PERFORMANCE ANALYSIS OF LINEAR QUADRATIC REGULATOR (LQR) CONTROLLER FOR D.C.MOTOR

A Thesis report

submitted towards the partial fulfillment of the

requirements of the degree of

Master of Engineering

in

Electronics Instrumentation and Control Engineering

Submitted by

Piyush Chandra Ojha

Roll No-800851025

Under the supervision of

Dr. Yaduvir Singh
Associate Professor, EIED
and
Dr. Hardeep Singh
Assistant Professor, ECED

DEPARTMENT OF ELECTRICAL AND INSTRUMENTATION ENGINEERING

THAPAR UNIVERSITY

PATIALA – 147004

JULY2010

DECLARATION

I hereby declare that the report entitled ***"Performance analysis of linear quadratic regulator (LQR) control of D.C. Motor"*** *is an authentic record of my own work carried out as requirements for the award of degree of M.E. (Electronic Instrumentation and Control) at Thapar University, Patiala, under the guidance of Dr. Yaduvir Singh (Associate Professor, EIED) and Dr. Hardeep Singh (Assistant Professor) during January to July 2010.*

Date: 06-07-2010

Piyush Chandra Ojha

Roll No. - 800851025

It is certified that the above statement made by the candidate is correct to best of my knowledge and belief.

Dr. Yaduvir Singh
Associate Professor, EIED
(Supervisor)
Thapar University, Patiala

Dr .Hardeep Singh
Assistant Professor, ECED
(Co-Supervisor)
Thapar University, Patiala

S. Ghosh 12/7/10

Dr. Smarajit Ghosh
Professor & Head, EIED
Thapar University, Patiala.

13.7.10

Dr. R.K. Sharma
Dean of Academic Affairs
Thapar University, Patiala

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Control engineering is one subject which is perceived as being the most theoretical and most difficult to understand. In industries, application of motor control system is important to operation some process. An average home in Malaysia uses a dozen or more electric motors. In some application the DC motor is required to maintain its desired speed when load is applied or disturbances occur. This kind of system can be controlled using PID, Fuzzy, LQR and other more.

Direct current (DC) motors have been widely used in many industrial applications such as electric vehicles, steel rolling mills, electric cranes, and robotic manipulators due to precise, wide, simple, and continuous control characteristics. Traditionally rheostatic armature control method was widely used for the speed control of low power dc motors. However the controllability, cheapness, higher efficiency, and higher current carrying capabilities of static power converters brought a major change in the performance of electrical drives. The desired torque-speed characteristics could be achieved by the use of conventional proportional integral-derivative (PID) controllers. As PID controllers require exact mathematical modeling, the performance of the system is questionable if there is parameter variation. In recent years neural network controllers (NNC) were effectively introduced to improve the performance of nonlinear systems. The application of NNC is very promising in system identification and control due to learning ability, massive parallelism, fast adaptation, inherent approximation capability, and high degree of tolerance.

The outer speed and inner current control loops are designed as PD or PI controllers. However, the cascaded control structure assumes that the inner loop dynamics are substantially faster than the outer one. The electrical and mechanical parameters of the

DC motor, i.e., resistance, inertia, back-EMF, damping are identified from observations of the open loop response. Coulomb friction is considered as the main cause of the nonlinear motor behavior and is adequately compensated by a feed forward control signal. The residual steady state error caused by minor nonlinearities and uncertainties in the model is compensated by an integral error feedback signal. The proposed controller is evaluated for high and low velocity reference profiles including velocity reversal to demonstrate its efficiency for high-performance servo applications. The proposed scheme attempts to bridge the current gap between the advance of control theory and the practice of DC actuator systems.

In this work, Linear Quadratic Regulator (LQR) controller is used in order to control the DC motor speed as we required. This techniques is used for tracking setpoint commands and reducing sensitivity to load disturbances. MATLAB is used to design and tune the LQR controller and be simulated to mathematical model of the DC motor.

The Linear Quadratic Regulator (LQR) controller is a new method of controlling the motor. Linear Quadratic Regulator (LQR) is theory of optimal control concerned with operating a dynamic system at minimum cost.

1.2 Problem Statement

The speed of the DC motor may change due to disturbance present surrounding it. This will make the desired speed sometimes change and will be not maintain. By using LQR control algorithm, the deviation of the speed can be minimized.

1.3 Objective

The objectives of this work are:

1-To control the speed of D.C.motor using Feedback,Feedforword control.

2 -To design the LQR controller and using MATLAB and to control the speed of

D.C.motor using it.

3-To compare the performance of feedback ,feedforward and linear quadratic regulator controller .

1.4 Scope

This work actually concentrates on derivation of the mathematical model of dc servomotor and gets the value of K in LQR algorithm. K is the gain of the close loop system. To get the value of K, the state-space of the servomotor must be define first. So, LQR algorithm was used by means to minimize the deviation of the dc motor speed. The LQR is used to tune the value of Q and R. The value of Q and R is tuned the get the stable system.

1.5 Overview of thesis

The structure of this thesis is set out into five sections.

- Chapter 1 gives an introduction to this thesis and a brief description of the different areas that make up the project.
- Chapter 2 will focus on the history, construction and principles of operation of the DC motor.
- Chapter 3 deals with the various control theory related to D.C.motor
- Chapter 4 deals with basics of MATLAB
- Chapter 5 reveals the results obtained and gives an analysis of the outcome of the project, including simulation and practical data.

CHAPTER 2

BASICS OF D.C. MOTOR

2.1 General Background of Electric motor

An electric motor uses electrical energy to produce mechanical energy, very typically through the interaction of magnetic fields and current-carrying conductors. The reverse process, producing electrical energy from mechanical energy, is accomplished by a generator or dynamo. Many types of electric motors can be run as generators, and vice versa. For example a starter/generator for a gas turbine or traction motors used on vehicles often perform both tasks.

Electric motors are found in applications as diverse as industrial fans, blowers and pumps, machine tools, household appliances, power tools, and disk drives. They may be powered by direct current (e.g., a battery powered portable device or motor vehicle), or by alternating current from a central electrical distribution grid. The smallest motors may be found in electric wristwatches. Medium-size motors of highly standardized dimensions and characteristics provide convenient mechanical power for industrial uses. The very largest electric motors are used for propulsion of large ships, and for such purposes as pipeline compressors, with ratings in the millions of watts. Electric motors may be classified by the source of electric power, by their internal construction, by their application, or by the type of motion they give.

The physical principle of production of mechanical force by the interactions of an electric current and a magnetic field was known as early as 1821. Electric motors of increasing efficiency were constructed throughout the 19th century, but commercial exploitation of electric motors on a large scale required efficient electrical generators and electrical distribution networks.

Some devices, such as magnetic solenoids and loudspeakers, although they generate some mechanical power, are not generally referred to as electric motors, and are usually termed actuators and transducers, respectively.

In today's world, almost all land-based electrical power supply networks are AC systems of generation, transformation, transmission and distribution. Thus there is little need for large DC generators. Furthermore, AC motors are used in industries wherever they are suitable or can give appropriate characteristics by means of power electronic devices. Yet there remain important fields of application when the DC machines can offer economic and technical advantage. The wonderful thing about DC machines is its versatility.

A DC machine can operate as either a generator or a motor but at present its use as a generator is limited because of the widespread use of AC power. Large DC motors are used in machine tools, printing presses, conveyors, fans, pumps, hoists, cranes, paper mills, textile mills and so forth. Small DC machines (in fractional horsepower rating) are used primarily as control devices such as tacho-generators for speed sensing and servomotors for positioning and tracking.

DC motors still dominate as traction motors used in transit cars and locomotives as the torque-speed characteristics of DC motor can be varied over a wide range while retaining high efficiency. The DC machine definitely plays an important role in industry.

2.2 History of D.C.motor

Electric motors exist to convert electrical energy into mechanical energy. This is done by two interacting magnetic fields -- one stationary, and another attached to a part that can move.

DC motors have the potential for very high torque capabilities (although this is generally a function of the physical size of the motor), are easy to miniaturize, and can be

"throttled" via adjusting their supply voltage. DC motors are also not only the simplest, but the oldest electric motors.

The basic principles of electromagnetic induction were discovered in the early 1800's by Oersted, Gauss, and Faraday. In 1819, Hans Christian Oersted and Andie Marie Ampere discovered that an electric current produces a magnetic field. The next 15 years saw a flurry of cross-Atlantic experimentation and innovation, leading finally to a simple DC rotary motor. A number of men were involved in the work. Below are three of the most famous people to have experimented about DC motor.

Michael Faraday (U.K.)

Fabled experimenter Michael Faraday decided to confirm or refute a number of speculations surrounding Oersted's and Ampere's results. He set to work devising an experiment to demonstrate whether or not a current-carrying wire produced a circular magnetic field around it, and in October of 1821, he succeeded in demonstrating this.

Faraday took a dish of mercury and placed a fixed magnet in the middle. Above this, he dangled a freely moving wire (the free end of the wire was long enough to dip into the mercury). When he connected a battery to form a circuit, the current-carrying wire circled around the magnet. Faraday then reversed the setup, this time with a fixed wire and a dangling magnet. Again the free part circled around the fixed part. This was the first demonstration of the conversion of electrical energy into motion, and as a result, Faraday is often credited with the invention of the electric motor.

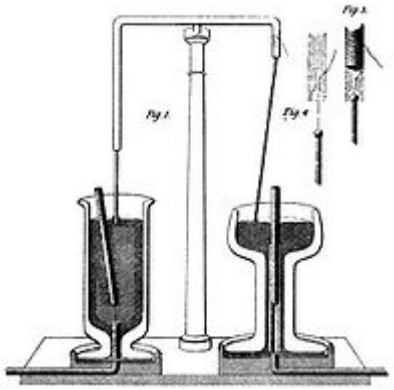

Figure 2.1: Faraday's experiment on the conversion of electrical energy into motion.

Joseph Henry (U.S.)

It took ten years, but by the summer of 1831 Joseph Henry had improved on Faraday's experimental motor. Henry built a simple device whose moving part was a straight electromagnet rocking on a horizontal axis. Its polarity was reversed automatically by its motion as pairs of wires projecting from its ends made connections alternately with two electrochemical cells. Two vertical permanent magnets alternately attracted and repelled the ends of the electromagnet, making it rock back and forth at 75 cycles per minute.

Henry considered his little machine to be merely a "philosophical toy," but nevertheless believed it was important as the first demonstration of continuous motion produced by magnetic attraction and repulsion. While being more mechanically useful than Faraday's motor, and being the first real use of electromagnets in a motor, it was still by and large a lab experiment. On the basis of his experiments, it was feasible to design both electric generators and electric motors

Figure 2.2: Joseph Henry experimental motor.

William Sturgeon (U.K.)

Just a year after Henry's motor was demonstrated, William Sturgeon invented the commutator, and with it the first rotary electric motor. In many ways, a rotary analogue of Henry's oscillating motor. Sturgeon's motor, while still simple, was the first to provide continuous rotary motion and contained essentially all the elements of a modern DC motor.

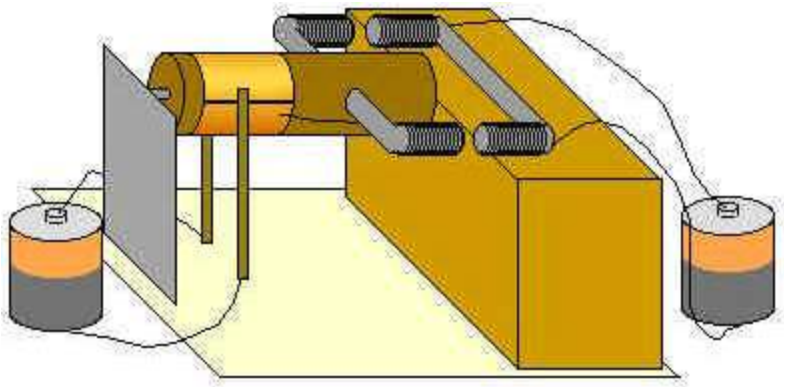

Figure 2.3: First rotary electric motor invented by William Sturgeon

Because of the work of these people, DC machines are one of the most commonly used machines for electromechanical energy conversion. Converters which are used continuously to convert electrical input to mechanical output or vice versa are called electric machines. An electric machine is therefore a link between an electrical system and a mechanical system. In these machines, the conversion is reversible. If the conversion is from mechanical to electrical, the machine is said to act as a generator. If the conversion is from electrical to mechanical, the machine is said to act as a motor. Therefore, the same electric machine can be made to operate as a generator as well as a motor.

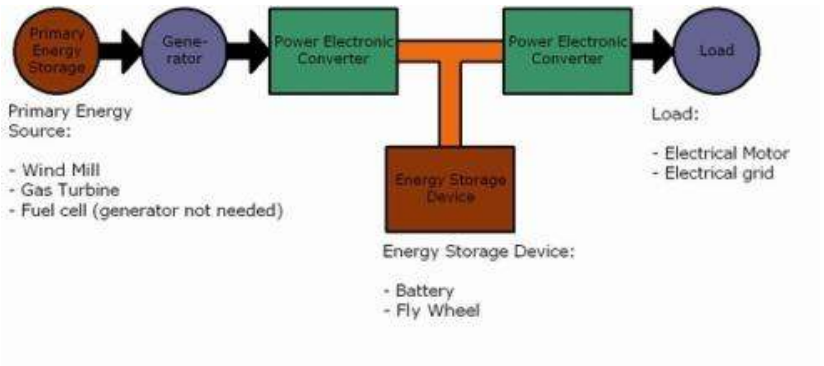

Figure 2.4: Electromechanical energy conversion

DC machines may also work as brakes. The brake mode is a generator action but with the electrical power either regenerated or dissipated within the machine system, thus developing a mechanical braking effect. It also converts some electrical or mechanical energy to heat, but this is undesired.

