Table of Contents

Table of Figures

List of tables

1. Abstract

This report is a comprehensive overview of experiments performed to model a permanent magnet DC (PMDC) motor as a complex engineering project for the course EE-374 Feedback and Control Systems in partial fulfillment of the requirements for BE Electrical Engineering at NEDUET.
The report first discusses the working principles of DC motors, their variants, and methods for controlling their rotational speed. It then focuses almost exclusively on modelling a practical PMDC motor as a control system.
In this regard, the report first derives a transfer function for the motor in terms of its electromechanical parameters. It then discusses an experimental procedure used to measure the step response of the motor in terms of its rotational speed. Also provided is the Simulink model used in conjunction with MATLAB's parameter estimation toolbox for deriving the aforementioned parameters. The report then goes on to assess the stability of the system through Routh's criteria, and also provides a pole-zero map with the system's root locus (loci???).
The same procedure is repeated for an experimental setup in which gears are coupled to the motor's shaft, and the effect of gears on the system's step response, parameters, stability, and root locus are analyzed.
The report then goes on to discuss a MATLAB/Simulink-based implementation of a controller for the PMDC motor's speed, before finally concluding with an analysis of all observations made during the course of the project.

2. DC Motors

2.1 Overview:

A machine that converts DC electrical power into mechanical power is known as a Direct Current (DC) motor. DC motor working is based on the principle that when a current carrying conductor is placed in a magnetic field, the conductor experiences a mechanical force. The direction of the mechanical force is given by **Fleming's Left-hand Rule** and its magnitude is given by $\mathbf{F = BIL}$ Newton.

According to ***Fleming's left-hand rule:***
"When an electric current passes through a coil in a magnetic field, the magnetic force produces a torque which turns the DC motor."

The direction of this force is perpendicular to both the wire and the magnetic field. There are two electrical elements of a DC motor, the field windings and the armature. The armature windings are made up of current carrying conductors that terminate at a commutator. DC voltage is applied to the armature windings through carbon brushes which ride on the commutator. In small DC motors, permanent magnets can be used for the stator. The field winding basically form an electromagnet, that produces field flux that cuts the armature windings and produces mechanical torque in the armature, causing it to rotate.

2.2 Types of DC motor:

- Permanent Magnet DC motors
- Series DC motors
- Shunt DC motors
- Compound DC motors

2.2.1 Permanent Magnet DC motors:

Permanent magnet motor uses a permanent magnet to create a magnetic field or flux. They have excellent starting torque capability with good speed regulation. A disadvantage of permanent magnet DC motors is they are limited in the amount of load they can drive. Another disadvantage is that the torque is usually limited to 150% of rated torque to prevent demagnetization of the permanent magnets.

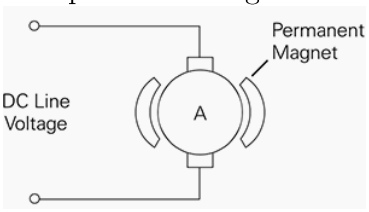


Figure 1: PMDC Motor

2.2.2 Series DC motors:

In a series DC motor the field is connected in series with the armature. The field is wound with a few turns of large wire because it must carry the full armature current. This form of DC motor is capable of producing large amount of starting torque. However, speed varies widely between no load and full load. Series motors cannot be used where a constant speed is required under varying loads.

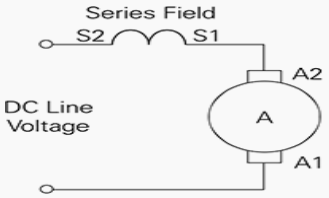


Figure 2: Series DC Motor

2.2.3 Shunt motors:

In a shunt motor, the field is connected in parallel (shunt) with the armature windings. The shunt-connected motor offers good speed regulation. The field winding can be separately excited or connected to the same source as the armature. An advantage to a separately excited shunt field is the ability of a variable speed drive to provide independent control of the armature and field.

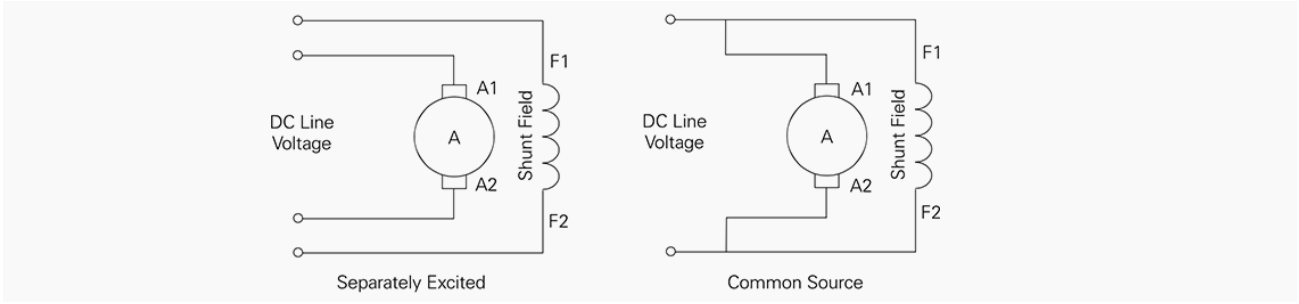


Figure 3: Shunt Motors

2.2.4 Compound motors:

Compound motors have a field connected in series with the armature and a separately excited shunt field. The series field provides better starting torque and the shunt field provides better speed regulation.

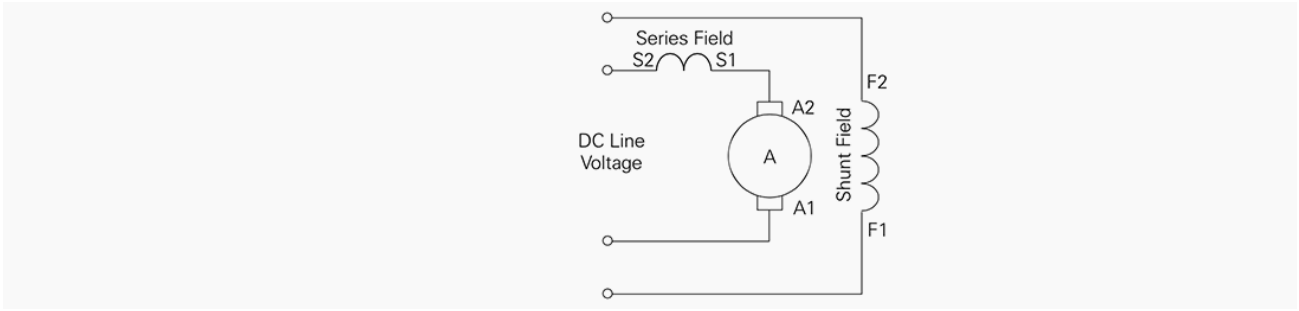


Figure 4: Compound Motors

2.3 Speed control:

DC motor speed control is one of the most useful features of the motor. By controlling the speed of the motor, you can vary the speed of the motor according to the requirements and can get the required operation. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage or the field current will change the rotor speed. Speed control can be achieved by variable battery tapings, variable supply voltage, resistors or electronic controls.

2.3.1 Why is this needed?

DC motor speed control is one of the most useful features of the motor. By controlling the speed of the motor, you can get the desired mechanical output according to the requirements of a specific task. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage or the field current will change the rotor speed. Speed control can be achieved by variable battery tapings, variable supply voltage, resistors or electronic controls.

2.3.2 Armature control:

In the armature control method, the speed of the DC motor is directly proportional to the back EMF E_b and $E_b = V - I_A R_A$. When supply voltage (V) and armature resistance R_A are kept constant, the speed is directly proportional to armature current I_A. Armature voltage control is used to vary speed up to rated speed. Torque remains fairly constant.

2.3.3 Field control:

In field control method, the field is weakened to increase the speed or it can be strengthened to reduce the motor's speed. The speed of a dc motor can be varied by varying the field current. The armature is also supplied by means of a phase controlled rectifier to maintain constant armature current. Field control is used speed up motor above rated value but at the cost of toque.

3. Our DC Motor

Our motor is a PMDC motor (or permanent magnet DC motor), a type of motor that uses a permanent magnet to create the magnetic field required for the operation of a DC motor. As the magnetic field strength of a permanent magnet is fixed, it cannot be controlled externally, field control of this type of DC motor cannot be possible.

3.1 Rated Parameters

Parameters	Rated values
DC Voltage	12V
Motor Current	0.13 A
Power rating	0.86 W
Max operating speed	630 rpm
Max operating torque	1.79 mN.m

Table 1: Rated Parameters from an online data sheet

These parameters are for model no. RF-500-tb12560 taken from datasheets.com

3.2 DC Motor Parameters

3.2.1 Armature Inertia

Inertia of a motor or armature inertia is the resistance to change in speed of motor. It is denoted by J_a. For the motor to efficiently control load during acceleration and deceleration, the motor and load inertias should theoretically be equal.

There's a direct relationship between motor inertia and torque.

$$\boldsymbol{Torque = J_a \times angular\ speed}$$

A motor having higher value of armature inertia will give more torque for same angular speed.

3.2.2 Viscous Friction

Viscous friction is velocity dependent friction. It is denoted by B in motor's transfer function. At zero velocity, viscous friction is zero and increases with the increase in velocity.

3.2.3 Armature Resistance

It is the resistance of armature circuit of DC motor. It is denoted by R_a. If a DC motor whose armature is stationary is switched directly to supply voltage, it can damage the motor winding due to unbearable current flow. This is because the armature resistance is very small. Thus, additional resistance must be added to armature circuit.

3.2.4 Armature Inductance

DC voltage may have ripples. The ripple voltage cause a ripple current to flow in the armature, but because of the armature inductance, the amplitude of the ripple current is small.

Armature inductance is important because of its effect on electrical time constant, and on the current waveform and torque when the motor is supplied from a phase-controlled converter or chopper.

3.2.5 Motor Constants (K_v & K_t)

K_V is the motor velocity constant. It is the ratio of unloaded speed to the applied voltage of a motor. It is measured in RPM per volt or radians per volt-second.

K_T is the torque produced divided by the armature current and can be calculated from the motor velocity constant as,

$$K_T = \frac{\tau}{I_A} = \frac{60}{2\pi K_V}$$

4. Deriving a DC Motor's Transfer Function

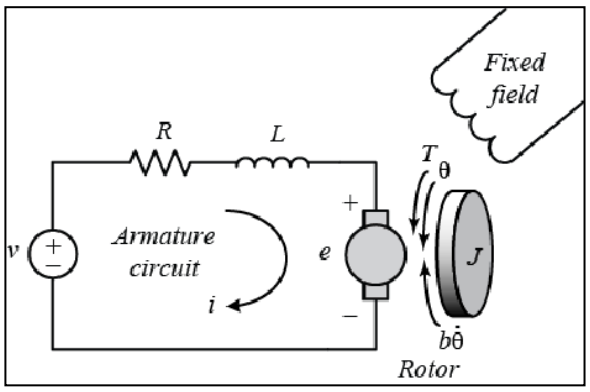


Figure 5: A simple Diagram of a DC Motor

As is evident from Figure 5, a DC motor is a combination of two, interrelated control systems:

- A mechanical control system that deals with purely rotational kinematic parameters such as moment of inertia, applied and shaft torque, the shaft's angular speed of rotation, and the shaft's angular displacement.
- An electrical control system which is concerned primarily with the effects of resistive, inductive, capacitive, and magnetic components in the both the armature circuit.

