UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

ECE 488 - Project 3 & 4

Prepared by:

Jianxiang (Jack) Xu [20658861]

17 March 2021

Table of Contents

1 Problem P3: Controller Design by Youla Parameterization for the SISO aiming system

1.1 (a) Controller Design $C_1(s)$ with Youla Parameterization

In order to achieve perfect steady-state tracking of steps, the final $C_1(s)$ must have an integrator in it. To achieve this, we may augment the plant with an integrator term (Equation (1)), and design an augmented controller $C_1^{aug}(s)$ for the augmented plant $P_1^{aug}(s)$ through Youla Parameterization.

$$P_1^{aug}(s) = \frac{1}{s} P_1(s) = \frac{8}{s\,(s-4.427)\,(s+4.427)} \tag{1}$$

To perform Youla Parameterization, coprime factorization is performed with a scale factor of 10 to avoid numerical issues. The resultant coprimes can be found below:

$$M(s) = \frac{s(s-4.427)(s+4.427)}{(s+30)(s+20)(s+10)} \tag{2}$$

$$N(s) = \frac{8}{(s+30)(s+20)(s+10)} \tag{3}$$

$$X(s) = \frac{255510(s^2+7.861s+17.61)}{(s+30)(s+20)(s+10)} \tag{4}$$

$$Y(s) = \frac{(s+65.16)(s^2+54.84s+2246)}{(s+30)(s+20)(s+10)} \tag{5}$$

For simplicity, we assume the $Q(s) = 1$, which may result a complex augmented controller:

$$C_1^{aug}(s) = \frac{(s+255500)(s^2+7.861s+17.61)}{(s+65.15)(s^2+54.85s+2246)} \tag{6}$$

We may now obtain the final controller by applying the integrator term to the augmented controller obtained just now:

$$C_1(s) = \frac{1}{s} C_1^{aug}(s) = \frac{(s+255500)(s^2+7.861s+17.61)}{s(s+65.15)(s^2+54.85s+2246)} \tag{7}$$

The detailed implementation can be found in Code 1.

1.2 (b) Closed-loop step response for controller designed in Section 1.1

We may now run the simulation with a step input of 0.5 [rad] same as the Lab 1 and Lab 2. We may find the closed-loop system is indeed stable and the steady-state tracking error is zero from Figure 1-1 below.

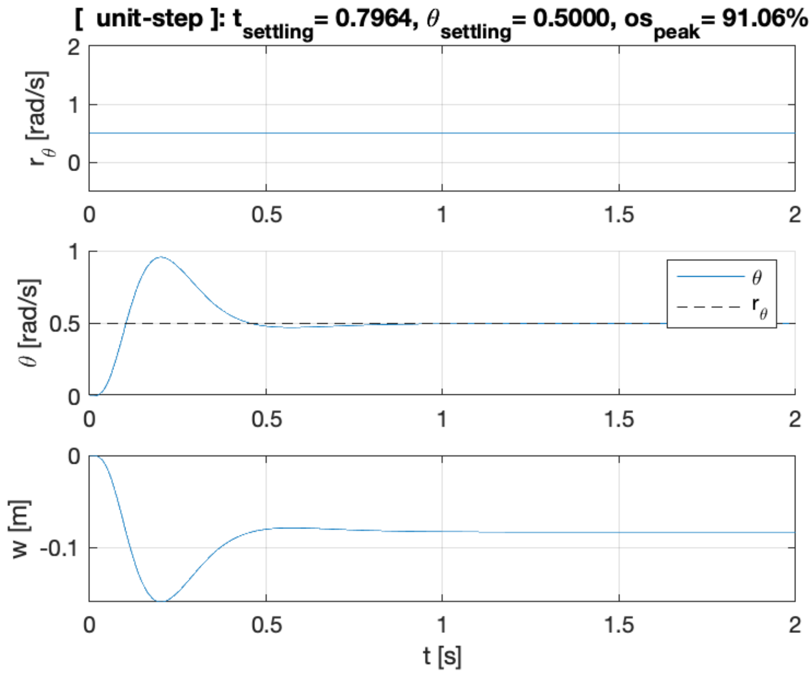

Figure 1-1. Closed-loop step response for controller $C_1(s)$ designed in Section 1.1

As expected, a simple $Q(s) = 1$ results a stable system, but there is no guarantee on the transient performance. From the Figure 1-1, we may observe a 91% overshoot and 0.7964 [s] settling time. The overshoot is quite significant.

To further analyze the system, a bode plot is generated for the closed-loop system, as shown in Figure 1-2 below.

