UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 488 - Project 5 & 6

Prepared by:

Jianxiang (Jack) Xu [20658861]

13 April 2021

Table of Contents

1 Problem P5: Decentralized MIMO control of the aiming system

Notation clarification: $(row, column)$: $(1,1) = r_w \to w$, $(2,1) = r_w \to \theta$, $(1,2) = r_\theta \to w$, and $(2,2) = r_\theta \to \theta$.

1.1 (a) Justification from MIMO Bode Plot of $P_3(s)$

As we may observed in the Figure 1-1 below, we may observe the magnitude gain for both $(1,2)$ and $(2,1)$ are below $0\,[\text{dB}]$, and the phase margin is $0\,[\text{deg}]$. This is a great indication that the cross-channel interference is minimal. For the DC gain (steady-state), the diagonal terms $(1,1)$ and $(2,2)$ of the plant dominant the entire system. Specifically, the (1,1) term indicates that the force input $u_1 = F$ dominants the effect of base displacement $y_1 = w$, and has negligible effect on rod angle $y_2 = \theta$ as $(2,1)$ bode suggested. Similarly, the (2,2) term indicates the torque input $u_2 = \tau$ dominants the effect of the rod angle $y_2 = \theta$.

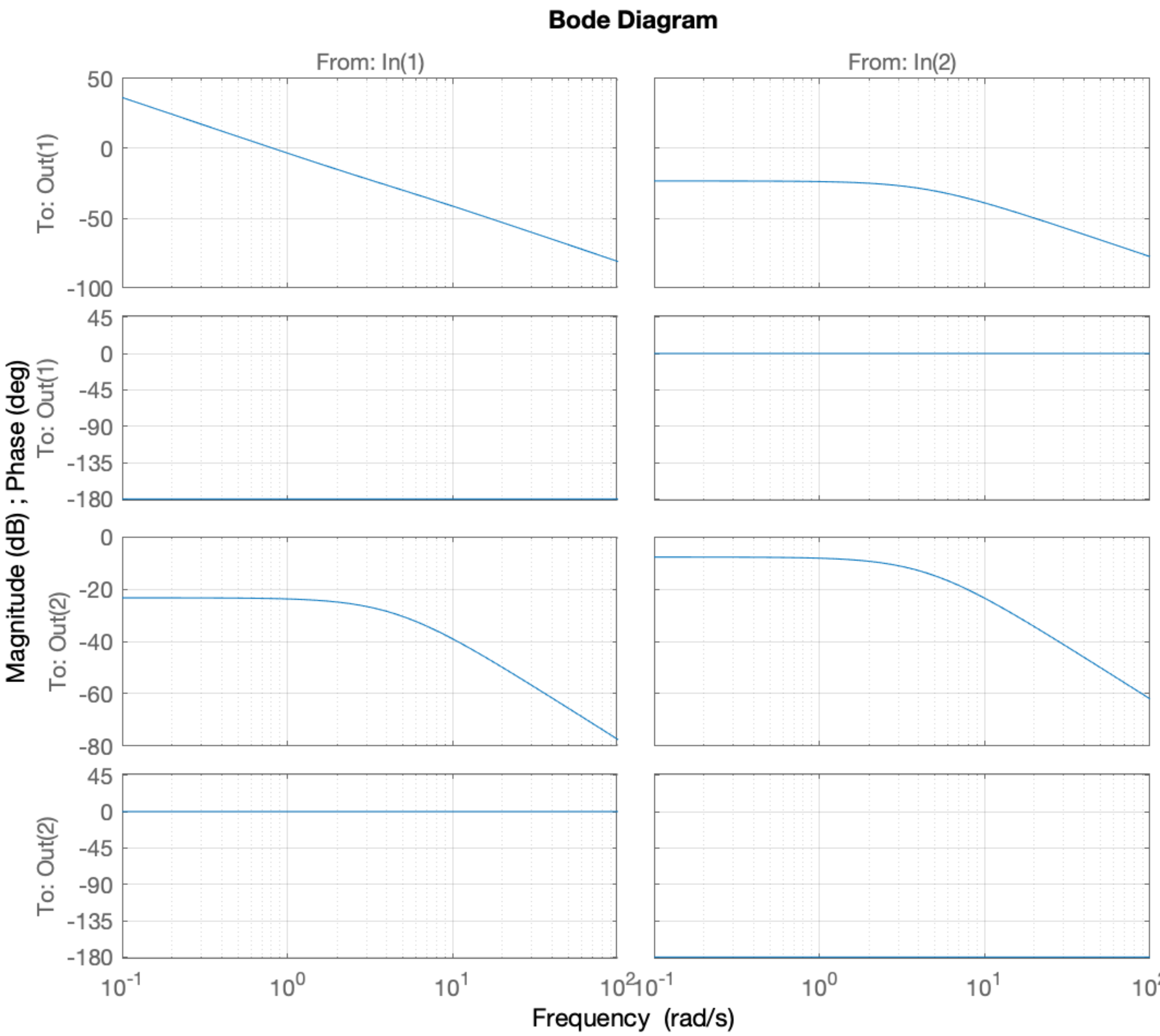


Figure 1-1. Bode Plot of $P_3(s)$

Hence, we can control the base position using the force input and control the rod angle using the torque input. Consequently, we can apply decentralized design strategy to this particular MIMO system.

1.2 (b) Derivation of $P_{3,\text{approx}}(s)$

Recall the actual $P_3(s)$, we have previously found there is little effect on the steady-state from the cross-channel. We may deal the system as a decentralized system.

$$P_3(s) = \begin{bmatrix} -\frac{0.8889\,(s+3.834)\,(s-3.834)}{s^2\,(s-4.427)\,(s+4.427)} & \frac{-1.3333\,s^2}{s^2\,(s-4.427)\,(s+4.427)} \\ -\frac{-1.3333}{(s-4.427)\,(s+4.427)} & \frac{8.0}{(s-4.427)\,(s+4.427)} \end{bmatrix} \tag{1}$$

However, instead of blindly ignoring the off-diagonal terms within $P_3(s)$, we can design an approximated plant by isolating the system:

Firstly, from the perspective of the force input, we can treat the entire system as whole:

$$F = (m+M)\ddot{w} \tag{2}$$

$$\textbf{Laplace Tranform + T.S. Expansion} \Rightarrow F(s) = (m+M)(s^2W(s) + s\cancelto{0}{\dot{w}(0)} + \cancelto{0}{w(0)}) \tag{3}$$

$$\Rightarrow P_{3,\text{approx},11} = \frac{W(s)}{F(s)} = \frac{1}{(m+M)\,s^2} \tag{4}$$

Lastly, from the perspective of the torque input, we can treat the position base is fixed, and only compute the effect of the rod:

$$\tau + mg\frac{L}{2}\sin\theta = J\ddot{\theta} \tag{5}$$

$$\textbf{Laplace Tranform + T.S. Expansion} \Rightarrow T(s) = \frac{mL^2}{3}\,(s^2\Theta(s)) - mg\frac{L}{2}\Theta(s) \tag{6}$$

$$\Rightarrow P_{3,\text{approx},22} = \frac{\Theta(s)}{T(s)} = \frac{6}{2mL^2\,s^2 - 3mgL} \tag{7}$$

As a result, we can form the approximated plant $P_{3,\text{approx}}(s)$ as:

$$P_{3,\text{approx}}(s) = \begin{bmatrix} \frac{1}{(m+M)\,s^2} & 0 \\ 0 & \frac{6}{2mL^2\,s^2-3mgL} \end{bmatrix} = \begin{bmatrix} \frac{0.66667}{s^2} & 0 \\ 0 & \frac{6.0}{(s-3.834)(s+3.834)} \end{bmatrix} \tag{8}$$

To further ensure the integrity of the approximated plant, we may plot the bode plot of the plant as well. As we may see from Figure 1-2 below, the diagonal channels indeed appear to be similar to the original bode plot in Figure 1-1.

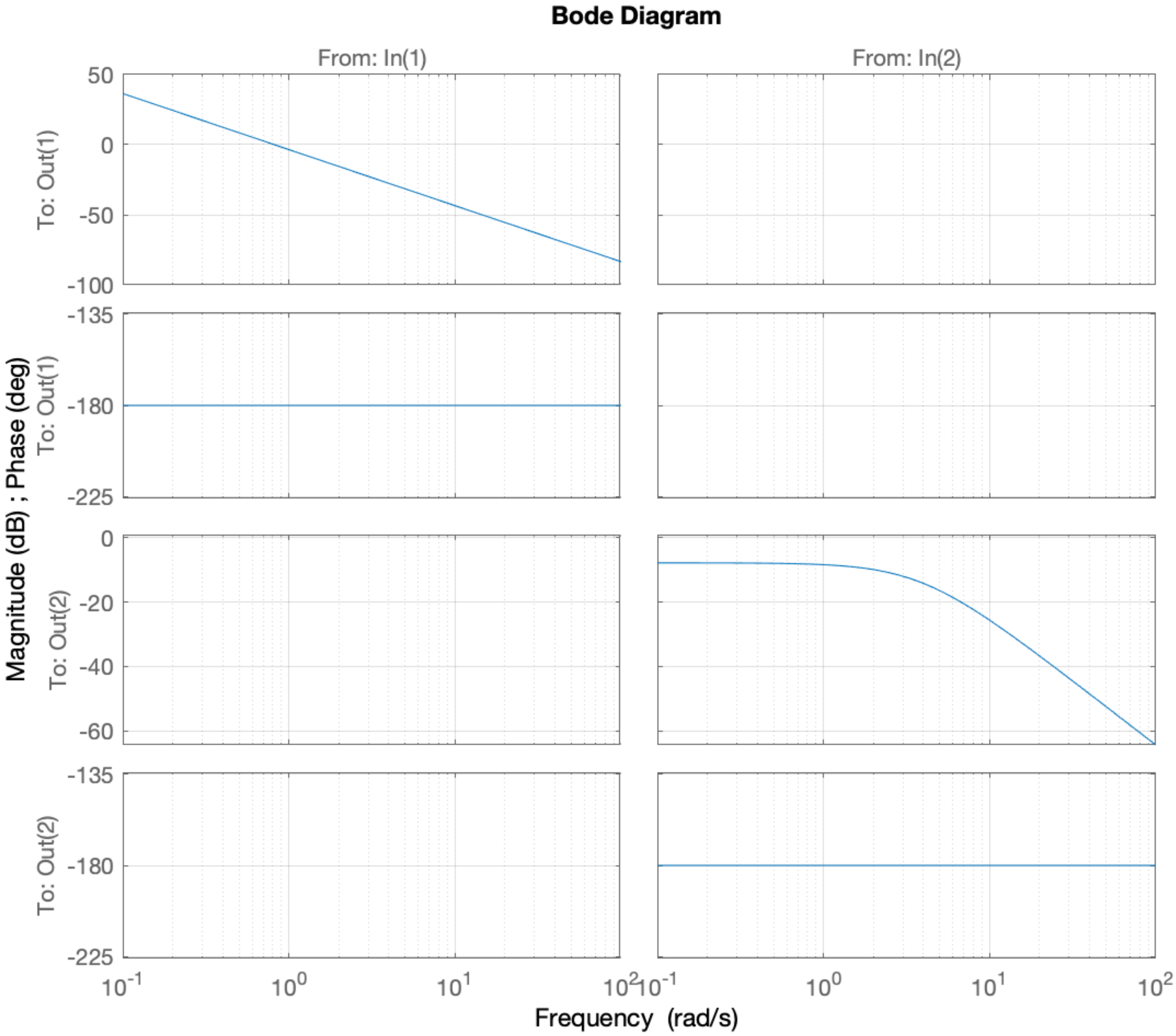


Figure 1-2. Bode Plot of $P_{3,\text{approx}}(s)$

1.3 (c) Design 1-DOF SISO Controller $C_{11}(s)$ for the (1,1) entry of $P_{3,\mathbf{approx}}(s)$

Three design specifications are imposed:

I Closed-loop stable
II Perfect steady-state tracking
III PM $\geq 50^{\circ}$

As we may observe from $(1,1)$ entry in the bode plot from Figure 1-2, the phase margin is 0°. As a result, we need a lead filter to pull up the phase margin to satisfy Spec III. Since, we have double integrator in the $P_{3,\text{approx},11}(s)$ plant, it naturally satisfy the perfect steady-state tracking Spec II. As a result, the closed loop system should be stable.

Using the `'sisotool'`, we may manually find the suitable controller quickly, as seen in Figure 1-3.