Deriving a single transfer function for the system involves combining transfer functions for both of the aforementioned systems.

4.1 Armature Circuit Transfer Function

Deriving the armature circuit's transfer function by applying Kirchhoff's Voltage Law to the armature circuit

$$v(t) = V_R + V_L + v_B$$

Where V_R the voltage drop is across the armature resistance, V_L is the voltage drop across the armature inductance, and v_B is the back EMF induced in the armature circuit. The back EMF v_B is proportional to the angular speed of the shaft ω which, in turn, is simply the rate of change of the shaft's angular displacement θ with respect to time

$$v_B \, \alpha \, \omega \because \omega = \frac{d\theta}{dt} \Rightarrow v_B \, \alpha \frac{d\theta}{dt}$$

$$\Rightarrow \boldsymbol{v_B = k\frac{d\theta}{dt}}$$

Applying the Laplace transform to this differential equation and assuming any initial conditions, including the initial angular displacement, were 0.

$$\mathcal{L}\{v_B\} = \mathcal{L}\left\{k\frac{d\theta}{dt}\right\} \Rightarrow \boldsymbol{V_B = ks\theta(s)}$$

Expressing the KVL equation in terms of armature current and applying the Laplace Transform to the result

Expressing this linear equation in the form of a block diagram

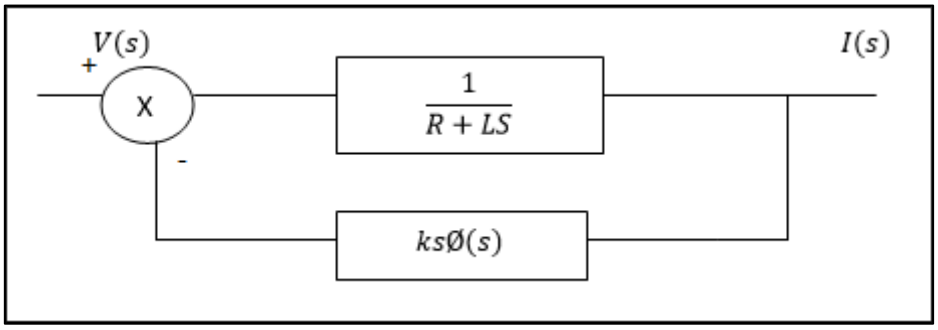


Figure 6: Block Diagram of armature circuit

This block diagram is for the armature circuit.

4.2 Mechanical Circuit Transfer Function

The total torque acting on the system is a combination of the torque due to the system's inertia τ_J as well as the opposing torque due to damping forces or friction τ_B.

$$\tau(t) = \tau_J + \tau_B$$

As the total torque in the system is proportional to the armature current I_A

$$\tau(t) \, \alpha \, I_A \Rightarrow \tau(t) = K_T I_A$$

And because the inertial torque and damper's opposing torque are both functions of the rate of change of angular displacement θ

$$\tau_J =$$

$$J\frac{d^2\theta}{dt^2} \; and \; \tau_B = B\frac{d\theta}{dt}$$

The differential equation can be expressed in terms of the armature current and angular displacement as follows

$$K_T I_A = J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt}$$

Applying the Laplace Transform to this differential equation under the assumption that all initial conditions are 0

$$\mathcal{L}\{K_T I_A\} = \mathcal{L}\left\{J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt}\right\}$$

$$\Rightarrow K_T I_A(s) = (Js + B)s\theta(s)$$

$$\Rightarrow \boldsymbol{\frac{s\theta(s)}{I_A(s)} = \frac{K_T}{Js + B}}$$

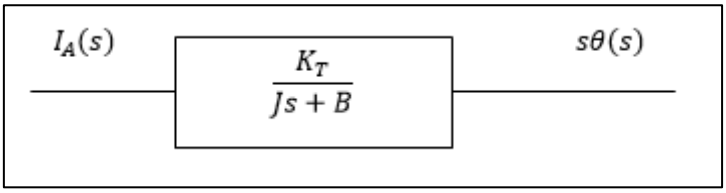


Figure 7: Block Diagram of Mechanical Part

This is block diagram for mechanical part.

Now putting both block diagrams in one,

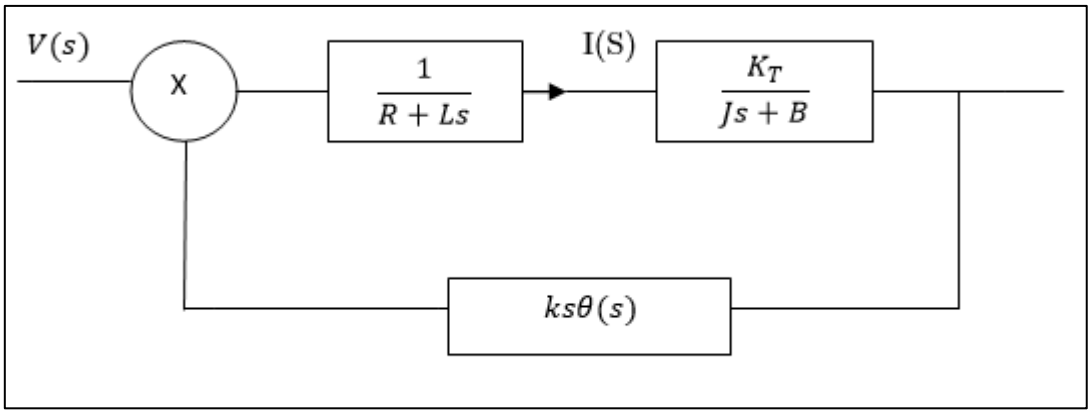


Figure 8: Combined Mechanical and Electrical Part

Now,

$$s\theta(s) = \omega(s) \Rightarrow \frac{\theta(s)}{\omega(s)} = \frac{1}{s}$$

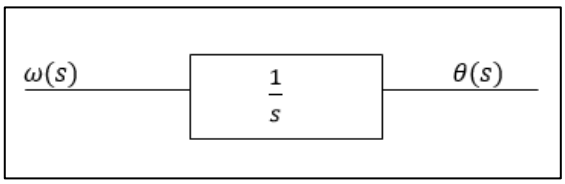


Figure 9: Integrator

And back EMF $V_b = Ks\theta(s) \Rightarrow K = \frac{V_b}{s\theta(s)}$

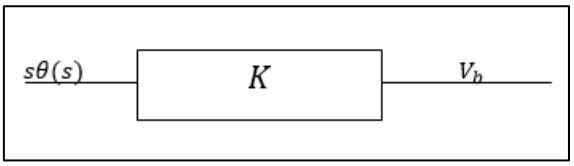


Figure 10: Back EMF Gain

Now combining all the blocks for the total transfer function.

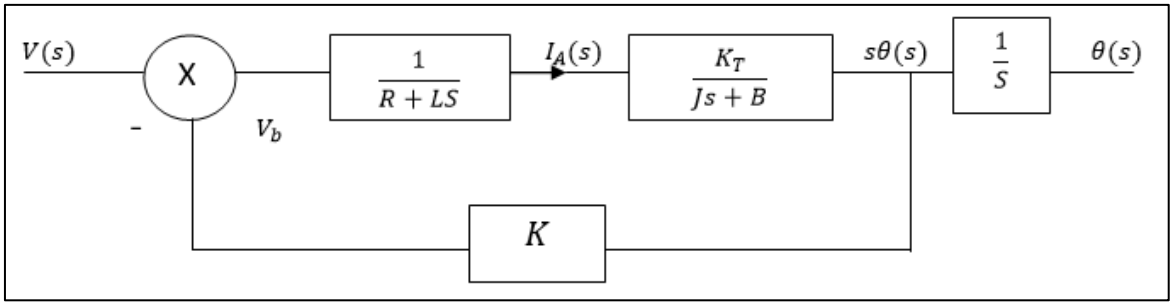


Figure 11: Final Block Diagram

$$\frac{\boldsymbol{\theta(s)}}{\boldsymbol{V(s)}} = \frac{\boldsymbol{K}}{\boldsymbol{(Js+B)(Ls+R)+K^2}} \qquad \left[\frac{\boldsymbol{rad/sec}}{\boldsymbol{V}}\right]$$

The state space representation is

$$\begin{bmatrix} \frac{d\omega}{dt} \\ \frac{dI_A}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{K}{J} \\ \frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ I_A \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V$$

While the output can similarly be represented as

$$Y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ I_A \end{bmatrix}$$

5. Experimental Setup

5.1 Procedure Plan

For caring out the task of parameter estimation or for any further experimentation, the DC motor's characteristics should be analyzed or we can say that the response of the DC motor output with respect to its input must be captured. The input and output of our system are

$$Input \ = \ Armature\ Voltage\ (V_a)$$
$$Output \ = \ Rotations\ per\ Minute\ at\ the\ Shaft\ of\ the\ motor$$

Since we plan of making a virtual model of our DC motor in the Simulink environment and further test the stability of our system we shall carry out an experiment that will tell us the step response of the system.

5.2 Equipment Used

The list of components and software required to carry out this experiment is given below:

a) Permeant magnet DC Motor
b) Slotted Disc
c) LM 393 RPM sensor
d) Arduino Uno Board
e) DC power Supply or adapter
f) Arduino IDE

5.3 Basic layout of the Setup

To capture the RPM data of the motor. A hole is made at the center of the slotted disc which is then fitted at the shaft of the motor. The motor is placed in such an orientation that the slots of the disc are made to pass within the gap of LM393 sensor. Now as the voltage is applied on motor's armature circuit the shaft rotates in turn causing the disc to revolve within the sensor's gap. The sensor's infrared beam is interrupted and this registers a pulse at the sensor output which is read by the Arduino code. The faster the disc rotates the more pulses the sensor registers, which are counted by the Arduino. The RPM data is data is displayed in the serial monitor of the Arduino IDE and later on copied in a excel file, which can be readily imported in the MATLAB workspace.

