MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Development of New Model Following Robust Motion Controller

Selin Kocaoğlu 150420037

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Assoc. Prof. Dr. Uğur Tümerdem
ISTANBUL, 2025

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Development of New Model Following Robust Motion Controller

by

Selin Kocaoğlu Jan 26, 2025 Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) ....

Department of Mechanical Engineering

Certified By ....

Project Supervisor, Department of Mechanical Engineering

Accepted By..

Head of the Department of Mechanical Engineering

Table of Contents

1) ABSTRACT

The objective of Development of New Model Following Robust Motion Controller is to design a new robust motion controller and disturbance observer. The proposed motion controller can be used for position, force, impedance control of robots. The observer can be used for external force estimation. The controller will be evaluated theoretically, in simulation using Matlab, and also on actual electric motors and robots.

2) INTRODUCTION

In the Development of New Model Following Robust Motion Controller project, it is planned to develop an original model following robust motion control system for motion control. Motion control is a general name used especially in robotic and mechatronic systems for position, speed, acceleration and force control. Due to the nonlinear dynamics of robotic systems and their interaction with unknown environments, successful disturbance compensation is required in these systems. For this purpose, in the literature, disturbance observers and robust acceleration controllers that can control position, speed, acceleration, force values and against model uncertainties and external disturbances are widely used. However, these algorithms are designed to send acceleration and torque commands to robots and require high performance internal speed measurement.

In this study, a controller that will only take motor position measurements via encoders and can perform other motion control functions (such as force control) by controlling motors and robots that can only operate in position control mode will be developed. Since this controller will be designed as a model following controller, the differences between the model and the system output can also be estimated and compensated for the disturbance effects.

The most researched methods in the development and modelling of model-following robust motion control systems are DOB and Adaptive control methods. DOB is a method that aims to minimize the effects of external disturbances and modelling errors on the system by predicting them using a controller. Adaptive Control is a method mostly used in the aviation field. It is a method used to maintain system performance when the parameters of the controlled system change over time or are not fully known.

Articles, books and videos used in the literature research are listed in the reference section. In addition, websites such as Academia, Science Direct, Google Scholar, where the articles are located, were used in the research.

3) PD CONTROLLER

PD (Proportional-Derivative) controller is a basic control algorithm used in control systems. This controller aims to reach the target state by using the system's error signal (difference between the reference and the current state). PD controller consists of proportional (P) and derivative (D) components.

General Structure of PD Controller The equation of PD controller

$$u(t) = Kp * e(t) + Kd * \frac{de(t)}{dt}$$

(3. 1)

where

$u(t)$: Control signal (output applied to the system).

$e(t)$: Error signal, $e(t) = r(t) - y(t)$ where $r(t)$ is the reference signal and $y(t)$ is the output of the system.

Kp: Proportional gain, contributes to the control signal according to the magnitude of the error.

Kd: Derivative gain, responds to the rate of change of the error and prevents the system from overreacting or oscillating.

<u>Theoretical Basis of PD Controller</u>

Proportional (P) Component

The proportional component depends on the current value of the error signal. Equation:

$$uP(t) = Kp * e(t) \tag{3. 2}$$

Purpose: It reduces the system error directly and allows it to approach the target. However, when only the P controller is used, a steady-state error usually occurs in the system.

Derivative (D) Component

The derivative component responds to the rate of change (derivative) of the error. Equation

$$uD(t) = Kd * \frac{de(t)}{dt} \tag{3. 3}$$

Purpose: To reduce the sensitivity of the system to sudden changes or oscillations and to make the system more stable. Thanks to the derivative gain, the system intervenes early by predicting the future error.By taking the Laplace transform of the equation in the time domain, the transfer function can be obtained. The PD equation in the time domain

$$u(t) = Kp * e(t) + Kd * \frac{de(t)}{dt} \tag{3. 4}$$

Laplace transform

$$U(s) = Kp * E(s) + Kd * s * E(s)$$

PD transfer function

$$GPD(s) = Kp + Kd * s \tag{3. 5}$$

Advantages and Disadvantages of PD Controller

Advantages:

- Speeds up system response.
- Reduces oscillations and overshoots.
- Increases stability and reduces system overshoot.

Disadvantages:

- Does not completely eliminate residual error
- Can be sensitive to noise because the derivative component can amplify high frequency noise.

In the design of a PD controller, the selection of Kp and Kd gains is important. These gains determine the pole-saliency placements and stability of the system. Root locus and Bode diagram analyses are performed on the open-loop and closed-loop transfer functions of the system to check the suitability of the controller.

4) POSITION CONTROL

4.1 Disturbance Observer

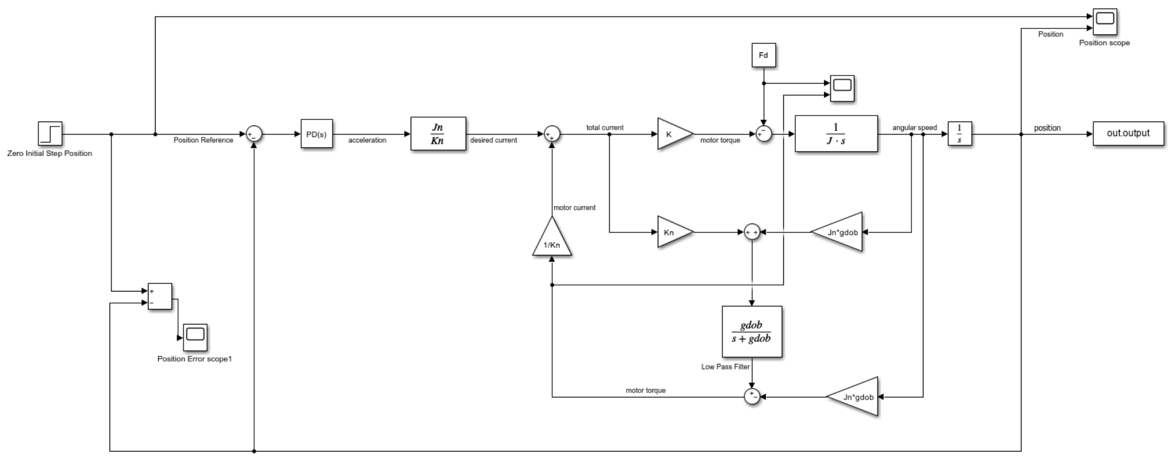


Figure 1: Position Control DOB on Simulink

In 1938, DOB was first proposed by Ohnishi et al. at the IPEC conference. After that, it has been used in various robust motion control applications such as robotics, industrial automation, automotive, etc. DOB estimates external disturbances and system uncertainties, such as friction, parameter variations, etc., and the robustness of the motion control system is achieved by feeding back the estimated disturbances in an inner loop. In order to achieve control objectives, such as position, force, or admittance control objectives, an outer loop controller is designed considering the nominal plant parameters since a DOB can nominalize an uncertain plant.

In the design of DOB, a low pass filter (LPF) is used, the inner loop of which matches the nominal inertia of the motor and the torque coefficient. The dynamic characteristics of the

LPF of DOB, the ratio between the uncertain and nominal plant parameters, significantly change the stability and robustness of a DOB-based motion control system. It is a well-known fact that it is desirable to set the bandwidth of the LPF of DOB as high as possible to estimate and compensate for disturbances over a wide frequency range. However, the bandwidth is limited by practical, i.e., noise and robustness constraints. The Development of a New Model Following the Robust Motion Controller Project aims to eliminate the performance limitation caused by LPF in high-performance systems.

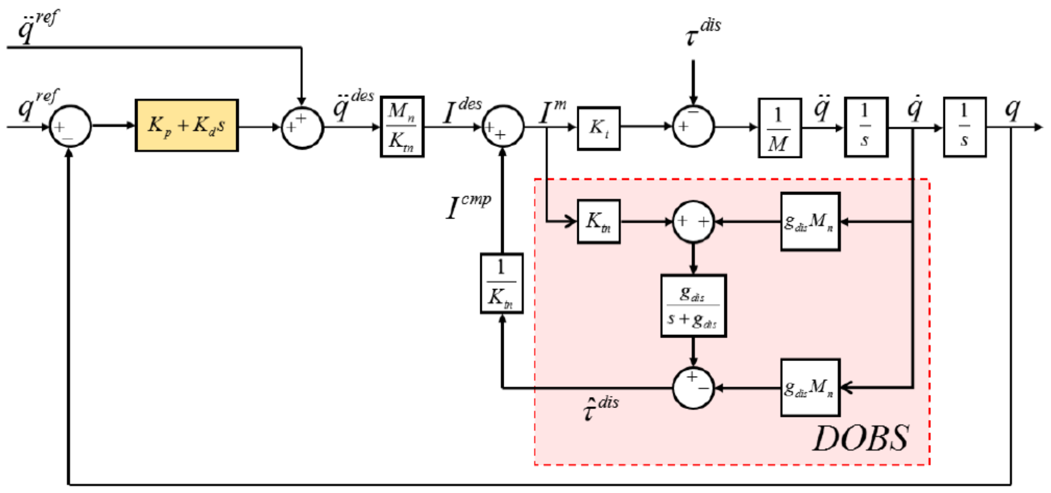


Figure 2: Position Control System based on PD controller

When the DOB equations are examined in detail,

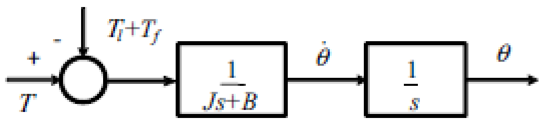


Figure 3: Block Diagram of servo motor

$$J\ddot{\theta} = T_m - T_l \tag{4. 1}$$

Where;

T: $Motor\ torque$

T_l: $Load\ torque$

J: $Motor\ inertia$

$\ddot{\theta}$: *Angular acceleration*

The load torque is considered as,

$$T_l = T_{int} + T_{ext} + (T_f + B\dot{\theta}) \tag{4. 2}$$

T_{int} the inertial torque, derived from the Lagrange motion equations is consists of inertia torque and gravity effect. T_{ext} consists of the torque external to the system.

Friction term $T_f + B\dot{\theta}$ is the sum of viscosity and coulomb terms. For a high gain current controller, input current can be assumed as the reference current. Therefore, for a DC servo motor,

$$J\ddot{\theta} = K_t I_f - (T_{int} + T_{ext} + T_f + B\dot{\theta}) \tag{4. 3}$$

Above equation has two parameters namely J and torque constant K_t. Inertia can be changed due to the mechanical configuration of the system.

$$J = J_n + \Delta J \tag{4. 4}$$

Similarly parameter K_t may change

$$K_t = K_{tn} + \Delta K_t \tag{4. 5}$$

Where J_n and K_{tn} are nominal inertia and the nominal torque constant of the motor respectively.

Disturbance torque T_{dis} is represented as,

$$T_{dis} = T_l + \Delta J\ddot{\theta} - \Delta K_t I_{Ref} \tag{4. 6}$$

Dynamıc equation yields,

$$(J_n + \Delta J)\ddot{\theta} = (K_{tn} + \Delta K_t)I_{Ref} - T_l \tag{4. 7}$$

By rearranging,

$$J_n\ddot{\theta} = K_{tn}I_{Ref} - T_{dis} \tag{4. 8}$$

Thus T_{dis} can be calculated as follows.

$$T_{dis} = K_{tn}I_{ref} - J_n\ddot{\theta} \tag{4. 9}$$

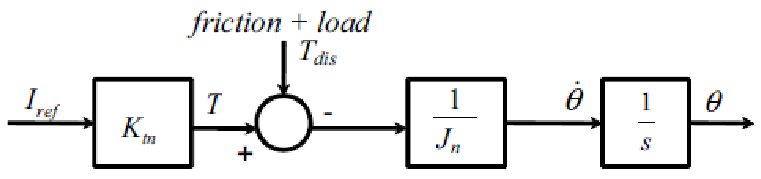


Figure 4: DC Motor with Disturbance Torque

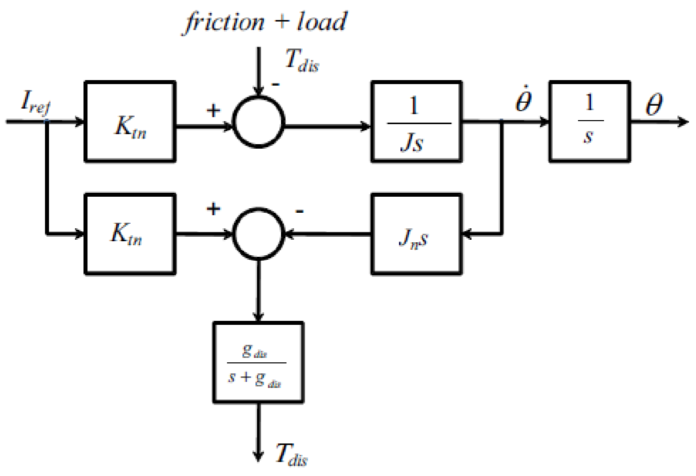
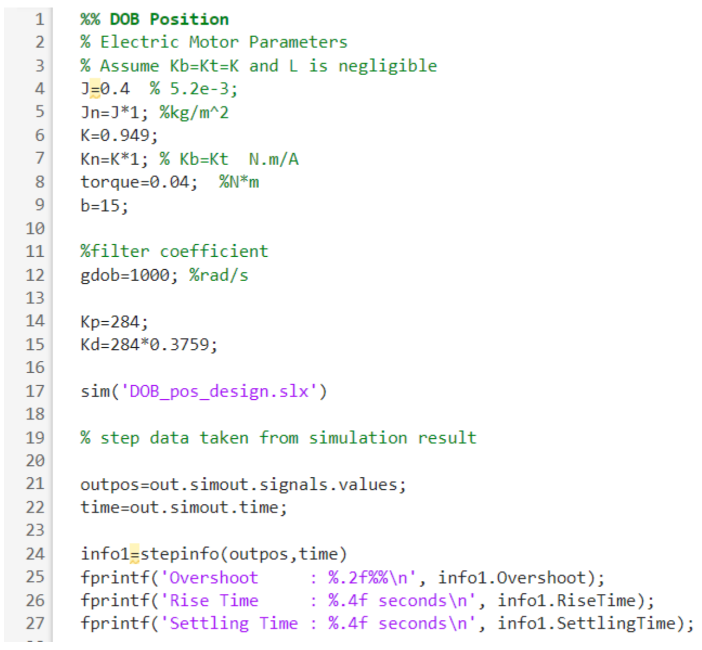


