechanical mass inside the sensor. These errors are called offset errors. They are added to sensor measurements as structural noise. Gyroscope and accelerometer have another error source because of misalignment of the micro mechanical mass. The misalignment error is caused by positioning of micro-mechanical mass in fabrication process. Scale factor error is produced by ADC unit of sensors, because some data is lost during the conversion. In order to overcome these structural noises, the sensors should be calibrated. Every sensor, in IMU, has its own calibration procedure.

In stationary condition, accelerometer sensor is positioned as figure 3.8, and the sensor measures the gravitational force (g=9.81m/s^2) in z axis. So accelerometer measurement equation can be defined as

$$\underbrace{\begin{bmatrix} a_{x\_raw} \\ a_{y\_raw} \\ a_{z\_raw} \end{bmatrix}}_{A_{raw}} = \underbrace{\begin{bmatrix} s_{ax} & 0 & 0 \\ 0 & s_{ay} & 0 \\ 0 & 0 & s_{az} \end{bmatrix}}_{S_a} \underbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}}_{C_a} \underbrace{\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}}_{A} + \underbrace{\begin{bmatrix} a_{x\_offset} \\ a_{y\_offset} \\ a_{z\_offset} \end{bmatrix}}_{A_{offset}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}}_{G} \quad \textbf{(3.1)}$$

Where, $\boldsymbol{a}_{x\_offset}$, $\boldsymbol{a}_{y\_offset}$, $\boldsymbol{a}_{z\_offset}$ represent offset error and, $\boldsymbol{s}_{ax}$, $\boldsymbol{s}_{ay}$ and $\boldsymbol{s}_{az}$ are scale factor error. $\boldsymbol{c}_{11}$, $\boldsymbol{c}_{12}$, $\boldsymbol{c}_{13}$, $\boldsymbol{c}_{21}$, $\boldsymbol{c}_{22}$, $\boldsymbol{c}_{23}$, $\boldsymbol{c}_{31}$, $\boldsymbol{c}_{32}$, $\boldsymbol{c}_{33}$ show misalignments effect to measurement axes. $\boldsymbol{a}_x$, $\boldsymbol{a}_y$ and $\boldsymbol{a}_z$ are real acceleration data. The $\boldsymbol{a}_{x\_raw}$, $\boldsymbol{a}_{y\_raw}$ and $\boldsymbol{a}_{z\_raw}$ are output of the accelerometer sensor with structural errors. The calibrated acceleration data can be expresses as:

$$A = [A_{raw} - A_{offset} - G][S_a C_a]^{-1} \quad \textbf{(3.2)}$$

A_{offset} and $[S_a C_a]^{-1}$ must be defined experimentally. For offset error, root mean squares of the N samples are used to get A_{offset}:

$$A_{offset} = \begin{bmatrix} \sqrt{\frac{1}{N}\sum_{i=0}^{N} a_{x\_raw}^2} \\ \sqrt{\frac{1}{N}\sum_{i=0}^{N} a_{y\_raw}^2} \\ \sqrt{\frac{1}{N}\sum_{i=0}^{N} a_{z\_raw}^2} \end{bmatrix} \quad \textbf{(3.3)}$$

Angles can be computed as follows:

$$\begin{pmatrix} \theta_{xz} \\ \theta_{yz} \end{pmatrix} = \begin{pmatrix} \tan^{-1}(\frac{a_x}{a_z}) \\ \tan^{-1}(\frac{a_y}{a_z}) \end{pmatrix} \quad \textbf{(3.4)}$$

MEMS based accelerometer provide roll and pitch angles according to its axis. θ_{xz} represents roll angle and θ_{yz} represents pitch angle.

MEMS based gyroscope sensors measure the angular rate change on its axis. The measurements of gyroscope sensor are effected from offset, misalignment and scale factor errors. In order to calibrate the gyroscope, the following is used:

$$\underbrace{\begin{bmatrix} \omega_{x\_raw} \\ \omega_{y\_raw} \\ \omega_{z\_raw} \end{bmatrix}}_{W_{raw}} = \underbrace{\begin{bmatrix} s_{wx} & 0 & 0 \\ 0 & s_{wy} & 0 \\ 0 & 0 & s_{wz} \end{bmatrix}}_{S_w} \underbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}}_{C_w} \underbrace{\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}}_{W} + \underbrace{\begin{bmatrix} \omega_{x\_offset} \\ \omega_{y\_offset} \\ \omega_{z\_offset} \end{bmatrix}}_{W_{offset}} \quad \textbf{(3.5)}$$

Where, $\boldsymbol{\omega}_{x\_offset}$, $\boldsymbol{\omega}_{y\_offset}$, $\boldsymbol{\omega}_{z\_offset}$ represent offset error and, $\boldsymbol{s}_{\omega x}$, $\boldsymbol{s}_{\omega y}$ and $\boldsymbol{s}_{\omega z}$ are scale factor error. $\boldsymbol{c}_{11}$, $\boldsymbol{c}_{12}$, $\boldsymbol{c}_{13}$, $\boldsymbol{c}_{21}$, $\boldsymbol{c}_{22}$, $\boldsymbol{c}_{23}$, $\boldsymbol{c}_{31}$, $\boldsymbol{c}_{32}$, $\boldsymbol{c}_{33}$ represent misalignments effect to measurement axes. $\boldsymbol{\omega}_x$, $\boldsymbol{\omega}_y$ and $\boldsymbol{\omega}_z$ are calibrated angular rate data. The $\boldsymbol{\omega}_{x\_raw}$, $\boldsymbol{\omega}_{y\_raw}$

and $\boldsymbol{\omega}_{z\_raw}$ are direct output of the gyroscope sensor with structural errors. The matrix representation of calibration procedure is given as:

$$W = [W_{raw} - W_{offset}][S_w C_w]^{-1} \tag{3.6}$$

W_{offset} and $[S_w C_w]^{-1}$ must be defined experimentally. For offset error, root mean square of the N sample measurements is used to get W_{offset}:

$$W_{offset} = \begin{bmatrix} \sqrt{\frac{1}{N}\sum_{i=0}^{N} \omega_{x\_raw}^2} \\ \sqrt{\frac{1}{N}\sum_{i=0}^{N} \omega_{y\_raw}^2} \\ \sqrt{\frac{1}{N}\sum_{i=0}^{N} \omega_{z\_raw}^2} \end{bmatrix} \tag{3.7}$$

Magnetometer is designed for measuring the earth magnetic field force and direction. Although, magnetic field direction and force, in the earth, is relatively constant in time, electric currents in the ionosphere can alter the magnetic field. However, magnetic field is assumed constant in time for MEMS type magnetic field sensor [35]. Besides this effect, MEMS based magnetometer measurements are corrupted by noises which are produced by electronic and metal components. Therefore, there are two main noise sources for magnetometers. They are called soft and hard iron noises [36]. The magnetometer measurement equation is given as:

$$\underbrace{\begin{bmatrix} m_{x\_raw} \\ m_{y\_raw} \\ m_{z\_raw} \end{bmatrix}}_{M_{raw}} = \underbrace{\begin{bmatrix} S_{mx} & 0 & 0 \\ 0 & S_{my} & 0 \\ 0 & 0 & S_{mz} \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}}_{M} + \underbrace{\begin{bmatrix} m_{x\_noise} \\ m_{y\_noise} \\ m_{z\_noise} \end{bmatrix}}_{M_{noise}} \tag{3.8}$$

Where, magnetic noises are represented by m_{xsin}, m_{ysin}, m_{zsin}. The scale factor errors are denoted by s_{mx}, s_{my}, and s_{mz}.

In the IMU, magnetometer is used to define yaw angle with composition of *x* and *y* component of magnetic field vectors. The yaw angle calculation is performed by the following;

$$\left(\theta_{xy}\right)=\left(\tan^{-1}\frac{m_y}{m_x}\right) \tag{3.9}$$

In the IMU, there is a barometric pressure sensor which consists of piezoelectric pressure and temperature sensor. The sensor provides altitude with pressure and temperature measurement. The barometric pressure sensor is not used any filtering and calibration process in this study.

The calibration is applied only to accelerometer and gyroscope sensors. For calibrating accelerometer and gyroscope, a sensor test mechanism is built. The mechanism can be seen in figure 3.9.

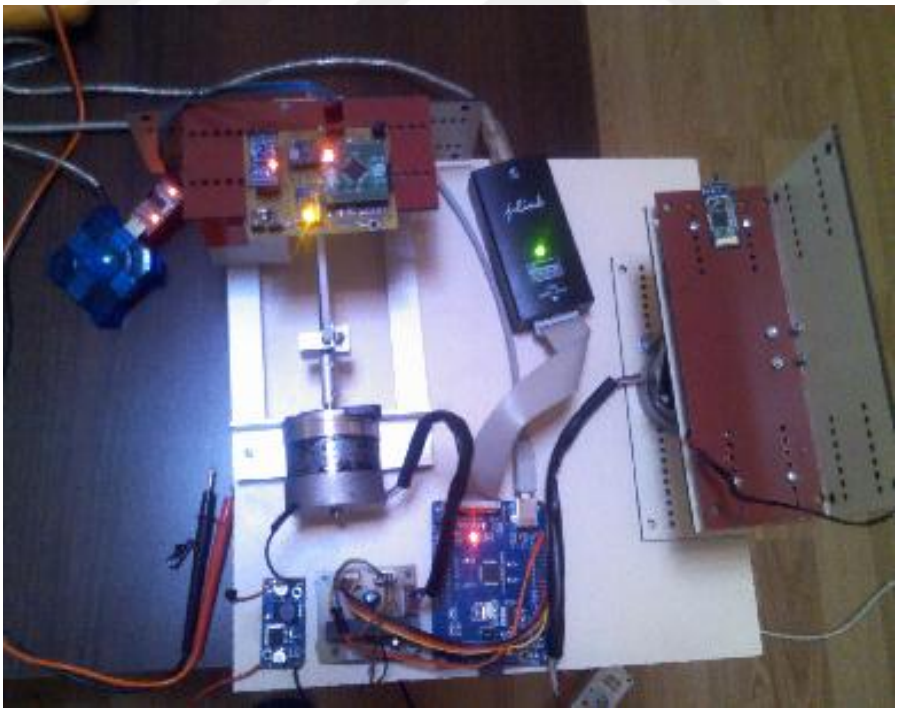


Figure 3.9 The calibration setup

There are two step motors in the mechanism. One step motor is used for three axes gyroscope calibration with constant rotation. The other motor is used for two axes accelerometer calibration with constant angle. In order to drive step motors, micro step motor driver is used. The mechanism has 0.225 degree/step resolution.

First, the offset error of gyroscope and accelerometer is defined for three axes with measurements in stationary condition. For offset error, (3.3) and (3.7) are used. After that, the resultant gravitational forces on accelerometer axes are measured for 18 degree steps and the gyroscope rotates with 52.25 degree/second for its three axes. For every test, the mean of 1000 measurements is taken and calibration matrix is calculated using (3.2) and (3.6). The calibration matrix of gyroscope and accelerometer is defined by:

$$[S_w C_w]^{-1} = \begin{bmatrix} 1.0657 & -0.04206 & 0.03751 \\ 0.00844 & 1.0734 & 0.01478 \\ -0.03256 & 0.00874 & 1.1636 \end{bmatrix} \quad \textbf{(3.10)}$$

$$[S_a C_a]^{-1} = \begin{bmatrix} 1.0978 & 0.001245 & -0.001245 \\ 0.001155 & 1.0965 & -0.002078 \\ 0.001965 & 0.001064 & 1.1943 \end{bmatrix} \quad \textbf{(3.11)}$$

In order to calibrate magnetometer sensor, the self-calibration methods, which is explained in datasheet of magnetometers, are used [37].

3.2.2. Linear Kalman Filter

Kalman filter is an optimal filter which is based on mean square error minimization. It uses a set of mathematical equations which estimates the state of a system. Kalman filter has a very wide application area, particularly in autonomous or assisted Navigation system. It is capable of providing more reliable sensor measurement in noisy environment. For this reason, Kalman filter is an important part of IMU system. It is used mainly for sensor fusion algorithm in IMU. Kalman filter algorithm has two main parts; one is time update and the other one is measurement update. These two parts call each other recursively. Because of the recursive structure,

Kalman filter can be used for real time systems. State equations of a discrete time linear measurement system are given as:

$$x_k = A\hat{x}_{k-1} + Bu_k + w_{k-1} \quad \textbf{(3.12)}$$

$$z_k = Hx_k + v_k \quad \textbf{(3.13)}$$

Where, *A*, *B* and *H* represent the state space matrices. White noises of the system are represented by w_{k-1} and v_k. In (3.12), x_k is signal value and u_k is control signal and in (3.13), z_k represents measurement value. For measurement system, there is no control signal, so Bu_k is negligible. The system covariance matrix is defined as:

$$E[w_k w_k^T] = Q_k \quad \textbf{(3.14)}$$

$$E[v_k v_k^T] = R_k \quad \textbf{(3.15)}$$

The system state estimation is defined as:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \quad \textbf{(3.16)}$$

In (3.16), K_k represents Kalman gain and is computed by:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad \textbf{(3.17)}$$

The prior and posterior error covariance matrices are given as:

$$P_k^- = AP_{k-1}A^T + Q \quad \textbf{(3.18)}$$

$$P_k = (I - K_k H)P_k^- \quad \textbf{(3.19)}$$

The recursive Kalman filter algorithm is shown in figure 3.10.