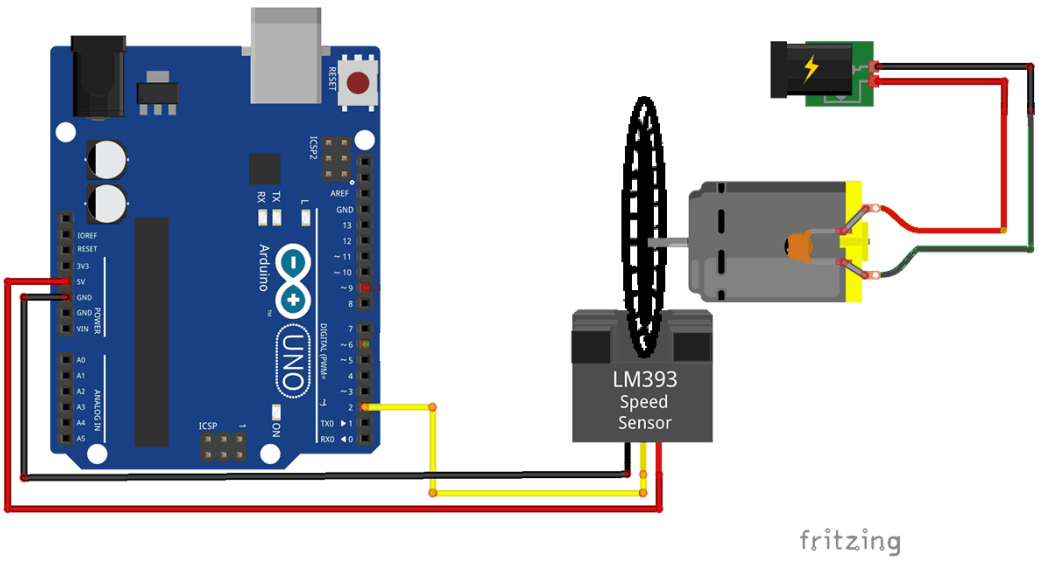


Figure 12: Circuit Diagram made on Fritzing

5.4 Arduino Program

The program used for counting the pulses is given below. The program uses a Timer object from the TimerOne library to measure the number of times the IR sensor detects interruptions in the IR signal caused by the rotation of the disc over a 0.1s interval.
The timer is first started and then interrupted every time a rising pulse is generated on pin number 2, which is connected to the IR sensor. The IR sensor will generate a rising pulse every time the disc interrupts the IR signal due to its rotation. This, in turn calls the docount() method which increments a counter variable that keeps track of the number of times the sensor's IR signal has been interrupted by the disc, and thus indirectly measures the speed of the motor in terms of the number of interruptions per 0.1 s.
At the end of 0.1s, the timer's operation is interrupted and it carries out the commands encapsulated in an interruption service routine (ISR), which defines the steps to be taken once the timer operation has been interrupted. In this case, the ISR will first stop the timer from keeping track of time and will then use the counter variable to calculate motor speed. Specifically, as there are 20 slots in the disc, the total number of interruptions recorded by the counter are divided by 20 to find the total number of complete revolutions made by the disc in the 0.1 s of the timer. Moreover, the resulting revolutions are multiplied by a factor of 10 to get revolutions per second as

$$RPS = \frac{revolutions}{0.1s} = \frac{revolutions}{\frac{1}{10}s} \therefore RPS = 10 \times revolutions$$

This data is then printed to the serial monitor as a decimal number (DEC), the counter is reset to 0 and the timer is enabled again. The process then repeats indefinitely, with the Arduino calculating the RPS using interruptions per 0.1s.

```
#include "TimerOne.h" // Including TimerOne library
unsigned int counter=0;

void docount()  // counts from the speed sensor
{
  counter++;  // increase +1 the counter value
}

void timerIsr()
{
  Timer1.detachInterrupt();  //stop the timer
  Serial.print("Motor Speed: "); // Print text on the
serial Monitor
  int rotation = 10*(counter / 20);
```

```
// divide by number of holes in Disc and multiplied by 10
to get RPS
// because the timer is set for 0.1 second
  Serial.print(rotation,DEC);


  Serial.println(" Rotation per seconds");
  counter=0;  //  reset counter to zero
  Timer1.attachInterrupt( timerIsr );  //enables the timer
again
}

void setup()
{
  Serial.begin(9600); // Enables the serial monitor

  Timer1.initialize(100000); // set timer for 0.1 second
interval

// Attach Interrupt corresponding to pin 2 on the Arduino
board
  attachInterrupt(0, docount, RISING);
// The above line calls an interrupt called "Docount" when
the pin 2 receives a rising signal. Which in turn increases
counter  variable when speed sensor pin goes High

// Attach a Timer Interrupt
  Timer1.attachInterrupt( timerIsr ); // enable the timer
}

void loop()
{

}
```

Arduino Code with Description

The code below is a simple MATLAB program used to visualize the RPS/RPM data acquired by the Arduino IR sensor. The data has been saved into an Excel spreadsheet that is then placed in the same directory as the MATLAB program.

The program first reads in the data and removes any extra null columns/rows i.e. rows/columns in the original spreadsheet that did not store sensor data. It then separates the data into RPS and RPM arrays. Both these arrays represent digital, discrete data that has been sampled at discrete time intervals, and thus not amenable to continuous plotting. As such, the MATLAB program used the built-in smooth function apply an exponential moving average filter to the data to store a smoother, more continuous variation.

```
1    %% dc_motor_viz.m - Visualize DC Motor RPM data for EE-374 FCS CEP
```

```
%%  Import data from Excel
raw_data = xlsread('DC Motor Data');

%% drop null cells
dc_data = raw_data(1:65, :);

%% extract RPM and RPS data into separate columns
rps = dc_data(1:65, 1);
rpm = dc_data(1:65, 2);

%% smoothen data with default exponential moving average
rps_avg = smooth(rps);
rpm_avg = smooth(rpm);

%% plot original vs smoothened data
figure();

% RPS
subplot(1, 2, 1); plot(rps); hold();plot(rps_avg);
legend('Original', 'Moving Average'); grid();
ylim([0, 30]); title('DC Motor Step Response - RPS');
xlabel('Sample (\it{n/arbitrary units})');
ylabel('Disc Speed (\it{\omega/RPS})')

% RPM
subplot(1, 2, 2); plot(rpm); hold();plot(rpm_avg);
legend('Original', 'Moving Average'); grid(); ylim([0, 17e3])
title('DC Motor Step Response - RPM');
xlabel('Sample (\it{n/arbitrary units})');
ylabel('Disc Speed (\it{\omega/RPM})');
```

The results of the visualization are shown in Figure 2. It is evident that the step response is far from ideal. Instead of the motor's speed rising to its maximum value instantly in response to an instantaneous/step supply voltage, the speed increases to a maximum value gradually, indicating a rise time that is characteristic of practical second order systems.

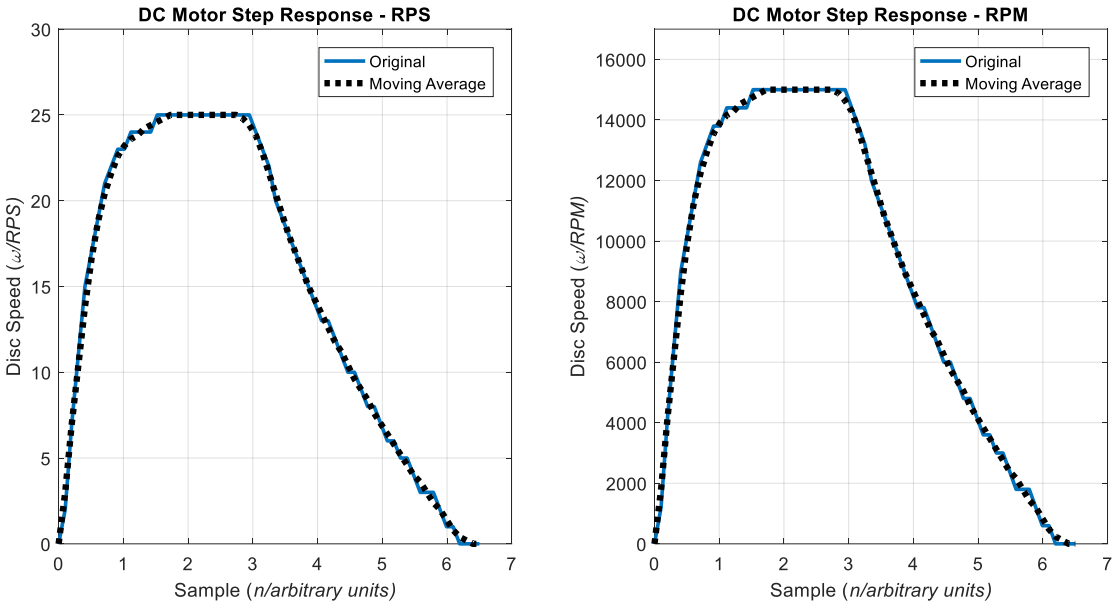


Figure 13: Plot of Raw RPM data of Motor

5.5 Speed Data Processing

Since the slotted disk used in the experiment has a radius of 2.5 cm, the speed displayed by the Arduino is not the actual speed of the motor, but is actually some multiple of the real speed of the motor at the shaft. To get the actual speed we need to preprocess the data and accommodate for the disc radius.

$$Rotation\ per\ second\ =\ RPS$$
$$Rotations\ per\ minute\ =\ RPM$$
$$Radius\ of\ disc\ =\ 2.5\ cm\ =\ 0.025\ meter$$
$$Radius\ of\ shaft\ =\ 0.001\ m\ (1\ mm)$$
$$RPM\ at\ Disc\ = \frac{RPS\ at\ Disc}{60}$$
$$RPM\ at\ shaft = \frac{RPM\ at\ disc}{Radius\ of\ Disc} \times Radius\ of\ shaft$$
$$RPM\ at\ shaft = \frac{RPS\ at\ disc}{60 \times Radius\ of\ Disc} \times Radius\ of\ shaft$$
$$\boldsymbol{RPM\ at\ Shaft = RPS\ at\ Disc\ \times 24}$$

Also to visualize the step response of the motor we only require the first 30 samples during which the motor's speed starts to rise in response to a step input voltage and settles at a steady state value.

Speed Data of DC motor Due to a step input

	Raw Speed Data		Speed Data after Processing
S. No.	**RPS at Disc**	**RPM at Disc**	**RPM at Shaft**
1.	0	0	0
2.	2	1200	48
3.	7	4200	168
4.	11	6600	264
5.	15	9000	360
6.	17	10200	408
7.	19	11400	456
8.	21	12600	504
9.	22	13200	528
10.	23	13800	552
11.	23	13800	552
12.	24	14400	576
13.	24	14400	576
14.	24	14400	576
15.	24	14400	576
16.	25	15000	600
17.	25	15000	600
18.	25	15000	600
19.	25	15000	600
20.	25	15000	600
21.	25	15000	600
22.	25	15000	600
23.	25	15000	600
24.	25	15000	600
25.	25	15000	600
26.	25	15000	600
27.	25	15000	600
28.	25	15000	600
29.	25	15000	600
30.	25	15000	600

Table 2: Raw and Processed RPM Data

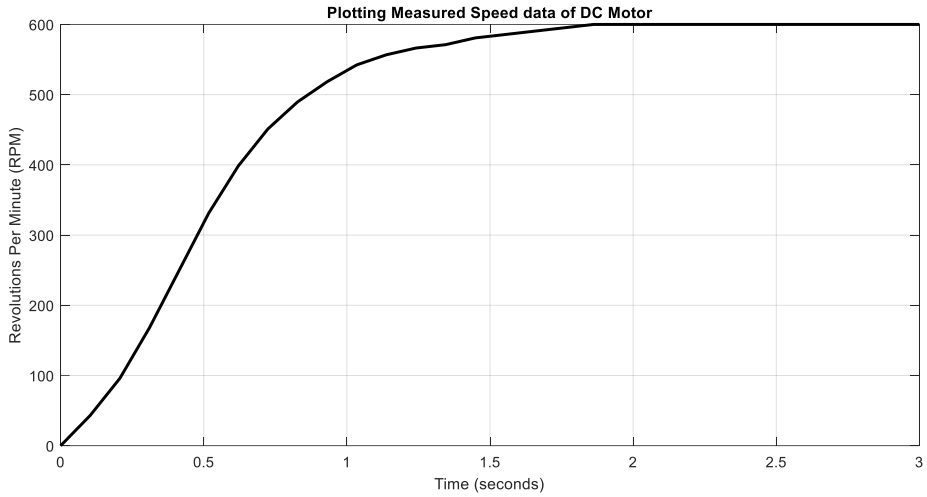


Figure 14: Plot Showing Step Response of Motor (RPM data)

6. System's response Without Gears

6.1 Step Response

Step response of a system is the output generated by it in response to a step function. Essentially a step response if the time behavior of the output of a system when its input changes from zero to any non-zero (positive) value in a very short period of time. From practical point of view knowing the step response of a dynamical system gives information on the stability of such a system, and on its ability to reach one stationary state when starting from another.