Figure 5: Disturbance based on acceleration with LPF

4.1.1 Simulink Model and Simulation Result

Before the Simulink analysis, the parameters of the STA1104-1116 actuator used in experimental studies in the intelligent machines laboratory were defined in MATLAB script.



```
%% DOB Position
% Electric Motor Parameters
% Assume Kb=Kt=K and L is negligible
J=0.4  % 5.2e-3;
Jn=J*1; %kg/m^2
K=0.949;
Kn=K*1; % Kb=Kt  N.m/A
torque=0.04;  %N*m
b=15;

%filter coefficient
gdob=1000; %rad/s

Kp=284;
Kd=284*0.3759;

sim('DOB_pos_design.slx')

% step data taken from simulation result

outpos=out.simout.signals.values;
time=out.simout.time;

info1=stepinfo(outpos,time)
fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

Figure 6: STA1104-1116 actuator parameters

For simulation results, DOB position model was established with block diagrams in Simulink.

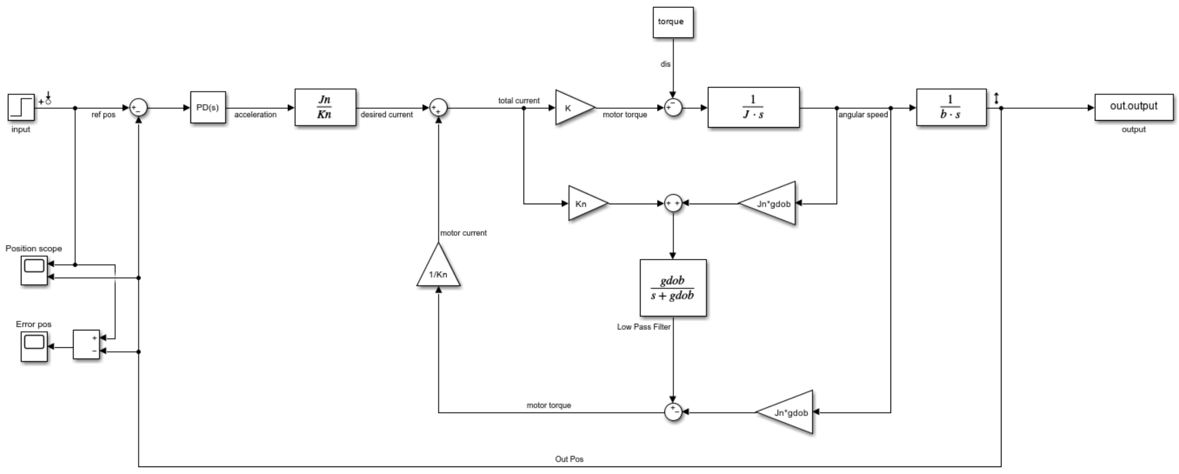


Figure 7: DOB position control model

In this system, two most important results were evaluated, one of which was the comparison of the input and output positions. The positions and the error between them were observed with scopes. The simulation results are as follows.

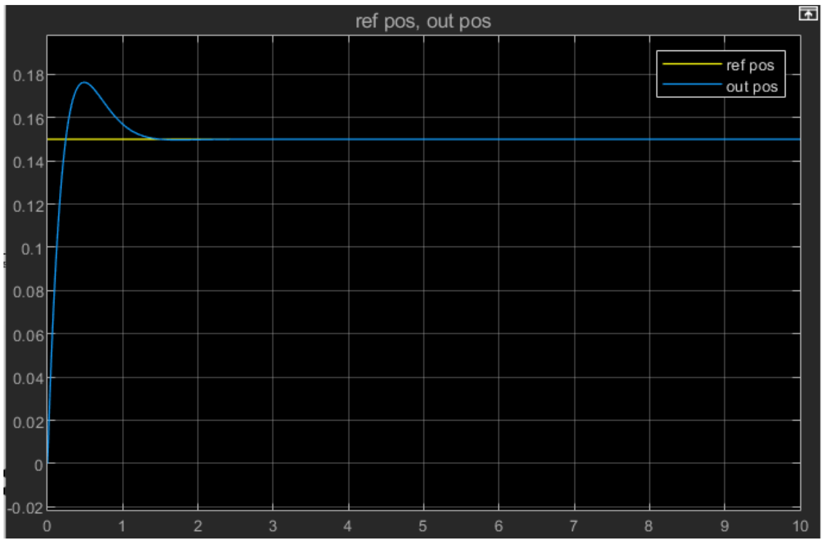


Figure 8: Result of in and out position

Result obtained from the Workspace block in Simulink calculates the time response characteristics of the system with the stepinfo command in the MATLAB Script.

Overshoot : 17.55%

Rise Time : 0.1821 seconds

Settling Time: 1.1668 seconds

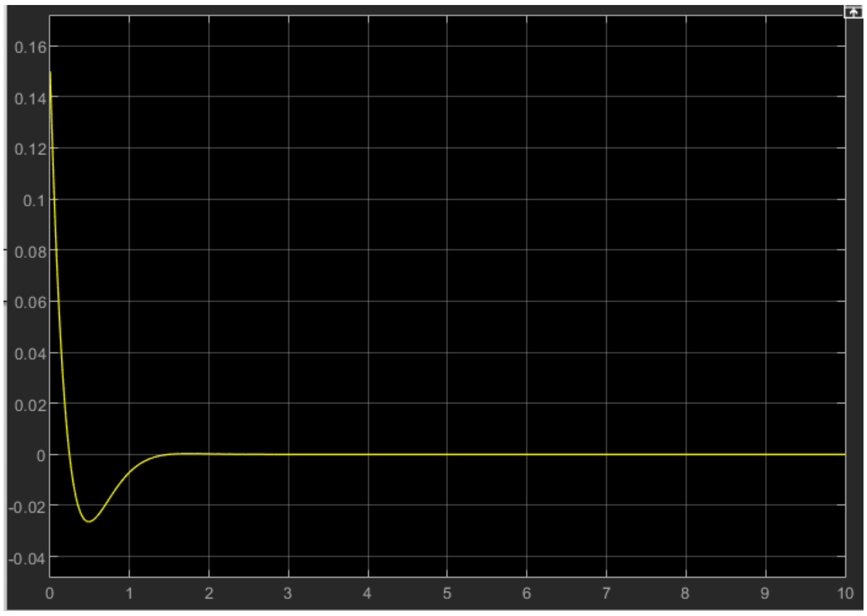


Figure 9: Error between in and out position

Another important result is the estimation of the disturbance entering the system and the disturbance in the system. The disturbances and the error between them are observed with scopes. The simulation results are as follows.

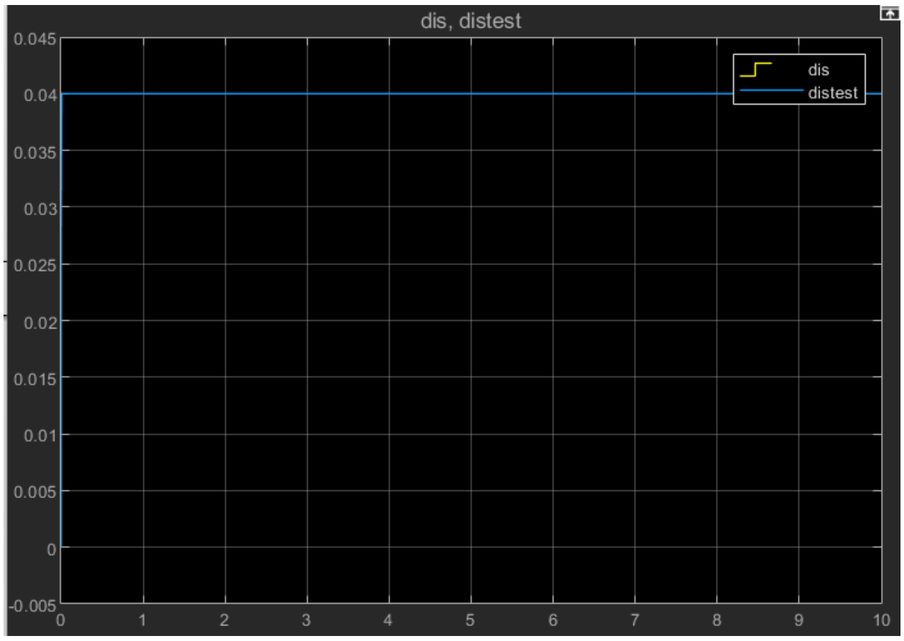


Figure 10: Result of disturbance and disturbance estimation

Zoomed in on the result to see the initial response of the system.

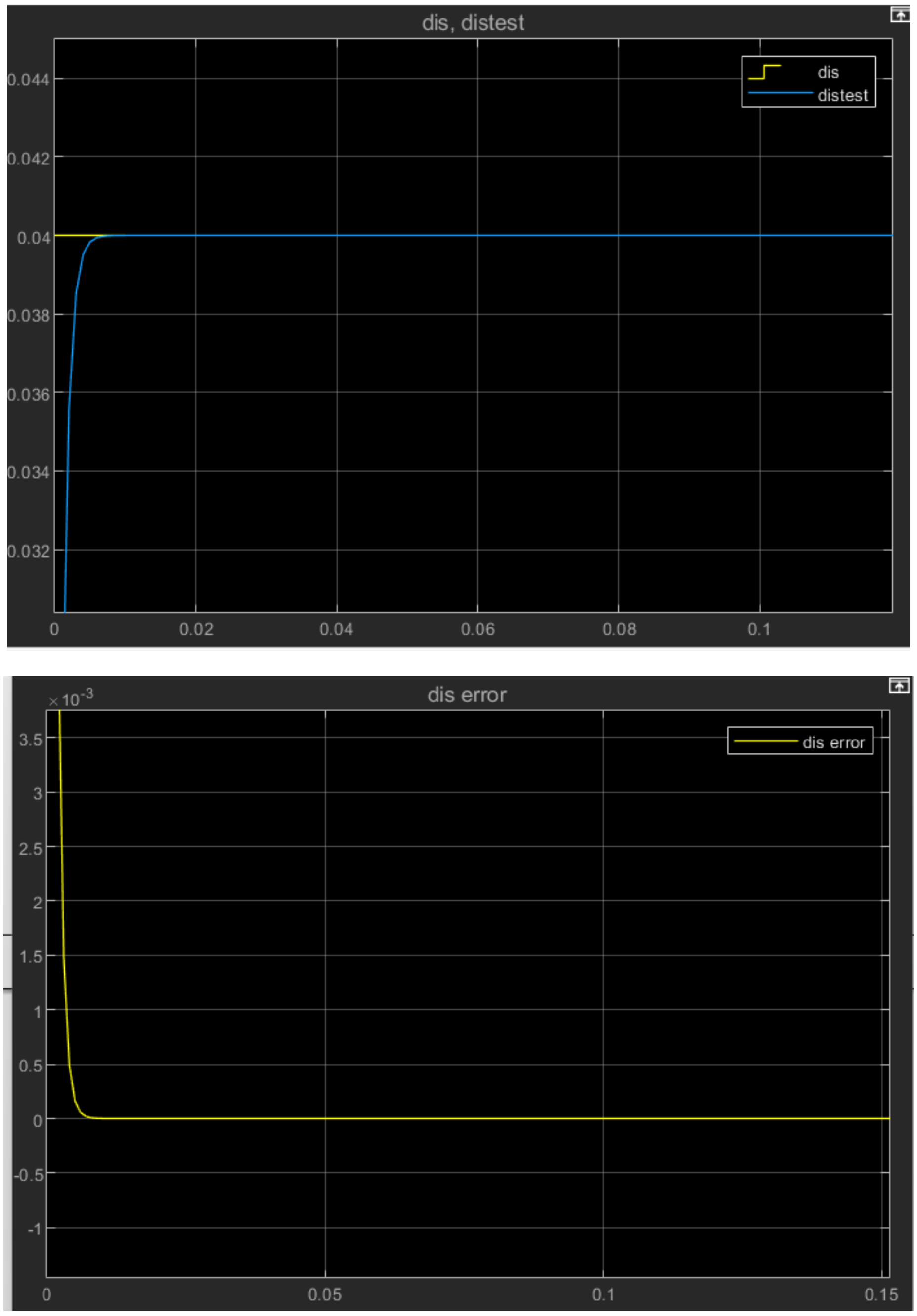


Figure 11: Error of disturbance and disturbance estimation

4.1.2 Experimental Model and Experiment Result

Experiments were carried out in the intelligent machines laboratory. The stages and the results obtained are described below. DOB position control was modelled in the Simulink environment and integrated into the physical system with a QUARC card. The motor data in the system is transmitted to the motor in the physical environment via the QUARC HIL Write Analog block, and the position information of the motor is measured with the encoder and transferred to the Simulink environment using the HIL Read Encoder block. The results are observed with scope blocks.

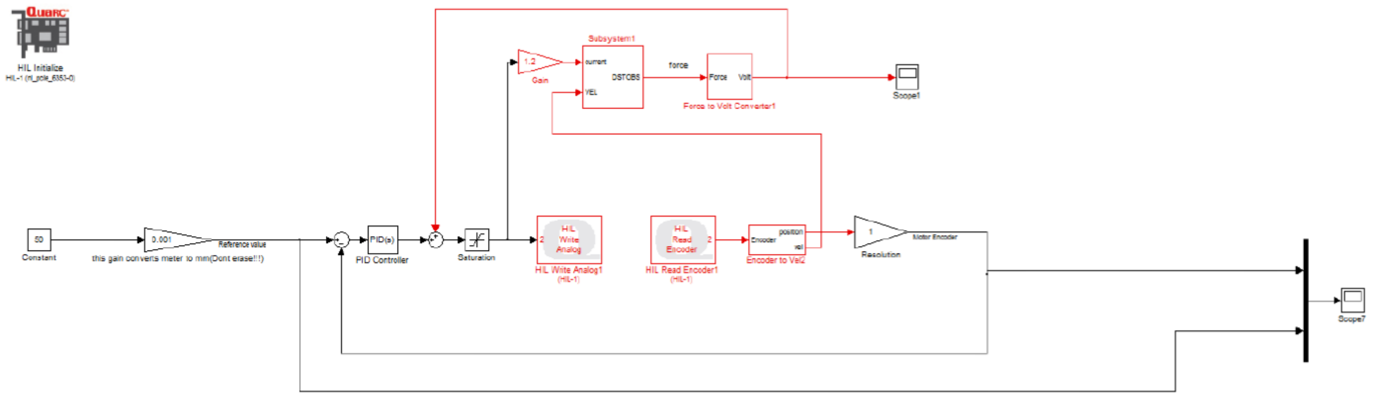
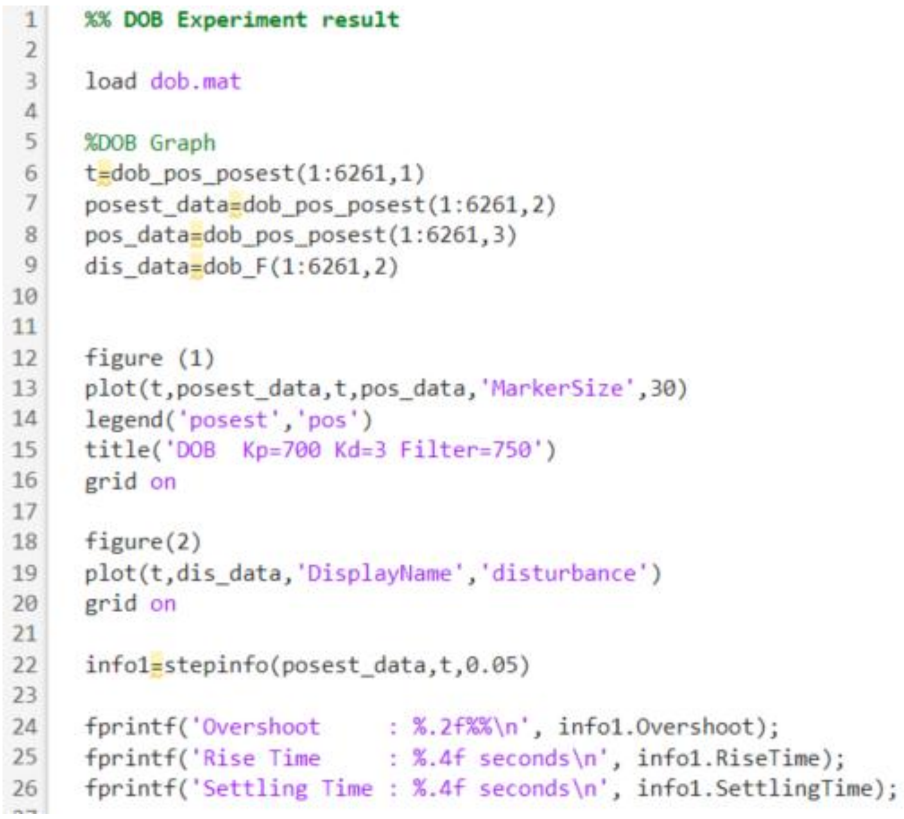


Figure 12: Experimental setup of DOB position

Matlab code written to observe the experimental data