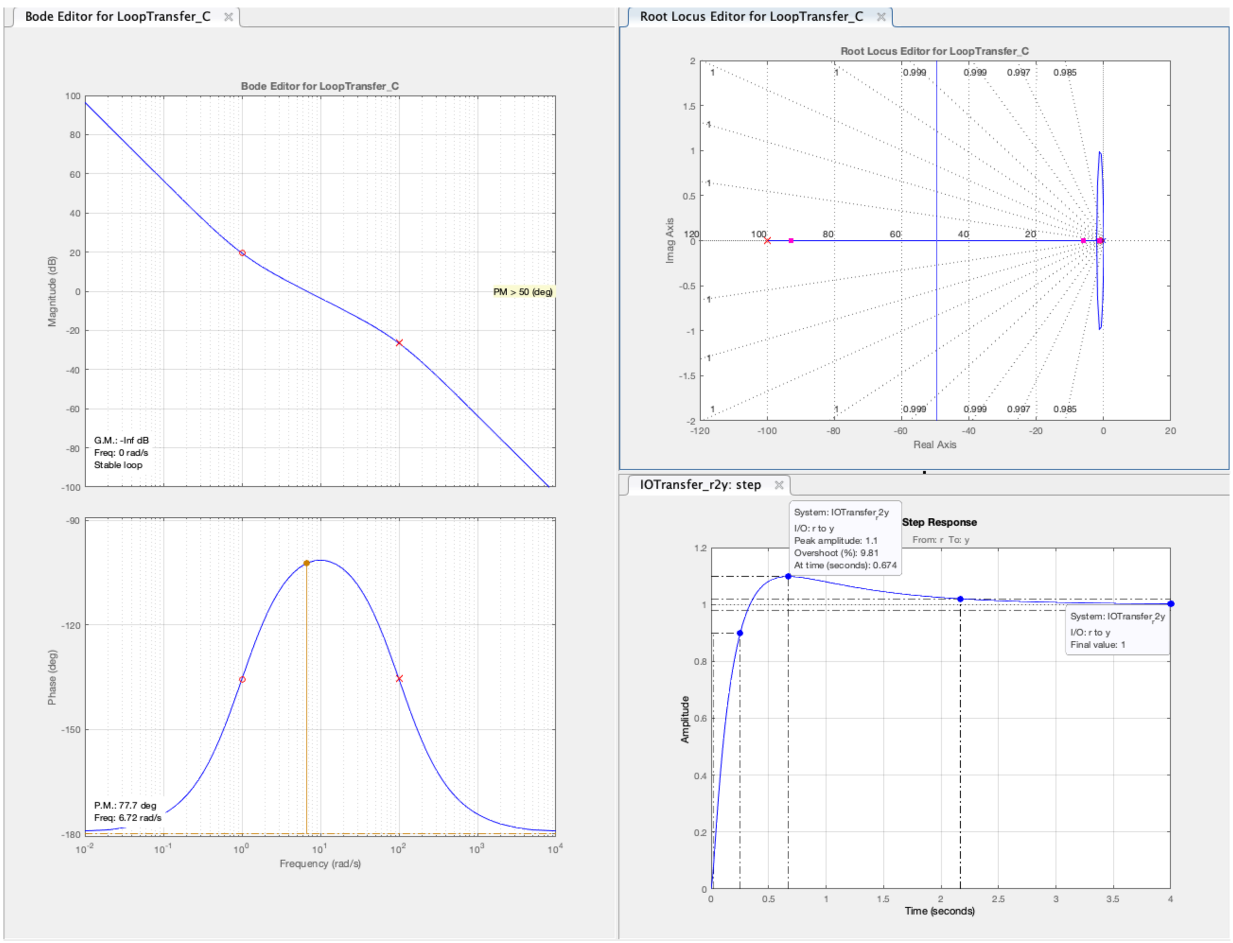


Figure 1-3. SISO design tool of closed-loop controller design for $P_{3,\text{approx},11}(s)$

As a result, the SISO controller we designed for $P_{3,\text{approx},11}(s)$ is:

$$C_{11}(s) = \frac{1000\,(s+1)}{(s+100)} \tag{9}$$

We may also plot the final plot of bode diagram, root locus, and the step response similar to the `'sisotool'`:

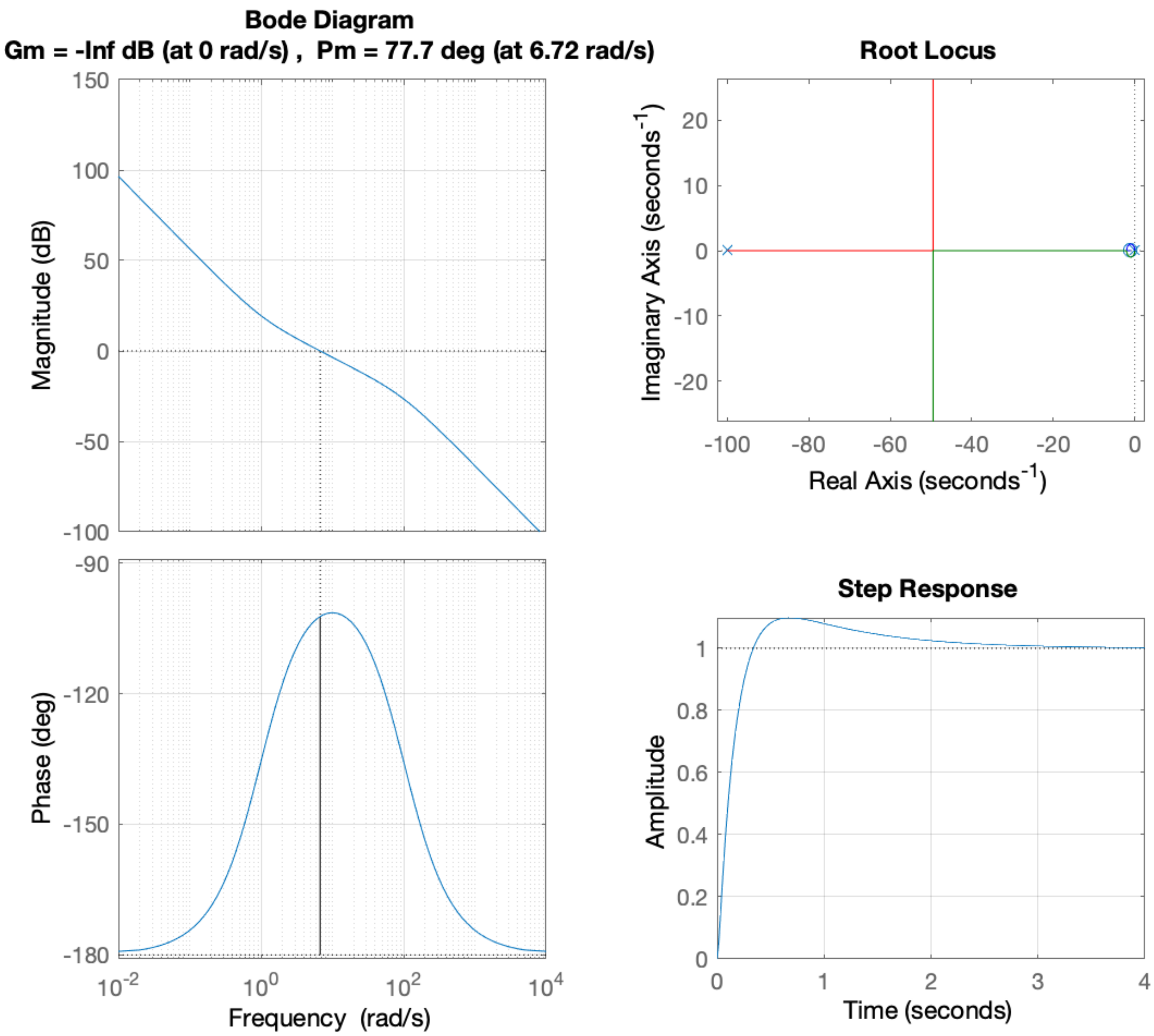


Figure 1-4. Final SISO plot of closed-loop controller design for $P_{3,\text{approx},11}(s)$

1.4 (d) Design 1-DOF SISO Controller $C_{22}(s)$ for the (2,2) entry of $P_{3,\mathbf{approx}}(s)$

Similarly, as we may observe from $(2,2)$ entry in the bode plot from Figure 1-2, the phase margin is $0°$. As a result, we need a lead filter to pull up the phase margin to satisfy Spec III. In this case, we do not have any integrator in the $P_{3,\text{approx},22}(s)$ plant, hence, there is a need to include an integrator inside the controller to achieve the perfect steady-state tracking Spec II.

To note, there exists a RHP pole in the plant. In order to make the closed loop system stable (Spec I), we need to introduce a LHP zero between the stable and unstable pole to pull the closed-loop pole to the LHP.

Using the `'sisotool'`, we may manually find the suitable controller quickly, as seen in Figure 1-5. Firstly, adding an integrator in the controller, and introduce a LHP zero at $s = -3$ and adjust the gain to make the closed-loop poles back into the OLHP. Lastly, adding the lead filter to make the phase margin above the Spec III.

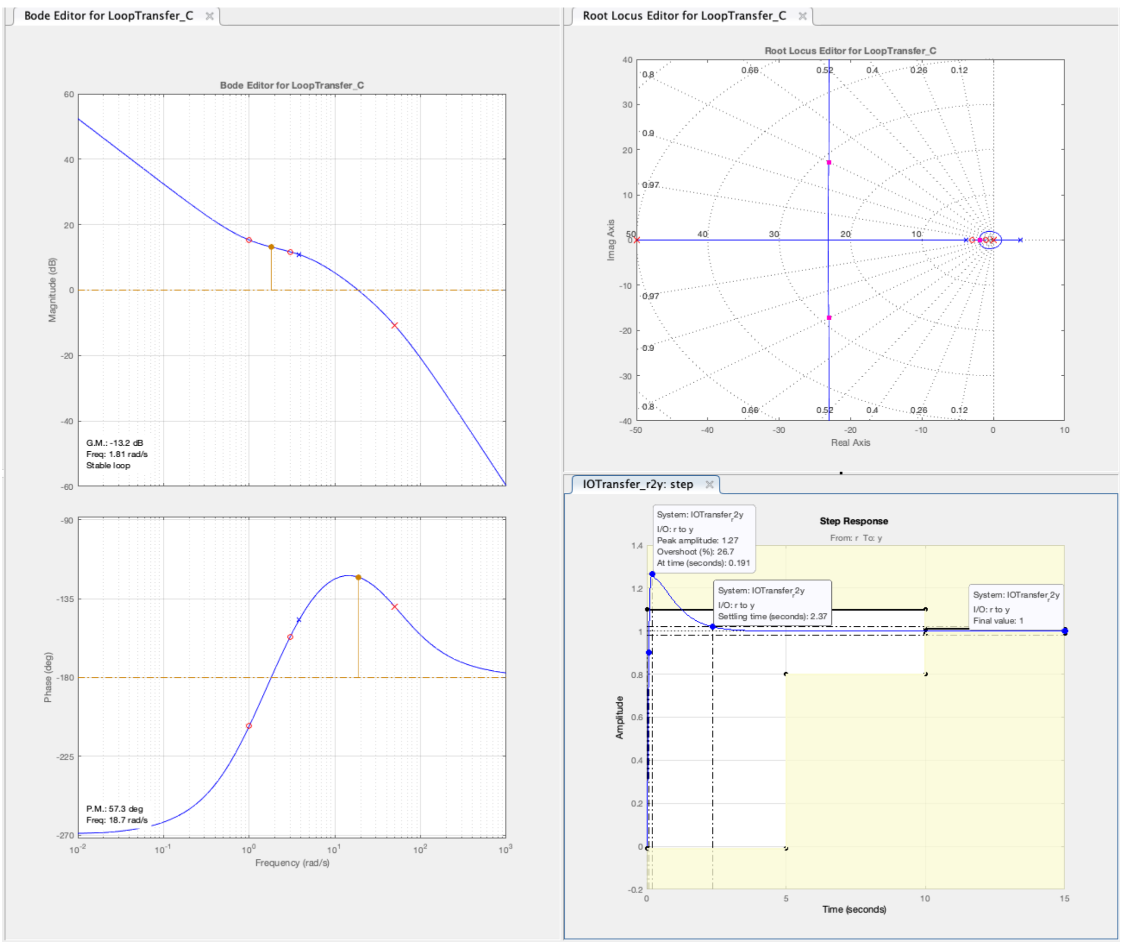


Figure 1-5. SISO design tool of closed-loop controller design for $P_{3,\text{approx},22}(s)$

As a result, the SISO controller we designed for $P_{3,\text{approx},22}(s)$ is:

$$C_{11}(s) = \frac{170.54\,(s+1)(s+3)}{(s+50)s} \tag{10}$$

We may also plot the final plot of bode diagram, root locus, and the step response similar to the `'sisotool'`:

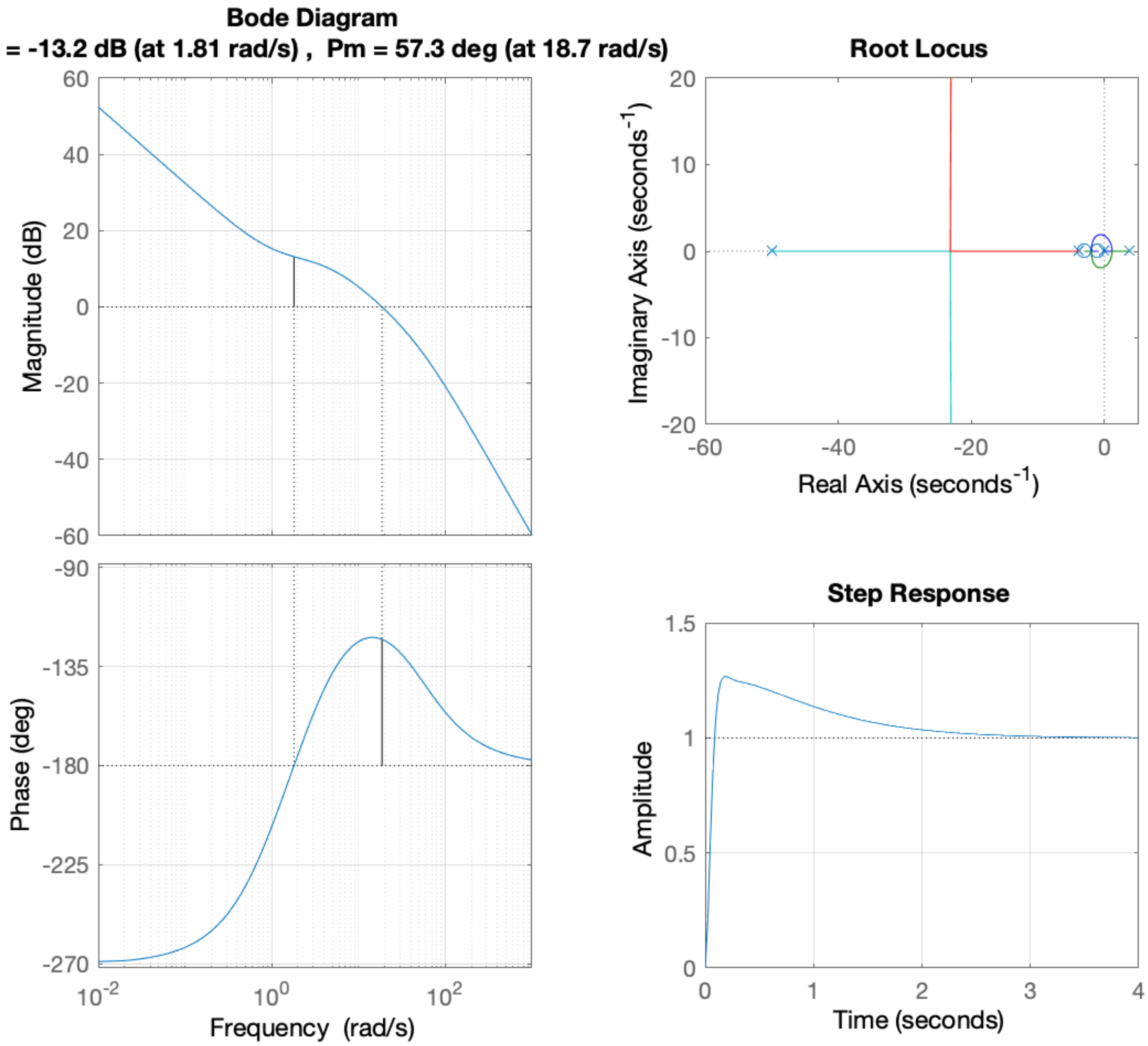


Figure 1-6. Final SISO plot of closed-loop controller design for $P_{3,\mathrm{approx},22}(s)$

1.5 (e) Analyze the designed controller $C_3(s)$

From Section 1.3 and Section 1.4, we may obtain the final controller designed for the approximated plant $P_{3,\text{approx}}$:

$$C_3(s) = \begin{bmatrix} \frac{1000(s+1)}{(s+100)} & 0 \\ 0 & \frac{170.54(s+1)(s+3)}{(s+50)s} \end{bmatrix} \tag{11}$$

Let's apply this final controller with the approximated plant $P_{3,\text{approx}}(s)$ it designed from. We may obtain the step response graph as shown in Figure 1-7 below and both transient and steady-state performance as shown in Table 1-1 below.

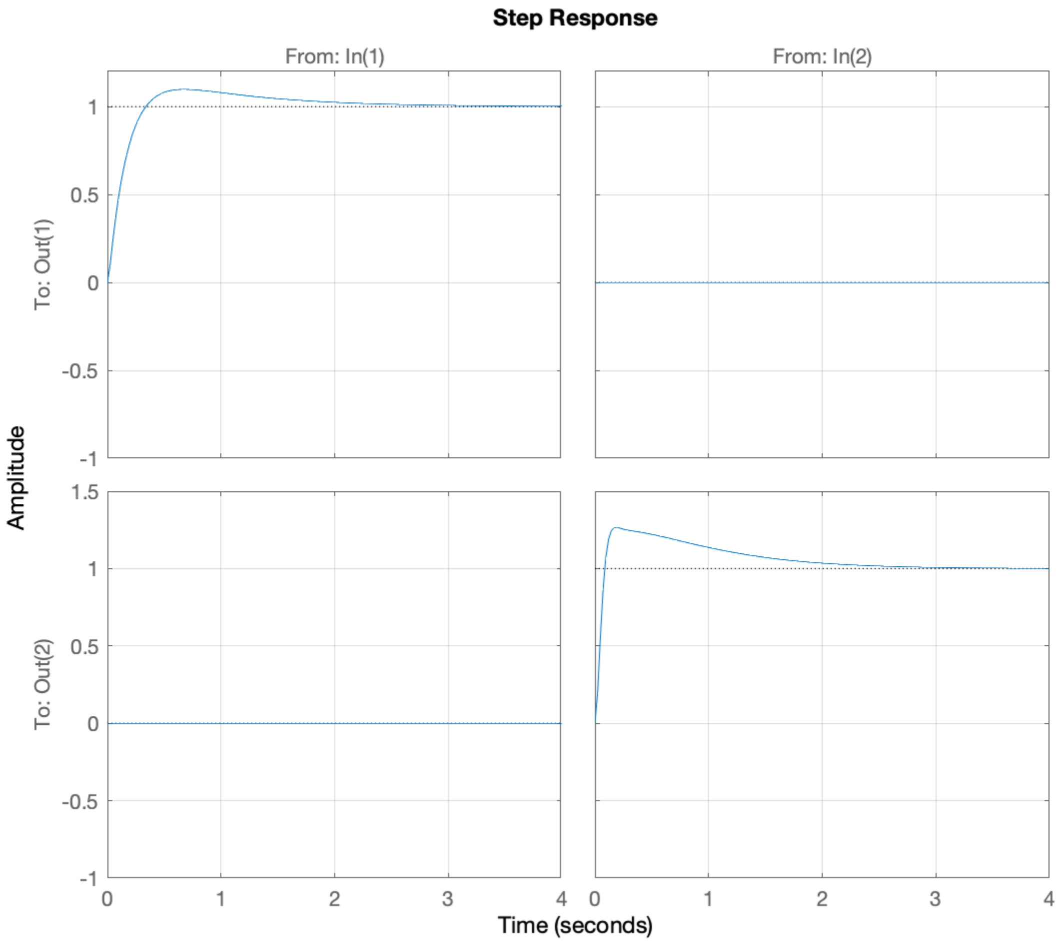


Figure 1-7. Ideal step response with the approximated plant $P_{3,\text{approx}}(s)$

Table 1-1. Ideal Performance Summary ($C_3(s)$ with $P_{3,\text{approx}}(s)$)

	t_{rise}	$t_{settling}$	t_{peak}	y_{peak}	y_{ss}	OS%	US%
(1,1)	0.2298	2.1687	0.67103	1.0981	1.0037	9.8058	0
(1,2)	0	0	0	0	0	∞	0
(2,1)	0	0	0	0	0	∞	0
(2,2)	0.064222	2.3662	0.19172	1.2666	1.003	26.6601	0

Similarly, let's apply this final controller with the actual plant $P_3(s)$ it designed from. We may obtain the step response graph as shown in Figure 1-8 below and both transient and steady-state performance as shown in Table 1-2 below.

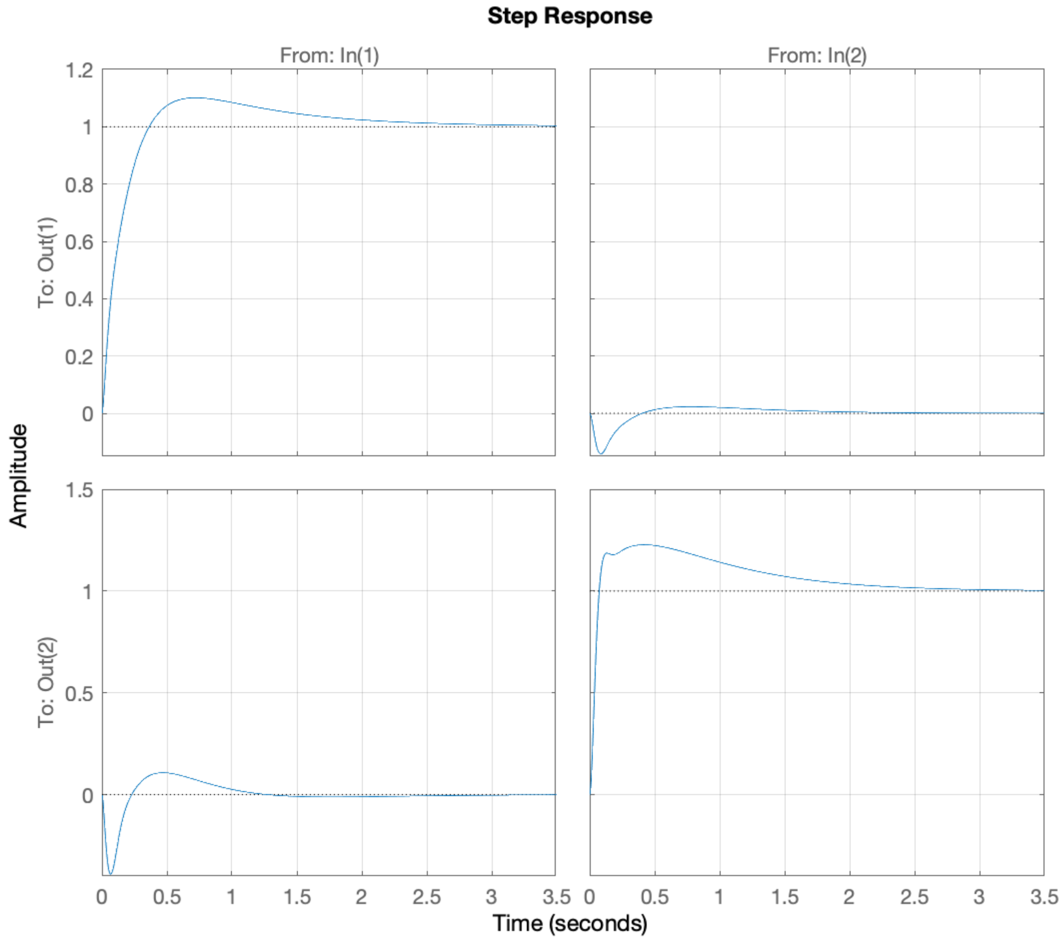


Figure 1-8. Actual step response with the actual plant $P_3(s)$

Table 1-2. Actual Performance Summary ($C_3(s)$ with $P_3(s)$)

	t_{rise}	$t_{settling}$	t_{peak}	y_{peak}	y_{ss}	$OS\%$	$US\%$
(1,1)	0.2506	2.143	0.71721	1.1012	1.0044	10.1206	0
(1,2)	2.1704e-13	2.2418	0.085915	0.14246	0.00015909	131127080892444.7	21200873797753.34
(2,1)	3.933e-13	2.305	0.067238	0.39121	-0.0014772	36100623416931.88	9911777297481.107
(2,2)	0.049924	2.3456	0.41837	1.2272	1.0038	22.7219	0

1.5.1 Assuming the SISO loops from parts (c) and (d) are stable, do you necessarily expect the MIMO closed-loop system to be stable?

As per discussion when justifying the decentralization strategy in Section 1.1, we can neglect the coupling effect from the cross-channels. As Figure 1-2 suggested, the approximation plant exhibits similar bode characteristics as the original plant. As a result, as long as we give a reasonable GM and PM when designing the controller for the approximated plant $P_{3,\text{approx}}(s)$, we do expect the MIMO likely to be closed-loop stable. But, there is no guarantee that the MIMO closed-loop system in the end to be stable, since the controller was designed neglecting the coupling effects, and a large gain in the controller could possibly excite the coupling term and make the system unstable.

In our controller designed above, the closed-loop system appear to be stable for both the SISO (decentralized MIMO) and actual MIMO system with a unit step input, as both ideal response in Figure 1-7 and actual response in Figure 1-8 shown above.

1.5.2 Assuming the MIMO closed-loop system is stable, is there a guarantee of perfect steady-state tracking for step references?

Assuming the MIMO closed-loop system is stable, there is no guarantee of 100% perfect steady-state tracking. Depending on the coupling term, the coupling term may have some effect on the final steady-state.

In our particular scenario, the actual steady-state performance is close to perfect (as stated in Table 1-2): 1.0044 and 1.0038 for diagonal terms, and 0.00016 and -0.0015 for the cross-channel terms. The actual steady-state performance is extremely close to the expectation in Table 1-1. As a result, the controller we designed is a great controller in terms of steady-state performance with about $< 2.5\,[\mathrm{s}]$ settling time.

1.5.3 Assuming the MIMO closed-loop system is stable, is there a guarantee that the step response transients will be identical to the step response transients of the individual SISO feedback loops from parts (c) and (d)?

There is no guarantee that the step response transients would be identical to the step response transients of the individual SISO feedback loops. Since the coupling term would bring some transient impacts to the system, unless there is no coupling effects in the original plant in the first place.

