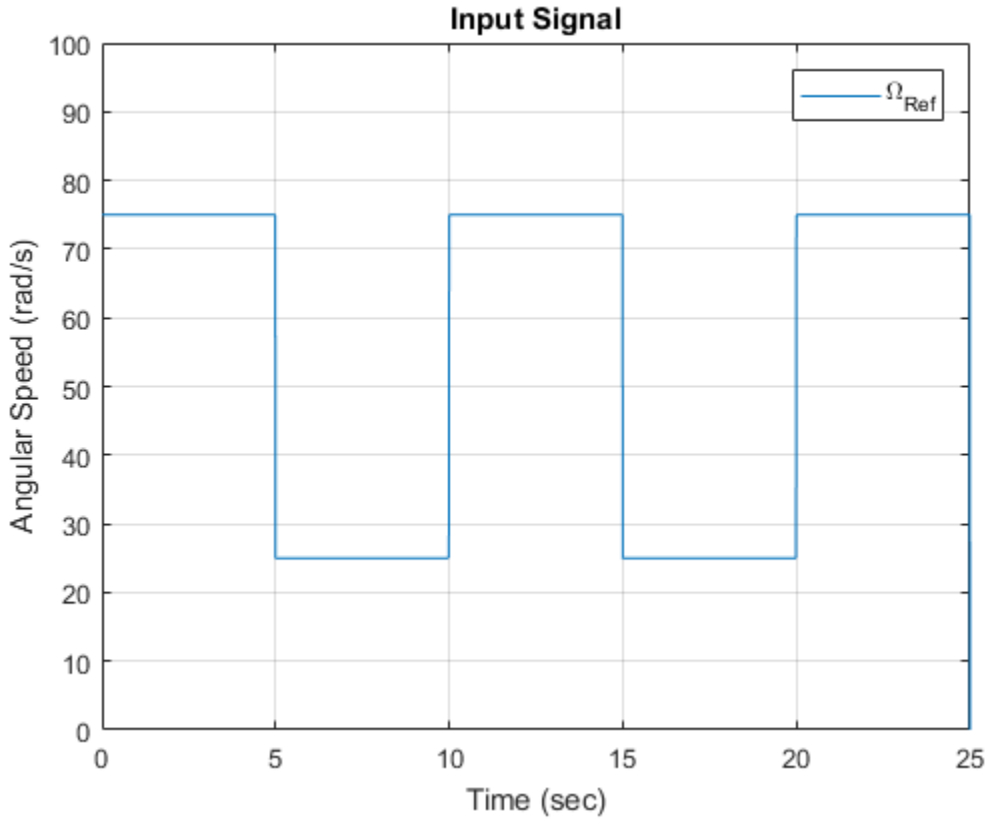
Introduction

We used the identified transfer function of the Qnet DC Motor in lab-03 to design a controller that matches some desired performances. We used MATLAB to drive the Qnet DC Motor and evaluate the designed controller.