```
%% DOB Experiment result

load dob.mat

%DOB Graph
t=dob_pos_posest(1:6261,1)
posest_data=dob_pos_posest(1:6261,2)
pos_data=dob_pos_posest(1:6261,3)
dis_data=dob_F(1:6261,2)


figure (1)
plot(t,posest_data,t,pos_data,'MarkerSize',30)
legend('posest','pos')
title('DOB  Kp=700 Kd=3 Filter=750')
grid on

figure(2)
plot(t,dis_data,'DisplayName','disturbance')
grid on

info1=stepinfo(posest_data,t,0.05)

fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

Figure 13: Experimental result

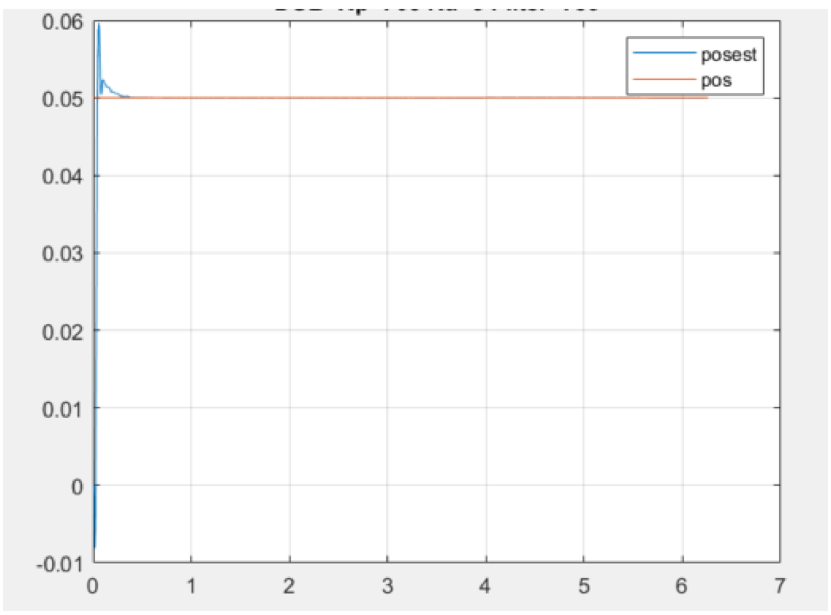


Figure 14: DOB Position result

Overshoot : 19.25%

Rise Time : 0.0139 seconds

Settling Time: 0.1798 seconds

4.2 New Controller Method

4.2.1 Mathematical Modeling

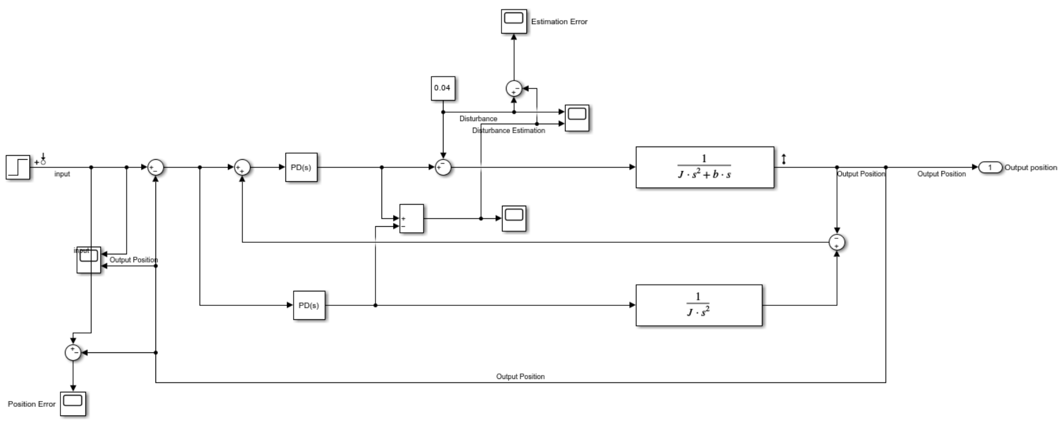
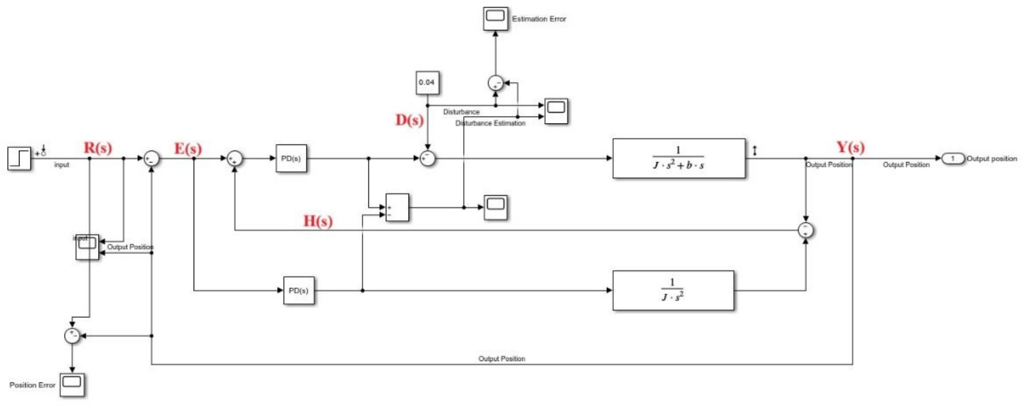


Figure 15: Mathematical Modelling of NCM



P: Plant transfer function

C: Controller transfer function

D: Disturbance signal

E: Error signal

Firstly, the transfer function between Y(s) and R(s) is found.

The signals R, E, D, H and Y are assigned which makes it easier to find the transfer function equation

$$H(s) = Y(s) - E(s) * \widetilde{C} * \widetilde{P} \qquad (4.4)$$

E(s) equation is the error signal. The equation is as follows:

$$E(s) = R(s) - Y(s) \qquad (4.5)$$

Finally, the equation of the output signal Y(s) is as follows:

$$Y(s) = P * [F(s) * C - D(s)] \qquad (4.6)$$

The equation solutions are as follows, firstly the relationship between the F signal and the R and Y signals is found.

$$F(s) = E(s) - H(s)$$

Substitute E(s) into the 2nd equation

$$F(s) = [R(s) - Y(s)] - H(s)$$

Substitute equation 1 for H(s)

$$F(s) = [R(s) - Y(s)] - \left[Y(s) - E(s) * \tilde{C} * \tilde{P}\right]$$

$$F(s) = [R(s) - Y(s)] - \{Y(s) - [R(s) - Y(s)] * \tilde{C} * \widetilde{P}\}$$

The equation is arranged

$$F(s) = R(s) - 2Y(s) + R(s) * \tilde{C} * \widetilde{P} - Y(s) * \tilde{C} * \widetilde{P}$$

$$F(s) = R(s) * (1 + \tilde{C} * \widetilde{P}) - Y(s) * (2 + \tilde{C} * \widetilde{P}) \qquad (4.7)$$

If the F(s) equation – 4.7 equation - is substituted into the Y(s) equation – 4.6 equation - the relationship between the Y signal and the R and D signals is found

$$Y(s) = P * [F(s) * C - D(s)]$$

$$F(s) = R(s) * (1 + \tilde{C} * \widetilde{P}) - Y(s) * (2 + \tilde{C} * \widetilde{P})$$

$$Y(s) = P * \{[R(s) * (1 + \tilde{C} * \widetilde{P}) - Y(s) * (2 + \tilde{C} * \widetilde{P})] * C - D(s)\}$$

The equation is arranged and distributed in parentheses P

$$Y(s) = [R(s) * (1 + \tilde{C} * \widetilde{P}) - Y(s) * (2 + \tilde{C} * \widetilde{P})] * C * P - D(s) * P$$

Y signals are collected on the other side of the equation

$$Y(s) = [R(s) * (1 + \tilde{C} * \widetilde{P}) - Y(s) * (2 + \tilde{C} * \widetilde{P})] * C * P - D(s) * P$$

$$Y(s)(1 + 2 * C * P + \tilde{C} * \widetilde{P} * C * P) = [R(s) * (C * P + \tilde{C} * \widetilde{P} * C * P)] - D(s) * P$$

$$Y(s) = \left[\frac{(C * P + \tilde{C} * \widetilde{P} * C * P)}{(1 + 2 * C * P + \tilde{C} * \widetilde{P} * C * P)} * R(s)\right] - \frac{P}{(1 + 2 * C * P + \tilde{C} * \widetilde{P} * C * P)} * D(s)$$

Closed-loop transfer function between reference position and output position

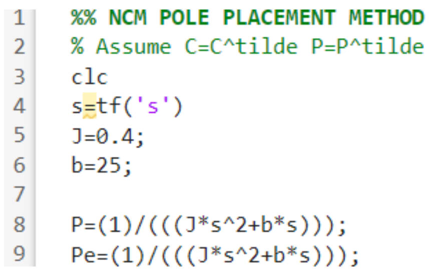
$$\frac{\mathrm{Y(s)}}{\mathrm{R(s)}} = \frac{\mathrm{C*P+\tilde{C}*\widetilde{P}*C*P}}{\mathrm{1+2*C*P+\tilde{C}*\widetilde{P}*C*P}} \qquad (4.8)$$

- If $C = \tilde{C}$ and $P = \widetilde{P}$ are equal, then $C * P = \tilde{C} * \widetilde{P} = a$

$$\frac{Y(s)}{R(s)} = \frac{a + a^2}{1 + 2 * a + a^2} = \frac{a * (a + 1)}{(a + 1) * (a + 1)} = \frac{a}{1 + a} = \frac{C * P}{1 + C * P}$$

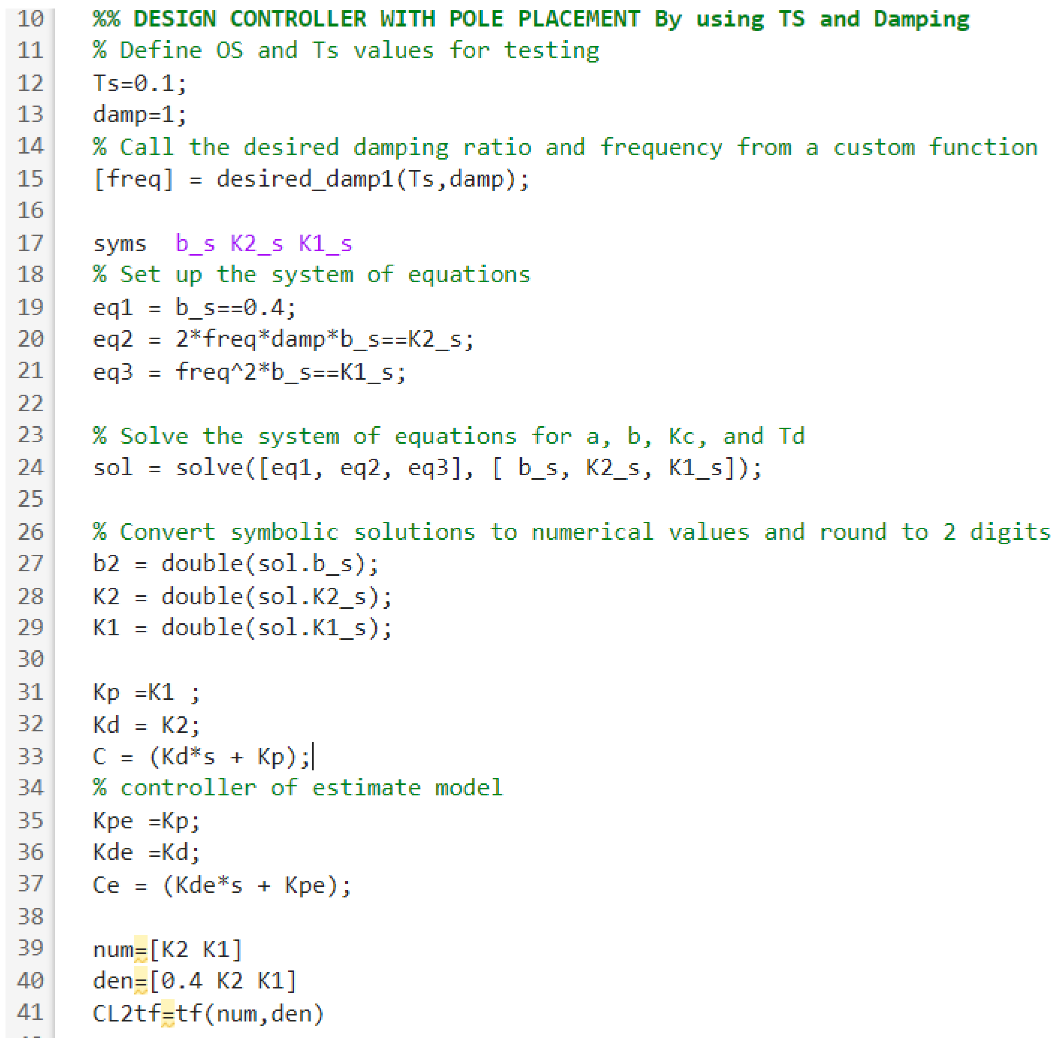
4.2.2 Simulink Model and Simulation Result

Actuator and estimate model parameters and plant transfer functions are defined in MATLAB Script.



```
%% NCM POLE PLACEMENT METHOD
% Assume C=C^tilde P=P^tilde
clc
s=tf('s')
J=0.4;
b=25;

P=(1)/(((J*s^2+b*s)));
Pe=(1)/(((J*s^2+b*s)));
```

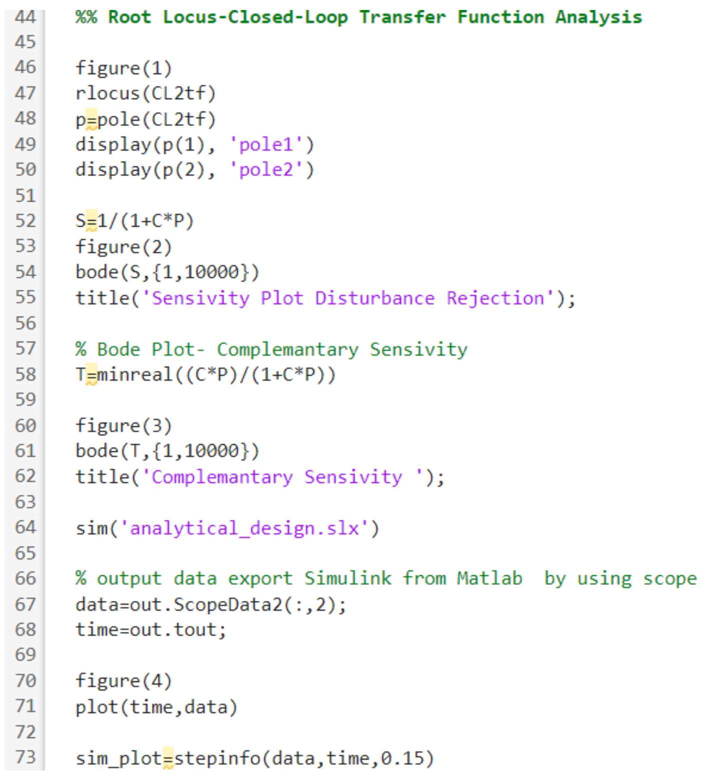
By assigning settling time and overshoot values, frequency value is calculated from the function file and desired equation of the second degree system is found. Desired and design polynomial are equated with residual value. Controller values are calculated by solving equation with symbolic representation.



```
%% DESIGN CONTROLLER WITH POLE PLACEMENT By using TS and Damping
% Define OS and Ts values for testing
Ts=0.1;
damp=1;
% Call the desired damping ratio and frequency from a custom function
[freq] = desired_damp1(Ts,damp);

syms  b_s K2_s K1_s
% Set up the system of equations
eq1 = b_s==0.4;
eq2 = 2*freq*damp*b_s==K2_s;
eq3 = freq^2*b_s==K1_s;

% Solve the system of equations for a, b, Kc, and Td
sol = solve([eq1, eq2, eq3], [ b_s, K2_s, K1_s]);

% Convert symbolic solutions to numerical values and round to 2 digits
b2 = double(sol.b_s);
K2 = double(sol.K2_s);
K1 = double(sol.K1_s);

Kp =K1 ;
Kd = K2;
C = (Kd*s + Kp);
% controller of estimate model
Kpe =Kp;
Kde =Kd;
Ce = (Kde*s + Kpe);

num=[K2 K1]
den=[0.4 K2 K1]
CL2tf=tf(num,den)
```

Figure 16: Pole Placement Method code in MATLAB Script

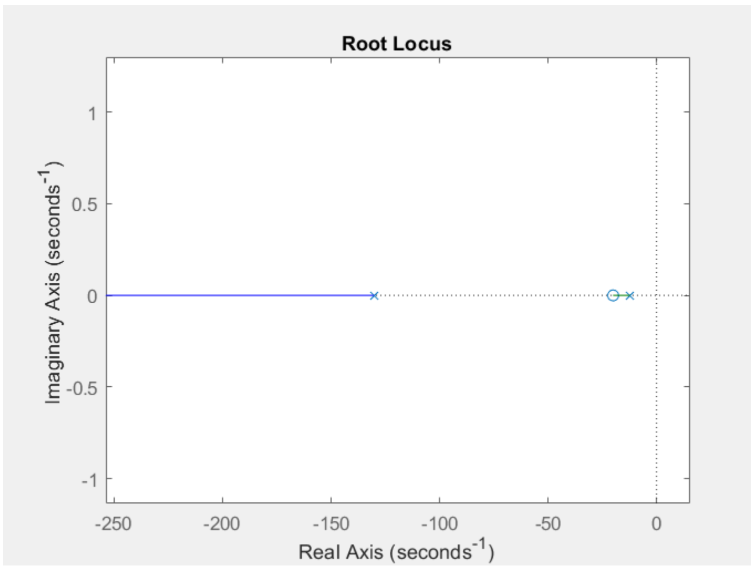
Before simulating the system, the root locus, sensitivity and complementary sensitivity plots of the system are observed to estimate its characteristic feature and to reach a stable system more quickly.



```
%% Root Locus-Closed-Loop Transfer Function Analysis

figure(1)
rlocus(CL2tf)
p=pole(CL2tf)
display(p(1), 'pole1')
display(p(2), 'pole2')

S=1/(1+C*P)
figure(2)
bode(S,{1,10000})
title('Sensivity Plot Disturbance Rejection');

% Bode Plot- Complemantary Sensivity
T=minreal((C*P)/(1+C*P))

figure(3)
bode(T,{1,10000})
title('Complemantary Sensivity ');

sim('analytical_design.slx')

% output data export Simulink from Matlab  by using scope
data=out.ScopeData2(:,2);
time=out.tout;

figure(4)
plot(time,data)

sim_plot=stepinfo(data,time,0.15)
```

Figure 17: Root Locus Closed-Loop transfer function code in MATLAB Script

As seen in the root locus graph, the system is second order and the system is overdamped because the poles are on the real axis and different from each other. Also as observed the second pole is far from zero which makes the system more stable.