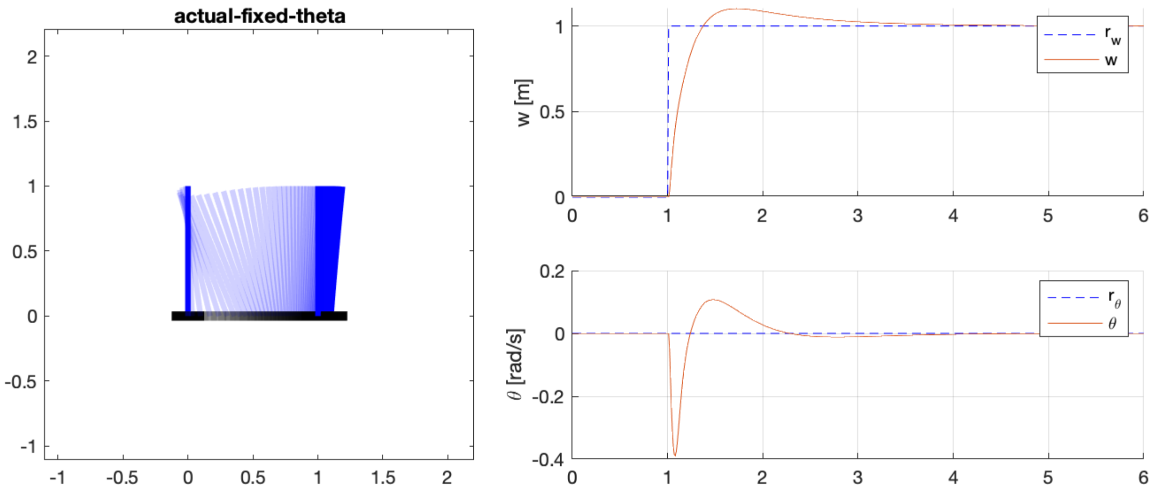
As we may see from the transient performance tabulated in Table 1-2 and Table 1-1, and the step response graph in Figure 1-8 and Figure 1-7 for the actual system and approximated system respectively, we can see there are quite difference in terms of transient performance (including rising time, settling time, peak time, peak, overshoot percentage and undershoot percentage. From the step response graph, we can see undershoot and overshoot effects in transient for coupling channels, which was expected to have no or little effect originally. In addition, there is a wobbling effect in the $(2, 2)$ entry of the step response. Hence, we can conclude the transient performance would not be identical to the individual SISO feedback, as long as there exists cross-channeling effects originally.

(We will see and further discuss about these effects visually in the simulation section (Section 1.6) below).

1.6 (f) [Optional] Simulation

1.6.0.1 Step Input of $r_w = 1\,[\mathrm{m}]$

Firstly, let's fixed the rod angle, and try to move the base without changing the rod angle.



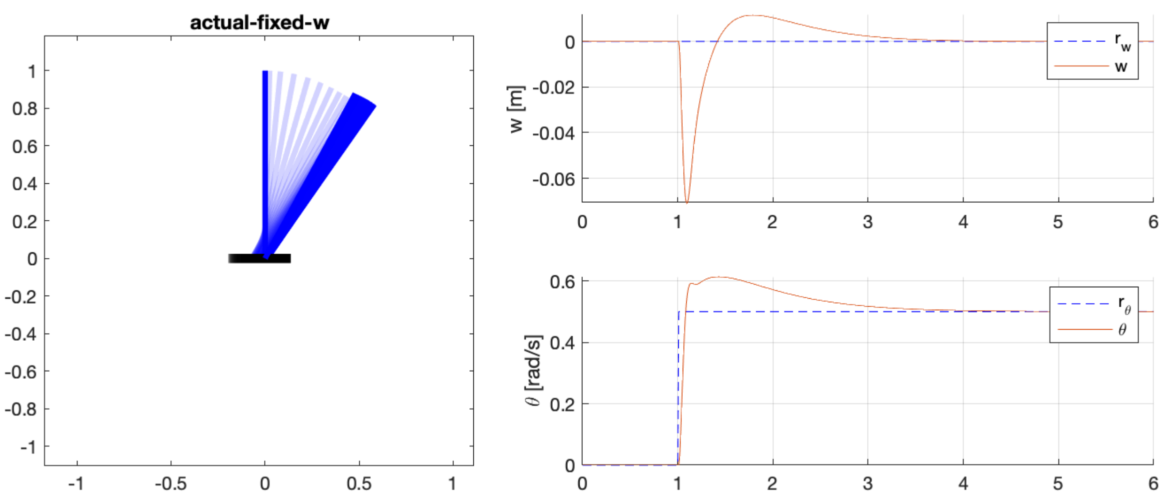
(a) Simulation (b) Step Response

Figure 1-9. Simulation and Step Response for step input of $r_w = 1\,[\mathrm{m}]$

As we have discussed, we observed the undershoot and overshoot in the actual plant due to the ignorance of cross-channelling effects when designing the controller. This causes a bad transient performance, but the steady-state of the system looks pretty decent. The coupling effect from the base position to the rod angular position is quite significant, and observable in the simulation plot in Figure 1-9.

1.6.0.2 Step Input of $r_\theta = 0.5\,[\mathrm{rad}]$

Now, let's fixed the base position, and try to rotate the rod angle without changing the base position.



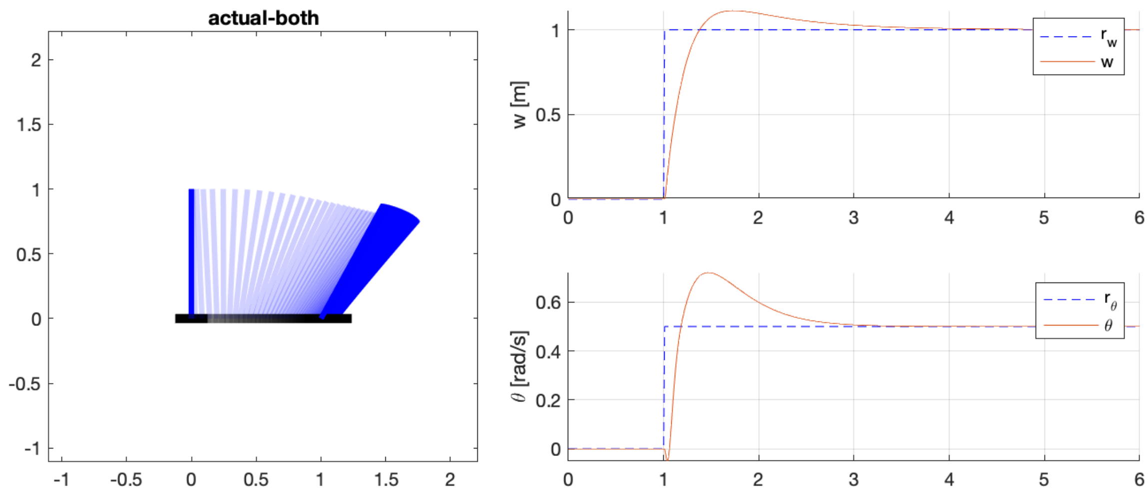
(a) Simulation (b) Step Response

Figure 1-10. Simulation and Step Response for step input of $r_\theta = 0.5\,[\mathrm{rad}]$

Similarly, we observed the undershoot and overshoot in the actual plant due to the ignorance of cross-channelling effects when designing the controller. The system indeed reaches a perfect steady-state, despite bad transient performance. To note, this transient effect is almost negligible, hence, we may conclude that the rod angle position has little coupling effect of the base position in this setup (as suggest by Figure 1-10).

1.6.0.3 Combined Step Input

Finally, let's combine both step input.



(a) Simulation (b) Step Response

Figure 1-11. Simulation and Step Response for step input of $r_\theta = 0.5\,[\text{rad}]$ and $r_w = 1\,[\text{m}]$

The result looks pretty good, except the undershoot in the rod angle from the coupling effects from the base motion. The final steady-state for both final position and final rod angle looks decent.

2 Problem P6: Use of state-space methods to control the MIMO aiming system

2.1 (a) MIMO one-rod aiming system

2.1.1 (i) Transfer Function

The transfer function from "`zpk`" is truely the same as given in Equation (1) in Problem P5.

$$P_3(s) = \begin{bmatrix} -\frac{0.8889\,(s+3.834)\,(s-3.834)}{s^2\,(s-4.427)\,(s+4.427)} & \frac{-1.3333\,s^2}{s^2\,(s-4.427)\,(s+4.427)} \\ -\frac{-1.3333}{(s-4.427)\,(s+4.427)} & \frac{8.0}{(s-4.427)\,(s+4.427)} \end{bmatrix} \tag{12}$$

2.1.2 (ii) Plain State-Feedback Control with Integral Action

2.1.2.1 Augmented System is Controllable

From the matlab, we can find the rank of the controllability matrix is 6 (full rank), hence, the augmented system is controllable.

2.1.2.2 Closed-loop Step Response

Poles are placed about $s = -20$, specifically: $\{-19.9900, -19.9800, -19.9700, -20.0100, -20.0200, -20.0300\}$.

The closed-loop step response is plotted as shown in Figure 2-1 below.

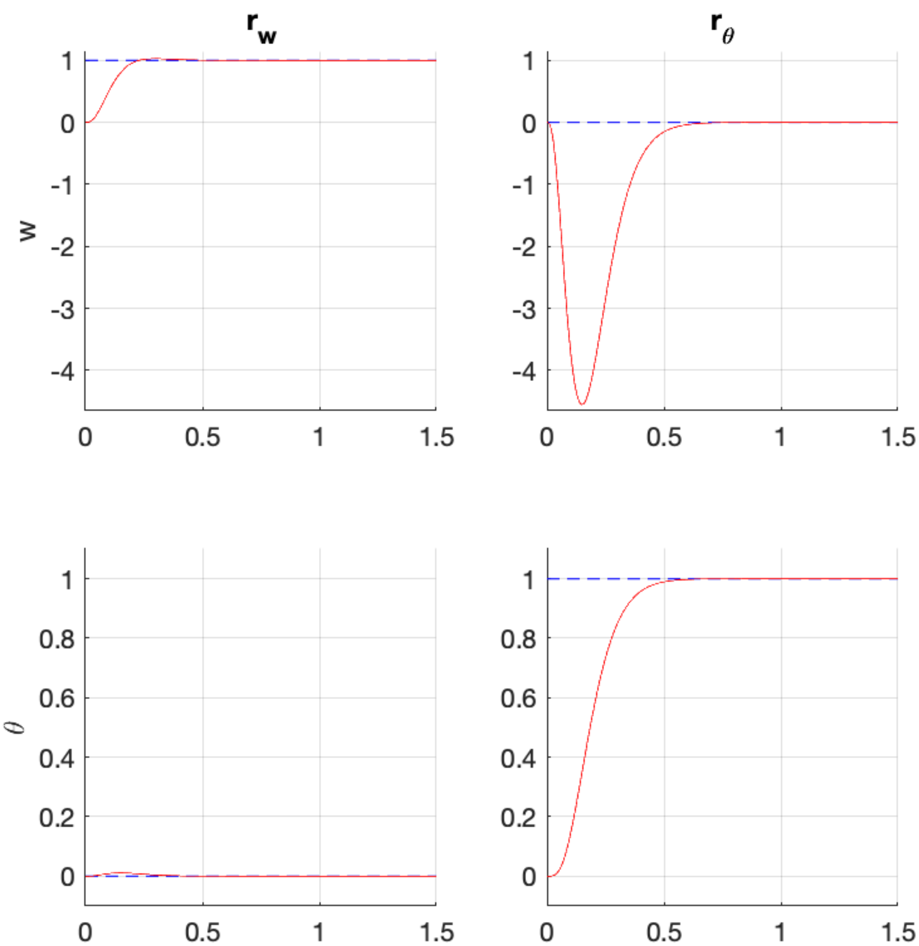
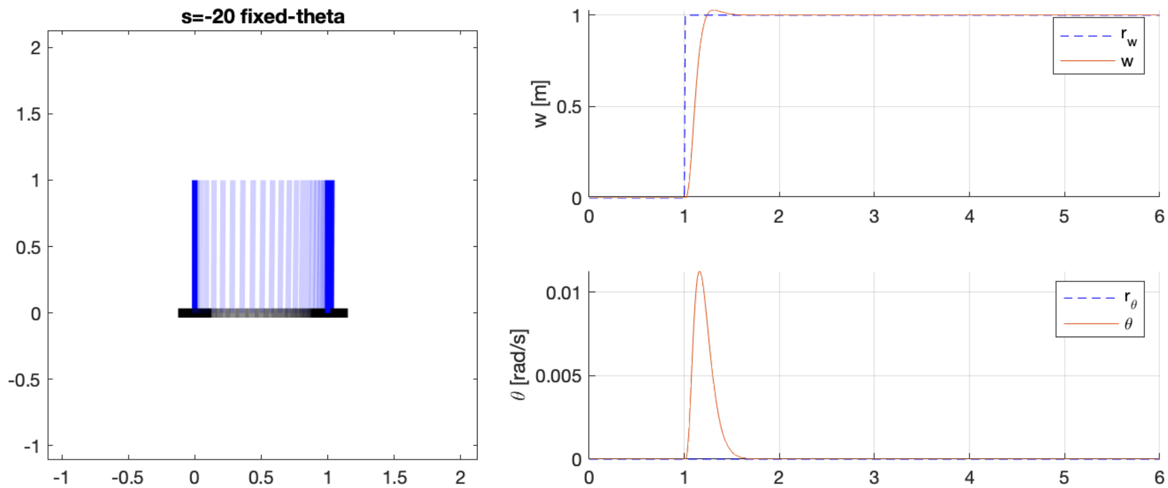


Figure 2-1. Closed Loop Step Response

2.1.2.3 [Optional] Simulation:

As we may see in the simulation result below, the base control is quite perfect and behaves much better than the decentralized design back in Section 1.6. However, the rod angle control suffers a sever undershoot to the base position in terms of transient performance, which is significantly worse than the decentralized design in Section 1.6. The system is much responsive at the cost of the significant coupling effects when controlling the θ angular position.

