People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfillment of

The Requirements of the Degree of

MASTER

In **Control Engineering**

Option: **Control Engineering**

Title:

Speed Self-Tuning Adaptive Control of DC Motor

Presented By:

- **MESSILEM Med Abdelmouamin**
- **KAIB Med Tahar Habib**

Supervisor:

Dr. OUADI Abderrahmane

Co-Supervisor:

Dr. MESSAI Adnan

Registration Number:...../2019

Abstract

This thesis presents a self tuning pole placement adaptive controller which updates its parameters automatically in order to conserve the desired characteristics of the controlled system.

PMDC Motor is used as plant (process) response of this adaptive controller. Two methods are used for offline system identification (Pattern Search Genetic Algorithm, Non-Linear Least Squares Estimation), data acquisition (NI-PCI 6221-37pin) is used to acquire data that will be used to identify the parameter of the motor (armature resistance, inductance, and inertia), armature current, input voltage and rotational speed of the PMDC Motor are the data acquired and used in system identification. DAQ tool box in MTLAB/SIMULINK is used to perform the analysis on the data acquired based on Non-Linear Least Squares Estimation and Pattern Search Genetic Algorithm. PID controller is designed based on the model estimated, and is implemented and tested using the motor in order to validate the obtained model. The Recursive Least Squares Estimator block in SIMULINK is used to perform the online estimation. The controller is implemented and simulated in SIMULINK (SIMULINK Desktop Real-Time), the behavior of the PMDC motor is close to the desired response.

Output response of the PMDC Motor can be changed due to a perturbation, environment changes or parameters changes. STC is able to track the system (plant or process) for maintaining the desired behavior.

Key Words:

System Identification, PID Controller, Online Estimation, Adaptive Controller, Self Tuning Adaptive Controller, Least Squares Method, Pattern Search Genetic Algorithm, Permanent Magnet DC Motor.

Dedication

This is wholeheartedly dedicated to our beloved parents, who has been our source of inspiration and gave us strength, who continually provide their moral, spiritual, emotional, and financial support.

To our brothers, sisters, friends and classmates who shared their words and advice and encouragement to finish this study.

Acknowledgement

We would like to thank and express our very great appreciation to our thesis supervisor Dr.OUADI Abdurrahman and co-supervisor Dr.MESSAI Adnan for their support, encouragement and professional assistance.

We would like to offer our special thanks to ZAHRA Bilal and KOUDIAH Nouredine whose advice that has been great help in completing this thesis, we are particularly grateful for their assistance.

Special thank for CRNB research center and IGEE university staffs member.

We would also like to extend our gratitude to our families for their support they have given us.

Contents

List of Tables

List of Figures

List of Abbreviations:

DAQ	Data Acquisition
PID	Proportional Integral Derivative
STC	Self Tuning Controller
DC	Direct Current
EMF	Electromotive Force
KVL	Kirchhoff's Voltage law
I/O	Input Output
PI	Proportional Integral
PD	Proportional Derivative
MRAC	Model Reference Adaptive Control
MDPP	Minimum Degree Pole Placement
PWM	Pulse Width Modulation
RPM	Revolution Per Minute
RPS	Revolution Per Second
Ics	Integrated Circuits
CPR	Cycle Per Revolution
RLSE	Recursive Least Squares Estimation
RSLM	Recursive Least Squares Method
PCI	Peripheral Component Interconnect
NI	National Instrument
LPWM	Left Pulse Width Modulation
RPWM	Right Pulse Width Modulation
R_EN	Right Enable
L_EN	Left Enable
GRN	Ground
PMDC Motor	Permanent Magnet Direct Curent Motor

List of Nomenclatures

Vs	Source Voltage
I_a	Armature Current
W_m	Rotational Velocity
K_t	Torque Constant
K_e	Back EMF Constant
R_a	Armature Resistance
L_a	Armature Conductance
T	Torque
E	Back EMF
T_f	Friction Torque
J	Moment of Inertia of the rotor
B	Viscous (damping) friction
ʎ	Forgetting Factor
φ	Concatenated I/O matrix
ζ	Damping Ratio
w_n	Natural Frequency

Introduction

Once DC motor technology got under way, it matured rapidly and was conveniently available to meet such heavy demands as those imposed by the automobile industry, garment manufacturing, and when deployed 'back- wards', for generating power for street lighting and general electrification purposes. Today, DC motors once again command widespread interest as exceptionally useful devices; this dramatic change has been brought about by the advent of solid-state rectifiers, new and exotic magnetic materials, electronic control techniques, electric vehicles, computers [1]. In addition they have high efficiency and high starting torque versus falling speed characteristics which helps high starting torque and helps to prevent sudden load rise [2]. Depending on the application the motor need to operate in particular stable region. But due to environmental changes, the region of operation will deviate and the behavior of system will be unstable. Therefore we need a controller to make the system stable, this controller is designed upon certain system parameter, this controller is known as classical controller. Due to plant uncertainties system parameters will change, the controller needs to be adaptable to these changes, this gives rise to modern controller which is known as adaptive controller.

Chapter 1: DC Motor

1.1 DC Motor Description:

Every system is characterized by its model, so first it is needed to give mathematical description of the motor. The DC motors are composed of:

- Frame
- Shaft
- Bearing
- Main field winding (stator)
- Armature (rotor)
- Commutator
- Brushes

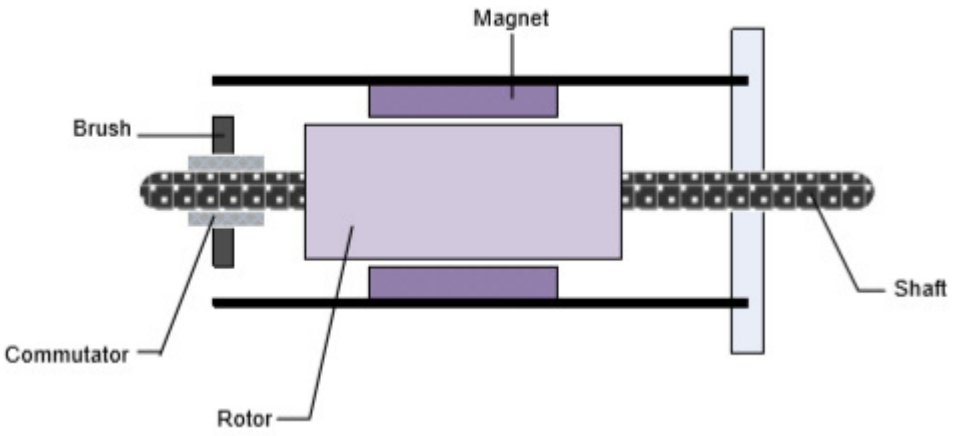

Fig 1.1: Sectional illustration of DC motor.

The stator provide constant magnetic field, the armature which is the rotating part is a simple coil, and it is connected to a DC power source through a pair of commutator rings. When the current flows through the coil, an electromagnetic force is induced on it according to Lorenz law, so the coil will start to rotate.

We notice that as the coil rotate the commutator rings connect to the power source with opposite polarity, as a result on one side of coil the electricity will always flow away and in the other side electricity will always flow towards. This ensures that the torque action is also in the same direction throughout the motion, so the coil will not change direction of rotation, it is known as commutation.

1.2 DC Motor Model

1.2.1 Mathematical Model Of Typical PMDC Motor

In order to give a model for motor, we need to give its equivalent circuit and analyze it. **Fig 1.2** and **Fig 1.3** show a typical Permanent Magnet DC Motor equivalent circuit:

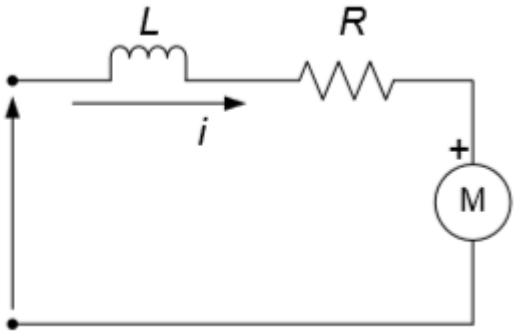

Fig 1.2: A Typical PMDC motor equivalent electrical circuit.

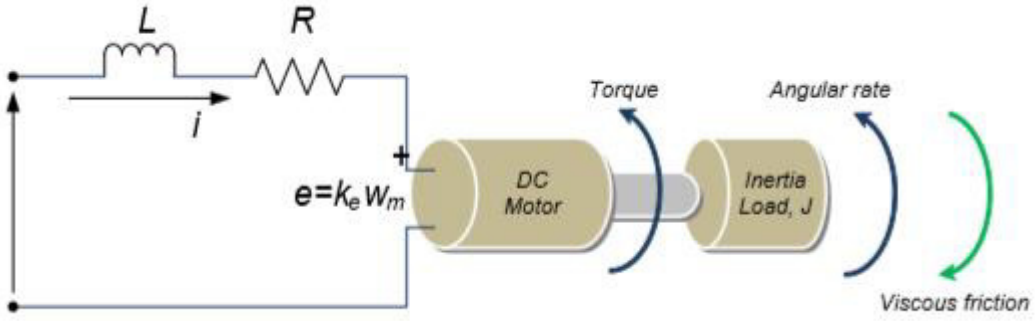

Fig 1.3: A Typical PMDC motor equivalent electrical circuit.

The armature is characterized by its resistance R and inductance L, the motor torque is related to the armature current by a constantK_t**:**

$$T = K_t * I_a \tag{1.1}$$

The back EMF E is related to the rotational velocity by constantK_e:

$$E = K_e * W_m \tag{1.2}$$

Analyzing the equivalent electrical circuit using Kirchhoff's law KVL, we obtain:

$$\mathrm{V_s} = R_a * I_a + L_a * \frac{di}{dt} + K_e * W_m \tag{1.3}$$

From the electromechanical system arrangement and using Newton's second law of motion (The product of inertia load, ***J*** and the rate of angular velocity, $\boldsymbol{W_m}$ is equal to the sum of all the torques) we obtain the following equation:

$$j\frac{\mathrm{dW}}{\mathrm{dt}} = \sum \tau \tag{1.4}$$

$$\mathrm{K_t} * \mathrm{I_a} = \mathrm{J}\frac{\mathrm{dW}}{\mathrm{dt}} + \mathrm{K_f} * \mathrm{W_m} + \mathrm{T_f} \tag{1.5}$$

Laplace transform of the equations (1.3) and (1.5) respectively are:

$$\mathrm{V_s} = \mathrm{R_a} * \mathrm{I_a} + \mathrm{L_a} * \mathrm{I_a} * \mathrm{s} + \mathrm{K_e} * \mathrm{W_m} \tag{1.6}$$

$$\mathrm{K_t} * \mathrm{I_a} = \mathrm{J} * \mathrm{W_m} * \mathrm{s} + \mathrm{B} * \mathrm{W_m} + \mathrm{T_f} \tag{1.7}$$

From equation (1.7):

$$\mathrm{W_m} = \frac{\mathrm{K_t} * \mathrm{I_a} - \mathrm{T_f}}{\mathrm{J} * \mathrm{s} + \mathrm{B}} \tag{1.8}$$

Substituting (1.8) into equation (1.6) and neglecting the friction torque to obtain the transfer function between input voltage Vs and the armature current $\mathrm{I_a}$:

$$\mathrm{V_s} = \mathrm{R_a} * \mathrm{I_a} + \mathrm{L_a} * \mathrm{I_a} * \mathrm{s} + \mathrm{K_e} * \frac{\mathrm{K_t} * \mathrm{I_a} - \mathrm{T_f}}{\mathrm{J} * \mathrm{s} + \mathrm{B}} \tag{1.9}$$

$$\frac{I_a}{V_s} = \frac{\frac{1}{L_a} * (s + \frac{B}{J})}{s^2 + \left(\frac{Ra}{L_a} + \frac{B}{J}\right) * s + (\frac{Ra * B}{La * J} + \frac{Ke * K_t}{La * J})} \tag{1.10}$$

$$\mathrm{I_a} = \frac{\mathrm{J} * \mathrm{W_m} * \mathrm{s} + \mathrm{B} * \mathrm{W_m} + \mathrm{T_f}}{\mathrm{K_t}} \tag{1.11}$$

Substituting $\mathrm{I_a}$ into equation (1.6):

$$\mathrm{V_s} = \mathrm{R_a} * \frac{\mathrm{J} * \mathrm{W_m} * \mathrm{s} + \mathrm{B} * \mathrm{W_m} + \mathrm{T_f}}{\mathrm{K_t}} + \mathrm{L_a} * \frac{\mathrm{J} * \mathrm{W_m} * \mathrm{s} + \mathrm{B} * \mathrm{W_m} + \mathrm{T_f}}{\mathrm{K_t}} * \mathrm{s} + \mathrm{K_e} * \frac{\mathrm{K_t} * \frac{\mathrm{J} * \mathrm{W_m} * \mathrm{s} + \mathrm{B} * \mathrm{W_m} + \mathrm{T_f}}{\mathrm{K_t}} - \mathrm{T_f}}{\mathrm{J} * \mathrm{s} + \mathrm{B}} \tag{1.12}$$

At no friction we can derive the transfer function between W(s) and V(s):

$$\frac{W_m}{V_s} = \frac{\frac{K_t}{La * J}}{s^2 + \left(\frac{Ra}{L_a} + \frac{B}{J}\right) * s + (\frac{Ra * B}{La * J} + \frac{Ke * K_t}{La * J})} \tag{1.13}$$

Chapter 2: System Identification

2.1 Introduction

System Identification is the process of identifying a mathematical model for a partially or completely unknown system based on collected data from this system [3].

The estimator takes the measured variables "e.g. Input of the system and its output" as its input and produces the estimated model parameters.

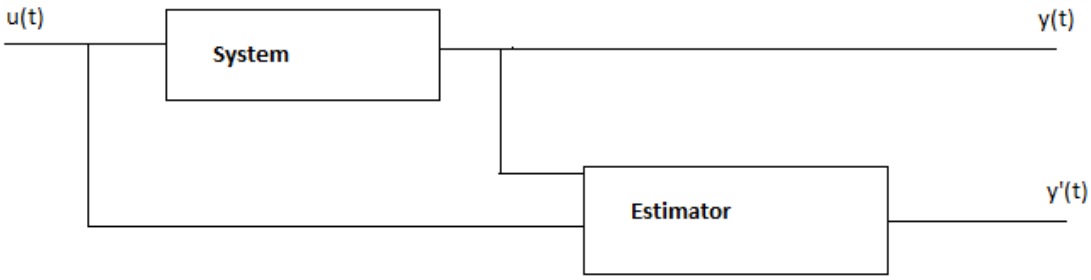

Fig 2.1: block diagram of estimation process.

The purposes of system identification are:

- to predict system behaviour.
- To explain the ineractions and relationships between the inputs and outputs.
- To design a controller or simulation of the system.

2.2 Basic Steps Of Identification

- **Collect Information About The System:**

In order to identify a system, a set of required information must be collected.

This part is the most critical in system identification because everything is build upon it.

- **Select Model Structure To Represent The System:**

There are several mathematical models, only one model can be chosen based on its suitability for system's application and less complicated. Here are some of these models:

There exist parametric models and nonparametric models for speed control application. It is better to work with parametric models which are described using a limited number of characteristic quantities called parameter of the system, like I/O transfer function of the system.

As it is known in real life systems are nonlinear, most of these systems can be approximated by a linear model assuming that their operation region can be linearized. A linear model is sufficient to implement controller for the system within the linearity region.

- **Matching And Validation Of The Results:**

The chosen model structure should match as well as possible to the available information about the system. In most cases, this is done by minimizing a criterion that measures goodness of the fit.

The model obtained must be tested and check if it matches the available data in time domain. In practice, the best model is the one that is simpler and within a specified error bound not the one with small error but very complicated.

2.3 Least Squares Method

It is a widely used technique; Least Squares Estimation is about estimating parameters of the parametric model by minimizing the squared errors between the observed data and their expected values.

2.3.1 Offline Parameter Estimation

The error e (k) is given by:

$$e(k) = y(k) - y'(k) \tag{2.1}$$

Where y (k) is the output obtained from the unknown (plant) system and y' (k) is its estimated output.

The cost function to be minimized J:

$$J = \frac{1}{N} * \sum_{k=1}^{N} (e(k))^2 \tag{2.2}$$

N is the number of samples taken.

The estimated output can be defined as:

$$y'(k)=Y' = \varphi B \tag{2.3}$$

φ is the concatenated input/output matrix of the order (N x 2n), n is the order of the system and B is vector of the dimension (2n x 1) of the estimated coefficients.

The model G(z) is a discrete time transfer function, and B are defined as:

$$B^T = [a_1\ a_2\ a_3\ \dots\ a_n\ b_1\ b_2,\ \dots\ b_n].$$

$$G(z)=\frac{b_1*z^{-1}+b_2*z^{-2}+\cdots+b_n*z^{-n}}{1+a_1*z^{-1}+a_2*z^{-2}+\cdots+a_n*z^{-n}} \tag{2.4}$$

Then the cost function, (2.2), becomes

$$J = \frac{1}{N}\left(Y - Y'\right)^T (Y - Y') \tag{2.5}$$

$$J = \frac{1}{N}(Y - \varphi B)^T (Y - \varphi B) \tag{2.6}$$

This implies that

$$J = \frac{1}{N}(Y^T Y - Y^T \varphi B - B^T \varphi^T Y + B^T \varphi^T \varphi B) \tag{2.7}$$

In order to minimize this cost function the first derivative with respect to B must be equal to zero

This implies

$$\frac{dJ}{dB} = 0 - \varphi^T Y - \varphi^T Y + \varphi^T \varphi B + \varphi^T \varphi B = 0 \tag{2.8}$$

After some algebraic manipulations, the formula of the estimated coefficients is obtained as:[4]

$$B = (\varphi^T \varphi\ )^{-1} \varphi^T Y \tag{2.9}$$

2.3.2 Online Parameter Estimation

The procedure is very simple we start by computing the estimated parameters for N input output data where N is greater or equal to 2n (n is the order of the model).

$$Y^T = [Y_N, y_{N+1}] \tag{2.10}$$

Y_N is the available data vector and y_{N+1} is the new obtained data

$$\varphi^T = [\varphi_N x_{N+1}^T] \tag{2.11}$$

Where x_{N+1}^T= [-y(N),… -y(N+1-n), u(N), … u(N+1-n)]

The new estimated values are given by [4]:

$$B_{N+1} = ([\varphi_N^T x_{N+1}][\varphi_N x_{N+1}^T]^T)^{-1}[\varphi_N^T x_{N+1}][Y_N\ y_{N+1}]^T \quad (2.12)$$

$$B_{N+1} = (\varphi_N^T \varphi_N + x_{N+1} x_{N+1}^T)^{-1}(\varphi_N^T Y_N + y_{n+1} x_{n+1}) \quad (2.13)$$

$$P_N^{-1} = \varphi_N^T \varphi_N \quad (2.14)$$

In the last equation, (2.14), the obtained matrix is called *Correlation Matrix.*

Substituting (2.14) in the last estimated values formula (2.13) we get

$$B_{N+1} = P_N \left(I - \frac{x_{N+1} x_{N+1}^T P_N}{1 + x_{N+1}^T P_N x_{N+1}} \right) (\varphi_N^T Y_N + y_{N+1} x_{N+1}) \quad (2.15)$$

This factor, λ, is the forgetting factor and it is very close to 1 for Linear Time Invariant systems, its effect is to exponentially weight out past data.

$$K_{N+1} = \frac{P_N x_{N+1}}{\lambda + x_{N+1}^T P_N x_{N+1}} \quad (2.16)$$

And

$$P_{N+1} = \frac{P_N}{\lambda} \left(\frac{I - x_{N+1} x_{N+1}^T P_N}{\lambda + x_{N+1}^T P_N x_{N+1}} \right) \quad (2.17)$$

The new estimated values can be defined as

$$B_{N+1} = B_N + K_{N+1}[y_{N+1} - x_{N+1}^T B_N] \quad (2.18)$$

New Estimation = Old Estimation +Gain [prediction error of model]

2.4 Genetic Algorithm

A genetic algorithm differs from other search techniques by the use of concepts taken from natural genetics and evolution theory. First, the algorithm works with a population of strings, searching many peaks in parallel. By employing genetic operators it exchanges information between the peaks, hence reducing the possibility of ending at a local minimum and missing the global minimum.[5]

The following outline summarizes how the genetic algorithm works:[6]

1. The algorithm begins by creating a random initial population.
2. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:

a. Scores each member of the current population by computing its fitness value. These values are called the raw fitness scores.
b. Scales the raw fitness scores to convert them into a more usable range of values. These scaled values are called expectation values.
c. Selects members, called parents, based on their expectation.
d. Some of the individuals in the current population that have lower fitness are chosen as *elite*. These elite individuals are passed to the next population.
e. Produces children from the parents. Children are produced either by making random changes to a single parent—*mutation*—or by combining the vector entries of a pair of parents—*crossover*.
f. Replaces the current population with the children to form the next generation.

