Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

Introduction

In this lab, we designed controllers in order to control the position and angular speed of the Qnet DC Motor. In this lab, we observed the output and performance of the system when PI and PD controllers are in use. We designed controllers to control the position and speed of the Qnet DC Motor to track a square wave with certain design specifications.

Question 1

a) To define the transfer function, the following code can be used:

```
% Define the transfer functions here
gain = 28.5;
tau = 0.16;

H = gain/(tau*s + 1);
G = (1/s)*(gain)/(tau*s+1);
```

b) This can be used the following Matlab code:

```
%Define the time here
T = 5;        % Simulation duration
dt = 0.01;     % Simulation step time
time = 0:dt:T;
```

c)

PI controller for H(s)
K = 0.0366 + 0.686 / s
Kp = 0.0366
Ki = 0.686

PD controller for G(s)
K = 0.686 + 0.0366 * s
Kp = 0.686
Ki = 0.0366

```
           1.042 s + 19.55
Hcl = -----------------------------------
       0.16 s^2 + 2.042 s + 19.55

           1.042 s + 19.55
Gcl = -----------------------------------
       0.16 s^2 + 2.042 s + 19.55
```

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

d) Simulation of the output and input

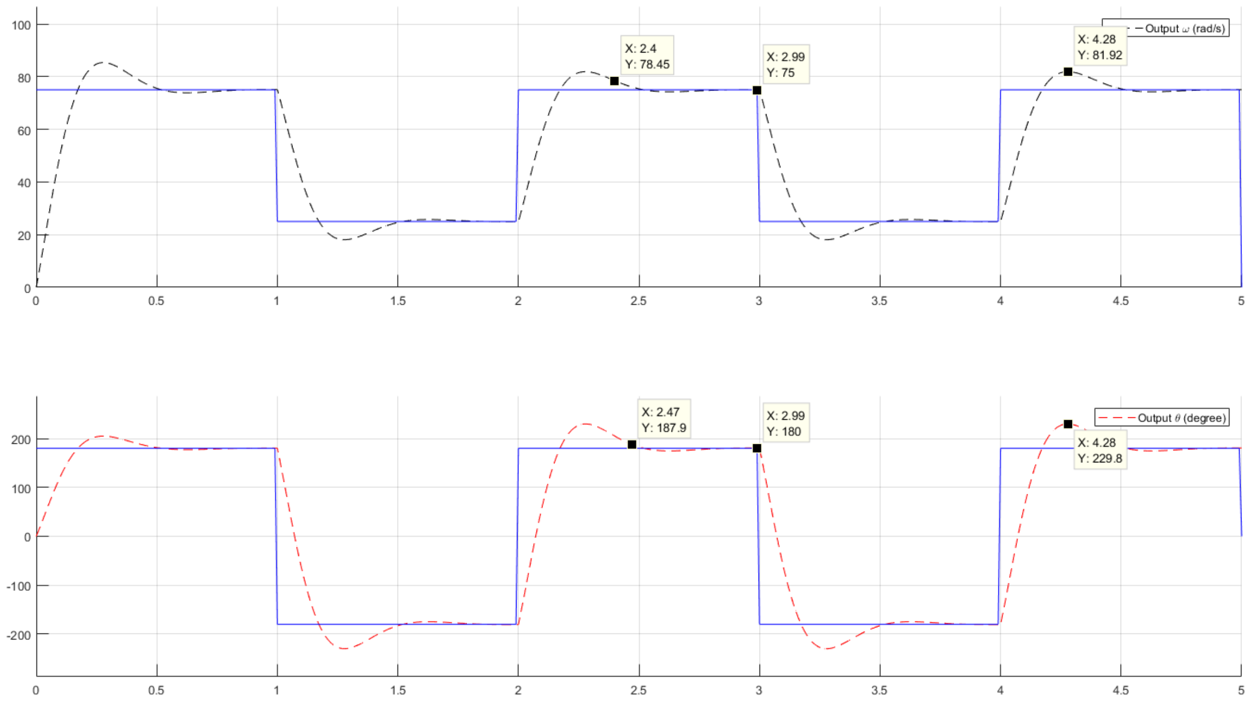

Figure 1: Simulation of system with Kp = 0.0366, Ki = 0.686 for ω and Kp = 0.686, Kd = 0.0366 for θ