Question 1

1. A square wave with amplitude 25 Ω, DC offset 50 Ω and frequency 0.1 H_z



2. Value of the input voltage V = Angular Speed/28.5

For Angular Speed = 25 Ω,

V = 25/28.5

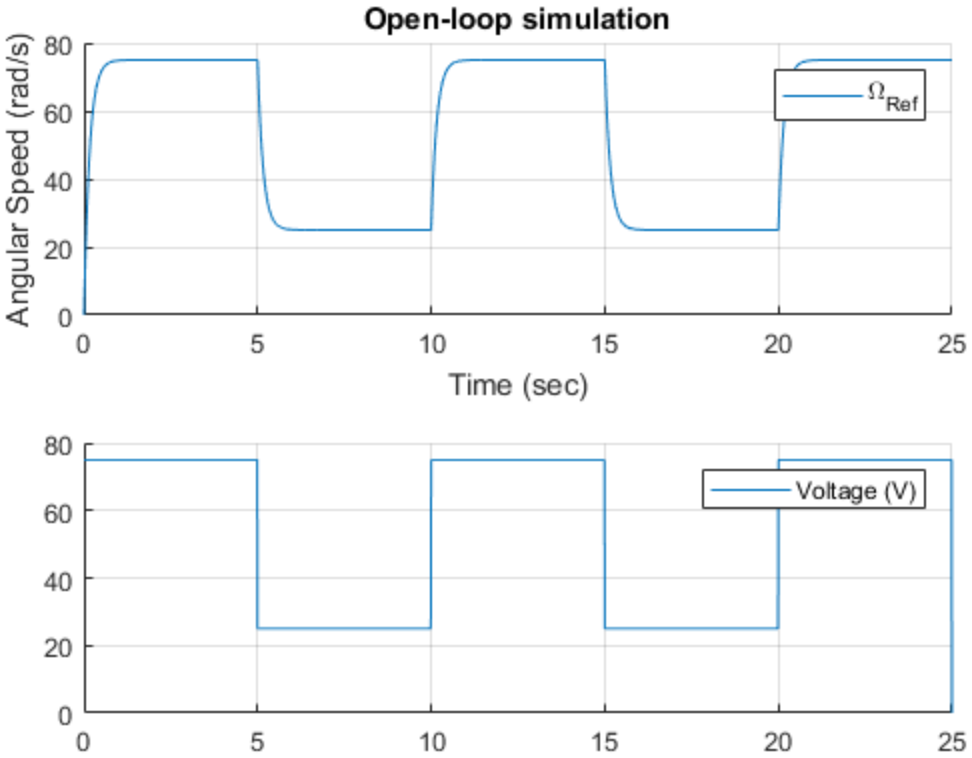
V = 0.8772 V

For Angular Speed = 75 Ω,

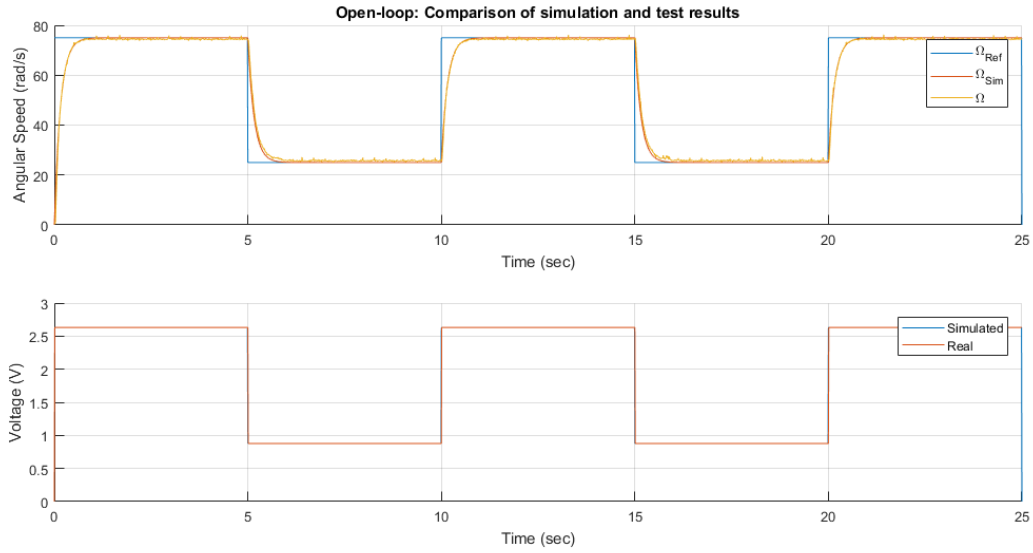
V = 75/28.5 V

V = 2.6316 V

3. Response of the transfer function H(s) using simulation, plotting both input and output



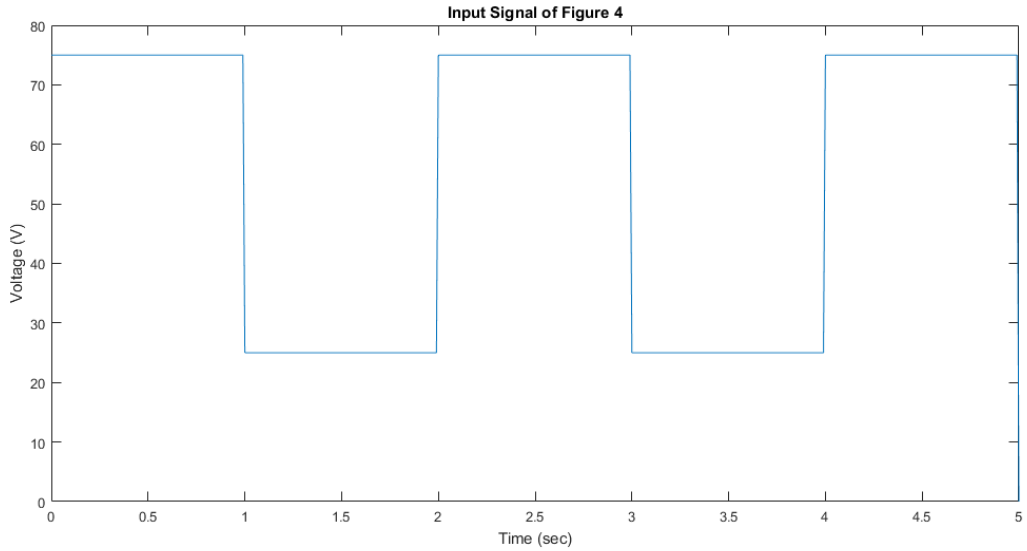
4. Response of the transfer function H(s), superimposing the Qnet DC Motor and the simulation



