The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Department of Electrical and Electronic Engineering Faculty of Engineering

Electrical Energy Conditioning and Control

(EEEE2045 UNUK) (FYR1 22-23)

Control Coursework

Author:
George Downing

Student Number:
20273662

October 24, 2022

Contents

1 Aim of the Lab

The aim of this Lab is to complete a set of exercises to demonstrate the understanding of the concepts of control systems. The exercises are based on the material covered in the lectures and the textbook.

Two design questions will also be completed, proving the understanding of the concepts of control systems, and the ability to apply them to a real world problem.

Matlab will be used to complete the exercises and design questions. The ability to use Matlab will be demonstrated by the through the use of commands and functions required. The code used to complete the exercises and design questions will be included in the report.

Problems will be approached both mathematically and graphically where appropriate. The graphs will be included in the report.

Analysis of system/plant/controller/feedback transfers functions for open and closed loop control systems will be demonstrated including the use of step response plots and root locus plots.

The concept of reducing transfer functions by analyzing the dominant roots is looked at and presented through the use of S-plane and Root locus plots.

2 Approach

Firstly a project folder was created, to manage the workspace for the report and the matlab scripts. The necessary files were then added to the project folder including a title page and LaTex template. The files were then added to the git repository. The git repository was then pushed to github.

The report was written in LaTex using TeX Live 2019/Debian. The –shell-escape flag must be used when compiling the report to give the latex script access to shell commands. This allows the latex script to run matlab's scripts from the terminal and input tikz pictures correctly .

The project required the use of matlab's control toolbox and matlab2tikz. The control toolbox and matlab2tikz were installed using the matlab package manager.

The matlab scripts were then written to produce the graphs required for the report. The matlab scripts were then integrated into the latex document using the matlab2tikz package.

Three primary matlab commands were used to create the graphs. step,rlocus and bode were used to create step response, root locus s-plane plots and bode diagrams respectively.the function matlab2tikz was then used to create a .tex file, to be included in the document.

The workspace as as above allows for any changes in the matlabs code, to be reflected in the report at run time, including automatically updating the graphs and appendix.

All figures, graphs, equations and tables were numbered, referenced and hyperlinked in the report, for easy viewing. A contents page was automatically generated, including sections,subsections, appendix and references.

For each exercise, the relevant mathematical notation for each system was written out in the report. The instructions were accurately followed to produce the necessary output. The graphs were included in the report using matlab2tikz. The Matlab code was then linked to the appendix. further comments and explanations were then added including graphs, equations and table were appropriate.

For each Question, the Initial constants were calculated from the given information in the question. Where the second question required simplification, justification and explanation was given. The problems were first solved mathematically then checked using a bode plot to ensure the correct answer was obtained graphically. The final answer was then compared against the initial specification using a step response plot of the closed loop transfer function. finally a bode plot was used to visualize the system and ascertain its bandwidth

3 Results and Discussion

3.1 Exercise 1

$$G_p(s) = \frac{a}{s+20} \tag{1}$$

$$\lim_{s \to 0} (G_p(s)) \tag{2}$$

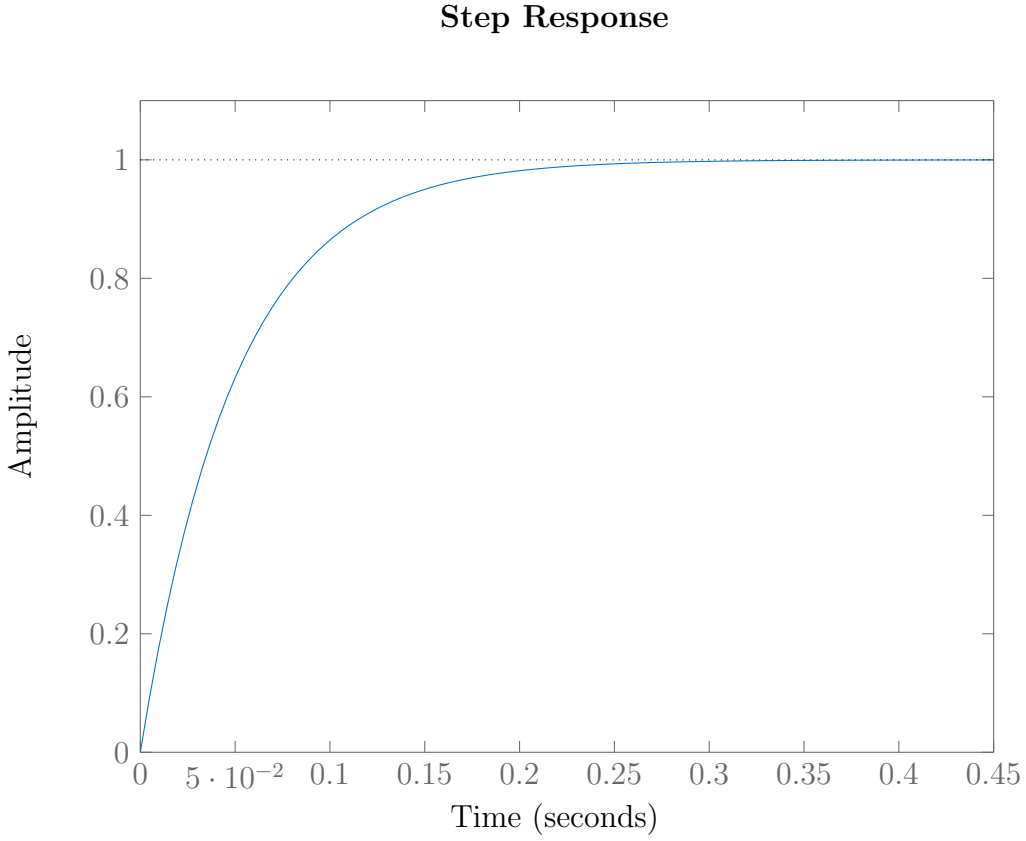

Figure 1: Graph showing step response of plant. This graph was automatically generated from code using matlab (Appendix A.1)

The plant transfer function for a simple first order system is given by (1) where a is the gain of the system and s is the Laplace variable. Steady state is achieved when s approaches 0 as described by (2). Therefore the steady state gain of the system is $a/20$.