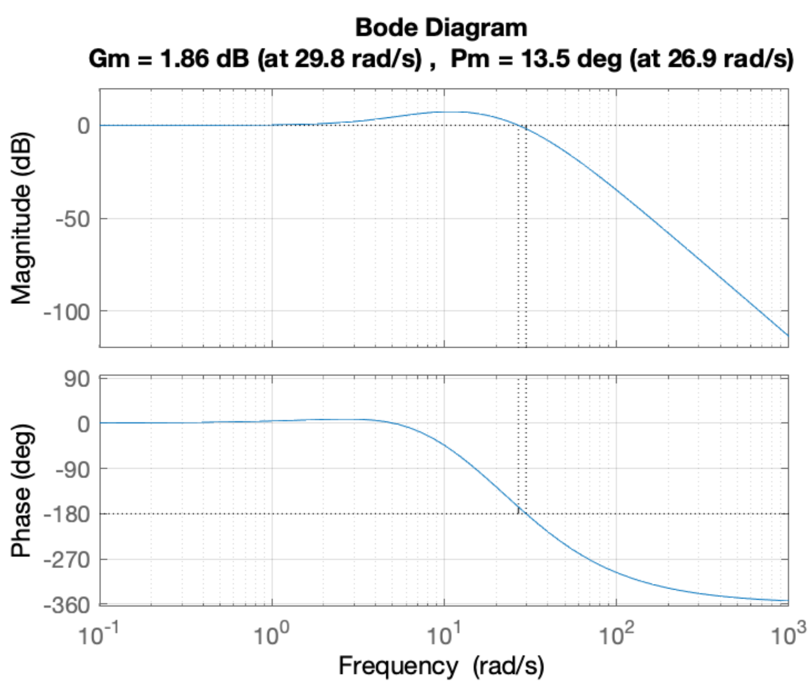

Figure 1-2. Bode plot for the closed-loop system designed in Section 1.1

From Figure 1-2, both gain margin (1.86 [dB]) and phase margin (13.5 [deg]) are extremely small, hence the system is quite sensitive to noise and disturbances. To improve the transients and phase margin, we can adjust the parameter $Q(s)$ such that the desired criterion is met. Based on the research, many system utilizes Youla Parameterization with stochastic gradient descent, graph theory, and genetic algorithm to improve the transient performance by manipulating the $Q(s)$. Since, any proper transfer function of $Q(s)$ would result a stable controller from the Youla formulation, hence, all we need to do is to adaptively find the optimal solution via various adaptive and un-supervised learning approach.

1.3 (c) [Optional] Simulation Visualization

The performance of the controller is visualized with the provided simulation, as shown in Figure 1-3 below.

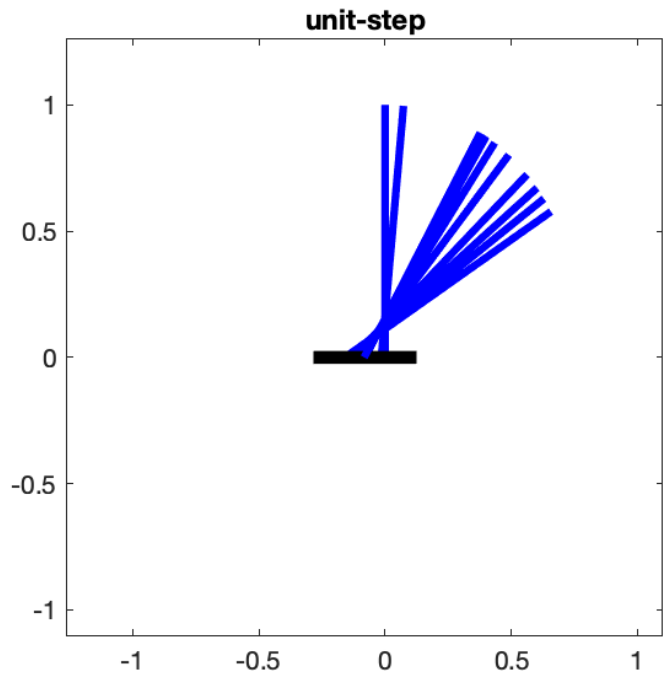

Figure 1-3. Final simulation result of a unit-step response for the closed-loop system designed in Section 1.1

2 Problem 4: Performance limitations associated with the SISO aiming system

2.1 (a) Performance limitations of one-rod system

2.1.1 (i) Overshoot y_{os} bound

For a unit-step of 0.5 [rad/s], the overshoot bound can be formulated in Equation (8), and its corresponding bounding curve can be seen in Figure 2-1 below.