3. The algorithm stops when one of the stopping criteria is met

Chapter 3: Controller Design

3.1 Introduction

Control design is used almost in every field of science because it helps us to make the work easier and more efficient. The main objective of Control Design is to make any system behaves as it is required. For example if an aircraft system is considered. The system input is a desired altitude or reference and the output is a real measured altitude. The system controller must be designed so that any output changes must be compensated so that the output tracks the desired input reference. Hence for a good controller, the error between the desired altitude and the real altitude should be as close as possible to zero, in other critical application it must be zero.

3.2 Types Of Controller

We can distinguish two fundamental types of controllers which are:

- Open Loop Controller.
- Closed Loop Controller.

3.2.1 Open Loop Controller

This type of controllers is based only on actuating device to control the process, **Fig 3.1**, which means it does not take into consideration the error between the desired value (set point) and real value from the output of the system to control the process.

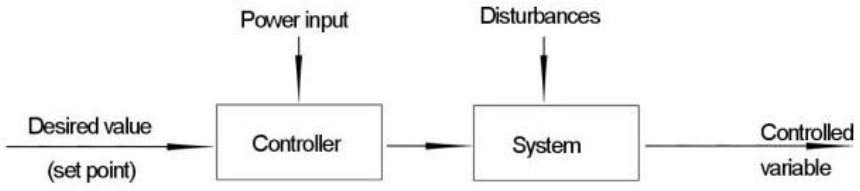

Fig 3.1: open loop control system.

3.2.2 Closed Loop Controller

Closed loop controller utilizes the error between the set point and measured value (real value) from the feedback to command the system in order to follow the desired value, as shown in **Fig 3.2**.

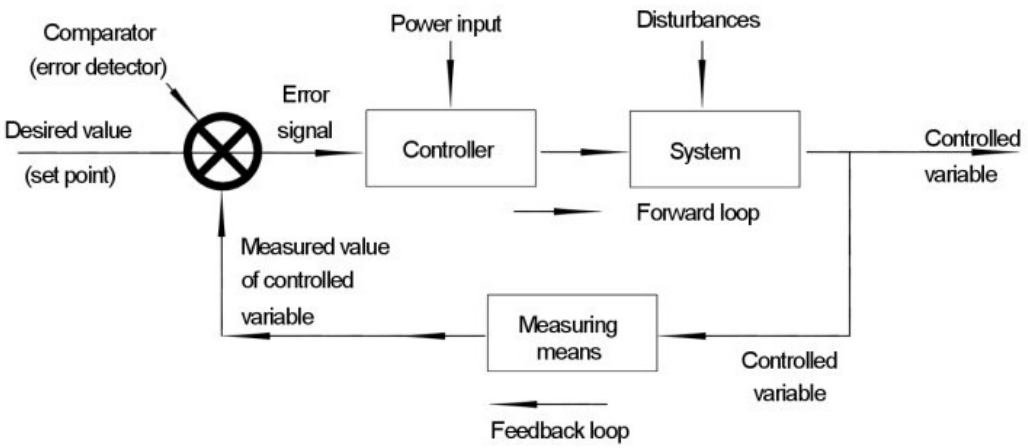

Fig 3.2: closed loop controller.

The following table, **Table 3.1**, demonstrates the main difference between these controllers

Table 3.1: open loop/closed loop comparison.

Open Loop Control System	**Closed Loop Control System**
Output variables do not affect the input variables.	Output variables do affect the input variables to maintain the desired behavior.
System will follow the desired reference commands if no unpredictable effect occurs.	Requires measurement of controlled variables and/or other variables.
Can be compensated for disturbances that are taken into account.	Requires control errors computed as the difference between the controlled variable and the reference command.
Does not change system stability.	Computes control input based on the control errors such that the control error is minimized.
///	Able to reject the effect of disturbances.
///	Can make the system unstable.

3.3 PID Controller

The PID controller is the most known and used controller in Control Design. It consists of proportional part, Integrator part and derivative part, as shown in **Fig 3.3**; in some applications we can use only two of them (e.g. PI controller, PD controller) it depends on the system to be controlled and the desired behavior.

The proportional part is defined as constant gain, Kp, the larger this constant gain is the faster is the system, but if we get too greedy and try to increase it more we may end up with unstable system.

The integral part consists of integrator and constant gain, Ki, this part of the controller has its effect on the steady state; the integrator is responsible of eliminating the steady state error of the system, but the smaller the steady state error is the larger the overshoot becomes.

The derivative part is consisted of derivation and constant gain, Kd, it has effect on the transient state, it helps to minimize the overshoot in the system.

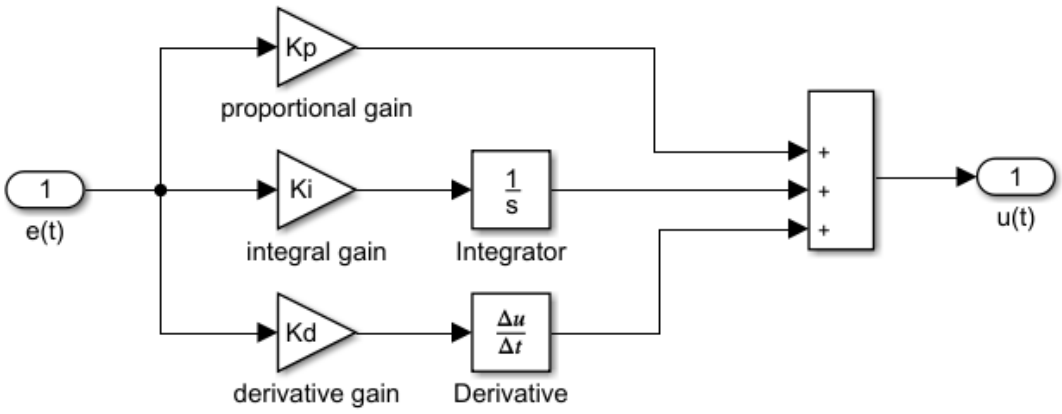

Fig 3.3: conventional PID circuit.

PIDs got this reputation of being the most known and used controllers because they are adjustable on-site, many different types of tuning rules have been proposed in literature. Also, automatic tuning methods have been developed and some of these PIDs may possess on-line automatic tuning capabilities [7].

The general and basic formula of PID controller is:

$$H(s) = Kp + Ki * \frac{1}{s} + Kd * s \quad (3.1)$$

As it is known in real applications signals always have noises which can be a problem if we use derivative block in PID controllers this, it will increase those noises which can make our system unstable, Furthermore since PID controllers are implemented on micro-controller devices this derivative will consume more resources of the micro-controller than integrator and constant gain.

As solution, the derivative block is replaced by a low-pass filter with order, N, and integrator, as shown in **Fig 3.4**, to reduce the effect of signals noises and reduce resources consumption.

The formula becomes as the following:

$$H(s) = Kp + Ki * \frac{1}{s} + Kd * \frac{N}{1+\frac{N}{s}} \quad (3.2)\ [6]$$

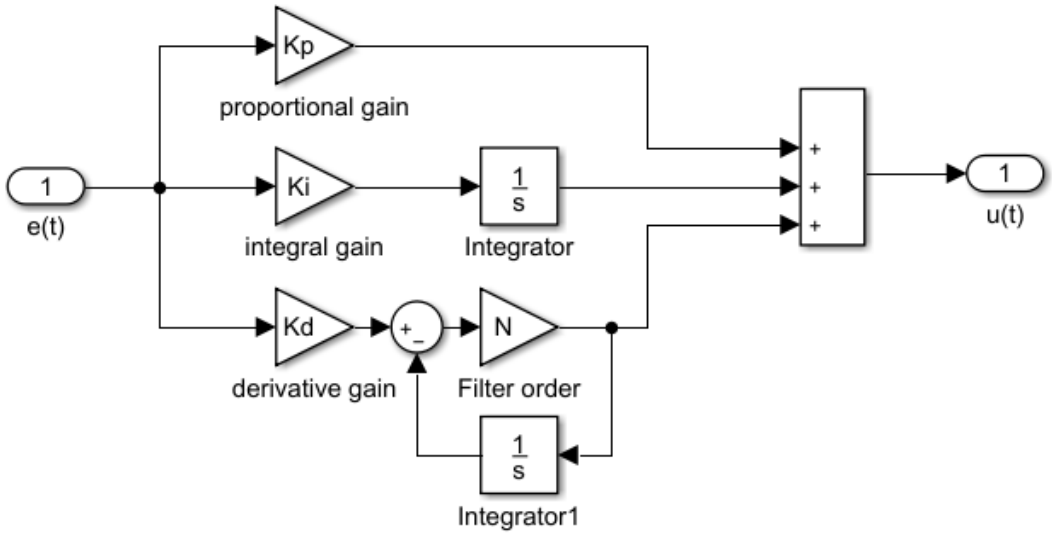

Fig 3.4: circuit of PID controller

The PID controller is compensated for a specific system, which means that if there is a change in system's parameters due to temperature changes, frictions inside the system, dust from outside the system, or one of the components is not working as it must be …etc, then we must change the controller's parameters.

As solution to this problem, adaptive controllers were introduced where the controller's parameters are automatically updated when the system's parameters change.

3.4 Adaptive Controller

The unknown and immeasurable variations of the process parameters degrade the performances of the control systems. Similarly to the disturbances acting upon the controlled variables, one can consider that the variations of the process parameters are caused by disturbances acting upon the parameters (called parameter disturbances). These parameter disturbances will affect the performance of the control systems. [8]

For any possible values of plant model parameters there is a controller with a fixed structure and complexity such that the specified performances can be achieved with appropriate values of the controller parameters.

3.4.1 Definition

An adaptive controller is a controller that can modify its behavior in response to the variations in the dynamics of the process and the character of the disturbances. The controller parameters may be adjusted by some mechanism. There is a wide variety of mechanisms for adjustment, among which system identification of the control object, and some performance measure of the system such as output variance. The aim of an adaptive controller is to automatically provide a competitive controller in situations where the dynamics of the control object may be varying [9]

3.4.2 Adaptive Schemes

Three types of adaptive system can be described: gain scheduling, model-reference adaptive control, self-tuning adaptive control.

a. Gain Scheduling:

In many cases it is possible to find measurable variable that correlate well with changes in process dynamics, these variables can then be used to change the controller parameters, as shown in **Fig 3.5**.

This approach is called gain scheduling because the scheme was originally used to measure the gain and then change, that is schedule, the controller to compensate for changes in the process gain [10]

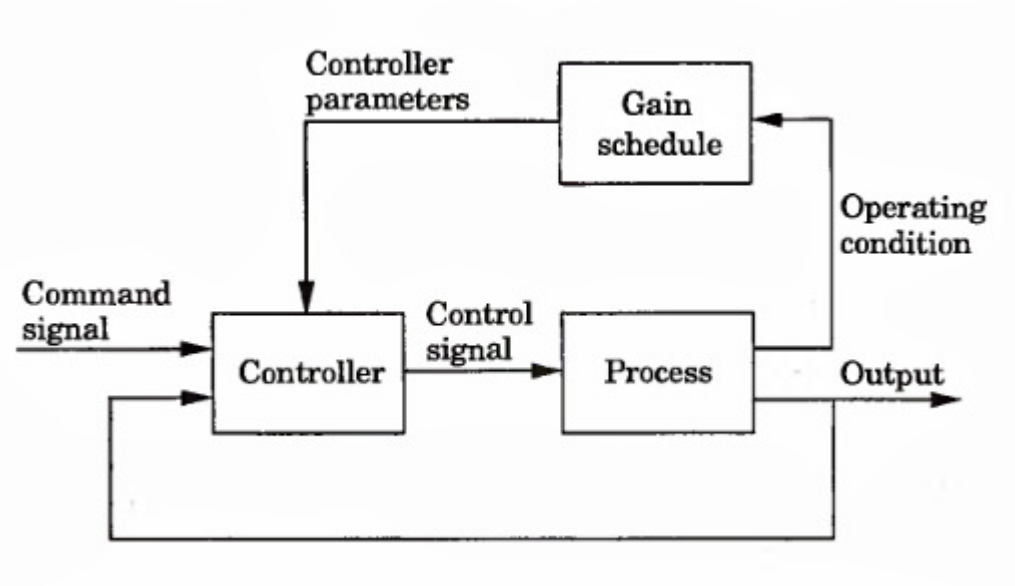

Fig 3.5: Block diagram of system with gain scheduling.

b. Model-Reference Adaptive Control:

The model reference adaptive system was originally proposed to solve problem in which the performance specification are given in terms of reference model. [10]

A reference model is used to specify a desired response of an adaptive control system to a command input, as shown in **Fig 3.6**. It is essentially a command shaping filter to achieve a desired command following. Since adaptive control is formulated as a command following or tracking control, the adaptation is operated on the tracking error between the reference model and the system output. A reference model must be designed properly for an adaptive control system to be able to follow. [11]

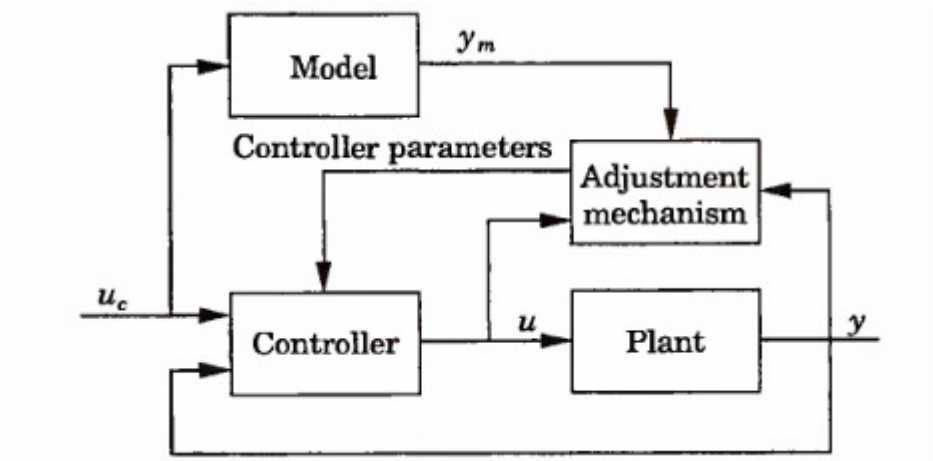

Fig 3.6: Block diagram of a model reference adaptive system.

Model Reference Adaptive Control (MRAC) is used for making a closed loop controller which adjusts the variables of the system dynamically by comparing the output of the plant with a standard reference response [11]

Further, MRAC is classified into two types:

1) Direct Control Type:

In this Control the system adjusts itself to the error signal which is described as the difference between the plant and the reference response. The controller parameter of the controller is updated in real time by adaptive law.

2) Indirect Control Type:

In Indirect control the system adjusts itself by comparing the plant output to online standard reference. The value of parameter is obtained by solving linear algebraic expressions that correlates with online model of the plant for each time (t) [8].

c. Self-Tuning Controller:

The self-tuning controller has two loops including an inner and an outer loop. The inner loop consists of the conventional controller, but with alternating parameters, and the outer loop consists of the identifier and design box which adjust these controller parameters [12], as shown in **Fig3.7**. The controller receives both the input signal and the output signal of the plant [12].

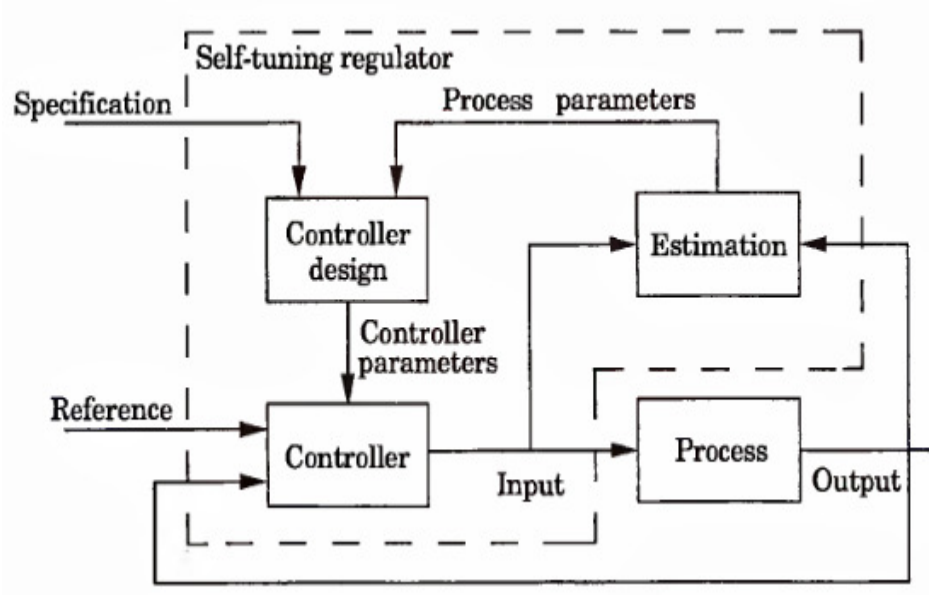

Fig3.7: Block diagram of Self-tuning controller

A digital self-tuning controller is a controller that during each sample interval performs three major steps [13].