The major advantages of DC machines are easy speed and torque regulation. The major parts of any machine are the stationary component, the stator, and the rotating component, the rotor.

2.3 Construction of a DC motor

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

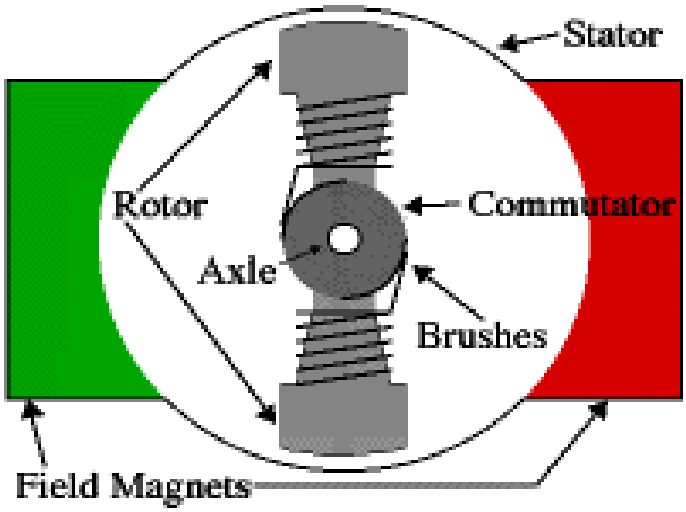

Figure 2.5: DC motor construction

The coils are connected in series. To keep the torque on a DC motor from reversing every time the coil moves through the plane perpendicular to the magnetic field, a split-ring device called a commutator is used to reverse the current at that point. The commutator shown in Figure 2.6 consists of insulated copper segments mounted in a cylinder. The electrical contacts to the rotating ring are called "brushes" since copper brush contacts were used in early motors. Modern motors normally use spring-loaded carbon contacts, but the historical name for the contacts has persisted. Two brushes are

pressed to the commutator to permit current flow. The brushes are placed in the neutral zone (magnetic field is close to zero) to reduce arcing.

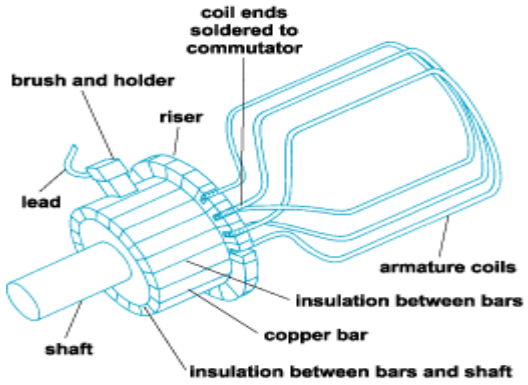

Figure 2.6: Concept of the commutator

Figure 2.7 below shows the stator of a large DC machine with several poles. The inter poles reduces the field in the neutral zone and eliminate arcing of the commutator. A compensation winding is placed on the main poles to increase field during high load. The iron core is supported by a cast iron frame.

Figure 2.7: DC motor stator construction

Figure 2.8 shows the rotor of a DC machine. The rotor iron core is mounted on the shaft. Coils are placed in the slots. The ends of the coils are bent and tied together to assure mechanical strength. The commutator mounted on the shaft consists of several copper segments, separated by insulation.

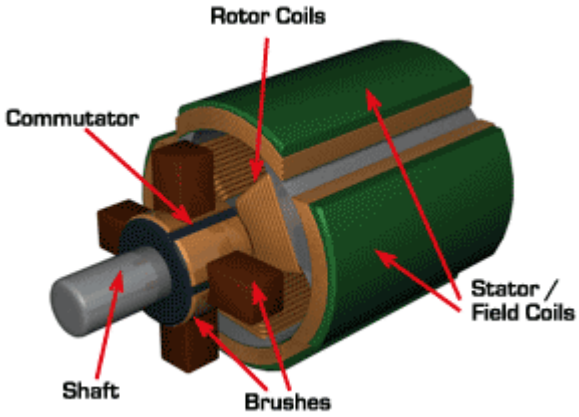

Figure 2.8: DC motor rotor construction

Figure 2.9 shows the commutator of a large DC machine. The segments are made out of copper and mica insulation and placed between the segments. The end of each segment has a flag attached. The coil endings are welded to these flags. An insulated ring is placed on the coil ends to assure proper mechanical strength.

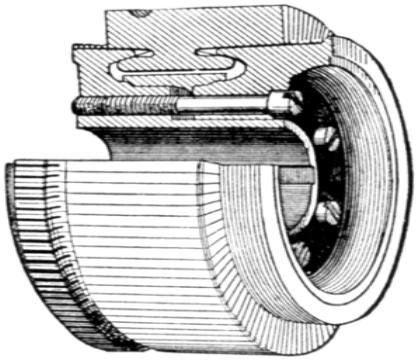

Figure 2.9: Commutator of a large DC machine

A DC motor is rarely installed in a situation where it is required to run at constant speed under constant load, since an AC induction motor perfo rms such duties satisfactorily, costs only a fraction of the price of a DC machine of equal power and speed and requires minimal maintenance.

Many simple variable-speed systems are inherently stable in operation, so that the

steady-state behaviour of a DC motor is frequently all that an engineer needs to take into consideration. For simple systems, a DC shunt motor excited from a single source is often satisfactory and provides a reasonable range of adjustable speed and torque.

2.4 Principles of Operation

In any electric motor, operation is based on simple electromagnetism. A current carrying conductor generates a magnetic field which when placed in an external magnetic field, it will experience a force proportional l to the current in the conductor and to the strength of the external magnetic field. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

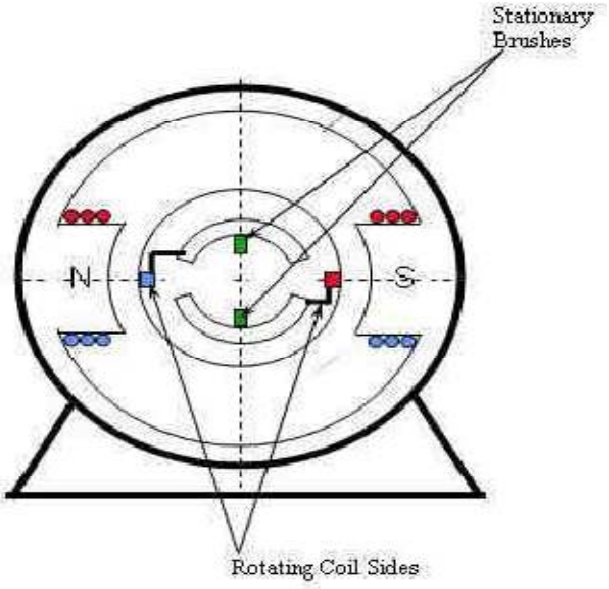

Figure 2.10: Concept of a DC motor operation

The geometry of the brushes, commutator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commutator contacts, and energize the next winding.

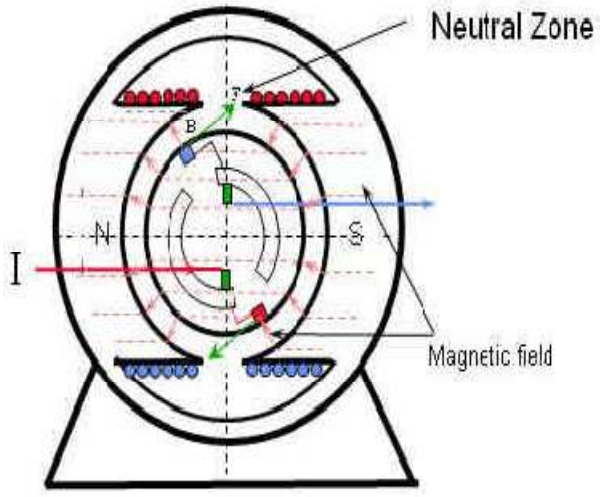

Figure 2.11: Current direction changes as the conductor passes through the neutral zone.

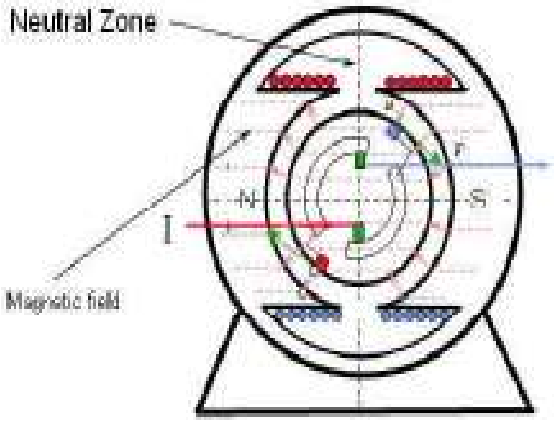

Figure 2.12: The direction of magnetic field also changes as the conductor passes through the neutral zone.

In real life, DC motors will always have more than two poles (three is a very common number). In particular, this avoids "dead spots" in the commutator. If the rotor is exactly at the middle of its rotation (perfectly aligned with the field magnets), it will get "stuck" there. Meanwhile, with a two-pole motor, there is a moment where the commutator shorts out the power supply (i.e., both brushes touch both commutator

contacts simultaneously). This would be bad for the power supply, waste energy, and damage the motor components as well. Yet another disadvantage of such a simple motor is that it would exhibit a high amount of torque "ripple" (the amount of torque it could produce is cyclic with the position of the rotor).

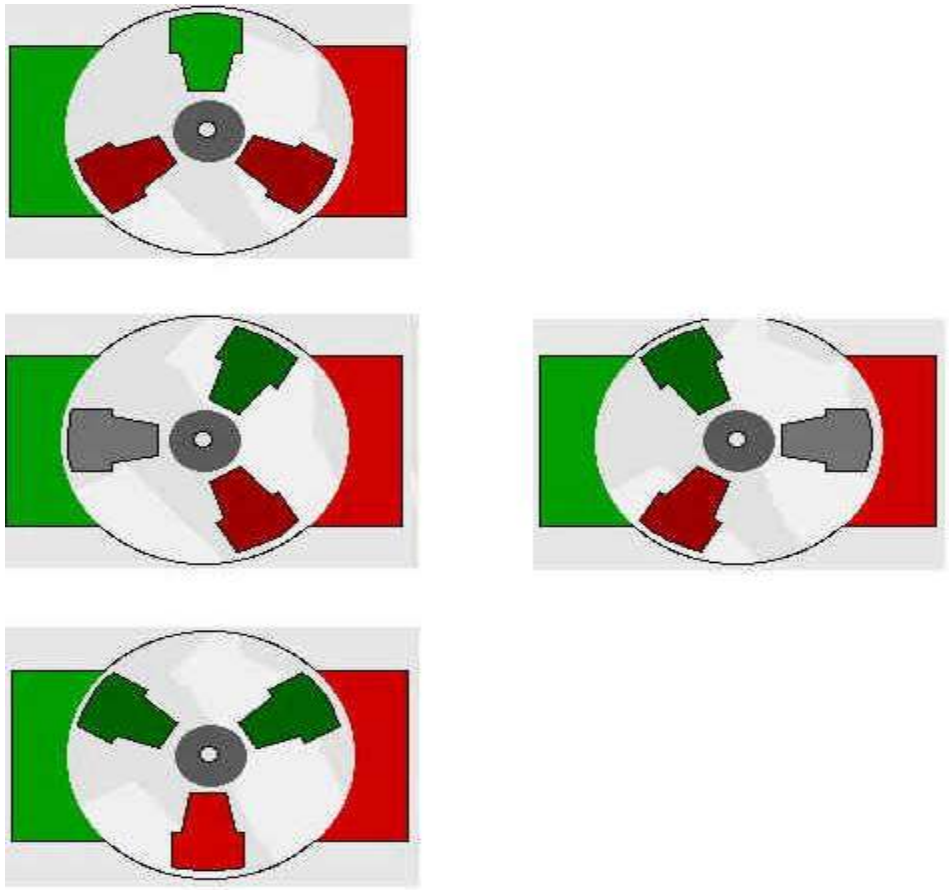

Figure 2.13: Rotor movement of a three-pole design motor

From Figure 2.13, one pole is fully energized at a time (but two others are "partially" energized). As each brush transitions from one commutator contact to the next, one coil's field will rapidly collapse, as the next coil's field will rapidly charge up (this occurs within a few microsecond).

2.5 Types of D.C. motor and their connections

The illustrations below schematically show the different methods of connecting the field and armature circuits in a DC motor. The circular symbol represents the armature circuit, and the squares at the side of the circle represent the brush commutator system. The direction of the arrows indicates the direction of the magnetic fields.

2.5.1 Externally - Excited DC Motor

This type of DC motor is constructed such that the field is not connected to the armature. This type of DC motor is not normally used.

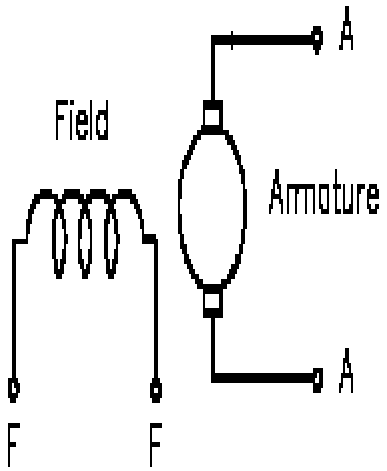

Figure 2.14: Externally - Excited DC Motor

2.5.2 Shunt DC Motor

The motor is called a "shunt" motor because the field is in parallel, or "shunts" the armature.

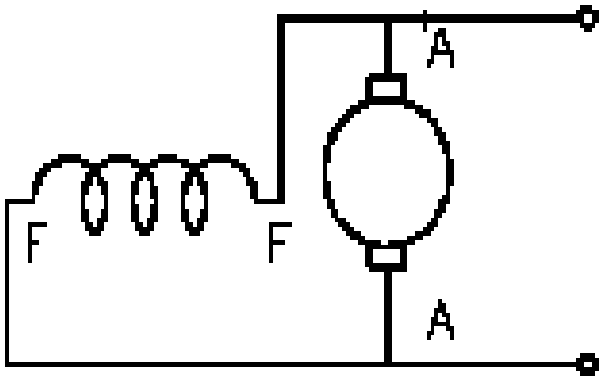

Figure 2.15: Shunt DC Motor

2.5.3 Series DC Motor

The motor field windings for a series motor are in series with the armature.