$$y_{os} \geq (1 - 0.9\, y_{\infty})(e^{p t_r} - 1) > 0 \tag{8}$$

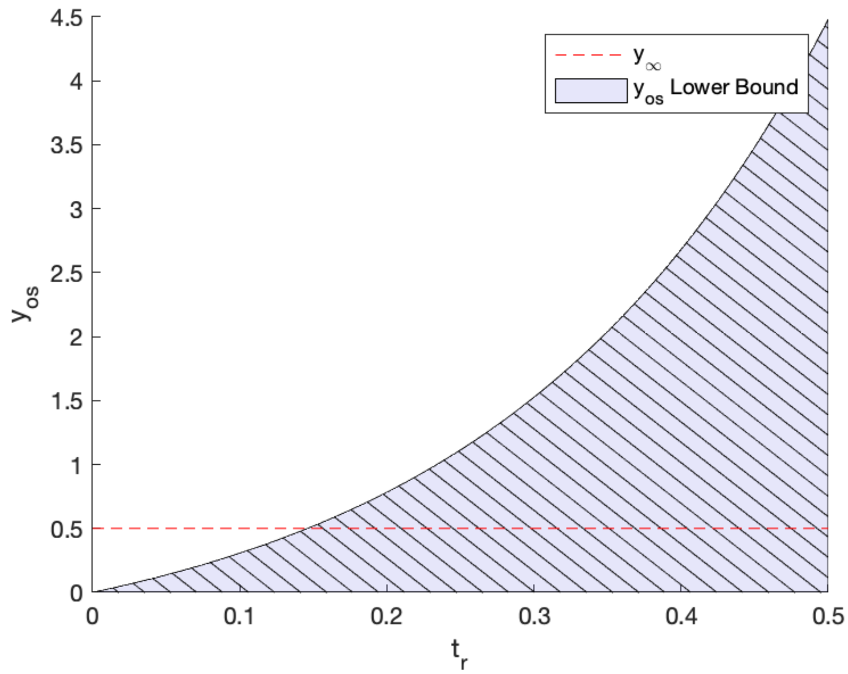

Figure 2-1. Overshoot y_{os} bound

2.1.2 (ii) Ω bound

Three additional design specifications are imposed:

I For tracking performance, $|S(j\omega)| \leq -20\,[\text{dB}]$ for $\omega \in [0, 0.1]\,[\text{rad/s}]$
II For robust stability, $\max_{\omega}|S(j\omega)| \leq 5\,[\text{dB}]$
III "Shut off" $\forall \omega > \Omega\,[\text{rad/s}]$

Since the controller stabilizes the closed loop system, and the loop gain has a relative degree of at least 2, the BSI (Bode Sensitivity Integral) holds. We may now derive the *Omega* boundary (Equation (15)) from the BSI inequality for the sensitivity function, as derived below:

$$\int_0^{\infty} \ln|S(j\omega)|d\omega = \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) \tag{9}$$

$$\text{Spec III \& Spec I} \Rightarrow \quad \int_0^{0.1} \ln|S(j\omega)|d\omega + \int_{0.1}^{\Omega} \ln|S(j\omega)|d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) \tag{10}$$

$$\text{Spec I} \Rightarrow \quad \ln\{\max_{\omega\in[0,0.1]}|S(j\omega)|\} \int_0^{0.1} 1 d\omega + \int_{0.1}^{\Omega} \ln|S(j\omega)|d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) \tag{11}$$

$$\int_{0.1}^{\Omega} \ln|S(j\omega)|d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) - 0.1\ln\{\max_{\omega\in[0,0.1]}|S(j\omega)|\} \tag{12}$$

$$\text{Spec II} \Rightarrow \quad \ln\{\max_{\omega\in[0.1,\Omega]}|S(j\omega)|\} \int_{0.1}^{\Omega} d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) - 0.1\ln\{\max_{\omega\in[0,0.1]}|S(j\omega)|\} \tag{13}$$

$$\ln\{\max_{\omega\in[0.1,\Omega]}|S(j\omega)|\}(\Omega - 0.1) \geq \pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) - 0.1\ln\{\max_{\omega\in[0,0.1]}|S(j\omega)|\} \tag{14}$$

$$\Omega \geq \frac{\pi \cdot \sum_{i=1}^{N_p} \text{Re}(p_i) - 0.1\ln\{\max_{\omega\in[0,0.1]}|S(j\omega)|\}}{\ln\{\max_{\omega\in[0.1,\Omega]}|S(j\omega)|\}} + 0.1 \tag{15}$$