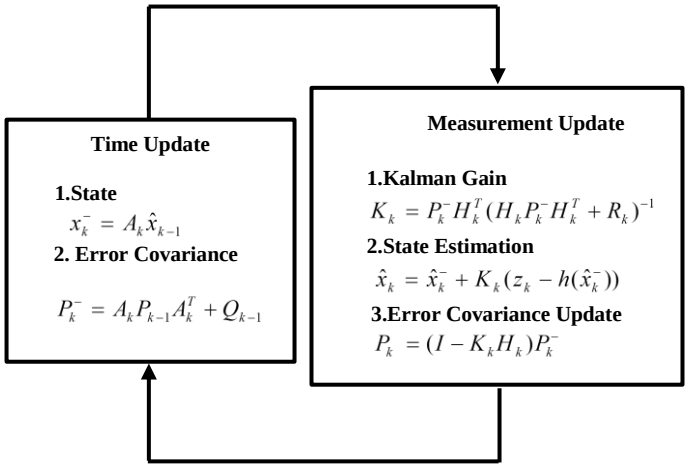


Figure 3.10 Kalman filter algorithm

The details of the above equations can be found in Welch and Bishop [38]. Kalman filter in IMU has six inputs and three outputs. The accelerometer sensor provides roll and pitch angles which are combined gyroscope roll and pitch angular rate changes with Kalman filter. The magnetometer sensor provide yaw angle which is combined gyroscope yaw angular rate change with Kalman filter.

Kalman filter gives roll, pitch and yaw angles of the system. Angle measurement block is shown in figure 3.11.

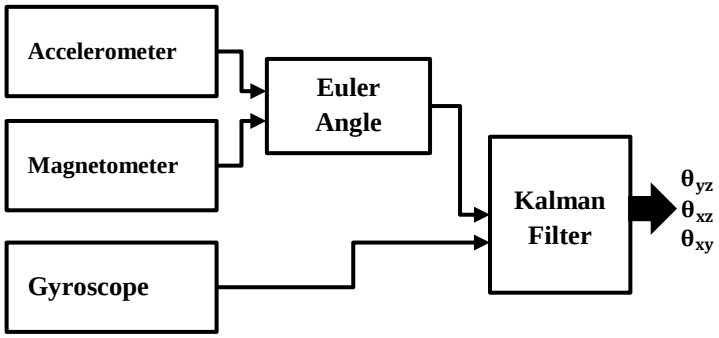


Figure 3.11 Sensor fusion block

3.2.3. Quaternion Based Extended Kalman Filter

Kalman filter is introduced as a solution for linear systems. However, many practical systems are not linear. Therefore, Extended Kalman Filter (EKF) provides solution for nonlinear systems.

Moreover, there is a gimbal lock problem for three axes orientation measurements with IMU. When one measurement axis of IMU can be parallel another, one degree of freedom is lost [39]. So this causes gimbal lock. In order to eliminate gimbal effect problem, quaternions are generally used. For many three axes IMU application, quaternion and EKF are used together. In the next sub sections, quaternion based extended Kalman filter implementation will be explained.

3.2.3.1. Quaternions

Quaternions are one of the ways of representing attitudes. Quaternions are introduced by Irish mathematician William Rowan Hamilton in 1843. It is widely used in representing orientation and rotation of object in three axes plane.

The basic definition of quaternions is

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{3.20}$$

A quaternion can be represented by a vector as follows:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.21}$$

The adjoint of quaternions is defined as:

$$q^* = \begin{bmatrix} q_0 - q_1 - q_2 - q_3 \end{bmatrix}^T \tag{3.22}$$

The norm and inverse of quaternions are defined as:

$$| q | = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3.23}$$

$$q^{-1} = \frac{q^*}{| q |} \tag{3.24}$$

The multiplication of *i, j, k* are defined as $i^2=j^2=k^2=ijk=1$, $j=k$, $jk=i$, $ki=j$, $ji=-k$, $kj=-i$, $ik=-j$.

The direct cosine matrix of unit quaternion is defined as:

$$C=\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_0q_3-q_1q_2) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1-q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \quad \textbf{(3.24)}$$

The Euler angle to unit quaternion conversion can be expressed as:

$$\begin{aligned} q_0 &= \cos(\phi/2)\cos(\theta/2)\cos(\psi/2)+\sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ q_1 &= -\cos(\phi/2)\sin(\theta/2)\sin(\psi/2)+\cos(\theta/2)\cos(\psi/2)\sin(\phi/2) \\ q_2 &= \cos(\phi/2)\cos(\psi/2)\sin(\theta/2)+\sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ q_3 &= \cos(\phi/2)\cos(\theta/2)\sin(\psi/2)-\sin(\phi/2)\cos(\psi/2)\sin(\theta/2) \end{aligned} \quad \textbf{(3.26)}$$

ϕ,θ,ψ represent roll, pitch and yaw angle. Unit quaternion to Euler angle conversion can be expressed as:

$$\begin{aligned} \phi &= \arctan 2(2q_2q_3+2q_0q_1, 1-2(q_1^2+q_2^2)) \\ \theta &= -\arcsin(2q_1q_3-2q_0q_2) \\ \psi &= \arctan 2(2q_1q_2+2q_0q_3, 1-2(q_2^2+q_3^2)) \end{aligned} \quad \textbf{(3.27)}$$

The unit quaternion propagation is defined with angular rate change with following expression:

$$\begin{aligned} \dot{q} &= \frac{1}{2}q.p(\Omega) \\ &= \frac{1}{2}\begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \end{aligned} \quad \textbf{(3.28)}$$

3.2.3.2. Extended Kalman Filter (EKF) Implementation

In EKF, nonlinear systems are linearized and then linear Kalman filter is applied [40, 41]. For nonlinear systems, process model can be defined as:

$$\dot{x}(t) = f(x(t), u(t), w(t)) \tag{3.29}$$

where x represents the state vector and system input vector is defined by u. The w represents process noise.

The measurement equation is given as:

$$y(t) = h(x(t)) + v(t) \tag{3.30}$$

where, v represents measurement noise. For linearization, an operation point is defined and the linearized model is given as:

$$\begin{aligned} \dot{\delta}x(t) &= \frac{\delta f}{\delta x}|_{\hat{x},\hat{u}} .\delta x(t) + \frac{\delta f}{\delta u}|_{\hat{x},\hat{u}} .\delta u(t) + \delta w(t) \\ \delta y(t) &= \frac{\delta h}{\delta x}|_{\hat{x},\hat{u}} .\delta x(t) + \delta v(t) \end{aligned} \tag{3.31}$$

where, $\hat{x},\hat{u}$ are operation points of system and δ denotes the variation in operating point. The linearized system can be expressed in terms of δ variable as:

$$\begin{aligned} \dot{\delta}x(t) &= A(\hat{x},\hat{u})\delta x(t) + B(\hat{x},\hat{u})\delta u(t) + \delta w(t) \\ \delta y(t) &= H(\hat{x})\delta x(t) + \delta v(t) \end{aligned} \tag{3.32}$$

For discrete system, linearized equations take the following form:

$$\begin{aligned} \delta x_{k+1} &= L_k \delta x_k + M_k \delta u_k + \delta w_k \\ \delta y_k &= H_k \delta x_k + \delta v_k \end{aligned} \tag{3.33}$$

where, L_k and H_k can be expressed in term of sampling period T.

$$\begin{aligned} L_k &= e^{A_k T} \\ H_k &= H \end{aligned} \tag{3.34}$$

For EKF, process update equation can be defined as:

$$\hat{x}_k^- = \hat{x}_{k-1}^- + \int_{t_{k-1}}^{t_k} f(x(t), u(t), w(t)).dt \quad \textbf{(3.35)}$$

$$P_k^- = L_k P_{k-1} L_k^T + Q_{k-1} \quad \textbf{(3.36)}$$

where, Q_k is process noise covariance matrix for discrete time systems. The Kalman gain can be expressed as:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad \textbf{(3.37)}$$

where, R_k is measurement covariance matrix for discrete time systems. The state estimation equation is expressed with new measurement as:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \quad \textbf{(3.38)}$$

The error covariance matrix is updated with

$$P_k = (I - K_k H_k) P_k^- \quad \textbf{(3.39)}$$

For implementation, state variable and system matrix can be simplified as

$$\hat{x}_k^- \approx \hat{x}_{k-1} + f(\hat{x}_{k-1}, u_{k-1}).T \quad \textbf{(3.40)}$$

$$L_k \approx I + A_k T \quad \textbf{(3.41)}$$

In order to implement the EKF with quaternions, the state equation is described with unit propagation equation of quaternions [42]

$$\begin{aligned} \dot{q}_k &= \frac{1}{2} q_k . p(\Omega_k) \\ &= \frac{1}{2} \Omega_k q_k \end{aligned} \quad \textbf{(3.42)}$$

where Ω_k is defined as:

$$\Omega_k = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & -\omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}_k \quad \textbf{(3.43)}$$

The angular rate measurement of gyroscope is biased, so this bias must be subtracted from angular rate measurements, as follows:

$$\Omega_k = \begin{bmatrix} 0 & -(\omega_x - \omega_{x\_bias}) & -(\omega_y - \omega_{y\_bias}) & -(\omega_z - \omega_{z\_bias}) \\ (\omega_x - \omega_{x\_bias}) & 0 & (\omega_z - \omega_{z\_bias}) & -(\omega_y - \omega_{y\_bias}) \\ (\omega_y - \omega_{y\_bias}) & -(\omega_z - \omega_{z\_bias}) & 0 & (\omega_x - \omega_{x\_bias}) \\ (\omega_z - \omega_{z\_bias}) & (\omega_y - \omega_{y\_bias}) & -(\omega_x - \omega_{x\_bias}) & 0 \end{bmatrix}_k \quad \textbf{(3.44)}$$

When (3.42) is extended with Ω_k matrix, the propagation equation can be expressed as:

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}_k = \frac{1}{2} \begin{bmatrix} -(\omega_x - \omega_{x\_bias}).q_1 - (\omega_y - \omega_{y\_bias}).q_2 - (\omega_z - \omega_{z\_bias}).q_3 \\ (\omega_x - \omega_{x\_bias}).q_0 + (\omega_z - \omega_{z\_bias}).q_2 - (\omega_y - \omega_{y\_bias}).q_3 \\ (\omega_y - \omega_{y\_bias}).q_0 - (\omega_z - \omega_{z\_bias}).q_1 + (\omega_x - \omega_{x\_bias}).q_3 \\ (\omega_z - \omega_{z\_bias}).q_0 + (\omega_y - \omega_{y\_bias}).q_1 - (\omega_x - \omega_{x\_bias}).q_2 \end{bmatrix}_k \quad \textbf{(3.45)}$$

The system state vector with bias is defined as:

$$x_k = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}_k = \begin{bmatrix} \omega_{x\_bias} \\ \omega_{y\_bias} \\ \omega_{z\_bias} \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}_k \quad \textbf{(3.46)}$$

The process model can be expressed as:

$$\dot{x}_k = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{bmatrix}_k = \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -(\omega_x - x_1)x_5 - (\omega_y - x_2)x_6 - (\omega_z - x_3)x_7 \\ (\omega_x - x_1)x_4 + (\omega_z - x_3)x_6 - (\omega_y - x_2)x_7 \\ (\omega_y - x_2)x_4 - (\omega_z - x_3)x_5 + (\omega_x - x_1)x_7 \\ (\omega_z - x_3)x_4 + (\omega_y - x_2)x_5 - (\omega_x - x_1)x_6 \end{bmatrix}_k \quad \textbf{(3.47)}$$

The linearized state transition matrix is defined as:

$$A_k = \frac{\partial f}{\partial x} \quad \textbf{(3.48)}$$

$$A_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ x_5 & x_6 & x_7 & 0 & -(\omega_x - x_1) & -(\omega_y - x_2) & -(\omega_z - x_3) \\ -x_4 & x_7 & x_6 & (\omega_x - x_1) & 0 & (\omega_z - x_3) & -(\omega_y - x_2) \\ -x_7 & -x_4 & x_5 & (\omega_y - x_2) & -(\omega_z - x_3) & 0 & (\omega_x - x_1) \\ x_6 & x_5 & -x_4 & (\omega_z - x_3) & (\omega_y - x_2) & -(\omega_x - x_1) & 0 \end{bmatrix} \quad \textbf{(3.49)}$$

The measurement matrix is represented with roll pitch and yaw angles as:

$$z_k = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \quad \textbf{(3.50)}$$

The measurement matrix is calculated from accelerometer and magnetometer with (3.4) and (3.9). In (3.50), ϕ, θ, ψ represent roll, pitch and yaw angles.