When $a = 10$ the steady state gain is $\frac{10}{20}$ which is 0.5.

Unity gain is achieved when the gain is equal to 1 at steady state. Hence the gain of the system is equal to 1 when $a = 20$. The step response of such system is shown in Figure 1.

The time constant is the time taken for the system to reach $1 - e^{-1}$ or approximately 63.2% of its final value. The time constant is equal to the reciprocal of the denominator at steady state. Hence the time constant is equal to 0.05 at all values of a.

3.2 Exercise 2

$$G_p(s) = \frac{a}{s + a} \tag{3}$$

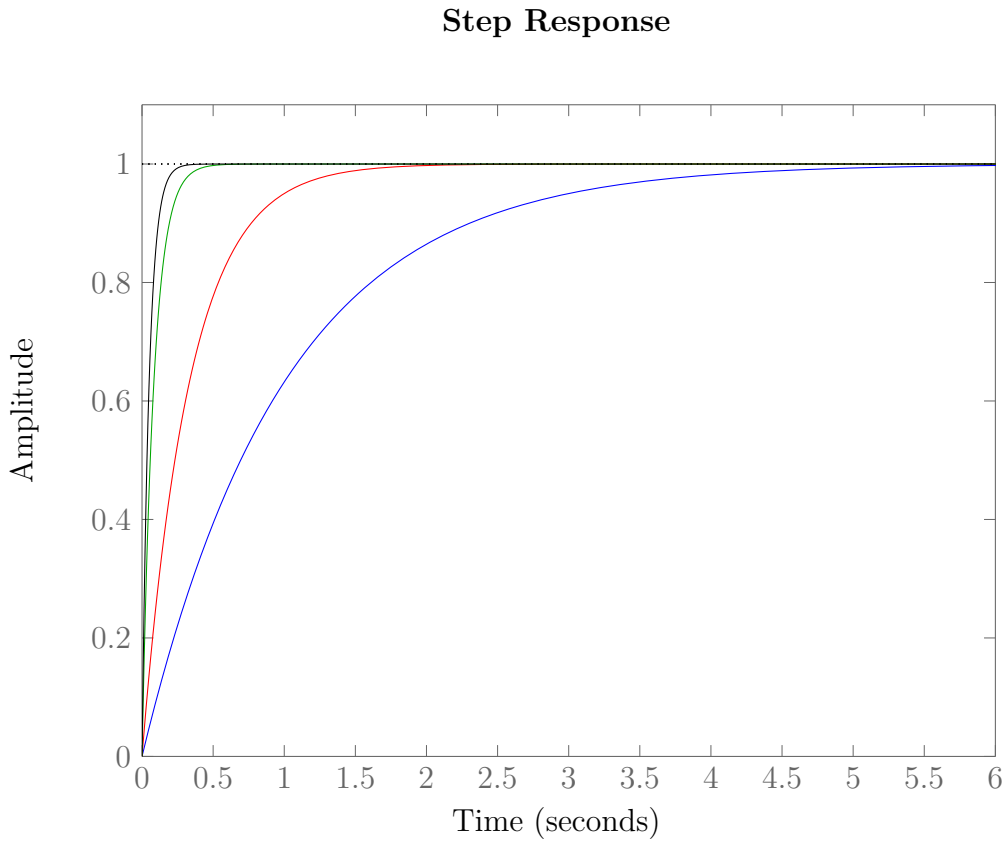

Figure 2: Graph showing step response of the transfer function shown in (3) where a = 1 (blue), 3 (red), 12 (green) and 20 (black). This graph was automatically generated from code using matlab (Appendix A.2)

a[arb]	Time Constant[s]
1	1
3	0.333
12	0.0883
20	0.05

Table 1: Table to show the time constant of the system for different values of a

Figure 2. shows the step response of the system for different values of a. The steady state gain of the system is a/a which is equal to 1 for all values of a. The time constant of the system is shown in Table 1. The time constant is equal to the reciprocal of the denominator at steady state. Hence the time constant is equal to $1/a$ at all values of a.