Now, we may solve the derived boundary equation as stated in Equation (15) numerically:

$$\Omega \geq \frac{\pi * 4.427 - 0.1 * \ln(0.1)}{\ln(1.7783)} + 0.1 = 24.66\,[\text{rad/s}] \tag{16}$$

$$\therefore \quad \Omega \geq 24.66\,[\text{rad/s}] \tag{17}$$

2.1.3 (iii) Ω bound if there were no unstable plant pole

Now, let's pretend there is no unstable plant pole, as a result the Ω bound is much smaller than the one in part (ii). This is because there is a need for extra increase in sensitivity as a cost of having to stabilize the unstable poles in the plant, resulting the positive area in high frequency regions for sensitivity much larger than the negative area. Consequently, the Ω is much larger for unstable pole to spread the large area over a wider range of frequencies, as the peak of the sensitivity is capped as Spec II.

Conversely, if there is no unstable plant pole, it only needs a smaller range of high frequencies to compensate the loss in low frequencies.

$$\Omega \geq \frac{0 - 0.1 * \ln(0.1)}{\ln(1.7783)} + 0.1 = 0.5\,[\mathrm{rad/s}] \tag{18}$$

$$\therefore \quad \Omega \geq 0.5\,[\mathrm{rad/s}] \tag{19}$$

2.2 (b) Performance limitations of two-rod system

2.2.1 (i) Overshoot y_{os} bound

For the given two-rod plant below:

$$P_2(s) = \frac{110.4(s+5.539)(s-5.539)}{(s-4.331)(s+4.331)(s-10.52)(s+10.52)} \tag{20}$$

There are two unstable poles and one unstable zeros in the plant. We may believe the unstable pole further into the ORHP is more sever than the pole closer to the origin. The unstable pole would result overshoot, and the unstable zero would result undershoot. In addition, it is believe that the current system is extremely undesirable as the severe unstable pole is on the right side of the unstable zero (as seen in Figure 2-2 below), resulting the system hard to stabilize nicely.

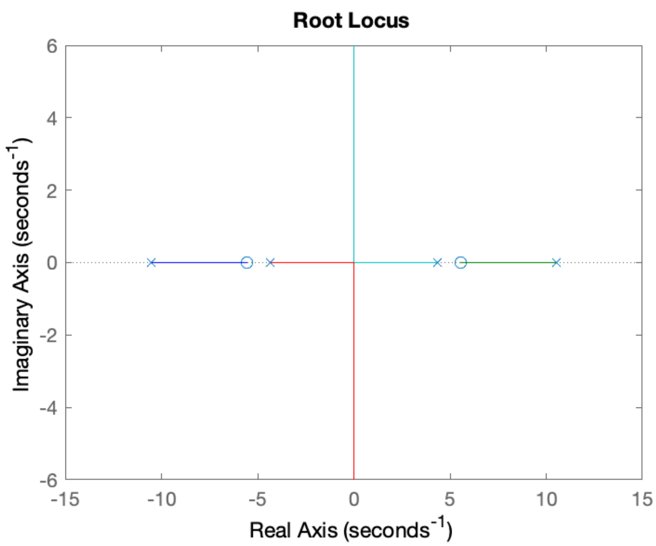

Figure 2-2. Root locus of the two-rod plant

Similar to Section 2.1.1, we would compute the overshoot y_{os} bound curve based on the most sever pole at the far right side ($s = 10.52$). After laying over against the one-rod system (in blue), as shown in Figure 2-3, we may see the overshoot bound for two-rod plant is much sever than the one-rod. As a result the two-rod system needs a much faster system to control.

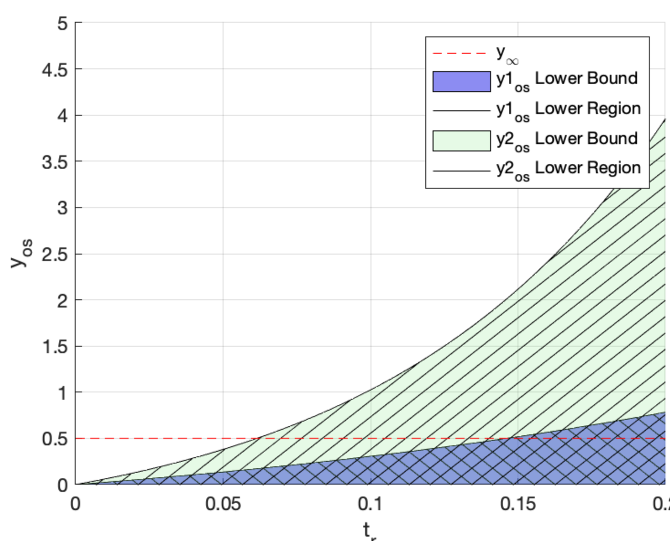

Figure 2-3. Overshoot bound curve ($y1_{os}$: one-rod, $y2_{os}$: two-rod) vs. rise time