To analyze the closed loop step response of our DC motor we shall make use of the Simulink model, the description of which can be found the previous section (section 5). The Simulink model is essentially based on the motor transfer function that we derived in section 4. The magnitude of the step function used for the simulation has been multiplied by a factor of 14.4 so as to mimic the actual step voltage input we provided to the DC motor during the physical experiment.
The step response generated by the Simulink model is initially used with dummy parameters values, which are given below

$$Resistance = R = 1\,\Omega$$
$$Indctance = L = 0.5\,H$$
$$Friction\ COnstant = B = 0.1\,N.m.s$$
$$Moment\ of\ Inertia = J = 0.01\,kg.m/\sec^2$$
$$Torque\ Constant = Kt = 0.01\,N.\frac{m}{amp}$$
$$Back\ EMF\ Constant = Kb = 0.05\,\frac{volt.sec}{rad}$$

The plot generated by the model is shown below

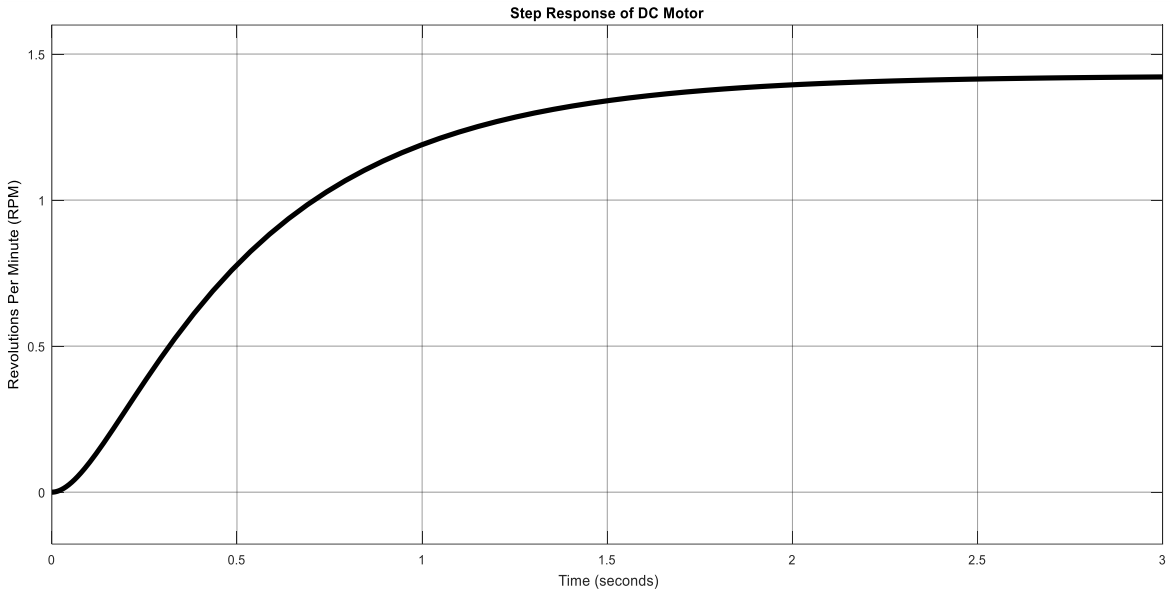


Figure 15: Step response with dummy parameters

The step response is applied at $t = 1\,sec$. It can be seen that the output response of the DC motor doesn't immediately changes from zero to the final value but follows a more natural delayed response and settles at around $t = 2.5\,sec$ to a final value of 1.43.
This response will be used in the next section for the task of parameter estimation.

6.2 Parameter Estimation

The term *parameter estimation* refers to the process of using sample data (in reliability engineering success data) to estimate the parameters of the selected distribution. For this experiment the parameter estimation technique is used to estimate the DC motor parameters of our physical motor and fundamentally transfer it to a digitized form which can be then be used for further experimentation.
The parameter estimation takes use of the speed data recorded by the RPM sensor we used in our physical setup, the step response we found in the previous section and uses estimation techniques to match both of them, while at the same time tuning the relevant parameters in the process that would generate the our desired response.

Below is a plot that helps us in visualizing both the data collected from the RPM sensor, labelled as measured data, and the step response data of the data, labelled as simulated data.

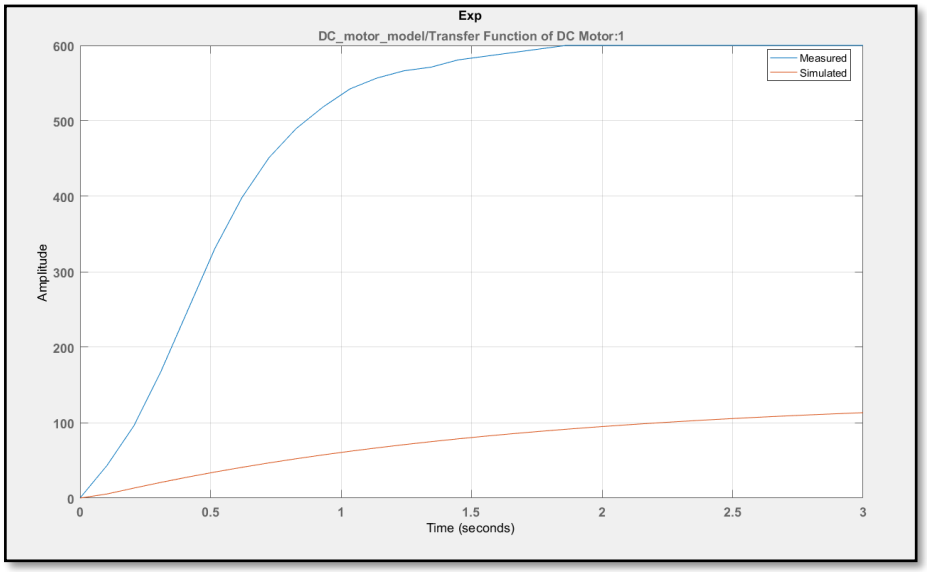


Figure 16: Simulated and Measured RPM data plot

It can be seen that the measured data hugely deviates from the simulated. The simulated response final value is 1.43 (from the previous section) while the measured data's final is 600, this is because the parameters of our Simulink model aren't tuned to match the characteristics of our actual DC motor. For this purpose we perform the parameter estimation task.

The parameters are estimated using the parameter estimation toolbox in MATLAB and function employed is sum squared error, which is one the data estimation methods available in MATLAB. Firstly the parameters which are to be optimized are selected, then the output of the Simulink model from where the simulated data is drawn is selected as the experiment output. Afterwards the time vector and the measured data are loaded in the experiment. Then the estimation button is pressed which generates the plots show below

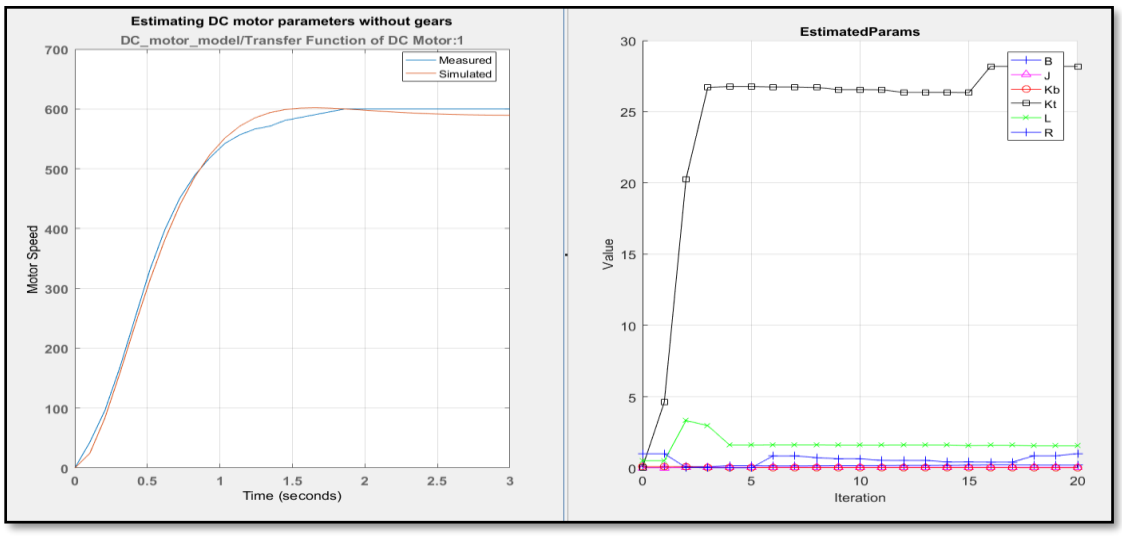


Figure 17: Plot detailing Parameter Estimation Process

The above figure contains 2 plots. The plot on the left shows the measured and the simulated data in the same figure which helps us to visualize the how close the data has gotten after the parameters have been tuned to fit the measured data. The plot in the right shows the deviation or the change in the parameters value on different iterations during the process of estimation.

The results of the experiment are summarized in the tables below:

a) Results of Experiment

Number of Iterations	20
Optimization started	22-Jun-2019 19:57:16
Estimation Converged	22-Jun-2019 19:59:01

b) Parameter Values

Parameters	Old Parameter Values	Estimated Values
Resistance (R)	1	1.0047
Inductance (L)	0.5	1.5537
Moment of Inertia (J)	0.01	0.0482
Constant of Friction (B)	0.1	0.1960
Torque Constant (Kt)	0.01	28.18
Back EMF Constant (Kb)	0.05	0.0174

The new response generated by the Simulink model after the parameters are estimated to fit the data of the our actual motor is given below

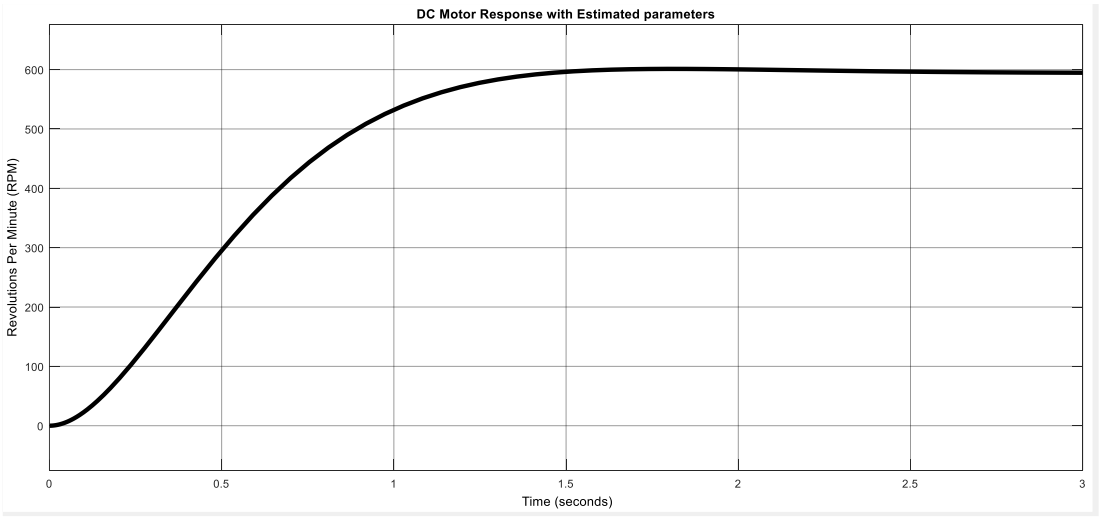


Figure 18: Step Response with Estimated parameters

Notice how the final value (or basically the speed of the motor) is now at 600 RPM instead of 1.439 RPM. Hence, in this way we have created a virtual copy of our actual physical motor, which can later on be used to perform further experimentation.