The nonlinear measurement model is defined as:

$$h(x_k) = \begin{bmatrix} \arctan 2(2x_6x_7 + 2x_4x_5, 1 - 2(x_5^2 + x_6^2)) \\ -\arcsin(2x_5x_7 - 2x_4x_6) \\ \arctan 2(2x_5x_6 + 2x_4x_7, 1 - 2(x_6^2 + x_7^2)) \end{bmatrix}_k \quad \textbf{(3.51)}$$

Then, linear measurement matrix is expressed as:

$$h(x_k)=\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}\approx H_k.x_k=\left[0_{3x3}\ \tilde{H}_k\right].x_k \tag{3.52}$$

$$\tilde{H}_k=\begin{bmatrix}\frac{\partial\phi}{\partial x_4} & \frac{\partial\phi}{\partial x_5} & \frac{\partial\phi}{\partial x_6} & \frac{\partial\phi}{\partial x_7}\\ \frac{\partial\theta}{\partial x_4} & \frac{\partial\theta}{\partial x_5} & \frac{\partial\theta}{\partial x_6} & \frac{\partial\theta}{\partial x_7}\\ \frac{\partial\psi}{\partial x_4} & \frac{\partial\psi}{\partial x_5} & \frac{\partial\psi}{\partial x_6} & \frac{\partial\psi}{\partial x_7}\end{bmatrix}_k \tag{3.53}$$

The linearized measurement matrix is defined as:

$$\tilde{H}_k=\begin{bmatrix}\frac{2a_{33}x_5}{a_{33}^2+a_{32}^2} & \frac{2(a_{33}x_4+2a_{32}x_5)}{a_{33}^2+a_{32}^2} & \frac{2(a_{33}x_7+2a_{32}x_6)}{a_{33}^2+a_{32}^2} & \frac{2a_{33}x_6}{a_{33}^2+a_{32}^2}\\ \frac{2x_6}{\sqrt{1-a_{31}^2}} & \frac{-2x_7}{\sqrt{1-a_{31}^2}} & \frac{2x_4}{\sqrt{1-a_{31}^2}} & \frac{-2x_5}{\sqrt{1-a_{31}^2}}\\ \frac{2a_{33}x_5}{a_{33}^2+a_{32}^2} & \frac{2a_{11}x_6}{a_{11}^2+a_{21}^2} & \frac{2(a_{11}x_5+2a_{21}x_6)}{a_{11}^2+a_{21}^2} & \frac{2(a_{11}x_4+2a_{21}x_7)}{a_{11}^2+a_{21}^2}\end{bmatrix}_k \tag{3.54}$$

$$\begin{aligned}a_{11}&=1-2(x_6+x_7)\\ a_{33}&=1-2(x_6+x_7)\\ a_{21}&=2(x_5x_6+x_4x_7)\\ a_{31}&=2(x_5x_7-x_4x_6)\\ a_{32}&=2(x_6x_7+x_4x_5)\end{aligned} \tag{3.55}$$

Thus, measurement and state matrices are linearized. After that point, linear Kalman algorithm can be used for sensor fusion. A good implementation of quaternion extended Kalman filter for low cost IMU was made by Samuel Fox (2008) [43]. The quaternion based extended Kalman filter is coded in embedded C environment and implemented to IMU.

3.2.4. Adaptive Vibration Filter for IMU

Although Kalman filter could overcome many structural and environmental noises, when sensors, in fusion algorithm, are affected simultaneously by same noise, Kalman filter could not eliminate this type noise, such as vibration. Both gyroscope and accelerometer are affected by vibration. Therefore, the IMU needs a mechanical and software based vibration filter. In this work an adaptive filter is implemented on the IMU as software based vibration filter.

Adaptive filter is a mathematical tool which provides modeling of two signals in real time. Adaptive filter is based on the input and output structure of a signal. Parameters of filter are altered with the input-output correlation to provide desired signal output. An adaptive filter often consists of two parts. They are linear filter and adaptation algorithm. Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters are most often linear filter parts of the adaptive filters. Least Mean Square (LMS) and Mean Square Error (MSE) are most often used for adaptation algorithm of the adaptive filters [44]. There are many applications of adaptive filters such as system identification, inverse modeling, linear prediction, feedforward control and Adaptive Noise Canceller (ANC). In this work, an ANC algorithm was implemented using LMS and FIR.

The Adaptive filter, in IMU, is based on Adaptive Noise Canceller (ANC). ANC is an optimal filtering that makes an estimate of the noise in the source signal and subtracts the noise from the source signal. The basic structure of the ANC for one noise input is shown in figure 3.12.

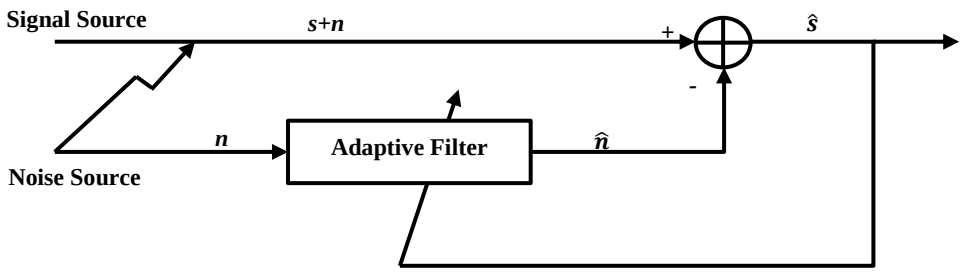


Figure 3.12 Basic adaptive noise canceller diagram

In figure 3.12, the source and the noise signals are represented by s and n symbols. $\hat{n}$ is estimated noise and $\hat{s}$ is filtered output of the adaptive filter. ANC is updated using filtering parameter of adaptation criteria. Filter and adaptation criteria are important parts of the ANC.

The criteria are generally based on the minimization of the error. Different types of algorithms can be used for the minimization criteria, such as, Least Mean Squares (LMS) algorithm, the Recursive Least Squares (RLS) algorithm, Normalized Least Mean Squares (NLMS) etc. In IMU, LMS is implemented for adaptation criteria. Because LMS is computationally easy and it could be applied on real time data simultaneously. In addition, for filtering, FIR is chosen. Adaptive filter structure could be seen in figure 3.13. LMS update the FIR filter coefficients using the steepest descent algorithm.

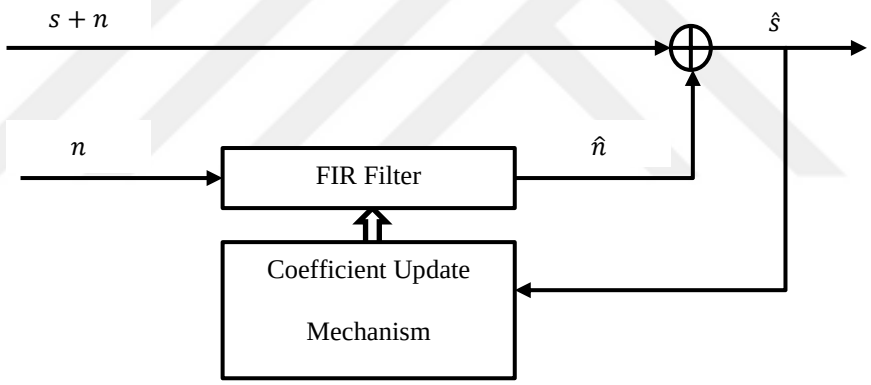


Figure 3.13 ANC block In IMU

Basic FIR filtering is defined as follows:

$$n_k = \sum_{i=0}^{K-1} b_i x_{k-i} \qquad \textbf{(3.56)}$$

Where, summation of input signal array multiplies with filter coefficient b_i. The output of the ANC is defined as.

$$\hat{s} = s + (n - \hat{n}) \qquad \textbf{(3.57)}$$

In (3.57), Noise in the signal is subtracted from noise correlated filter output. Therefore, the ANC output is expected to be close to the main signal.

Taking expectation of (3.58) yields:

$$E\left[\hat{s}^2\right] = E\left[s^2\right] + E\left[(n-\hat{n})^2\right] \tag{3.58}$$

Since signal power will not be affected by minimization which gives the following:

$$\min E\left[\hat{s}^2\right] = E\left[s^2\right] + \min E\left[(n-\hat{n})^2\right] \tag{3.59}$$

Therefore, the minimization criteria is

$$J = E\left[\hat{s}^2\right] \tag{3.60}$$

In order to obtain mean square error solution, partial derivative of (3.60) is taken with respect to filtering coefficients. So the result is shown in found as:

$$\frac{\partial J}{\partial b_i} = 2E\left[\hat{s}_k x_{k-i}\right] = 2E\left[s_k x_{k-i}\right] - 2\sum_{m=0}^{K-1} b_m E\left[x_{k-m} x_{k-i}\right] \tag{3.61}$$

Optimum solution is given as:

$$(S_x b = x_{xs}) \tag{3.62}$$

Iterative gradient algorithm of the filter coefficient is defined as:

$$\forall i \in \left[0, K-1\right] b_i(k+1) = b_i(k) + \delta E(s_k x_{k-i}) \tag{3.63}$$

where, δ represents adaptation step. The convergence criteria depend on the adaptation step as follows:

$$\left|\delta\right| < \frac{2}{\lambda_{\max}} \tag{3.64}$$

where, $\lambda_{\max}$ is maximum eigen value of the S_x. However, the mean value of $E(\hat{s}_k x_{k-i})$ is unknown, so the mean value of $E(\hat{s}_k x_{k-i})$ is replaced with $\hat{s}_k x_{k-i}$ in stochastic gradient algorithm. The convergences depend on small adaptation steps as:

$$\forall i \in [0, K-1] b_i(k+1) = b_i(k) + \delta(s_k x_{k-i}) \quad \textbf{(3.65)}$$

In IMU, three axes gyro measurements are filtered using ANC. Since analog gyroscope sensor used as vibration sensor, noise signal is correlated with main signal. But analog gyro sensor is capable of only one axis measurement.

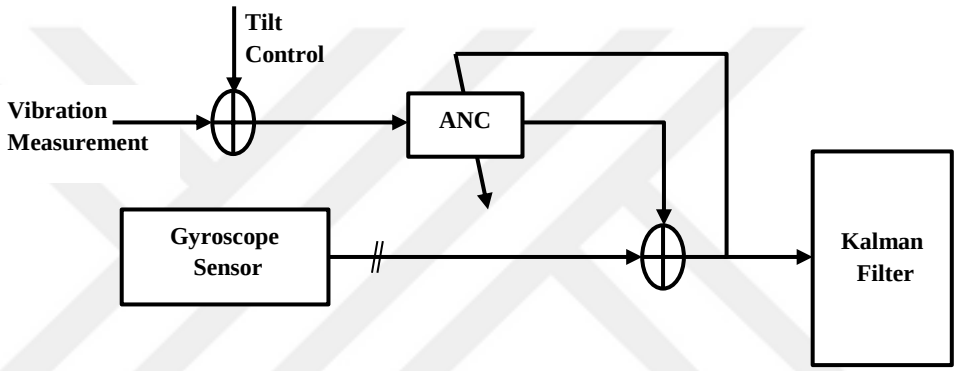


Figure 3.14 Adaptive noise canceller implementation of IMU

Therefore, analog gyroscope sensor is settled on pitch axis of system and tilt angle change is subtracted from analog gyroscope sensor. ANC works when system is stationary. The IMU implementation of ANC is shown in figure 3.14.

3.2.4.1. Experimental Results of Adaptive Vibration Filter

The designed ANC algorithm is implemented on dual tilt rotor system which has two brushless motors and two servo motors.

The brushless motors are driven by Electronic Speed Controller (ESC). The servos and brushless motors are controlled by Pulse Width Modulated (PWM) signal which is generated by the IMU. Servos are used for changing tilt angle of brushless motors. The Brushless motors apply force in the vertical direction using three blade propellers. There are two servo motor to control the pitch angle of the brushless

motor. These servo motors are also connected to the IMU. In the experiments, these servo motors are only used for arranging the brushless motor in *z* axis.

Figure 3.15 Experimental setup for ANC

The test system can be seen on figure 3.15. There are many vibration sources on the system such as mechanical structure of system, servos, and brushless motors. But main vibration sources are brushless motors and propellers. In order to increase the vibration in the system, both propellers are driven in Clock Wise (CW) direction and both brushless motors are set in CW direction. The IMU is mounted on center of the main arm with mechanical vibration absorber. Vibration sensor is mounted directly on mechanical skeleton on center of the system.

In the experiment, first, the IMU system operates and measures yaw, pitch and roll angle changes, after 20 seconds, brushless motors operates with initial speed (approximately 2700 RPM) then in every 10 seconds, the motor speed is increased 400 RPM automatically. When brushless motor speed reaches approximately 4300 RPM, the system is stopped. During the test process, IMU sends vibration value, yaw, pitch and roll angle data to computer over serial interface of microprocessor.