However, it is also not ideal to have a too fast system, since, the unstable zero needs a slower system to minimize the undershoot, as shown in Figure 2-4.

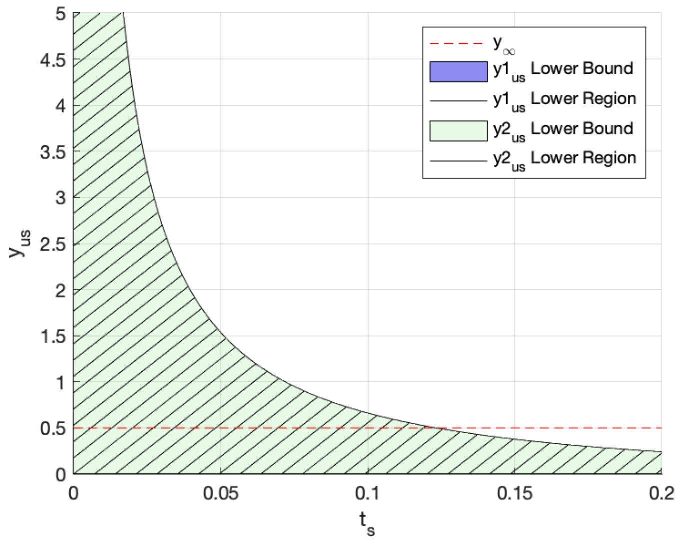

Figure 2-4. Undershoot bound curve ($y1_{us} = 0$: one-rod, $y2_{us}$: two-rod) vs. settling time

As a result, the two-rod system is more difficult to control than the one-rod system, since the two-rod system would have to suffer either or both severe undershoot and overshoot, whereas, the one-rod system can achieve simply by making the system faster.

Now, let's compute a meaningful bound on overshoot for the two-rod system using (1) on page 161. We may realize the sever pole results a negative bound which is meaningless, and the less sever pole results a positive bound of 3.5853.

$$y2_{os}^{severe-pole} = \frac{p}{z-p} = -2.112 \tag{21}$$

$$y2_{os}^{moderate-pole} = \frac{p}{z-p} = 3.5853 \tag{22}$$

Since the pg161 equation is only insightful for $p > z$, and in our case, we only know the fact that the transient performance is considerably pretty bad for the meaningful pole, since the pole and zero is close to each other. In addition, there exists a pole that is on right side of zero, resulting a negative overshoot, indicating, the situation is hopeless, and we would expect a pretty bad system. This bound tells us a combined effects of unstable zeros and poles, whereas the curve bounds tell us more insight about how the overshoot bound changes depending on rise time. Both bounds are telling us how bad this system is, but from two different perspectives.

2.2.2 (ii) Ω bound with BSI

Similar to Section 2.1.2, we may derive the lower bound on Ω from BSI, using the formulation of Equation (15). As a result, we get a lower bound of 81.55 [rad/s]. As expected, the effects of double unstable more severe poles result the system needs a much larger positive region to compensate the reduced sensitivity in the low frequency region.

$$\Omega \geq \frac{\pi * (4.3310 + 10.5200) - 0.1 * \ln(0.1)}{\ln(1.7783)} + 0.1 = 81.55\,[\mathrm{rad/s}] \tag{23}$$

$$\therefore \quad \Omega \geq 81.55\,[\mathrm{rad/s}] \tag{24}$$

As a result, we may conclude, the two-rod system is definitely more difficult to control, since it requires increased width of high frequency regions with increasing sensitivity to stabilize the system nicely.

2.2.3 (iii) Poisson Integral

Since there exists a real ORHP zero at $s = 5.539$, hence the PoI holds, as a result we may derive a sensitivity boundary function.

Let's assume the max peak sensitivity in the positive region of the Poisson Integral is $M = \max_{0.1<\omega<\Omega} |S(j\omega)|$. For simplicity, we let $\varepsilon = \max_{\omega\in[0,0.1]} |S(j\omega)|$.