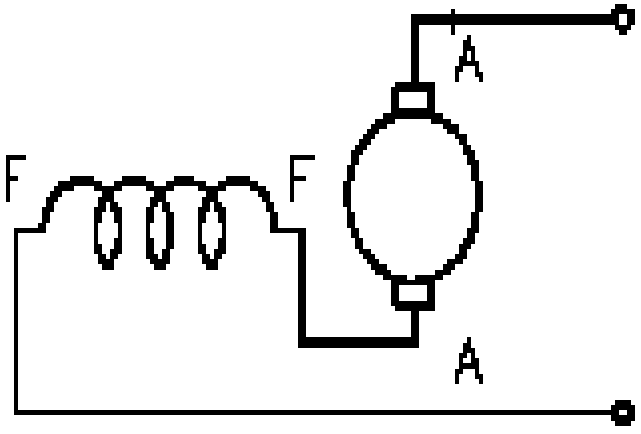

Figure 2.16: Series DC Motor

2.5.4 Compound DC Motor

A compound DC motor is constructed so that it contains both a shunt and a series field. This particular schematic shows a cumulatively-compounded" DC motor because the shunt and series fields are aiding one another.

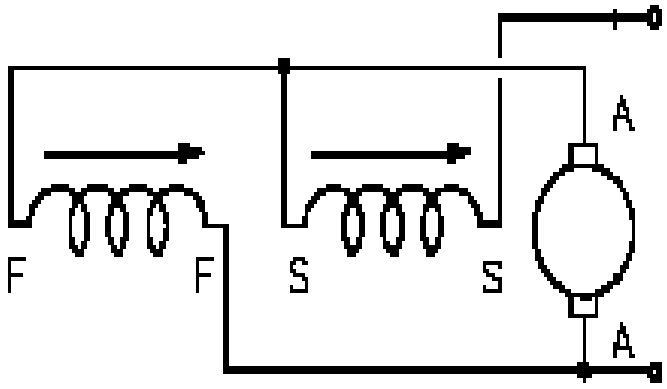

Figure 2.17:Compound DC Motor

2.5.5 Compounded DC Motor

Compound DC Motor is also called a "differentially-compounded" DC motor because the shunt and series field oppose one another.

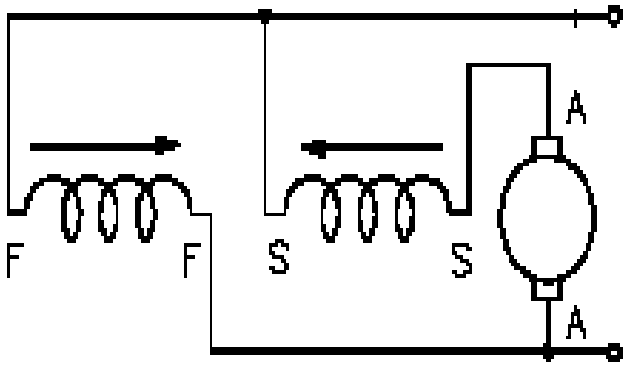

Figure 2.18: Compounded DC Motor

2.6 Mathematical Modeling of D.C.Motor

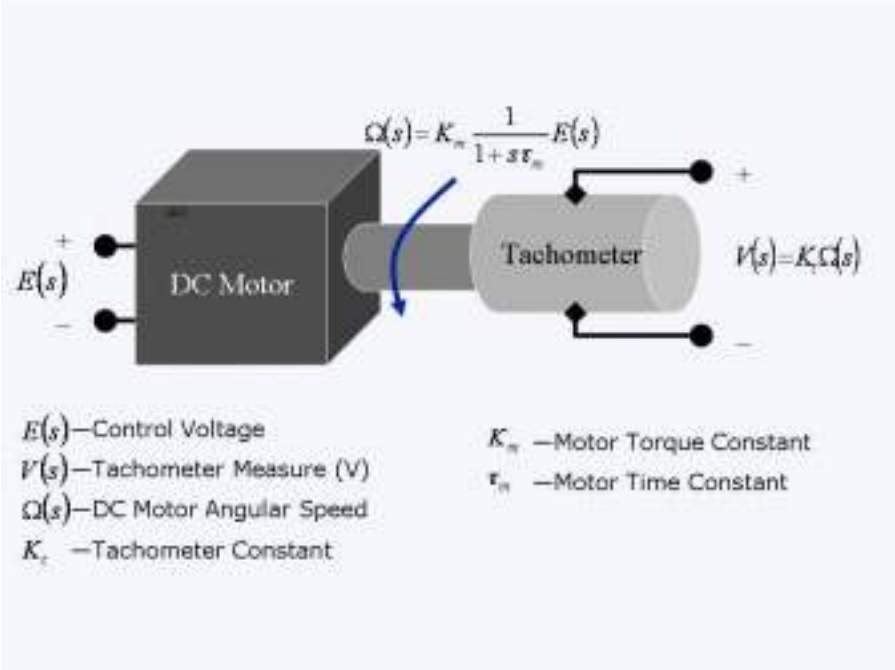

Figure 2.19: Schematic diagram of a DC motor

To design the control algorithm, first find the transfer function to develop the block diagrams of the open and close loop systems. These transfer function are obtained using the differential equation that descried the system dynamic. Kirchhoff's voltage is use to map the armature circuitry dynamic of the motor.

$$\frac{di_i}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a}\varpi_r + \frac{1}{L_a} u_a \qquad (2.1)$$

Using Newton's second law

$$\sum \vec{T} = J\vec{\alpha} = J\frac{d\vec{\omega}}{dt} \qquad (2.2)$$

The electromagnetic torque developed by the permanent-magnet DC motor is found as:

$$T_e = k_a i_a \tag{2.3}$$

The viscous friction torque

$$T_{viscous} = B_m \omega_r \tag{2.4}$$

The load torque is denoted as T_L. Use the Newton's second law, we have

$$\frac{d\omega_r}{dt} = \frac{1}{J}(T_e - T_{viscous} - T_L) = \frac{1}{J}(k_a i_a - B_m \omega_r - T_L) \tag{2.5}$$

The dynamics of the rotor angular displacement

$$\frac{d\theta_r}{dt} = \omega_r \tag{2.6}$$

To find the transfer function, the derived three first-order differential equation

$$\frac{di_i}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a} \varpi_r + \frac{1}{L_a} u_a \tag{2.7}$$

$$\frac{d\omega_r}{dt} = \frac{1}{J}(k_a i_a - B_m \omega_r - T_L) \tag{2.8}$$

and

$$\frac{d\theta_r}{dt} = \omega_r \tag{2.9}$$

Using the Laplace operator s= $\frac{d}{dv}$

$$\left(s+\frac{r_a}{L_a}\right)i_a(s)=-\frac{k_a}{L_a}\omega_r(s)+\frac{1}{L_a}u_a(s) \qquad (2.10)$$

$$\left(s+\frac{B_m}{J}\right)\omega_r(s)=\frac{1}{J}k_a i_a(s)-\frac{1}{J}T_L(s) \qquad (2.11)$$

$$s\theta_r(s)=\omega_r(s) \qquad (2.12)$$

From the transfer function, the block diagram of the permanent-magnet DC motor is illustrated by figure 2.20

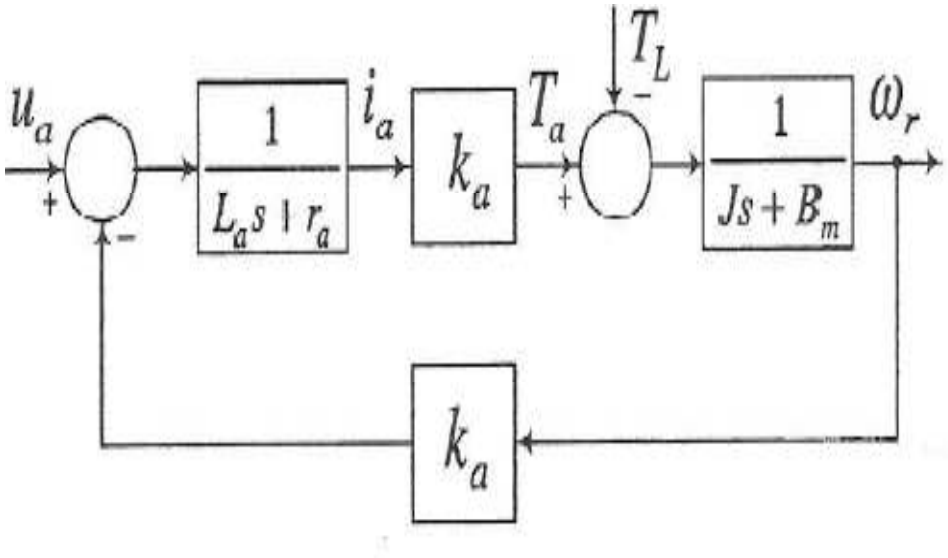

Figure 2.20: Block diagram of the open-loop DC motor

A simplified model of the DC motor is shown above. The torque Td models load disturbances. The speed variations induced by such disturbances can be minimised

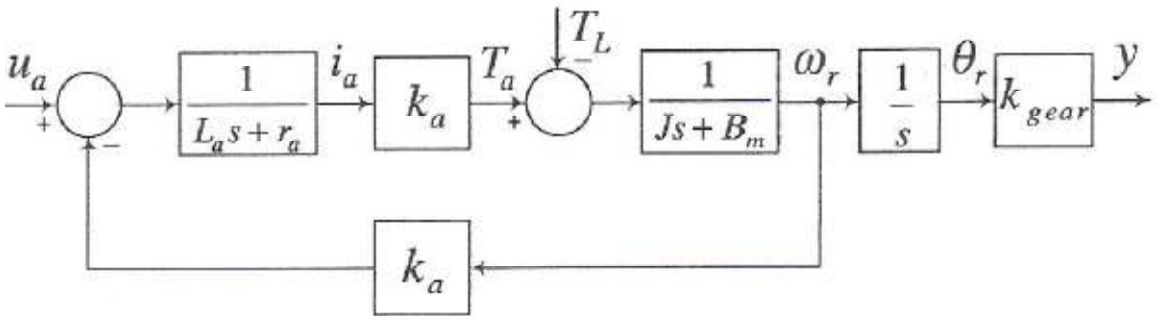

Figure 2.21: Block diagram of the open-loop servo actuated by permanent-magnet DC motor.

From equation (to find matrix)

$$\frac{di_i}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a} \varpi_r + \frac{1}{L_a} u_a \tag{2.13}$$

$$\frac{d\omega_r}{dt} = \frac{1}{J}(k_a i_a - B_m \omega_r - T_L) \tag{2.14}$$

the dynamic equations in state-space form are the following:

$$\frac{d}{dt}\begin{bmatrix} i \\ \theta \end{bmatrix} = \begin{bmatrix} -\frac{r_a}{L_a} & -\frac{k_a}{L_a} \\ \frac{k_a}{J} & -\frac{B_m}{J} \end{bmatrix} \begin{bmatrix} i \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} v \tag{2.15}$$

$$\theta = [0 \;\; 1] \begin{bmatrix} i \\ \theta \end{bmatrix} \tag{2.16}$$

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \tag{2.17}$$

$$A = \begin{bmatrix} -\frac{r_a}{L_a} & -\frac{k_a}{L_a} \\ \frac{K_a}{J} & -\frac{B_m}{J} \end{bmatrix} \qquad B = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} \tag{2.18}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad D = 0 \tag{2.19}$$

CHAPTER 3

D.C. MOTOR CONTROL THEORY

3.1 Introduction

A control system is a device or set of devices to manage, command, direct or regulate the behavior of other devices or systems.

There are two common classes of control systems, with many variations and combinations: logic or sequential controls, and feedback or linear controls. There is also fuzzy logic, which attempts to combine some of the design simplicity of logic with the utility of linear control. Some devices or systems are inherently not controllable.

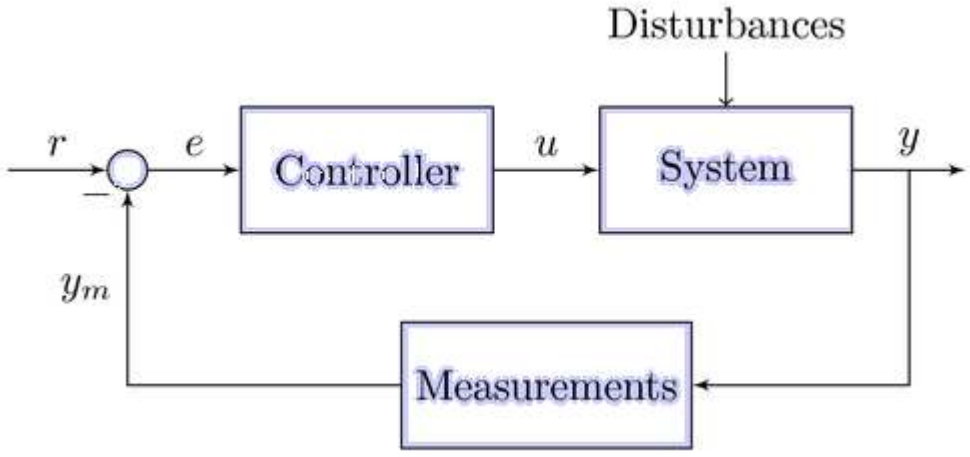

Figure 3.1: A control system

Control engineering is the engineering discipline that focuses on the modeling of a diverse range of dynamic systems (e.g. mechanical systems) and the design of controllers that will cause these systems to behave in the desired manner. Although such controllers need not be electrical many are and hence control engineering is often viewed as a subfield of electrical engineering. However, the falling price of microprocessors is making the actual implementation of a control system essentially trivial. As a result,

focus is shifting back to the mechanical engineering discipline, as intimate knowledge of the physical system being controlled is often desired.