The Bode plot is used to analyze the system’s disturbance rejection performance.

$$e = \frac{1}{1+L(s)} * r - \frac{1}{1+L(s)} * G_d(s) * d + \frac{L(s)}{1+L(s)} * n \qquad (4.\ 9)$$

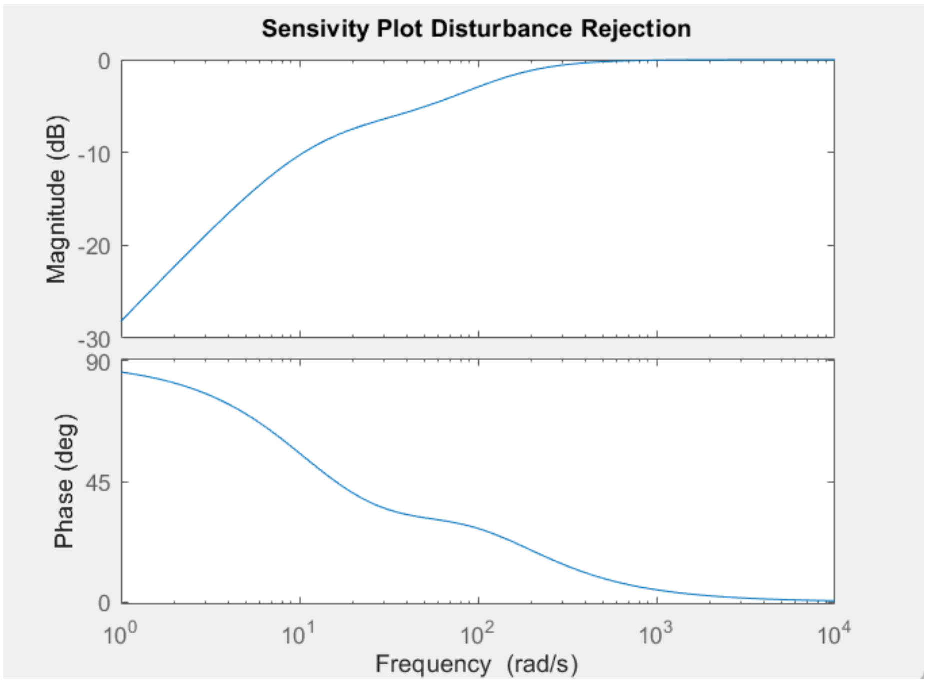
Performance requirements can be approximated by requirements for $L(j\omega)$

$|L(j\omega)| \gg 1 \;\rightarrow\; |S(j\omega)| \ll 1$ (Good tracking performance)

$|L(j\omega)| = 1$ gives bandwith $\approx\; \omega_c$

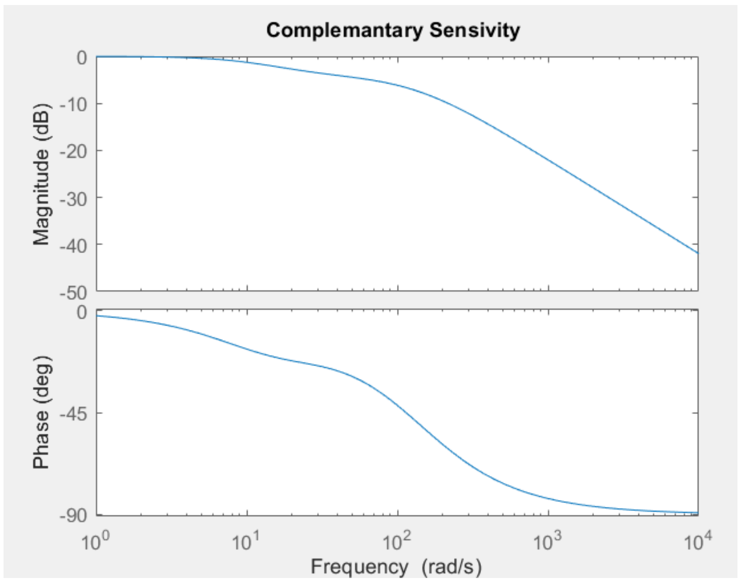
$|L(j\omega)| \ll 1 \;\rightarrow\; |T(j\omega)| \ll 1$ (Good noise rejection)

As observed in the sensitivity diagram, when the magnitude starts below 0 dB, it indicates that the system can suppress the input disturbances. However, if the magnitude remains well below 0 dB in the very long frequency range, the system may become sensitive to noise at high frequencies, which is an undesirable situation.



Sensitivity (S) and Complementary Sensitivity (T) plots are completed to 1, because the system balances its response to input disturbances (S) and output noise (T). The magnitude of the Complementary Sensitivity function T, which is less than 1, indicates that it has good noise rejection performance, especially at high frequencies.

$$S(j\omega) + T(j\omega) = 1 \qquad (4.10)$$



Simulink model of NCM Position Control

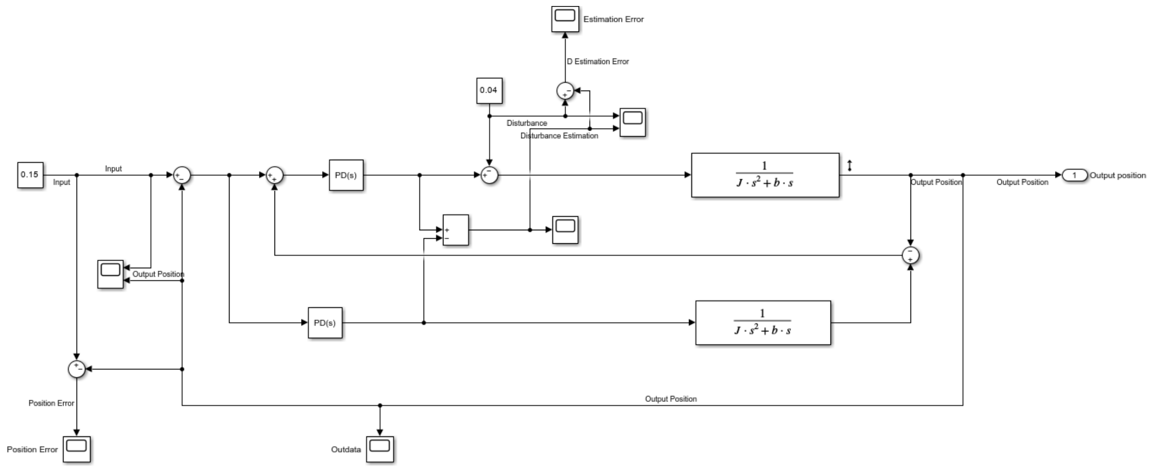


Figure 18: NCM position of Simulink model

In this system, two most important results were evaluated, one of which was the comparison of the input and output positions. The positions and the error between them were observed with scopes. The simulation results are as follows.

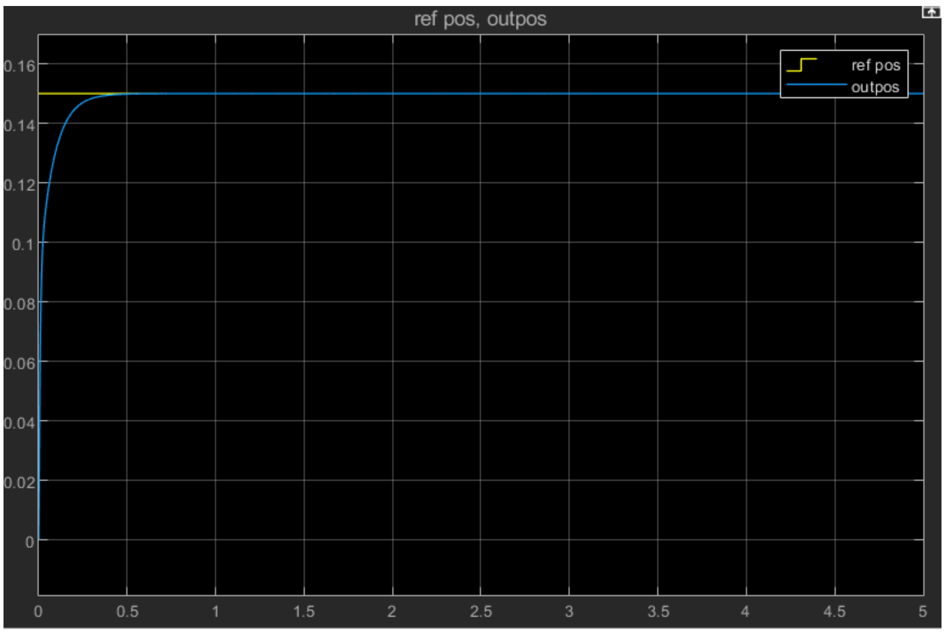


Figure 19: Result of in and out position

Result obtained from the Workspace block in Simulink calculates the time response characteristics of the system with the stepinfo command in the MATLAB Script.