We may now derive an relationship of the M relative to the choice of attenuation frequency Ω:

$$\int_0^\infty \ln|S(j\omega)|W(\omega)d\omega = \pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| \tag{25}$$

$$\text{Spec III \& Spec I} \Rightarrow \quad \int_0^{0.1} \ln|S(j\omega)|W(\omega)d\omega + \int_{0.1}^{\Omega} \ln|S(j\omega)|W(\omega)d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| \tag{26}$$

$$\text{Spec I} \Rightarrow \quad \ln\{\max_{\omega\in[0,0.1]} |S(j\omega)|\} \int_0^{0.1} W(\omega)d\omega + \ln\{\max_{\omega\in[0,0.1]} |S(j\omega)|\} \int_{0.1}^{\Omega} W(\omega)d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| \tag{27}$$

$$\ln(\varepsilon) \int_0^{0.1} W(\omega)d\omega + \ln(M) \int_{0.1}^{\Omega} W(\omega)d\omega \geq \pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| \tag{28}$$

$$\ln(\varepsilon) \left[2\tan^{-1}(\frac{\omega}{z})\right]_0^{0.1} + \ln(M) \left[2\tan^{-1}(\frac{\omega}{z})\right]_{0.1}^{\Omega} \geq \pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| \tag{29}$$

After rearrange:

$$\left[2\tan^{-1}(\frac{\omega}{z})\right]_{0.1}^{\Omega} \geq 0 \Rightarrow \quad \ln(M) \geq \frac{\pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| - \ln(\varepsilon)\left[2\tan^{-1}(\frac{\omega}{z})\right]_0^{0.1}}{\left[2\tan^{-1}(\frac{\omega}{z})\right]_{0.1}^{\Omega}} \tag{30}$$

$$\ln(M) \geq \ln(M) \frac{\pi \cdot \sum_{i=1}^{N_p} \ln|\frac{p_i+z}{p_i^*-z}| - \ln(\varepsilon)\left[\tan^{-1}(\frac{0.1}{z})\right]}{\left[\tan^{-1}(\frac{\Omega}{z})\right] - \left[\tan^{-1}(\frac{0.1}{z})\right]} \tag{31}$$

$$\ln(M) \geq \frac{10.266}{\tan^{-1}(0.18054\,\Omega) - 0.018052} \tag{32}$$

As a result, we can now plot the boundary equation Equation (32), and overlay the max peak requirement from Spec II:

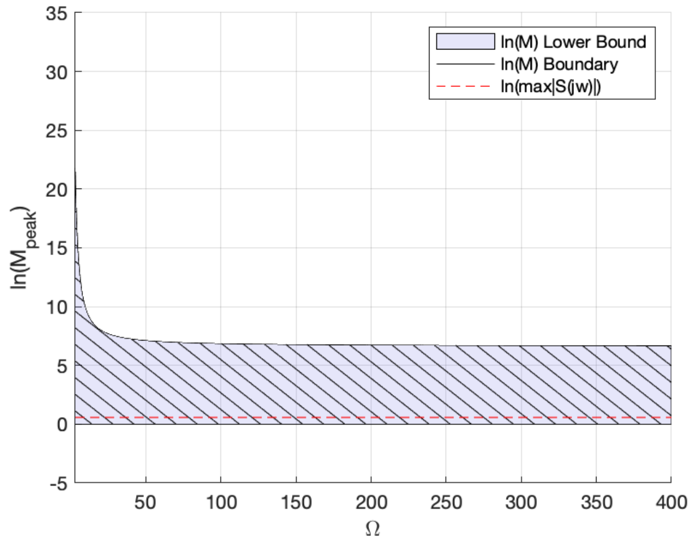

Figure 2-5. M peak bound vs. Ω

From the plot in Figure 2-5, we may find that no matter how we increase the attenuating frequency Ω, the peak of the high frequency region would exceed the max peak sensitivity requirement (Spec II). As Equation (33) suggested, we may find the fact that $\ln(M(\Omega)) \gg \ln(\varepsilon) \forall \Omega$.

$$\ln_{\Omega \to \infty}(M) \geq \frac{10.266}{\tan^{-1}(0.18054\,(\Omega \to \infty)) - 0.018052} = \frac{10.266}{\pi/2 - 0.018052} = 6.6115 \gg \ln(\varepsilon) = 0.5756 \tag{33}$$

In short, we can conclude it is impossible to satisfy the three design specifications given in Spec I, Spec II, and Spec III.