Electrical circuits, digital signal processors and microcontrollers can all be used to implement Control systems. Control engineering has a wide range of applications from the flight and propulsion systems of commercial airliners to the cruise control present in many modern automobiles.

In most of the cases, control engineers utilize feedback when designing control systems. This is often accomplished using a PID controller system. For example, in an automobile with cruise control the vehicle's speed is continuously monitored and fed back to the system which adjusts the motor's torque accordingly. Where there is regular feedback, control theory can be used to determine how the system responds to such feedback. In practically all such systems stability is important and control theory can help ensure stability is achieved.

Although feedback is an important aspect of control engineering, control engineers may also work on the control of systems without feedback. This is known as open loop control. A classic example of open loop control is a washing machine that runs through a pre-determined cycle without the use of sensors.

3.2 Overview of control system

The term "control system" may be applied to the essentially manual controls that allow an operator to, for example, close and open a hydraulic press, where the logic requires that it cannot be moved unless safety guards are in place.

An automatic sequential control system may trigger a series of mechanical actuators in the correct sequence to perform a task. For example various electric and pneumatic transducers may fold and glue a cardboard box, fill it with product and then seal it in an automatic packaging machine.

In the case of linear feedback systems, a control loop, including sensors, control algorithms and actuators, is arranged in such a fashion as to try to regulate a variable at a setpoint or reference value. An example of this may increase the fuel supply to a furnace when a measured temperature drops. PID controllers are common and effective in cases such as this. Control systems that include some sensing of the results they are trying to achieve are making use of feedback and so can, to some extent, adapt to varying circumstances. Open-loop control systems do not directly make use of feedback, but run only in pre-arranged ways.

Modern day control engineering is a relatively new field of study that gained a significant attention during 20th century with the advancement in technology. It can be broadly defined as practical application of control theory. Control engineering has an essential role in a wide range of control systems, from simple household washing machines to high-performance F-16 fighter aircraft. It seeks to understand physical systems, using mathematical modeling, in terms of inputs, outputs and various components with different behaviors; use control systems design tools to develop controllers for those systems; and implement controllers in physical systems employing available technology. A system can be mechanical, electrical, fluid, chemical, financial and even biological, and the mathematical modeling, analysis and controller design uses control theory in one or many of the time, frequency and complex-s domains, depending on the nature of the design problem.

The term "control system" may be applied to the essentially manual controls that allow an operator to, for example, close and open a hydraulic press, where the logic requires that it cannot be moved unless safety guards are in place.

3.3 History of control system

Before it emerged as a unique discipline, control engineering was practiced as a part of mechanical engineering and control theory was studied as a part of electrical engineering, since electrical circuits can often be easily described using control theory techniques. In the very first control relationships, a current output was represented with a voltage

control input. However, not having proper technology to implement electrical control systems, designers left with the option of less efficient and slow responding mechanical systems. A very effective mechanical controller that is still widely used in some hydro plants is the governor. Later on, previous to modern power electronics, process control systems for industrial applications were devised by mechanical engineers using pneumatic and hydraulic control devices, many of which are still in use today.

Although control systems of various types date back to antiquity, a more formal analysis of the field began with a dynamics analysis of the centrifugal governor, conducted by the physicist James Clerk Maxwell in 1868 entitled On Governors. This described and analyzed the phenomenon of "hunting", in which lags in the system can lead to overcompensation and unstable behavior. This generated a flurry of interest in the topic, during which Maxwell's classmate Edward John Routh generalized the results of Maxwell for the general class of linear systems.Independently, Adolf Hurwitz analyzed system stability using differential equations in 1877. This result is called the Routh-Hurwitz theorem.

A notable application of dynamic control was in the area of manned flight. The Wright Brothers made their first successful test flights on December 17, 1903 and were distinguished by their ability to control their flights for substantial periods (more so than the ability to produce lift from an airfoil, which was known). Control of the airplane was necessary for safe flight.

By World War II, control theory was an important part of fire-control systems, guidance systems and electronics. The Space Race also depended on accurate spacecraft control. However, control theory also saw an increasing use in fields such as economics.

3.4 Examples of control system

In the furnace example, suppose the temperature is increasing towards a set point at which, say, 50% of the available power will be required for steady-state. At low temperatures, 100% of available power is applied. When the MV is within, say 10° of the SP the heat input begins to be reduced by the proportional controller. (Note that this

implies a 20° "proportional band" (PB) from full to no power input, evenly spread around the setpoint value). At the setpoint the controller will be applying 50% power as required, but stray stored heat within the heater sub-system and in the walls of the furnace will keep the measured temperature rising beyond what is required. At 10° above SP, we reach the top of the proportional band (PB) and no power is applied, but the temperature may continue to rise even further before beginning to fall back. Eventually as the MV falls back into the PB, heat is applied again, but now the heater and the furnace walls are too cool and the temperature falls too low before its fall is arrested, so that the oscillations continue.

The temperature oscillations that an under-damped furnace control system produces are unacceptable for many reasons, including the waste of fuel and time (each oscillation cycle may take many minutes), as well as the likelihood of seriously overheating both the furnace and its contents.

Suppose that the gain of the control system is reduced drastically and it is restarted. As the temperature approaches, say 30° below SP (60° proportional band or PB now), the heat input begins to be reduced, the rate of heating of the furnace has time to slow and, as the heat is still further reduced, it eventually is brought up to set point, just as 50% power input is reached and the furnace is operating as required. There was some wasted time while the furnace crept to its final temperature using only 52% then 51% of available power, but at least no harm was done. By carefully increasing the gain (i.e. reducing the width of the PB) this over-damped and sluggish behavior can be improved until the system is critically damped for this SP temperature. Doing this is known as 'tuning' the control system. A well-tuned proportional furnace temperature control system will usually be more effective than on-off control, but will still respond slower than the furnace could under skillful manual control.

An open-loop controller, also called a non-feedback controller, is a type of controller which computes its input into a system using only the current state and its model of the system.

A characteristic of the open-loop controller is that it does not use feedback to determine if its output has achieved the desired goal of the input. This means that the system does not observe the output of the processes that it is controlling. Consequently, a true open-loop system can not engage in machine learning and also cannot correct any errors that it could make. It also may not compensate for disturbances in the system.

For example, an irrigation sprinkler system, programmed to turn on at set times could be an example of an open-loop system if it does not measure soil moisture as a form of feedback. Even if rain is pouring down on the lawn, the sprinkler system would activate on schedule, wasting water.

Open-loop control is useful for well-defined systems where the relationship between input and the resultant state can be modeled by a mathematical formula. For example determining the voltage to be fed to an electric motor that drives a constant load, in order to achieve a desired speed would be a good application of open-loop control. If the load were not predictable, on the other hand, the motor's speed might vary as a function of the load as well as of the voltage, and an open-loop controller would therefore be insufficient to ensure repeatable control of the velocity.

An example of this is a conveyor system that is required to travel at a constant speed. For a constant voltage, the conveyor will move at a different speed depending on the load on the motor (represented here by the weight of objects on the conveyor). In order for the conveyor to run at a constant speed, the voltage of the motor must be adjusted depending on the load. In this case, a closed-loop control system would be necessary.

An open-loop controller is often used in simple processes because of its simplicity and low-cost, especially in systems where feedback is not critical. A typical example would be a conventional washing machine, for which the length of machine wash time is entirely dependent on the judgment and estimation of the human operator. Generally, to obtain a more accurate or more adaptive control, it is necessary to feed the output of the system back to the inputs of the controller. This type of system is called a closed-loop system.

In a closed-loop control system, a sensor monitors the output (the vehicle's speed) and feeds the data to a computer which continuously adjusts the control input (the throttle) as necessary to keep the control error to a minimum (that is, to maintain the desired speed). Feedback on how the system is actually performing allows the controller (vehicle's on board computer) to dynamically compensate for disturbances to the system, such as changes in slope of the ground or wind speed. An ideal feedback control system cancels out all errors, effectively mitigating the effects of any forces that might or might not arise during operation and producing a response in the system that perfectly matches the user's wishes. In reality, this cannot be achieved due to measurement errors in the sensors, delays in the controller, and imperfections in the control input.

3.5 Types of control theory

There are two major divisions in control theory, namely, classical and modern, which have direct implications over the control engineering applications. The scope of classical control theory is limited to single-input and single-output (SISO) system design. The system analysis is carried out in time domain using differential equations, in complex-s domain with Laplace transform or in frequency domain by transforming from the complex-s domain. All systems are assumed to be second order and single variable, and higher-order system responses and multivariable effects are ignored. A controller designed using classical theory usually requires on-site tuning due to design approximations. Yet, due to easier physical implementation of classical controller designs as compared to systems designed using modern control theory, these controllers are preferred in most industrial applications. The most common controllers designed using classical control theories are PID controllers.

In contrast, modern control theory is carried out strictly in the complex-s or the frequency domain, and can deal with multi-input and multi-output (MIMO) systems. This overcomes the limitations of classical control theory in more sophisticated design problems, such as fighter aircraft control. In modern design, a system is represented as a set of first order differential equations defined using state variables. Nonlinear, multivariable, adaptive and robust control theories come under this division. Being fairly

new, modern control theory has many areas yet to be explored. Scholars like Rudolf E. Kalman and Aleksandr Lyapunov are well-known among the people who have shaped modern control theory.

3.5.1 Classic control theory

To avoid the problems of the open-loop controller, control theory introduces feedback. A closed-loop controller uses feedback to control states or outputs of a dynamical system. Its name comes from the information path in the system: process inputs (e.g. voltage applied to an electric motor) have an effect on the process outputs (e.g. velocity or torque of the motor), which is measured with sensors and processed by the controller; the result (the control signal) is used as input to the process, closing the loop.

Closed-loop controllers have the following advantages over open-loop controllers:

1. disturbance rejection (such as unmeasured friction in a motor)
2. guaranteed performance even with model uncertainties, when the model structure does not match perfectly the real process and the model parameters are not exact
3. unstable processes can be stabilized
4. reduced sensitivity to parameter variations
5. improved reference tracking performance

In some systems, closed-loop and open-loop control are used simultaneously. In such systems, the open-loop control is termed feed-forward and serves to further improve reference tracking performance.

A common closed-loop controller architecture is the PID controller. A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems – a PID is the most commonly used feedback controller. A PID controller calculates an "error" value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process control inputs. In the absence of knowledge of the underlying process, PID controllers are the best controllers.However,

for best performance, the PID parameters used in the calculation must be tuned according to the nature of the system – while the design is generic, the parameters depend on the specific system.

The PID controller calculation (algorithm) involves three separate parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element. Heuristically, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.

By tuning the three constants in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set-point and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two modes to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral value may prevent the system from reaching its target value due to the control action.

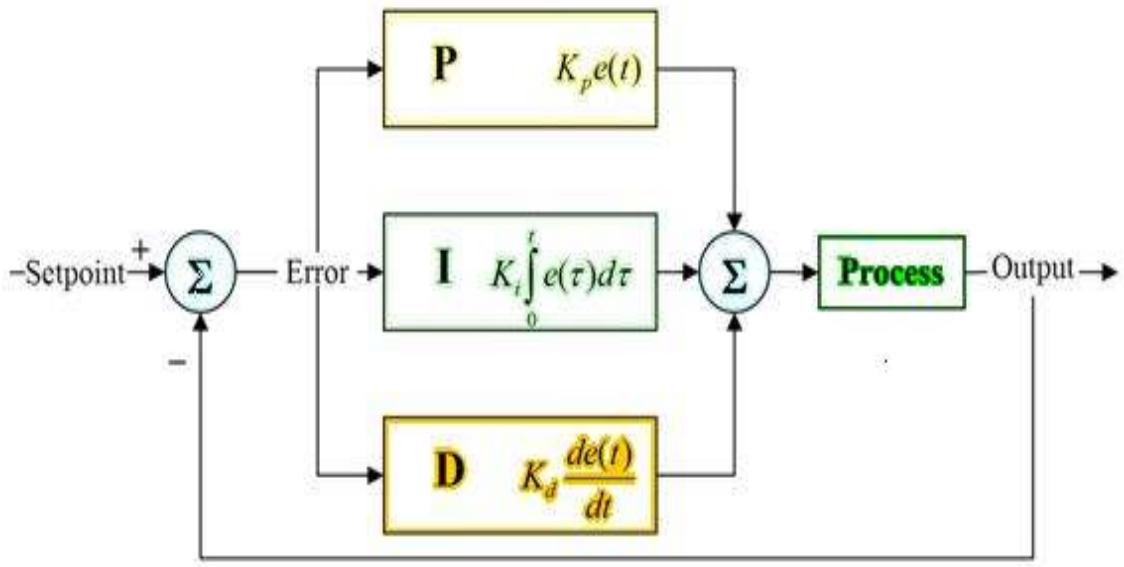

Figure 3.2: A PID controller

The classical control theory considers linear time-invariant (LTI) systems with single input and single output (SISO). In continuous-time, constant coefficient linear differential equation for signal *y(t)* with input *u(t)* is

$$\sum_{i=0}^{n} a_i \frac{d^i y(t)}{dt^i} = \sum_{j=0}^{m} b_j \frac{d^j u(t)}{dt^j} \qquad (3.1)$$

The analysis is based on Laplace transforms of the differential equations and their transfer functions.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i} \qquad (3.2)$$

3.5.2 State space concept

A system can be depicted by a black box with a number of accessible terminals, as shown in Figure 3.3.The input terminals represent a set of input variables, u_i; the output terminals describe a set of output variables, or response ,y_j . The state variables, x_k, are

embedded inside the box and are thus inaccessible. If the system is describable by linear differential equation, the state equation and output equation of the system can be represented in matrix notation as:

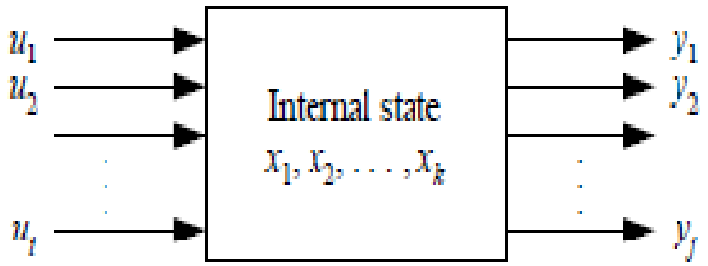

Figure3.3. State variable representation of a system

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdot & \cdot & b_{1r} \\ b_{21} & b_{22} & \cdot & \cdot & b_{2r} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ b_{n1} & b_{n2} & \cdot & \cdot & b_{nr} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \cdot \\ \cdot \\ u_r(t) \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ \cdot \\ \cdot \\ y_m(t) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdot & \cdot & c_{1n} \\ c_{21} & c_{22} & \cdot & \cdot & c_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{m1} & c_{m2} & \cdot & \cdot & c_{mn} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} & \cdot & \cdot & d_{1r} \\ d_{21} & d_{22} & \cdot & \cdot & d_{2r} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ d_{m1} & d_{m2} & \cdot & \cdot & d_{mr} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \cdot \\ \cdot \\ u_r(t) \end{bmatrix} \tag{3.4}$$

which can be simplified to the general form of state-space representation as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{3.5}$$

where A(t) is the system matrix, B(t) is the input matrix,C(t) is the output matrix and D(t) is the feed-forward matrix of the system.