Overshoot	: 0.01%
Rise Time	: 0.1151 seconds
Settling Time:	0.2499 seconds

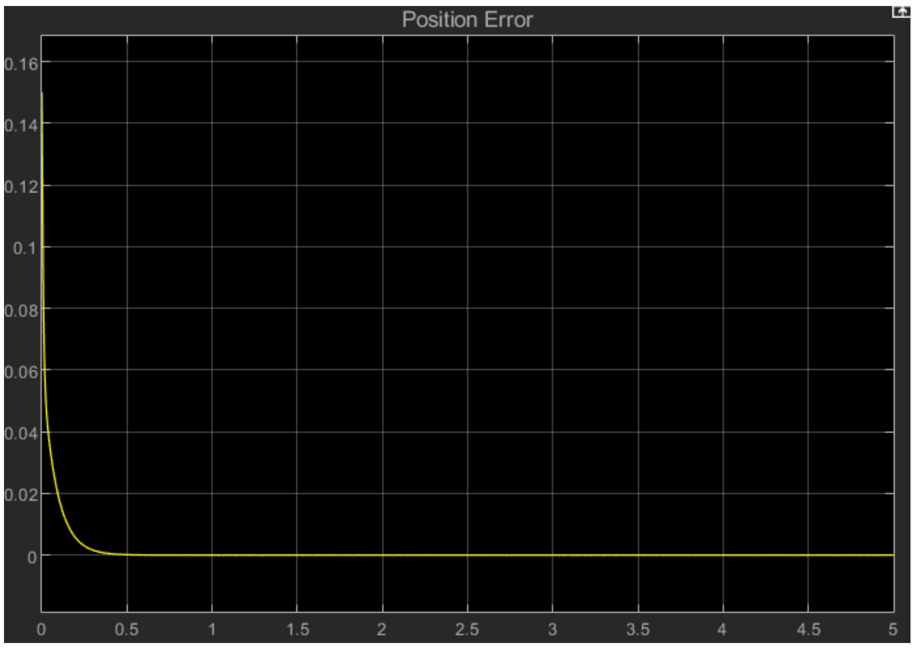


Figure 20: Error of in and out position

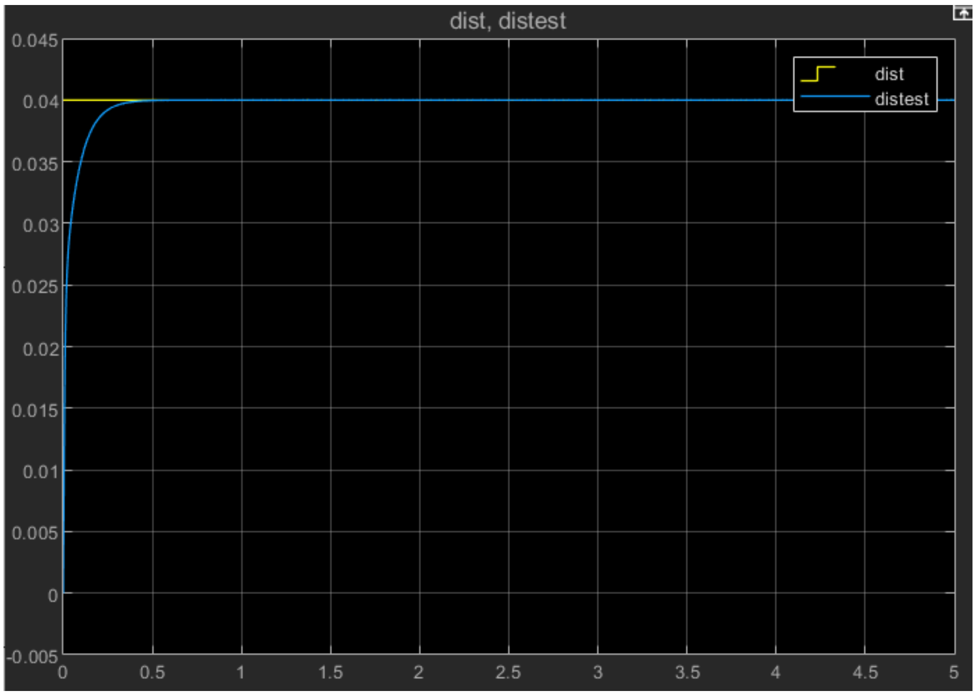


Figure 21: Result of disturbance and disturbance estimation

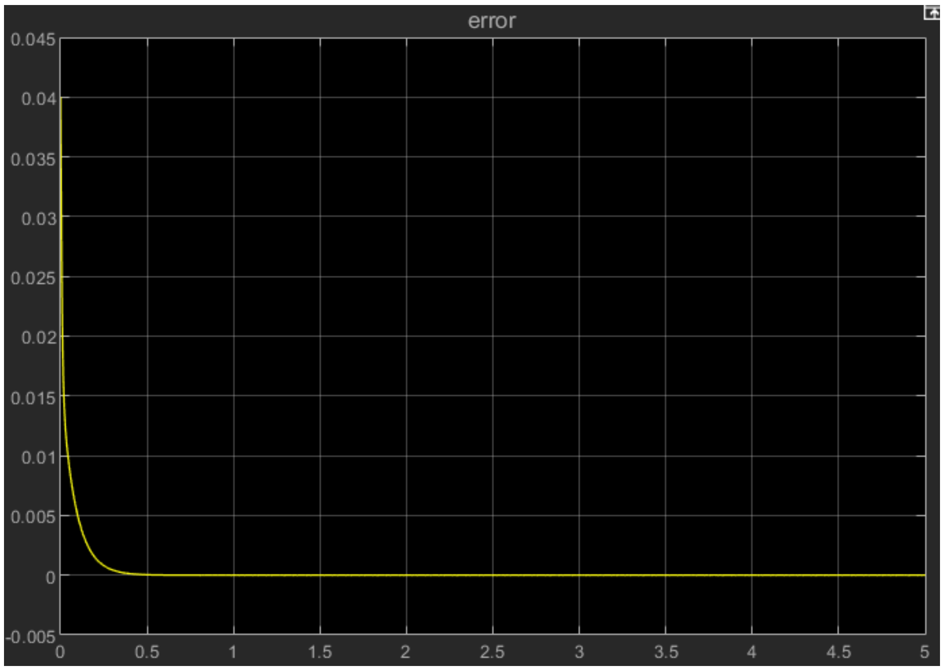


Figure 22: Error between disturbance and disturbance estimation

4.2.3 Experimental Model and Experiment Result

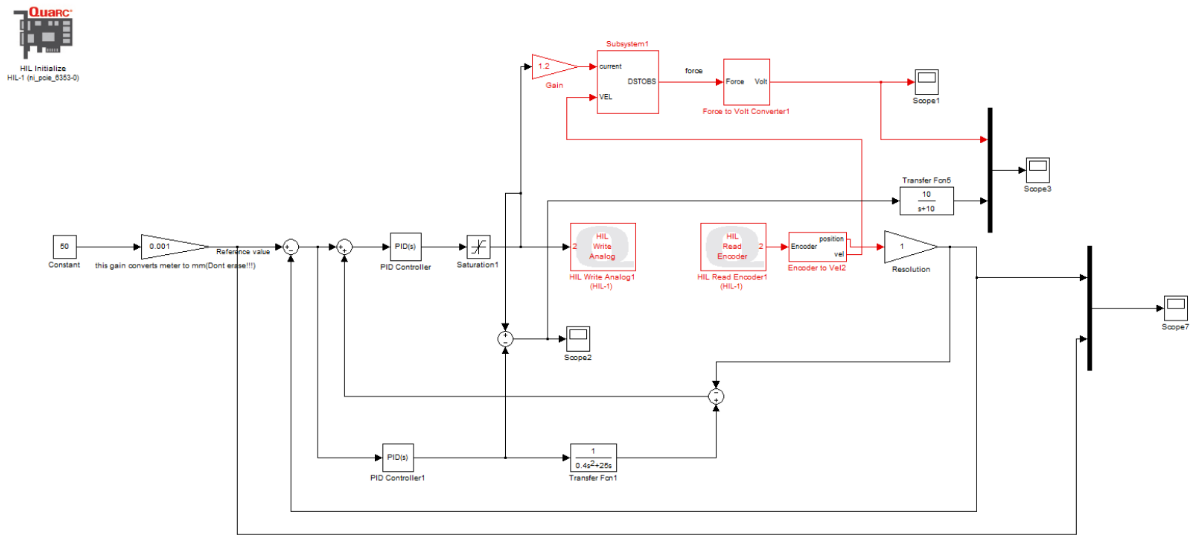
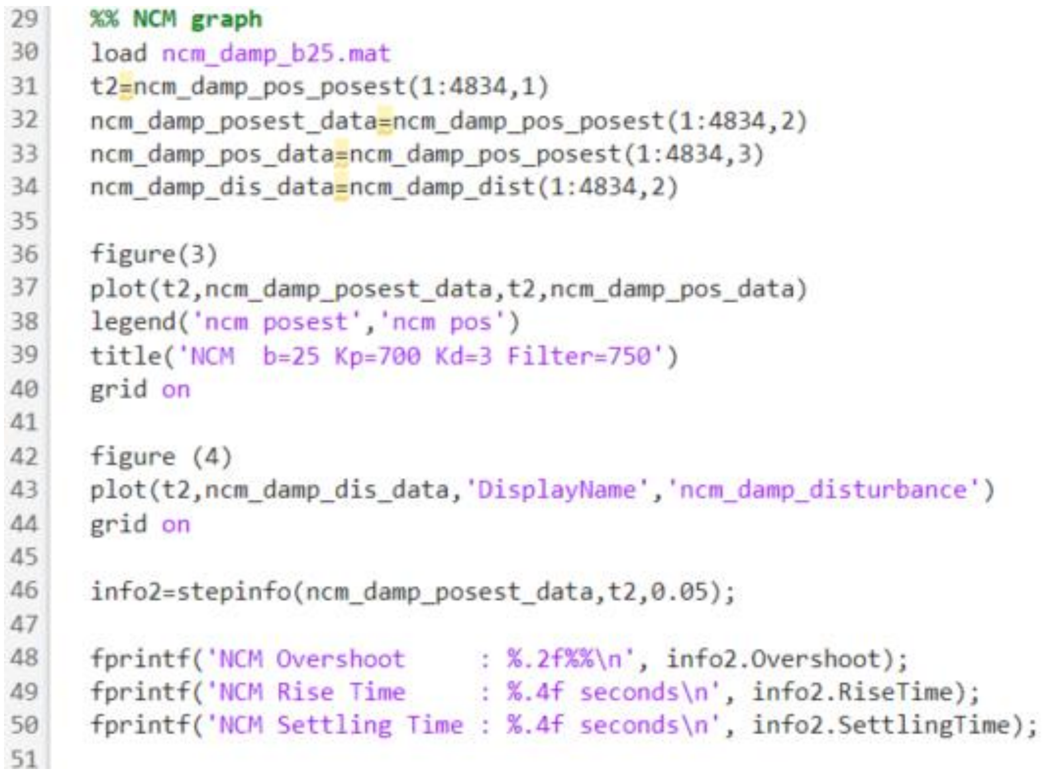
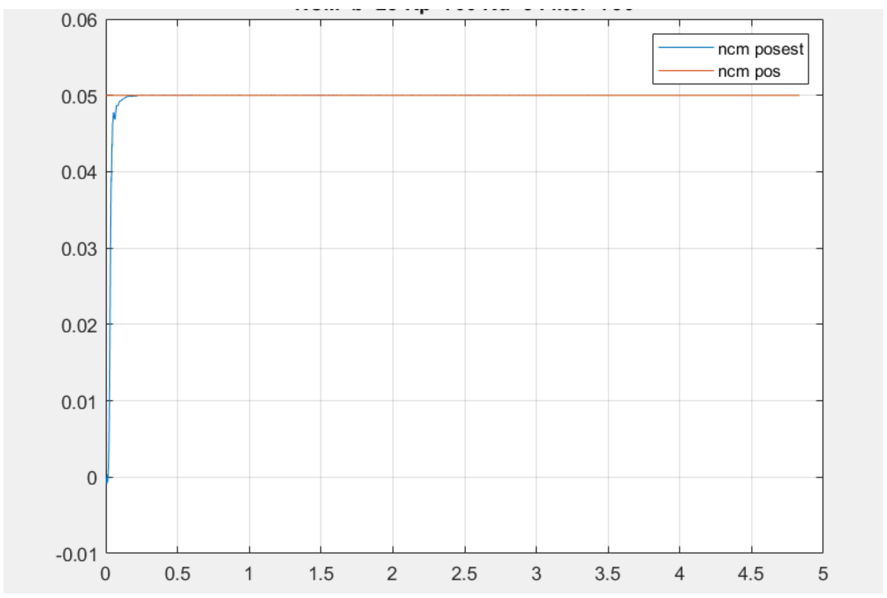


Figure 23: Experimental setup of NCM position control



```
%% NCM graph
load ncm_damp_b25.mat
t2=ncm_damp_pos_posest(1:4834,1)
ncm_damp_posest_data=ncm_damp_pos_posest(1:4834,2)
ncm_damp_pos_data=ncm_damp_pos_posest(1:4834,3)
ncm_damp_dis_data=ncm_damp_dist(1:4834,2)

figure(3)
plot(t2,ncm_damp_posest_data,t2,ncm_damp_pos_data)
legend('ncm posest','ncm pos')
title('NCM  b=25 Kp=700 Kd=3 Filter=750')
grid on

figure (4)
plot(t2,ncm_damp_dis_data,'DisplayName','ncm_damp_disturbance')
grid on

info2=stepinfo(ncm_damp_posest_data,t2,0.05);

fprintf('NCM Overshoot      : %.2f%%\n', info2.Overshoot);
fprintf('NCM Rise Time      : %.4f seconds\n', info2.RiseTime);
fprintf('NCM Settling Time : %.4f seconds\n', info2.SettlingTime);
```

Figure 24: NCM position data from electric motor



NCM Overshoot : 0.02%

NCM Rise Time : 0.0234 seconds

NCM Settling Time: 0.0921 seconds

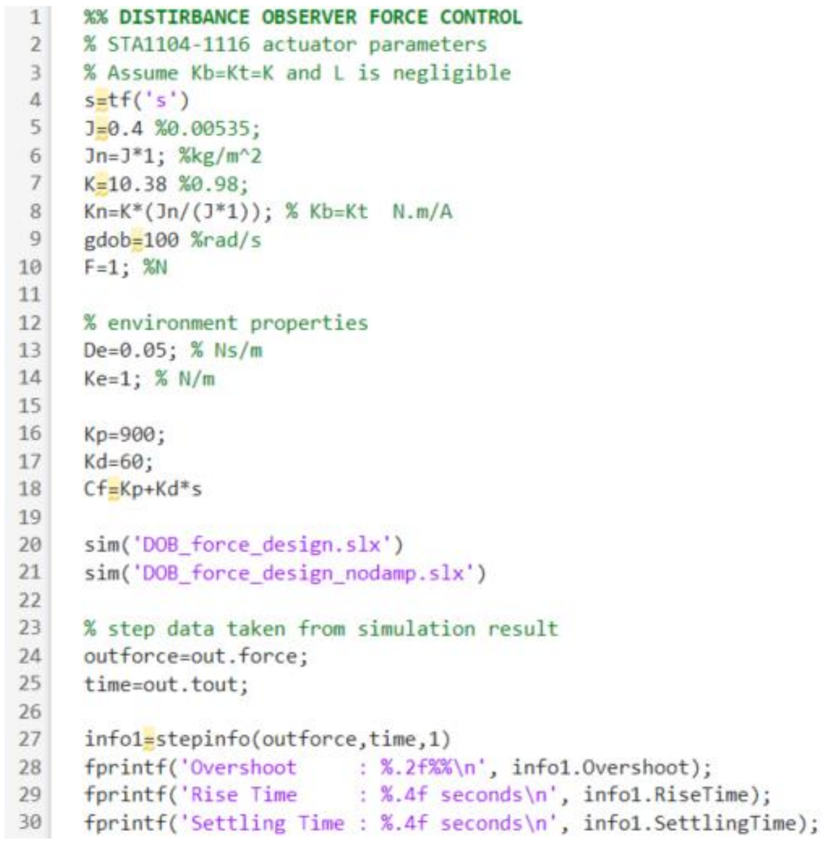
5)FORCE CONTROL

5.1Disturbance Observer

The motor disturbance includes friction effects and the effects of parameter changes as shown in equation (4.9) and this structure includes a differentiator that can increase the noise in the system. Usually this disturbance cannot be measured. However, if the system is controlled using a controller as shown in Figure 5, the motor disturbance (T_{dis}) becomes measurable and is usually removed after a low-pass filter as shown in Figure 5. g_{dis} is the angular cut-off frequency of the low-pass filter. The disturbance observer observes the disturbance force in the system without using force sensors.

5.1.1 Simulink Model and Simulation Result

For DOB Force control, STA1104-1116 actuator properties and other parameters in the model were defined in the MATLAB script. Also, while designing the DOB nominal model, J_n and K_n parameters are used, these parameters are known or assumed values, thus DOB aims to increase the stability and performance of the system by estimating the disturbance between the real system and the nominal model.



```
%% DISTIRBANCE OBSERVER FORCE CONTROL
% STA1104-1116 actuator parameters
% Assume Kb=Kt=K and L is negligible
s=tf('s')
J=0.4 %0.00535;
Jn=J*1; %kg/m^2
K=10.38 %0.98;
Kn=K*(Jn/(J*1)); % Kb=Kt  N.m/A
gdob=100 %rad/s
F=1; %N

% environment properties
De=0.05; % Ns/m
Ke=1; % N/m

Kp=900;
Kd=60;
Cf=Kp+Kd*s

sim('DOB_force_design.slx')
sim('DOB_force_design_nodamp.slx')

% step data taken from simulation result
outforce=out.force;
time=out.tout;

info1=stepinfo(outforce,time,1)
fprintf('Overshoot      : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time      : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

Figure 25: DOB Force control MATLAB code

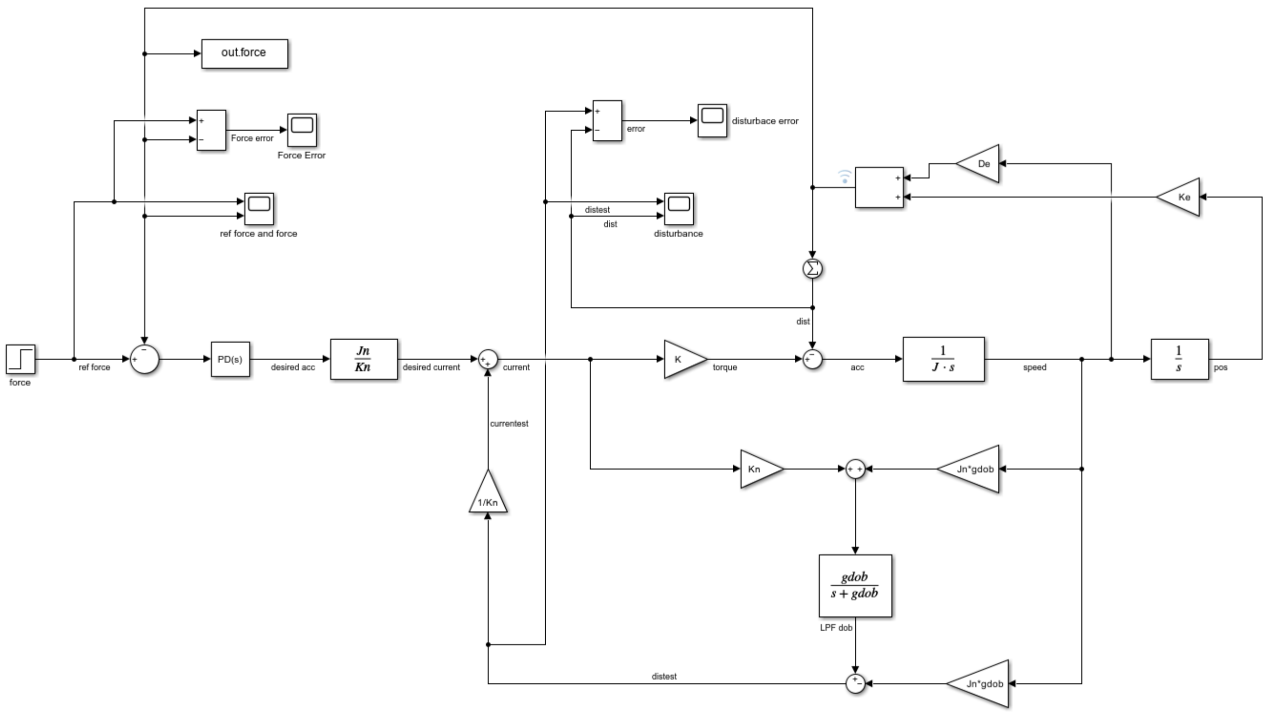


Figure 26: Force Control DOB on Simulink

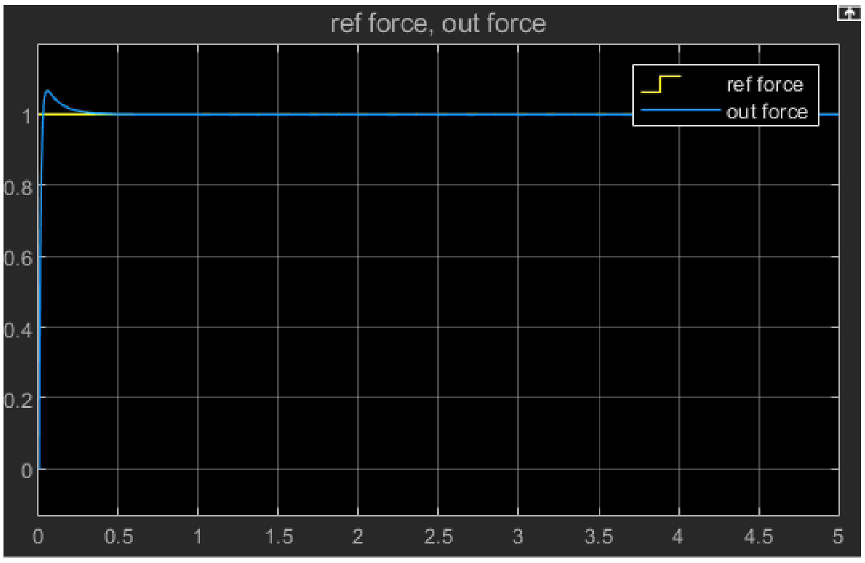


Figure 27: Result of reference and out force

Result obtained from the Workspace block in Simulink calculates the time response characteristics of the system with the stepinfo command in the MATLAB Script.

Overshoot : 3.95%

Rise Time : 0.0284 seconds

Settling Time: 0.2406 seconds

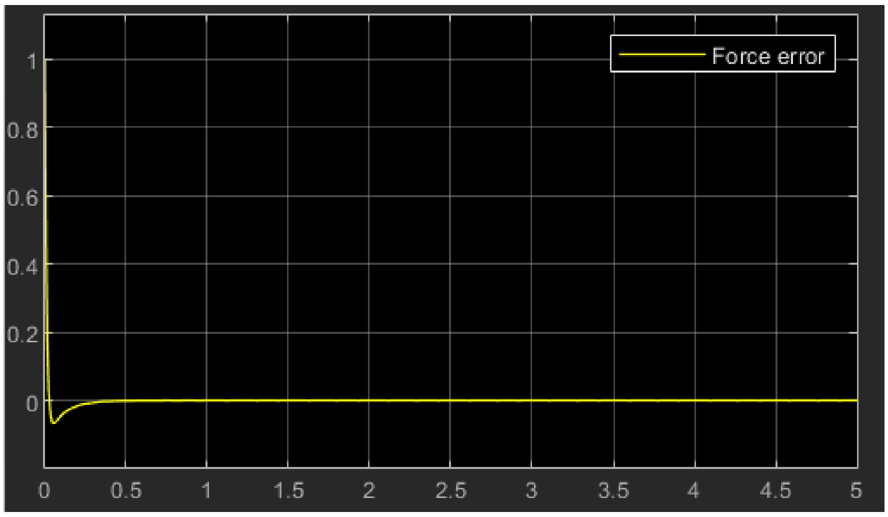


Figure 28: Error between reference and out force

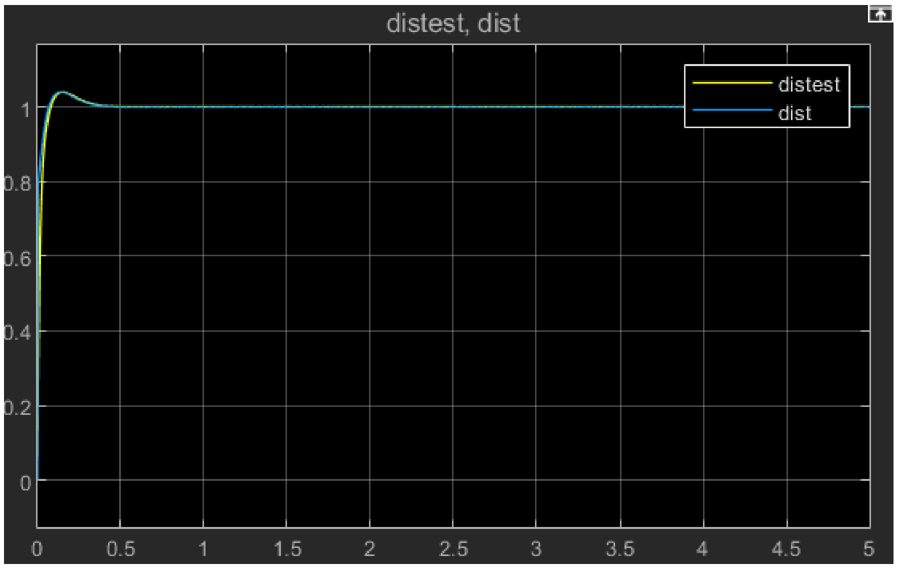


Figure 29: Result of disturbance and disturbance estimation

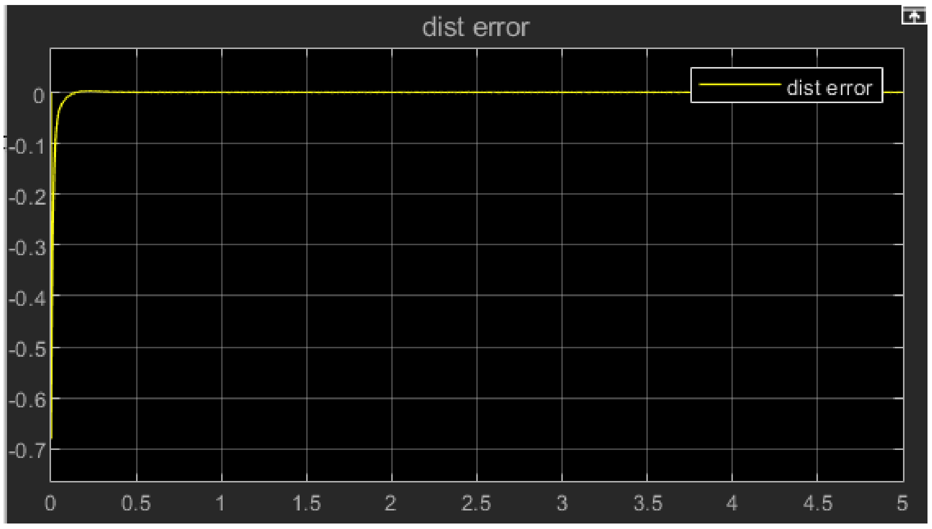


Figure 30: Error between disturbance and disturbance estimation

5.1.2 Experimental Model and Experiment Result

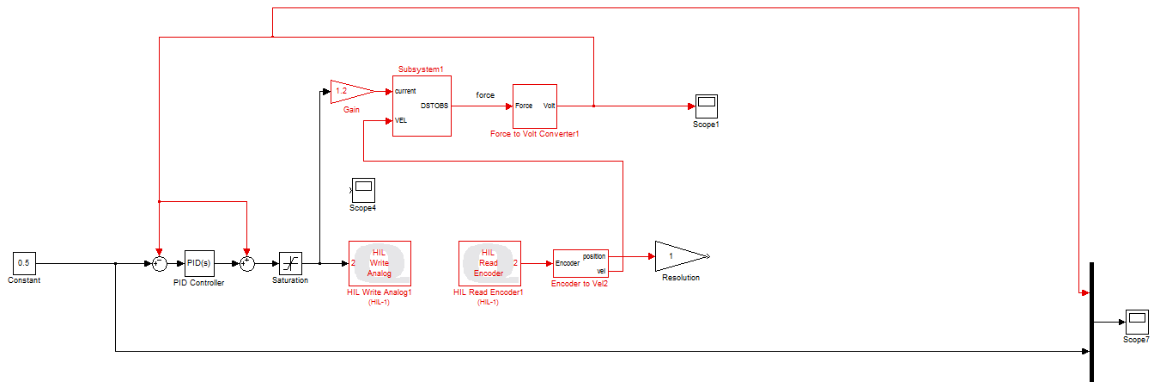
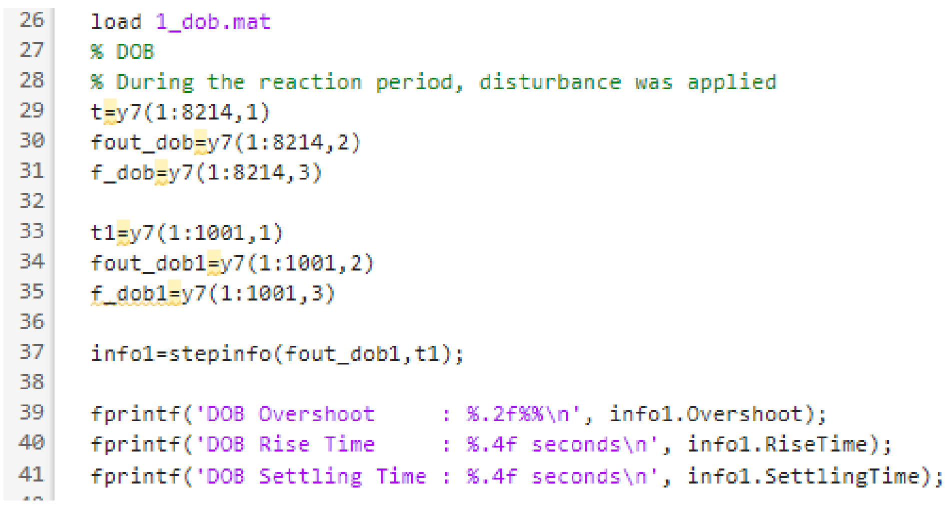
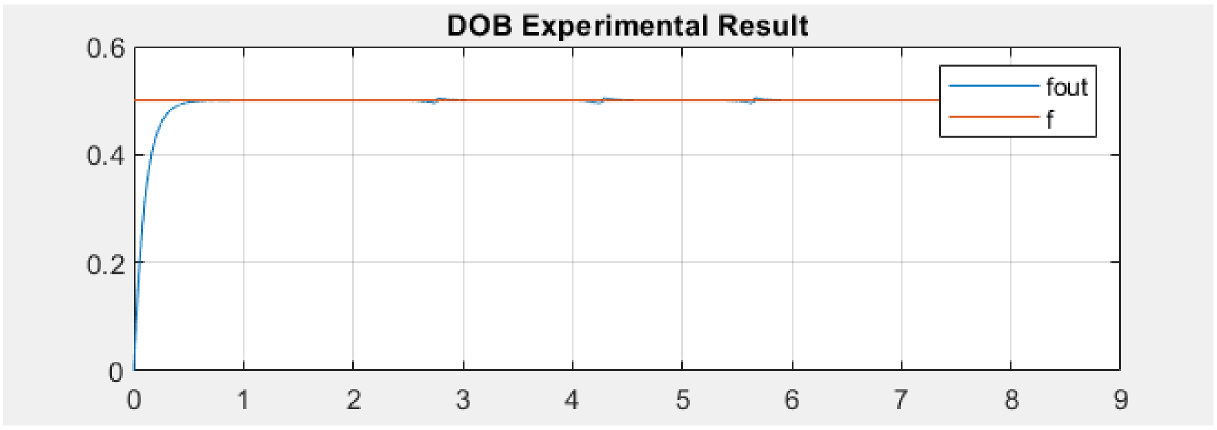


Figure 31: Experimental setup of DOB force control