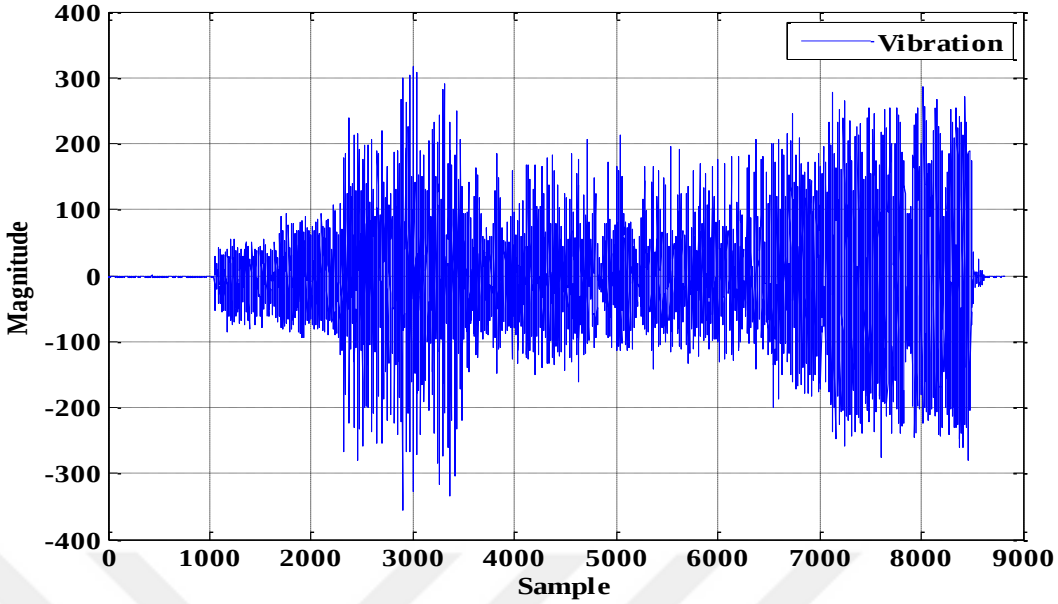


Figure 3.16 Vibration measurement

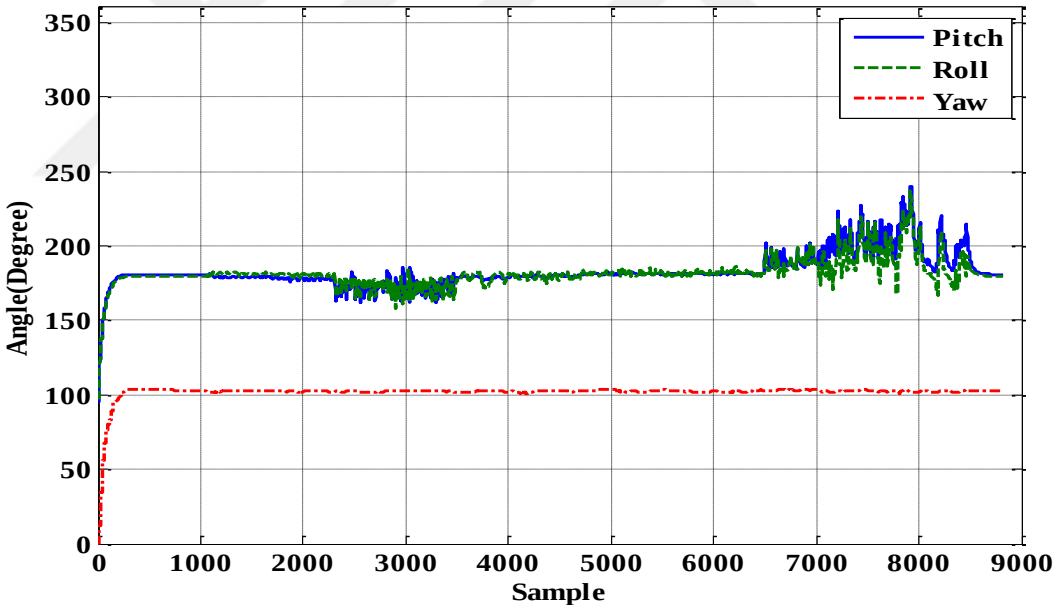


Figure 3.17 Sensor fusion output without ANC

Vibration measurement data are shown in figure 3.16. Its characteristic varies according to the speed of the system. Low and high speed of system causes more vibration; however speed change does not linearly depend on vibration because of mechanical structure of the test system.

The IMU output signal without ANC can be seen in figure 3.17. The pitch and roll angle outputs are affected by vibration of the system. But the yaw output is not affected by vibration because magnetometer does not have mechanical mass inside.

In addition, the IMU output has different behavior. Because, Kalman filter in the IMU can overcome some vibration but not all. When both gyroscope and accelerometer are affected from vibration noise, Kalman filter gives output signal as if there is an orientation change.

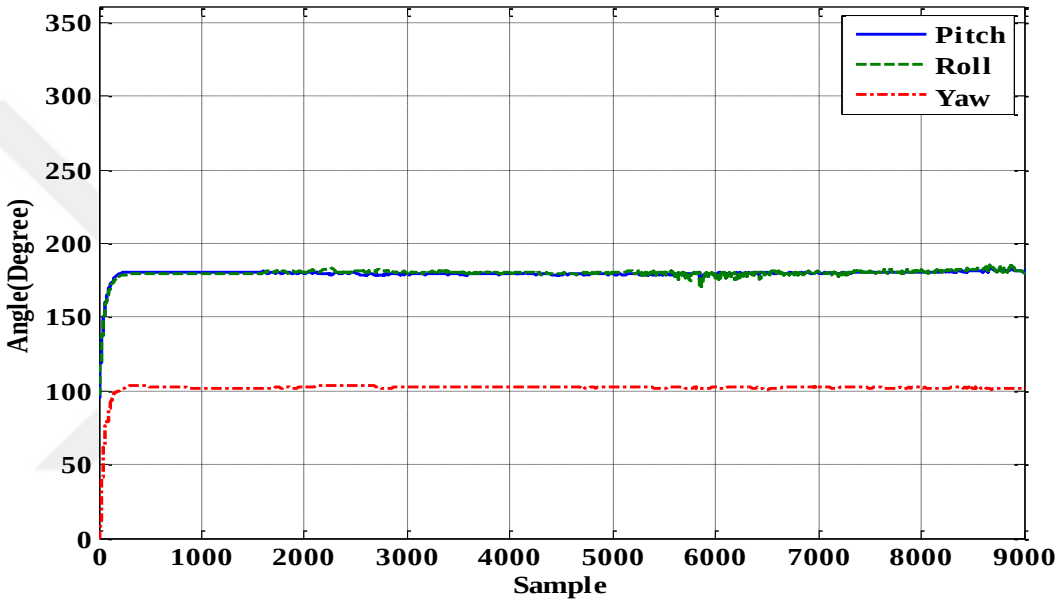


Figure 3.18 Sensor fusion output with ANC

On the other hand, when the ANC suppresses the vibration noise in gyroscope sensor, vibration affection tremendously decreases. The figure 3.18 shows the Kalman filter output signal with ANC.

In this work, an adaptive noise canceller is implemented for low cost IMU in order to suppress vibration which corrupts the Kalman filter output. Adaptive noise canceller is based on MSE and FIR filtering. All algorithms run on the microcontroller. One axis analog gyroscope sensor is used as vibration sensor. Therefore vibration data contains both tilt angle change and vibration noise. To obtain the vibration, tilt angle control input signal is subtracted from vibration sensor measurements. IMU output signal is compared with ANC and without ANC algorithm. Experimental results

show that ANC algorithm suppresses the vibration of test setup and Kalman filter provides more reliable and stable roll and pitch angle measurements. On the other hand, the vibration sensor has slow response time and ANC algorithm consumes approximately 6.3 ms CPU time for each update cycle. Therefore, system refresh time must be limited at 50 Hz, but it is possible to improve the refresh frequency using a different vibration sensor with higher response time and a faster microcontroller having higher clock frequency. The ANC algorithm is coded in embedded C environment and implemented to IMU.

3.3. Mechanical Tilt Platform

A mechanical tilt platform, which works like an inverse pendulum, is built. The platform has an actuator system which provides force for the arm of the platform. There are two rotors and propeller in the actuator. The propellers and rotors are mounted on an arm as parallel to each other. The center of the rotor arm is mounted on the pendulum arm which has a rotary end. In figure 3.19, basic drawing of the platform can be seen.

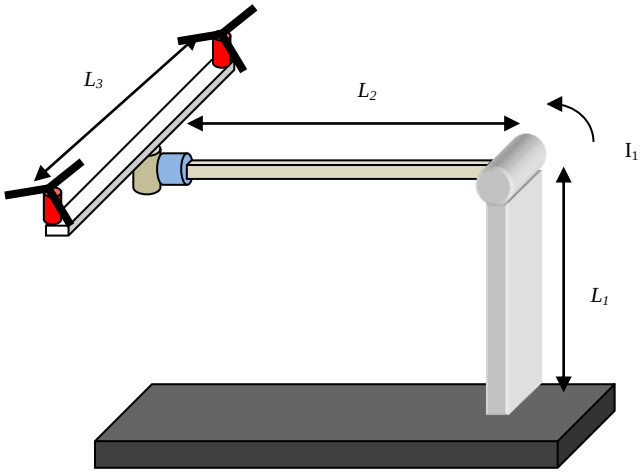
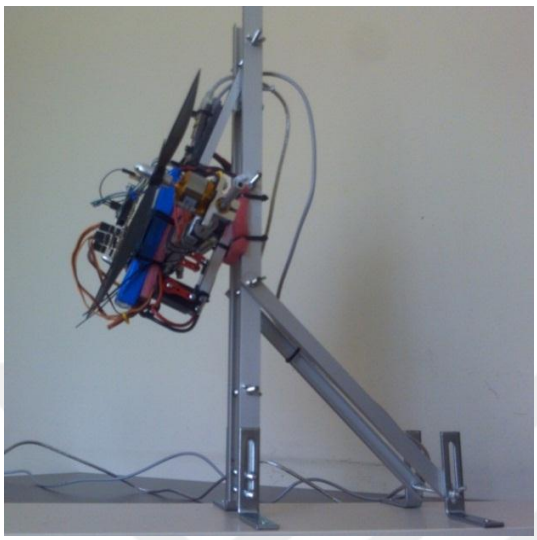


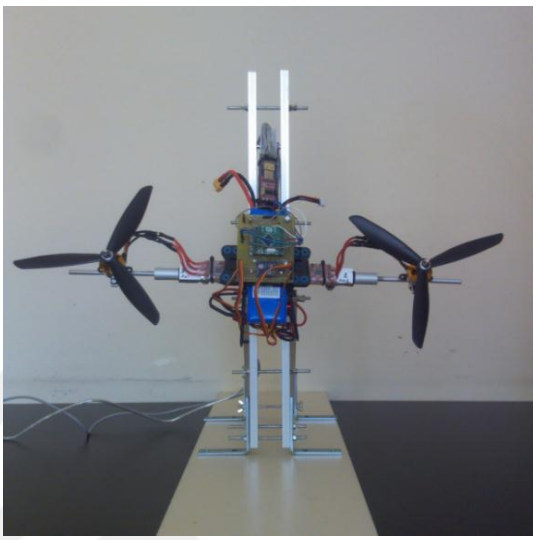
Figure 3.19 The tilt platform drawing

L_1, L_2, and L_3 are the arms of the platform. For this work, only L_2 is allowed to be capable of rotating so there is only one inertial element in the platform and its Inertial constant is denoted by I_1. There are two brushless motors with propellers on L_3 arm. The brushless motors are driven with Electronic Speed Controllers (ESC) with

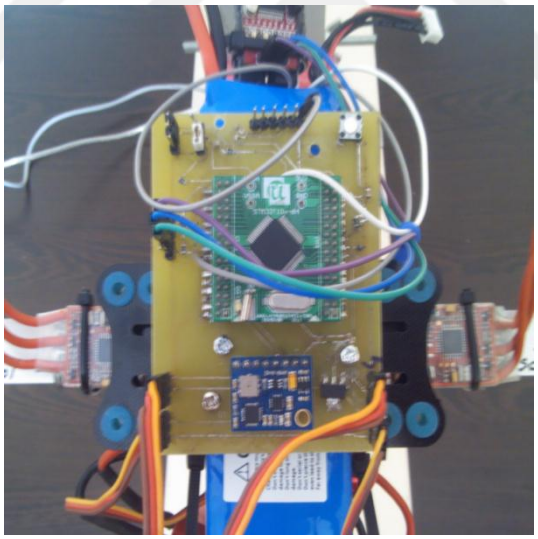
PWM. Therefore, PI controller output is applied as PWM signals. The brushless motors are 2700KV outrunner type. The propellers are chosen 8 inch three blades. The center of mass is placed on negative z axis.



(a)



(b)



(c)

Figure 3.20 a) The side view of tilt platform, **b)** The front view of tilt platform, **c)** The view of IMU board

A 11.1 volt Lithium polymer battery is used to supply the platform. The platform has also rotor tilt system which has two servo motors and rotary shaft system. The rotor tilt system is not used in this work. In the test platform, there are two different microcontrollers, despite simultaneous software processing of the controller. One of

the processors is responsible for inertial measurement, sensor fusion and the PI controller. The IMU board is used for this purpose. The figure 3.21 shows the process distribution of the controller.

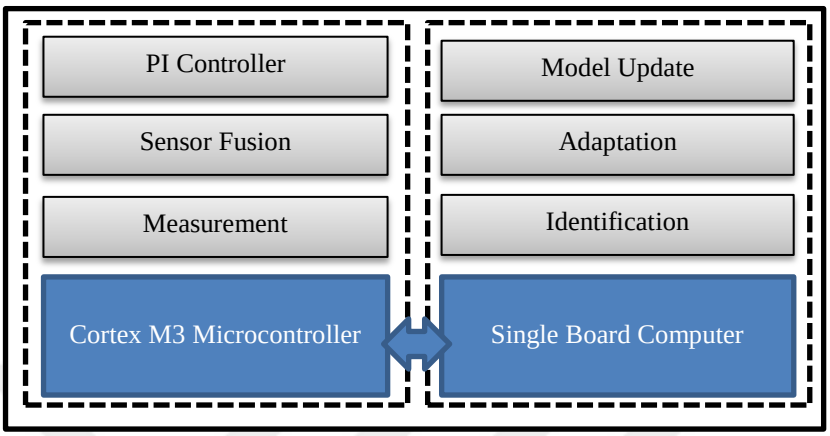


Figure 3.21 Process management for tilt platform

The other processor is responsible for adaptation, identifications, model update process. These processes need faster processor and more memory. Therefore, a Cortex A9 based Single Board Computer (SBC) is used. The figure 3.22 shows the SBC.

Figure 3.22 Single board computer

The SBC has 1 GHZ quad core ARM processor and 1 GB RAM and 4 GB NAND flash memories. There is communication interface between IMU and SBC. The SBC has a Linux operating system. Each process, in the system, has particular refresh frequency and number of samples.