3.5.3 Modern control theory

A state-space representation consists of Equation (3.4), the simultaneous, first-order differential equations from which the state variables can be solved and Equation (3.5), the algebra output equation from which all other system variables can be found . State-space representation is not unique since a different choice of state variables leads to different representation of the same system. The transformation of the similar systems is done by manipulating the transfer function, drawing signal-flow graph and then writing the state equation from signal flow graph.

Similarity transformation can be done without using the transfer function and signal-flow graph but only transformation matrix P. Transformation matrix can convert a state vector x in x_1,x_2 plane to state vector z in z1z2 plane. To obtain the transformation matrix, Eigen values and eigenvectors of the system have to be studied. The solution of a state-space representation can be obtained in the time domain by solving the corresponding matrix differential equation directly using transition matrix Φ(t).

Normally, Φ(t) can be found using Sylvester's interpolation formula. State equation can be obtained from the equation

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(t-\tau)\mathbf{B}\mathbf{u}(\tau)d\tau \qquad (3.6)$$

The time domain method, expressed in terms of state variables, can be utilized to design a suitable compensation scheme for a control system. Typically ,the system with a control signal, u(t), which is a function of several measurable state variables is controlled. The controller design concept is based on achieving a desired closed-loop state equation or a desired control ratio.

To control the location of all closed-loop poles, state feedback method is introduced. However, state feedback design often cannot be realize directly since it requires that all elements of the state vector to be available for measurement. Therefore, it is important to

find an approximation of the state variables, $\hat{x}(t) = x(t)$ .This can be done by constructing another scheme, called the observer or estimator, connected to the system under consideration, whose role is to produce good estimates of the state-space variable. The main purpose of designing controller and observer is to find their gains.

Controller and observer design only manage to obtain the desired characteristic of transient response but not zero steady state error. Steady state error is also a significant study in modern control system. Steady state analysis via input substitution can be used for multiple input multiple-output systems.

3.6 Feed-forward control

Feed-forward is a term describing an element or pathway within a control system which passes a controlling signal from a source in the control system's external environment, often a command signal from an external operator, to a load elsewhere in its external environment. A control system which has only feed-forward behavior responds to its control signal in a pre-defined way without responding to how the load reacts; it is in contrast with a system that also has feedback, which adjusts the output to take account of how it affects the load, and how the load itself may vary unpredictably; the load is considered to belong to the external environment of the system.

Some prerequisites are needed for control scheme to be reliable by pure feed-forward without feedback: the external command or controlling signal must be available, and the effect of the output of the system on the load should be known (that usually means that the load must be predictably unchanging with time). Sometimes pure feed-forward control without feedback is called 'ballistic', because once a control signal has been sent, it cannot be further adjusted; any corrective adjustment must be by way of a new control signal. In contrast 'cruise control' adjusts the output in response to the load that it encounters, by a feedback mechanism.

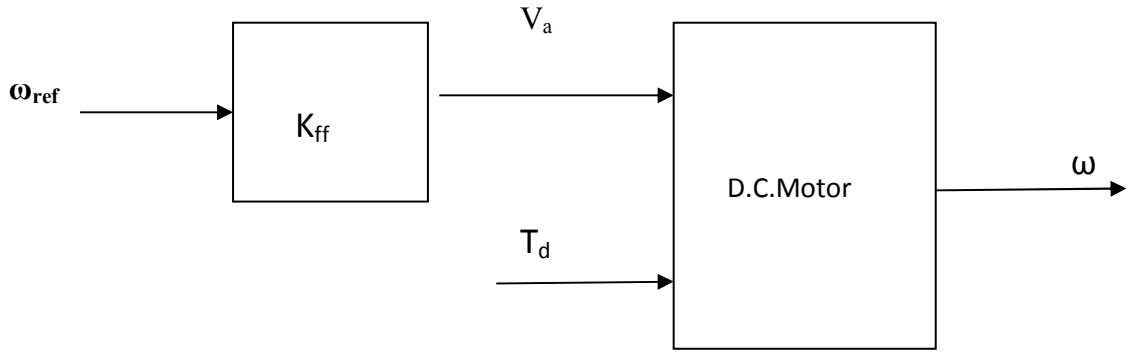

Figure 3.4 Feedforward control of D.C.motor

3.7 Feedback control

Feedback describes the situation when output from (or information about the result of) an event or phenomenon in the past will influence an occurrence or occurrences of the same (i.e. same defined) event / phenomenon (or the continuation / development of the original phenomenon) in the present or future. When an event is part of a chain of cause-and-effect that forms a circuit or loop, then the event is said to "feedback" into itself.

Feedback is a mechanism, process or signal that is looped back to control a system within itself. Such a loop is called a feedback loop. In systems containing an input and output, feeding back part of the output so as to increase the input is positive feedback; feeding back part of the output in such a way as to partially oppose the input is negative feedback.

In more general terms, a control system has input from an external signal source and output to an external load; this defines a natural sense (or direction) or path of propagation of signal; the feedforward sense or path describes the signal propagation from input to output; feedback describes signal propagation in the reverse sense. When a sample of the output of the system is fed back, in the reverse sense, by a distinct feedback path into the interior of the system, to contribute to the input of one of its internal feedforward components, especially an active device or a substance that is consumed in

an irreversible reaction, it is called the "feedback". The propagation of the signal around the feedback loop takes a finite time because it is causal.

The natural sense of feedforward is defined chemically by some irreversible reaction, or electronically by an active circuit element that has access to an auxiliary power supply, so as to be able to provide power gain to amplify the signal as it propagates from input to output. For example, an amplifier can use power from its controlled power reservoir, such as its battery, to provide power gain to amplify the signal; but the reverse is not possible: the signal cannot provide power to re-charge the battery of the amplifier.

Feedforward, feedback and regulation are self related. The feedforward carries the signal from source to load.

Negative feedback helps to maintain stability in a system in spite of external changes. It is related to homeostasis. For example, in a population of foxes (predators) and rabbits (prey), an increase in the number of foxes will cause a reduction in the number of rabbits; the smaller rabbit population will sustain fewer foxes, and the fox population will fall back. In an electronic amplifier feeding back a negative copy of the output to the input will tend to cancel distortion, making the output a more accurate replica of the input signal.

Positive feedback amplifies possibilities of divergences (evolution, change of goals); it is the condition to change, evolution, growth; it gives the system the ability to access new points of equilibrium.

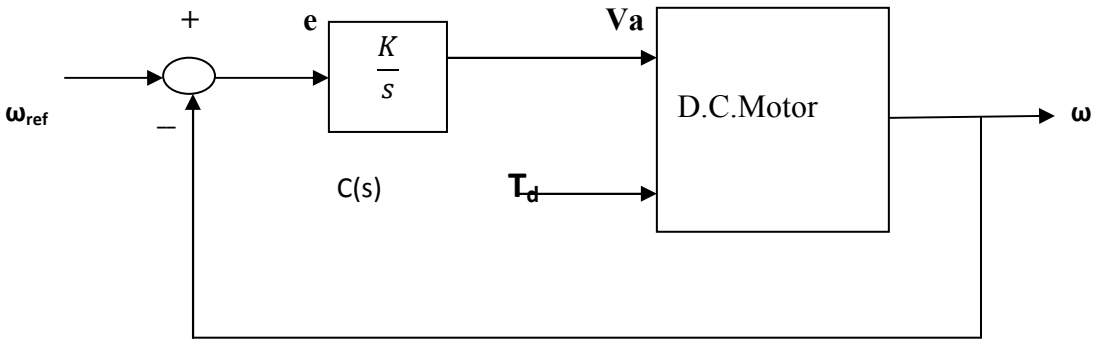

Figure 3.5 Feedback control of D.C. Motor

3.8 Linear-quadratic regulator

The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic functional is called the LQ problem. One of the main results in the theory is that the solution is provided by the linear-quadratic regulator (LQR), a feedback controller whose equations are given below.

3.8.1 General description of LQR

In layman's terms this means that the settings of a (regulating) controller governing either a machine or process (like an airplane or chemical reactor) are found by using a mathematical algorithm that minimizes a cost function with weighting factors supplied by a human (engineer). The "cost" (function) is often defined as a sum of the deviations of key measurements from their desired values. In effect this algorithm therefore finds those controller settings that minimize the undesired deviations, like deviations from desired altitude or process temperature. Often the magnitude of the control action itself is included in this sum as to keep the energy expended by the control action itself limited.

In effect, the LQR algorithm takes care of the tedious work done by the control systems engineer in optimizing the controller. However, the engineer still needs to specify the weighting factors and compare the results with the specified design goals. Often this means that controller synthesis will still be an iterative process where the engineer judges the produced "optimal" controllers through simulation and then adjusts the weighting factors to get a controller more in line with the specified design goals.

The LQR algorithm is, at its core, just an automated way of finding an appropriate state-feedback controller. And as such it is not uncommon to find that control engineers prefer alternative methods like full state feedback (also known as pole placement) to find a controller over the use of the LQR algorithm. With these the engineer has a much clearer linkage between adjusted parameters and the resulting changes in controller behaviour. Difficulty in finding the right weighting factors limits the application of the LQR based controller synthesis.

The linear quadratic regulator (LQR) is a well-known design technique that provides practical feedback gains. For the derivation of the linear quadratic regulator, assume that the plant to be written in state-space form as:

$$a^{'} = Ax + Bu \tag{3.7}$$

and that all of the **n** states **x** are available for the controller. The feedback gain is a matrix **K** of the optimal control vector

$$u(t) = -Kx(t) \tag{3.8}$$

so as to minimize the performance index

$$J = \int_0^\infty (x \cdot Qx + u \cdot Ru)dt \tag{3.9}$$

where Q is a positive-definite (or positive-semidefinite) Hermitian or real symmetric matrix and R is a positive-definite Hermitian or real symmetric matrix. Note that the second term on the right-hand side of the Equation 3.9 accounts for the expenditure of the energy of the control signals. The matrices Q and R determine the relative importance of the error and the expenditure of this energy. In this problem, assume that the control vector *u(t)* is unconstrained.

As will be seen later, the linear control law given by Equation 3.8 is the optimal control law. Therefore, if the unknown elements of the matrix K are determined so as to minimize performance index, then *u(t)=-Kx(t)* is optimal for any initial state x(0). The block diagram showing the optimal configuration is shown in Figure below:

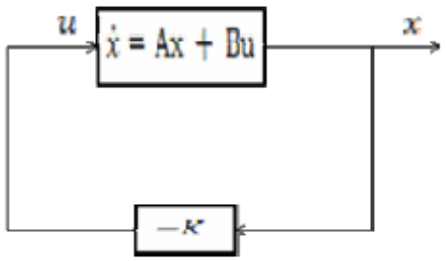

Figure 3.6: Optimal Regulator System

Now let solve the optimization problem. Substituting Equation (3.8) into Equation (3.7)

$$x = Ax - BKx' = (A - BK)x \tag{3.10}$$

In the following derivations, assume that the matrix *A-BK* is stable, or that the eigenvalues *A-BK* of have negative real parts.

Substituting Equation 3.8 into Equation 3.9 yields:

$$J = \int_0^\infty (x \cdot Qx + x \cdot K \cdot RKx)dt$$
$$= \int_0^\infty (x(\cdot Q + K \cdot RK)x)dt \tag{3.11}$$

Let set,

$$x \cdot (Q + K \cdot RK)x = -\frac{d}{dt(x \cdot Px)} \tag{3.12}$$

Where P is a positive-definite Hermitian or real symmetric matrix. Then obtain

$$x \cdot (Q + K \cdot RK)x = -x \cdot Px - x \cdot Px = -x\left[(A - BK) \cdot P + P(A - BK)\right]x \tag{3.13}$$

Comparing both sides of this last equation and noting that this equation must hold true for any x, it require that

$$(A - BK) \cdot P + P(A - BK) = -(Q + K \cdot RK) \tag{3.14}$$

It can be proved that if *A-BK* is a stable matrix, there exists a positive-definite matrix P that satisfies Equation 3.14. Hence the next procedure is to determine the elements of P from Equation 3.14 and see if it is positive definite. (Note that more than one matrix P may satisfy this equation. If the system is stable, there always exists one-positive matrix P to satisfy this equation. This means that, if to solve this equation and find one positive-definite matrix P, the system is stable. Other P matrices that satisfy this equation are not positive definite and must be discarded.)