```
load 1_dob.mat
% DOB
% During the reaction period, disturbance was applied
t=y7(1:8214,1)
fout_dob=y7(1:8214,2)
f_dob=y7(1:8214,3)

t1=y7(1:1001,1)
fout_dob1=y7(1:1001,2)
f_dob1=y7(1:1001,3)

info1=stepinfo(fout_dob1,t1);

fprintf('DOB Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('DOB Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('DOB Settling Time : %.4f seconds\n', info1.SettlingTime);
```

Figure 32: DOB force data from electric motor



DOB Overshoot : 0.02%

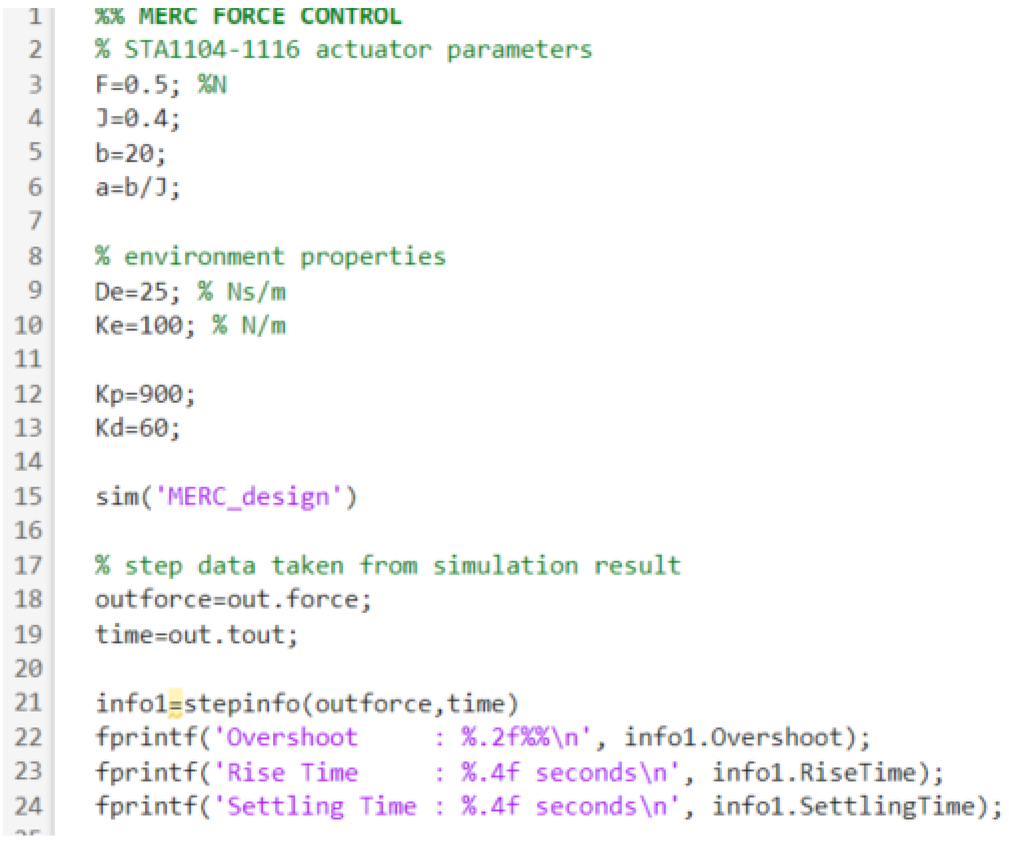
DOB Rise Time : 0.2197 seconds

DOB Settling Time: 0.3920 seconds

5.2MERC

5.2.1 Simulink Model and Simulation Result

For DOB Force control, STA1104-1116 actuator properties and other parameters in the model were defined in the MATLAB script.



```
%% MERC FORCE CONTROL
% STA1104-1116 actuator parameters
F=0.5; %N
J=0.4;
b=20;
a=b/J;

% environment properties
De=25; % Ns/m
Ke=100; % N/m

Kp=900;
Kd=60;

sim('MERC_design')

% step data taken from simulation result
outforce=out.force;
time=out.tout;

info1=stepinfo(outforce,time)
fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

Figure 33: MERC Force control code in MATLAB

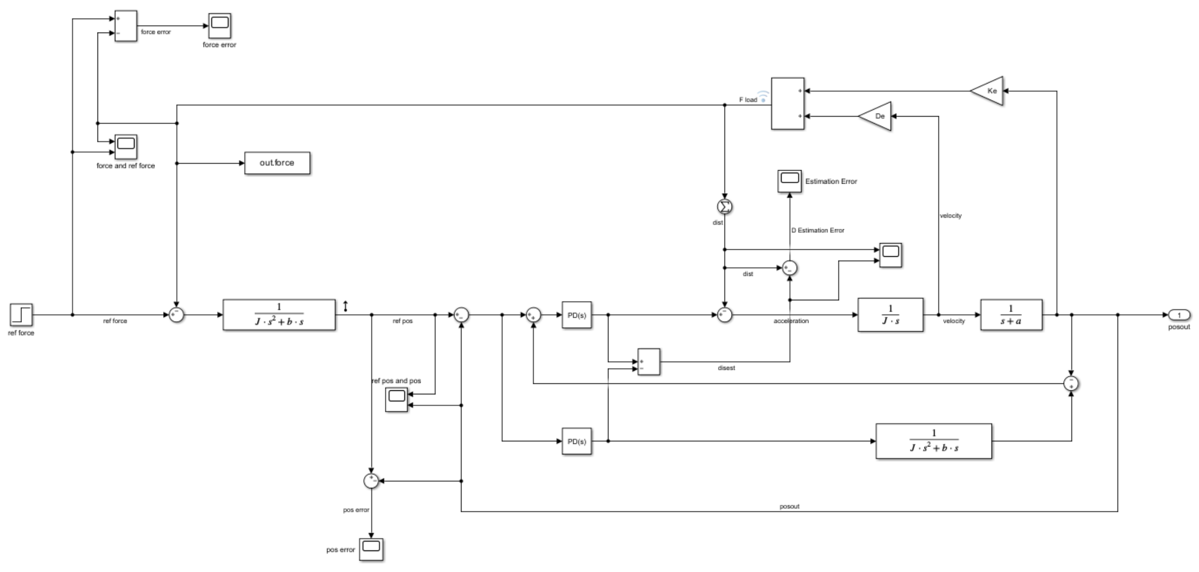


Figure 34: MERC Force control on Simulink

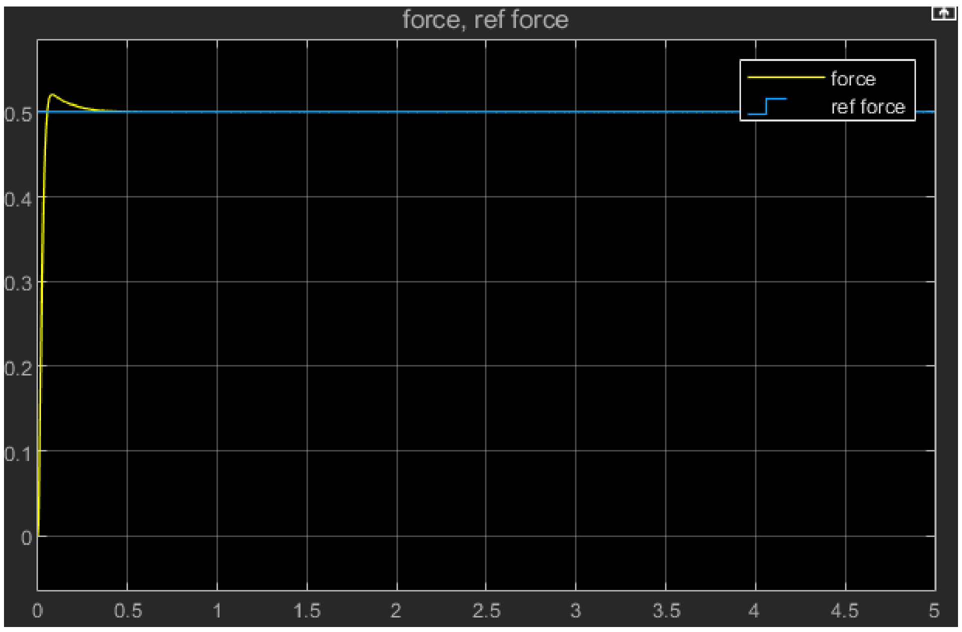


Figure 35: Result of reference and out force

Result obtained from the Workspace block in Simulink calculates the time response characteristics of the system with the stepinfo command in the MATLAB Script.