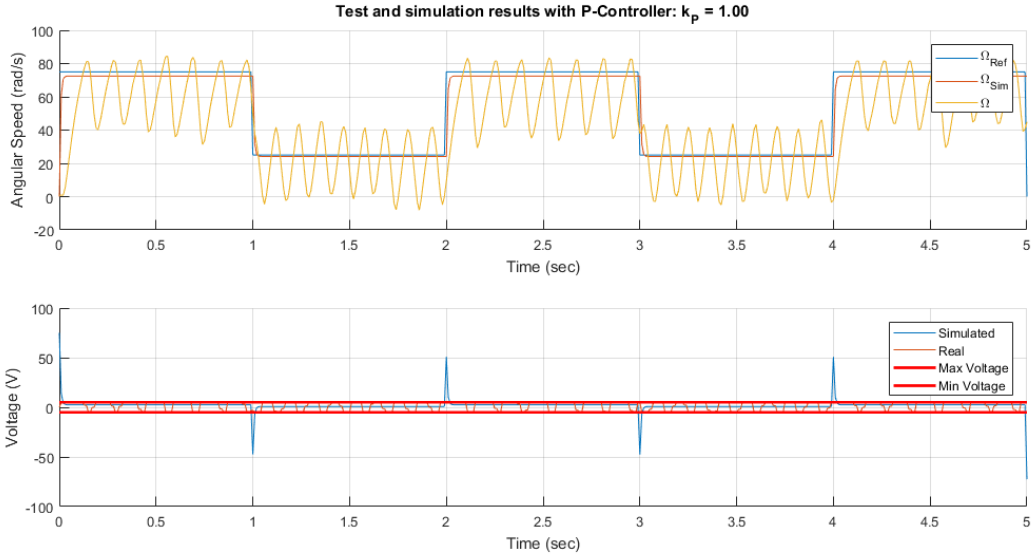
5. Outcome of the discussion: The proposed model H(s) describes the Qnet DC Motor with satisfactory. The model tracks the motor output, except for a difference in the rise time. If the frequency is five times bigger, the output would be more curved and the model will not be able to properly track the Qnet DC Motor.

Question 2

1. A square wave with amplitude 25 Ω, DC offset 50 Ω and frequency 0.1 Hz



2. Closed-loop transfer function
   $H(s) = \Omega(s) / \Omega_{ref}(s) = \{28.5*k_p / (28.5*k_p + 1)\} / [\{0.16*s / (28.5*k_p + 1)\} + 1]$



3. Controller design

For Overshoot < 10%, as the system is in first order, the Overshoot is 0 for k_p = real number