Table 3.5 Processes refresh frequency and samples for tilt platform

Process	*Frequency*	*Process Samples*
Model Update	6.25 Hz	16
Adaptation	25 Hz	4
Identifications	6.25 Hz	16
PI Control	100 Hz	1
Sensor Fusion	100 Hz	1
Measurement	100 Hz	1

For the test platform, the refresh frequencies and sample number is given in table 3.5. The model output is updated for every 16 samples. The identification processes also run for every 16 samples. The adaptation processes run for every 4 samples. The PI controller, sensor fusion and measurement processes are made for every sample. So the system base refresh time is 10 ms. So, the highest refresh frequency is 100 Hz.

Table 3.6 Fuzzy rule base for tilt platform

	$\dot{\theta}_2$ \ $\dot{\theta}_1$	**NB**	**NS**	**Z**	**PS**	**PB**
ΔK_p	**NB**	NB	NB	Z	PS	PS
	NS	NB	NS	Z	PS	PS
	Z	NS	NS	Z	PS	PS
	PS	NS	Z	PS	PS	PB
	PB	NS	Z	PS	PB	PB
ΔK_i	**NB**	NB	NB	NS	PS	Z
	NS	NB	NS	NS	PS	Z
	Z	Z	Z	Z	Z	Z
	PS	Z	PS	PS	PS	PB
	PB	Z	PS	PS	PB	PB

The fuzzy rule base for tilt platform can be seen in table 3.6. The fuzzy rule base is the same as basic servo system.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1. Experimental Results of DC Motor Setup

The DC motor setup is tested only with PID controller before the adaptive controller implementation. The PID parameters are set to force the system to oscillate. The reference signal is arranged 60 RPM for first 500 samples and then its value is decreased to 40 RPM. The figure 4.1 shows the system response while the *Kp*, *Ki* and *Kd* values are set as 15, 8 and 2.1.

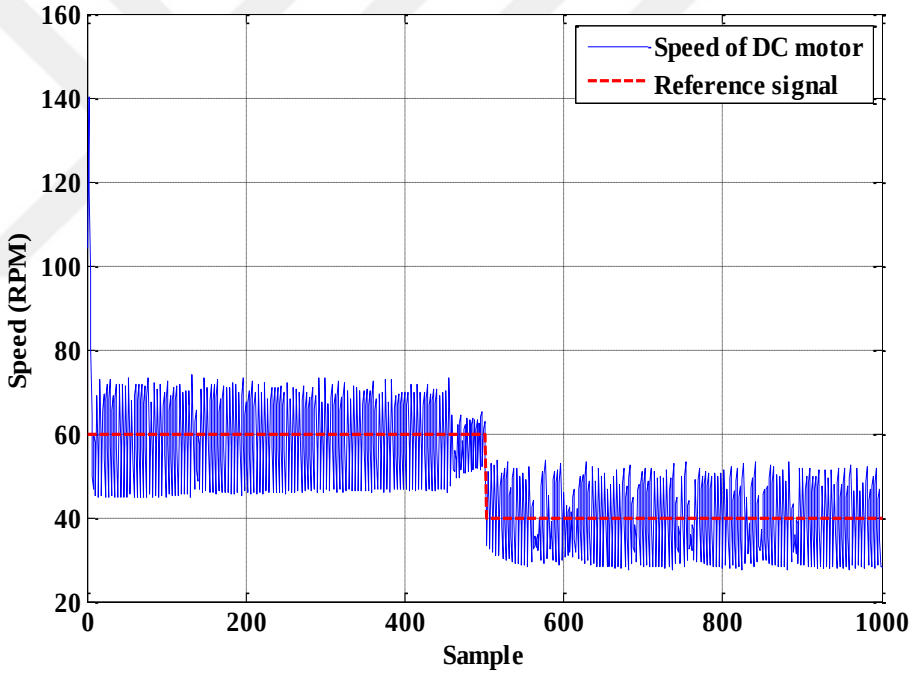


Figure 4.1 PID controller response of DC motor setup with *Kp=15*, *Ki=8* and *Kd=2.1*

In figure 4.1, The DC motor speed is oscillated around the reference signal. The deviation from the reference speed can reach ±14 RPM. After the PID controller test, the adaptive controller is implemented to the system. And the initial PID parameters are settled *Kp=15*, *Ki=8* and *Ki=2.1*. The *Kp* range is defined as 1 to 18 interval, *Ki* range is defined as 1 to 12 interval and *Kd* range is defined as 1 to 6

interval. The deadband of PI controller, which is defined for error between reference signal and plant output, is set to 1. The deadband of model and plant output error is set to 1. The final deadband, which is defined for model and reference signal error, is set to 1. For fuzzy block, PID fuzzy rule table is used (see table 3.1). The model update frequency is arranged for 4 samples. For every 4 samples, output model is revised by identification process. The linear fitting algorithm is used for identification with using GSL library. The system response with adaptive controller can be seen in figure 4.2.

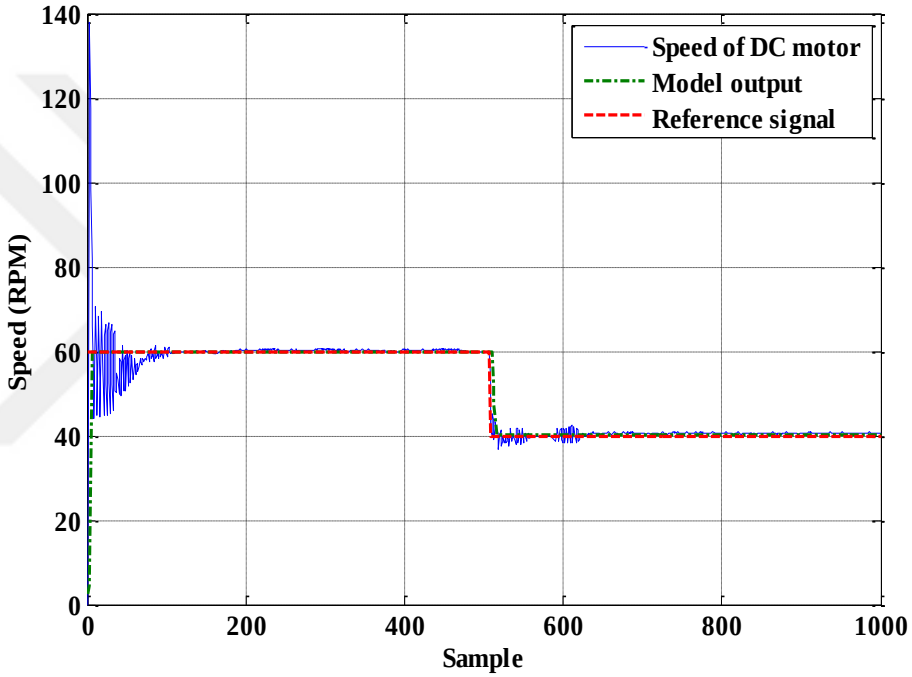


Figure 4.2 The adaptive controller response of DC motor setup

According to the figure 4.2, the system starts with oscillating, and the oscillating is decreased during the first 100 samples. After that, system state closes to and tracks the reference signals. Reference signal change causes corruption in the system response. However, the adaptation is also capable of suppressing the corruption. The adaptive controller tunes PID parameter according to the model. The model and plant error feeds Lyapunov adaptive rules and according to Lyapunov analysis, the error is expected to be close to zero. The figure 4.3 shows that the error goes to zero.

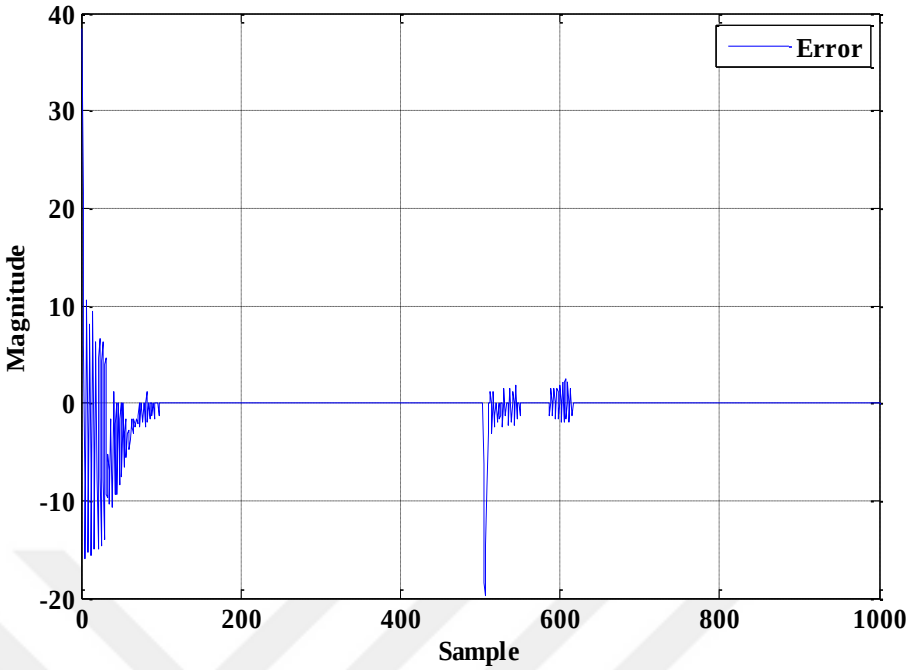


Figure 4.3 The error of model and DC motor output

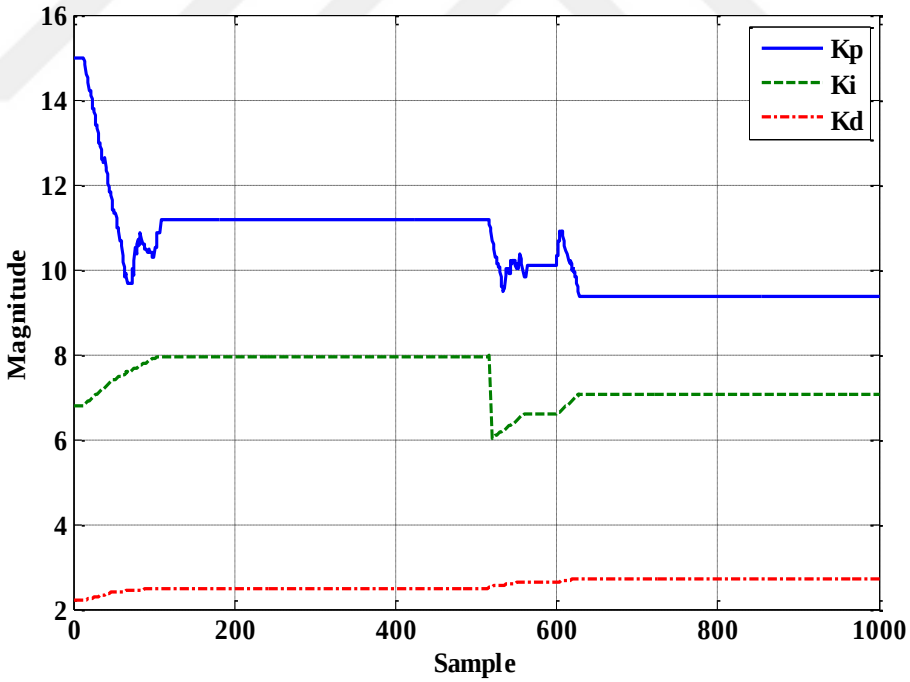


Figure 4.4 The adaptive controller response of DC motor setup

The adaptation depends on the error. When error reaches the zero, adaptation process stops and waits. In this condition, PID parameters stay at the last values. The figure

4.4 shows the parameter change of the PID controller with adaptive controller. A variable load experiment is also applied to DC motor setup. A servo motor provides variable friction on the motor shaft. The PID parameters are settled as *Kp=8*, *Ki=5* and *Kd=1.2*. First experiment is made using only PID controller with fixed parameter under load. The system response can be seen in figure 4.5. During the experiment, friction on the shaft is increased with PWM signal of the servo motor. The PWM signal is increased over servo PWM reference level. The PWM step size is defined as 8. After every 100 sample, the PWM signal is increased step by step. The final value of the PWM is 80. In figure 4.5, the load change represents the PWM signal of servo motor. The PID controller can cope with the load change for a point. After servo PWM value reaches 40, the system reponse goes to oscilating and the oscilation is increased by increasing the servo PWM signal.

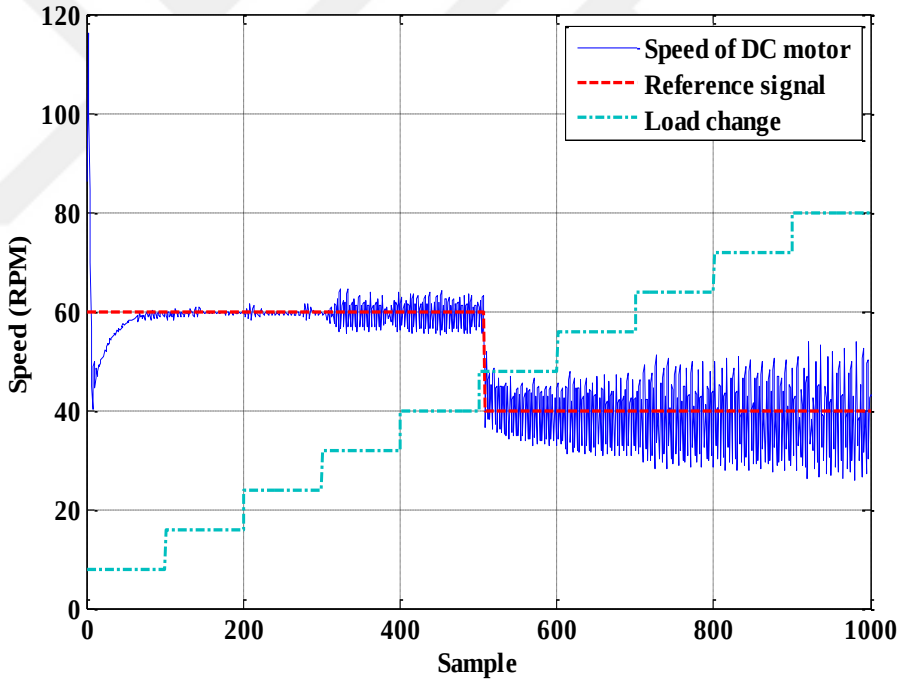


Figure 4.5 The PID response of dc motor setup under variable load

After the PID controller load experiment, the adaptive controller is implemented on the system. The figure 4.6 shows the system response with adaptive controller. The adaptive controller arranges PID controller for coping with the variation of load and forces the system track the reference signals.