The performance index J can be evaluated as

$$J = \int_0^\infty (x(\cdot Q + K \cdot RK)x)dt = -x \cdot Px \tag{3.15}$$

Since all eigen values of *A-BK* are assumed to have negative real parts, it have x(∞) → 0

Therefore, J can be obtain

$$J = x \cdot (0)Px(0) \tag{3.16}$$

Thus, the performance index J can be obtained in terms of the initial condition x(0) and P. To obtain the solution to the quadratic optimal control problem, proceed as follows: Since R has been assumed to be a positive-definite Hermitian or real symmetric matrix, it can be write

$$R=T.T \tag{3.17}$$

Where T is a nonsingular matrix. Then Equation 3.14 can be written as

$$(A-BK)\cdot P + (A-BK) + Q + K \cdot T \cdot TK = 0 \tag{3.18}$$

Which can be rewritten as:

$$A \cdot P + PA + \left[TK - (T\cdot)^{-1}B \cdot P\right] \cdot \left[TK - (T \cdot 0^{-1}B \cdot P\right] - PBR^{-1}B \cdot P + Q = 0 \tag{3.19}$$

The minimization of J with respect to K requires the minimization of

$$x\left[TK-(T\cdot)^{-1}B\cdot P\right]\cdot\left[TK-(T\cdot)^{-1}B\cdot P\right]x \tag{3.20}$$

with respect to K. Since this last expression is nonnegative, the minimum occurs when it is zero, or when

$$TK = (T\cdot)^{-1}B \cdot P \tag{3.21}$$

Hence,

$$K = T^{-1}(T\cdot)^{-1}B \cdot P = R^{-1}B \cdot P \tag{3.22}$$

Equation 3.22 gives optimal matrix K. Thus, the optimal control law to the quadratic optimal control problem when the performance index is given by Equation 3.9 is linear and is given by

$$u(t) = -Kx(t) = R^{-1}B \cdot Px(t) \tag{3.23}$$

The matrix P in Equation 3.22 must satisfy Equation 3.10 or the following reduced equation:

$$A \cdot P + PA - PBR^{-1}B \cdot P + Q = 0 \tag{3.24}$$

Equation 3.24 is called the reduced-matrix Riccati equation. The design steps may be stated as follows:

1. Solve Equation 3.24, the reduced-matrix Riccati equation, for the matrix P.
2. [If a positive-definite matrix P exists (certain systems may not have a positive-definite matrix P), the system is stable, or matrix *A-BK* is stable.]
3. Substitute this matrix P into Equation 3.22. The resulting matrix K is the optimal matrix.

Note that if the matrix *A-BK* is stable, the present method always gives the correct result.Finally, note that if the performance index is given in terms of the output vector rather than the state vector, that is

$$J = \int_0^\infty y \cdot Qy + u \cdot Ru)dt \tag{3.25}$$

Then, the index can be modified by using the output equation

$$y = Cx \tag{3.26}$$

To,

$$J = \int_0^{\infty} (x \cdot C \cdot QCx + u \cdot Ru)dt \tag{3.27}$$

and the design steps presented in this part can be applied to obtain optimal matrix K.

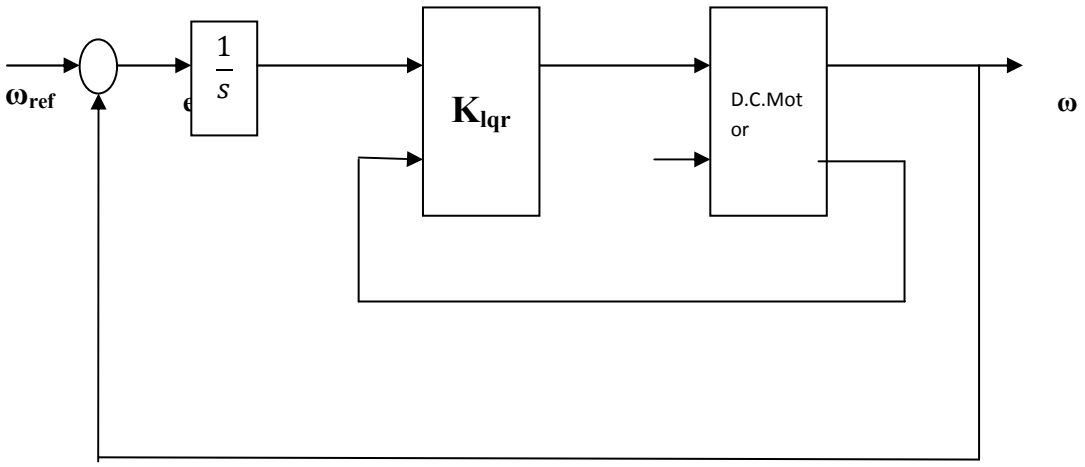

Figure 3.7: Linear Quadratic regulator control of D.C. Motor

CHAPTER 4

THEORY OF MATLAB

4.1 Introduction

MATLAB stands for "MATrix LABoratory" and is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems. One of the many advantages of
MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra course. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations.

In 2004, MathWorks claimed that MATLAB was used by more than one million people across the industry and the academic world.MATLAB users come from various backgrounds of engineering, science, and economics. Among these users are academic and research institutions such as Massachusetts Institute of Technology, Georgia Institute of Technology, NASA, Max Planck Society, and RWTH Aachen University as well as

industrial enterprises such as ABB Group, Boeing, Ford Motor, Halliburton, Lockheed Martin, Motorola, Novartis, Pfizer, Philips, Toyota, and UniCredit Bank.

4.2 History of MATLAB

MATLAB was created in the late 1970s by Cleve Moler, then chairman of the computer science department at the University of New Mexico. He designed it to give his students access to LINPACK and EISPACK without having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC.[citation needed] In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by control design engineers, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved with image processing.

4.3 Syntax in MATLAB

MATLAB, the application, is built around the MATLAB language. The simplest way to execute MATLAB code is to type it in at the prompt, >> , in the Command Window, one of the elements of the MATLAB Desktop. In this way, MATLAB can be used as an interactive mathematical shell. Sequences of commands can be saved in a text file, typically using the MATLAB Editor, as a script or encapsulated into a function, extending the commands available.

4.3.1 Variables

Variables are defined with the assignment operator, =. MATLAB is a weakly dynamically typed programming language. It is a weakly typed language because types

are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```
>> x = 17
x =
 17
>> x = 'hat'
x =
hat
>> y = x + 0
y =
     104      97     116
>> x = [3*4, pi/2]
x =
  12.0000    1.5708
>> y = 3*sin(x)
y =
  -1.6097    3.0000
```

MATLAB has several functions for rounding fractional values to integers:

1. round(X): round to nearest integer, trailing 5 rounds to the nearest integer away from zero. For example, round(2.5) returns 3; round(-2.5) returns -3.
2. fix(X): round to nearest integer toward zero (truncate). For example, fix(2.7) returns 2; fix(-2.7) returns -2
3. floor(X): round to the nearest integer toward minus infinity (round to the nearest integer less than or equal to X). For example, floor(2.7) returns 2; floor(-2.3) returns -3.

4. ceil(X): round to the nearest integer toward positive infinity (round to the nearest integer greater than or equal to X); for example, ceil(2.3) returns 3; ceil(-2.7) returns -2

4.3.2 Vectors/matrices

MATLAB is a "Matrix Laboratory", and as such it provides many convenient ways for creating vectors, matrices, and multi-dimensional arrays. In the MATLAB vernacular, a vector refers to a one dimensional (1×N or N×1) matrix, commonly referred to as an array in other programming languages. A matrix generally refers to a 2-dimensional array, i.e. an m×n array where m and n are greater than or equal to 1. Arrays with more than two dimensions are referred to as multidimensional arrays.

MATLAB provides a simple way to define simple arrays using the syntax: init:increment:terminator. For instance:

```
>> array = 1:2:9
array =
 1 3 5 7 9
```

defines a variable named array (or assigns a new value to an existing variable with the name array) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the init value), increments with each step from the previous value by 2 (the increment value), and stops once it reaches (or to avoid exceeding) 9 (the terminator value).

```
>> array = 1:3:9
array =
 1 4 7
```

the increment value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>> ari = 1:5
ari =
 1 2 3 4 5
```

assigns to the variable named ari an array with the values 1, 2, 3, 4, and 5, since the default value of 1 is used as the incrementer.

Indexing is one-based, which is the usual convention for matrices in mathematics, although not for some programming languages.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets: []. Parentheses: () are used to access elements and subarrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
 16  3  2 13
  5 10 11  8
  9  6  7 12
  4 15 14  1

>> A(2,3)
ans =
 11
```

Sets of indices can be specified by expressions such as "2:4", which evaluates to [2, 3, 4]. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>> A(2:4,3:4)
ans =
 11 8
```

```
 7 12
 14 1
```

A square identity matrix of size n can be generated using the function eye, and matrices of any size with zeros or ones can be generated with the functions zeros and ones, respectively

```
>> eye(3)
ans =
 1 0 0
 0 1 0
 0 0 1
>> zeros(2,3)
ans =
 0 0 0
 0 0 0
>> ones(2,3)
ans =
 1 1 1
 1 1 1
```

Most MATLAB functions can accept matrices and will apply themselves to each element. For example, mod(2*J,n) will multiply every element in "J" by 2, and then reduce each element modulo "n". MATLAB does include standard "for" and "while" loops, but using MATLAB's vectorized notation often produces code that is easier to read and faster to execute. This code, excerpted from the function magic.m, creates a magic square M for odd values of n (MATLAB function meshgrid is used here to generate square matrices I and J containing 1:n).

```
[J,I] = meshgrid(1:n);
A = mod(I+J-(n+3)/2,n);
```

```
B = mod(I+2*J-2,n);
M = n*A + B + 1;
```

4.3.3 Semicolon

Unlike many other languages, where the semicolon is used to terminate commands, in MATLAB the semicolon serves to suppress the output of the line that it concludes (it serves a similar purpose in Mathematica.)

4.3.4 Graphics

Function plot can be used to produce a graph from two vectors x and y. The code:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

produces the following figure of the sine function:

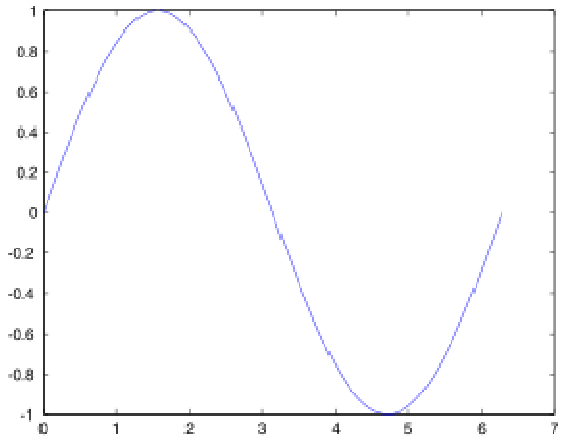

Figure 4.1 Sine wave generation in MATLAB

Three-dimensional graphics can be produced using the functions surf, plot3 or mesh.

```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
surf(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
```

This code produces a surface 3D plot of the two-dimensional unnormalized sinc function.

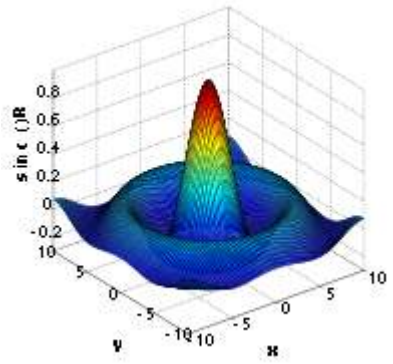

Figure 4.2: surface 3D plot of the two-dimensional unnormalized sinc function in MATLAB

4.3.5 Structures

MATLAB supports structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations etc). Unfortunately, MATLAB JIT does not support MATLAB structures, therefore just a simple bundling of various variables into a structure will come at a cost.

4.3.6 Function handles

MATLAB supports elements of lambda-calculus by introducing function handles, a references to functions, which are implemented either in .m files or anonymous/nested functions.

4.3.7 Secondary programming

MATLAB also carries secondary programming which incorporates the MATLAB standard code into a more user friendly way to represent a function or system.

Simulink

Simulink is a secondary program incorporated with MATLAB. It is a way to create or collaborate Equation(s) of Motion using an infrastructure of click, drag, and connecting blocks. These blocks are used to do many things including: defining variables, inputs, EOMs, Scopes, etc.

4.3.8 Classes

MATLAB supports classes, however the syntax and calling conventions are significantly different than in other languagues, because MATLAB does not have reference data types. For example, a call to a method

object.method();

cannot normally alter any variables of object variable. To create an impression that the method alters the state of variable, MATLAB toolboxes use evalin() command, which has its own restrictions.

4.4 Object-oriented programming

MATLAB's support for object-oriented programming includes classes, inheritance, virtual dispatch, packages, pass-by-value semantics, and pass-by-reference semantics.

```
classdef hello
    methods
        function doit(this)
            disp('hello')
        end
    end
end
```

When put into a file named hello.m, this can be executed with the following commands:

```
>> x = hello;
>> x.doit;
hello
```

4.5 Interactions with other languages

MATLAB can call functions and subroutines written in the C programming language or Fortran. A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "MEX-files" (for MATLAB executable).

Libraries written in Java, ActiveX or .NET can be directly called from MATLAB and many MATLAB libraries (for example XML or SQL support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with MATLAB extension, which is sold separately by MathWorks, or using an undocumented mechanism called JMI (Java-to-Matlab Interface), which should not be confused with the unrelated Java Metadata Interface that is also called JMI.