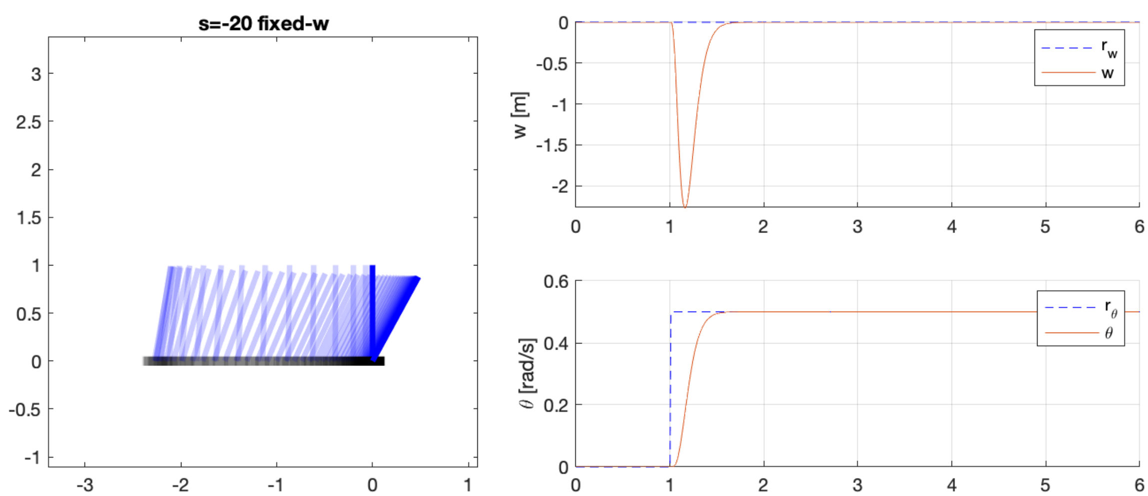
- Step Input of $r_w = 1\,[\mathrm{m}]$:



(a) Simulation (b) Step Response

Figure 2-2. Simulation and Step Response for step input of $r_w = 1$

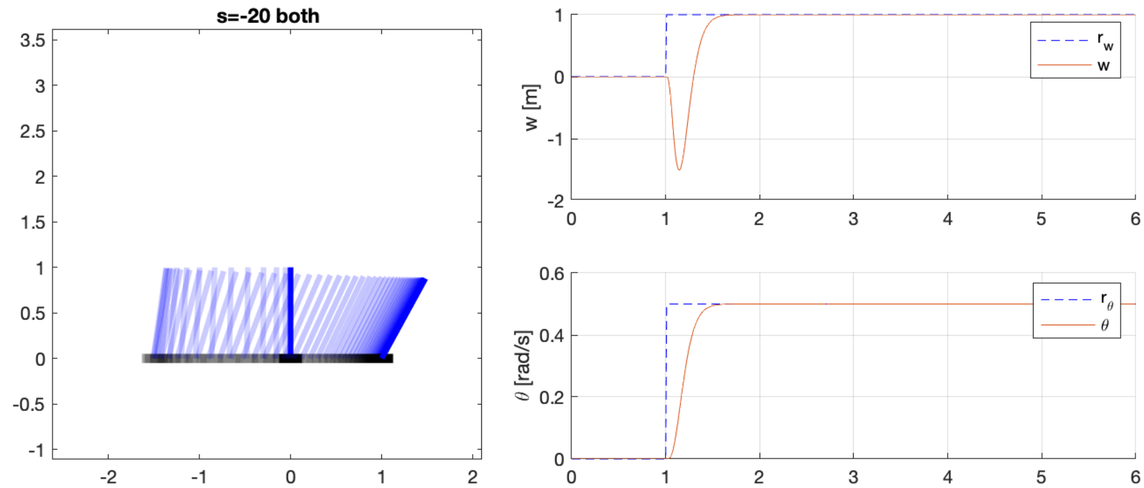
- Step Input of $r_\theta = 0.5\,[\mathrm{rad}]$:



(a) Simulation (b) Step Response

Figure 2-3. Simulation and Step Response for step input of $r_\theta = 0.5$

- Combined Step Input:



(a) Simulation (b) Step Response

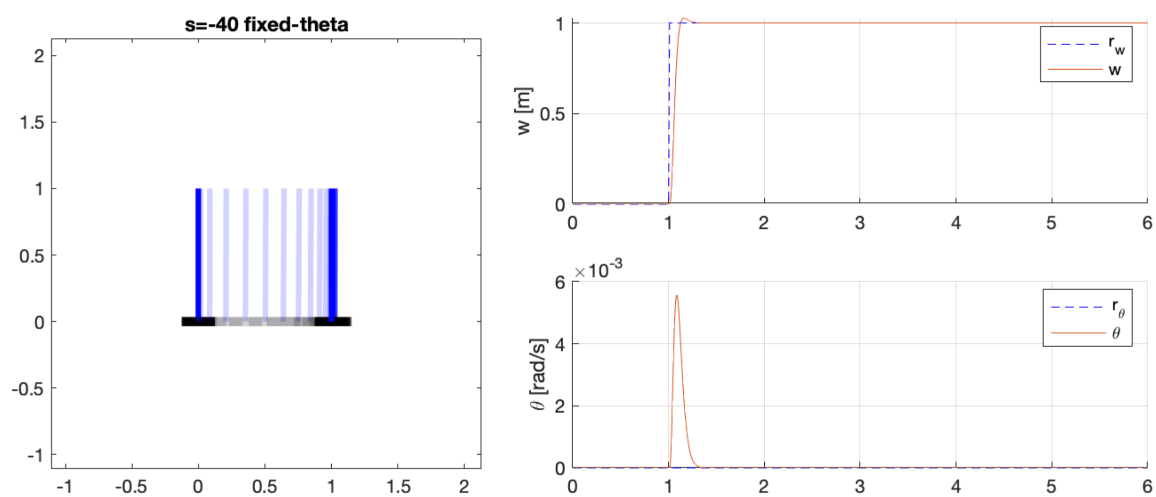
Figure 2-4. Simulation and Step Response for step input of $r_w = 1$ and $r_\theta = 0.5$

2.1.2.4 [Optional] Place the poles further from the origin ($s = -40$):

As we increase the pole further into LHP, the system response speeds up, and it also result a much significant undershoot of the base position when controlling the angle of the joint as shown in Figure 2-6.

As we increase further r, the system is on the verge of instability.

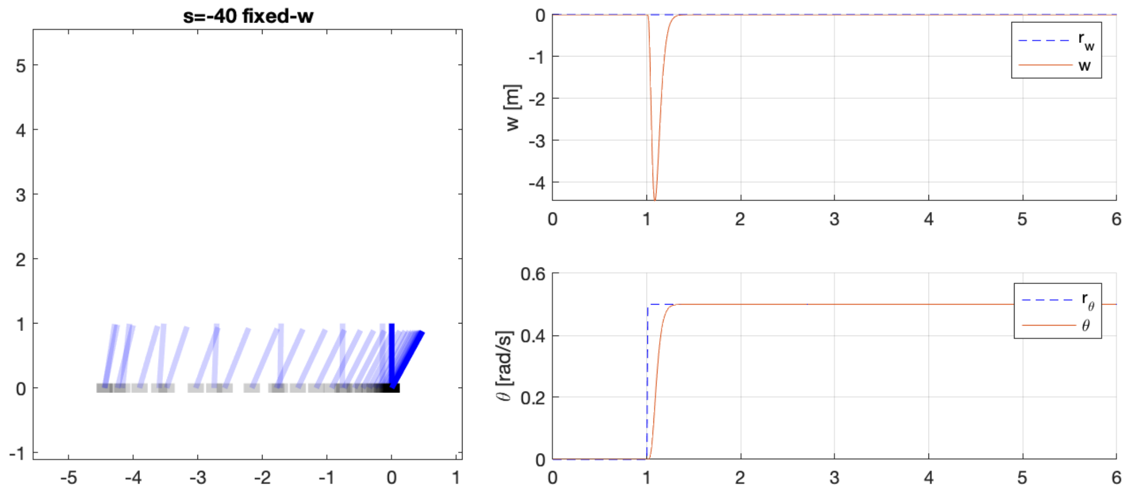
- Step Input of $r_w = 1\,[\mathrm{m}]$:



(a) Simulation (b) Step Response

Figure 2-5. Simulation and Step Response for step input of $r_w = 1$

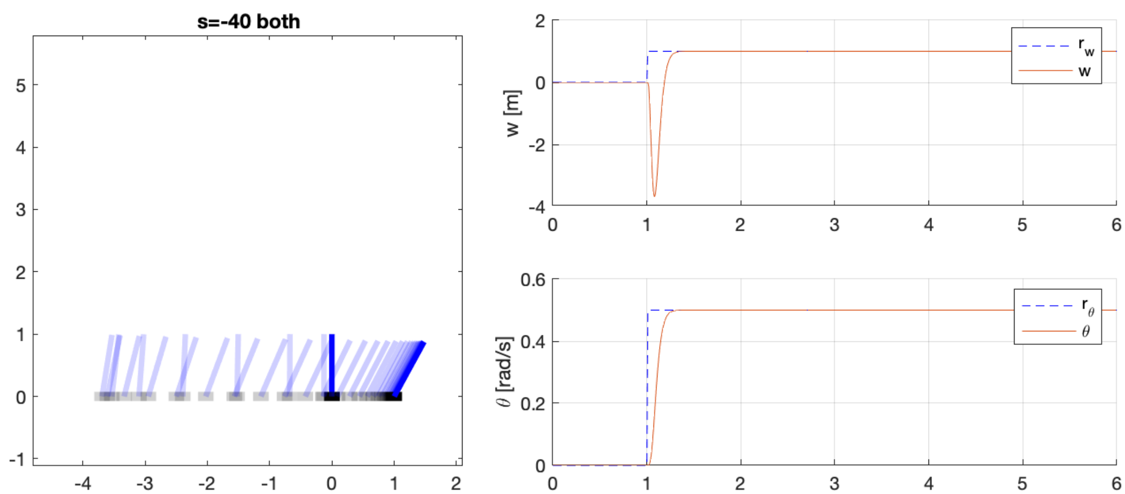
- Step Input of $r_\theta = 0.5\,[\mathrm{rad}]$:



(a) Simulation (b) Step Response

Figure 2-6. Simulation and Step Response for step input of $r_\theta = 0.5$

- Combined Step Input:



(a) Simulation (b) Step Response

Figure 2-7. Simulation and Step Response for step input of $r_w = 1$ and $r_\theta = 0.5$

2.2 (b) MIMO two-rod aiming system

2.2.1 (i) Transfer Function

From matlbab, we may find the transfer function for two-rod system:

$$P_6(s) = \begin{bmatrix} \frac{0.9333(s+10.16)(s+3.788)(s-10.16)(s-3.788)}{s^2(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{-2.4(s+7.668)(s-7.668)(s^2+1.199e-14)}{s^2(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{0.8(s^2+7.27e-15)(s^2+176.4)}{s^2(s+10.52)(s+4.331)(s-10.52)(s-4.331)} \\ \frac{-2.4(s-7.668)(s+7.668)}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{33.6(s-5.797)(s+5.797)}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{-43.2(s-1.032e-07)(s+1.032e-07)}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} \\ \frac{0.8(s^2+176.4)}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{-43.2s^2}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} & \frac{110.4(s-5.539)(s+5.539)}{(s+10.52)(s+4.331)(s-10.52)(s-4.331)} \end{bmatrix} \tag{13}$$

2.2.2 (ii) Plain State-Feedback Control with Integral Action

2.2.2.1 Augmented System is Controllable

From matlab, we can find the controllability rank is 9, hence it is full rank and the system is controllable.

2.2.2.2 Closed-loop Step Response

Poles are placed about $s = -20$, and the closed-loop step response is plotted as shown in Figure 2-8 below.

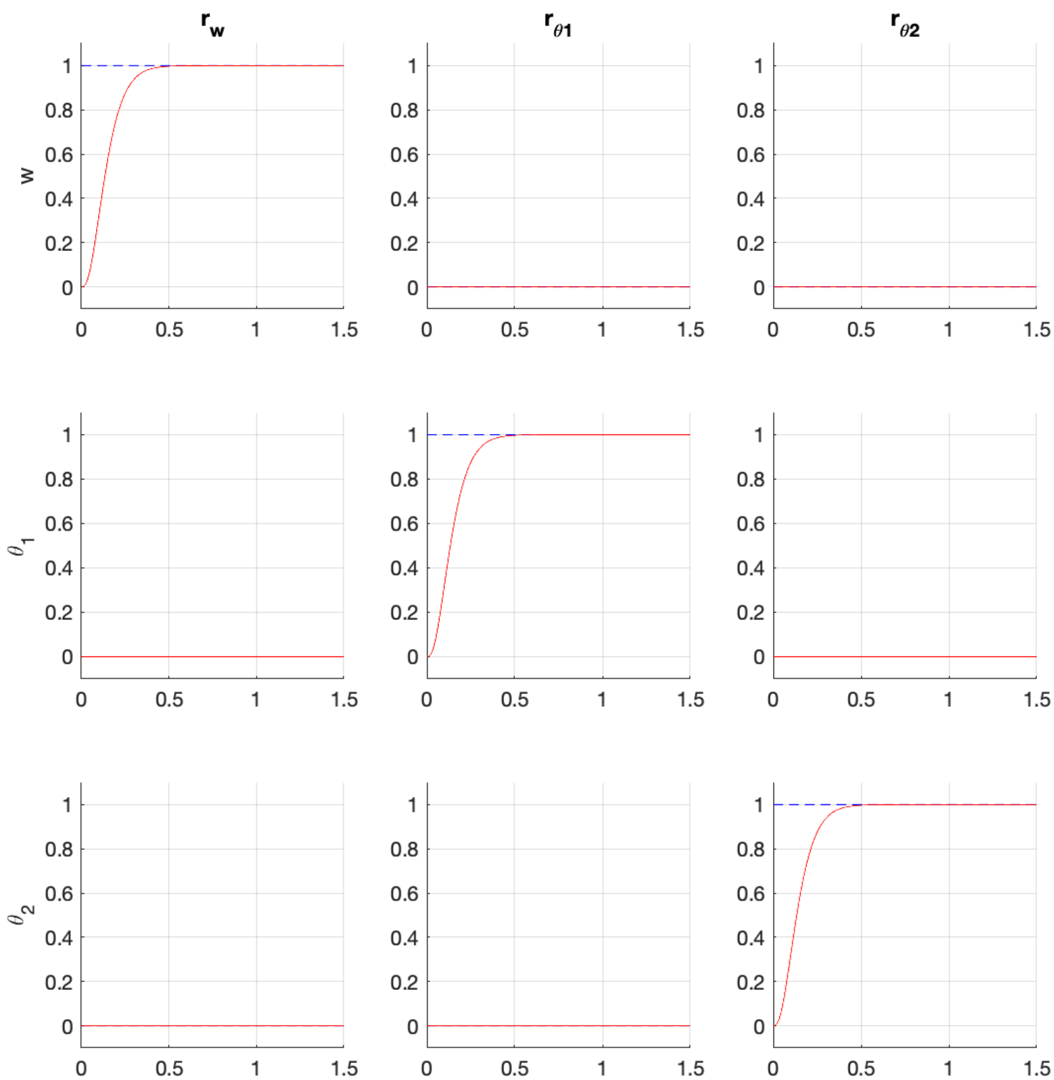


Figure 2-8. Closed Loop Step Response

2.2.3 (iii) Observer-based State-Feedback Control with Integral Action

2.2.3.1 Observability

From matlab, we can find the observability rank is 6, hence the state-space realization is fully observable. After placing poles at $s = -20$ and poles of observer error at $s = -4$, we can obtain the closed-loop step response in red color in Figure 2-9 below. If we change the initial error values of 0.5 in `'lsim'`, we can see the ''locking on" effects as shown in orange color in Figure 2-9. We may see the observer-based state-feedback control is quite powerful, and it is capable to reach state equilibrium quickly.