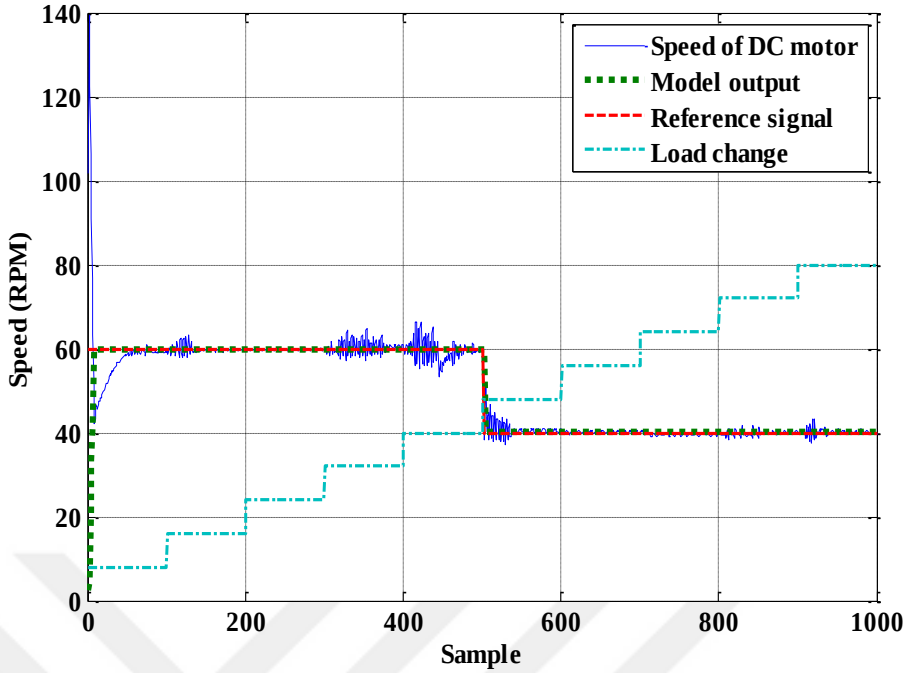


Figure 4.6 The adaptive controller response of DC motor setup under variable load

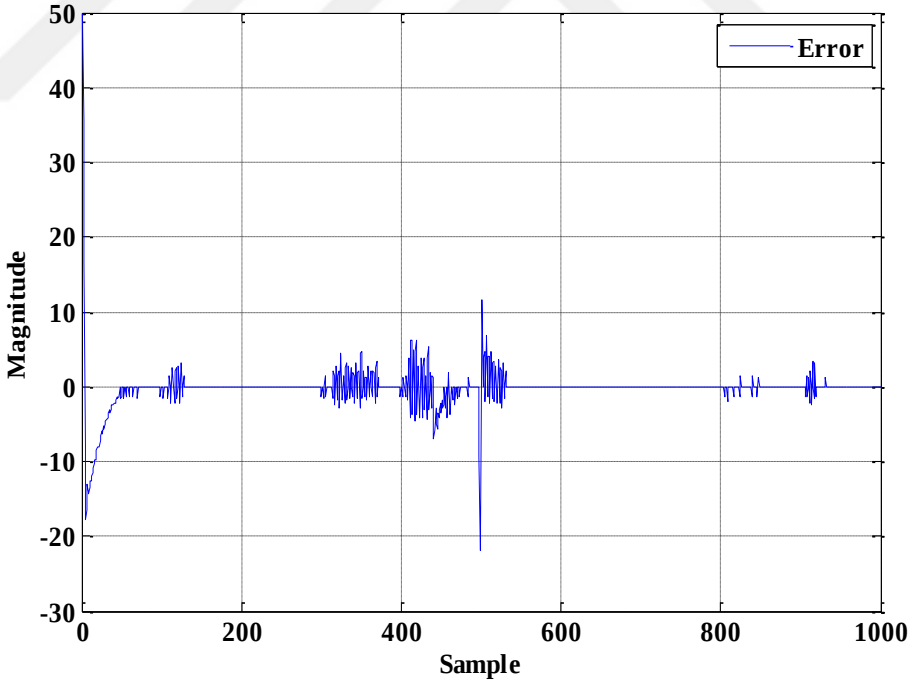


Figure 4.7 The error of model and DC motor output under variable load

The model and system error behavior can be seen in figure 4.7. The figure 4.8 shows the PID parameter change.

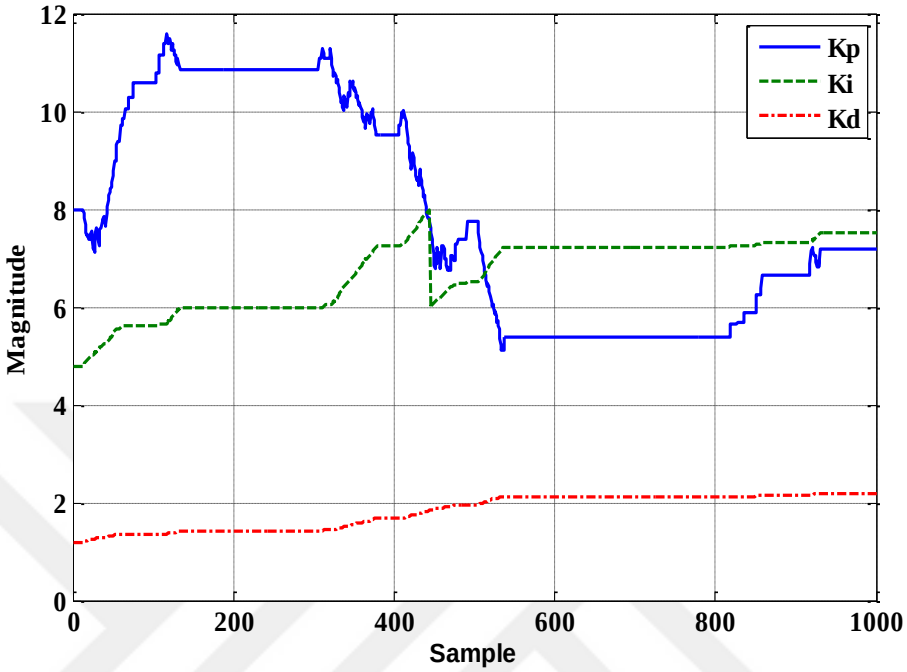


Figure 4.8 The PID parameter change of DC motor setup under variable load

4.2. Experimental Results of Basic Servo System

The basic servo system is tested only with PI controller before the adaptive controller implementation. The PI parameters are set to force the system provide overshoot from reference signal. The reference signal is arranged 20 degree for first 100 samples and then its value is increased to 60 degree for next 100 samples. This procedure is repeated for two cycles. The figure 4.9 shows the system response while the *Kp* and *Ki* values are set to as 15 and 8. In figure 4.9, the basic servo system shows overshoots and damping around the reference signal. After the PI controller experiment, the adaptive controller is implemented to the system. And the initial PI parameters are set to *Kp=15* and *Ki=8*. The *Kp* range of PI controller is defined as 1 to 18 interval, *Ki* range is defined as 1 to 12 interval. The deadband of PI controller, which is defined for error between reference signal and plant output, is set to 1.

The deadband of model and plant output error is set to 1. The final deadband, which is defined for model and reference signal error, is set to 1.

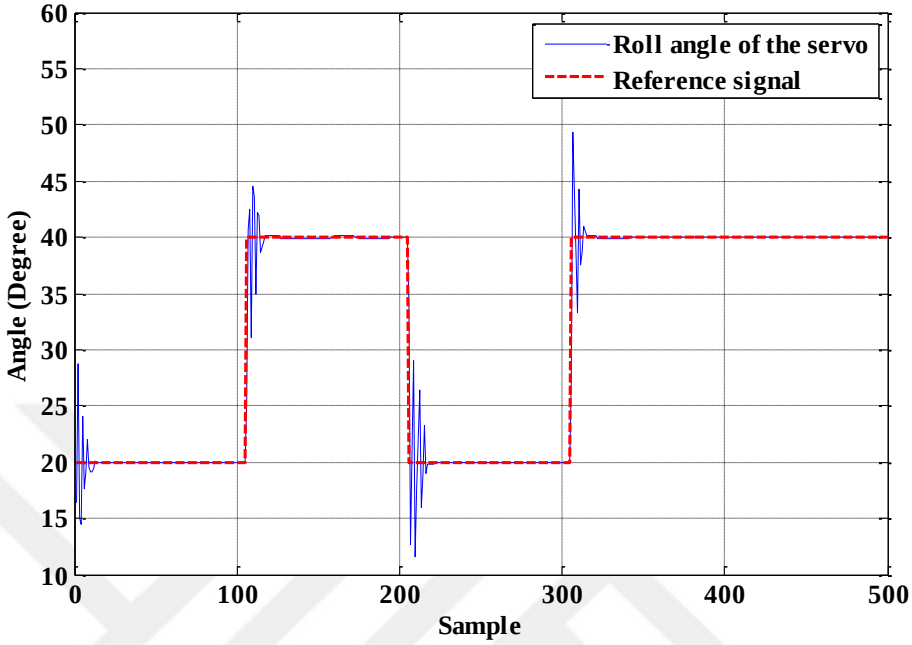


Figure 4.9 The PI response of basic servo system

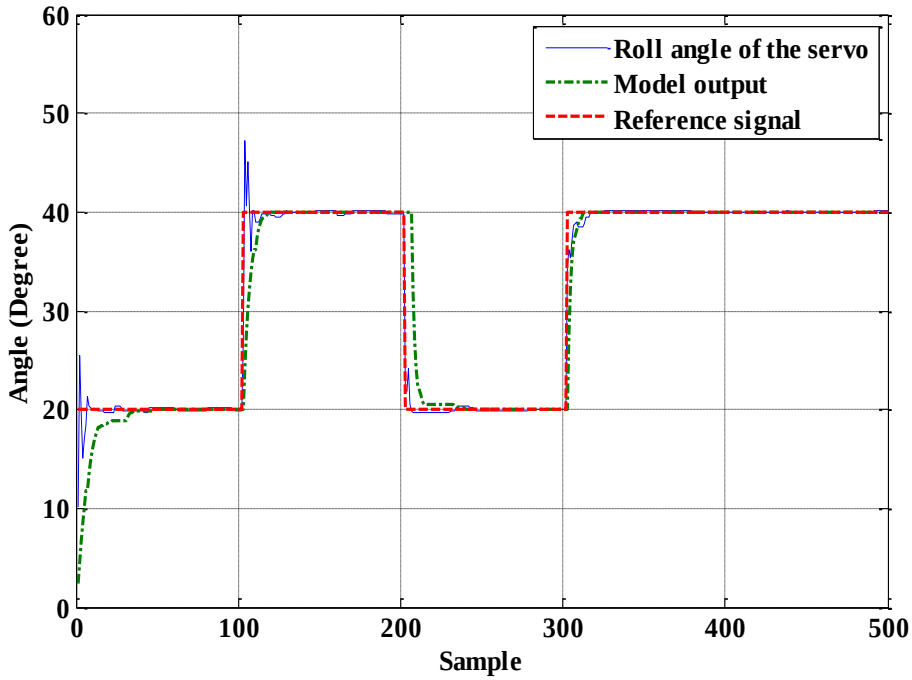


Figure 4.10 The adaptive controller response of basic servo system

The model update frequency is settled as 16 samples. For every 16 samples, output model is revised by identification process. The system response with adaptive controller can be seen in figure 4.10.

According the figure 4.10, the system starts with overshoot, and then system state closes to and tracks the reference signals. The model output is also altered by model update algorithm. Model update algorithm tries to force the system output to close reference signal without overshoot. The rise time of system output reaches its best value in the final cycle.