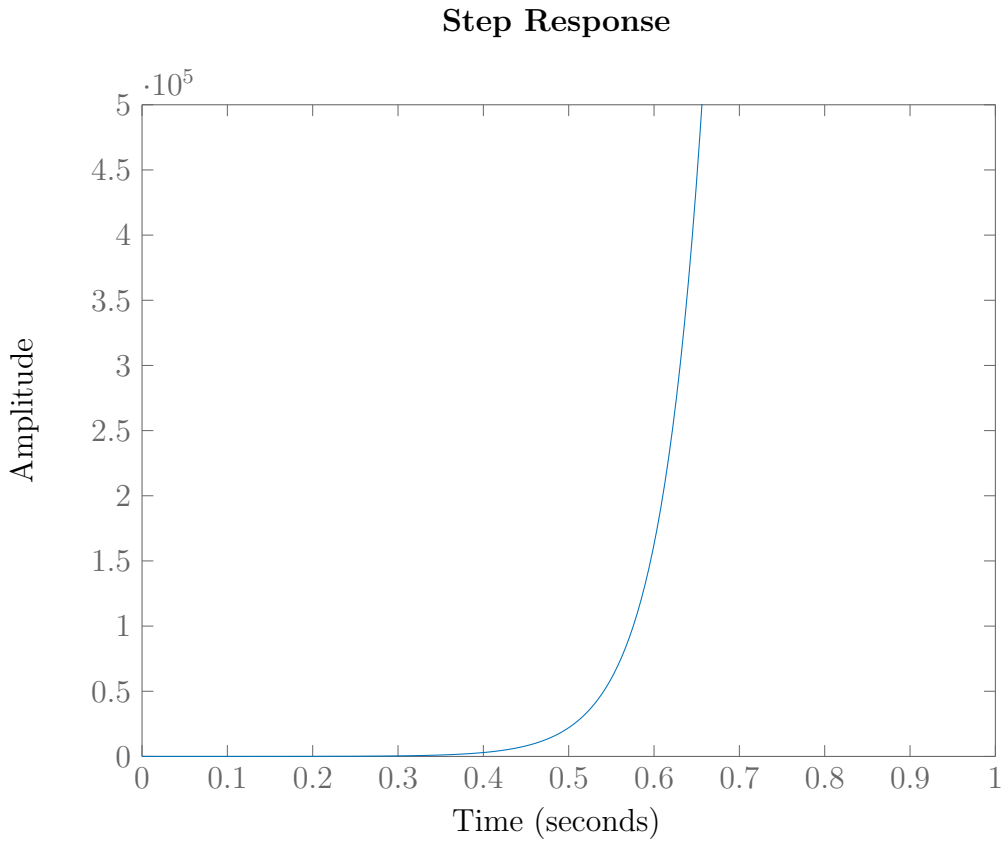

Figure 3: Graph showing the step response of the transfer function shown in (4).This graph was automatically generated from code using matlab (Appendix A.2)

$$G_p(s) = \frac{a}{s-a} \tag{4}$$

Figure 3. shows the step response of the transfer function shown in (4). where the plant has a positive pole the system is unstable and will not converge to a steady state. the rate of change of the system is increasing and will continue to increase until the system is destroyed.

3.3 Exercise 3

$$G_p(s) = \frac{10a}{(s+10)(s+a)} \tag{5}$$

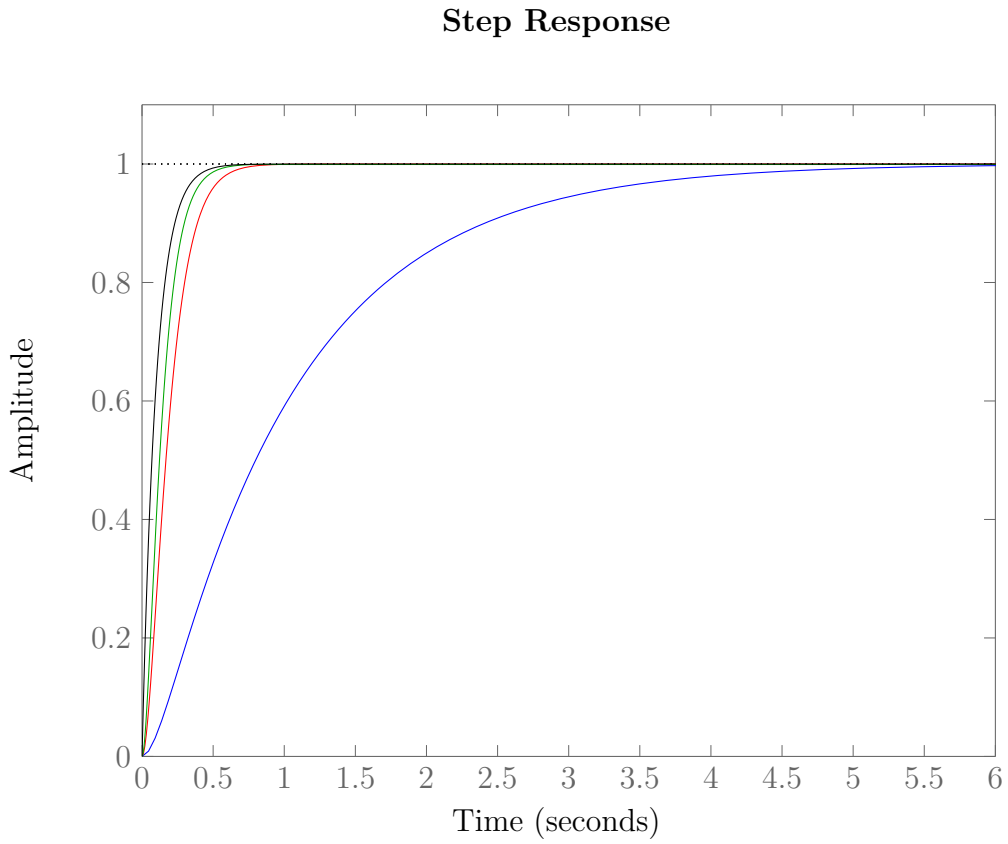

Figure 4: Graph showing step response of the transfer function shown in (5) where a = 1 (blue), 10 (red), 20 (green) and 100 (black).This graph was automatically generated from code using matlab (Appendix A.3)

a[arb]	Time Constant τ[s]
1	1.1
10	0.215
20	0.159
100	0.111

Table 2: Table to show the time constant of the system (5) for different values of a

Figure 4. shows the step response of the system for different values of a. The steady state gain of the system is $10a/10a$ which is equal to 1 for all values of a. The time constant of the system is shown in Table 2. when a is very small, the pole at -a is considered dominant, therefore the time constant approaches the reciprocal of -a. When a is large, the pole at -10 is considered dominant and therefore the time constant is approaches the reciprocal of -10.