- Estimates the parameters of the discrete plant model.
- Calculates the controller parameters using the estimated plant model parameters.
- Calculates and implements the new control signal.

Also self-tuning control has two types, direct and indirect method, as shown in **Fig 3.8 & 3.9**. In direct method the adjustment mechanism tells directly how the controller should be updated. In indirect method the controller parameter are obtained from the solution of design problem using the estimated parameter [10].

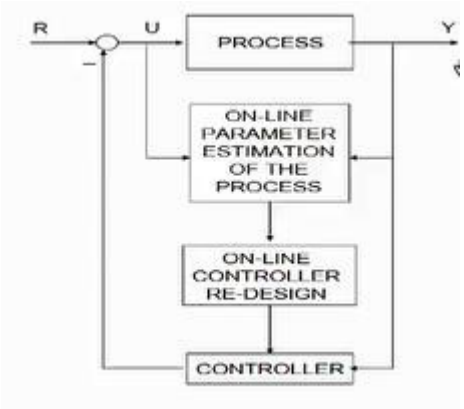

Fig 3.8: Block diagram of Indirect STC

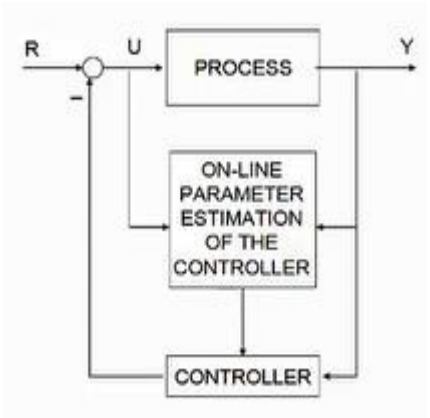

Fig 3.9: Block diagram of Direct STC

3.4.3 Pole Placement Adaptive Control

Controller based on the pole assignment of a closed control loop is designed to achieving the pre-set poles of the characteristic polynomial [14].
Consider a plant characterized by the transfer function:

$$Gp = \frac{B}{A} \tag{3.3}$$

Where A and B are polynomial in the forward shift operator, It is assumed that the degree of polynomials A is greater than the degree of polynomial B and that the transfer function Gp is causal, It is desired to design a regulator such that the transfer function from the reference value y_r to the process output y is given by the desired transfer function or what is called the model:

$$Gm = \frac{Q}{P} \tag{3.4}$$

Q and P are polynomial in the forward shift operator; it is assumed that the degree of polynomials P is greater than the degree of polynomial Q and that the transfer function Gm is causal.

Consider also the general controller:

$$Ru = Ty_r - Sy \tag{3.5}$$

R, S, and T are polynomials. This control law represents a negative feedback with the transfer operator -S/R and a feed forward with the transfer operator T/R.

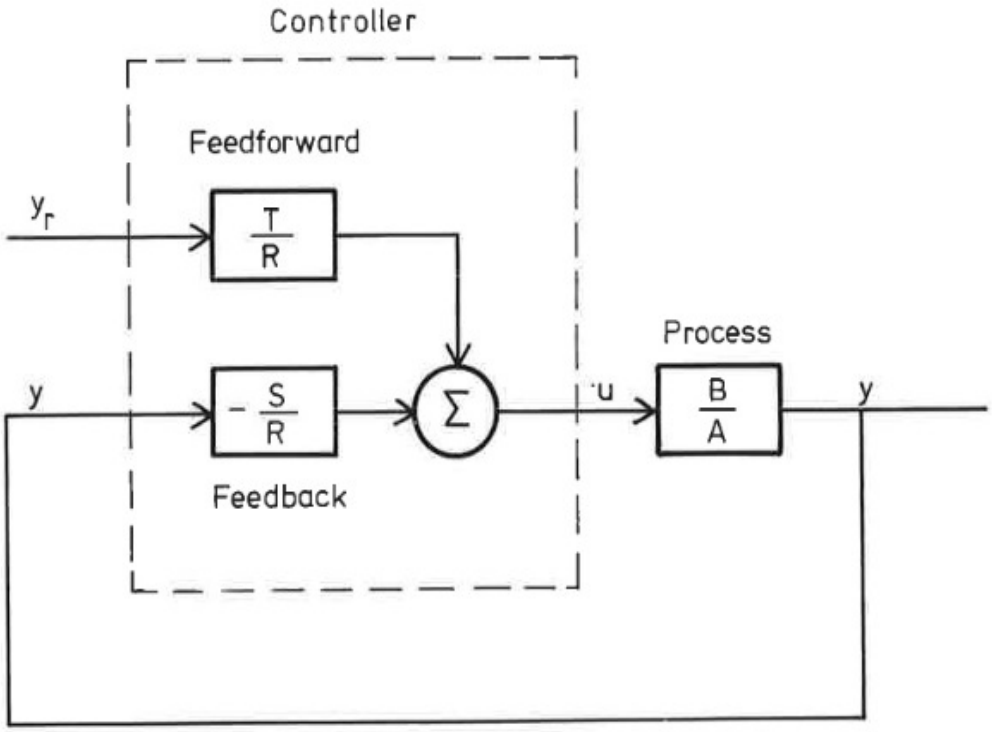

Fig 3.10: Block diagram of closed loop system.

The closed loop transfer function from the command signal y_r to the output of the closed loop system y is given by:

$$G = \frac{TB}{AR+BS} \tag{3.6}$$

The problem is to find polynomial T, B and S such that the closed loop transfer function G is equal to the desired transfer function Gm

$$G = \frac{TB}{AR+BS} = Gm = \frac{Q}{P} \tag{3.7}$$

The degree of the polynomial AR+BS is normally higher than the degree of the polynomial P. This means that the polynomials AR+BS and TB have common factors. By a direct comparison with state space design composed of state feedback and an observer, it can be shown that the common factors of TB and AR+BS correspond to the observer poles. The polynomial T can thus be factored into two parts, one part

corresponds to the desired observer poles and the other corresponds to the desired closed loop zeros.

The polynomials A and B characterize the process dynamics where the polynomials P and Q describe the desired closed loop transfer function.

- **Step 1:** Find greatest common divisor Q_2 of B and Q. Factor Q and B as

$$Q = Q_1 Q_2 \tag{3.8}$$

$$B = B_1 Q_2 \tag{3.9}$$

- **Step 2:** Determine desired observer poles, specified by the polynomial T1. Choose

$$T = T_1 Q_1 \tag{3.10}$$

- **Step 3:** Solve the following equation with respect to R and S

$$AR + BS = PB_1 T_1 \tag{3.11}$$

This is called Diophantine equation, it follows from this equation that B1 divides R:

$$R = B_1 R_1 \tag{3.12}$$

$$AR_1 + Q_2 S = PT_1 \tag{3.13}$$

The solution of the problem must give conditions to guarantee that there exist solutions to eq. ($AR_1 + Q_2 S = PT_1$) which gives a proper (continuous-time) or causal (discrete time) control law [10].

$$\deg P - \deg Q \geq \deg A - \deg B \tag{3.14}$$

$$\deg T_1 \geq 2 * \deg A - \deg P - \deg B_1 - 1 \tag{3.15}$$

- **Minimum Degree Pole Placement (MDPP)**

Data: polynomials A and B

Specification: polynomials Q, P and T_1

Compatibility conditions:

$$\deg P = \deg A \tag{3.16}$$

$$\deg Q = \deg B \tag{3.17}$$

$$\deg T_1 = \deg A - \deg B_1 - 1 \tag{3.18}$$

- **Step 1:**

Decompose B as

$$B = B_1 Q_2$$

- **Step2:**

Solve the Diophantine equation below to get R1 and S with

$$\deg S < deg\ R \tag{3.19}$$

$$AR_1 + Q_2 S = T_1 P \tag{3.20}$$

- **Step 3:**

From control signal

$$T = T_1 Q_1$$

$$R = B_1 R_1$$

Compute the control signal from the control signal

$$Ru = Ty_r - S\,y$$

For a desired behavior of controlled system, second order under-damped system; we need to locate the poles of characteristic polynomial, for that behavior using relationships between poles and systems characteristics "such as maximal overshoot, settling time, rise time,...ect"

Second order under-damped system step response transfer function is as follows

$$T(s) = \frac{{w_n}^2}{(s+\zeta w_n + jw_d)(s+\zeta w_n - jw_d)} \quad (3.21)$$

The poles of the system are $p_1 = -\zeta w_n - jw_d$ & $p_2 = -\zeta w_n + jw_d$.

Where $w_d = w_n\sqrt{1-\zeta^2}$ is called damped natural frequency.

The following figure, **Fig 3.10,** represent a typical under-damped second order system.

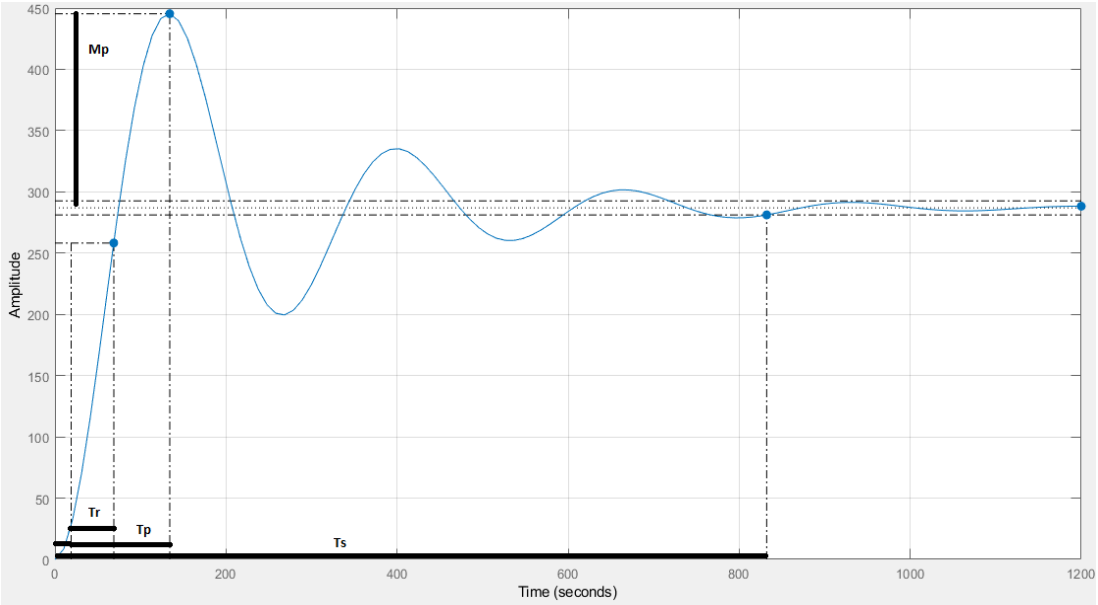

Fig 3.11: step response of 2nd order system.

- **Rise Time, T_r:** it is the time required for the response to rise from 10% to 90% of its steady state value.

$$T_r = \frac{\pi - \vartheta}{w_d} \quad (3.22)$$

ϑ is the angle of the pole from negative real axis.

- **Peak Time, T_p:** the peak time is the time at which the response reaches its maximum peak value.

$$T_p = \frac{\pi}{w_d} \tag{3.23}$$

- **Maximum Overshoot, M_p:** it is the difference between the maximum peak value of the response and steady state response of the system.

$$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{3.24}$$

- **Settling Time, T_s:** it is the time required for the response to reach and stay within a range about final steady state value, this range can be 5% of steady state value or 2% of steady state value.

$$T_s = \frac{4}{\zeta w_n} \text{ for 2\% criterion} \tag{3.25}$$

$$T_s = \frac{3}{\zeta w_n} \text{ for 5\% criterion} \tag{3.26}$$

Using those specifications we can define region of poles that cause the desired response of the controlled system.

The concept of adaptive pole placement controller is to identify system and put back poles in desired region.

As any adaptive controller, we need on-line system identification using recursive least squares estimation and a controller that can be updated when it needs to be.

The procedure to design a indirect pole assignment self tuning adaptive controller is as follows

- Define region of the desired poles.
- Build a transfer function with desired poles in s-domain.
- Convert obtained transfer function to the suitable z-domain equivalent, this will represent desired behavior.
- On-line recursive least squares estimation for tracking controlled system.
- Controller design parameters.
- Update parameters of the controller and the input filter.

Where the motor discrete-time transfer function is defined as

$$G(z) = \frac{b_0 z^1 + b_1}{z^2 + a_1 z^1 + a_0} \tag{3.27}$$

The desired behavior transfer function

$$Gm(z) = \frac{B_0 z^1 + B_1}{z^2 + A_1 z^1 + A_0} \tag{3.28}$$

The input filter

$$F(b,z) = \frac{B_0 z^1 + B_1}{b_0 z^1 + b_1} \tag{3.29}$$

The feedback controller

$$H(h,z) = \frac{h_1 z^1 + h_0}{b_0 z^1 + b_1} \tag{3.30}$$

Where

$$h_n = A_n - \alpha_n \tag{3.31}$$

The following, **Fig 3.11**, figure represents a full scheme of the controller and process

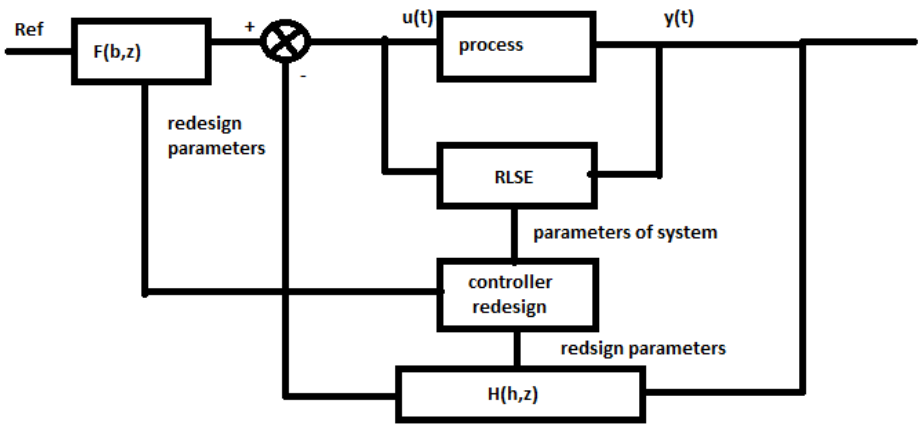

Fig 3.12: adaptive pole placement scheme

Chapter 4: Experimental And Simulation Results

4.1 Hardware Design

The experiment requires the following components:

- PMDC motor with optical encoder
- Driver shield
- Sensing circuit (current and voltage sensing)
- Power supply
- NI-DAQ PCI 6221-37 pin

The experiment is organized as follows:

Drive the motor with DC voltage through driver shield, A PWM circuit is designed and implemented. The driver receives 24 volts and PWM signal and generates an average voltage between 0 and 24 volts according to the duty cycle of the PWM. PCI is used to acquire the data (current voltage and speed)

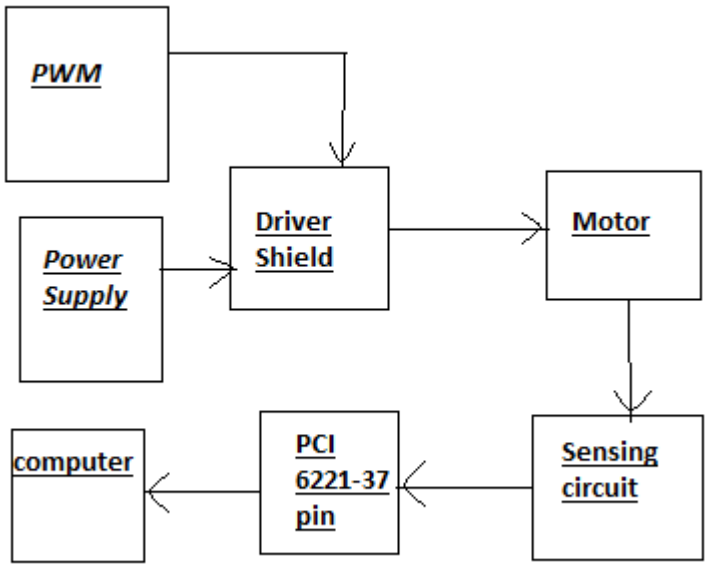

Fig 4.1: Architecture of the system

4.1.1 PMDC motor

MMP S17-400C-24V GP42-169

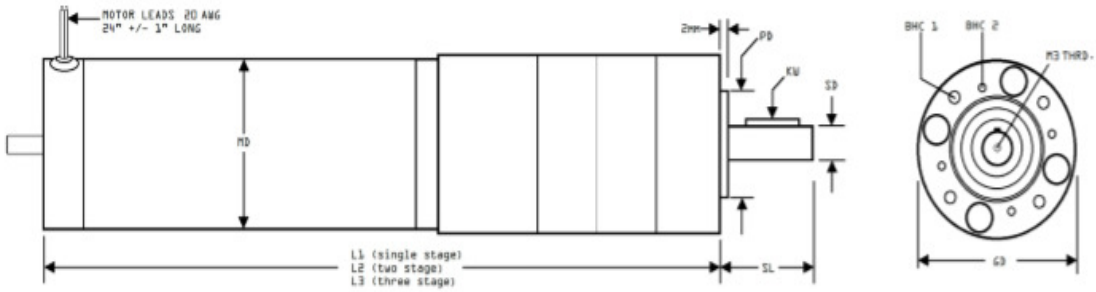

Fig 4.2: PMDC motor with gear head.

The motor in the experiment is connected to a gearhead and they are considered as one system, more characteristics are shown in the datasheet of the motor (Appendix B).

4.1.2 DC Motor Driver

This small size driver provides a cost optimized solution for protected high current PWM motor drives

Features

- Operating Voltage 5 to 27V (B+)
- Control motor speed by PWM up to 25 kHz.
- Motor forward and backward motion control
- Switched mode current limitation for reduced power dissipation.
- Current limitation level of 30 A Current sense capability
- Over---temperature shut down Over---voltage lock out.
- Large size heat sink is mounted to driver.