2.2.3.2 Closed-loop step response

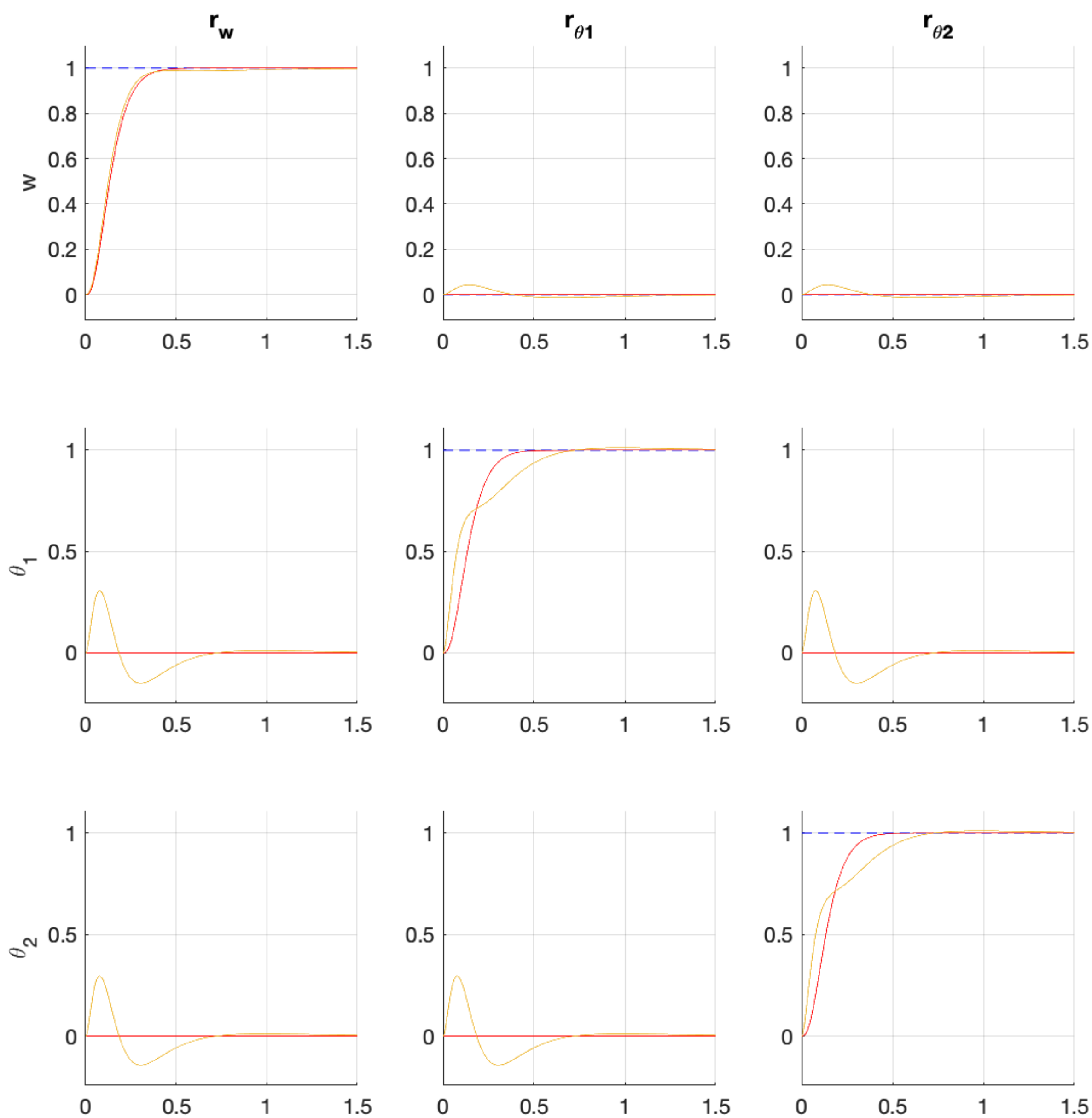
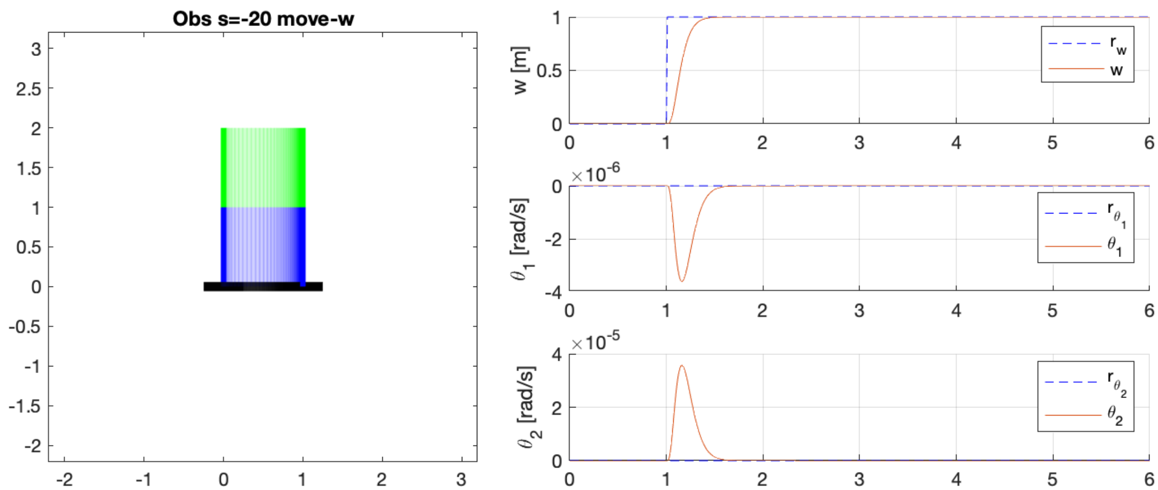


Figure 2-9. Closed Loop Step Response (Observer-based)

2.2.3.3 [Optional] Simulation:

As we may expected and seen from the simulation and step response below, the response is nearly perfect, and no significant overshoot nor cross-channeling disturbance observed.

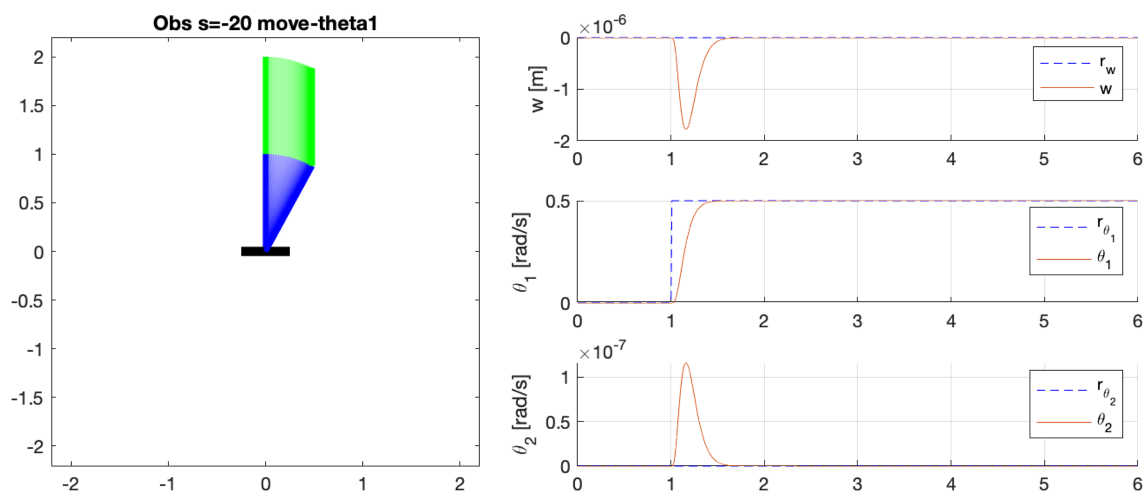
- Step Input of $r_w = 1\,[\mathrm{m}]$**:**



(a) Simulation (b) Step Response

Figure 2-10. Simulation and Step Response for step input of $r_w = 1$

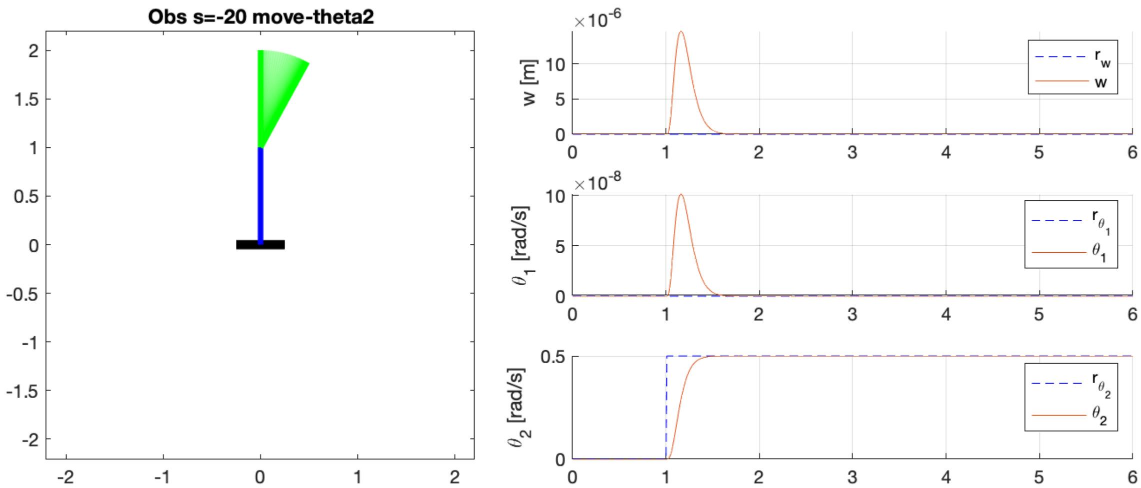
- Step Input of $r_{\theta_1} = 0.5\,[\mathrm{rad}]$**:**



(a) Simulation (b) Step Response

Figure 2-11. Simulation and Step Response for step input of $r_{\theta_1} = 0.5$

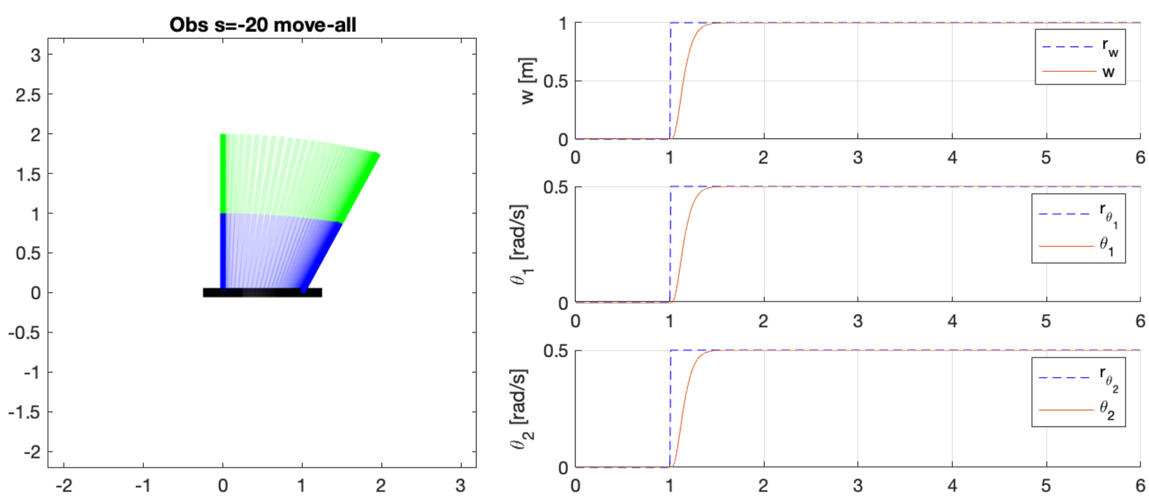
- Step Input of $r_{\theta_2} = 0.5\,[\mathrm{rad}]$**:**



(a) Simulation (b) Step Response

Figure 2-12. Simulation and Step Response for step input of $r_{\theta_2} = 0.5$

- Combined Step Input:



(a) Simulation (b) Step Response

Figure 2-13. Simulation and Step Response for step input of $r_w = 1$ and $r_\theta = 0.5$

Glossary

BSI Bode Sensitivity Integral.