For 5%-settling time < 0.25s,

$3*\tau < 0.25$ – equation 1

$\tau = 0.16 / (28.5*k_p + 1)$ – equation 2

Solving equation 1 and 2, we get

$k_p > 0.0323$

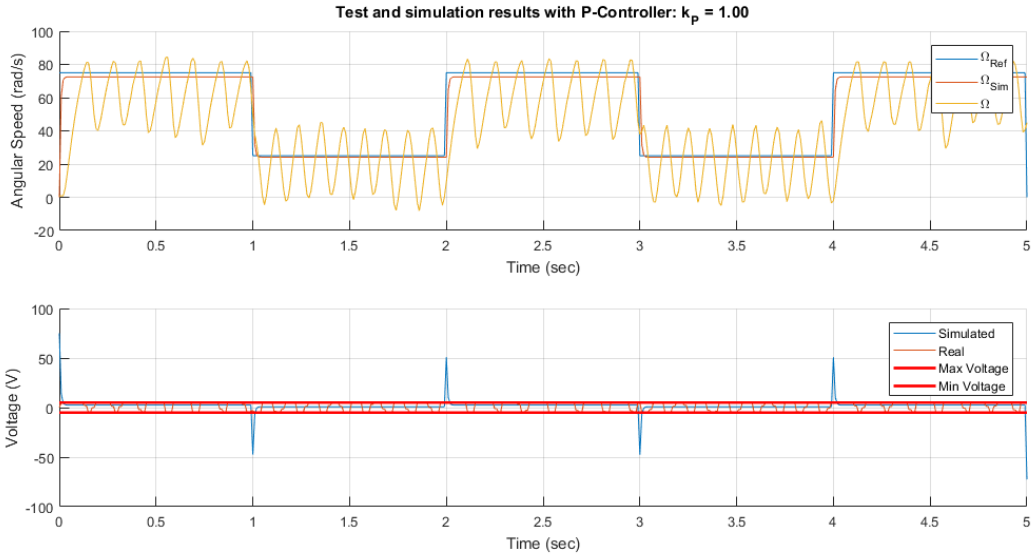
For steady-state error < 5%

Which would mean the DC Gain > 95 %

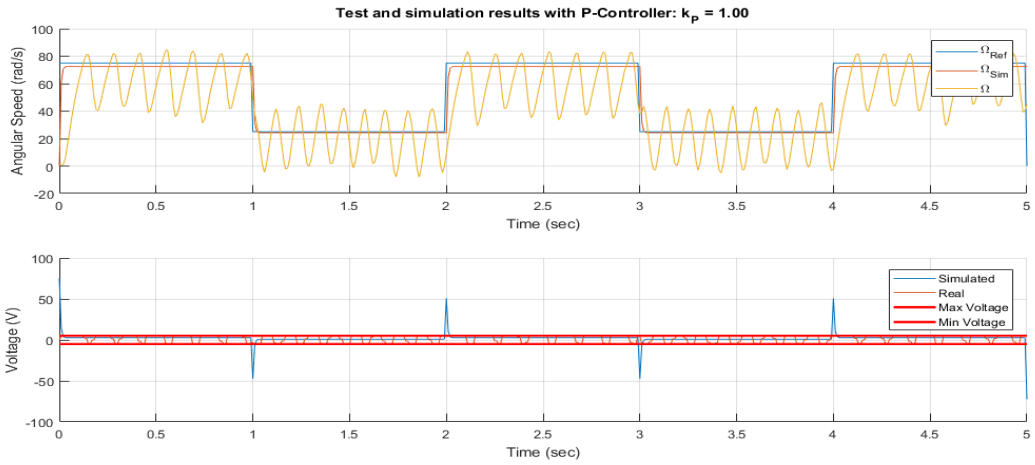
$=28.5*k_p / (28.5*k_p + 1 > 0.95$

$=k_p > 0.67$

4. Response of the transfer function using simulation, plotting both input and output



5. Response of the transfer function using simulation, plotting both input, output and controlled output



No, because the addition of the controller has introduced oscillation in the system output, which is not desired.

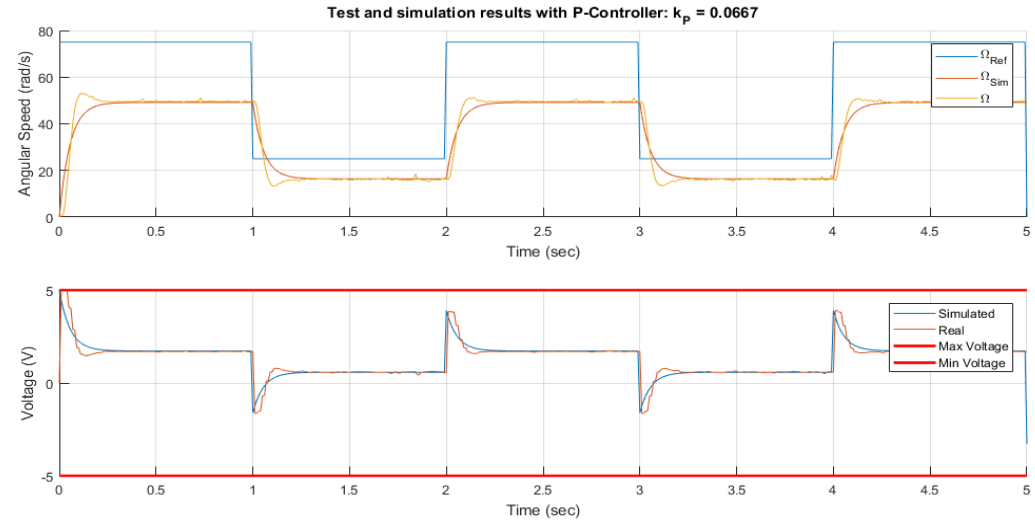
6. Voltage hits saturation at t = 0 s, 1 s, 2 s, 3 s, 4 s, 5 s.
transfer function $V(s) / \Omega_{ref}(s) = \{k_p + (k_p * \tau s)\} / \{(\tau s + G*k_p) + 1\}$
when t = 0, s goes to infinity, $V(s) / \Omega_{ref}(s) = k_p$