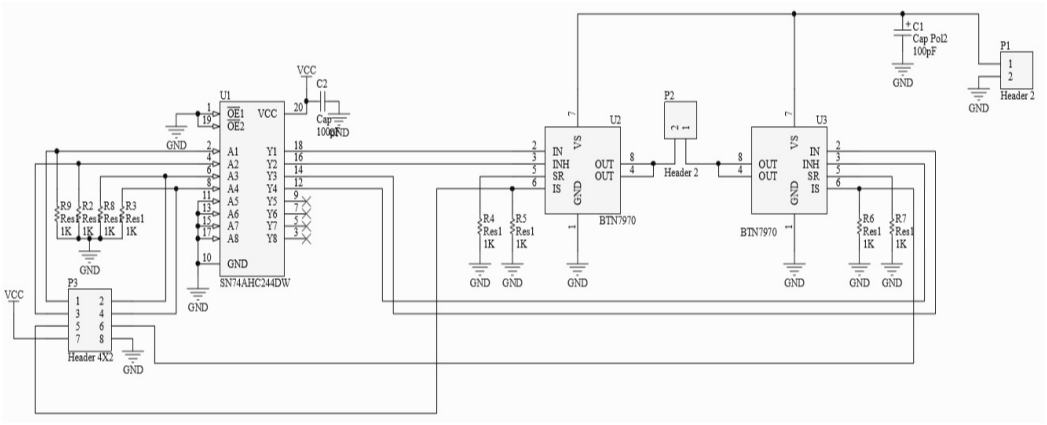

Fig 4.3: IBT-2 schematic

The driver consist of 3 main ICs, buffer and 2 BTN 7970 to provide an H-bridge

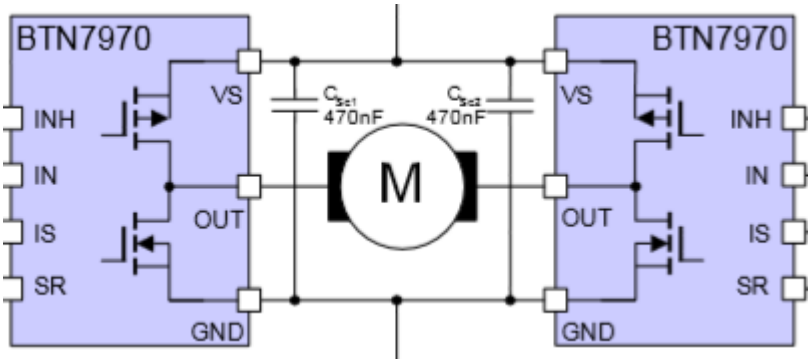

Fig 4.4: High current H-bridge using BTN7970

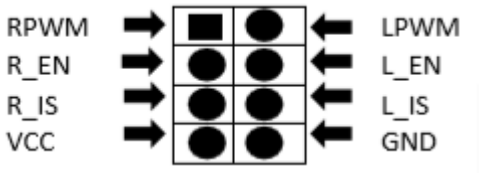

Fig 4.5: Input/output of the driver

Table 4.1: input/output of the driver

1	RPWM	Forward level PWM signal input active high
2	LPWM	Reverse level PWM signal input active high
3	R_EN	Forward drive enable input, HIGH enable, LOW Close

4	L_EN	Reverse drive enable input, HIGH enable, LOW Close
5	R_IS	Forward drive ´ side current alarm output
6	L_IS	Reverse drive ´ side current alarm output
7	VCC	5V power input
8	GRN	Ground
9	P1 and p2	24V power input and (0-24)v power output

4.1.3 Pulse Width Modulation

PWM is mostly used in controlling a voltage delivered to a load for example a motor. It is a square wave characterized by its frequency and duty cycle. The frequency is fixed according to the motor speed and the duty cycle is percentage variable. So the voltage delivered to the motor is determined by the duty cycle

$$V_{Delivred} = V_{Supplied} * Dutycycle$$

In this work the PWM frequency is fixed at 1Khz.the frequency is obtained from the speed of the motor, it is usually set to be at 5 times the rotation of the motor (RPS), our motor run at 6670 RPM (111 RPS), and the PWM frequency should be higher than 555Hz, so 1 KHz is largely enough value. A PWM circuit is designed and implemented based on integrator and comparator. A resistor and capacitor are chosen to make the frequency of PWM at 1 KHz.

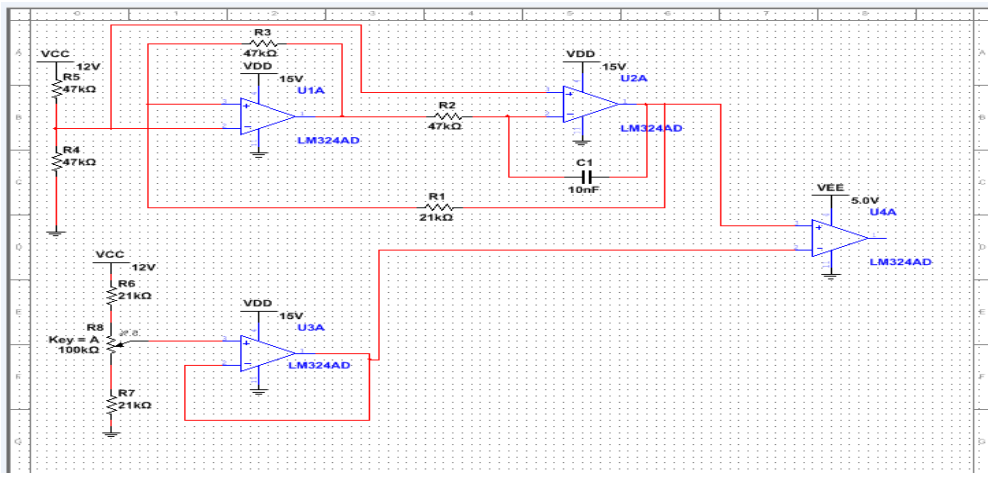

Fig 4.6: PWM circuit

4.1.4 Sensing Circuit

a. Current Sensing

A shunt resistor is used to measure the current flowing through the motor. Simply the principle of ohm's law is applied here .it is stated that there is a voltage drop across any resistor when current is flowing through it. Current sensing resistor is designed for low resistance so as to minimize power consumption. As result the shunt resistor is put in series with the motor, the calibrated resistance senses the current flowing through it in the form of voltage drop which detected and monitored by the PCI.

b. Voltage Sensing

In order to read the voltage by the data acquisition the voltage should be between 0 and 5 volts, so a voltage divider is used to map the voltage from (0-24) volts to (0-5) volts. A simple example of potential divider is two resistors in series, the output is going to be the voltage across one resistor, and it has linear relationship with the input voltage.

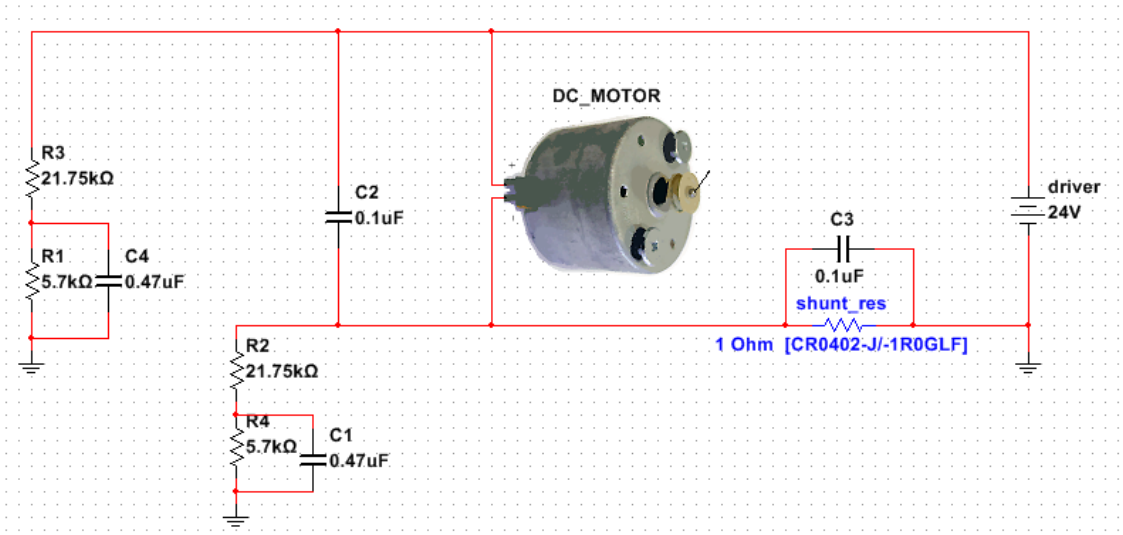

Fig 4.7: Sensing circuit

c. Motor Speed

An optical encoder is used to sense the speed. The encoder has a resolution of 500 slits per revolution which provides high precision. So the output of the encoder from one of its channels will be a square wave depending on the speed of the motor.

Motor's speed is measured either by using A channel or B channel that have the same precision (500 CPR) but they are phase shifted, this shift is used to determine the direction of the rotation.

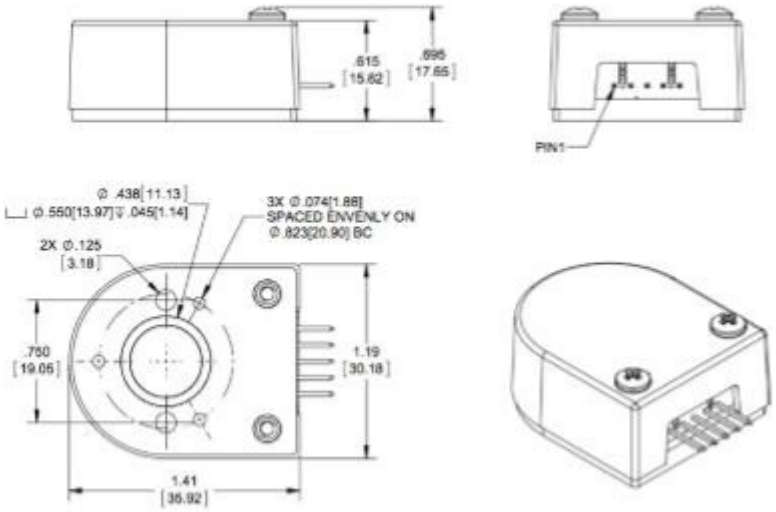

Fig 4.8: Mechanical drawing of the encoder

The encoder has 5 pins:

Table 4.2: pin assignment of the encoder.

Pin	Description
1	Ground
2	Index
3	A channel
4	5V dc power
5	B channel

4.1.5 Data Acquisition (NI-PCI 6221-37pin)

The motor speed can be increased up to 6670 rpm. So the output of the encoder will be a square wave with frequency of 55KHz.and we are also going to measure the current and the voltage in parallel. This is why we need data acquisition with high sampling rate.

PCI has 16 analog input (8 differential or 16 single ended) .its maximum sample rate is 250KS/s. It has 2 analog outputs with sampling rate of 833KS/s, and 8 digital I/O with sampling rate of 1MS/s.

Fig 4.9: PCI 6221-37 pin

With appropriate code in MATLAB, two analog inputs are used to acquire current and voltage data and one digital input to acquire the pulses data from the encoder. The sampling rate of each input to is set 125KS/s .the data is saved to workspace in order to be used later for parameter identification.

Table 4.3: Used Pins of PCI card.

Channel	Mode	Purpose
A0	Differential	Measuring the current flowing through the motor
A1	Differential	Measuring the voltage across the motor
PFI 0	Digital input	Acquire pulses from optical encoder to measure the speed

4.1.6 Results

Using an appropriate code (appendix A), speed of the motor is calculated, an average filter is applied to the speed, current and voltage.

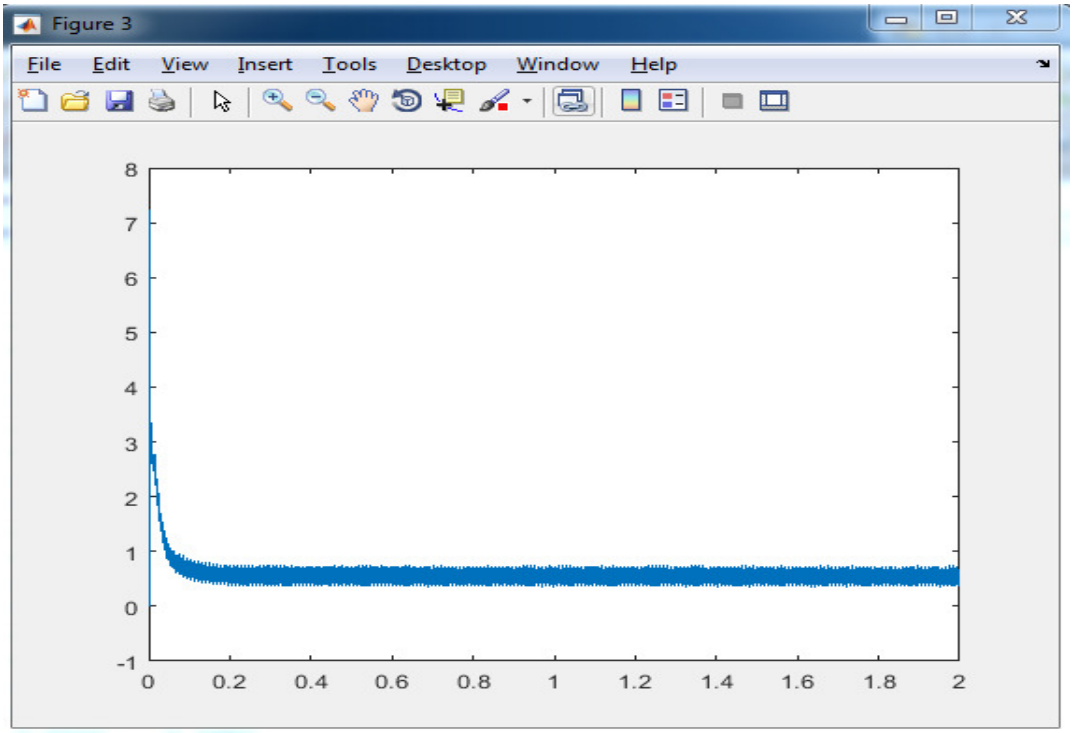

Fig 4.10: Armature average current

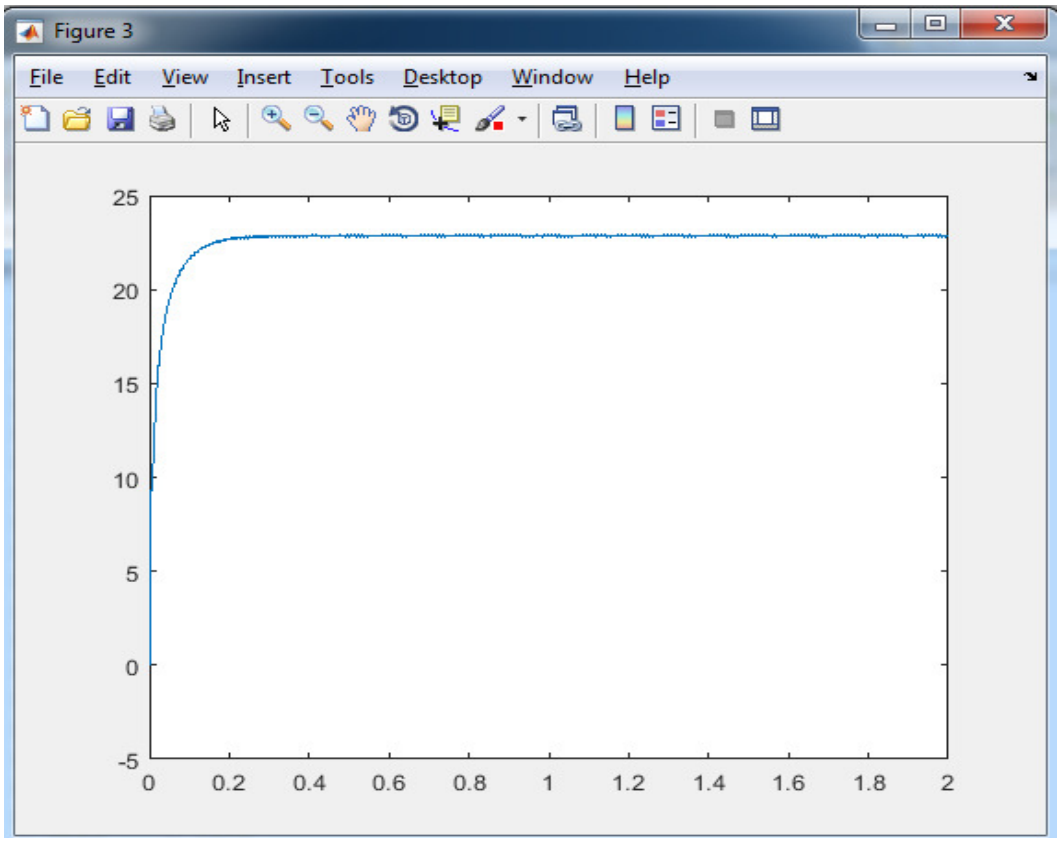

Fig 4.11: Input voltage

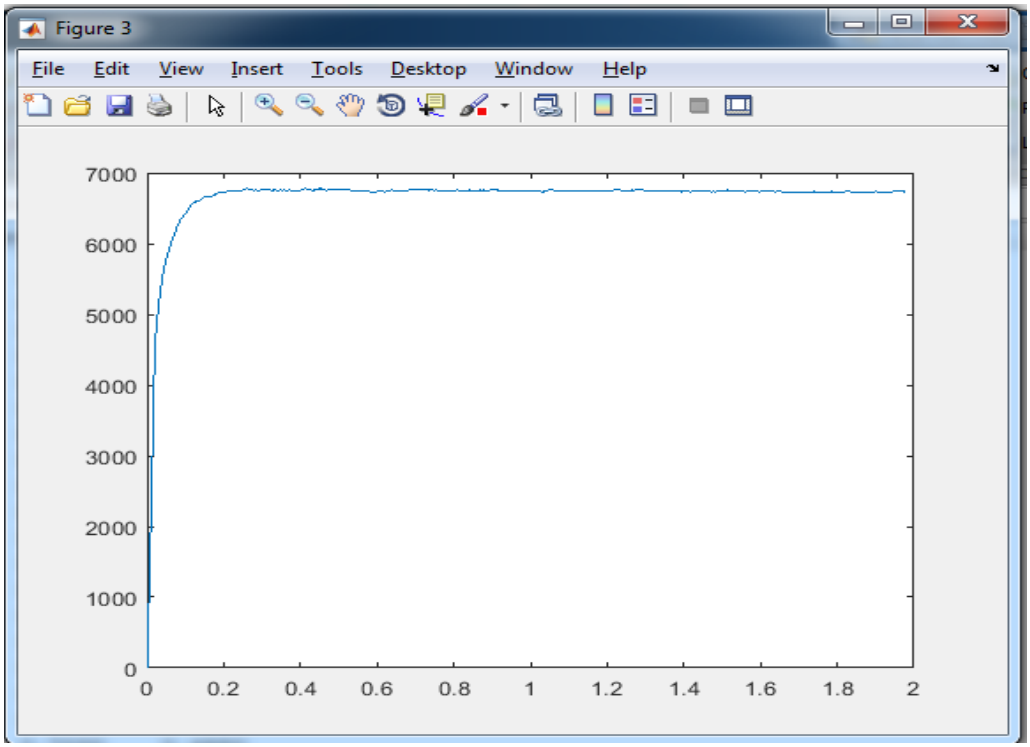

Fig 4.12: Speed of the motor (rpm)

Comments

Input voltage of 24 volts is applied to the motor. Speed of the motor is initially zero, that is to avoid the motor starting inrush current required when starts rotating (highly increased inertia if non-zero speed is desired), as the speed of the motor increased the inertia decreased and the current is decreasing. So the current has relationship with the inertia of the motor.