6.3 System Response without Gears

Gears are simple machines that are used to transmit power from one part of the machine to another. They alter the speed, torque and the direction of the power source.
You can have any number of gears connected together and they can be in different shapes and sizes. Each time you pass power from one gear wheel to another, you can do one of three things:

- **Increase speed**: If you connect two gears together and the first one has more teeth than the second one (generally that means it's a bigger-sized wheel), the second one has to turn round much faster to keep up. So this arrangement means the second wheel turns faster than the first one but with less force.
- **Increase Torque**: If the second wheel in a pair of gears has more teeth than the first one (that is, if it's a larger wheel), it turns slower than the first one but with more force
- **Change direction**: When two gears mesh together, the second one always turns in the opposite direction. So if the first one turns clockwise, the second one must turn counterclockwise. You can also use specially shaped gears to make the power of a machine turn through an angle.

The gears used in this experiment are plastic gears, one of which is mounted on the motor shaft while the other is mounted on an independent shaft and the two gears are mated with their grooves touching each other. The slotted disc is mounted in the independent shaft and now rotates at the speed which is transferred from the DC motor via the gear ratio. The number of teethes on gears are:

$$Number\ of\ teeths\ on\ drive\ gear = 10$$
$$Number\ of\ teeths\ on\ driven\ gear = 25$$

6.4 Stability Of System By Routh's Criteria

The transfer function of our system is,

$$\frac{\theta(s)}{V(s)} = \frac{K}{JLs^2 + (JR + BL)s + BR + K_t K_b}$$

Putting values J=1.5537, L= 1.5537, R= 1.0047, B= 0.1960 and K_t=28.18 and K_b = 0.0174 in transfer function we get

$$\frac{\theta(s)}{V(s)} = \frac{28.18}{0.0748s^2 + 0.3529s + 0.6878}$$

Routh's table for above transfer function is given below

S^2	0.0748	0.6878
S^1	0.3529	0
S^0	0.6878	0

Since all values of first column are positive, therefore it is concluded that there is no right pole and thus, the system is stable.

7. System's response With Gears

7.1 Step Response

To analyze the closed loop step response of our DC motor, which is now attached with gears we shall again make use of the Simulink model. The Simulink model is essentially based on the motor transfer function that we derived in section 4. The magnitude of the step function used for the simulation has been multiplied by a factor of 14.4 so as to mimic the actual step voltage input we provided to the DC motor during the physical experiment. The step response generated by the Simulink model is initially used with dummy parameters values, which are given below

$$Resistance = R = 1\,\Omega$$
$$Indctance = L = 0.5\,H$$
$$Friction\ COnstant = B = 0.1\,N.m.s$$
$$Moment\ of\ Inertia = J = 0.01\,kg.m/\sec^2$$
$$Torque\ Constant = Kt = 0.01\,N.\frac{m}{amp}$$
$$Back\ EMF\ Constant = Kb = 0.05\,\frac{volt.sec}{rad}$$

This step response is later on passed to the parameter estimation toolbox which will tune the above stated parameters to match our DC motor (with gears) step response. The plot generated by the model is shown below

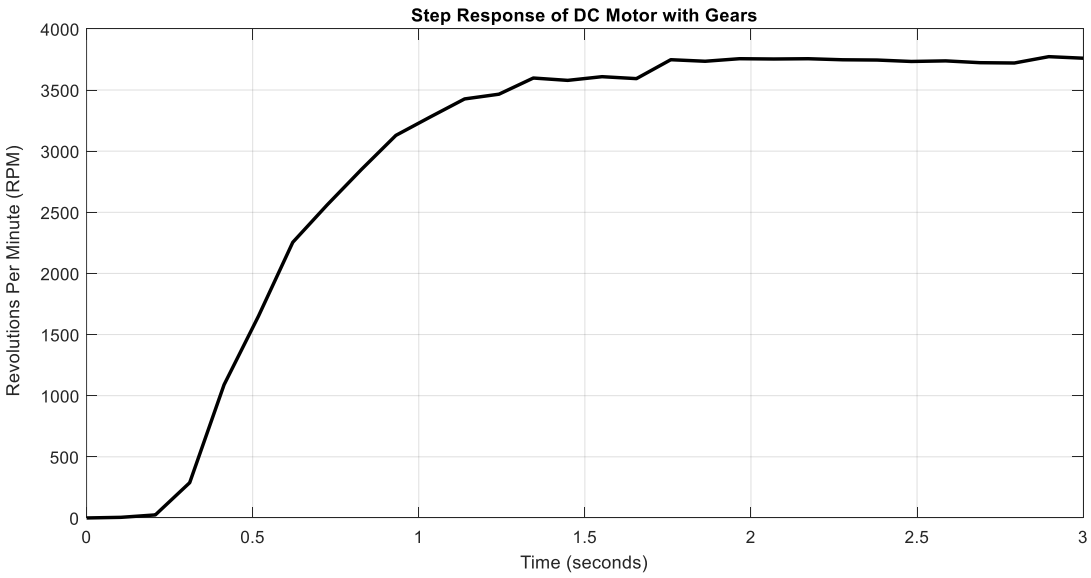


Figure 19: RPM data plot of motor with gears in response to step input voltage

7.2 Processing Data

Since our physical setup with gears wasn't too sturdy the RPM values have abrupt changes in it. For the purpose of making our data more accurate we preprocessed it using the smooth function in MATLAB. The code given below performs this function and the output plot is also attached.

- Import data from Excel
- drop null cells
- Processing and Plotting data

Import data from Excel

```
raw_data = xlsread('DC Motor Data');

data_index_lim = 30;        % index of spreadsheet data that should be read
% Defined separately here because originally 65 entries but we only want
% entries until constant speed
```

drop null cells

```
dc_data = raw_data(1:data_index_lim, :);
```

Processing and Plotting data

```
figure
rpm_gear = smooth (noisy_data_speed);
plot (time,noisy_data_speed,time, rpm_gear);
```

MATLAB Code for Preprocessing Data

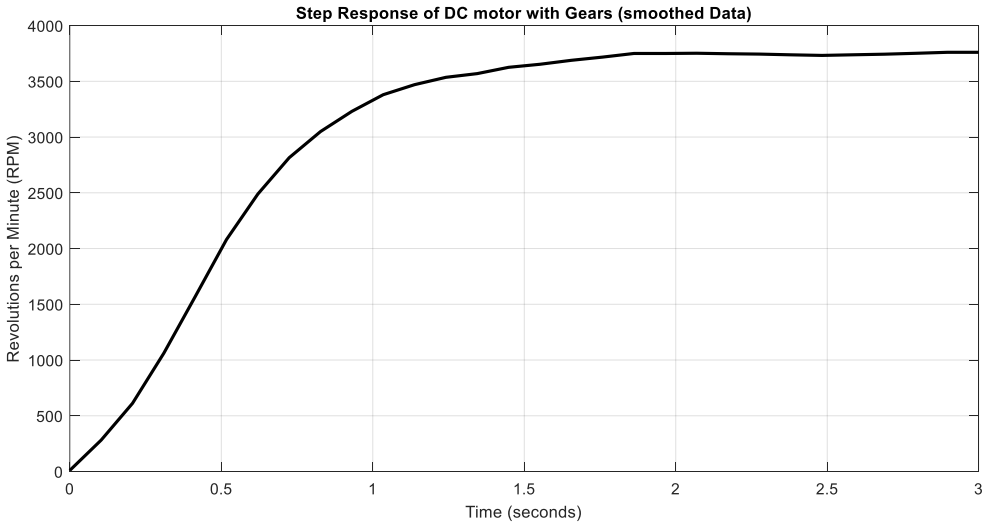


Figure 20: Plot of Processed RPM data

7.3 Parameter Estimation

Using the MATLAB parameter estimation toolbox again we matched our measured RPM of DC motor with that of the Simulink Model. Below are the two plots which shows the procedure. The plot on the left shows the two data sets being matched by MATLAB and the plot on the left details the tuning of the parameters during the estimation process.

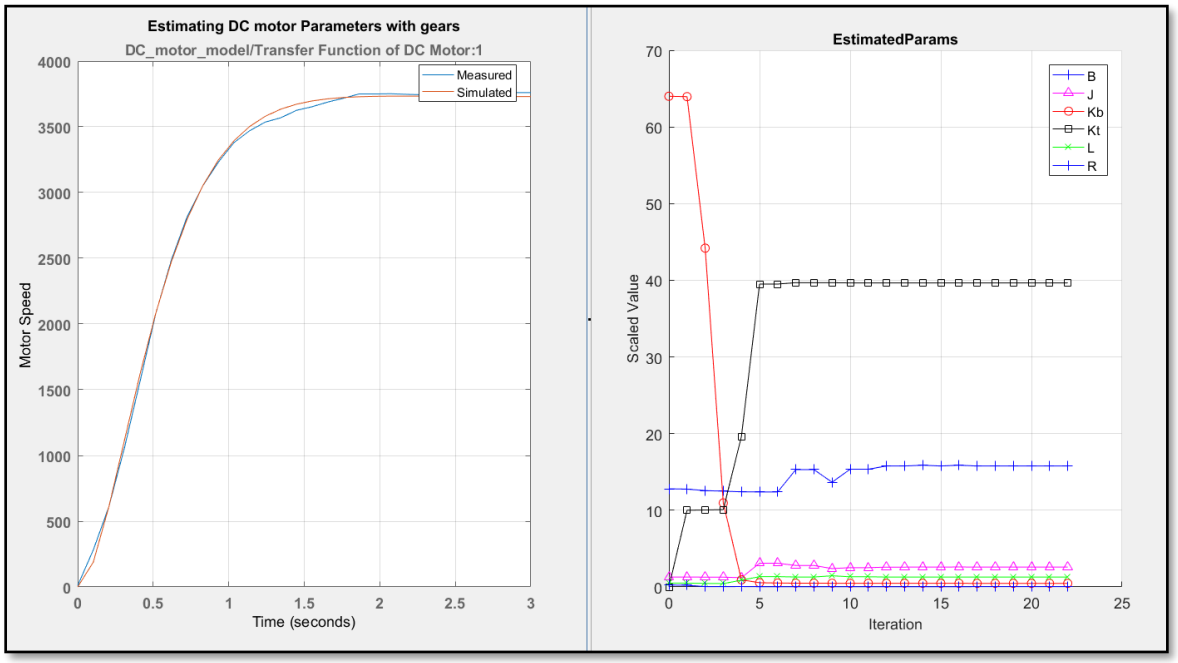


Figure 21: parameter Estimation of DC motor with gears

The results of the experiment are summarized in the tables below:

a) Results of Experiment

Number of Iterations	22
Optimization started	22-Jun-2019 20:00:06
Estimation Converged	22-Jun-2019 24:55:06

b) Parameter Values

Parameters	Old Parameter Values	Estimated Values
Resistance (R)	1	0.1458
Inductance (L)	0.5	1.2944
Moment of Inertia (J)	0.01	0.0202
Constant of Friction (B)	0.1	0.1232
Torque Constant (Kt)	0.01	79.26