3.4 Exercise 4a

$$G_p(s) = \frac{10}{s^2 + as + 10} \tag{6}$$

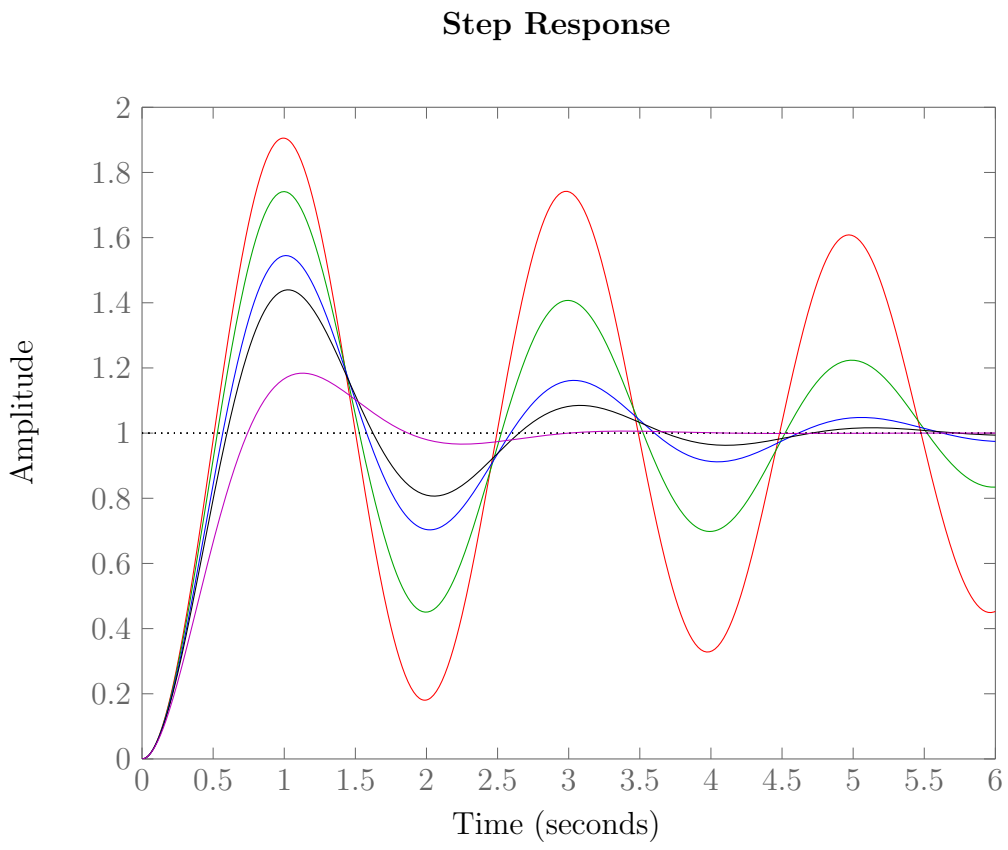

Figure 5: Graph showing step response of the transfer function shown in (6) where a = 0.2 (red), 0.6 (green), 1.2 (blue), 1.6 (black) and 3.0 (magenta). This graph was automatically generated from code using matlab (Appendix A.4)

a[arb]	Percentage Peak Overshoot $M_p[\%]$	damping factor $\xi[arb]$
0.2	91	0.0316
0.6	74	0.9487
1.2	54	0.1897
1.6	44	0.2530
3.0	18	0.4743

Table 3: Table to show the percentage peak overshoot and damping factor of the system (6) for different values of a

$$\begin{aligned} \omega_0 &= \sqrt{10} \\ \therefore \xi &= \frac{a}{2\omega_0} \end{aligned} \tag{7}$$

Eq. (7) shows the working to calculate the damping factor ξ in table 3. The damping factor is the ratio of the damping coefficient to the natural frequency of the system. The damping coefficient is the coefficient of the second order term in the differential equation. The damping coefficient is equal to a in this case. The natural frequency is the square root of the coefficient of the first order term in the differential equation. The coefficient of the first order term ξ is equal to 10 in this case. Therefore the damping factor is equal to $\frac{a}{2\sqrt{10}}$.

Figure 5. shows the step response of the system for different values of a. The steady state gain of the system is 10/10 which is equal to 1 for all values of a. The percentage peak overshoot and damping factor of the system is shown in Table 3. when a is small the percentage peak overshoot approaches 1 and the damping factor is small. When a is large, the percentage peak overshoot is small and the damping factor is large.

The variable a changes the roots of the denominator of the transfer function, separating the roots as a gets larger. As the ratio of imaginary to real parts of the roots increases, the damping factor increases. As the ratio of imaginary to real parts of the roots decreases, the damping factor decreases.

$$\frac{\pi}{\omega_0\sqrt{1-\xi^2}} \tag{8}$$

While the damping factor is less than 1, the system is underdamped. This means the system will oscillate about the steady state value. The time to maximum overshoot

can only be observed when the system is underdamped and is characterized by the equation (8). when the damping factor is small, the time to maximum overshoot approaches a constant value. When the damping value approaches 1, the time to maximum overshoot approaches infinity.