As alternatives to the MuPAD based Symbolic Math Toolbox available from MathWorks, MATLAB can be connected to Maple or Mathematica

4.6 Limitations of MATLAB

MATLAB is a proprietary product of MathWorks, so users are subject to vendor lock-in. Although MATLAB Builder can deploy MATLAB functions as library files which can be used with .NET or Java application building environment, future development will still be tied to the MATLAB language.

MATLAB, like Fortran, Visual Basic and Ada, uses parentheses, e.g. y = f(x), for both indexing into an array and calling a function. Although this syntax can facilitate a switch between a procedure and a lookup table, both of which correspond to mathematical functions, a careful reading of the code may be required to establish the intent. Furthermore, the syntax is inconsistent between arrays and functions, with [a,y]=f(1:2) valid for a function but a syntax error for an array, complicating the supposed advantage.

Mathematical matrix functions generally accept an optional argument to specify a direction, while others, like plot, do not, and so require additional checks. There are other cases where MATLAB's interpretation of code may not be consistently what the user intended] (e.g. how spaces are handled inside brackets as separators where it makes sense but not where it doesn't, or backslash escape sequences which are interpreted by some functions like fprintf but not directly by the language parser because it wouldn't be convenient for Windows directories). What might be considered as a convenience for commands typed interactively where the user can check that MATLAB does what the user wants may be less supportive of the need to construct reusable code.

Array indexing is one-based which is the common convention for matrices in mathematics, but does not accommodate any indexing convention of sequences that have zero or negative indices. For instance, in MATLAB the DFT (or FFT) is defined with the DC component at index 1 instead of index 0, which is not consistent with the standard

definition of the DFT in any literature. This one-based indexing convention is hard coded into MATLAB, making it difficult for a user to define their own zero-based or negative-indexed arrays to concisely model an idea having non-positive indices. A workaround can be constructed to create an ancillary frequency labeling array with element values equal to the index less 1 that would be used instead of the index, or the MATLAB programmer can remember to subtract 1 from the index obtained from functions that return an index such as find(), min(), max().

MATLAB built-in datatypes are always passed by value. Therefore, all input parameters to a function are usually copied (later MATLAB releases introduced lazy copy where if an input parameters are not being altered, a copy is not being made, however this has certain restrictions). Instances of user-defined classes are also copied and passed by value as the default, however user-defined classes can be made to exhibit reference behavior by inheriting from the abstract handle class. Variables of such classes store a reference to the instance. An alternative to using references is global variables, however MATLAB JIT does not support globals, as well as structures, therefore the code performance will degrade.

The MATLAB editor does not have code completion, references searches, or refactoring. Since MATLAB is weakly typed, these tools would be hard, if not impossible, to provide in future releases since it is unknown which methods of an object can be called without knowing its type. Without code completion, there is a tendency to use short cryptic names for variables, function and methods, making code hard to read. Without code completion or reference searches it can take a long time to text search for what functions and methods can be called in a given context. Without refactoring, it is time consuming and error prone to change variable, method, or function names, or to modify an API.

Productivity of a development team working on a large software project in MATLAB will likely be several times slower and result in a higher defect rate than development in a language and IDE with type checking, code completion, reference search, refactoring tools, and unit testing support.

It is very difficult to employ agile programming practices, such as Extreme Programming, in MATLAB due to the lack of refactoring tools.

GPU-based acceleration for MATLAB is available through Jacket by AccelerEyes for both single and multiple GPU applications. MATLAB lacks native support for GPU computing.

4.7 Alternatives

MATLAB has a number of competitors

1. The commercial software package Mathematica
2. Maple - a general-purpose computer algebra system developed and sold commercially by Maplesoft.

There are free open source alternatives to MATLAB, in particular GNU Octave, FreeMat, and Scilab which are intended to be mostly compatible with the MATLAB language (but not the MATLAB desktop environment). Among other languages that treat arrays as basic entities (array programming languages) are APL and J, Fortran 95 and 2003, as well as the statistical language S (the main implementations of S are S-PLUS and the popular open source language R).

There is one web / cloud based alternative - Monkey Analytics - which provides a modern web interface on top of GNU Octave or Python (including matplotlib, NumPy, SciPy and more).

There are also several libraries to add similar functionality to existing languages, such as Perl Data Language for Perl and SciPy together with NumPy and Matplotlib for Python.

CHAPTER 5

SIMULATION AND TESTING

In armature-controlled DC motors, the applied voltage Va controls the angular velocity w of the shaft.In this experiment two techniques for reducing the sensitivity of w to load variations (changes in the torque opposed by the motor load) is used.

A simplified model of the DC motor has been discussed in previous chapters. The torque Td models load disturbances. The speed variations induced by such disturbances are minimize. The physical constants of motor are:

R = 2.0 Ohms

L = 0.5 Henrys

Km = 0.1

Kb = 0.1

Kf = 0.2

J = 0.02

5.1 The angular velocity response of D.C.Motor

A state-space model of the DC motor with two inputs (Va,Td) and one output (w) is constructed. The plot of the angular velocity response to a step change in voltage Va is shown in Fig 5.1.

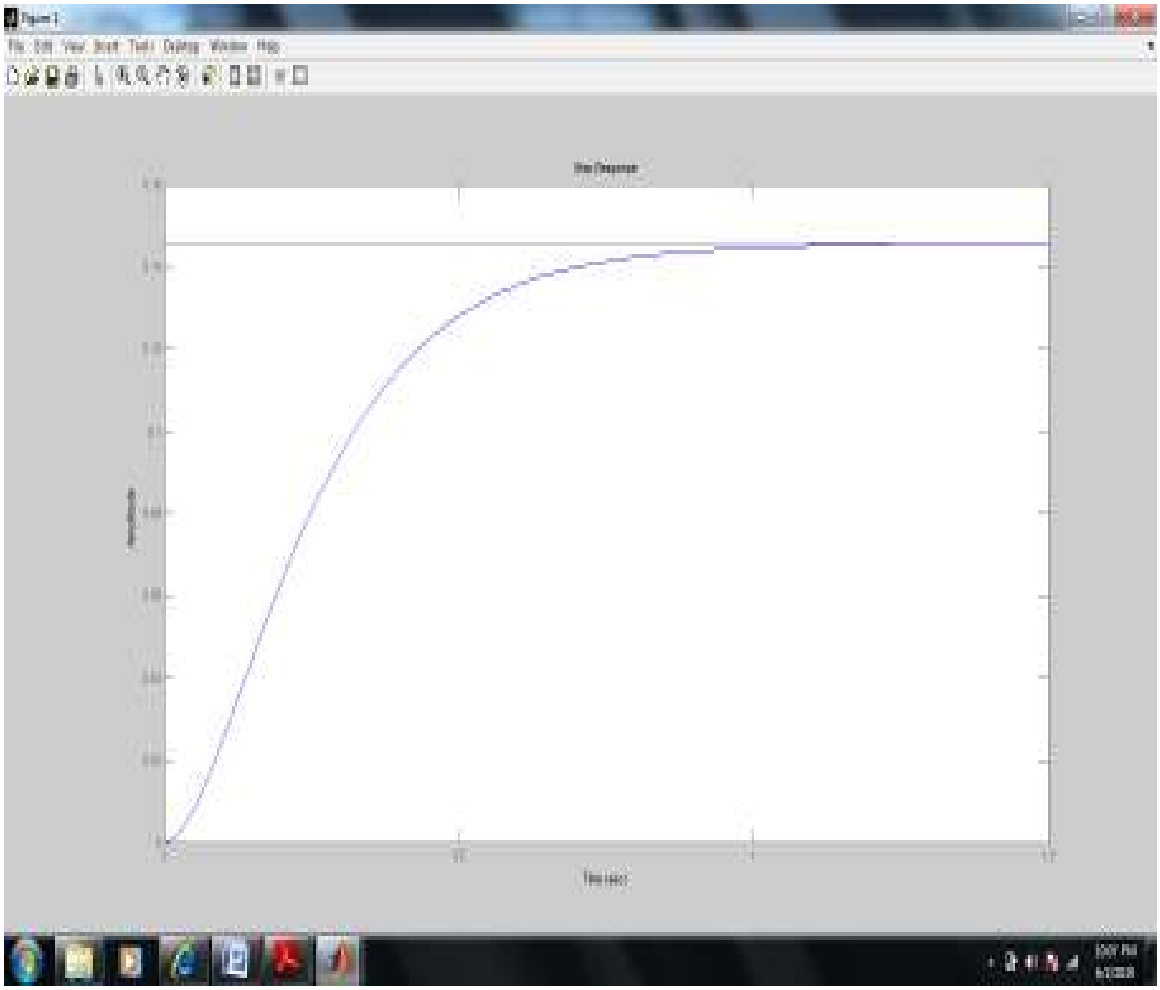

Figure 5.1 The plot of angular velocity response to a step change in voltage.

5.2 Response of D.C.Motor using Feedforward control

In feedforward control of D.C.motor The feedforward gain Kff should be set to the reciprocal of the DC gain from Va to w. To evaluate the feedforward design in the face of load disturbances, the response to a step command w_ref=1 with a disturbance Td = -0.1 Nm between t=5 and t=10 seconds is simulated:

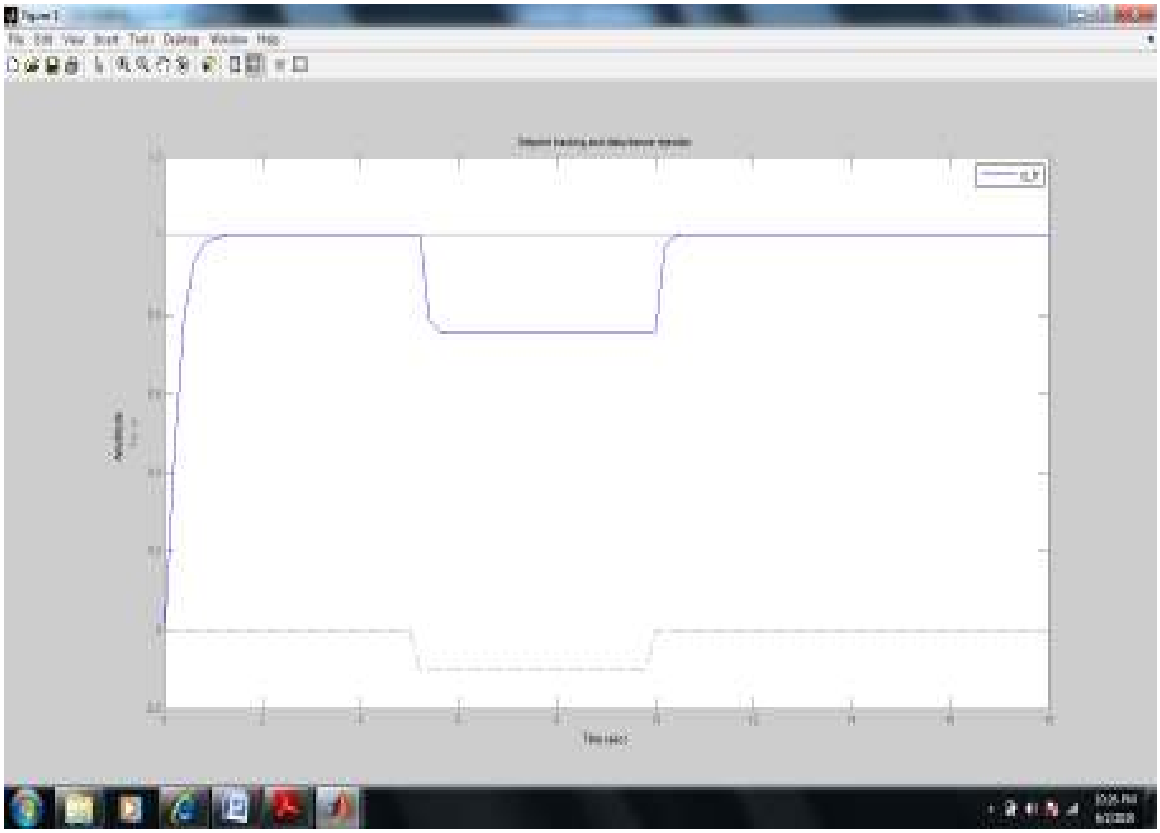

Figure 5.2 Feedforward control response of D.C.motor

It can be observed very clearly from the above figure that feedforward control handles load disturbances poorly.

5.3 Response of D.C.Motor using Feedback control

To enforce zero steady-state error, use integral control of the form

C(s) = K/s

where K is to was determined.

To determine the gain K, you can use the root locus technique applied to the open-loop 1/s * transfer(Va->w).

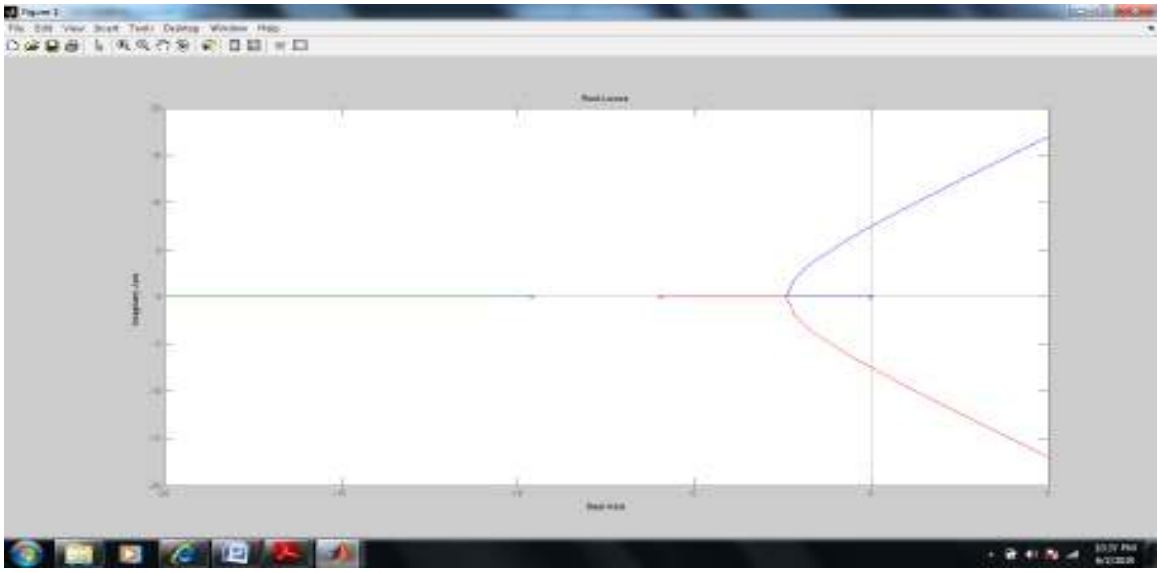

Figure 5.3 Root locus plot for feedback control response of D.C. Motor

Here K=8.