at t = 0, $\Omega_{ref}(s)$ = 75 V.

To get V(s) < +-5 V, we get the inequality: $k_p * 75 < +-5$

We get, $k_p <= 0.07$

7. Taking k_p = 0.0667, response of the transfer function using simulation, superimposing the Qnet DC Motor and the simulation



8. Steady-state error:
(75-49.61) / 75 * 100 % = 33.85%

5%-Settling time:

49.61 + (49.61*0.05) = 52.09

49.61 – (49.61*0.05) = 47.13

System is only between these values from t = 0.15 s

Thus, 5%-Settling time = 0.15 s

Overshoot = 0

Conclusion

When we finished implementing the controller we saw that output of the Qnet DC Motor did not track the simulated output. That is because the controller designed was not in the specified range such that its steady-state error would be within 5%.

Appendix

```
% ------------------------------------------------------------------------%
% title    : Lab 5                                                        %
% subtitle : Speed Control of Qnet DC Motor                               %
```

```
% date     : Week of 06 November, 2017                                    %
% ------------------------------------------------------------------------%

%% 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();
%% ---------------------- 2. The open-loop system ---------------------- %%
%% 2.2 Example: Simulating a first order model of DC Motor
T = 25;       % Simulation duration
dt = 0.01;    % Simulation step time
time = 0:dt:T;% Time array

% % Define the transfer function
gain = 28.5; % DC gain
tau = 0.16;  % Time constant
H = tf(gain, [tau 1]); % Define the transfer function
% % Simulate
uSim = 0*(time<1) + 2*(time>=1 & time < T-1) + 0*(time >= T); % Voltage array
ySim = lsim(H, uSim, time);      % Simulate H
% % Plot results
figure(1); clf;
subplot(2,1,1)
title('Input Signal')
hold on
```

```
plot(time, ySim, '--k')
ylim([0 1.25*max(ySim)])
xlim([0 T])
xlabel('Time (sec)')
ylabel('Angular Speed (rad/s)')
legend('\Omega_{Sim}')
grid on
subplot(2,1,2)
hold on
plot(time, uSim, '--r')
ylim([0 1.25*max(uSim)])
xlim([0 T])
xlabel('Time (sec)')
ylabel('Voltage (V)')
legend('Simulated')
grid on


%% 2.3 Question 1
% 2.3.1. Squarewave signal


% Generate square wave signal
yRef = 75 *(time<5) + 25*(time>=5 & time < 10) + 75*(time >= 10 & time < 15) + 25*(time>= 15 &
time < 20) + 75*(time>= 20 & time < 25) + 25*(time > 25); % Voltage array
% figure(2);
% clf;
% plot(time, yRef)
% ylim([0 100])
% xlim([0 T])
% xlabel('Time (sec)')
% ylabel('Angular Speed (rad/s)')
% legend('\Omega_{Ref}')
% title('Input Signal')
% grid on
```

```
% 2.3.3. Simulation of Squarewave signal

% Generate square wave
uSim = yRef/28.5;
% Simulate
ySim = lsim(H, uSim, time);
% Plot results
% figure(3); clf;
% subplot(2,1,1)
% title('Open-loop simulation')
% hold on
%
% plot(time, ySim)
% % your_plots_for_output_goes_here
% xlabel('Time (sec)')
% ylabel('Angular Speed (rad/s)')
% legend('\Omega_{Ref}', '\Omega_{Sim}')
% grid on
% subplot(2,1,2)
% hold on
% % your_plots_for_command_goes_here
% plot(time, yRef)
% legend('Voltage (V)')
% grid on

% 2.3.4. Test of Square wave signal

% Drive the Qnet DC Motor