Back EMF Constant (Kb)	0.05	0.0036

7.4 Stability Of System By Routh's Criteria

The transfer function of our system is,

$$\frac{\theta(s)}{V(s)} = \frac{K}{JLs^2 + (JR + BL)s + BR + K_t K_b}$$

Putting values J=0.0202, L = 1.2944, R = 1.2994, B = 0.1960 and $K_t = 28.18$ and $K_b = 0.0174$ in transfer function we get

$$\frac{\theta(s)}{V(s)} = \frac{79.26}{0.0261s^2 + 0.1624s + 0.3032}$$

Routh's table for above transfer function is given below

S^2	0.0261	0.3032
S^1	0.1624	0
S^0	0.3032	0

Since all values of first column are positive, therefore it is concluded that there is no right pole and thus, the system is stable.

8. Root Locus

Figure (22) is a diagram with the root loci for the PMDC motor when no gears are coupled to its shaft. Figure (23) shows a similar diagram for the same PMDC motor with gears.

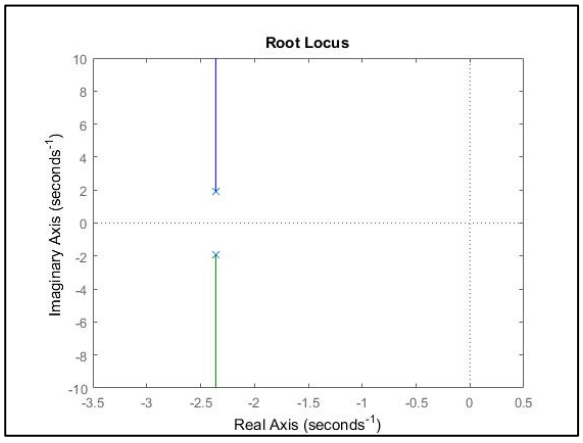


Figure 22: Root Locus Diagram - Without Gears

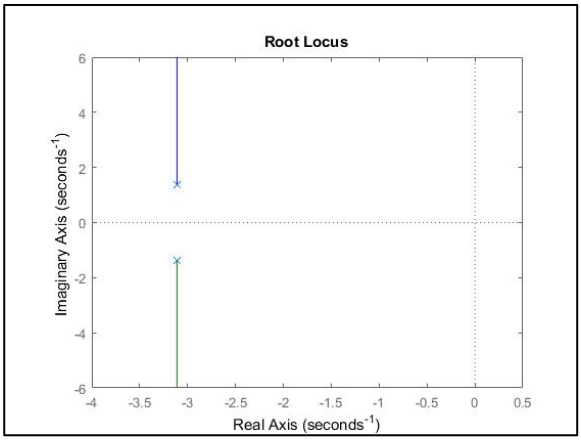


Figure 23: Root Locus Diagram - With Gears

Analysis of Results

- The root locus plot shows how a system's response changes with variation in an arbitrary gain parameter K, which usually represents a controller used to tune the system's response.

- As such, the root locus can be used to ascertain if what range of controller gain values will lead to a response with a desired damping ratio, natural frequency, and frequency of oscillation.
- The root loci for the motor with and without gears is a vertical line on the s-plane, which suggests the variation of the gain parameter affects only the speed of oscillation and not the speed of decay of the system's natural response.
- In both cases, the poles of the system lie on the left hand s-plane, which suggests that the system is stable, regardless of the presence of gears on the motor's shaft.
- As root loci never cross the y-axis in either case, it can be assumed that the PMDC motor is a stable system for the default range of controller gains investigated via MATLAB's rlocus function.
- The presence of gears does seem to have an effect on the speed of the system's natural response.
- Adding gears shifts the poles from ~-2.4 on the σ axis to ~-.3.1. In the time domain, this corresponds to the motor's natural response decaying faster when gears are added, possibly due to the effects of additional inertia and friction on the motor's shaft.
- In both cases, the poles are complex which means the system is underdamped. The addition of gears (and the friction and inertia this entails) therefore does not affect the system enough to change the nature of its damping from underdamped to critically or over damped.
- However, the addition of gears does decrease the frequency of oscillation of the system's response from 2ω to 1ω.
- As the gain increases, the system's damping ratio decreases. Figure (zz) show that the damping ratio decreases from 0.06 to 0.012 as the gain increases from ~0 to ~100.
- Overall, gears have no marked effect on the motor's stability, but do affect the rate of decay of its transient response.
- Increasing the controller gain also has no effect on system stability or the rate of decay of the transient response, but decreases the frequency of oscillations as well and decreases the damping ratio, making the system slightly less underdamped.

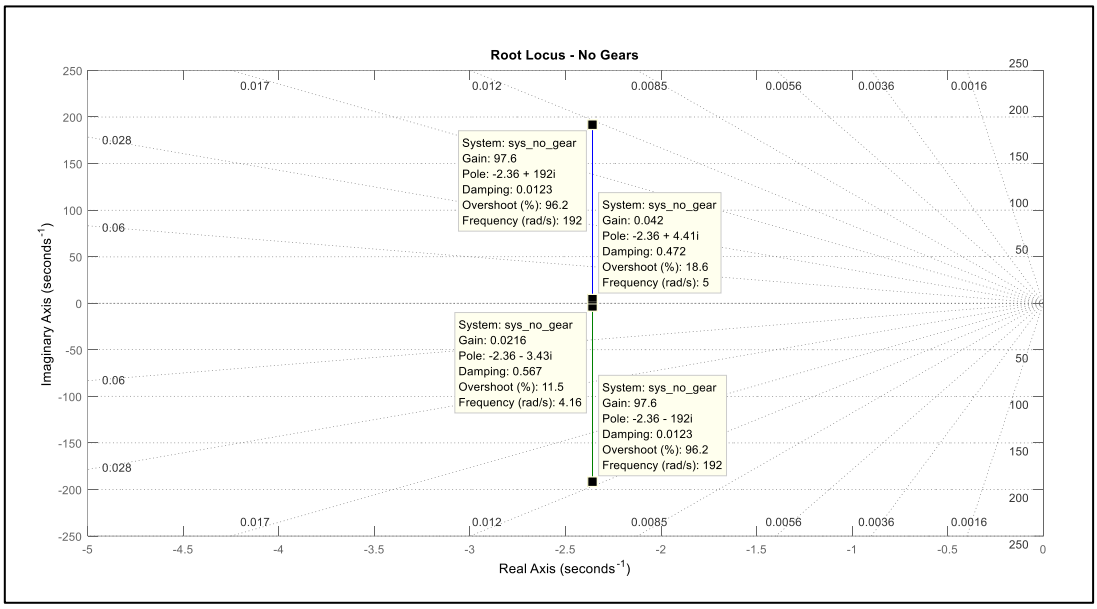
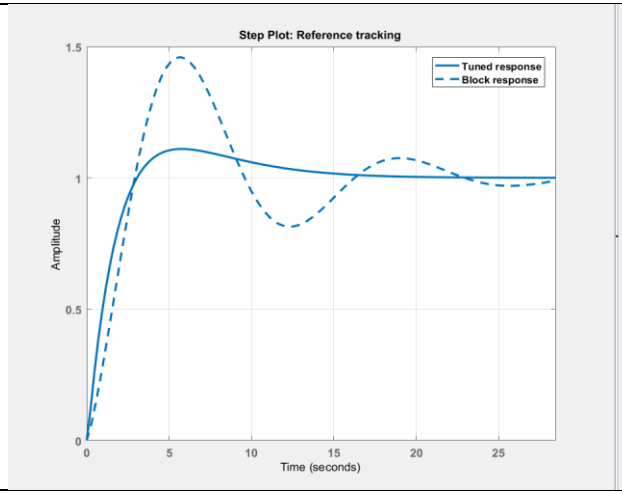
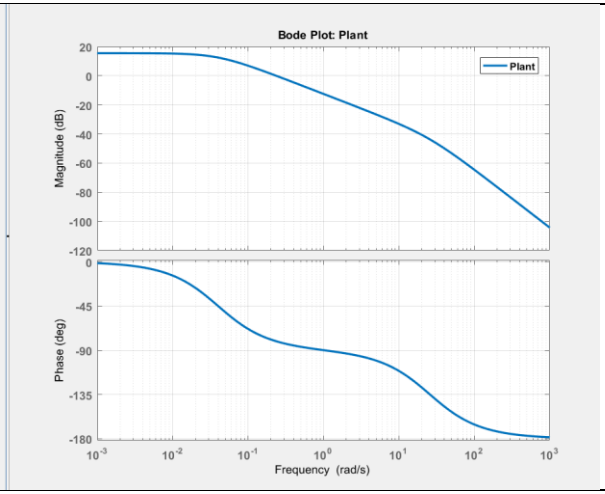


Figure 24: Root Locus without Gears

9. System's response with Various Controllers without Gears

PID





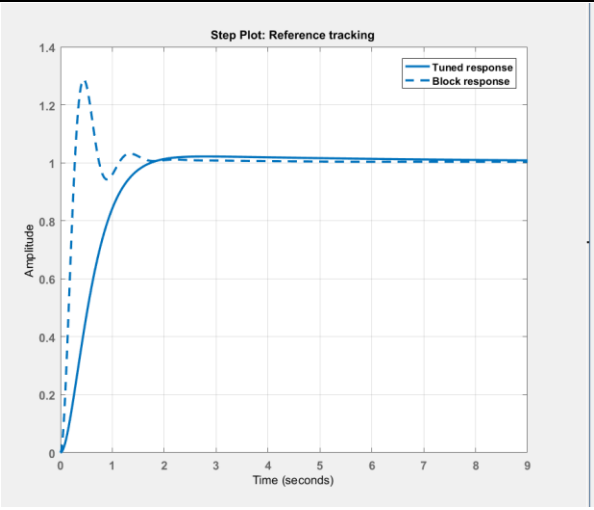
Controller Parameters

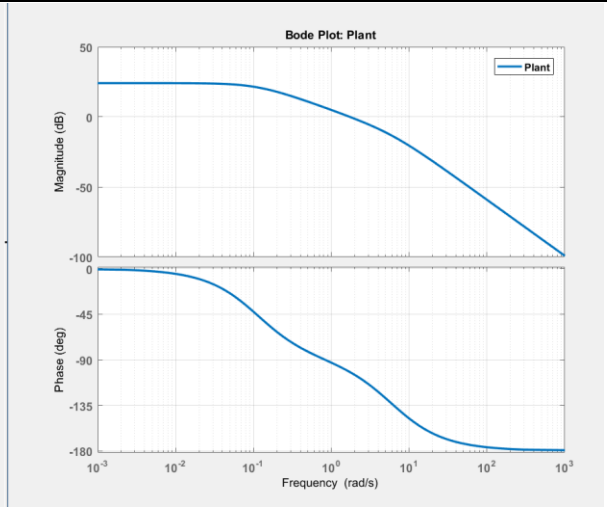
	Tuned	Block
P	1.0028	5.3514
I	0.47363	1.5607
D	0.19147	2.4043
N	2.4324	3.7114