Overshoot : 3.77%

Rise Time : 0.0347 seconds

Settling Time: 0.2112 seconds

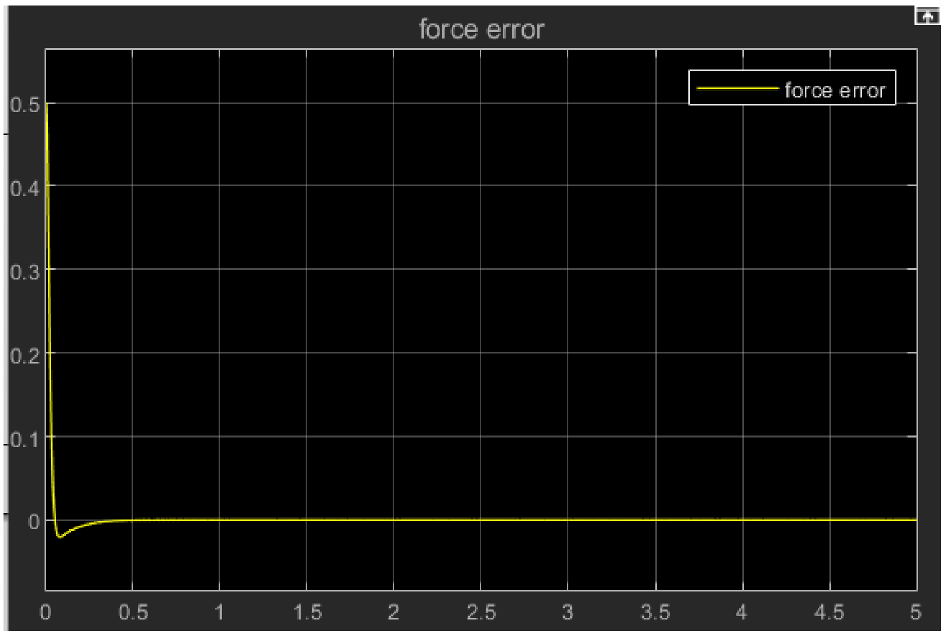


Figure 36: Error between force and force estimation

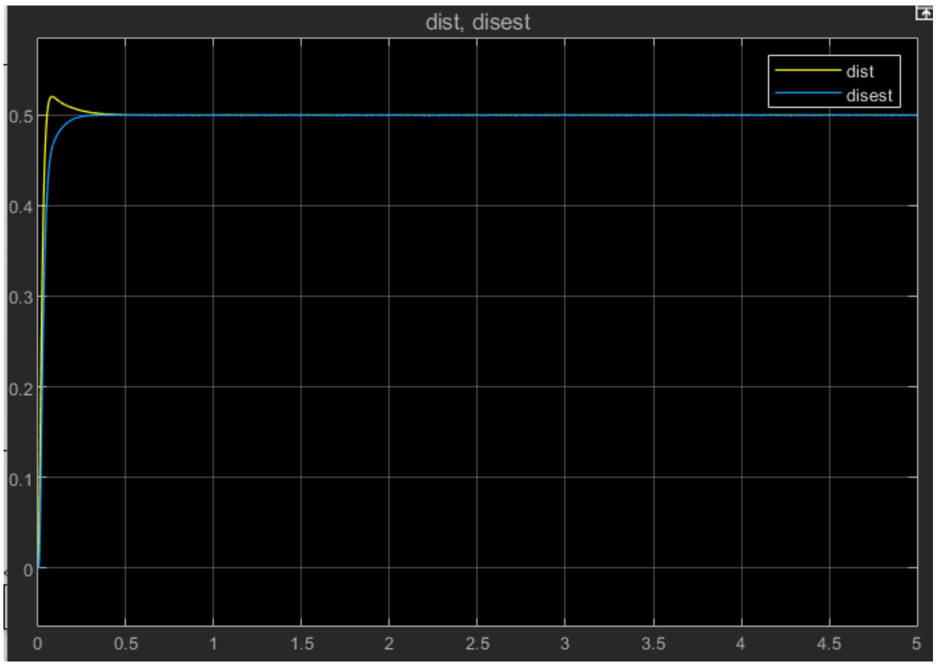


Figure 37: Result of disturbance and disturbance force

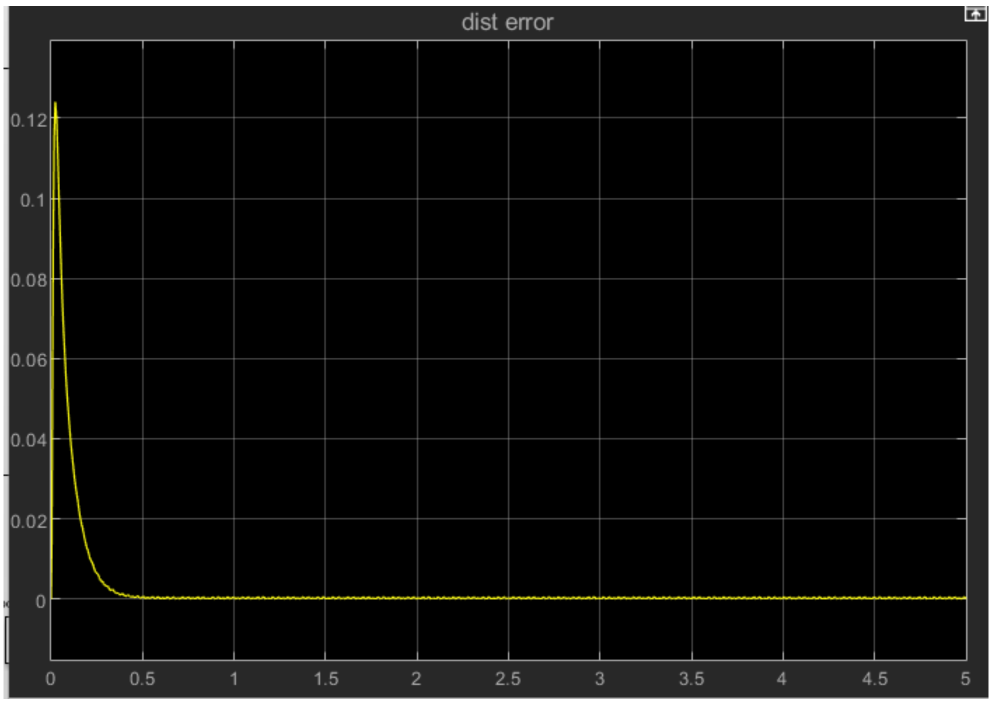


Figure 38: Error between disturbance and disturbance estimation

5.2.2 Experimental Model and Experiment Result

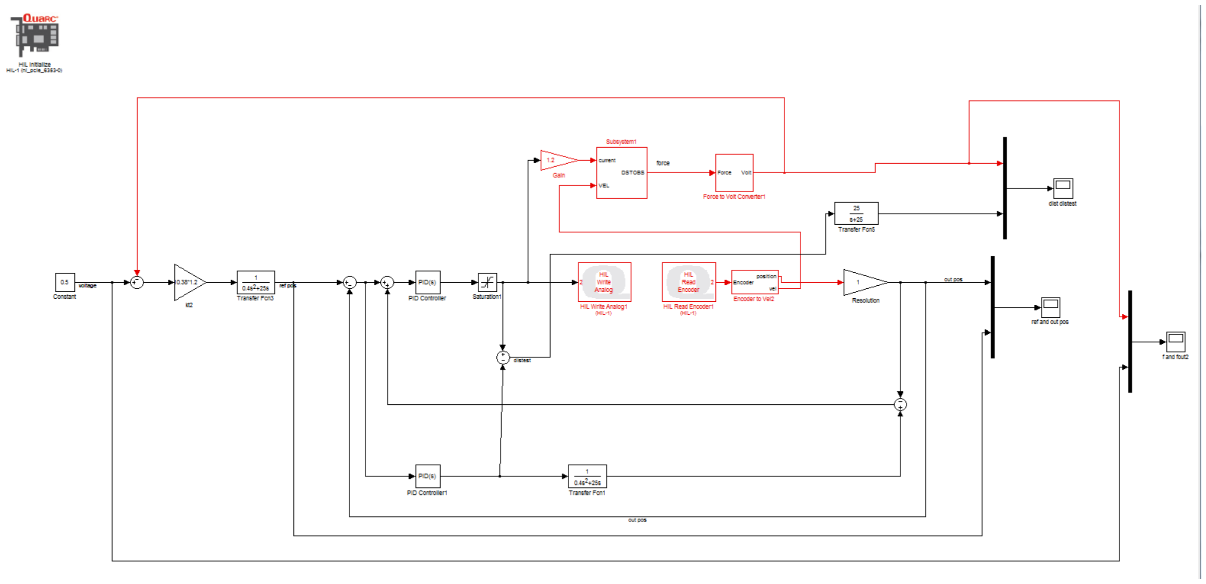
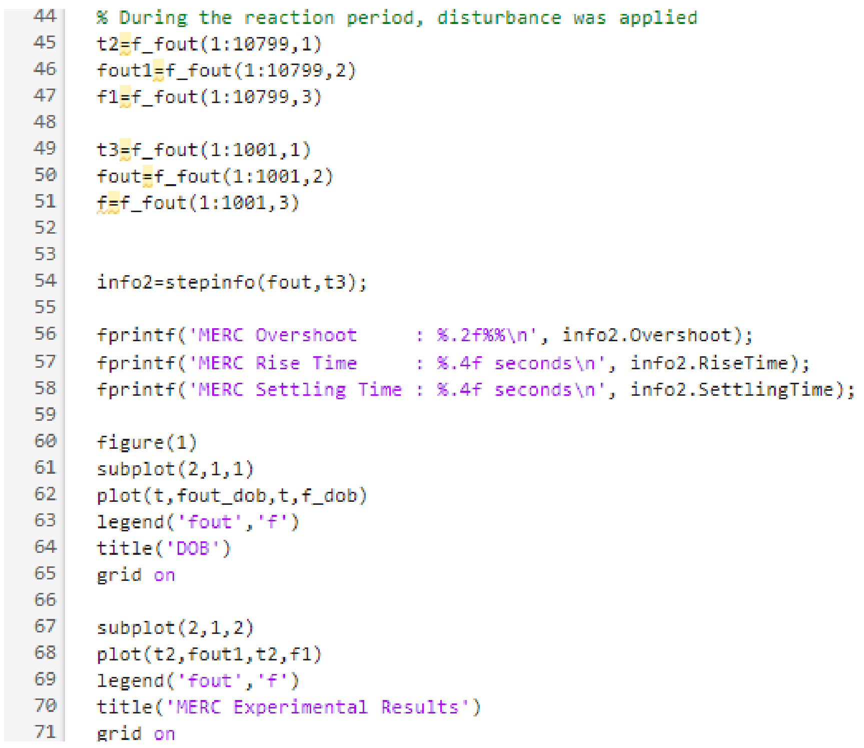
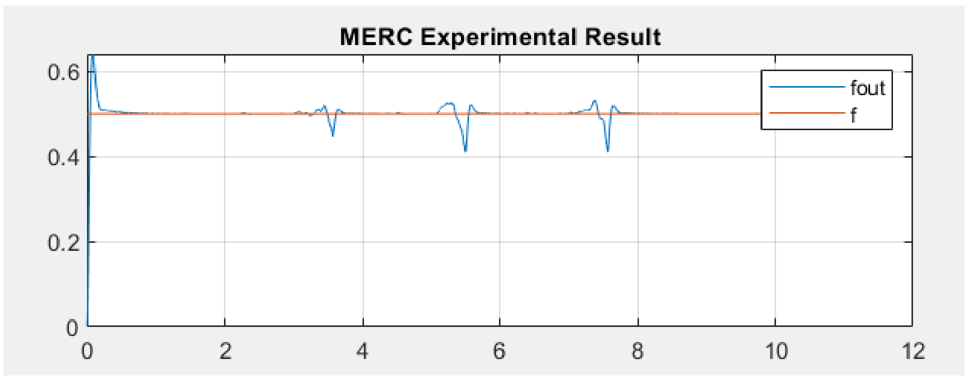


Figure 39: Experimental setup of MERC force control



```
% During the reaction period, disturbance was applied
t2=f_fout(1:10799,1)
fout1=f_fout(1:10799,2)
f1=f_fout(1:10799,3)

t3=f_fout(1:1001,1)
fout=f_fout(1:1001,2)
f=f_fout(1:1001,3)


info2=stepinfo(fout,t3);

fprintf('MERC Overshoot     : %.2f%%\n', info2.Overshoot);
fprintf('MERC Rise Time     : %.4f seconds\n', info2.RiseTime);
fprintf('MERC Settling Time : %.4f seconds\n', info2.SettlingTime);

figure(1)
subplot(2,1,1)
plot(t,fout_dob,t,f_dob)
legend('fout','f')
title('DOB')
grid on

subplot(2,1,2)
plot(t2,fout1,t2,f1)
legend('fout','f')
title('MERC Experimental Results')
grid on
```

Figure 40: MERC force data from electric motor



MERC Overshoot : 27.81%

MERC Rise Time : 0.0312 seconds

MERC Settling Time: 0.1915 seconds

6)RESULT

In conclusion, in this study, two different control methods—Disturbance Observer Based Control (DOB) and Nominal/Model Error Robust Control (NCM/MERC)—are compared for both position and force control. The performances of the systems are evaluated with respect to time response characteristics such as overshoot, rise time, and settling time, and both theoretical (simulation-based) and experimental results are analyzed for both control methods.

In theoretical simulations, DOB-based position control is characterized by 17.55% overshoot, 0.1821 seconds rise time, and 1.1668 seconds settling time. These results show that the system responds relatively quickly but operates less stably due to the high overshoot and settling time. In contrast, the NCM method yielded much more successful results: 0.01% overshoot, 0.1151 seconds rise time, and 0.2499 seconds settling time. These values show that the system is faster and more stable, reaching the desired position in a short time with very small deviation.

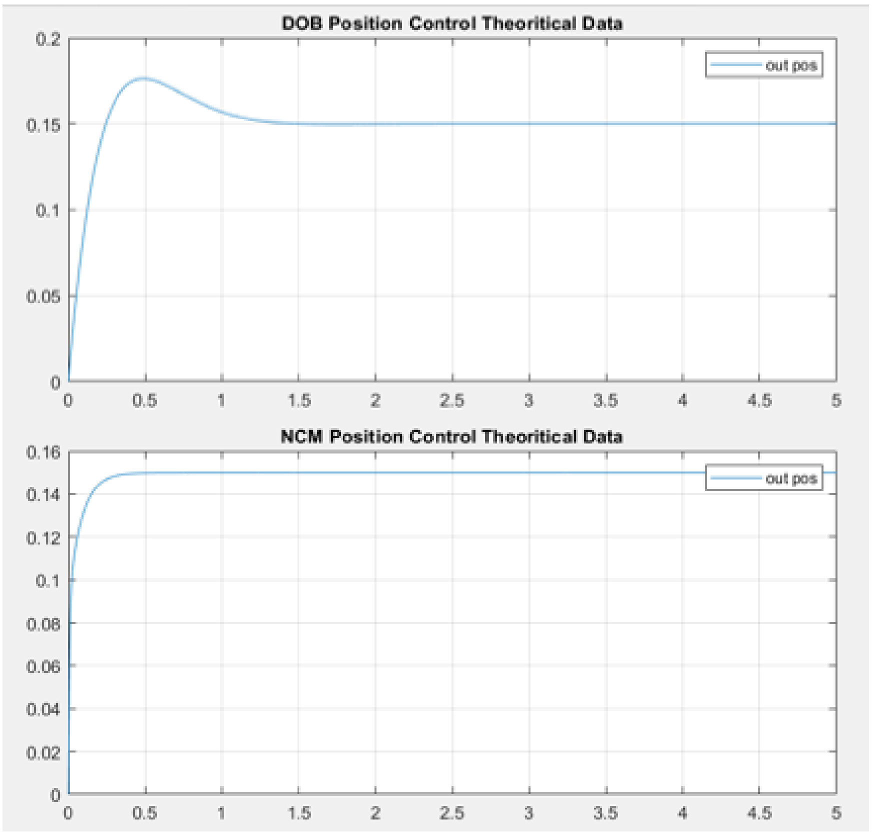


Figure 41: Theoretical Results of Position Control

Similarly, the experimental results also provided results that confirmed the simulation findings. The DOB method showed 19.25% overshoot, 0.0139 seconds rise time and 0.1798 seconds settling time in the experimental environment. Although the response time of the system was much faster than the theoretical model, an increase in the overshoot was observed. This situation may be caused by unmodeled dynamics or measurement noise in the system.

The NCM method also showed the best performance experimentally: 0.02% overshoot, 0.0234 seconds rise time and 0.0921 seconds settling time provided very fast and stable control. These results indicate that the NCM method should be preferred in applications requiring position control.