Motor.reset();  % reset the motor internal variable
for n = 1:length(time)
   t_ = time(n);
   u_ = uSim(n);
   Motor.drive(u_, t_, dt);
```

```
end
Motor.off();
% Get results of driving Qnet DC Motor
y = Motor.velocity(0, T);
u = Motor.voltage(0, T);
% Plot results

%%
figure(3); clf;
subplot(2,1,1)

title('Open-loop: Comparison of simulation and test results')
hold on
% your_plots_for_output_goes_here
plot(time, yRef)
plot(time, ySim)
plot(time, y)
xlabel('Time (sec)')
ylabel('Angular Speed (rad/s)')
legend('\Omega_{Ref}', '\Omega_{Sim}', '\Omega')
grid on
subplot(2,1,2)
hold on
% your_plots_for_command_goes_here
plot(time, uSim)
plot(time, u)
legend('Simulated', 'Real')
xlabel('Time (sec)')
ylabel('Voltage (V)')
grid on
%% ------------------------ 3. Profile tracking ------------------------ %%
% Specs:
% - 5%-settling time of 0.25s
```

```
% - Overshoot less than 5%
% - Steady-state error 5%
T = 5;       % Simulation duration
dt = 0.01;    % Simulation step time
time = 0:dt:T;% Time array

%% 3.2. Question 2: P-controller
% 3.2.1. Squarewave signal
yRef = 75 *(time<1) + 25*(time>=1 & time < 2) + 75*(time >= 2 & time < 3) + 25*(time>= 3 &
time < 4) + 75*(time>= 4 & time < 5) + 25*(time > 5);
figure()
plot(time, yRef);
xlabel('Time (sec)')
ylabel('Voltage (V)')
title("Input Signal of Figure 4")

% (3.2.4. & 3.2.7.) Simulation of P-controller

% Define the transfer function
Kp = 1/15;
C_P = pid(Kp);

Hol = series(C_P, H);
Hcl = feedback(Hol, 1);
Hu = feedback(C_P, H);
% Simulate
ySim = lsim(Hcl, yRef, time);
uSim = lsim(Hu, yRef, time);
% Plot results
figure(4); clf;
subplot(2,1,1)
title(sprintf('Simulation results with P-Controller: k_{P} = %04.2f', Kp))
hold on
% your_plots_for_output_goes_here
```

```
plot(time, yRef)
plot(time, ySim)
xlabel('Time (sec)')
ylabel('Angular Speed (rad/s)')
legend('\Omega_{Ref}', '\Omega_{Sim}')
grid on
subplot(2,1,2)
hold on
% your_plots_for_command_goes_here
plot(time, uSim)
plot(time, [Motor.MaxVoltage;Motor.MinVoltage]*ones(size(time)), 'r', 'linewidth', 2.0)
xlabel('Time (sec)')
ylabel('Voltage (V)')
legend('Simulated', 'Max Voltage', 'Min Voltage')
grid on

%% (3.2.5. & 3.2.7.) Test of P-controller

% % Drive the Qnet DC Motor
Motor.reset();  % reset the motor internal variable
for n = 1:length(time)
   t_ = time(n);
   y_ = Motor.velocity(t_);
   u_ = Kp*(yRef(n) - y_);
   Motor.drive(u_, t_, dt);
end
Motor.off();
% Get results of driving Qnet DC Motor
y = Motor.velocity(0, T);
u = Motor.voltage(0, T);
%% Plot results

figure(4); clf;
```

```
subplot(2,1,1)
title(sprintf('Test and simulation results with P-Controller: k_{P} = %04.4f', Kp))
hold on
% your_plots_for_output_goes_here
plot(time, yRef)
plot(time, ySim)
plot(time, y)
xlabel('Time (sec)')
ylabel('Angular Speed (rad/s)')
legend('\Omega_{Ref}', '\Omega_{Sim}', '\Omega')
grid on
subplot(2,1,2)
hold on
% your_plots_for_command_goes_here
plot(time, uSim)
plot(time, u)
plot(time, [Motor.MaxVoltage;Motor.MinVoltage]*ones(size(time)), 'r', 'linewidth', 2.0)
xlabel('Time (sec)')
ylabel('Voltage (V)')
legend('Simulated', 'Real', 'Max Voltage', 'Min Voltage')
grid on
```