For ω(s), the 5% settling time of the system can be observed from the graph to be around 0.40 seconds, the overshoot is 6.92, which is 9.23%, and the steady state error is roughly 0. For the PI controller, Kp = 0.0366, Ki = 0.686. The output of the simulation can be seen below:

For θ(s), the 5% settling time of the system can be observed from the graph to be around 0.47 seconds, the overshoot is 49.8, which is 27.67%, and the steady state error is roughly 0. For the PI controller, Kp = 0.686, Kd = 0.0366.

Simulation of the output and input, after tuning the controller to meet Zero steady state-error, 5%-Settling time of 0.25 s and Overshoot less than 10%.

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

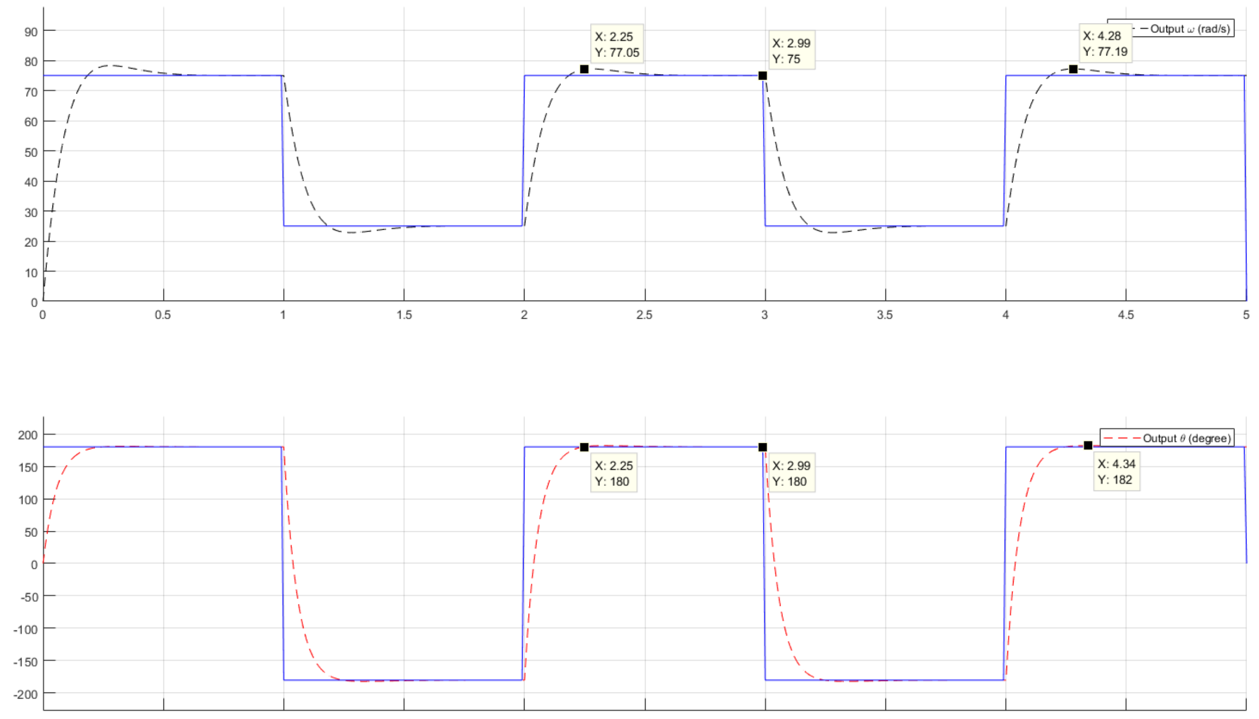

Figure 2: Kp = 0.07, Ki = 0.686 for ω, Kp = 0.686, Kd = 0.1 for θ

For ω(s), the 5% settling time of the system can be observed from the graph to be around 0.25 seconds, the overshoot is 2.05, which is 2.73%, and the steady state error is roughly 0. For the PI controller, Kp = 0.07, Ki = 0.686. The output of the simulation can be seen below:

For θ(s), the 5% settling time of the system can be observed from the graph to be around 0.25 seconds, the overshoot is 2, which is 1.11%, and the steady state error is roughly 0. For the PI controller, Kp = 0.686, Kd = 0.1.