3.5 Exercise 4b

$$G_p(s) = \frac{100a}{(s^2 + 12s + 100)(s + a)} \tag{9}$$

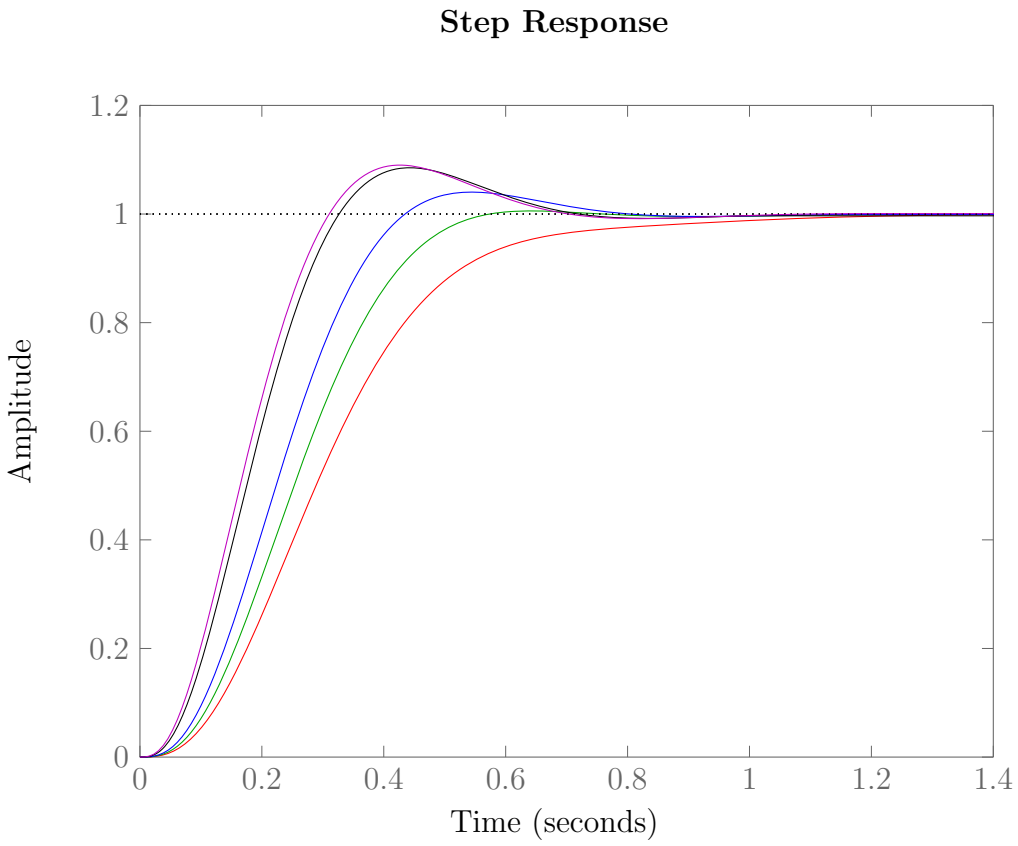

Figure 6: Graph showing step response of the transfer function shown in (9) where a = 5 (red), 7 (green), 10 (blue), 25 (black) and 35 (magenta).This graph was automatically generated from code using matlab (Appendix A.5)

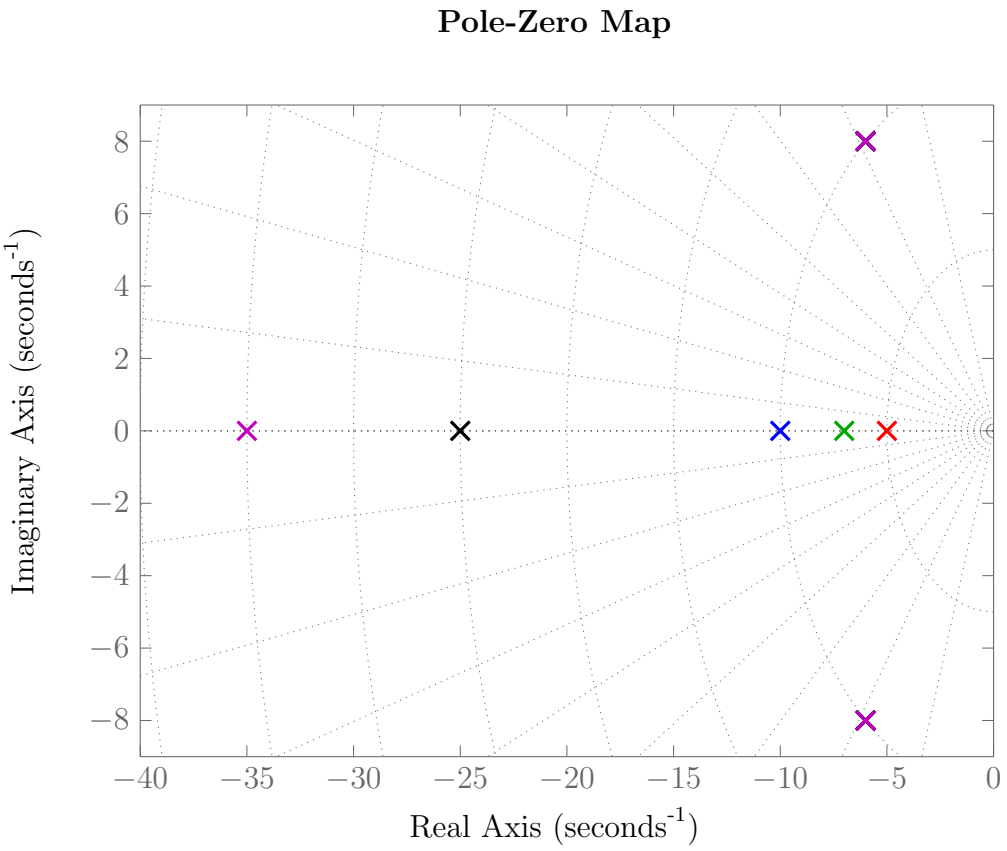

Figure 7: Graph showing Pole-Zero Map of the transfer function shown in (9) where a = 5 (red), 7 (green), 10 (blue), 25 (black) and 35 (magenta), The poles at -6$\pm j8$ exist for all values of a. This graph was automatically generated from code using matlab (Appendix A.5)

Figure 6. shows the step response of the system (9) for different values of a. The steady state gain of the system is $100a/100a$ which is equal to 1 for all values of a. The poles of the system are shown in Figure 7. The poles at -6$\pm j8$ exist for all values of a. The poles at -a and -6$\pm j8$ are considered dominant for small and large values of a respectively.