4.2 Offline Parameter Estimation

In order to describe our system mathematically as close as possible, we have used MATLAB SIMULINK parameter estimation tool, which is a powerful tool to identify our system.

At first we need to collect experimental data which are:

- Armature Current (A).
- Motor Speed (rad/sec).

After the acquisition of the needed experimental data, we build the SIMULINK model as shown in **Fig 4.13**

Where the relationships between input Motor Voltage V(s) and the outputs Armature Current Ia(s) and Motor Speed W(s) are describe as follows:

$$\frac{Ia(s)}{V(s)} = \frac{s+\frac{B}{J}}{La.s^2+\left(Ra+\frac{La.B}{J}\right)+\left(\frac{Ra.B}{J}+\frac{Ke.Kt}{J}\right)}$$

$$\frac{W(s)}{V(s)} = \frac{\frac{Kt}{J}}{La.s^2+\left(Ra+\frac{La.B}{J}\right)+\left(\frac{Ra.B}{J}+\frac{Ke.Kt}{J}\right)}$$

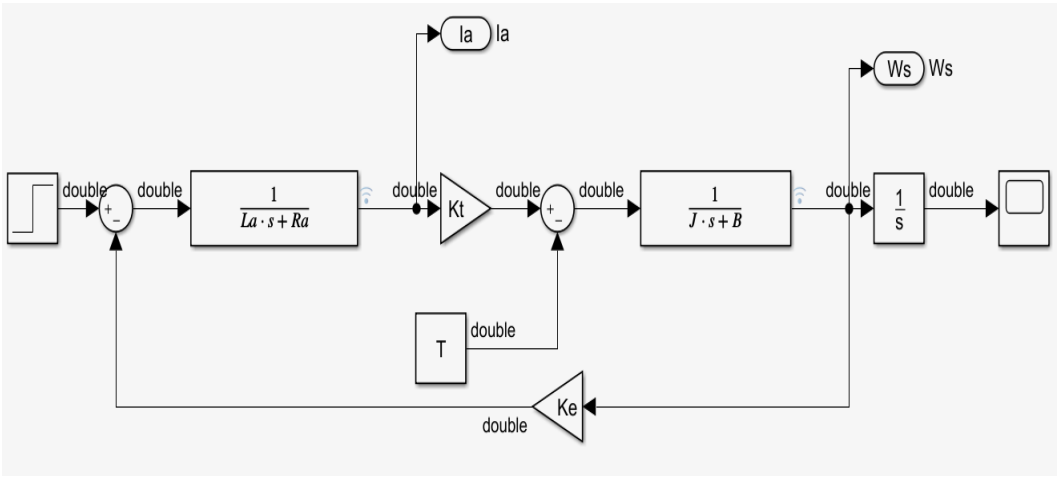

Fig 4.13: SIMULINK model of the DC Motor.

After we have built the motor model on SIMULINK we go to parameter estimation toolbox.

Select Analysis => Parameter Estimation => select new experiment => assign inputs and outputs => specify parameters to estimate.

Select cost function to minimize => select more option => choose Method used for estimation => start estimation => validate results (estimated parameters).

As a cost function we have chosen **Sum of Squared Errors** which is the cost function by default**.**

As algorithm (Method) we have used two different methods, in order to have the best identification for our system. These different methods are

- Non-linear Least Squares.
- Pattern Search Genetic Algorithm.

In the following figures we show you the estimated parameters, validation of these parameters and their residuals with measured signals.

We start by Non-linear Least Squares:

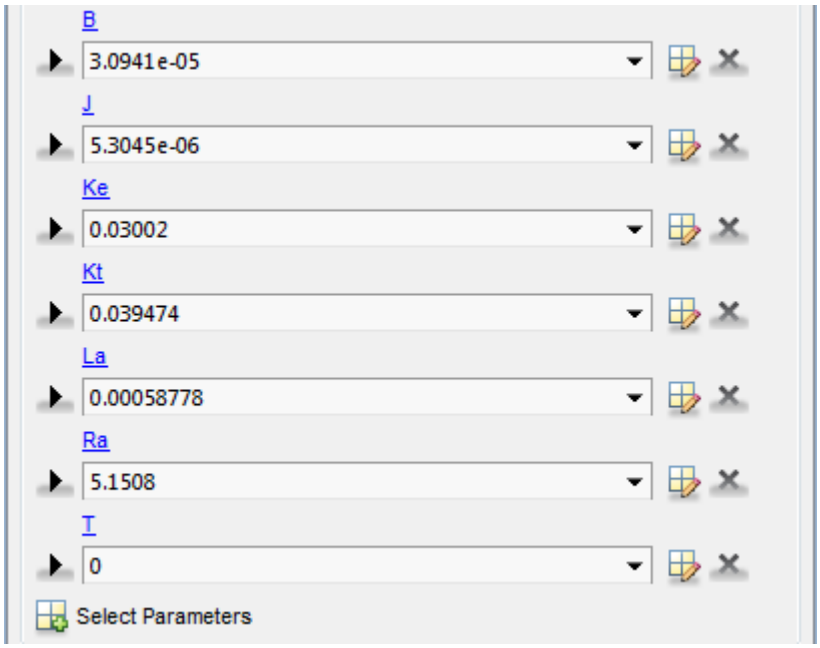

Fig 4.14: Estimated Parameters

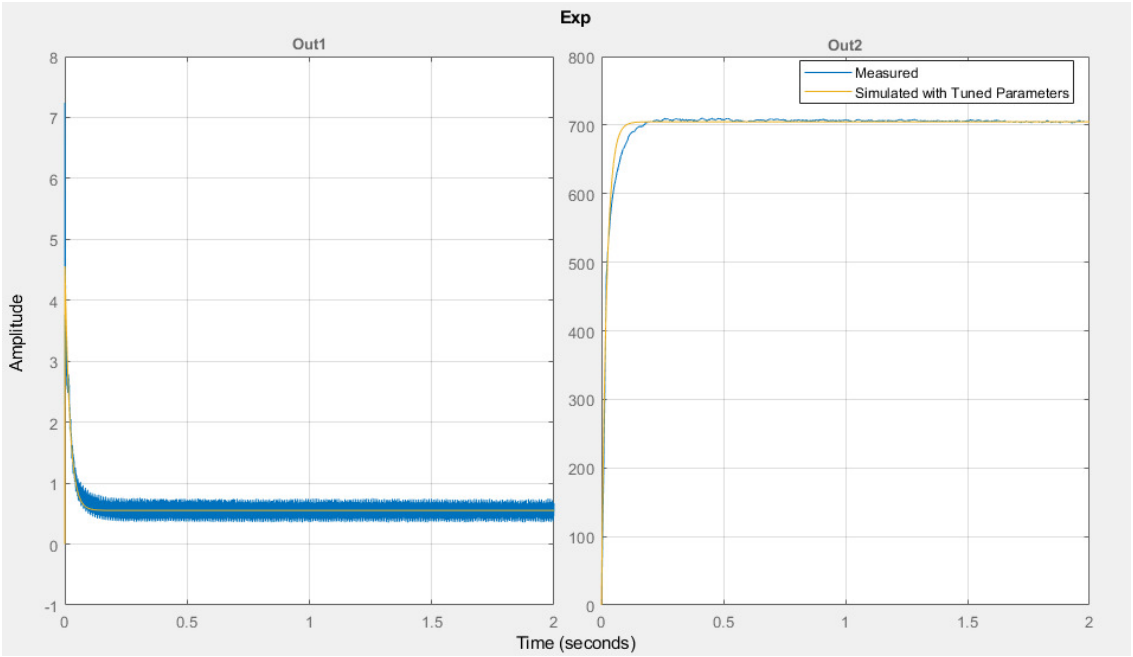

Fig 4.15: measured signals and simulated with tuned parameters signals

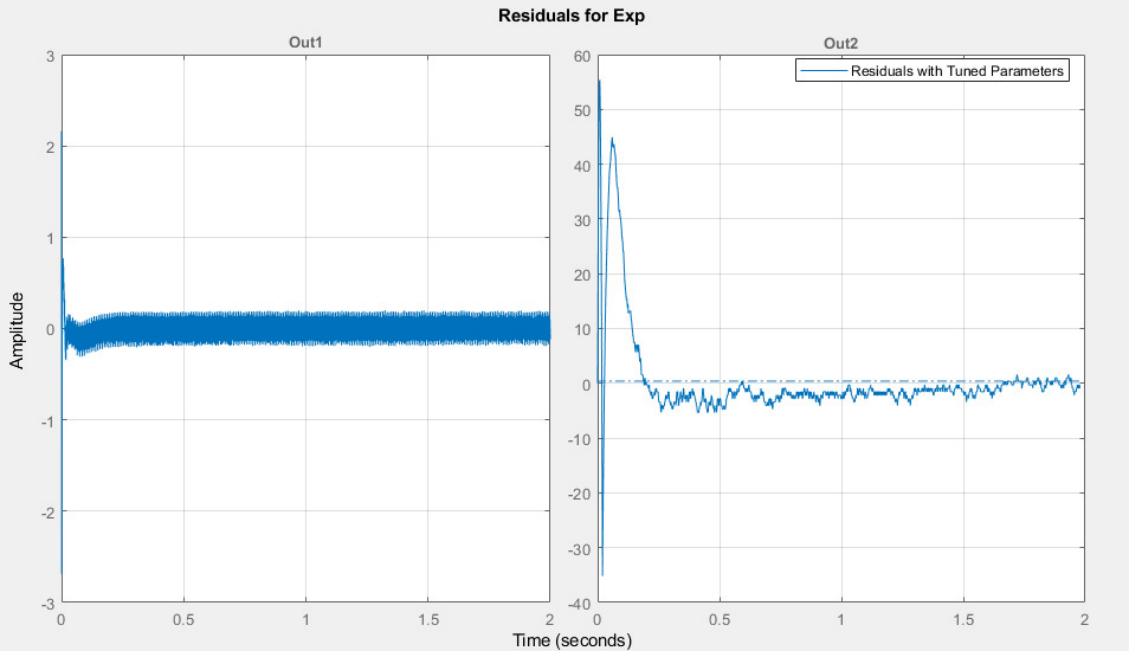

Fig 4.16: residuals of estimation.

Now we move to the second used algorithm Pattern Search Genetic Algorithm:

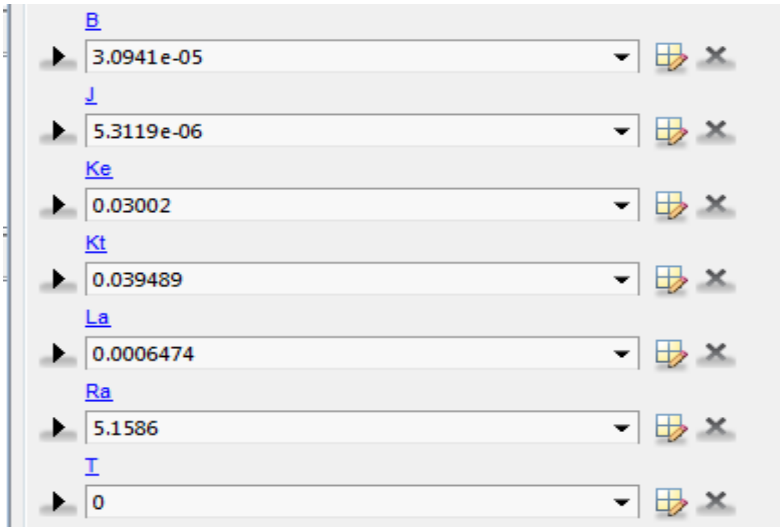

Fig 4.17: estimated parameters

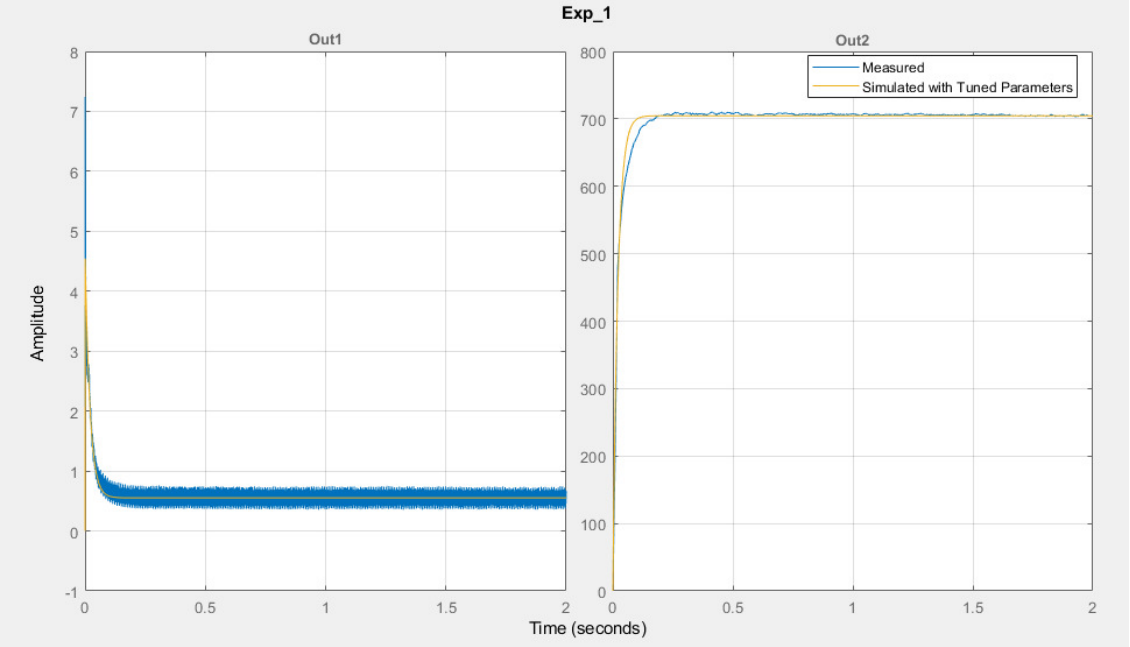

Fig 4.18: measured signals and simulated with tuned parameters signals.

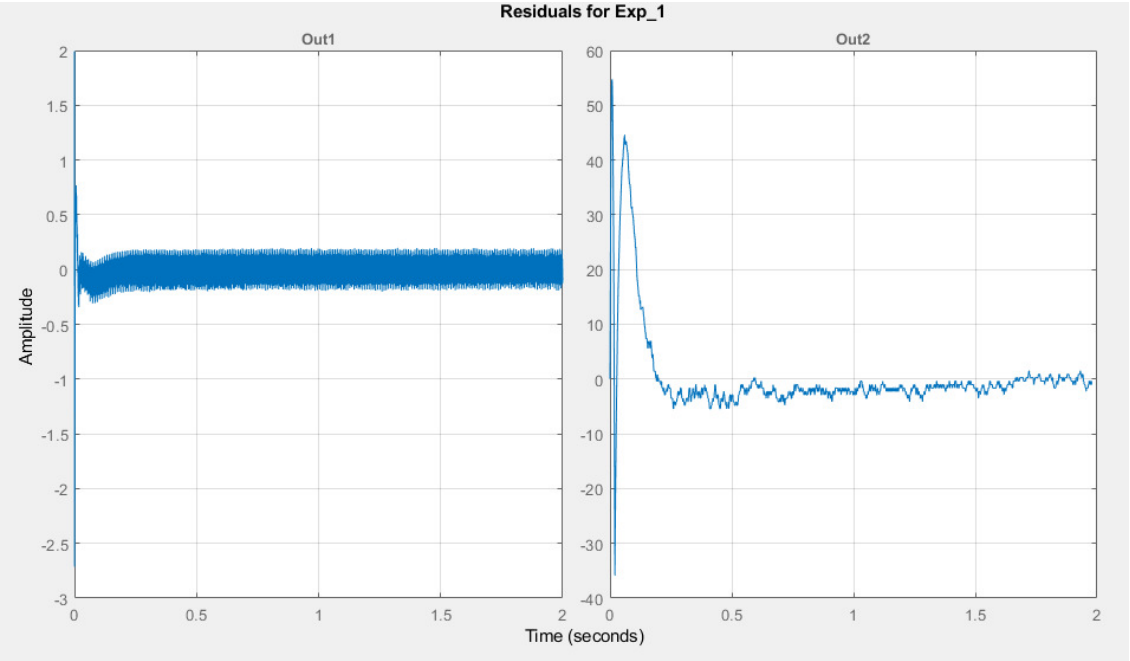

Fig 4.19: residuals of estimation.

After gathering all those data we need to choose the most accurate ones, in the following table we do the comparison of them with some data from the motor's data sheet.

Table 4.4: obtained parameters from the two different methods.

Parameters	Pattern Search Genetic Algorithm	Non-Linear Least Squares
Ra(Ω)	5.1586	5.1508
La(mH)	0.6474	0.58778
Ke(V/(rad/sec))	0.03002	0.03002
Kt(N.m/A)	0.039489	0.039474
B	$3.0941\ 10^{-5}$	$3.0941\ 10^{-5}$
J	$5.3119\ 10^{-6}$	$5.3045\ 10^{-6}$

From **Table 4.5**, we see that the results obtained using Non-Linear Least squares method is closer to the ones obtained by Pattern Search Genetic Algorithm. In this thesis we've worked with Non-Linear Least Squares method because it does not take too much time to converge as the Pattern Search Genetic Algorithm does; however as trade of fast convergence we need to choose a good initial guess in order to converge to the satisfied parameters because if we choose randomly the initial parameters we may end up with a different system's parameters.

$$\frac{I_a}{V_s} = \frac{1.7013e{+}03*(s{+}5.8330)}{s^2{+}8.7690e{+}03*s{+}4.3118e{+}05}$$

$$\frac{\mathrm{W(s)}}{\mathrm{V(s)}} = \frac{4.3740}{s^2{+}8.7690e{+}03*s{+}4.3118e{+}05}$$

4.3 Validation Of Estimated Model

After the identification of the system we will implement a PID controller to validate our system, using SIMULINK PID tuner to build a robust PID controller.