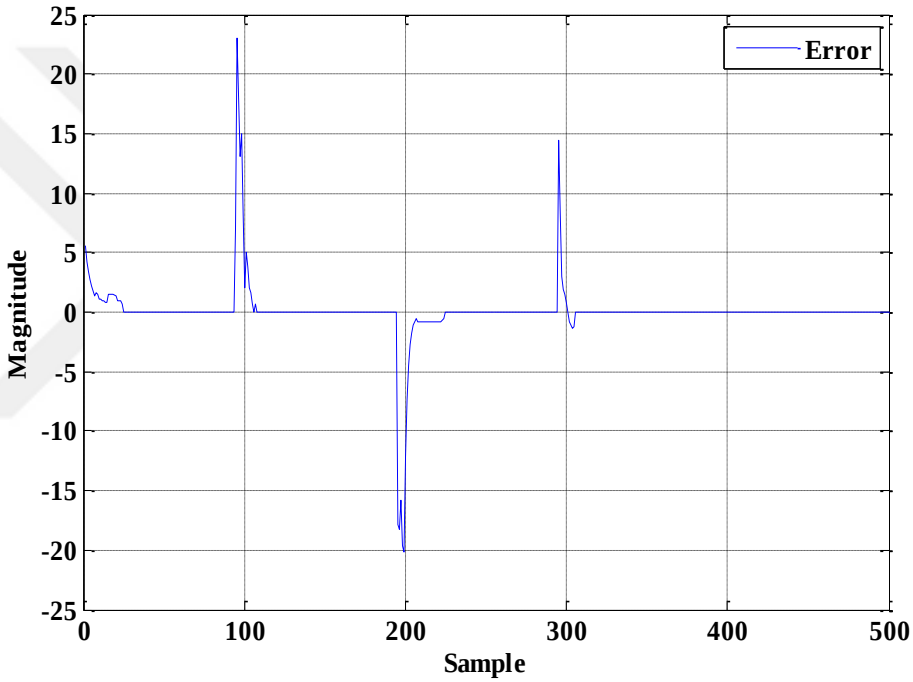


Figure 4.11 The error of model and basic servo output

The figure 4.11 shows model and basic servo output error. The error goes to zero for every reference signal change and the magnitude and interval of error is decreased along the experiment. The adaptive controller tunes PI parameters according to the model. The PI parameter change can be seen in figure 4.12. The integral term is decreased by algorithm and the proportional term is also decreased except for the first cycle. For basic servo system, the algorithm can arrange the system for desired response. It can alter PI parameter to decrease damping and overshoot.

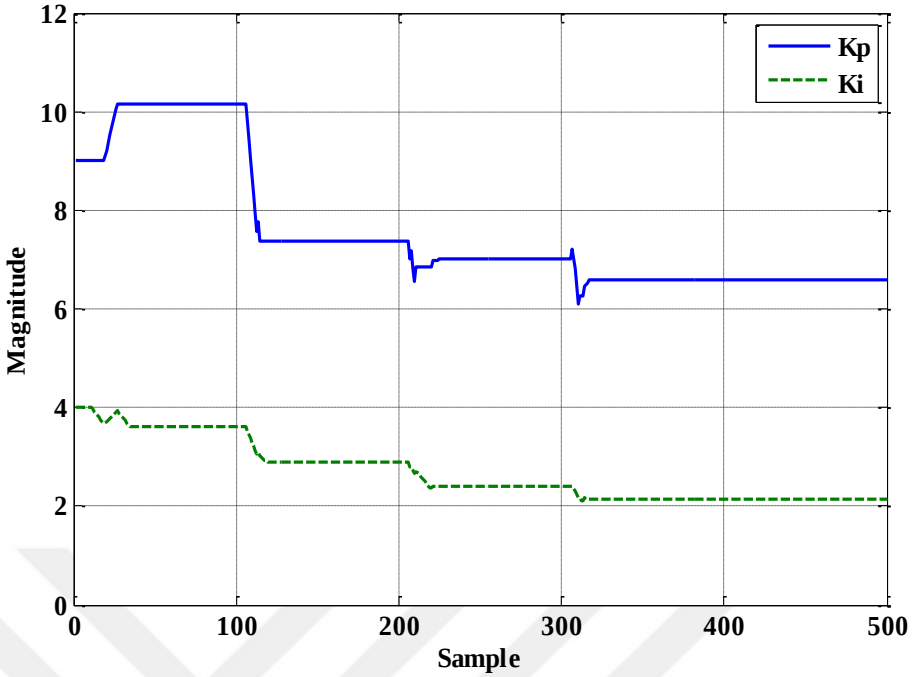


Figure 4.12 The adaptive controller of basic servo system

4.3. Experimental Results of Tilt Platform

The tilt platform is tested for different reference signal. The platform moving range is settled as 20 to 90 degree. The reference signal is applied as 28 degree then reference signal is reduced to 24 degree. For every 100 sample, the reference signal is changed for four cycles. The *Kp* range of PI controller is defined as 1 to 8 interval and *Ki* is defined as 1 to 6 interval. Before the experiment, the plant is controlled by only PI controller and parameters of PI is determined for oscillating response. The plant is oscillated at *Kp=5* and *Ki=4*. The PI controller response of tilt platform can be seen in figure 4.13. After PI controller, the designed adaptive controller is implemented to tilt platform. In initial state, the PI controller parameters are settled as *Kp=5* and *Ki=4* and model update frequency is settled 16 samples. For every 16 samples, output model is revised by identification process. The platform is controlled only PI controller until initial tilt angle reaches 22 degree and then adaptive controller takes over the platform. Three deadbands are defined for implementation. The deadband of PI controller, which is defined for error between reference signal and plant output, is set to 0.4. The deadband of output error is set to 0.8.

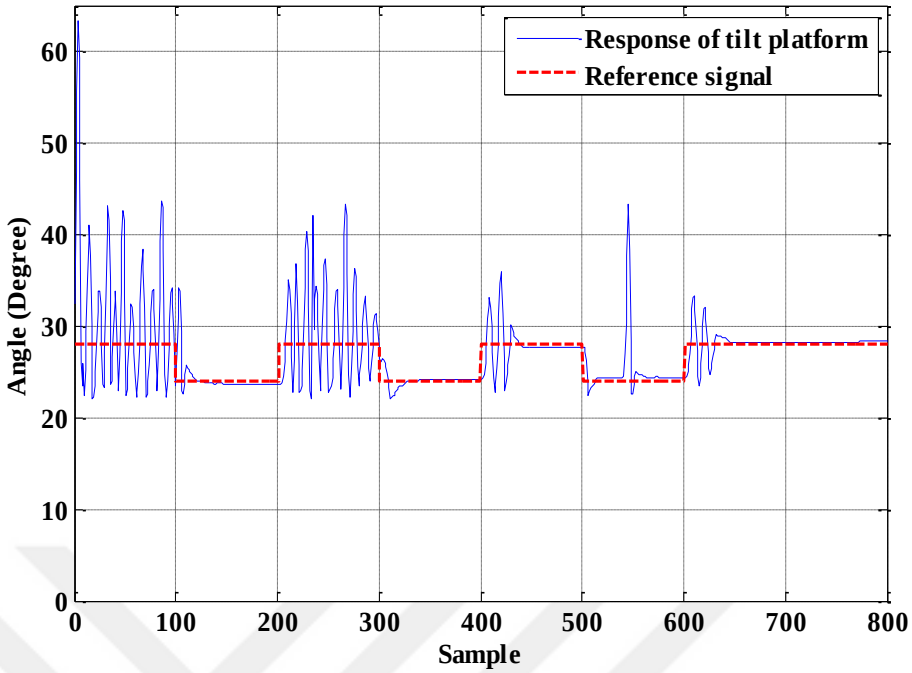


Figure 4.13 The tilt platform response with only PI controller

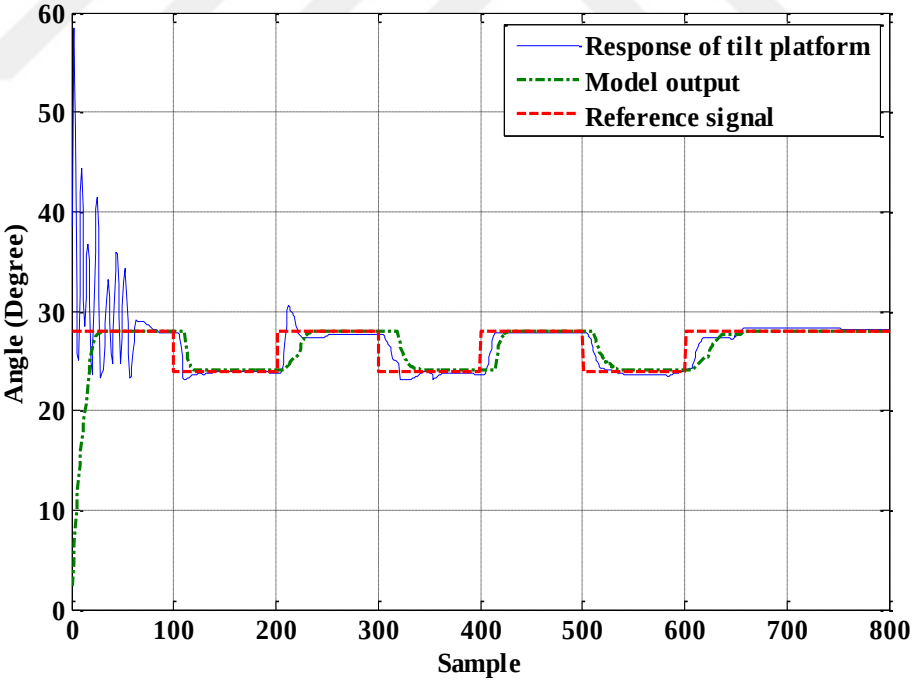


Figure 4.14 The tilt platform response with the designed adaptive controller

The final deadband, which is defined for model and reference signal error, is settled as 0.2.

The figure 4.14 shows the plant and model output with the designed adaptive controller. The plant is initially under oscillating. After the first cycle, the plant response adapts the model output and reference signals.

The model is updated by model update algorithm. The model is altered for every cycle. The rise time of the model output is decreased. The model update algorithm is arranged for slow rise time.

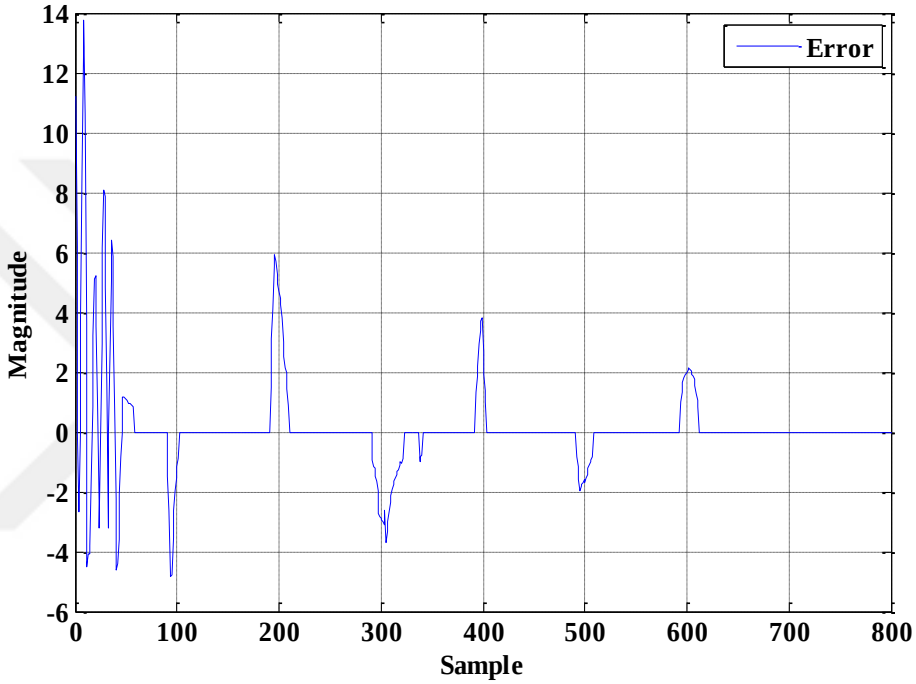


Figure 4.15 The model and tilt platform output error

The model and tilt platform output error is given in the figure 4.15. The error goes to zero for every reference signal change and the magnitude of error is decreased along the experiment.

The figure 4.16 shows the parameter change of the PI controller. The proportional and integral terms are tremendously decreased for initial response. In third cycle, the proportional term starts increasing however integral term goes on decreasing. The aim of the designed adaptive controller is to force the plant to track the model output with tuning the parameter of PI controller. So, for last two cycles, the parameters reach its own proper values.

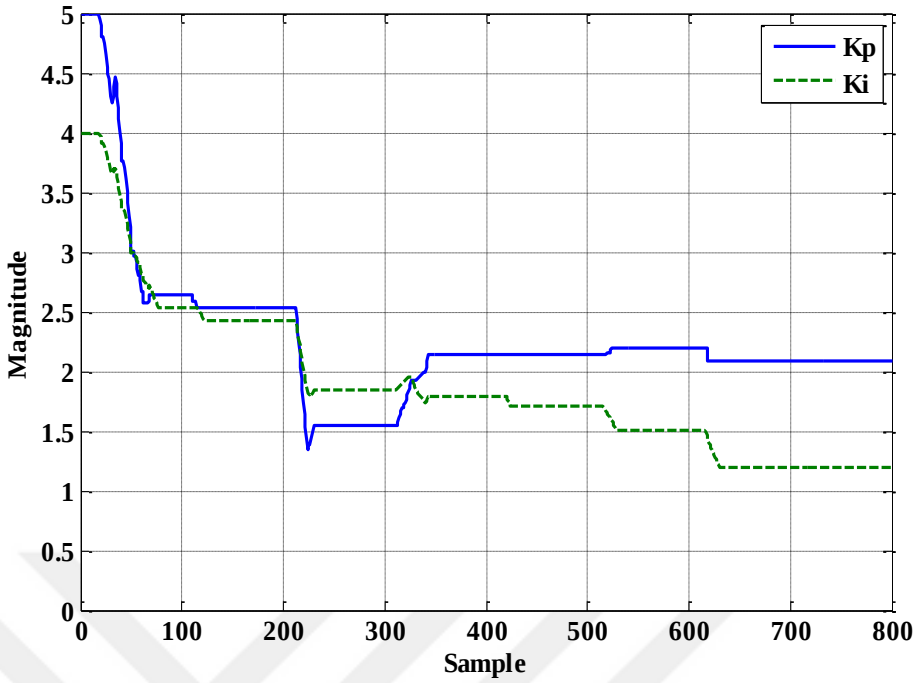


Figure 4.16 The PI parameter change for tilt platform

The tilt platform is unstable. Therefore it is very difficult to tune PI controller for the platform. However, the designed controller can control the platform and tune the PI controller successfully.