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;

%% init.
helper.createFolder("output", false);
helper.createFolder("output/p3", false);
helper.createFolder("output/p4", false);
disp("=== Output Folder Inited for P3 and P4 ===")

%% User Param:
ENV_P3 = true;
ENV_P4 = true;
EN_SIM  = true;

% P3:
COPRIME_SCALE_FACTOR = 10;
P3_Q = 1;

% P4:


% MISC: Conditional Params [For Simulation]
FIG_SIZE = [400, 300]
Y_peak  = 0.5;
T       = 0.001;
T_END   = 2; %[s]

%% TF : Common for P3 and P4
fprintf("=== Init MATLAB [Simulation:%d] ... \n", EN_SIM);
% tf
s = tf('s');
P1 = 8/(s-4.427)/(s+4.427);
P1w = -8/6/(s-4.427)/(s+4.427);
P2 = 110.4*(s+5.539)*(s-5.539)/(s-4.331)/(s+4.331)/(s-10.52)/(s+10.52);

% Generate Test Data
t = 0:T:T_END;
r_theta = Y_peak * ones(1, size(t,2)); % step @ 0.5 [rad/s]
% r_theta_sin_lf = Y_peak * sin((t ./ 2) * 2 * pi); % sin wave @ 0.5 [Hz]
% r_theta_sin_hf = Y_peak * sin((t .* 2) * 2 * pi); % sin wave @ 2 [Hz]
% r_theta_sin_hf2 = Y_peak * sin((t .* 10) * 2 * pi); % sin wave @ 10 [Hz]
% r_theta_sin_hf100 = Y_peak * sin((t .* 100) * 2 * pi); % sin wave @ 100 [Hz]


%% Problem 3 Solution %%%%%%%%%%%%%%%%%%%%%%
fprintf("=== Perform Computation [P3:%d] ... \n", ENV_P3);
if ENV_P3
    % augment the plant with an integrator for perfect steady-state
    %   tracking:
    P1_aug = P1 / s;
    % fetch coprime:
    [M, N, X, Y] = coprime(P1_aug, COPRIME_SCALE_FACTOR);
    % report:
    zpk(M)
    zpk(N)
```

```
    zpk(X)
    zpk(Y)
    % define Q(s)
    Q = P3_Q;
    % construct C_1-aug(s) for P_1-aug
    C_1_aug = (X + M * Q)/(Y - N * Q);
    C_1_aug = minreal(C_1_aug);
    fprintf("\n\n>>> [C1-augmented]:");
    zpk(C_1_aug)
    % de-aug C_1_aug(s) to get C1(s) for P1(s)
    C_1 = C_1_aug / s;
    C_1 = minreal(C_1);
    fprintf("\n\n>>> [C1]:");
    zpk(C_1)

    % compute TF:
    L_1 = minreal(P1 * C_1);
    TF_r2theta = minreal( 1 - 1 / (1 + L_1) ); % sensitivity
    TF_r2w = minreal( P1w  * (C_1) / (1 + L_1) );
    % bode plot:
    helper.bode_plot_margin(TF_r2theta, "bode_plot_r2theta", "p3")
    helper.bode_plot_margin(TF_r2w, "bode_plot_r2w", "p3")
    % Simulation
    helper.simulation_and_plot(TF_r2theta, TF_r2w, r_theta, t, "unit-step", EN_SIM, "p3", true)
end

%% Problem 4 Solution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("=== Perform Computation [P4:%d] ... \n", ENV_P4);
if ENV_P4
    %% a)
    % i) Minimum bound on y_os vs. t_r :
    y_inf = Y_peak;
    p_ORHP = 4.427;
    T_end4 = 0.5;
    t_r = 0:T:T_end4;
    y_os = (1 - 0.9 * y_inf) * (exp(p_ORHP * t_r) - 1);
    % plot:
    figure()
    hold on;
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    patch1 = patch([t_r T_end4 0], [y_os 0 0], [0.5,0.5,0.9]);
    hatchfill(patch1, 'single','HatchAngle',-45,'SpeckleWidth',10);
    alpha(patch1,.2)
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    legend(["y_{\infty}", "y_{os} Lower Bound"]);
    xlabel("t_r");
    ylabel("y_{os}");
    helper.saveFigure(FIG_SIZE, "p4", "y_os-bound");