When a is 35, the poles at -6$\pm j8$ are considered almost completely dominant and therefore the graph looks like an underdamped third order system. When a is 5, the pole at -a is slightly more dominant over the poles at -6$\pm j8$ therefore the graph looks like a critically damped second order system. The extent to which the systems order can be approximated, can be visualized with the help of the pole-zero map shown in Figure 7. The ratio between real parts of each pole determines how dominant it

is, where the smaller the ratio the more dominant the pole is.

3.6 Exercise 5

$$G_p(s) = \frac{10}{s+10} \tag{10}$$

$$\begin{aligned} G(s) &= G_p(s) \cdot G_c(s) \\ &= \frac{k \cdot 10}{s+10} \end{aligned} \tag{11}$$

where: $G_p(s) = \frac{10}{s+10}$
$G_c(s) = \text{k}$

$$\begin{aligned} G(s) &= G_p(s) \cdot G_c(s) \\ &= \frac{10}{s+10} \cdot \frac{k(s+a)}{s} \\ &= \frac{10k(s+205)}{s(s+10)} \end{aligned} \tag{12}$$

where: $G_p(s) = \frac{10}{s+10}$
$G_c(s) = \text{k}$
$a = 205$

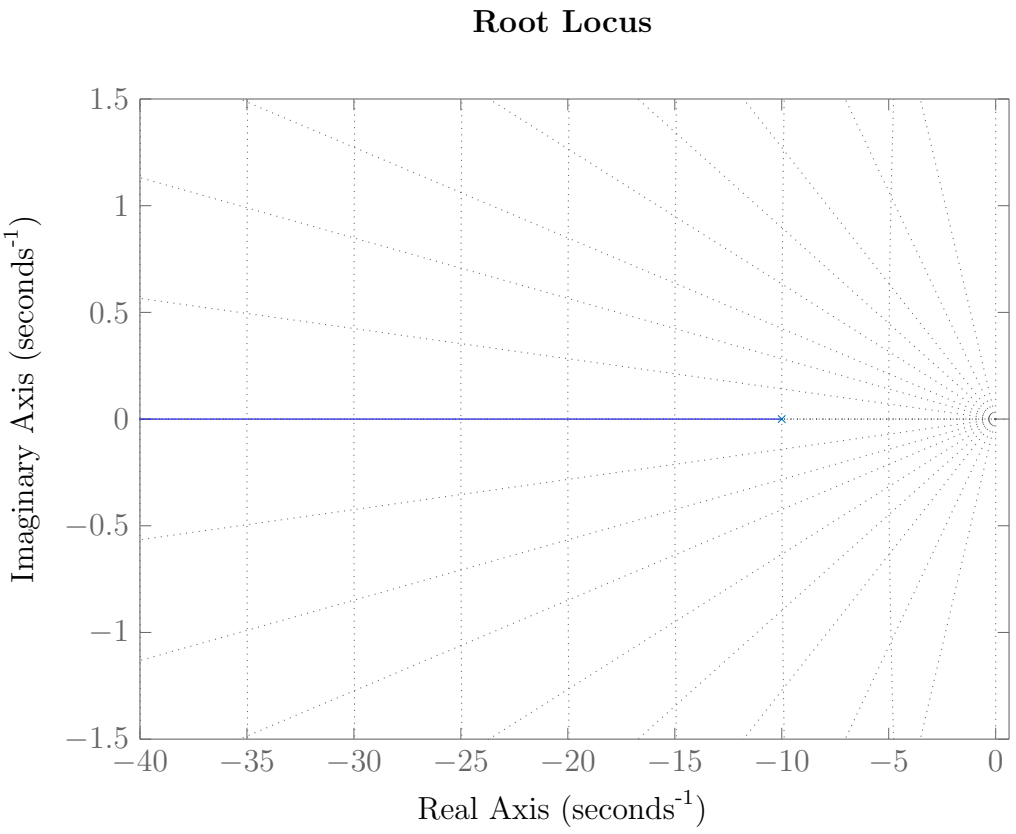

Figure 8: Root Locus plot of the forward path transfer function (13) of a proportional controller applied to plant (10) This graph was automatically generated from code using matlab (Appendix A.6)

Figure 8 only has one pole on the real axis, which is at $s = -10$. This means that the system is stable. The root locus plot shows that the system is stable for all positive values of k. However it does not pass through out design point at $-200 \pm j200$ and does not meet the design specification.

Therefore in order to meet the design specification we need a second order controller for example a proportional integral controller.