5.4 Comparison between feedforward and feedback control of D.C.Motor with root locus

It can be observed from fig 5.4 that the root locus design is better at rejecting load disturbances.

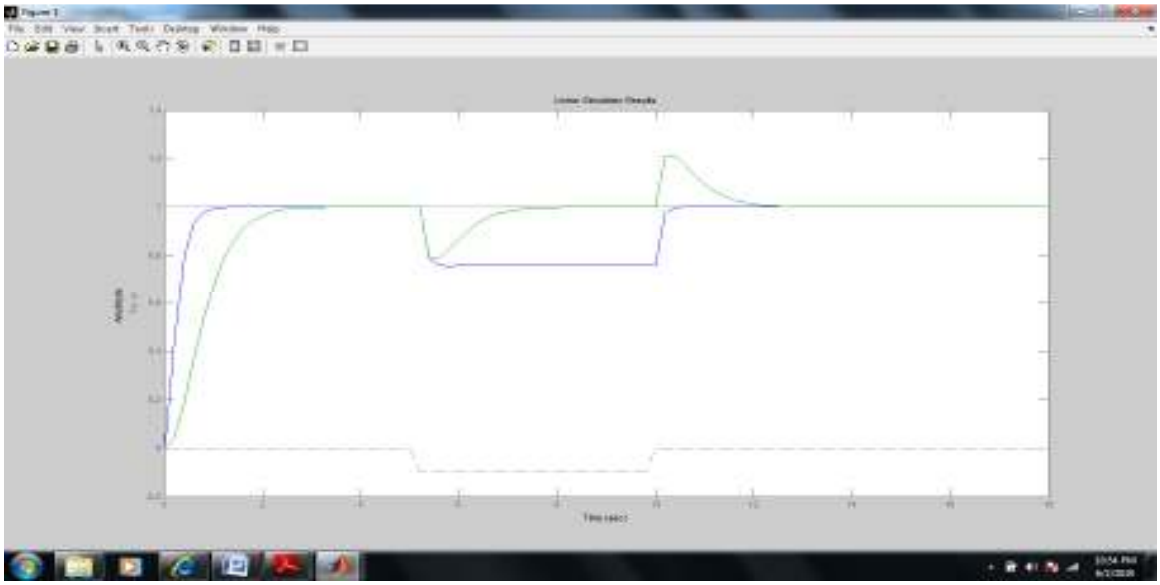

Figure 5.4 Comparison between feedforward and feedback control with root locus

In the above figure the blue line represents the response of feedforward control and green line represents response of feedback control with root locus.

5.5 Comparison between Bode plot of feedforward, feedback and linear quadratic regulator control response of D.C.Motor

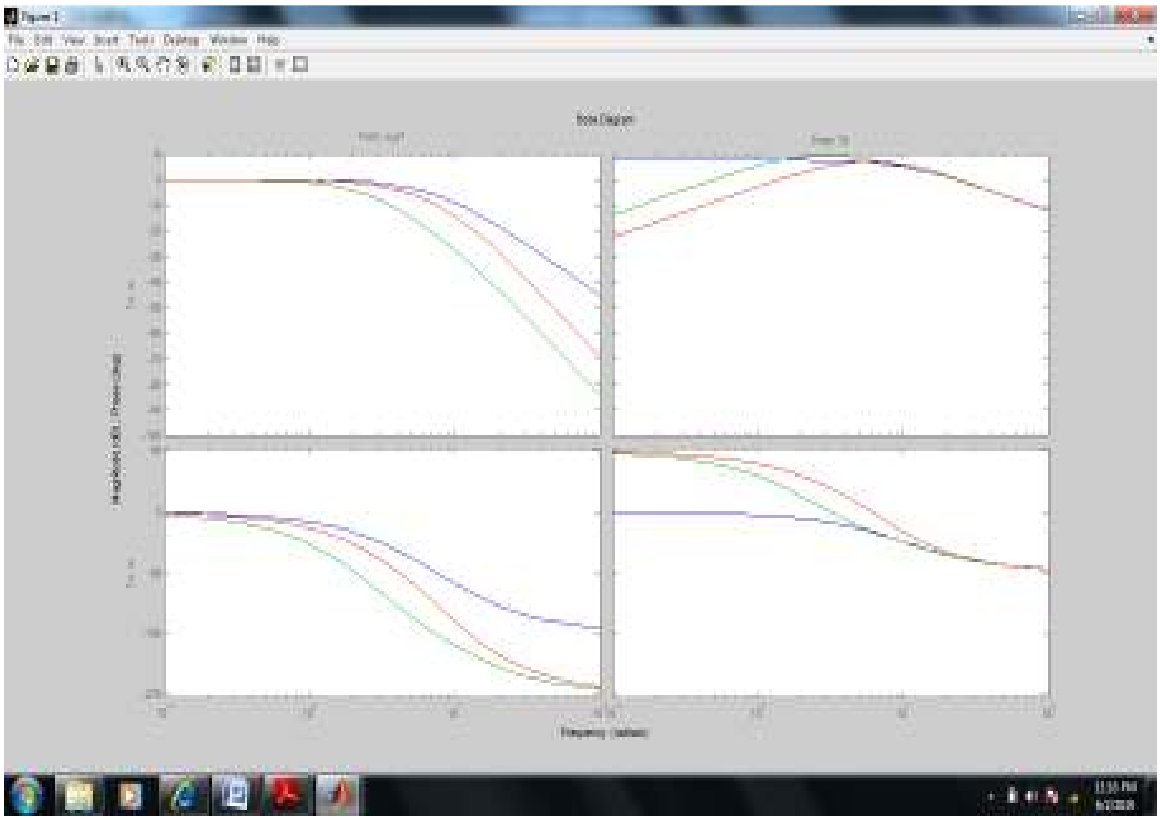

Figure 5.5: Comparison between Bode plot of feedforward, feedback and linear quadratic regulator control response of D.C. Motor

In the above figure the blue line represents the Bode plot response of feedforward control and green line represents the Bode plot response of feedback and red line represents the Bode plot response of linear quadratic regulator control of D.C.Motor.

5.6 Comparison between the designs of feedforward, feedback and linear quadratic regulator control response of D.C.Motor

Figure 5.6 Comparison between the designs of feedforward, feedback and linear quadratic regulator control response of D.C. Motor

In the above figure the blue line represents the response of feedforward control and green line represents the Bode plot response of feedback and red line represents the Bode plot response of linear quadratic regulator control of D.C.Motor.

It can be observed from the above plot that the Linear Quadratic Regulator compensator performs best at rejecting load disturbances than feedback and feedforward control.

CONCLUSION AND FUTURE SCOPE

This experiment was meant to verify that using MATLAB to develop an Linear quadratic regulator control system offers many advantages over the typical method of hand translation from pseudocode to source code. There is an inherent advantage in using MATLAB to model the control system. It saves time and effort, allowing the engineer to design the system in a straightforward manner, rather than wasting time writing source code from scratch.

In the above experiment it is observed very clearly from the above figure that feedforward control handles load disturbances poorly which can be improved by using feedback control. Finally it was observed that Linear Quadratic Regulator compensator performs best at rejecting load disturbances than feedback and feedforward control.

Lots of future work can be done to exploit the advantages of MATLAB and Simulink and their hardware targeting capabilities. The above process can be designed by using microcontrollers and DSP processors. Some more work could be done to analyze the differences between the continuous and discrete closed-loop systems. It's important to model the motor very accurately. Experimentation could let the user determine how fast the sampling frequency of the discrete system must be to approach the performance of the continuous system.

REFERENCES

[1] S. C. Won, D. J. Lim and D. H. Chyung , " D.C. Motor driven robotic simulator control" Proceedings of 24th Conference on Decision and Control ,pp 564-569, December 1985

[2] Kuribayashi K. and Nakajima K ., ''Active dancer roller system for tension control of wine and sheet." IFAC- Proceedings-Series, Volume 4: Process Industries, Power systems. pp 747-1752, 1985

[3] Shelton John,"Dynamics of Web tension control with velocity or torque control" proceedings of the American Control Conference -IEEE, New York, pp 1423-1427, I986

[4]Samart Yachiangkam,Cherdchai Prapanavarat,Udomsak Yungyuen and Sakorn Pongam " Speed Sensorless seperately excited D.C. Motor drive with an adaptive observer" IEEE Trans. On Industrial Drives, Vol. IE37, No.6, pp.562-575, 1990

[5] Hang C-C. K. J. Astrom and W. K. Ho, ''Refinements of the Ziegler-Nichols tuning formula ", IEE Proc. Part D, vol. 138, No. 2, pp. 111-118, March 1991

[6] M. Zhuang and D. P. Atherton, "Optimum Cascade PID Controller Design for SISO Systems", IEE Control, vol. 1, pp. 606-611, 1994

[7] Tongwen Chen and B.A. Francis, Optimal Sampled-Data Control Systems, Springer, London, Vol.5,pp 765-771,1995

[8] M.-R. Issa and E. Barbieri, "Optimal PI-Lead Controller Design", IEEE Proc. System theory, pp. 364-368, 1996

[9] Allan R. Hambley, "Electrical Engineering: Principles and Application", Prentice Hall, New Jersey, 1997

[10] Samir Mehta, Member, IEEE, and John Chiasson, Member, IEEE" Nonlinear Control of a Series DC Motor: Theory and Experiment" IEEE tractions on industrial Electronics, VOL. 45, NO. 1,pp 435-438, February 1998

[11] J.-B. He, Q.-G. Wang and T.-H. Lee, "PID Controller Tuning Via LQR Approach", IEEE Conference on Decision & Control, pp. 1177-1182, December 1998

[12] Chee-Mun Ong, "Dynamic Simulation of Electric Machinery", Prentice Hall, New Jersey, 1998

[13] Harsan Noura, Domanique Sauter, Frederic Hamelin, Didier Theilliol, “Fault-Tolerant Control in Dynamic System: Application to a Winding Machine”, IEEE Control Systems Magazine, pp33-49, 2000

[14] 0. Moseler and R Isermann, “Application of Model-Based Fault Detection to a Brushless DC motor”, IEEE Trans., vol. 35, No. 12, pp. 1015-1020,2000

[15] Chao Hu, “The Computerized Automobile Belt Test System by Modulating the Motor Speed with Inverter,” Mechanical & Electrical Engineering Magazine, Vo1.17, pp.67-68 , 2000

[16] Chao Hu and Xiangguang Cheng, “Electric Drive System for Winding Shaper of Automobile Belt”, Proceedings of IEEE Conference an Electric Machines and Systems, Shengyang, China, August , 2001

[17] S.H. Yu and J.S. Hu, "Error-Corrective Feedback Compensation to the Feedforward Tracking Control," ASME Journal of Dynamic Systems,Measurement, and Control, Vol. 123, Issue 3, pp. 324-329, 2001.

[18] M.MeenakshiFirst “Microprocessor Based Digital PID Controller for Speed Control of D.C .Motor” International Conference on Emerging Trends in Engineering and Technology, 2001

[19] G. Marro, D. Prattichizzo, and E. Zattoni,“Geometric Insight Into Discrete-Time Cheap and Singular Linear Quadratic Riccati (LQR) Problems”,IEEE Trans. Automatic Control, vol. 47, No. I, pp. 102-107, 2002.

[20]H. Koc. D. Knittel. M. de Mathelin, and G. Abba “Modeling and Robust Control of Winding Systems for Elastic Webs”, IEEE Transaction on Control Syslem Technology, Vol.lO, No.2, pp 197-208, 2002

[21] Chao Hu; Max Meng, Peter Xiaoping Liu, Xiang Wang,-“Optimal Digital Control System Design for Winding Shaping Process of Automobile Belt”, IEEE 2003. Canadian Conference on Electrical and Computer Engineering, pp.1763-1766, Montreal, Canada, May 4-7, 2003

[22] Mustafa Aboelhassan “Speed control of D.C. Motor using combined armature and field control IEEE. Ind. Applications., vol. 39. no 2, March/April 2003

[23] Voda A. A. and I. D. Landaq “A method of the autocalibration of PID controllers”, IEEE, vol. 3 1,No. 1, pp. 41-53,1995

[24] C.-L. Lin, H.-Y. Jan, and N.-C. Shieh, “GA-Based Multiobjective PID Control for a Linear Brushless DC Motor”, IEE, Trans. Mechatronics, vol. 8, No. 1,pp. 56-65,2003

[25] Chao Hu, Max Qinghu Meng and Peter Xiaoping Liu “Observer-Based LQR Control of Shaping Process of Automobile Belt” Proceedings of the 5" World Congress on Intelligent Control and Automation,pp 879-883 June 15-19. 2004

[26] M. Ruderman, J. Krettek, F. Hoffmann, T. Bertram “Optimal State Space Control of DC Motor” The International Federation of Automatic Control, 2003

[27] Moleykutty George “Speed Control of Separately Excited DC Motor” American Journal of Applied Sciences 5 (3); pp 227-233, 2008

[28] Ivo Petr´as” Fractional order feedback control of a D.C. Motor” Journal of Electrical Engineering , VOL. 60, NO. 3, 117–128, 2009