First we start by the implementation of the circuit shown in the following figure, which represent a feedback PID controller with internal saturation from 0V to 24V to control our estimated system.

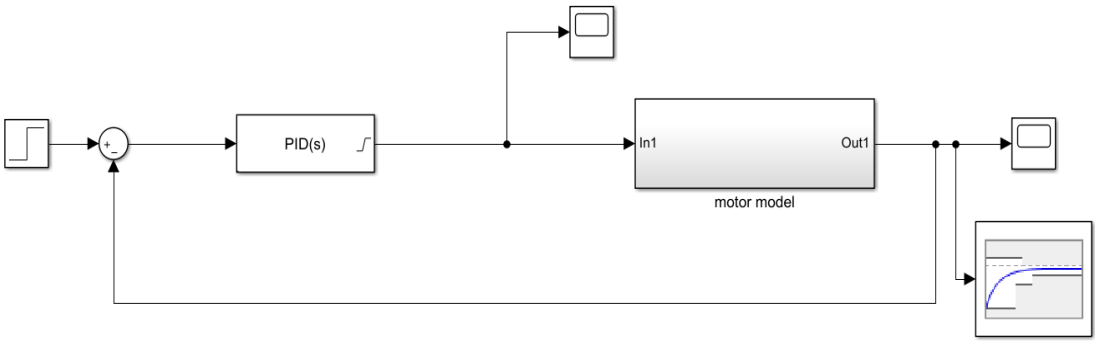

Fig 4.20: feedback PID controller circuit.

Then we give initial values for our PID controller, before obtaining optimized ones, as shown in the figure bellow

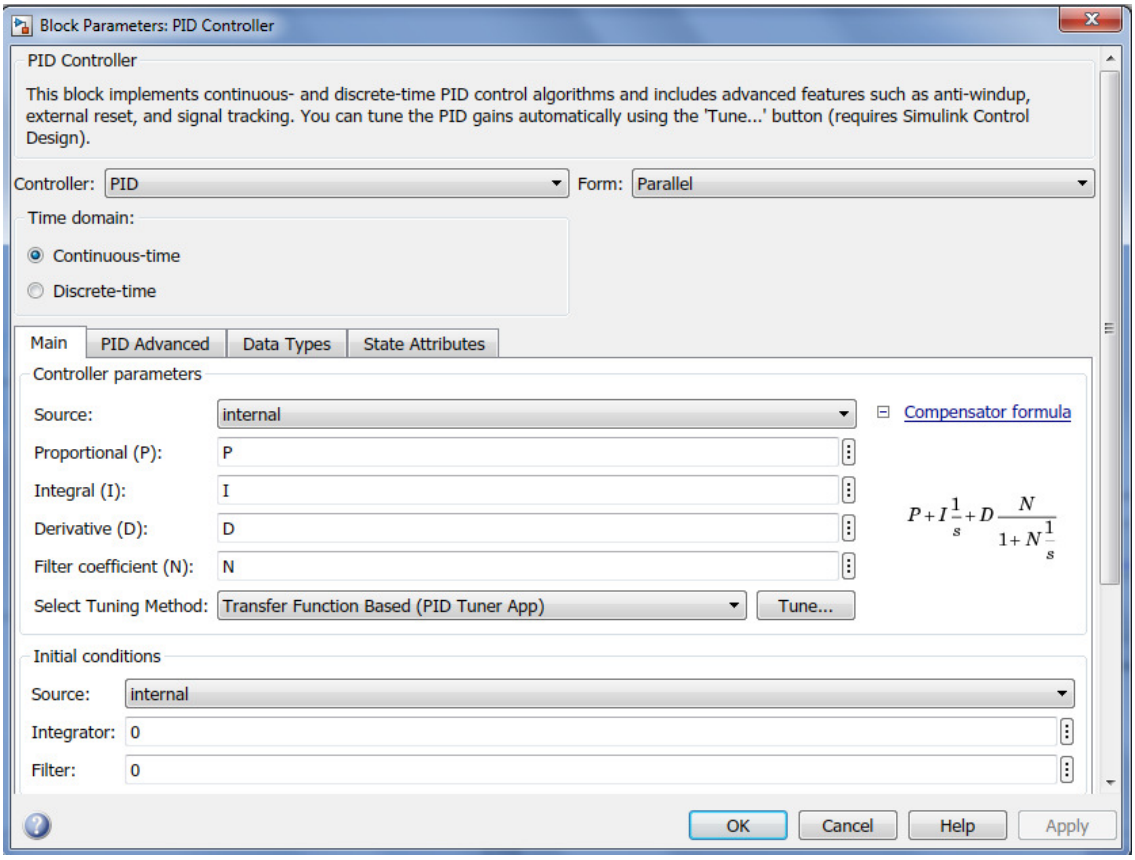

Fig 4.21: initial values of PID block.

For the optimization of controller values we will use check step response characteristics block from SIMULINK library, and set our desired criterion as follow

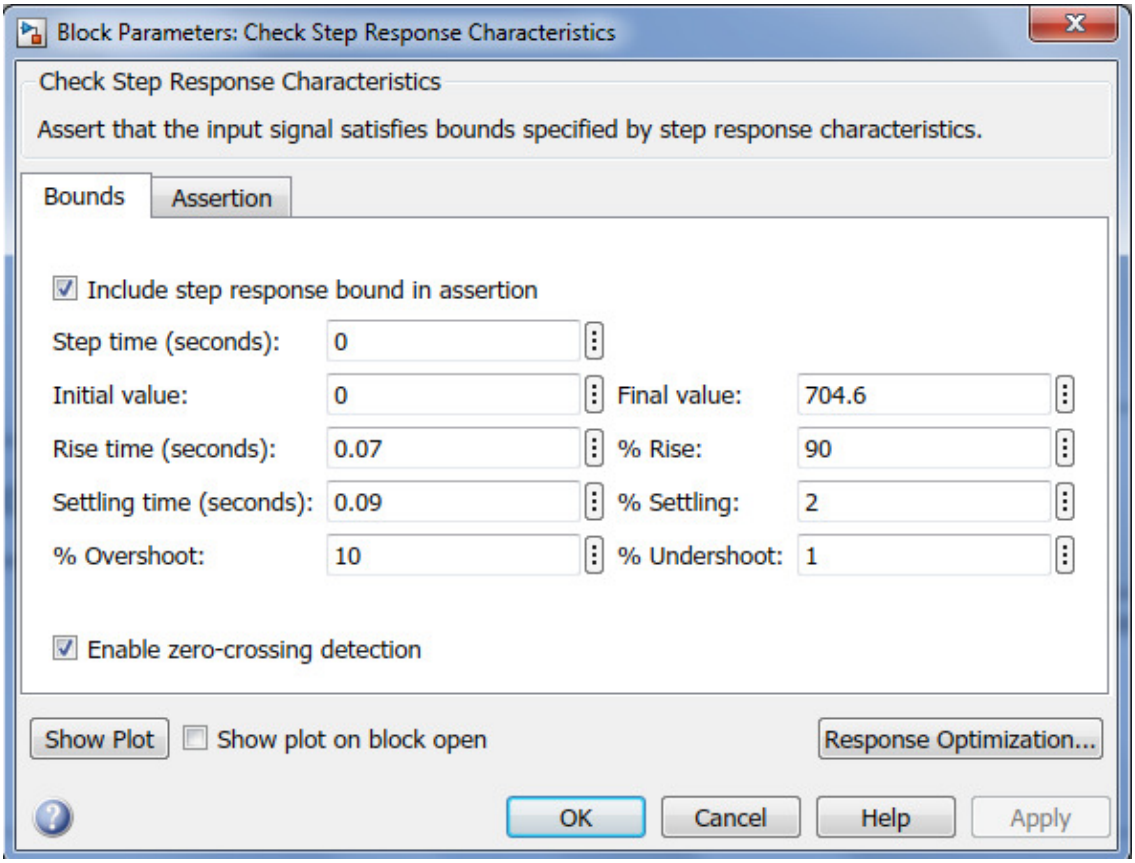

Fig 4.22: criterion for our optimization.

Click on apply button then click on Response Optimization button.

New window will pop up select new variables => select variables of the controller P, I, D and N => click on plot system response then click on run optimization button, and optimization will start until there is a convergence.

The result of this procedure is shown in the following figures, **Fig 4.23, Fig 4.24,** this PID will be implemented to control the real system and see if it works correctly.

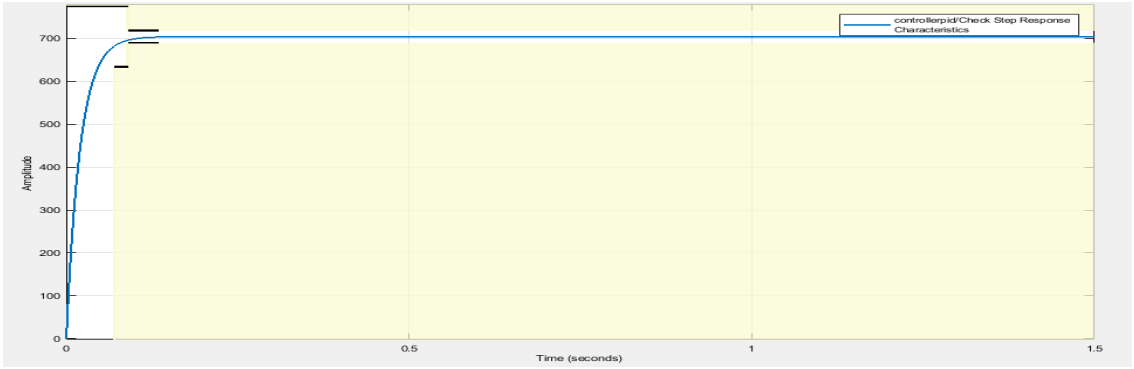

Fig4.23: optimized response.

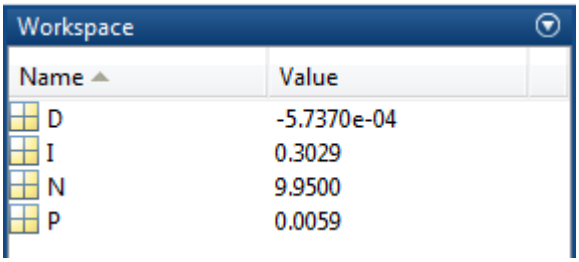

Fig 4.24: optimized parameters.

After obtaining parameters of the controller we will apply this controller to our motor to see if the estimated system is as close possible to the real one, the following figure **Fig4.25** Represent the SIMULINK circuit

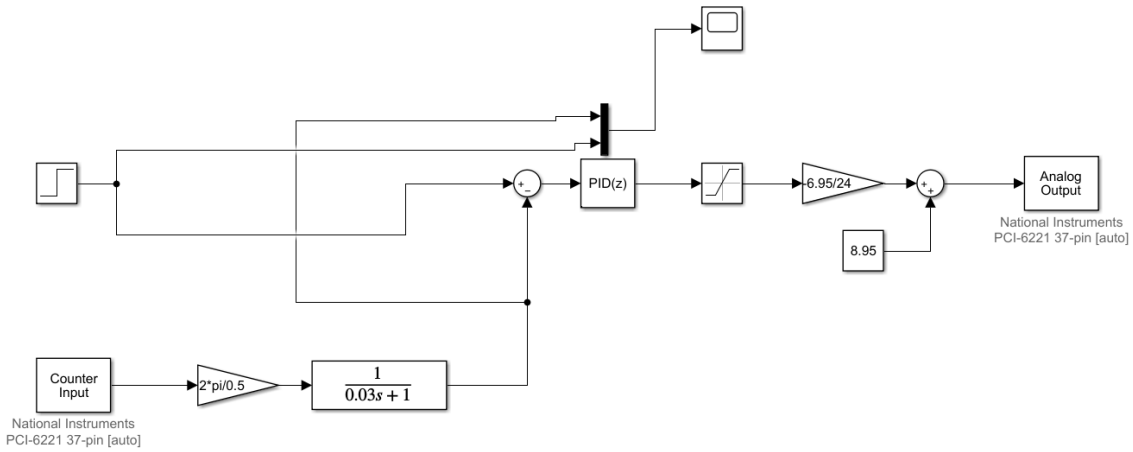

Fig 4.25: application of the PID on the real system.

The following figure, **Fig4.26,** shows how the real system responses to the reference speed.

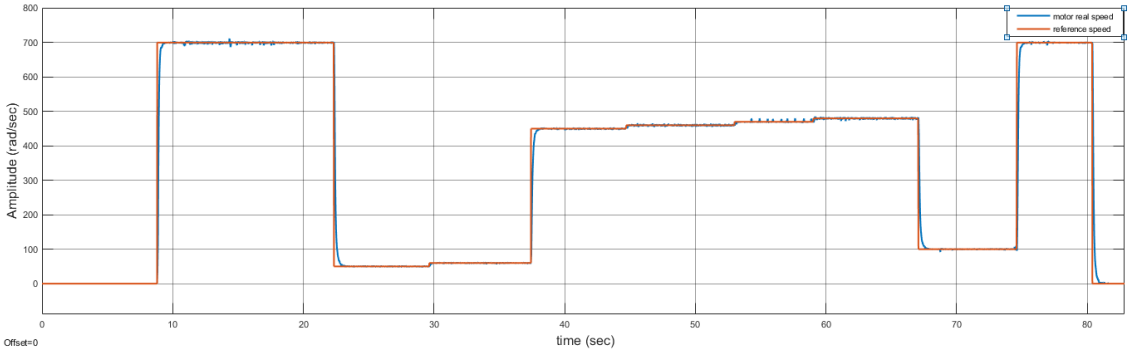

Fig 4.26: real motor speed.

Comment: from this result we see that we have a good estimation of our system since we have almost the same response from the estimated and real system.

4.4 Online Parameter Estimation

The model of the system is a second order system. So far offline parameter estimation is done (sec. 4.2). This method obtains a fixed parameter based on some collected data. During the operation of physical system this parameter can change. This is why we need an online parameter estimation based on new data. Online parameter estimation is typically performed using a recursive algorithm. To estimate the parameter values at a time step, recursive algorithms use the current measurements and previous parameter estimates. Therefore, recursive algorithms are efficient in terms of memory usage [8].

The model of the system in discrete form is:

$$M(z) = \frac{Y(z)}{U(z)} = \frac{b1Z^{-1}+b0z^{-2}}{1+a1Z^{-1}+a0Z^{-2}}$$

Where [b1, b0, a1, a0] are parameters to be estimated. From the model, two previous input data and two previous and one new output data and the previous parameters estimated are needed to calculate the new parameters.

The Recursive Least Squares Estimator block in SIMULINK is used to perform the online estimation. It estimates the parameters of a system using a model; that is linear in those parameters. Such a system has the following form:

$$y(t) = H(t).\theta(t)$$

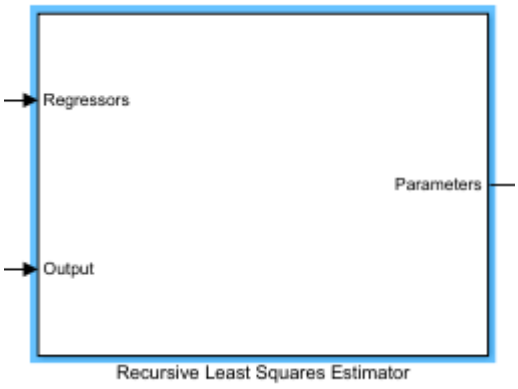

Fig 4.27: SIMULINK RLSE block

y and *H* are known quantities that we provide to the block to estimate θ. So the block has two in-ports and one out-port, the sample time should be specified in the

experiment, 0.001 is fixed as sample time. The regressors block is designed based on the model obtained. it has two inputs (the input voltage and the output speed)

Fig 4.28: regressors' subsystem block

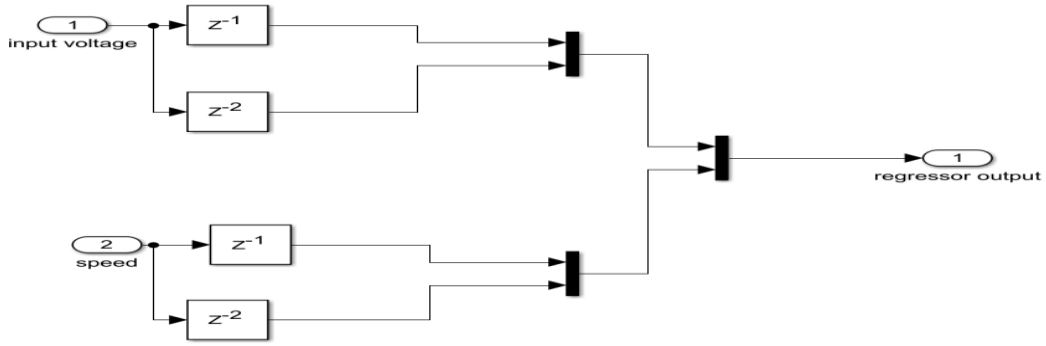

Fig 4.29: inside connection of the regressors block

We have used PCI to measure and determine the speed of the motor; we have designed an appropriate SIMULINK model in order to calculate the model parameters and validate it by comparing the response of the real system and the obtained model. The obtained model is a discrete transfer function with variable parameter.

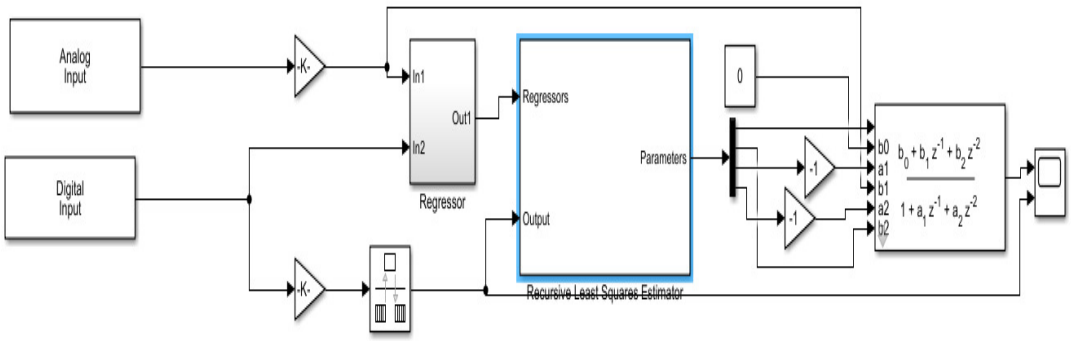

Fig 4.30: Simulink model for online parameter estimation

the response of real system and estimated one due to the same input voltage is displayed online in the same scope.