    % ii)
    Omega2 = (pi * 4.427 - 0.1 * log(0.1))/log(1.7783) + 0.1

    % iii)
    Omega3 = (0 - 0.1 * log(0.1))/log(1.7783) + 0.1
    %% b) Two-rods System
    %% i) Minimum bound on y_os and y_us vs. t_r :
    p2_ORHP = [4.331, 10.52];
    z2_ORHP = 5.539;

    t_s = t_r;
    y2_os = (1 - 0.9 * y_inf) * (exp(max(p2_ORHP) * t_r) - 1);
    y2_us = (0.98 * y_inf) ./ (exp(z2_ORHP * t_s) - 1);

    % plot os:
    figure()
    hold on;
    grid on;
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    patch1 = patch([t_r T_end4 0], [y_os 0 0], [0.1,0.1,0.9]);
    patch2 = patch([t_r T_end4 0], [y2_os 0 0], [0.5,0.9,0.5]);
    h1 = hatchfill(patch1, 'single','HatchAngle',-45,'SpeckleWidth',10);
    h2 = hatchfill(patch2, 'single','HatchAngle',45,'SpeckleWidth',10);
```

```
    alpha(patch1,.5)
    alpha(patch2,.2)
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    legend(["y_{\infty}", "y1_{os} Lower Bound", "y1_{os} Lower Region", ...
        "y2_{os} Lower Bound", "y2_{os} Lower Region"]);
    xlabel("t_r");
    ylabel("y_{os}");
    xlim([0,0.2]);
    ylim([0,5]);
    helper.saveFigure(FIG_SIZE, "p4", "y2_os-bound");

    % plot us:
    figure()
    hold on;
    grid on;
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    patch1 = patch([0 T_end4 T_end4 0], [0 0 0 0], [0.1,0.1,0.9]);
    patch2 = patch([0 t_s(10:length(t_s)) T_end4 0], [5 y2_us(10:length(t_s)) 0 0], [0.5,0.9,0.5]);
    h1 = hatchfill(patch1, 'single','HatchAngle',-45,'SpeckleWidth',10);
    h2 = hatchfill(patch2, 'single','HatchAngle',45,'SpeckleWidth',10);
    alpha(patch1,.5)
    alpha(patch2,.2)
    plot([0, T_end4], [y_inf, y_inf], 'r--');
    legend(["y_{\infty}", "y1_{us} Lower Bound", "y1_{us} Lower Region", ...
        "y2_{us} Lower Bound", "y2_{us} Lower Region"]);
    xlabel("t_s");
    ylabel("y_{us}");
    xlim([0,0.2]);
    ylim([0,5]);
    helper.saveFigure(FIG_SIZE, "p4", "y2_us-bound");

    % compute bound with 161:
    y2_us_min = max(z2_ORHP ./ (p2_ORHP - z2_ORHP))
    y2_os_min = max(p2_ORHP ./ (z2_ORHP - p2_ORHP))
    % root locus ? (not req.)
    figure()
    rlocus(P2)
    helper.saveFigure(FIG_SIZE, "p4", "y2_root-locus");

    disp("Yes, its more difficult, faster response for overshoot, and there exists undershoot")
    %% ii) BSI => lower bound on \Omega
    MAX_SENSITIVITY = db2mag(5);
    omega_lower_bound_BSI = (pi * sum(p2_ORHP) - 0.1 * log(0.1))/(log(MAX_SENSITIVITY)) + 0.1
    %% iii) PI => lower bound on \Omega
    Omega_max = 400;
    Omega = 0.1:0.1:Omega_max;
    z = z2_ORHP;
    Const = pi * sum(log(abs((p2_ORHP + z2_ORHP)./(p2_ORHP - z2_ORHP))))
    M_lower_bound_log = (((Const - log(MAX_SENSITIVITY) * atan(0.1/z)))./(atan(Omega / z) - atan(0.1/z))
    );
    figure()
    hold on;
    grid on;
%     plot([0.1, 0.1], [0, 1.8], 'r--');
    patch1 = patch([Omega(20:length(Omega)) Omega_max 0], ...
        [M_lower_bound_log(20:length(Omega)) 0 0], [0.5,0.5,0.9]);
    hatchfill(patch1, 'single','HatchAngle',-45,'SpeckleWidth',10);
    plot([0, Omega_max], [log(MAX_SENSITIVITY), log(MAX_SENSITIVITY)], 'r--');
    alpha(patch1,.2)
    legend(["ln(M) Lower Bound", "ln(M) Boundary", "ln(max|S(jw)|)"]);
    xlabel("\Omega")
    ylabel("ln(M_{peak})")
    xlim([3, Omega_max])
    helper.saveFigure(FIG_SIZE, "p4", "M-bound");
end
```

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
        %% 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
    end
end
```

Code 2: Helper and commonly used functions by main