Performance and Robustness

	Tuned	Block
Rise time	0.669 seconds	0.113 seconds
Settling time	5.83 seconds	1.06 seconds
Overshoot	8.85 %	26.2 %
Peak	1.09	1.26
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	74.5 deg @ 2.05 rad/s	38.8 deg @ 11.5 rad/s
Closed-loop stability	Stable	Stable

PI





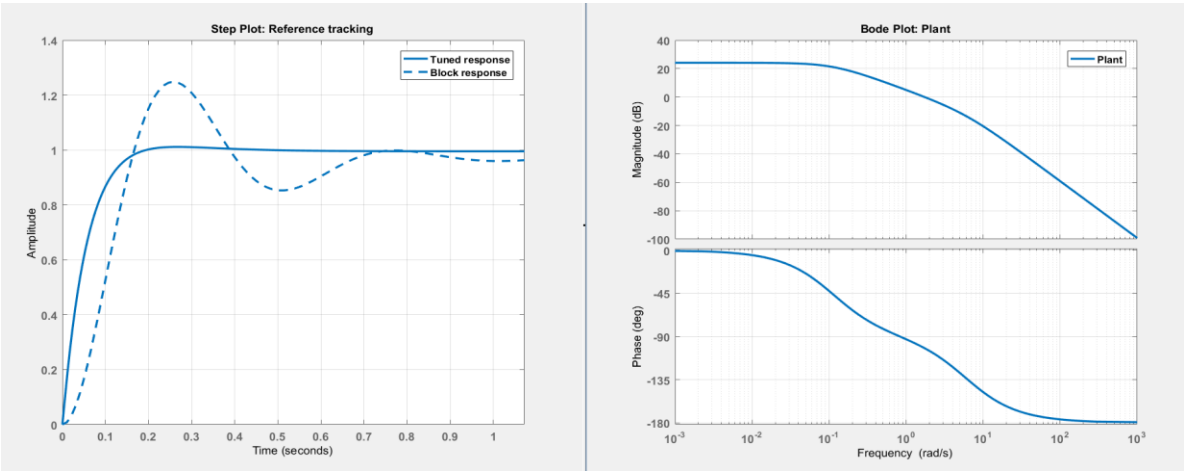
Controller Parameters

	Tuned	Block
P	0.88363	5.3514
I	0.14097	1.5607
D	n/a	n/a
N	n/a	n/a

Performance and Robustness

	Tuned	Block
Rise time	0.981 seconds	0.187 seconds
Settling time	3.63 seconds	1.53 seconds
Overshoot	2.18 %	28.8 %
Peak	1.02	1.29
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	74 deg @ 1.55 rad/s	41.3 deg @ 6.57 rad/s
Closed-loop stability	Stable	Stable

PD



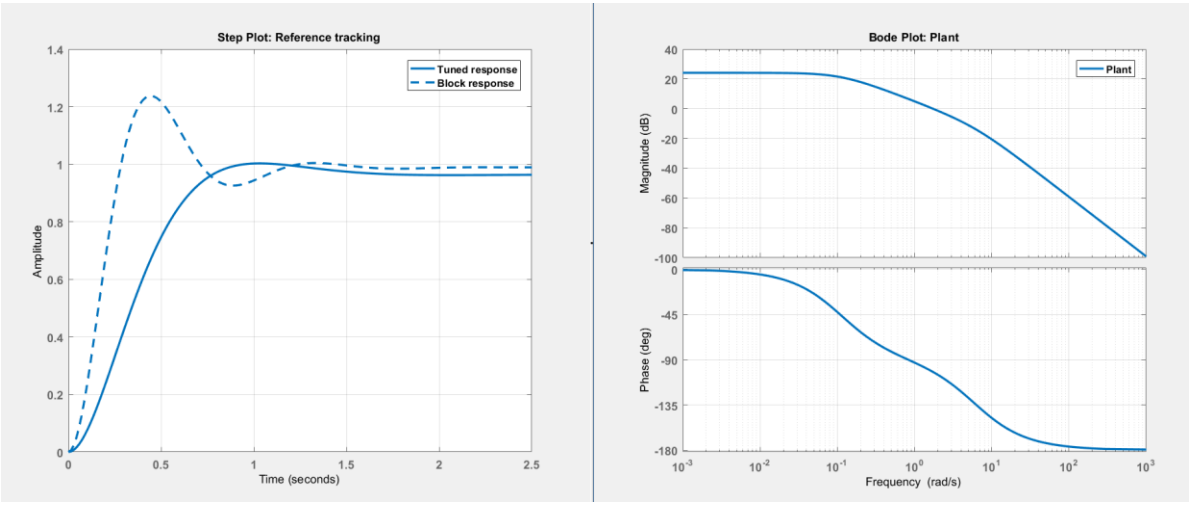
Controller Parameters

	Tuned	Block
P	12.4181	5.3514
I	n/a	n/a
D	1.6471	2.4043
N	2131.5412	3.7114

Performance and Robustness

	Tuned	Block
Rise time	0.104 seconds	0.111 seconds
Settling time	0.157 seconds	1.13 seconds
Overshoot	1.63 %	26.3 %
Peak	1.01	1.25
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	86 deg @ 18.7 rad/s	39.3 deg @ 11.5 rad/s
Closed-loop stability	Stable	Stable

P



Controller Parameters

	Tuned	Block
P	1.6549	5.3514
I	n/a	n/a
D	n/a	n/a
N	n/a	n/a

Performance and Robustness

	Tuned	Block
Rise time	0.499 seconds	0.19 seconds
Settling time	1.37 seconds	1.09 seconds
Overshoot	4.07 %	25.2 %
Peak	1	1.24
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	68.3 deg @ 2.73 rad/s	43.8 deg @ 6.57 rad/s
Closed-loop stability	Stable	Stable

Table 3: Stable responses of DC motor without gears u various controllers

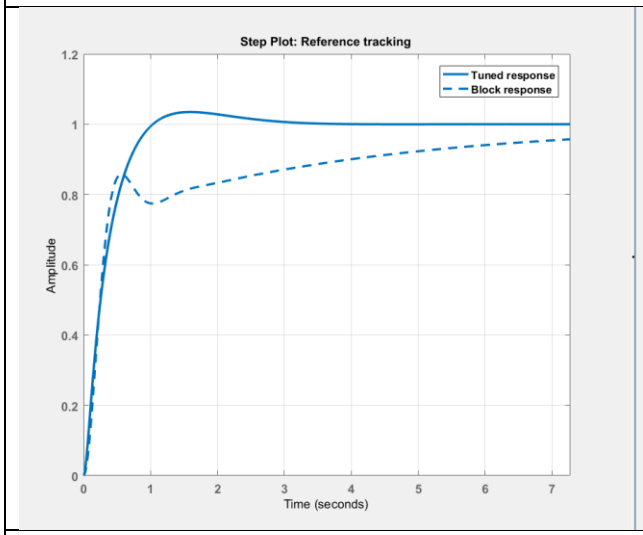
9.1Analysis:

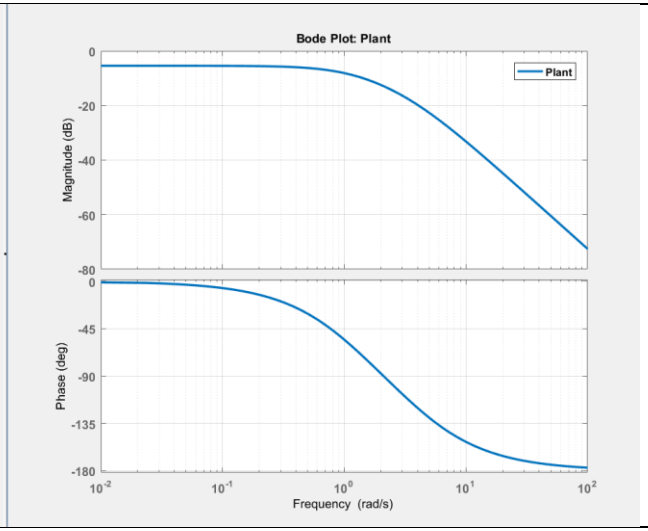
To control the response of the DC motor we have added control parameter in series with the plant transfer function block. The control parameters are changed one by one and in this manner 4 control configurations (basically 4 controller) are used. Auto tuning the controller with built in MATLAB algorithms bears the results shown above. The necessary bode plots generated as a result of using a controller are also attached. We have used 4 type of controllers and presented a comparative analysis below which lets us choose the best possible controller for our system

- After adding gears with the motor the PID controller gives an overshoot of 8.85%. Which is a little higher than the desired 5%. But it still has improved drastically from the actual block response with an overshot of 26.2%.
- The rise time almost 0.7 seconds, which is actually increased from the actual value of 0.113 seconds.
- Same is the case with settling time which has increased from 1.06 to 5.83 seconds.
- The PI controller has a good overshoot% control value which is only 2.18 %, which is the second best in all of the cases.
- But just like with the PID controller, both the rise time and the selling time has increased.
- Just like with the motor's case of without gears, the PD controller has shown the best performance values for controlling out DC motor.
- The over shoot is least among all the controller, which is just 1.63%.
- The rise time and the settling time both have decreased in this case, unlike in the previous two cases. The rise time after tuning is 0.104 seconds and the selling time 0.157 seconds.
- The proportional controller's overshoot has decreased as compared to its block response.
- But the rise time and settling both have increased and now sits at 0.449 and 1.37 seconds respectively.