LHP Left Hand Plane.

ORHP Open Right Hand Plane.

PoI Poisson Integral.

RHP Right Hand Plane.

Appendix A Code for Main

```
close all;
clear all;
clc;

%% User Param:
REBUILD = true ; % Force clear output folder!

ENV_P5  = false|| REBUILD;
ENV_P6  = false|| REBUILD;
ENV_P6b = false|| REBUILD;
EN_SIM  = true  || REBUILD;

% P5 P6:
A3 =  [0      1.0000          0           0
       0           0   -3.26666666666666667          0
       0           0          0     1.0000
       0           0   19.6000           0];
B3 = [ 0           0
       0.8888888888888888889    -1.33333333333333333
       0           0
      -1.33333333333333333      8.0000  ];
C3 = [1 0 0 0; 0 0 1 0];
D3 = zeros(2,2);
P3 = ss(A3,B3,C3,D3);

% P6:
A6 =  [0         1          0          0       0        0;
       0         0      -4.41          0       0.49     0;
       0         0          0          1       0        0;
       0         0          61.74      0       -26.46   0;
       0         0          0          0       0        1;
       0         0          -79.38     0       67.62    0];
B6 = [ 0         0          0
       0.9333    -2.40      0.80;
       0         0          0;
      -2.40      33.60      -43.20;
       0         0          0;
       0.80      -43.20     110.40];
C6 = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0];
D6 = zeros(3,3);
P6 = ss(A6,B6,C6,D6);

% MISC: Conditional Params [For Simulation]
FIG_SIZE = [400, 300];
Y_peak  = 0.5;
T       = 0.01;
T_END   = 6; %[s]

%% init.
helper.createFolder("output", REBUILD);
helper.createFolder("output/p5", false);
helper.createFolder("output/p6", false);
helper.createFolder("output/p6b", false);
disp("=== Output Folder Inited for P5 and P6 ===")

%% TF : Common for P5 and P6
```

```
fprintf("=== Init MATLAB [Simulation:%d] ... \n", EN_SIM);
% tf
s = tf('s');
% Generate Test Data
t0 = 0:T:1;
t1 = (1+T):T:T_END;
t  = 0:T:T_END;
r_step = [zeros(1, size(t0,2)) ones(1, size(t1,2))];
r_zero = r_step * 0;

%% Problem 5 Solution %%%%%%%%%%%%%%%%%%%%
fprintf("=== Perform Computation [P5:%d] ... \n", ENV_P5);
if ENV_P5
    zpk(P3) % verify it works!

    %% a) Unstable pz cancellation (1. 2) entry
    %       - show MIMO Bode plot of P3(s)
    figure()
    bodeplot(P3)
    grid on;
    helper.saveFigure([600 500], "p5", "P3_bode_plot")

    %% b) off-diagonal approximation:
    m = 0.5;
    L = 1.0;
    g = 9.8;
    M = 1;
    P3_aug = [[ 1/(m+M)/(s^2)     0                          ];
              [ 0                 6/(2*m*L^2*s^2 - 3*m*g*L) ]]
    % bode:
    figure()
    bodeplot(P3_aug)
    grid on;
    helper.saveFigure([600 500], "p5", "P3_aug_bode_plot")

    %% c) C11:
    % For P3_aug(1,1), we need a lead filter to pull the PM from 0 to > 50
    % To design:
    % sisotool(P3_aug(1,1));
    C11 = 1000 * (s + 1) / (s + 100);

    %% d) C22:
    % sisotool: 1 integrator + 1 lead filter
    % save:
    C22 = 170.54 * (s+1) * (s+3) / (s * (s+50));

    %% e) Analyze
    C_aug = [C11 0; 0 C22]
    %% ideal:
    L_aug = minreal(P3_aug * C_aug);
    TF_ideal = minreal(L_aug * inv(eye(2) + L_aug));

    % mimo step:
    figure()
    step(TF_ideal);
    grid on;
    helper.saveFigure([600 500], "p5", "Ideal_step_response")

    % siso figures: bode + rl + step
    helper.sisoPlot(zpk(L_aug(1,1)), [600 500], "p5", "Ideal(1,1)")
    helper.sisoPlot(zpk(L_aug(2,2)), [600 500], "p5", "Ideal(2,2)")

    % report performance:
    perform_table_ideal = helper.performance_table_mimo2x2(TF_ideal);

    % step response grid
    IC = [0,0,0,0]
    helper.mimo_step_response(TF_ideal, 6, ["w", "\theta"], ...
        ["r_w", "r_{\theta}"], IC, "Ideal IC=0", "p5");

    %% actual:
    L_act = minreal(P3 * C_aug);
```

```
    TF_actual = minreal(L_act * inv(eye(2) + L_act));

    % mimo step:
    figure()
    step(TF_actual);
    grid on;
    helper.saveFigure([600 500], "p5", "Actual_step_response")

    % siso figures: bode + rl + step
    helper.sisoPlot(zpk(L_aug(1,1)), [600 500], "p5", "Ideal(1,1)")
    helper.sisoPlot(zpk(L_aug(1,2)), [600 500], "p5", "Ideal(1,2)")
    helper.sisoPlot(zpk(L_aug(2,1)), [600 500], "p5", "Ideal(2,1)")
    helper.sisoPlot(zpk(L_aug(2,2)), [600 500], "p5", "Ideal(2,2)")

    % report actual performance:
    perform_table_actual = helper.performance_table_mimo2x2(TF_actual);

    % step response grid
    IC = [0,0,0,0,0,0,0]
    helper.mimo_step_response(TF_actual, 6, ["w", "\theta"], ...
        ["r_w", "r_{\theta}"], IC, "Actual IC=0", "p5");

    %% [Optional] f) simulation:
    r_theta = 0.5 * r_step'; % scale down to 0.5 [rad] peak
    r_w = r_step';
    helper.simulation_and_plot_mimo(TF_actual, [r_w r_theta], t', "actual-both", EN_SIM, "p5", true);

    % fix base:
    r_w_0 = r_w * 0;
    helper.simulation_and_plot_mimo(TF_actual, [r_w_0 r_theta], t', "actual-fixed-w", EN_SIM, "p5", true);

    %% fix theta:
    r_theta_0 = r_theta * 0;
    helper.simulation_and_plot_mimo(TF_actual, [r_w r_theta_0], t', "actual-fixed-theta", EN_SIM, "p5", true);
end
close all; % end of program

%% Problem 6 (a) Solution %%%%%%%%%%%%%%%%%%%%
fprintf("=== Perform Computation [P6(a):%d] ... \n", ENV_P6);
if ENV_P6
    %% i) TF:
    zpk(P3) % verify it works!
    %% ii) Plain state-feedback control with integral action control:
    % check controllability:
    Atilda = [A3 zeros(4, 2); C3 zeros(2,2)]
    Btilda = [B3; zeros(2, 2)]
    rk_ctrl = rank(ctrb(Atilda, Btilda))
    if (rk_ctrl == length(Atilda))
        disp("> Augmented Controller is Controllable");
    else
        disp("> Augmented Controller is NOT Controllable");
    end
    % Arbitrarily place 6 closed-loop poles about s=-20
    pole_s = -20
    Poles = pole_s * ones(1, 6) + [0 0.01 0.03 -0.01 -0.02 -0.03]
    K = place(Atilda, Btilda, Poles)
    K1 = K(1:2, 1:4)
    K2 = K(1:2, 5:6)
    % check closed-loop step response
    A_cls = [A3-B3*K1 -B3*K2; C3 zeros(2, 2)]
    B_cls = [zeros(4, 2); -eye(2)]
    C_cls = [C3 zeros(2,2)]
    D_cls = zeros(2,2)
    CLS = ss(A_cls, B_cls, C_cls, D_cls)
    % step response grid
    IC = [0,0,0,0,0,0]
    helper.mimo_step_response(CLS, 1.5, ["w", "\theta"], ...
        ["r_w", "r_{\theta}"], IC, "IC=0", "p6");
    figure()

```

```
    %% plot step response and simulation:
    helper.simulation_and_plot_mimo(CLS, [r_step' r_zero'], t', "s=-20 fixed-theta", EN_SIM, "p6", true);
    helper.simulation_and_plot_mimo(CLS, [r_zero' 0.5 * r_step'], t', "s=-20 fixed-w", EN_SIM, "p6", true);
    helper.simulation_and_plot_mimo(CLS, [r_step' 0.5 * r_step'], t', "s=-20 both", EN_SIM, "p6", true);
    %% [Optional] Poles further away to speed up:
    pole_s = -40
    Poles = pole_s * ones(1, 6) + [0 0.01 0.03 -0.01 -0.02 -0.03]
    K = place(Atilda, Btilda, Poles);
    K1 = K(1:2, 1:4);
    K2 = K(1:2, 5:6);
    % check closed-loop step response
    A_cls = [A3-B3*K1 -B3*K2; C3 zeros(2, 2)];
    B_cls = [zeros(4, 2); -eye(2)];
    C_cls = [C3 zeros(2,2)];
    D_cls = zeros(2,2);
    CLS = ss(A_cls, B_cls, C_cls, D_cls);
    % step response grid
    IC = [0,0,0,0,0,0]
    helper.mimo_step_response(CLS, 1.5, ["w", "\theta"], ...
        ["r_w", "r_{\theta}"], IC, "IC=0-s=-40", "p6");
    % plot step response and simulation:
    helper.simulation_and_plot_mimo(CLS, [r_step' r_zero'], t', "s=-40 fixed-theta", EN_SIM, "p6", true);
    helper.simulation_and_plot_mimo(CLS, [r_zero' 0.5 * r_step'], t', "s=-40 fixed-w", EN_SIM, "p6", true);
    helper.simulation_and_plot_mimo(CLS, [r_step' 0.5 * r_step'], t', "s=-40 both", EN_SIM, "p6", true);
end
close all; % end of program

%% Problem 6 (b) Solution %%%%%%%%%%%%%%%%%%%%%
fprintf("=== Perform Computation [P6(b):%d] ... \n", ENV_P6b);
if ENV_P6b
    %% i) TF:
    zpk(P6) % verify it works!
    %% ii) Plain state-feedback control with integral action control:
     % check controllability:
    Atilda = [A6 zeros(6, 3); C6 zeros(3,3)]
    Btilda = [B6; zeros(3, 3)]
    rk_ctrl = rank(ctrb(Atilda, Btilda))
    if (rk_ctrl == length(Atilda))
        disp("> Augmented Controller is Controllable");
    else
        disp("> Augmented Controller is NOT Controllable");
    end
    % Arbitrarily place 9 closed-loop poles about s=-20
    pole_s = -20
    Poles = pole_s * ones(1, 9) + [0 0.01 0.03 -0.01 -0.02 -0.03 -0.04 0.04 0.05]
%     Ks = place(Atilda, Btilda, Poles)
    Ks = place(Atilda,Btilda,Poles);
    K1 = Ks(1:3, 1:6)
    K2 = Ks(1:3, 7:9)
    % check closed-loop step response
    A_cls = [A6-B6*K1 -B6*K2; C6 zeros(3,3)]
    B_cls = [zeros(6, 3); -eye(3)]
    C_cls = [C6 zeros(3,3)]
    D_cls = zeros(3,3)
    CLS = ss(A_cls, B_cls, C_cls, D_cls)

    figure()
    step(CLS)
    %% plot step response grid:
    IC = [0,0,0,0,0,0,0,0,0]
    helper.mimo_step_response(CLS, 1.5, ["w", "\theta_1", "\theta_2"], ...
        ["r_w", "r_\theta_1", "r_\theta_2"], IC, "IC=0", "p6b");

    if EN_SIM
        % plot step response and simulation:
        helper.simulation_and_plot_mimo_double_pendulum(CLS, ...
            [r_step' r_zero' r_zero'], t', "s=-20 move-w", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS, ...
```

```
            [r_zero' 0.5 * r_step' r_zero'], t', "s=-20 move-theta1", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS, ...
            [r_zero' r_zero' 0.5 * r_step'], t', "s=-20 move-theta2", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS, ...
            [r_step' 0.5 * r_step' 0.5 * r_step'], t', "s=-20 move-all", EN_SIM, "p6b", true);
    end

    %% iii) Observer-based Control:
    % check observability:
    rk_obs = rank(obsv(A6, C6))
    if (rk_obs == length(A6))
        disp("> Augmented Controller is Observable");
    else
        disp("> Augmented Controller is NOT Observable");
    end

    % place observer error dynamics at s = -4
    Poles_obs = -4 * ones(1, 6) + [0 0.01 0.03 -0.01 -0.02 -0.03]
    H = place(A6', C6', Poles_obs)'

    %%% find optimal:
    %opt = lqr(A6', C6', eye(6), eye(3))'

    % close loop:
    Acl = [A6-B6*K1 -B6*K2 B6*K1; C6 zeros(3,3) zeros(3,6); zeros(6,6) zeros(6,3) A6-H*C6];
    Bcl = [zeros(6,3); -eye(3); zeros(6,3)];
    Ccl = [C6 zeros(3,3) zeros(3,6)];
    Dcl = zeros(3, 3);
    CLS_obs = ss(Acl,Bcl,Ccl,Dcl);

    %% plot step response grid:
    IC = [[0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0]
          [0,0,0,0,0, 0,0,0,0,0, 0.5,0.5,0.5,0.5,0.5]]
    helper.mimo_step_response(CLS_obs, 1.5, ["w", "\theta_1", "\theta_2"], ...
        ["r_w", "r_\theta_1", "r_\theta_2"], IC, "Cls_obs IC=0", "p6b");

     %%
    if EN_SIM
        % plot step response and simulation:
        helper.simulation_and_plot_mimo_double_pendulum(CLS_obs, ...
            [r_step' r_zero' r_zero'], t', "Obs s=-20 move-w", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS_obs, ...
            [r_zero' 0.5 * r_step' r_zero'], t', "Obs s=-20 move-theta1", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS_obs, ...
            [r_zero' r_zero' 0.5 * r_step'], t', "Obs s=-20 move-theta2", EN_SIM, "p6b", true);
        helper.simulation_and_plot_mimo_double_pendulum(CLS_obs, ...
            [r_step' 0.5 * r_step' 0.5 * r_step'], t', "Obs s=-20 move-all", EN_SIM, "p6b", true);
    end
end

close all; % end of program


%%
%% Two Rod
```