Question 2

1) The 5% settling time of the system can be observed from the graph shown below to be around 0.34 seconds, the overshoot is 10.26, which is 13.67%, and the steady state error is roughly 0.04. The value of Kp = 0.0366, Ki = 0.686. The output of the QNet DC Motor can be seen below:

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

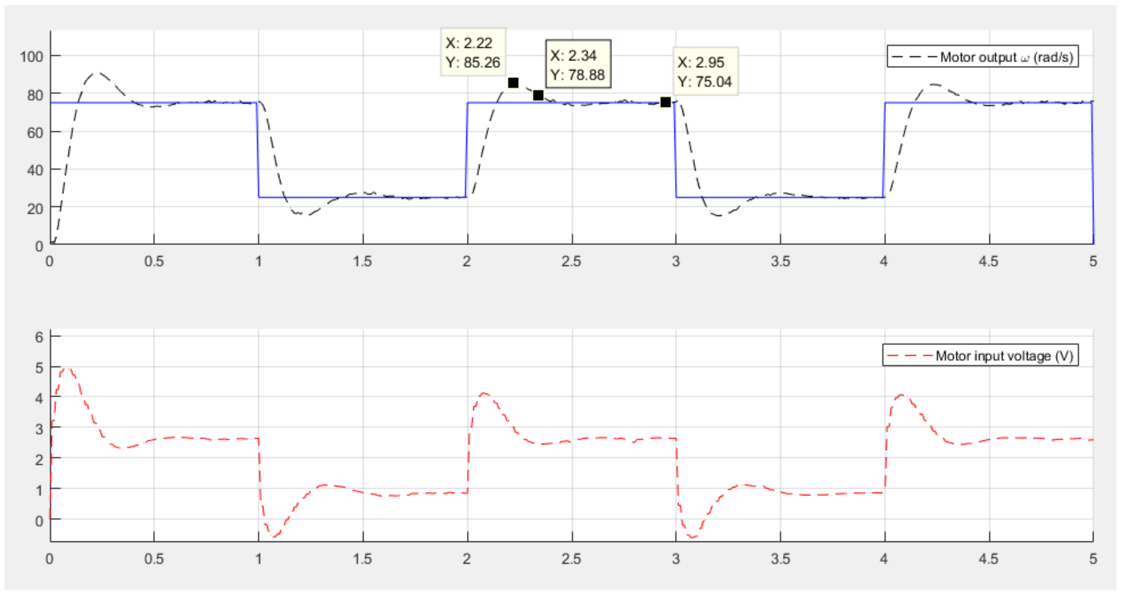

Figure 3: Motor Output Kp = 0.0366, Ki = 0.686

2) The 5% settling time of the system can be observed from the graph shown below to be around 0.24 seconds, the overshoot is 4.56, which is 6.08%, and the steady state error is 0. The values of the controller are Kp = 0.7 and Ki = 0.06. The output is shown below:

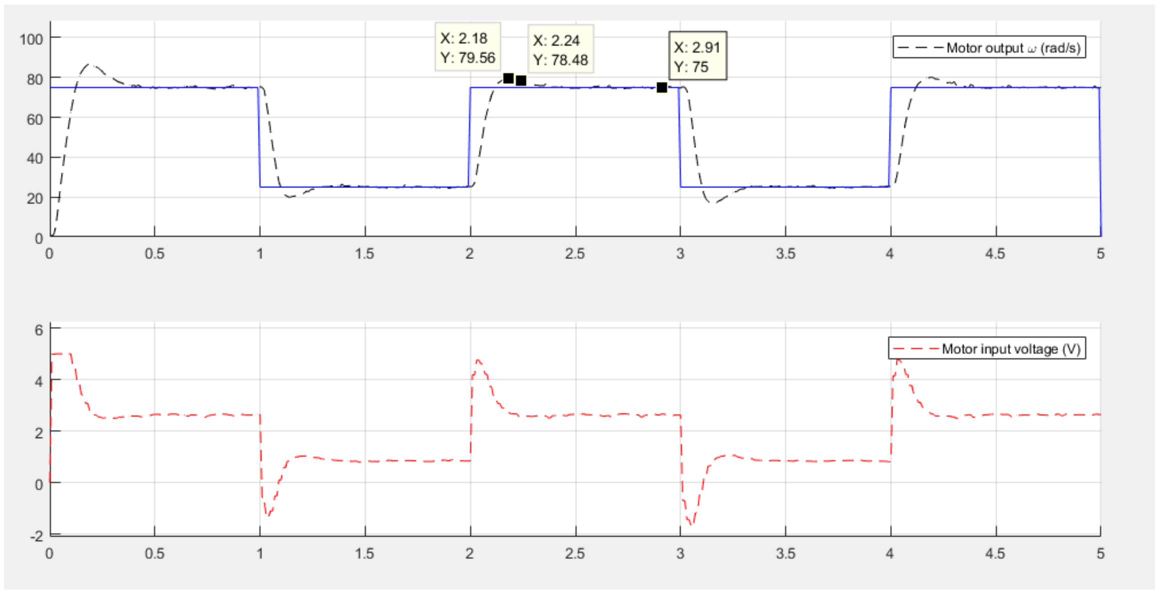

Figure 4: Motor Output Kp = 0.7, Ki = 0.6

3) We partially have saturation at around time = 0, however throughout the regular cycling process of the system, there is no saturation and the motor input voltage ranges from +5V to -1.8V.

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

Question 3

1) Using the controller gains found in Question 1, the output is observed as follows:

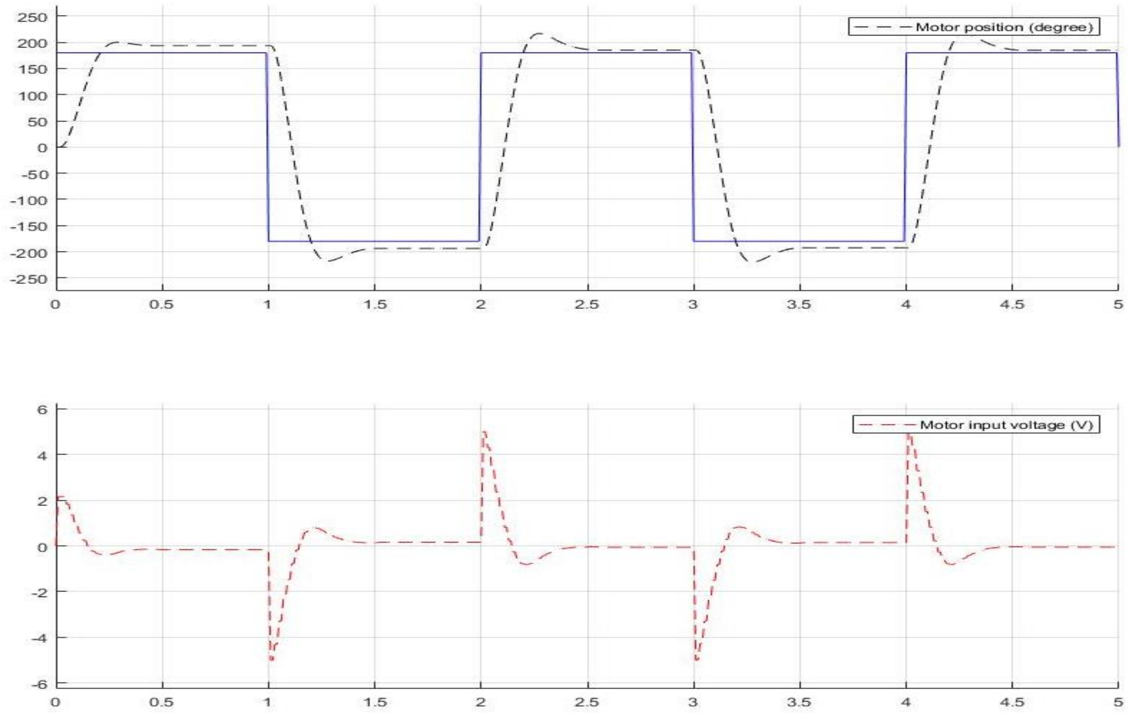

Figure 5: Motor Output Kp = 0.07, Ki = 0.686 for ω, Kp = 0.686, Kd = 0.0366 for θ

2) Beginning from the values derived in Question 1, namely, Kp = 0.686, Ki = 0.0366 we tune the controller system to the values of Kp = 0.9, Kd = 0.0366. The result is shown below:

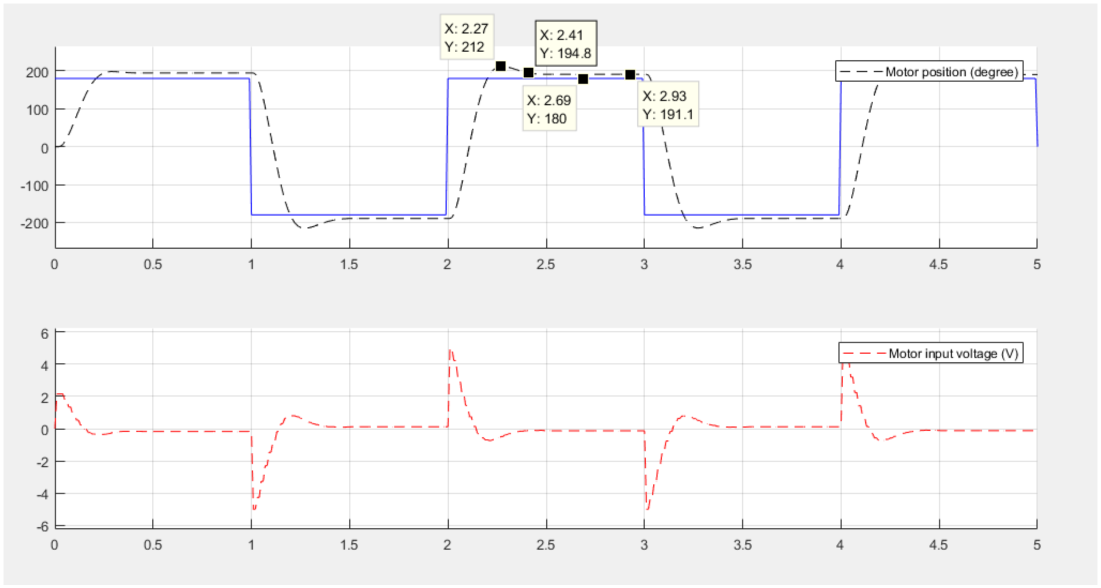

Figure 6: Motor Output Kp = 0.686, Ki = 0.0366 for ω, Kp = 0.9, Kd = 0.0366 for θ

The value of the system overshoot is 32, or 17%. The rise time is 0.27 seconds and the 5% settling time is infinite as the system never settles within 5% of the input. We were unable to meet the specifications during the lab time due to shortage of time.

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

Question 4

1) The equation to calculate the number of cycles the motor must make is given by:

$$\text{Number of cycles} = \frac{\text{distance(m)}}{\pi \times 0.05}$$

As we want the motor to travel 0.4m forward, the number of cycles it must complete is given by: $(0.4)/(pi * 0.05) = 2.546$cycles.

This number must be multiplied by 2*pi to get the desired value in radians, this can be calculated to be ***16 radians*** to move the motor forward 0.4m. To move the motor in reverse, the value will be calculated using the same method to be ***8 radians***.