10. System's response with Various Controllers with Gears

PID





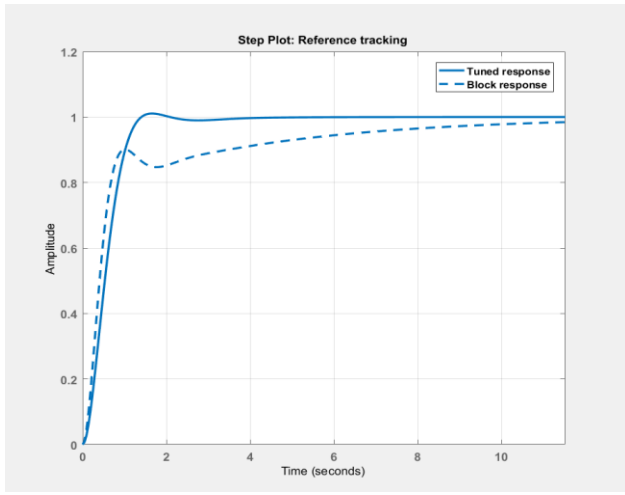
Controller Parameters

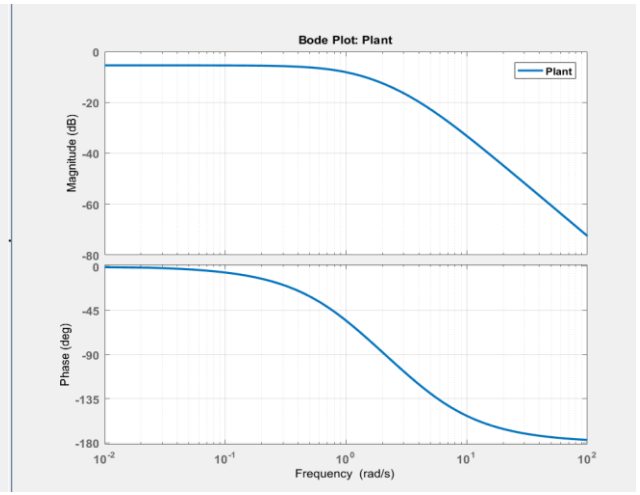
	Tuned	Block
P	5.629	5.3514
I	6.8336	1.5607
D	1.0636	2.4043
N	28.8527	3.7114

Performance and Robustness

	Tuned	Block
Rise time	0.627 seconds	3.9 seconds
Settling time	2.3 seconds	10.2 seconds
Overshoot	3.47 %	0 %
Peak	1.03	0.999
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	81 deg @ 2.75 rad/s	76 deg @ 4.44 rad/s
Closed-loop stability	Stable	Stable

PI





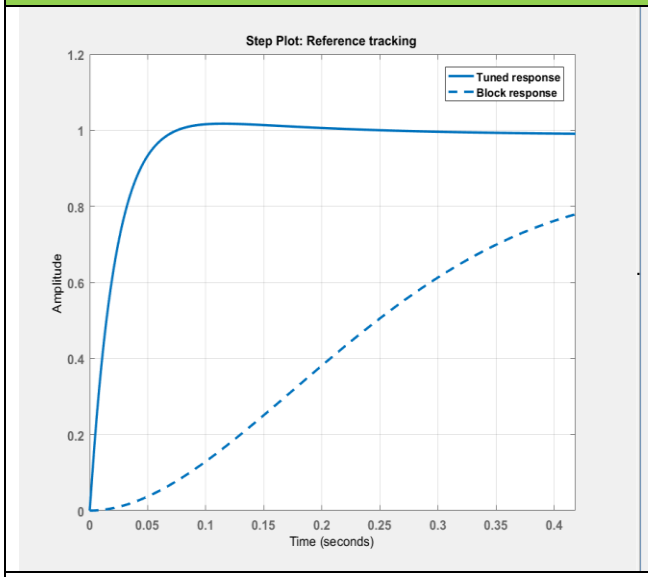
Controller Parameters

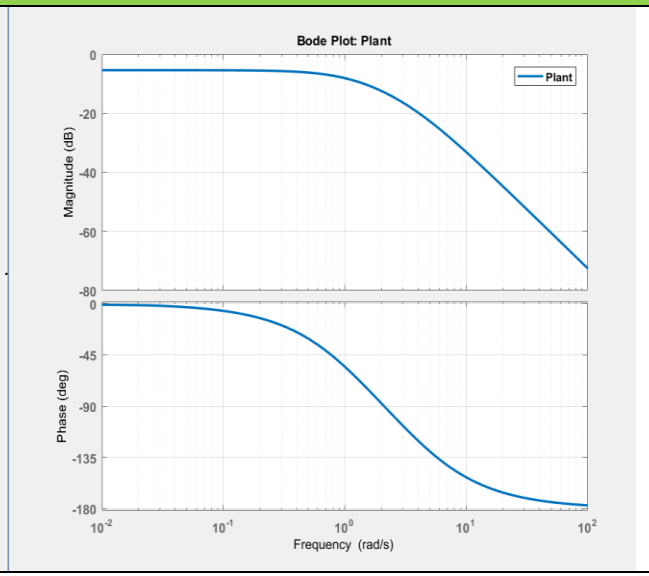
	Tuned	Block
P	3.1846	5.3514
I	3.226	1.5607
D	n/a	n/a
N	n/a	n/a

Performance and Robustness

	Tuned	Block
Rise time	0.831 seconds	0.834 seconds
Settling time	1.26 seconds	10.4 seconds
Overshoot	1.07 %	0 %
Peak	1.01	0.998
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	69 deg @ 1.74 rad/s	74.3 deg @ 2.54 rad/s
Closed-loop stability	Stable	Stable

PD





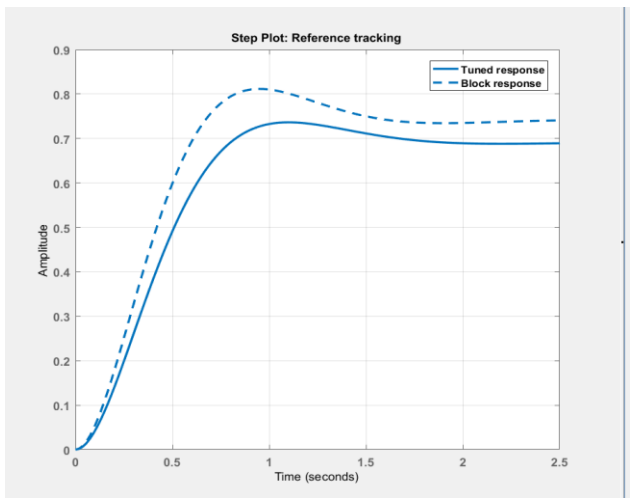
Controller Parameters

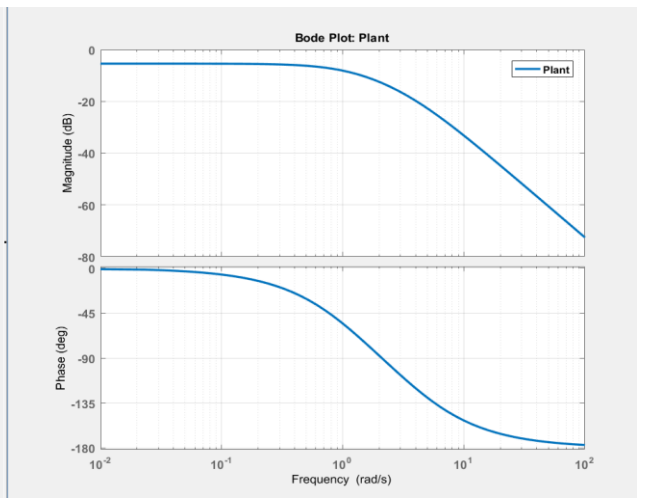
	Tuned	Block
P	142.2266	5.3514
I	n/a	n/a
D	20.0588	2.4043
N	5462.6615	3.7114	

Performance and Robustness

	Tuned	Block
Rise time	0.0397 seconds	0.257 seconds
Settling time	0.193 seconds	1.2 seconds
Overshoot	3.05 %	11.9 %
Peak	1.02	0.829
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	87 deg @ 47.8 rad/s	77.1 deg @ 4.48 rad/s
Closed-loop stability	Stable	Stable

P





Controller Parameters

	Tuned	Block
P	4.1807	5.3514
I	n/a	n/a
D	n/a	n/a
N	n/a	n/a	

Performance and Robustness

	Tuned	Block
Rise time	0.528 seconds	0.449 seconds
Settling time	1.59 seconds	1.44 seconds
Overshoot	6.56 %	9.49 %
Peak	0.736	0.812
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	92.6 deg @ 1.99 rad/s	81.2 deg @ 2.52 rad/s
Closed-loop stability	Stable	Stable

Table 4: Stable responses of DC motor with gears using various controllers

10.1 Analysis:

- The block response after adding a PID controller has huge response dip just before the 1 second mark. After tuning the PID parameters the response is smoothen out.
- The rise time improved from 3.9 seconds to just 0.627 second sand settling time has also improved from 10.2 to 2.3 seconds.
- Although the overshoot of the system has increased from 0 to 3.47%, it is still within the desired limit of 5%.
- The output after adding and tuning the PID controller is stable as termed by MATLAB.
- In case of PI controller the block response is the same, but the tuned response have changed and is comparatively better.
- The overshoot has gone down to 1.07%, similarly the settling time has also improved to 1.26 seconds.
- But there improvements have increased our rise time to 0.831 seconds which was 0.627 seconds in the previous case.
- The response of the PD controller is the best among them all. The overshoot is just 3.5% within the desired stable range.
- The rise time is 0.0397 seconds and settling time is 0.193 seconds. Both of which are lower than the all the other cases.
- The response of the PD controller is termed as stable by MATLAB.
- The proportional controller has higher overshoot value of 6.56%. This makes the controller's response highly fluctuating in the beginning.
- The settling time is also higher and is about 1.59 seconds. While the rise time is 0.528 seconds.
- Hence the PD controller is the best tradeoff between our system's performance parameters and is suitable for our DC motor control.

11. Results

In this CEP, the speed control of a PMDC motor through a variable DC supply voltage was investigated.

For an excitation voltage of 14.4 V, the motor was found to have steady state speeds of ~600 RPM and ~3600 RPM without and with gears at the shaft respectively. This proves that an appropriate choice of gears and gear ratios can help increase or decrease the speed of rotation for a practical application without exceeding the rated speed of the PMDC motor.

Parameters	Without Gears	With Gears
Resistance (R)	1.0047	0.1458
Inductance (L)	1.5537	1.2944
Moment of Inertia (J)	0.0482	0.0202
Constant of Friction (B)	0.1960	0.1232
Torque Constant (Kt)	28.18	79.26
Back EMF Constant (Kb)	0.0174	0.0036

Table 5: Results of DC motor Modeling Experiment

Parameter estimation results show that the addition of gears leads to a marked decrease in effective armature circuit resistance and minor decrease in armature inductance. The torque constant increases drastically with the addition of gears, which makes sense as more torque is required to produce the turning effect needed to rotate interlocked gears. The moment of inertia is almost halved, suggesting that the presence of gears makes it easier for the rotor shaft to change its state of rest or motion, which may be related to the decrease in the coefficient of friction. The back EMF constant decreases by almost 5 times, suggesting that the addition of gears decreases the opposing EMF generated in the motor's armature circuit, which would also explain why the effective armature resistance has decreased: with a larger effective armature voltage (due to lower counter EMF), the current through the armature circuit becomes larger, which MATLAB interprets as a lower armature resistance.

Routh table and root locus for the PMDC motor (with and without gears) also showed that the PMDC motor is a stable, underdamped, second order control system. The addition of gears seems to increase the frequency of exponential decay in the motor's time response, but has no effect on stability. Indeed, for controller gains ranging from 0.1 - 100, the system remains stable although the rate of its oscillations increases substantially while the damping ratio decreases.