Code 1: Main Lab Contents

Appendix B Code for Helper Class

```
%% Helper Functions %%
classdef helper
    methods(Static)
        function createFolder(path, clear_folder)
            if ~exist(path)
                mkdir(path)
                fprintf("[HELPER] Folder created!\n");
            else
                if ~isempty(path) & clear_folder
                    rmdir(path, 's');
                    mkdir(path);
```

```
                    fprintf("[HELPER] Folder is emptied, %s\n", path);
                else
                    fprintf("[HELPER] Folder already existed!\n");
                end
            end
        end
        function RH_criterion(coeffs) % [ n , .... , 0 ]
            num = size(coeffs,2);
            n = ceil(num / 2);

            if mod(num,2) == 1 % odd number
                A = [coeffs, 0];
            else
                A = coeffs;
            end

            RH_mat = reshape(A, 2, n);

            for j = 1:n
                b = sym(zeros(1, n));
                for i = 1:n-1
                    b(i) = RH_mat(j, 1) * RH_mat(j+1, i+1);
                    b(i) = RH_mat(j+1, 1) * RH_mat(j, i+1) - b(i);
                    b(i) = b(i)/RH_mat(j+1, 1);
                    b(i) = simplifyFraction(b(i))
                end
                RH_mat = [RH_mat; b];
            end
            disp(RH_mat)
        end
        function saveFigure(DIMENSION, FOLDER, FILE_NAME)
            set(gcf,'units','points','position',[0, 0, DIMENSION(1), DIMENSION(2)]);
            exportgraphics(gcf,sprintf('output/%s/%s.png', ...
                FOLDER, FILE_NAME),'BackgroundColor','white');
        end
        function sisoPlot(L_TF, DIMENSION, FOLDER, TAG)
            % Plot
            figure()

            subplot(2, 2, [1,3])
            margin(L_TF)
            grid on

            subplot(2, 2, 2)
            rlocus(L_TF)

            subplot(2, 2, 4)
            G_TF = minreal(L_TF/(L_TF + 1));
            step(G_TF)
            grid on

            helper.saveFigure(DIMENSION, FOLDER, sprintf("siso_plot_%s", TAG))
        end
        %% Simulation
        function simulation_and_plot(TF_r2theta, TF_r2w, r_theta, t, tag, ifsim, FOLDER, verbose)
            fprintf("=== SIMULATION [%s:%s] ===\n", FOLDER, tag)
            % sim
            y_theta = lsim(TF_r2theta, r_theta, t);
            y_w = lsim(TF_r2w, r_theta, t);

            % analysis
            rinfo = stepinfo(r_theta,t);
            yinfo = stepinfo(y_theta,t);
            t_delay = (yinfo.PeakTime - rinfo.PeakTime);
            e_peak = (yinfo.Peak - rinfo.Peak);
            os = e_peak/ rinfo.Peak * 100;
            info_str = sprintf("[ %10s ]: t_{settling}= %.4f, \\theta_{settling}= %.4f, os_{peak}= %.2f %%", ...
                tag, yinfo.SettlingTime, y_theta(length(y_theta)), os)
            if verbose
                disp("r_theta info")
                disp(rinfo)
```

```
                disp("y_theta info")
                disp(yinfo)
            end

            % Plot
            figure()

            subplot(3, 1, 1)
            plot(t, r_theta)
            grid on;
            ylabel("r_{\theta} [rad/s]")
            title(info_str)

            subplot(3, 1, 2)
            hold on;
            plot(t, y_theta)
            plot(t, r_theta, '--', 'color', '#222222')
            grid on;
            ylabel("\theta [rad/s]")
            legend(["\theta", "r_{\theta}"])

            subplot(3, 1, 3)
            plot(t, y_w)
            grid on;
            ylabel("w [m]")
            xlabel("t [s]")

            helper.saveFigure([400, 300], FOLDER, sprintf("step_response_%s", tag))

            % Simulate
            if ifsim
                figure()
                single_pend_fancy_sim(t, [y_w, y_theta], [zeros(length(t),1) r_theta'], 1, 50, tag);
                helper.saveFigure([300, 300], FOLDER, sprintf("sim_%s", tag))
            end
        end
        function simulation_and_plot_mimo(TF, r, t, tag, ifsim, FOLDER, verbose)
            fprintf("=== SIMULATION [%s:%s] ===\n", FOLDER, tag)
            y = lsim(TF, r, t);

            % plot ref & resp:
            figure()
            subplot(2, 1, 1)
            hold on;
            plot(t, r(:,1), '--', 'color', 'blue')
            plot(t, y(:,1))
            grid on;
            ylabel("w [m]")
            legend(["r_{w}", "w"])

            subplot(2, 1, 2)
            hold on;
            plot(t, r(:,2), '--', 'color', 'blue')
            plot(t, y(:,2))
            grid on;
            ylabel("\theta [rad/s]")
            legend(["r_{\theta}", "\theta"])

            helper.saveFigure([400, 300], FOLDER, sprintf("square_response_%s", tag))

            % Simulate
            if ifsim
                figure()
                single_pend_fancy_sim(t, y, r, 1, 1, tag);
                helper.saveFigure([300, 300], FOLDER, sprintf("mimo_sim_%s", tag))
            end
        end
        function simulation_and_plot_mimo_double_pendulum(TF, r, t, tag, ifsim, FOLDER, verbose)
            fprintf("=== SIMULATION [%s:%s] ===\n", FOLDER, tag)
            y = lsim(TF, r, t);

            % plot ref & resp:
```

```
        figure()
        subplot(3, 1, 1)
        hold on;
        plot(t, r(:,1), '--', 'color', 'blue')
        plot(t, y(:,1))
        grid on;
        ylabel("w [m]")
        legend(["r_{w}", "w"])

        subplot(3, 1, 2)
        hold on;
        plot(t, r(:,2), '--', 'color', 'blue')
        plot(t, y(:,2))
        grid on;
        ylabel("\theta_1 [rad/s]")
        legend(["r_{\theta_1}", "\theta_1"])

        subplot(3, 1, 3)
        hold on;
        plot(t, r(:,3), '--', 'color', 'blue')
        plot(t, y(:,3))
        grid on;
        ylabel("\theta_2 [rad/s]")
        legend(["r_{\theta_2}", "\theta_2"])

        helper.saveFigure([400, 300], FOLDER, sprintf("response_%s", tag))

        % Simulate
        if ifsim
            figure()
            double_pend_fancy_sim(t, y, r, 1, 1, 1, tag);
            helper.saveFigure([300, 300], FOLDER, sprintf("mimo_sim_%s", tag))
        end
    end
    function mimo_step_response(TF, T_END, OUT_LABEL, IN_LABEL, IC, tag, FOLDER)
        fprintf("=== GENERATE STEP RESPONSE [%s:%s] ===\n", FOLDER, tag)
        t   = 0:0.001:T_END;
        r_step = ones(1, length(t))';
        r_zero = zeros(1, length(t))';

        [n, d] = size(TF);
        [icn, icd]=size(IC);

        figure()
        for i = 1:d
            r = repmat(r_zero,1,n);
            r(:, i) = r_step;

            for k = 1:icn
                y = lsim(TF, r, t, IC(k,:));
                for j = 1:n
                    ax(j,i) = subplot(n, d, j*n+i-n);
                    hold on;
                    grid on;
                    if k == 1
                        plot(t, r(:,j), '--', 'color', 'blue')
                        plot(t, y(:,j), 'red')
                    else
                        plot(t, y(:,j))
                    end

                    if i == 1
                        ylabel(OUT_LABEL(j));
                    end
                    if j == 1
                        title(IN_LABEL(i));
                    end
                    ylim([min(y(:,j))-0.1 max(y(:,j))+0.1]);
                end
            end
        end

```

```
            for j = 1:n
                linkaxes(ax(j,:),'y');
            end

            helper.saveFigure([n*200, d*200], FOLDER, sprintf("mimo_response_%s", tag))
        end
        %% Custom Bode
        function bode_plot_custom(TF, tag, FOLDER, verbose)
            fprintf("=== BODE PLOT [%s:%s] ===\n", FOLDER, tag);
            % custom bode plot
            [mag,phase,wout] = bode(TF);
            figure()
            subplot(2,1,1)
            semilogx(wout, mag2db(abs(mag(:))))
            grid on
            ylabel("Magnitude [dB]")
            hold on
            yline(0, 'r--')
            legend(["W(s)", "0 dB"])

            subplot(2,1,2)
            semilogx(wout, phase(:))
            grid on
            ylabel("Phase [deg]")
            xlabel("\omega [rad/s]")

            if verbose
                m = allmargin(TF);
                disp(m)
                title(sprintf("BODE PLOT [%s:%s]", FOLDER, tag));
            end

            helper.saveFigure([400, 300], FOLDER, tag)
        end
        function bode_plot_margin(TF, tag, FOLDER)
            fprintf("=== BODE PLOT [%s:%s] ===\n", FOLDER, tag);
            % custom bode plot
            figure()
            margin(TF)
            grid on
            helper.saveFigure([400, 300], FOLDER, tag)
        end
        %% misc
        function tflatex(TF)
            [num,den] = tfdata(TF);
            syms s
            t_sym = poly2sym(cell2mat(num),s)/poly2sym(cell2mat(den),s);
            latex(vpa(t_sym, 5))
        end
        function [Table] = performance_table_mimo2x2(TF_actual)
            [y,t]=step(TF_actual);
            status = stepinfo(TF_actual);
            A = status(1,1);
            B = status(1,2);
            C = status(2,1);
            D = status(2,2);
            CONTENT = [ ...
                [A.RiseTime, A.SettlingTime, A.PeakTime, A.Peak, y(length(y)-1,1,1), A.Overshoot, A.Undershoot]; ...
                [B.RiseTime, B.SettlingTime, B.PeakTime, B.Peak, y(length(y)-1,1,2), B.Overshoot, B.Undershoot]; ...
                [C.RiseTime, C.SettlingTime, C.PeakTime, C.Peak, y(length(y)-1,2,1), C.Overshoot, C.Undershoot]; ...
                [D.RiseTime, D.SettlingTime, D.PeakTime, D.Peak, y(length(y)-1,2,2), D.Overshoot, D.Undershoot]; ...
            ];
            Table = array2table(CONTENT, ...
                'VariableNames',{'t_{rise}','t_{settling}','t_{peak}','y_{peak}','y_{ss}','OS\%','US\%'}, ...
                'RowName',{'(1,1)','(1,2)', '(2,1)', '(2,2)'});
            disp("Performance Summary:")
            disp(Table)
```

```
        end
        function table2latex(T, filename)
            % Error detection and default parameters
            if nargin < 2
                filename = 'table.tex';
                fprintf('Output path is not defined. The table will be written in %s.\n', filename);
            elseif ~ischar(filename)
                error('The output file name must be a string.');
            else
                if ~strcmp(filename(end-3:end), '.tex')
                    filename = [filename '.tex'];
                end
            end
            if nargin < 1, error('Not enough parameters.'); end
            if ~istable(T), error('Input must be a table.'); end

            % Parameters
            n_col = size(T,2);
            col_spec = [];
            for c = 1:n_col, col_spec = [col_spec 'l']; end
            col_names = strjoin(T.Properties.VariableNames, ' & ');
            row_names = T.Properties.RowNames;
            if ~isempty(row_names)
                col_spec = ['l' col_spec];
                col_names = ['& ' col_names];
            end

            % Writing header
            fileID = fopen(filename, 'w');
            fprintf(fileID, '\\begin{tabular}{%s}\n', col_spec);
            fprintf(fileID, '%s \\\\ \n', col_names);
            fprintf(fileID, '\\hline \n');

            % Writing the data
            try
                for row = 1:size(T,1)
                    temp{1,n_col} = [];
                    for col = 1:n_col
                        value = T{row,col};
                        if isstruct(value), error('Table must not contain structs.'); end
                        while iscell(value), value = value{1,1}; end
                        if isinf(value), value = '$\infty$'; end
                        temp{1,col} = num2str(value);
                    end
                    if ~isempty(row_names)
                        temp = [row_names{row}, temp];
                    end
                    fprintf(fileID, '%s \\\\ \n', strjoin(temp, ' & '));
                    clear temp;
                end
            catch
                error('Unknown error. Make sure that table only contains chars, strings or numeric values.');
            end

            % Closing the file
            fprintf(fileID, '\\hline \n');
            fprintf(fileID, '\\end{tabular}');
            fclose(fileID);
        end
    end
end
```

Code 2: Helper and commonly used functions by main