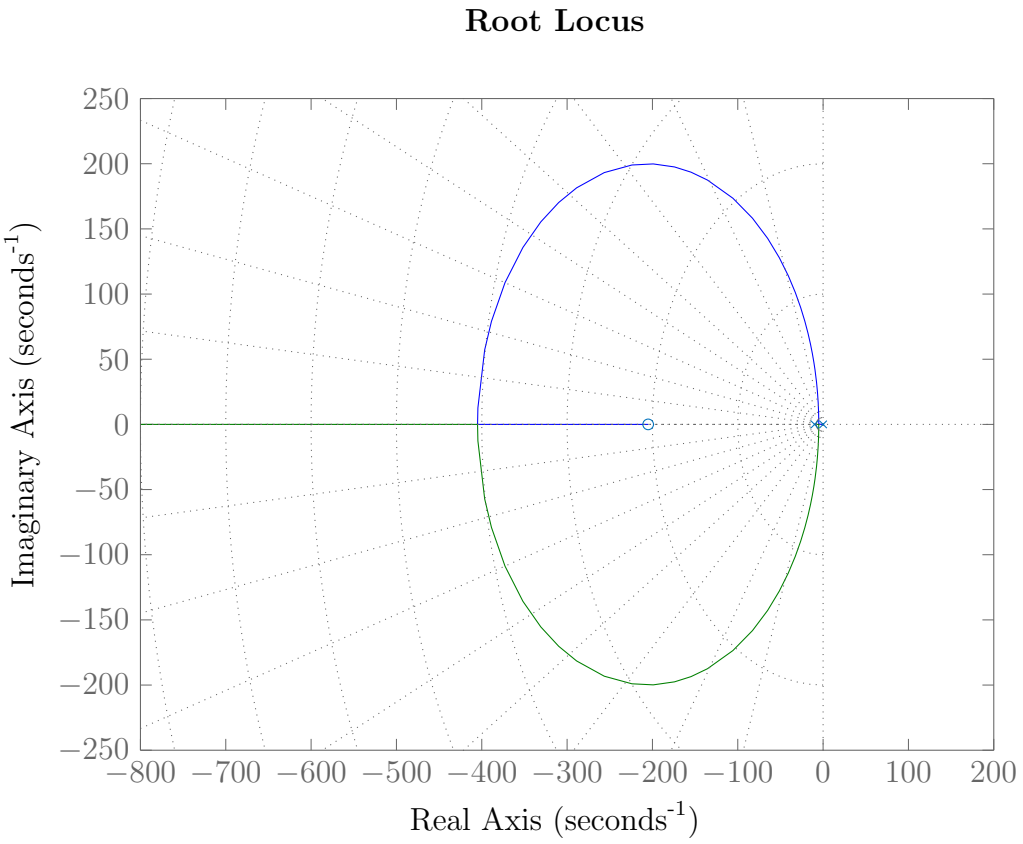

Figure 9: Root Locus plot of the forward path transfer function (13) of a proportional Integral controller applied to plant (10) This graph was automatically generated from code using matlab (Appendix A.6)

Figure 9 shows that the system is stable for all positive values of k and passes through the design point at $-200 \pm j200$, where k = 39.1. This means that the system meets the design specification and is the same value for k as mathematically calculated.

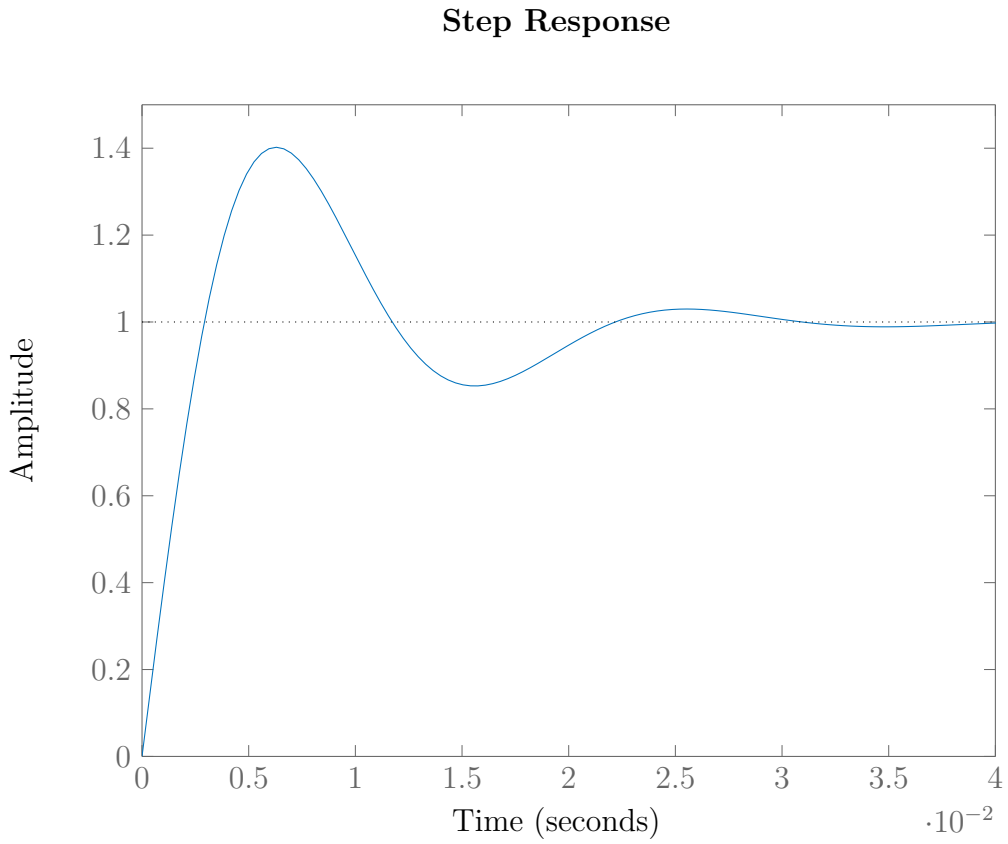

Figure 10: Step Response of the forward path transfer function (13) of a proportional Integral controller applied to plant (10) This graph was automatically generated from code using matlab (Appendix A.6)

$$\begin{aligned} \omega_0 &= \frac{-\ln(\frac{\%}{100})}{t \cdot \xi} \\ &= 8.69338\ rad^s{-1} \end{aligned} \tag{13}$$

where: $\% = 2$
$t \ = 0.75\text{s}$
$\xi \ = 0.6$

Figure 10 shows the step response of the system. The system is stable and the output is as expected. at 20ms the amplitude is 0.948. this is outside of the specified

response of $\pm 2\%$ at 20ms. At 27.3ms the response is within the $\pm 2\%$. This is because the equation (13) used to calculate ω_0 is based off a first order decay. The system is second order therefore is only approximated by this equation.