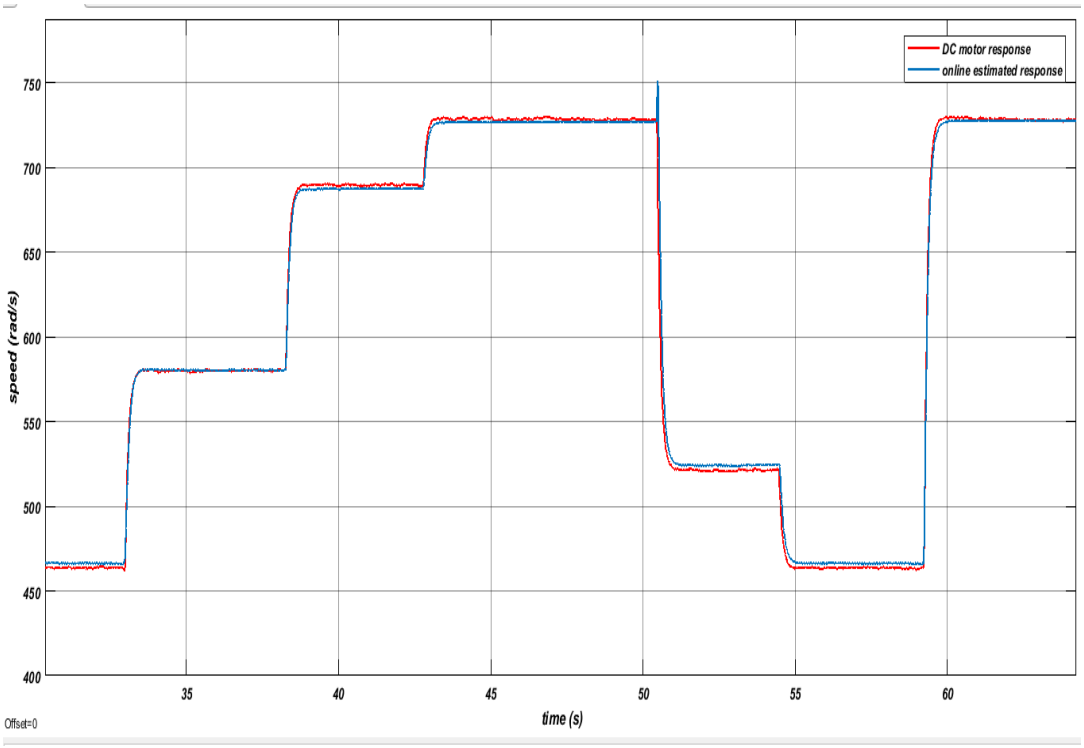

Fig 4.31: response of the real system and the estimated one.

From **Fig.4.31**, the estimated system follows the real system with small error, which means that the online parameter estimation has been done successfully.

4.5 Explicit (Indirect) Self-Tuning Control

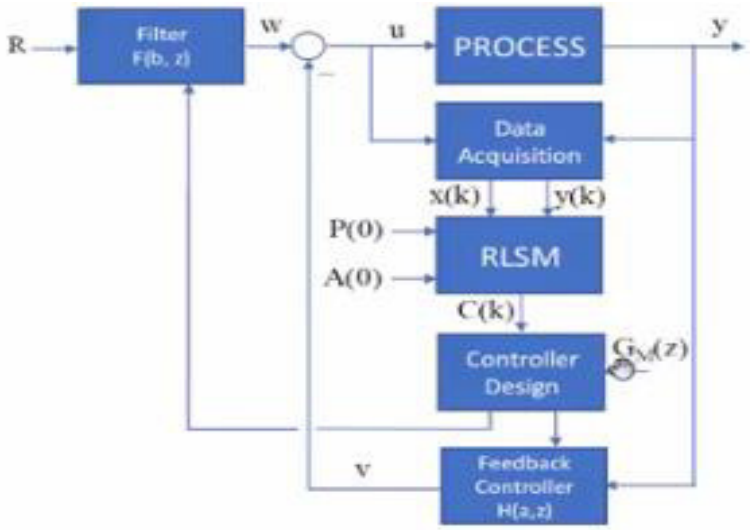

Fig 4.32: Block diagram of self tuning control.

From the block diagram:

Monitoring the input and the output of the process, this data is provided to the data acquisition block; that is used for tuning and estimating the parameter of the process. The next block is the RLSM block that picks up data created by the data acquisition (analog signals) and converts it into numerical values of the new transfer function parameters. These parameters are used by the controller design block, which provides information to the feedback controller and the filter.

4.5.1 Controller Design

This functionality requires that design specifications of the control system be represented by an S-domain transfer function of closed loop system (model transfer function) Gm(s). This transfer function must be converted into the Z-domain transfer function Gm(z) using time step Ts and the zero-order-hold option .

$$G_m(z) = \frac{\beta 1\, Z+\beta 0}{Z^2+a_1*z+a_0}$$

This transfer function will not change during the entire STC procedure, following each step of the RLSM procedure, the system controller H(z) and the filter F(z) will be redefined as follows

$$F(z) = \frac{\beta 1 Z+\beta 0}{b_1*z+b_0}$$

$$H(z) = \frac{h_1*z+h_0}{b_1*z+b_0} \quad \text{Where} \quad h_1=\alpha_1-a_1 \text{ and } h_0=\alpha_0-a_0$$

The desired transfer function response is calculated based on settling time and percentage overshoot as given by:

$$Ts=0.2s \text{ and } PO=5\%$$

$$G_m(z) = \frac{2.555e^4}{S^2+40S+839.9}$$

Converting into discrete time using sampling time Ts=0.001 (using Zero Order Hold).

$$G_m(z) = \frac{0.0126*z+0.01244}{z^2-1.96*z+0.9608}$$

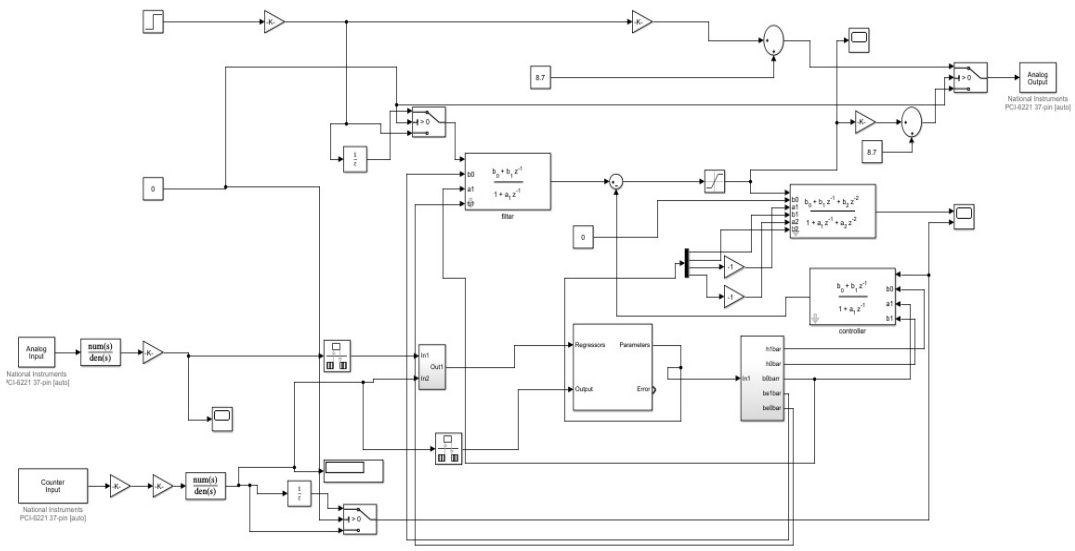

Fig 4.33: SIMULINK block diagram of STC

From the block diagram:

NI-PCI data acquisition board is used as interface between the controller which is in the computer and the DC motor.

1. Data acquisition is performed using NI-PCI
2. This data is used to estimate the parameter of the model using RLSE block in SIMULINK.
3. Controller design block tune and calculate the new controller and filter parameter. So calculations, data acquisition, controller and filter are performed in SIMULINK.
4. Control signal is outputted using analog output of NI-PCI then converted to PWM.

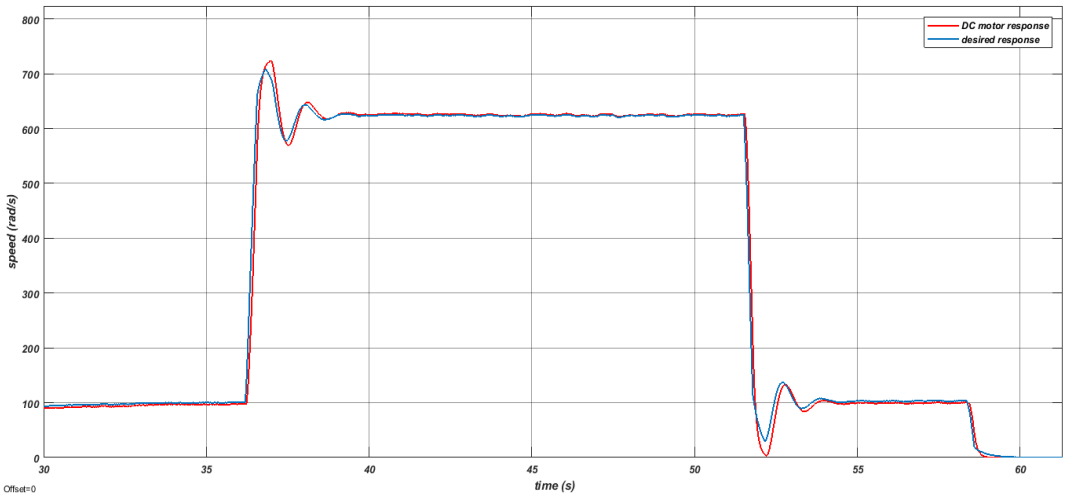

Fig 4.34: DC motor and desired response

Following results may be concluded from the DC motor and desired response graph:

- The plant response followed the reference model until closed to it after some time due to the estimation process.
- The plant and desired model follow the same behavior.

4.5.2 Effect Of The Load On The Speed

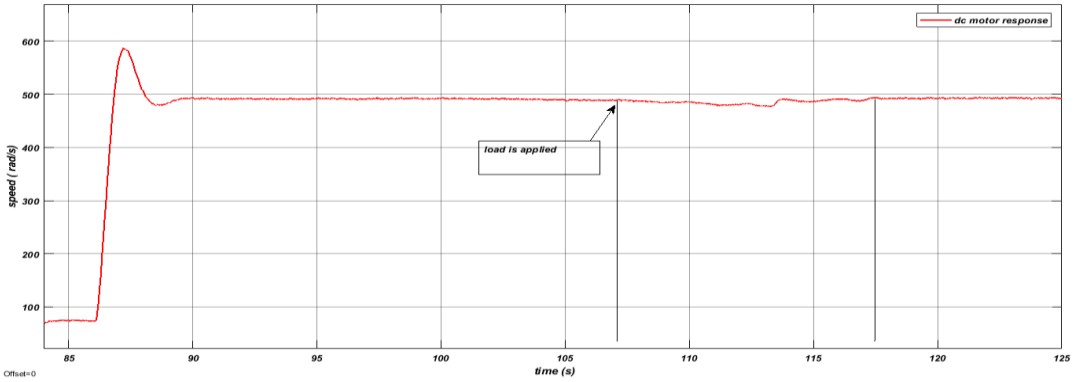

Fig 4.35: effect of the load on the motor

In this experiment a ramp load is applied, so the speed start to decrease for a while and then it converged to its desired value. As con

When changing the gain of the DC motor or changing the moment of inertia or adding load or increasing it; the tracking error is reduced until becomes zero and the STC response is closed to the desired response.

4.5.3 Effect Of Load On The Armature Current

1. **Step Load:**

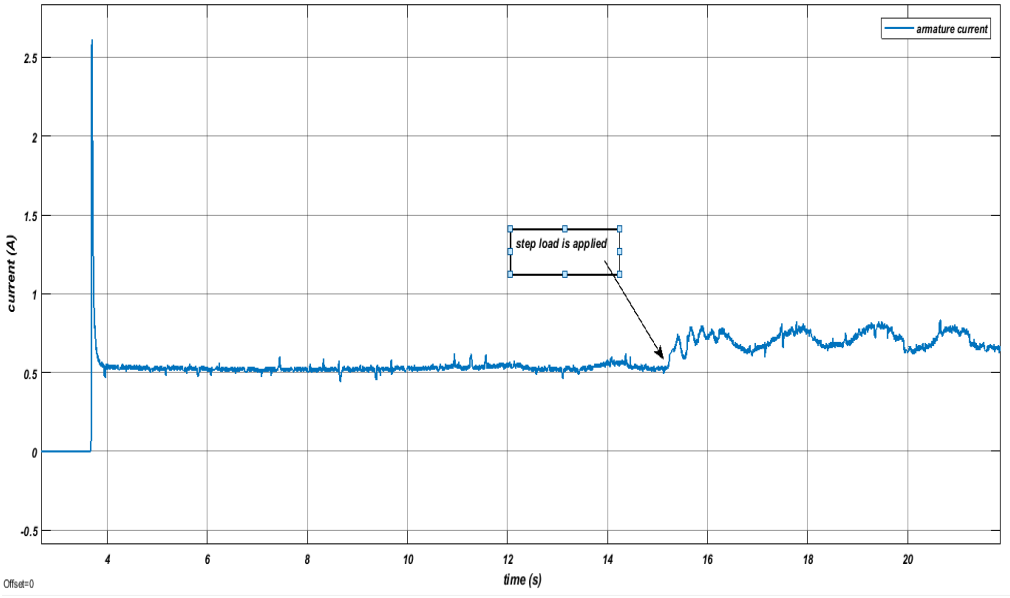

Fig 4.36: armature current response (due to step load)

2. Ramp Load:

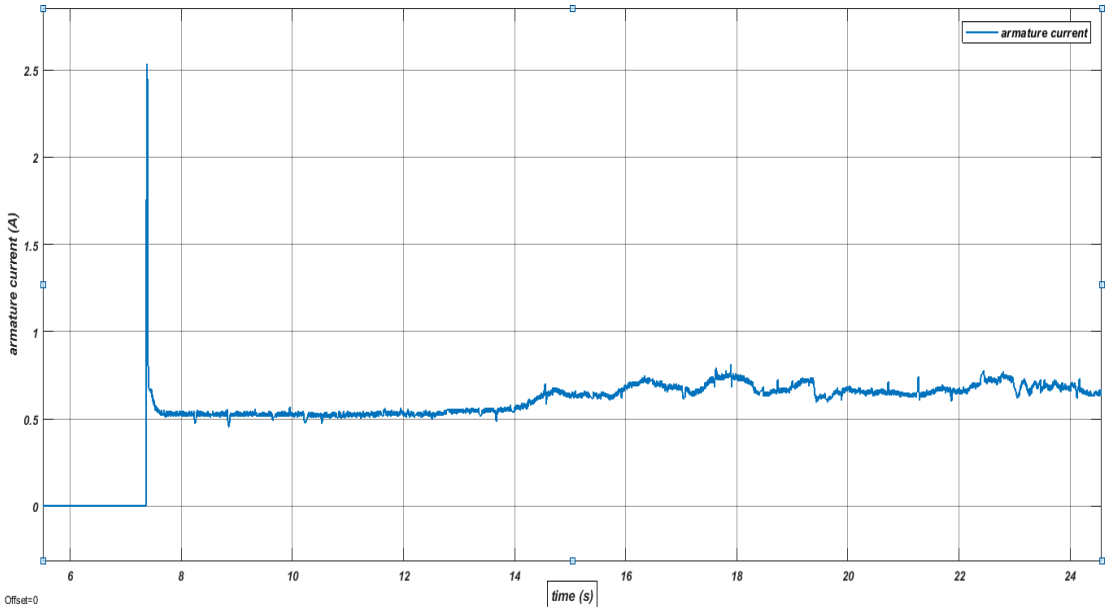

Fig 4.37: armature current response (due to ramp load)

when the load is applied the torque needed to make the motor rotate is increased, then it needs more power, so the current is increased to provide sufficient torque. The response of the current to the ramp load and step load is different , in the step load case the current is increased abraptly, in the case of ramp load the current is increased gradually.

4.6 Simulation Results

Effect of plant parameters changes on the system response :

To see the effect of parameter changes on the system response, we keep the controller fixed (fixed parameter) and then bring some changes on the system parameters as given in the following table:

Table 4.5: illustration how parameters change.