The designed PD controller is Kp = 0.9, Kd = 0.05. The output is shown below:

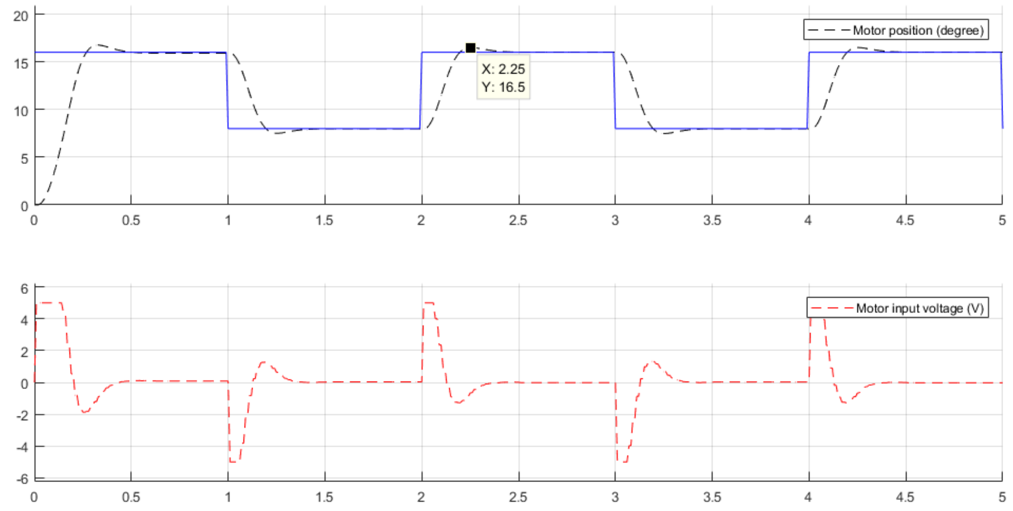

Figure 7: Ouput for Kp = 0.9, Kd = 0.05

2) The physical interpretation of the rise time is a measure of how quickly the motor will be able to respond to stimulus. The settling time would be the time required for the motor speed to stabilize to within a desired settling band. The overshoot is the maximum amount of system deviates from the desired input signal, in our case, it is measured as the amount which the motor goes beyond the desired signal and must correct.

3) The maximum distance the car can travel will be dictated by the maximum speed that the car can travel at a given time. By increasing our input signal to a high number, we can determine what is the distance the motor can travel in a given time frame, in our case, the distance it can travel in one second. This result is shown in the graph below:

ECSE 307 - Lab 6

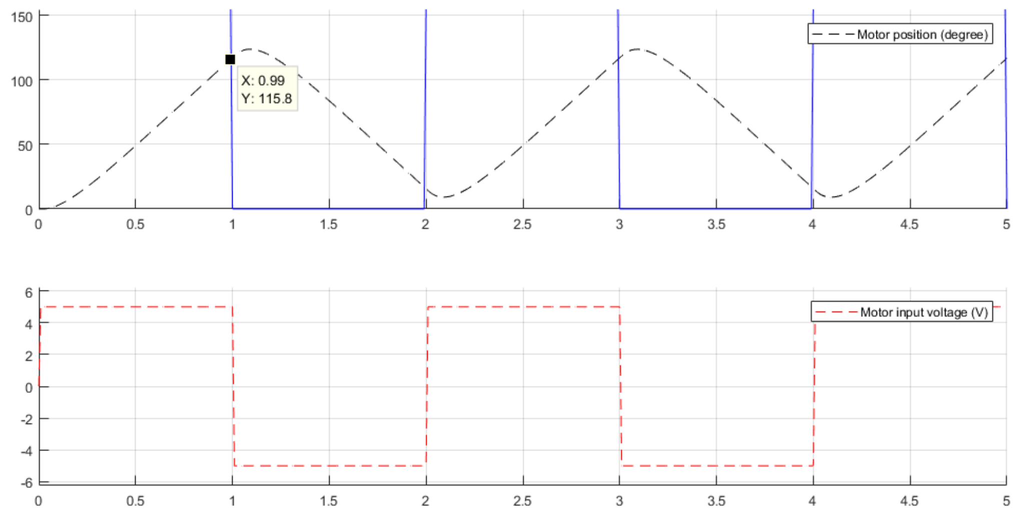

Figure 8: Kp = 1000, Kd = 0.05

As can be seen in the graph seen above, the maximum rotation the motor is able to do in one second is 115.8 degrees or 0.0527 m.