CHAPTER 5

CONCLUSION

This study presents a flexible model based adaptive control algorithm which is based on parameter adaptation of a base controller. The aim of the study is to design an adaptive controller which can be applied to any actuator based system. The basic MRAC scheme is used for adaptation. The model of the MRAC is defined for a short term low order model which supported update mechanism. This structure provides generic solution for modeling of designed adaptive controller. The adaptation mechanism is based on low order modeling and identifications of the plant. The identification process is based on least square fitting of plant input and output. Model update mechanism uses the identification process to estimate model parameters for next cycle. For parameter adaptation, the direct method of Lyapunov is used for extracting adaptive rules. In addition, a fuzzy algorithm is used to connect adaptive rules and parameters of PID controller. Although plant model is assumed linear in the controller, model update mechanism provides also practical solution for nonlinear plants.

For implementation, the designed adaptive algorithm is coded in C and C++ environment. The process of the algorithm is shared between microcontroller and computer systems. In addition, in order to make the system run standalone, the adaptation and identification process are implemented to a single board computer. All codes are built with using open source libraries.

For experiment, three different test platforms are built; the DC motor setup, basic servo system and tilt platform. The adaptive controller implemented to the platforms successfully. Moreover, an IMU is designed for basic servo system and tilt platform. The IMU is supported with quaternion based extended Kalman filter and adaptive noise canceler in order to improve measurements.

The DC motor setup includes encoder and gear box and there is servo motor in the DC motor setup. The servo motor is used to change the load on the DC motor. In

experiments the speed of DC motor is controlled with PID controller and designed adaptive controller. The DC motor setup was tested without and with a variable load. The experimental results of DC motor setup show that the designed adaptive controller provides tuning the parameters of PID controller for better response under no load and load conditions.

In the basic servo system, the position control system is tested, and the position is measured by an IMU and controlled by PI controller and designed adaptive controller. The experimental results of basic servo system show that the designed adaptive controller can tune PI controller to decrease system overshoot and provide better positioning.

The tilt platform consists of dual brushless motors, electronic speed controller and an IMU. The tilt angle of the platform is measured by the IMU. The tilt platform is unstable and subject to various noise sources. In order to improve measurement accuracy, extended Kalman filter and adaptive vibration filter are implemented to the IMU. The tilt platform is controlled by only PI controller. The parameter is arranged to oscillate the tilt platform and then the designed adaptive controller takes over the platform. The results of tilt platform experiment show that the designed adaptive controller made the system stable.

In conclusion, the experiments show that the designed adaptive control algorithm can easily be implemented on different plants with minor modifications. In addition, the results of experiments show that the designed control algorithm tracks the desired response successfully. For future work, the designed adaptive controller will be extended for multiple input and multiple output cases. In addition, a frequency domain implementation will be made.

REFERENCES

[1] Astrom K. J., "*Adaptive Feedback Control"*, In proceedings of the IEEE, vol. 75, No.2, 1987.

[2] Sastry S., Bodson M., "*Adaptive Control Stability, Convergence, and Robustness"*, Prentice-Hall Advanced Reference Series (Engineering), 1989-1994.

[3] Astrom K. J., Wittenmark B., "*Adaptive Control"*, second edition *Addison*-Wesley Publishing Company, USA, 1995.

[4] Jitendra R. Raol, Girija Gopalratnam, J. Singh, "*Modeling and Parameter Estimation of Dynamic Systems"*, IET control engineering series 65, 2004.

[5] Ioannou Petros A., Kokotovic Peter V., "*Adaptive Systems with Reduced Models"*, Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1983.

[6] Pawar R. J., Parvat B. J., "*Design and Implementation of MRAC and Modified MRAC technique for Inverted Pendulum"*, International Conference on Pervasive Computing (ICPC), 2015.

[7] Pirabakaran K., Becerra V.M., "*Automatic tuning of PID controller using model reference adaptive control technique"*, the 27th annual conference of IEEE industrial electronic society, 2001.

[8] Trajkov T. N. , Koppe H., Gabbert U., "*Direct model reference adaptive control (MRAC) design and simulation for the vibration suppression of piezoelectric smart structures"*, Communications in Nonlinear Science and Numerical Simulation 13,1896–1909, 2008.

[9] Montanaroa U., Josep M. Olmb, "*Discrete-time integral MRAC with minimal controller synthesis and parameter projection"*, Journal of the Franklin Institute 352, 5415–5436, 2008

[10] Swarnkar P., Jain S., Nema R. K., "*Effect of Adaptation Gain in Model Reference Adaptive Controlled Second Order System"*, ETASR - Engineering, Technology & Applied Science Research Vol. 1, no.3, 70-75, 2003

[11] Nguyen N., "*Hybrid Adaptive Flight Control with Model Inversion Adaptation"*, NASA Ames Research Center United States of America, 2008

[12] Nair A. P., Selvaganesan N., Lalithambika V. R., "*Lyapunov based PD/PID in model reference adaptive control for satellite launch vehicle systems*", Aerospace Science and Technology 51,70–77, 2016.

[13] Matthew Kuipers & Petros Ioannou, "*Multiple Model Adaptive Control With Mixing"*, IEEE Transactions On Automatic Control, Vol. 55, No. 8, August 2010.

[14] Lin Yan, Liu Hsu, Sun Xiuxia, "*A variable structure MRAC with expected transient and steady-state Performance"*, Automatica, 42 805 – 813, 2006

[15] Pi-Cheng Tung, Sun-Run Wang, Fu-Yee Hong, "*Application of MRAC theory for adaptive control of a constrained robot manipulator"*, International Journal of Machine Tools & Manufacture 40 2083–2097, 2000.

[16] Blazic S., Skrjanc I, Matko D., "Globally *stable direct fuzzy model reference adaptive control*, Fuzzy Sets and Systems, 139 3–33, 2003.

[17] Bertolissi E., Birattari M., Bontempi G., Duchteau A., Bersini H., "*Data-Driven techniques for direct adaptive control: the lazy and fuzzy approaches"*, Fuzzy Sets and Systems 128, 3-14, 2002.

[18] Zhu W., Sun Z., "*Data-based fuzzy adaptive control for a flexible-link manipulator"*, Conference on intelligent control and information processing, pp. 223-227, 2012.

[19] Hou Z. , Jin S., "*Model Free Adaptive Control Theory and Applications"*, CRC press, Taylor & Francis Group, LLC, 2014.

[20] Draper C. S., Li Y. T., *"Principles of optimizing control systems and an application to the internal combustion engine"*, ASME Publications, Sep. 1951.

[21] Li Y.T. , Vander Velde W. E., "*Philosophy of Nonlinear Adaptive Systems"*, presented at first IFAC congress Moscow USSR; June 27-July 8, 1960.

[22] Feldbaum A.A, "*Dual control theory, Part I"*, Automation and Remote Control 21 (9): 874–880, April 1961.

[23] Emel'yanov S.V, "*Variable Structure Automatic Control Systems"*, , Nauka, Moscow, 1967.

[24] Emel'yanov, S.V., Utkin, V.I., Taran,V.A., Kostyleva,N.E., Subladze, A.M., Ezerov, V.B., Dubrovskij, E.N., "*Theory of Variable Structure Systems"*, Nauka, Moscow, 1970.

[25] Astrom K. J., Wittenmark B., "*On self tuning regulators"*, Automatica Volume 9, Issue 2, Pages 185-199, March 1973.

[26] R. V. Monopoli, "*Model reference adaptive control with an augmented error"*, IEEE Transactions on Automatic Control, pages 474–484, October 1974.

[27] Kraft G , Campagna David P., "*A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems"*, Presented at the American Control Conference, Pittsburgh, Pennsylvania, June 2 1-23, 1989.

[28] Landau Y. D., "*Adaptive Control: The Model Reference Approach"*, New York, Marcel Dekker, 1979.

[29] Whitaker H.P., "*The MIT adaptive autopilot"*, in P.C. Gregory, Ed., Proc Self Adaptive Flight Control Systems Symp.,WADC Tech. Rep. 59-49 (Wright Air Development Center, Wright-Patterson Air Force Base. OH), 1959.

[30] Oltean S. E.,Dulau Mircea,Duka V. D., "*Model refernce Adaptive Control Design For Slow Processes. A Case Study on Level Process Control"*, 9th. International Conference Interdisciplinary in Engineering, Inter-ENG, 2015.

[31] Parks P. C., "*Lyapunov Redesign of Model Reference Adaptive Control Systems"*, IEEE Transactions on Automatic Control, vol. 11 pp. 362-367, 1966.

[32] Zadeh, L. *A., "Fuzzy sets and systems*.", In: Fox J, editor. System Theory, Brooklyn, NY: Polytechnic Press, 1965: 29–39.

[33] Mamdani, E. H., "*Application of fuzzy logic to approximate reasoning using linguistic synthesis"*, IEEE Transactions on Computers 26(12): 1182–1191, 1977.

[34] *GNU Scientific Library*, Reference Manual Edition 2.1, for GSL Version 2.1 11, pp 422-455, November 2015.

[35] Vectornav, "*Magnetometers* [Online]*",*

http://www.vectornav.com/support/library/Magnetometer [11 April 2016].

[36] C. Konvalin, "*Compensating for Tilt, Hard Iron, and soft Iron Effects* [Online]", http://www.sensorsmag.com/sensors/motion-velocity-isplacement/compensating-tilt-hard-iron-and-soft-iron-effects-6475, [11-April-2015].

[37] Honeywell, "*3-Axis Digital Compass IC HMC5883L"*, technical datasheet of HMC5883L, Honeywell International Inc., February 2013.

[38] Welch G. & Bishop G., "*An introduction to the Kalman Filter"*, University of Nort Carolina at Chapel Hill Department of Computer Science, 2001.

[39] Wikipedia, "*Gimbal lock*[Online]" ,https://en.wikipedia.org/wiki/Gimbal_lock, [11 April 2016].

[40] Julier S. J., Uhlmann J. K., "*A New Extension of the Kalman Filter to Nonlinear Systems"*, SPIE AeroSense Symposium, April 21 - 24, 1997.

[41] Ribeiro M. I., "*Kalman and Extended Kalman Filters: Concept, Derivation and Properties"*, Institute for Systems and Robotics Instituto Superior Tecnico, February 2004.

[42] Kuipers J. B., *"Quaternions and Rotation Sequences"*, Princeton University Press, Princeton, New Jersey, 1998.

[43] Fux S., "*Development of a planar low cost Inertial Measurement Unit for UAVs and MAVs"*, Master Thesis, Swiss Federal Institute of Technology Zurich, Autonomous Systems Lap, 2008.

[44] Douglas, S.C. *"Introduction to Adaptive Filters Digital Signal Processing Handbook Ed"*, Vijay K. Madisetti and Douglas B. Williams Boca Raton: CRC Press LLC, 1999.

RESUME

Ufuk GÜNER

E-mail: gunerufuk@hotmail.com

PERSONAL INFORMATIONS

Date of birth/place : 25.03.1980

Nationality : T.C.

Military Obligation : Done

Marital Status : Married

EDUCATION

Yildirim Beyazit University, Electronics and Communication Engineering, **MS (2014-2016).**

Dumlupinar University, Faculty of Engineering, Electrical Electronics Engineering, **BS (2001-2005).**

CAREER PROFILE

Yildirim Beyazit University, Faculty of Engineering, Electrical Electronics Engineering– Ankara, Turkey, **Research Assistant (2015 - ….).**

Erzurum Technical University, Faculty of Engineering, Electrical Electronics Engineering– Erzurum, Turkey, **Research Assistant (2014 - ….).**

Turkish National Police, Communication Department, Application and Development - Ankara, Turkey, **Electric Electronic Engineer (22.10.2008 – 01.04.2014)**.

Kentkart Ege Electronic R&D - Izmir, Turkey, **Senior Software Developer (24.07.2006 - 01.08.2008).**

PUBLICATIONS

1. Unluturk A., Guner U., Aydogdu O.," *The Real Time Remote Motion Control of Two Wheeled Mobile Balance Robot by Using Video Streaming"*, The 3rd International Conference on Control, Mechatronics and Automation, ICCMA 2015, December 21-22, Barcelona, Spain.
2. Guner U., Canbolat H., Unluturk A., "*Design and Implementation of Adaptive Vibration Filter for MEMS Based Low Cost IMU"*, The 9th International Conference on Electrical and Electronics Engineering, ELECO 2015, November 26-28, Bursa, Turkey.
3. Guner U., Canbolat H., Unluturk A., Aydogdu O., "*MEMS Tipi Sensörler Kullanılarak Genel Amaçlı Atalet Ölçüm Birimi Tasarımı ve Kalibrasyonu"*, TOK 2015, 10-12 September,2015, Denizli, Turkey.
4. Unluturk A, Guner U., Aydogdu O., "*Yeni Bir Pi-V Tipi Denetleyici Tasarimi Ve Denge Robotu Üzerinde Uygulanmasi"*, TOK 2014, 11-13 September 2014, Kocaeli, Turkey
5. Unluturk A., Aydogdu O. Guner U., "*Design and PID Control of Two Wheeled Autonomous Balance Robot*", The Tenth International Conference on Electronics Computer and Computation, ICECCO 2013, on 7-9 November, Ankara, Turkey

2016 ANKARA	**Department of Electronics and Communication Engineering**	**Ufuk GÜNER**