G(Z)	Q	Results
$\frac{b_1*z+(b0+q)}{z^2+a_1*z+a_2}$	[0 to 0.001]	Fig 4.38
$\frac{Z+b0}{z^2+(a_1+q)*z+a_2}$	[0 to 0.001]	Fig 4.39

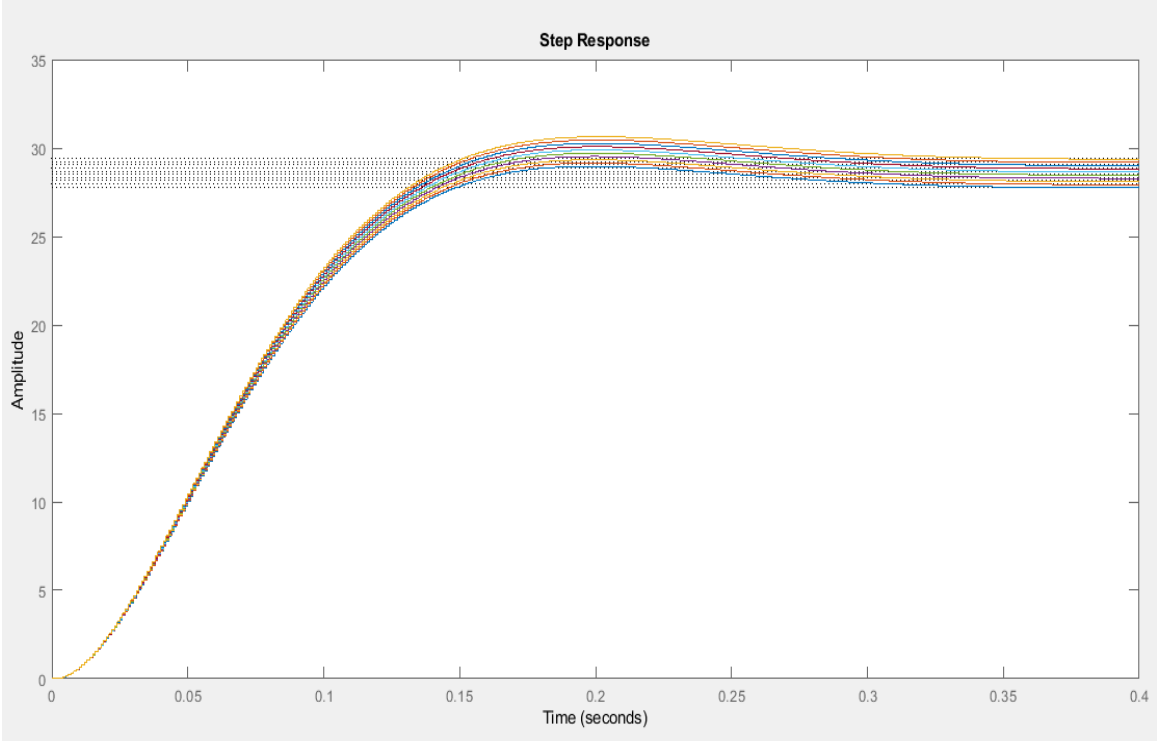

Fig 4.38: step response for different parameters

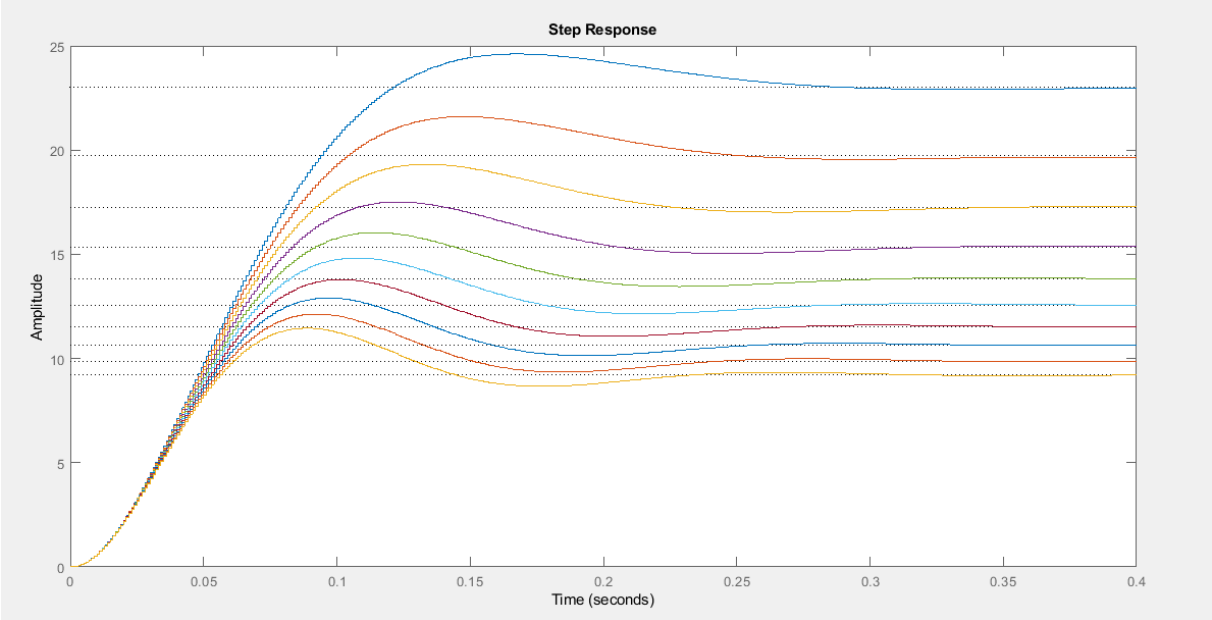

Fig 4.39: step response for different parameters

In this part, the effect of parameters variations on the closed loop response has been considered. In addition, these variations can cause stability problem in closed loop system.

To test if our controller will follow the desired behavior, we will do a simple simulation where the all system is changed to a new system totally different from the first, one condition is needed about the system is that, it must be of a second order system.

First we implement the circuit shown in the figure, **Fig4.40**, to test our controller

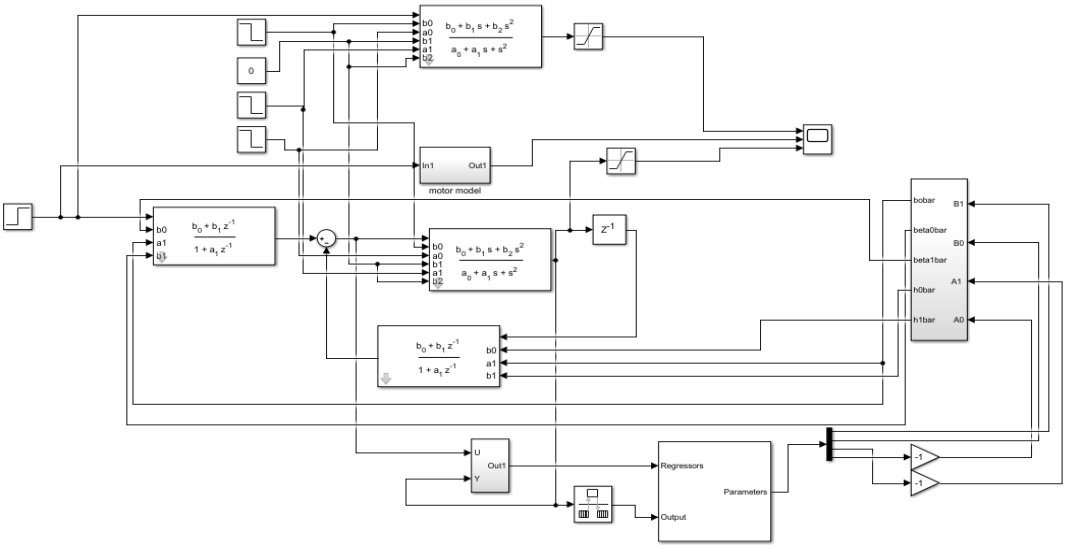

Fig 4.40: circuit of adaptive controller with system swap.

The first system is described as $G_1(s)$:

$$G_1(s) = \frac{7442}{0.0005878*s^2+5.154*s+253.4}$$

Then after a while this system will be replaced by a new system $G_2(s)$, which is as follows

$$G_1(s) = \frac{25}{s^2+7*s+25}$$

The response of that system is shown in **Fig 4.41,** as we can see in the figure the controlled system has tracked the desired speed and did not diverge.

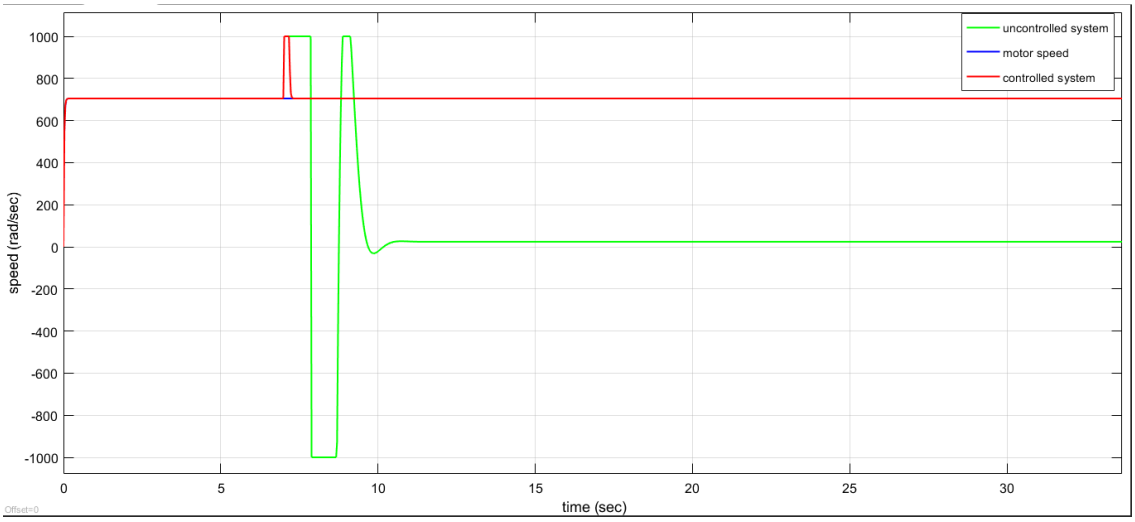

Fig 4.41: response of the adaptive system with system swap.

To see the role of the adaptive controller, we will implement a PID controller in the previous system.

The PID controller is designed to control the first system$G_1(s)$, the circuit is shown in **Fig4.42**, and the result is shown in figure **4.43.**

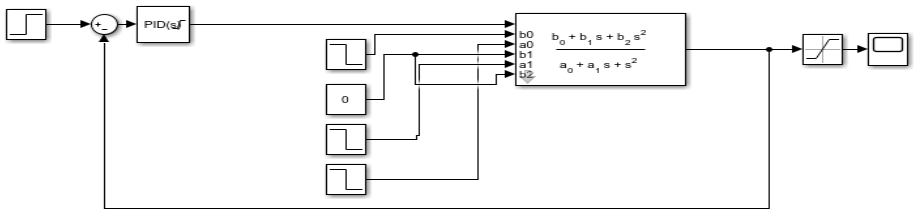

Fig 4.42: circuit with system changes.

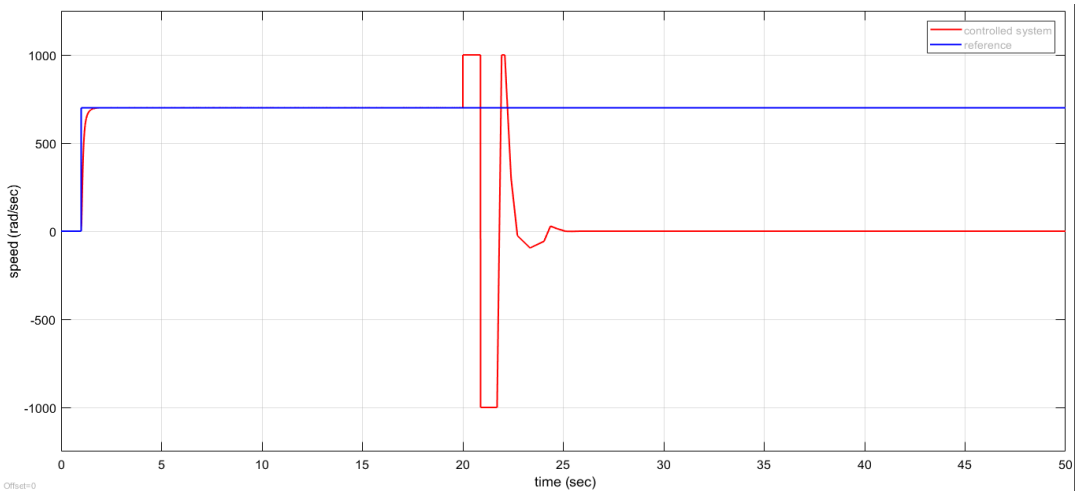

Fig 4.43: result of system swap.

As we've said in the beginning of our works, conventional controllers are used for a specified system. While any change in that system is followed by a change in controller parameters.

Conclusion

The implementation and simulation of self-tuning pole placement controller has been performed successfully.

The experimental work was performed on PMDC motor, two different methods of parameter identification have been applied using SIMULINK parameter estimation toolbox, and these two methods are the Non-linear least-squares algorithm and Pattern Search Genetic algorithm which have been successfully implemented and experimentally tested. Based on this experimental work, the Non-Linear Least Squares Algorithm gives better performance because it requires much less computation time than Genetic Algorithm.

For the experimentation implementation, a national instrument PCI 6221 37-pin data acquisition card is used. The offline parameters estimation, we've used a MATLAB code (see appendix A) for system or process I/O measurement using data acquisition. A SIMULINK model is used for the online parameter estimation (real-time) using blocks available in SIMULINK library.

Online parameter estimation was performed based on SIMULINK block (Recursive Least Squares' block). To confirm that online estimation is reliable, we have implemented a SIMULINK model that estimates DC motor's parameters, and as we have expected the model and the real system were close to each other.

After accomplishing the DC motor model estimation, the PID controller design is considered to control the speed of the motor. This PID controller was built in SIMULINK, and the data acquisition is performed using a NI-PCI card. PWM circuit was designed using external circuit; the PID controller was optimized using SIMULINK's check response characteristics block. The overall system controller has been tested and good results are obtained for the system desired motor speed response.

A step further, a pole Placement Adaptive Controller is considered. First a reference model for our controller is selected, then the controller is implemented in SIMULINK using RLSE's block as its online system model estimation. This controller system dynamic response has been carried out practically where the desired speed for both loaded and without load situations is successfully obtained. The effect of the load has

been visualized in the armature current. The Pole Placement Adaptive Controller was effectively designed implemented and tested using a desired speed profile.

To monitor the necessity of the Adaptive controller, we have illustrated the effect of changing system's parameters on the step response of the system; these parameters were changed with small values.

In order to show why we use adaptive controller instead of the very known and used controller (PID Controller), we did a simulation were the system is heavily changed; for the PID controller the result was neither acceptable nor satisfying. The Adaptive controller, in the other hand, has maintained the desired speed, right after it estimated the new system. Adaptive controller is better than non-self tuning PID controller because even for system parameters' change the desired speed is maintained.

Since most of the systems are slow-time varying process, adaptive controller is of vital importance in the control filed, parameter of the adaptive control will adapt itself according to the changes of the plant parameter.

Future Work:

Since we've successfully controlled the speed of the motor with adaptive controller, we are considering applying this controller for position control using DC motor. Also, we suggest to implement this adaptive controller (e.g. speed and position controller) using stand-alone Micro-Controller for application requiring such type of controller as robot arm control.

References

[1] practical electrical motor handbook Irving M.gottlieb PE

[2] transformer and motors George Patrick shultz

[3] Linear Stochastic Control Systems De Goong Chen, Guanrong Chen, Shih-Hsun Hsu

[4] System Identification Course EE555. Presented by Dr. Akroum

[5] System Identification and Control Using Genetic Algorithms Kristinn Kristinsson, Member, IEEE, and Guy A. Dumont, Senior Member, IEEE

[6] MATLAB & SIMULINK MATHWORKS

[7] Modern Control Engineering, fifth edition, Katsuhiko Ogata.

[8] M. M' Saad I. D. Landau, M. Dugue and M.Samaan, "Example Applications of the Partial State Model Reference Adaptive Design Technique", int. J. of Adaptive Control and Signal Processing vol.3, 1989.

[9] Brad Schofield "Subspace Based Identification for Adaptive Control", M. Sc. Thesis. Department of Automatic Control, Lund Jnstitute of Technology. June, 2003.

[10] by karl johan astrom and and Bjorn wittenmark lund institute of technology "adaptive control"

[11] N.T. Nyugen, Model Reference Adaptive Control, Moffen Field, CA: Springer International Publishing, pp.83-106,2018

[12] S. Sastry and M. Bodson, Adaptive Control: Stability, Convergence, and Robustness. New Jersey: Prentice-Hall, 1989.

[13] Fatima Tahri, Ali Tahri, Ahmed Allali and Samir Flazi, "The Digital Self-Tuning Control of Step a Down DC-DC Converter" Department of Electrotechnics,University of Sciences and Technology of Oran, Algeria. Acta Polytechnica Hungarica. Vol. 9, No. 6, 2012

[14] 18 K. J. °Astr¨om and B. Wittenmark, Computer-Controlled Systems: Theory and Design. New Jersey: Englewood Cliffs Prentice Hall, 1984

Appendix A

```
daqreset
global time
voltage_ratio=((5.53+21.75)/5.53); %% voltage divider circuit
current_ratio=(1/1.1); %%shunt resistor admitance
s=daq.createSession('ni');
addAnalogInputChannel(s,'Dev1',0,'Voltage');
addAnalogOutputChannel(s,'Dev1',0,'Voltage');
addDigitalChannel(s,'dev1','Port0/Line0','InputOnly');
addAnalogInputChannel(s,'Dev1',1,'Voltage');
Fs = 125000;
s.Rate=Fs;
run=1
outputData = linspace(10,10, 125000*2)';
queueOutputData(s,outputData);
[data,time] = startForeground(s);%%collecting data
pause(2.0);
voltage = data(:,1);
current = data(:,3);
pulses = 5*data(:,2);
voltage = voltage * voltage_ratio;%%motor real voltage value
current = current * current_ratio;%%motor real current value
fs = Fs;
duration = 0.001; %%duration for speed measurement
totalrpm = [];
totaltt = [];
orgdata = pulses;
v_threshold = 2.5;
numsamples= length(orgdata);
nol= round(numsamples/(fs*duration))-1;%%number of loops with 0.01 sec duration
for k=0:nol-1
   dt = orgdata( 1+fs*k*duration : fs*duration*(k+1) );
   offset = [dt(2:end);NaN];
   RE = find( dt < v_threshold & offset > v_threshold);
   FE = find( dt> v_threshold & offset < v_threshold);
   if isempty(time(RE))
     nop=0;
   else
     nop = length(RE);
   end
   rpm = (60/duration)*(nop/500); %%500 is the number of slits in our encoder E2-500
   totalrpm = [totalrpm rpm];
   totaltt = [totaltt k*duration];
end
%%use average filter to reduce signals noises for better identification
m=3;
n=2;
a=[1 zeros(1,length(m))];
b=(1/m)*[ones(1,m)];
voltage_avg = filter(b,a,voltage);
```

```
a=[1 zeros(1,length(m))];
b=(1/m)*[ones(1,m)];
current_avg = filter(b,a,current);
a=[1 zeros(1,length(n))];
b=(1/n)*[ones(1,n)];
totalrpm_avg=filter(b,a,totalrpm);
totalrpm_avg_sim=[0 totalrpm(21:end)];
totaltt_sim=totaltt(1:end-19);
a=[1 zeros(1,length(n))];
b=(1/n)*[ones(1,n)];
totalrpm_avg_sim=filter(b,a,totalrpm_avg_sim);
rpm_system=totalrpm_avg_sim/168.84;
%%data for simulink model
Ws = [totaltt_sim',rpm_system'];
Vs = [time , voltage_avg];
Ia = [time, current_avg];
```

Appendix B

PLANETARY GEARMOTOR OUTPUT PARAMETERS:	Value	Units	Tolerance
Gearhead Ratio (exact)	168.84:1		
Gearhead Shaft Output Speed (at full -load)	35	RPM	MAX
Gearmotor Rated Continuous Torque	148	IN-lbs	MAX
Gearmotor Rated Peak Torque	265++	IN-lbs	…….
DC MOTOR PERFORMANCE PARAMETERS:	Value	Units	tolerance
Rated DC Voltage	24	DC volts	…….
Rated Continuous Current	5.2	Amperes	…….
No-Load Speed	6670	RPM	MAX
Rated Speed	5870	RPM	+/- 15%
Rated Continuous Power Out	87	WATTS	+/- 15%
Rated Continuous Torque	20	OZ-IN	…….
Peak Torque (motor only)	130	OZ-IN	…….
No-Load Current	0.63	AMPERES	MAX
Back EMF Constant (Ke)	3.6	V/KRPM	+/- 10%
Torque Constant (Kt)	4.8	OZ-IN/AMP	+/- 10%
DC Armature Resistance	0.8	OHMS	+/- 15%