Conclusion

The purpose of this laboratory was to tune given controllers, PI and PD, to meet certain design specifications for the system, in terms of rise time, settling time overshoot. We also got to understand the physical interpretation of rise time, settling time and overshoot in a said particular system. We also learned that the output of a system would become saturated as we increased the input to the system in a practical scenario, in contrast to a theoretical one where output would just be the product of the transfer function and the input with no bound for saturation. The end goal was to design controllers to control the position and speed of the Qnet DC Motor to track a square wave with certain design specifications and at the same time understand its saturation limit.

Appendix:

```
% ------------------------------------------------------------------
------%
% title    : Lab 6
%
% subtitle : Speed and Position Control of Qnet DC Motor
%
% date     : Week of November 20, 2017
%
% ------------------------------------------------------------------
------%
```

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

```
%% Initialization (can be called only once)
% Clear all input and output from the Command Window
clc,

% Change the default figure window style to docked
set(0,'DefaultFigureWindowStyle', 'docked')

% We start by adding `Interface` sub-directory to Matlab's search
path.
addpath('Interface');

% Create an object handle to interface with Qnet DC Motor
Motor = QnetDCMotor();

s = tf('s')

%% PI and PD Controllers Design in MATLAB


%
% % Define the transfer functions here
% gain = 28.5;
% tau = 0.16;

H = gain/(tau*s + 1);
G = (1/s)*(gain)/(tau*s+1);
%Define the time here
T = 5;       % Simulation duration
dt = 0.01;    % Simulation step time
time = 0:dt:T;

%Complete the reference signals here
w_ref = 75*(time<1) + 25*(time>=1 & time<2) + 75*(time>=2 & time<3)
+ 25*(time>=3 & time<4) + 75*(time>=4 & time<5)
p_ref = 180*(time<1) + -180*(time>=1 & time<2) + 180*(time>=2 &
time<3) + -180*(time>=3 & time<4) + 180*(time>=4 & time<5)

% Plot results here
figure(1); clf;
subplot(2,1,1)
hold on
plot(time, w_ref, '--k')
ylim([0 1.25*max(w_ref)])
xlim([0 T])
legend('Reference \omega (rad/s)')
grid on
```

```
subplot(2,1,2)
hold on
plot(time, p_ref, '--r')
ylim([1.25*min(p_ref) 1.25*max(p_ref)])
xlim([0 T])
legend('Reference \theta (degree)')
grid on

% Design PI and PD controllers for speed and position control here
% C_PI = pidtune(H,'PI');
% C_PD = pidtune(G,'PD');

% Change the values of the gains in controllers here
C_PI = pid(0.07,0.686,0);
C_PD = pid(0.686,0,0.1);

% Find the closed loop controlled transfer function here
Ho = series(C_PI,H);
Hcl = feedback(Ho,1);

Go = series(C_PD,G);
Gcl = feedback(Go,1);

%Find the output here
w_out = lsim(Hcl, w_ref, time);
p_out = lsim(Gcl, p_ref, time);

% % Plot results
figure(3); clf;
subplot(2,1,1)
hold on
plot(time, w_out, '--k')
hold on
plot(time,w_ref,'b')
ylim([0 1.25*max(w_out)])
xlim([0 T])
legend('Output \omega (rad/s)')
grid on
subplot(2,1,2)
hold on
plot(time, p_out, '--r')
hold on
plot(time,p_ref,'b')
ylim([1.25*min(p_out) 1.25*max(p_out)])
xlim([0 T])
legend('Output \theta (degree)')
grid on
```