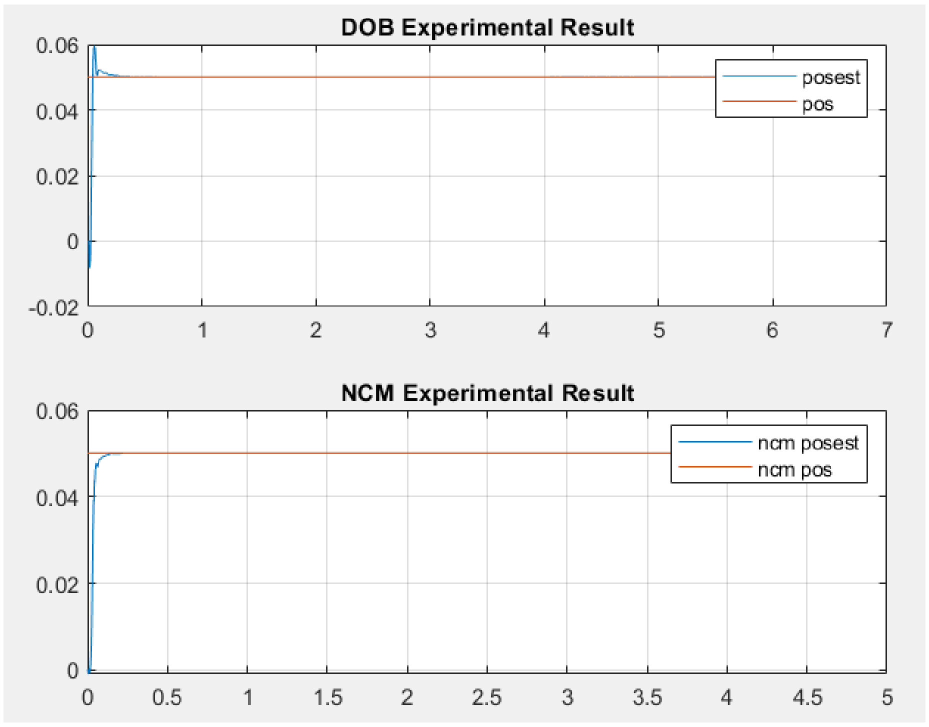


Figure 42: Experimental Results of Position Control

In theoretical analyses performed in force control, the DOB method achieved 3.95% overshoot, 0.0284 seconds rise time and 0.2406 seconds settling time. These values show that the system operates sufficiently fast and with acceptable overshoot.

The MERC method, on the other hand, provided slightly lower overshoot and shorter settling time than the DOB method with 3.77% overshoot, 0.0347 seconds rise time and 0.2112 seconds settling time. Although the rise time was slightly longer, it exhibited a more successful performance in terms of reaching general balance. This situation reveals that the MERC control method can provide more balanced and stable results.

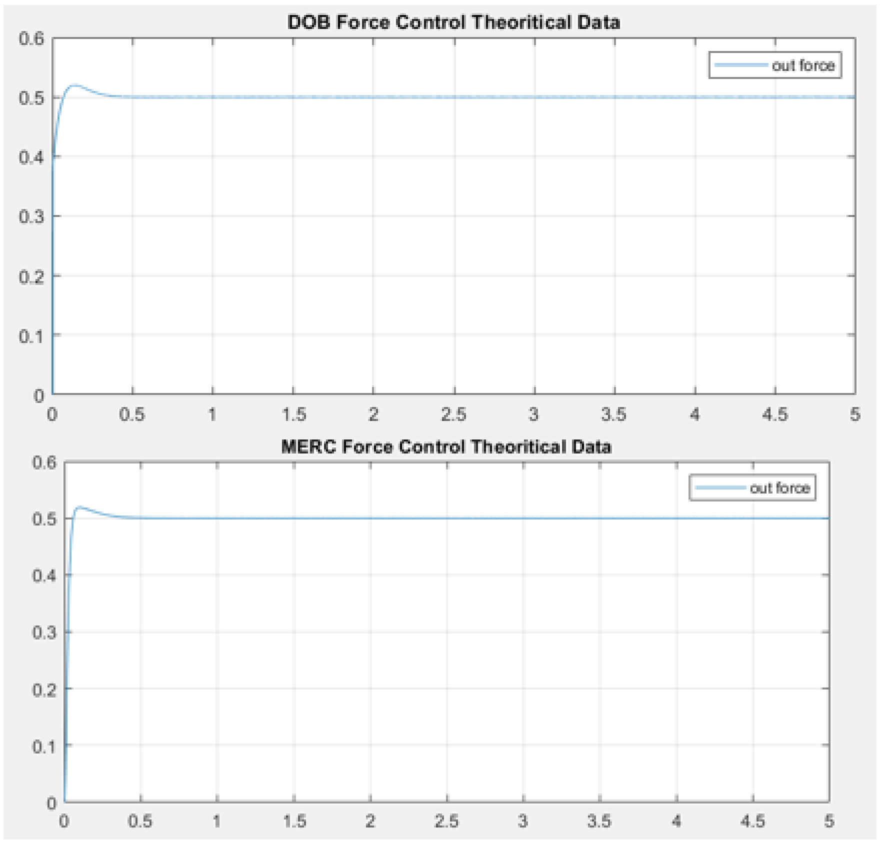


Figure 43: Theoretical Results of Force Control

The DOB controller produced a low overshoot of 0.02%, but had a relatively long rise time of 0.2197 seconds and a settling time of 0.3920 seconds. These results indicate that the DOB method maintained accuracy with negligible overshoot, while its experimental response was significantly slower compared to theoretical simulations.

In contrast, the MERC controller exhibited a much faster response with a rise time of 0.0312 seconds and a settling time of 0.1915 seconds. However, this speed improvement came with a significant overshoot of 27.81%.

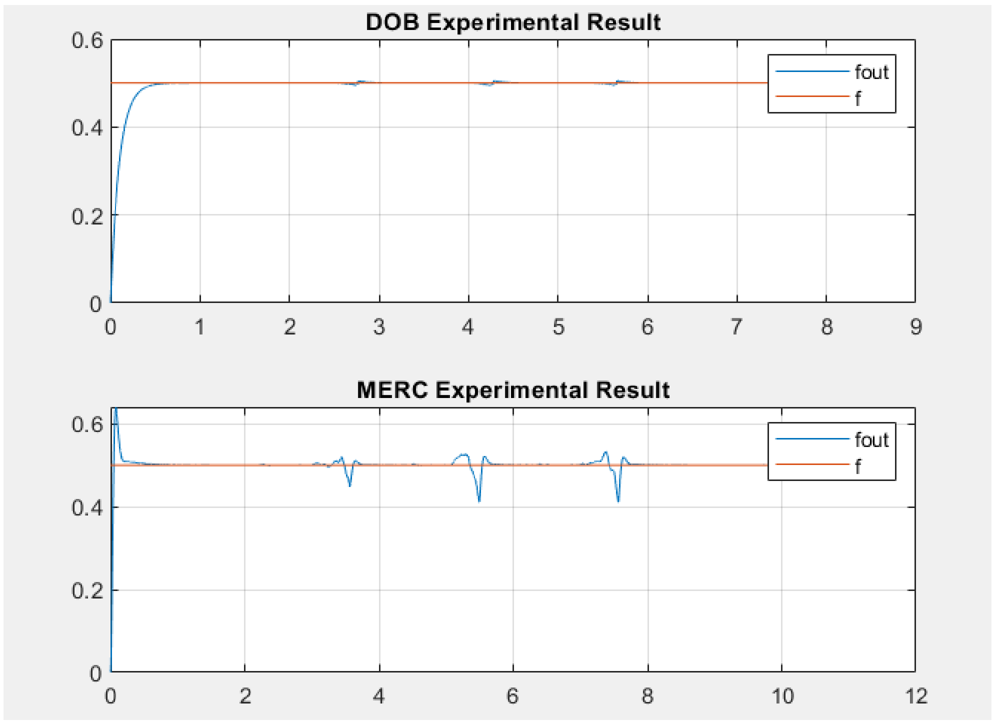


Figure 44: Experimental Results of Force Control

7) APPENDIX

7.1 DOB Position control MATLAB Code

```
%% DOB Position
% Electric Motor Parameters
% Assume Kb=Kt=K and L is negligible
J=0.4  % 5.2e-3;
Jn=J*1; %kg/m^2
K=0.949;
Kn=K*1; % Kb=Kt  N.m/A
torque=0.04;  %N*m
b=15;

%filter coefficient
gdob=1000; %rad/s

Kp=284;
Kd=284*0.3759;

sim('DOB_pos_design.slx')

% step data taken from simulation result

outpos=out.simout.signals.values;
time=out.simout.time;

info1=stepinfo(outpos,time)
```

```
fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

7.2 DOB Position control Time Response Characteristics Result

Overshoot   : 17.55%

Rise Time   : 0.1821 seconds

Settling Time: 1.1668 seconds

7.3 DOB Position control Experimental Data Code

```
%% DOB Experiment result

load dob.mat

%DOB Graph
t=dob_pos_posest(1:6261,1)
posest_data=dob_pos_posest(1:6261,2)
pos_data=dob_pos_posest(1:6261,3)
dis_data=dob_F(1:6261,2)


figure (1)
plot(t,posest_data,t,pos_data,'MarkerSize',30)
legend('posest','pos')
title('DOB  Kp=700 Kd=3 Filter=750')
grid on

figure(2)
plot(t,dis_data,'DisplayName','disturbance')
grid on

info1=stepinfo(posest_data,t,0.05)

fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

7.4 NCM Position control MATLAB Code

```
%% NCM POLE PLACEMENT METHOD
% Assume C=C^tilde P=P^tilde
clc
s=tf('s')
J=0.4;
b=25;

P=(1)/(((J*s^2+b*s)));
Pe=(1)/(((J*s^2+b*s)));
%% DESIGN CONTROLLER WITH POLE PLACEMENT By using TS and Damping
% Define OS and Ts values for testing
```

```
Ts=0.1;
damp=1;
% Call the desired damping ratio and frequency from a custom function
[freq] = freq_calculator(Ts,damp);

syms  b_s K2_s K1_s
% Set up the system of equations
eq1 = b_s==0.4;
eq2 = 2*freq*damp*b_s==K2_s;
eq3 = freq^2*b_s==K1_s;

% Solve the system of equations for a, b, Kc, and Td
sol = solve([eq1, eq2, eq3], [ b_s, K2_s, K1_s]);

% Convert symbolic solutions to numerical values and round to 2 digits
b2 = double(sol.b_s);
K2 = double(sol.K2_s);
K1 = double(sol.K1_s);

Kp =K1 ;
Kd = K2;
C = (Kd*s + Kp);
% controller of estimate model
Kpe =Kp;
Kde =Kd;
Ce = (Kde*s + Kpe);

num=[K2 K1]
den=[0.4 K2+b K1]
CL2tf=tf(num,den)


%% Root Locus-Closed-Loop Transfer Function Analysis

figure(1)
rlocus(CL2tf)
p=pole(CL2tf)
display(p(1), 'pole1')
display(p(2), 'pole2')

S=1/(1+C*P)
figure(2)
bode(S,{1,10000})
title('Sensivity Plot Disturbance Rejection');

% Bode Plot- Complemantary Sensivity
T=minreal((C*P)/(1+C*P))

figure(3)
bode(T,{1,10000})
title('Complemantary Sensivity ');

sim('NCM_pos_poleplacement_design.slx')

% output data export Simulink from Matlab  by using scope
outpos=out.ScopeData2(:,2);
time=out.ScopeData2(:,1);
```

```
figure(4)
plot(time,outpos)

info1=stepinfo(outpos,time,0.15)

fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

7.5 NCM Position control Time Response Characteristics Result

Overshoot : 0.78%

Rise Time : 0.0611 seconds

Settling Time: 0.0926 seconds

7.6 NCM Position control Experimental Data Code

```
%% NCM graph
load ncm_damp_b25.mat
t2=ncm_damp_pos_posest(1:4834,1)
ncm_damp_posest_data=ncm_damp_pos_posest(1:4834,2)
ncm_damp_pos_data=ncm_damp_pos_posest(1:4834,3)
ncm_damp_dis_data=ncm_damp_dist(1:4834,2)

figure(3)
plot(t2,ncm_damp_posest_data,t2,ncm_damp_pos_data)
legend('ncm posest','ncm pos')
title('NCM  b=25 Kp=700 Kd=3 Filter=750')
grid on

figure (4)
plot(t2,ncm_damp_dis_data,'DisplayName','ncm_damp_disturbance')
grid on

info2=stepinfo(ncm_damp_posest_data,t2,0.05);

fprintf('NCM Overshoot     : %.2f%%\n', info2.Overshoot);
fprintf('NCM Rise Time     : %.4f seconds\n', info2.RiseTime);
fprintf('NCM Settling Time : %.4f seconds\n', info2.SettlingTime);
```

7.7 DOB Force control MATLAB Code

```
%% DISTIRBANCE OBSERVER FORCE CONTROL
% STA1104-1116 actuator parameters
% Assume Kb=Kt=K and L is negligible
s=tf('s')
J=0.4 %0.00535;
Jn=J*1; %kg/m^2
K=10.38 %0.98;
Kn=K*(Jn/(J*1)); % Kb=Kt  N.m/A
gdob=100 %rad/s
F=1; %N
```

```
% environment properties
De=0.05; % Ns/m
Ke=1; % N/m

Kp=900;
Kd=60;
Cf=Kp+Kd*s

sim('DOB_force_design.slx')
sim('DOB_force_design_nodamp.slx')

% step data taken from simulation result
outforce=out.force;
time=out.tout;

info1=stepinfo(outforce,time,1)
fprintf('Overshoot     : %.2f%%\n', info1.Overshoot);
fprintf('Rise Time     : %.4f seconds\n', info1.RiseTime);
fprintf('Settling Time : %.4f seconds\n', info1.SettlingTime);
```

7.8 DOB Force control Time Response Characteristics Result

Overshoot : 3.95%

Rise Time : 0.0284 seconds

Settling Time: 0.2406 seconds

8) REFERENCE

[1] Abeykoon, A.M.H.S., and Senevirathne, H.R. 2016. "Disturbance Observer Based Current Controller for a Brushed DC Motor", IEEE Transactions on Industrial Electronics, 63(6), 3412-3420.

[2] K., Ohnishi, K., and Komoriya, K. 1994. "A Design Method for Manipulator Control Based on Disturbance Observer", IEEE Transactions on Robotics and Automation, 10(5), 681-694.

[3] Katsura, S., Matsumoto, Y., and Ohnishi, K. 2016. "Modeling of Force Sensing and Validation of Disturbance Observer for Force Control", IEEE Transactions on Industrial Electronics, 63(4), 2152-2160.

[4] Koksal, M. 2013. "Position Control of a Permanent Magnet DC Motor by Model Reference Adaptive

Control", IEEE Transactions on Industrial Electronics, 60(9), 3487-3495.

[5] MathWorks. “Control Systems in Practice: Part 10 - Nichols Chart, Nyquist Diagram, and Bode Plot.” https://www.mathworks.com/support/search.html/videos/control-systems-in-practice-part-10-nichols-chart-

[6] MathWorks. “Nyquist.” https://www.mathworks.com/help/ident/ref/dynamicsystem.nyquist.html

[7] MathWorks. “Root Locus.”

https://www.mathworks.com/help/control/ref/dynamicsystem.rlocus.html

[8] Mitsantisuk, C., Katsura, S., and Ohnishi, K. 2013. "Force Control of Human–Robot Interaction Using Twin Direct-Drive Motor System Based on Modal Space Design", IEEE Transactions on Industrial

[9] Nise, N. 2011. "Frequency Response." Control Systems Engineering. 6th ed. Hoboken, NJ: John Wiley & Sons.

[10] Nise, N. 2011. "Root Locus Techniques." Control Systems Engineering. 6th ed. Hoboken, NJ: John Wiley & Sons.

[11] Nise, N. 2011. "Stability." Control Systems Engineering. 6th ed. Hoboken, NJ: John Wiley & Sons.

[12] Ohnishi, K., Shibata, M., and Murakami, T. 2018. "Motion Control for Advanced Mechatronics", IEEE

[13] Ogata, K. 2010. "Mathematical Modeling of Control Systems." Modern Control Engineering. 5th ed. Upper Saddle River, NJ: Prentice Hall.

[14] Ogata, K. 2010. "PID Controllers and Modified PID Controllers." Modern Control Engineering. 5th ed. Upper Saddle River, NJ: Prentice Hall.

[15] Pietrusewicz, K., Waszczuk, P., and Kubicki, M. 2016. "MFC/IMC Control Algorithm for Reduction of Load Torque Disturbance in PMSM Servo Drive Systems", IEEE Transactions on Industrial Electronics,

[16] Sariyıldız, E., and Ohnishi, K. 2014. "A Guide to Design Disturbance Observer Based Motion Control Systems", IEEE Transactions on Industrial Informatics, 9(4), 1810-1822.

[17] Sariyıldız, E. 2020. "A Stability Analysis for the Reaction Torque Observer-based Sensorless Force Control Systems", IEEE Transactions on Industrial Electronics, 67(10), 8571-8579.

[18] Sariyıldız, E., and Ohnishi, K. 2018. "On the Explicit Robust Force Control via Disturbance Observer", IEEE Transactions on Industrial Electronics, 65(9), 7201-7209.

[19] Sariyıldız, E., and Ohnishi, K. 2021. "Stability and Robustness of Disturbance-Observer-Based Motion Control Systems", IEEE Transactions on Industrial Electronics, 68(5), 3982-3990.

[20] Shimada, A. 2012. "Basic of Disturbance Observer." Disturbance Observer for Advanced Motion

[21] Tewari, A. (2002). *Modern Control Design with MATLAB and SIMULINK*. Wiley.

[22] Bazman, M. (2019). *Development of a Novel 4-DOF Wrist-Gripper Mechanism for Robotic Minimally Invasive Surgery* (Master's Thesis). Department of Mechanical Engineering, Supervisor: Assoc. Prof. Dr. Uğur Tümerdem.