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

```
%% PI Controller for Qnet DC Motor Speed Control

Kp = 0.06;
Ki = 0.7;

% Drive the Qnet DC Motor

w_error = zeros(size(time)); %Initialize an error array.

Motor.reset();  % reset the motor internal variable

for n = 1:length(time)
 t_ = time(n);
 w_ = Motor.velocity(t_);
 w_error(n) = w_ref(n)-w_;
 u_ = Kp* w_error(n) + Ki*sum(w_error)*dt;
 Motor.drive(u_, t_, dt);
end
Motor.off();

% Get results of driving Qnet DC Motor

w_motor = Motor.velocity(0, T);
u = Motor.voltage(0, T);

% % Plot results
figure(3); clf;
subplot(2,1,1)
hold on
plot(time, w_motor, '--k')
hold on
plot(time,w_ref,'b')
ylim([0 1.25*max(w_motor)])
xlim([0 T])
legend('Motor output \omega (rad/s)')
grid on
subplot(2,1,2)
hold on
plot(time, u, '--r')
ylim([1.25*min(u) 1.25*max(u)])
xlim([0 T])
legend('Motor input voltage (V)')
grid on

%% PD Controller for Qnet DC Motor Position control
```

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

ECSE 307 - Lab 6

```
Kp = 0.9;
Kd = 0.0366;

% % Drive the Qnet DC Motor
p_error = zeros(size(time)); %Initialize an array for error.

p_ref = p_ref*pi/180; %change the reference signal from degree to
radian

Motor.reset();  % reset the motor internal variable
for n = 1:length(time)  % drive the control signal as well as the
output of the system
 t_ = time(n);
 p_ = Motor.angle(t_);
 p_error(n) = p_ref(n)-p_;
 if(n==1)
     u_ = Kp * p_error(1) + Kd/dt * (p_error(1)-pi);
 else
     u_ = Kp * p_error(n) + Kd/dt * (p_error(n)-p_error(n-1));
 end

 Motor.drive(u_, t_, dt);
end
Motor.off();

% % Get results of driving Qnet DC Motor

p_motor = Motor.angle(0, T);
u = Motor.voltage(0, T);

p_motor = p_motor*180/pi; % change the units again from radian to
degree.
p_ref = p_ref*180/pi;


% Plot results
figure(4); clf;
subplot(2,1,1)
hold on
plot(time, p_motor, '--k')
hold on
plot(time,p_ref,'b')
ylim([1.25*min(p_motor) 1.25*max(p_motor)])
xlim([0 T])
legend('Motor position (degree)')
grid on
subplot(2,1,2)
```

```
hold on
plot(time, u, '--r')
ylim([1.25*min(u) 1.25*max(u)])
xlim([0 T])
legend('Motor input voltage (V)')
grid on

%% PD Position Control- Question 4
a = 10*2*pi*(0.4/(pi*0.05));
b = 0;
%
p_ref = a*(time<1) + b*(time>=1 & time<2)+a*(time>=2 & time<3)...
   +b*(time>=3 & time<4)+a*(time>=4 & time<5)+b*(time>=5);
%
Kp = 0.9;
Kd = 0.05;

% Drive the Qnet DC Motor
p_error = zeros(size(time)); %Define an array for error

Motor.reset();  % reset the motor internal variable
for n = 1:length(time)
    t_ = time(n);
    p_ = Motor.angle(t_);
    p_error(n) = p_ref(n)-p_;
    if(n==1)
        u_ = Kp * p_error(1) + Kd/dt * (p_error(1)-pi);
    else
        u_ = Kp * p_error(n) + Kd/dt * (p_error(n)-p_error(n-1));
    end
    Motor.drive(u_, t_, dt);
end
Motor.off();

% % Get results of driving Qnet DC Motor
p_motor = Motor.angle(0, T);
u = Motor.voltage(0, T);

% Plot results
figure(5); clf;
subplot(2,1,1)
hold on
plot(time, p_motor, '--k')
hold on
plot(time,p_ref,'b')
ylim([1.25*min(p_motor) 1.25*max(p_motor)])
xlim([0 T])
```

Ali Shobeiri - 260665549
Ismail Faruk - 260663521

```
legend('Motor position (degree)')
grid on
subplot(2,1,2)
hold on
plot(time, u, '--r')
ylim([1.25*min(u) 1.25*max(u)])
xlim([0 T])
legend('Motor input voltage (V)')
grid on
```