4 Design Questions and Solutions

4.1 1

$$\begin{aligned} 0 &= s^2 + 2\omega_0\xi s \ + \omega_0^2 \\ \therefore s &= -13.5 \pm j\frac{3\sqrt{19}}{2} \end{aligned} \tag{14}$$

where: $\omega_0 = 15$
$\xi \ = 0.9$

Equation (14) shows the calculation for the approximate position of the dominant poles based on the values of ω_0 and ξ.

$$G_c(s) = \frac{k(s+a)}{s} \qquad G_p(s) = \frac{6}{15s+1} \tag{15}$$

where: k = Constant of proportionality
a = Constant of integration

$$\begin{aligned} 0 &= 1 \ + G_p(s) \cdot G_c(s) \\ &= s^2 + 2\omega_0\xi s \ + \omega_0^2 \\ &= s^2 + \frac{(6k+1)}{15}s + \frac{6ka}{15} \end{aligned} \tag{16}$$

The plant and general model for a PI controller are given by Eq. (15). Setting the denominator of the system to zero, we re-arrange the controller to fit the general equation Eq. (16).

$$\begin{aligned} 2\omega_0\xi &= \frac{(6k+1)}{15} \\ \therefore k &= 67.33 \end{aligned} \tag{17}$$

where: $\omega_0 = 15$
$\xi \ = 0.9$

$$\begin{aligned} \omega_0^2 &= \frac{6ka}{15} \\ \therefore a &= 8.354 \end{aligned} \tag{18}$$

where: $\omega_0 = 15$
$k = 67.33$

Equation (17)-(18) shows the calculation to find the constants of proportionality and integration k and a by comparing coefficients using Eq. (16) such that the system meets the specification set out by ω_0 and ξ.

$$G(s) = \frac{6}{15s+1} \cdot \frac{K(s+8.354)}{s} \tag{19}$$

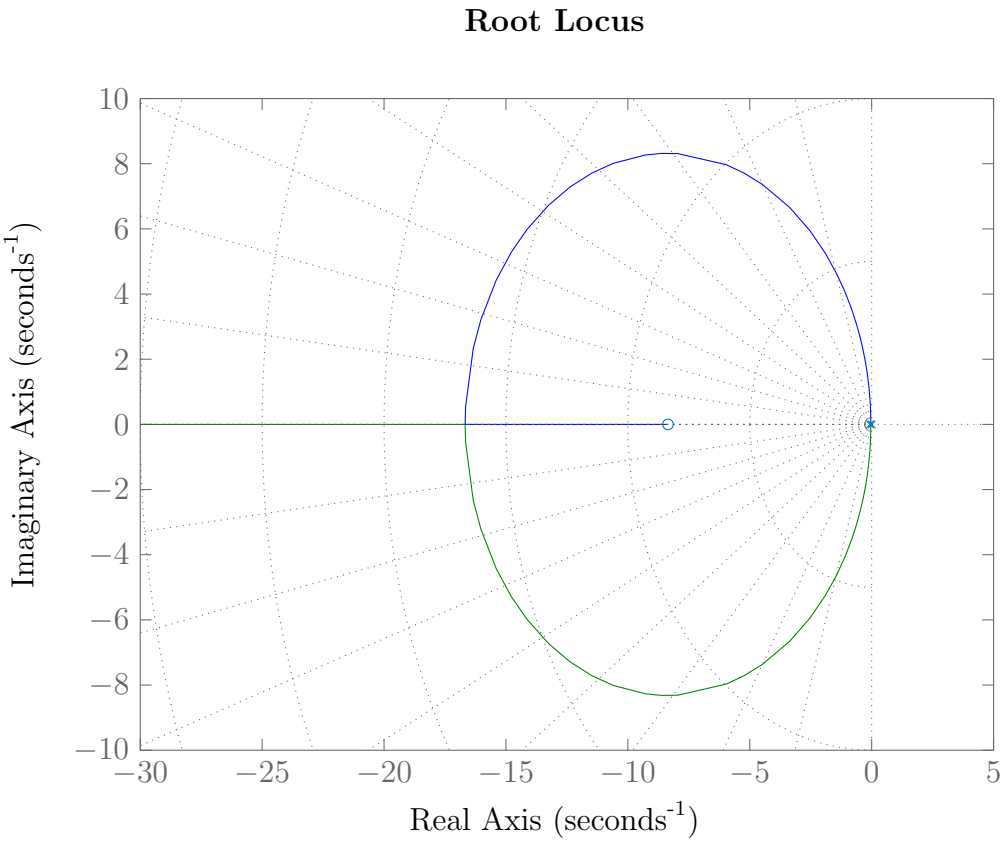

Figure 11: Root Locus plot of the forward path transfer function (19) This graph was automatically generated from code using matlab (Appendix A.7)

Figure 11 shows the root locus plot passing through the poles at $-13.5 \pm j\frac{3\sqrt{19}}{2}$ as found in Eq. (14). The gain at $-13.5 \pm j\frac{3\sqrt{19}}{2}$ is 67.33 confirming the value found in Eq. (17) to be correct.

$$G_c(s) = \frac{67.33\,(s + 8.354)}{s} \tag{20}$$

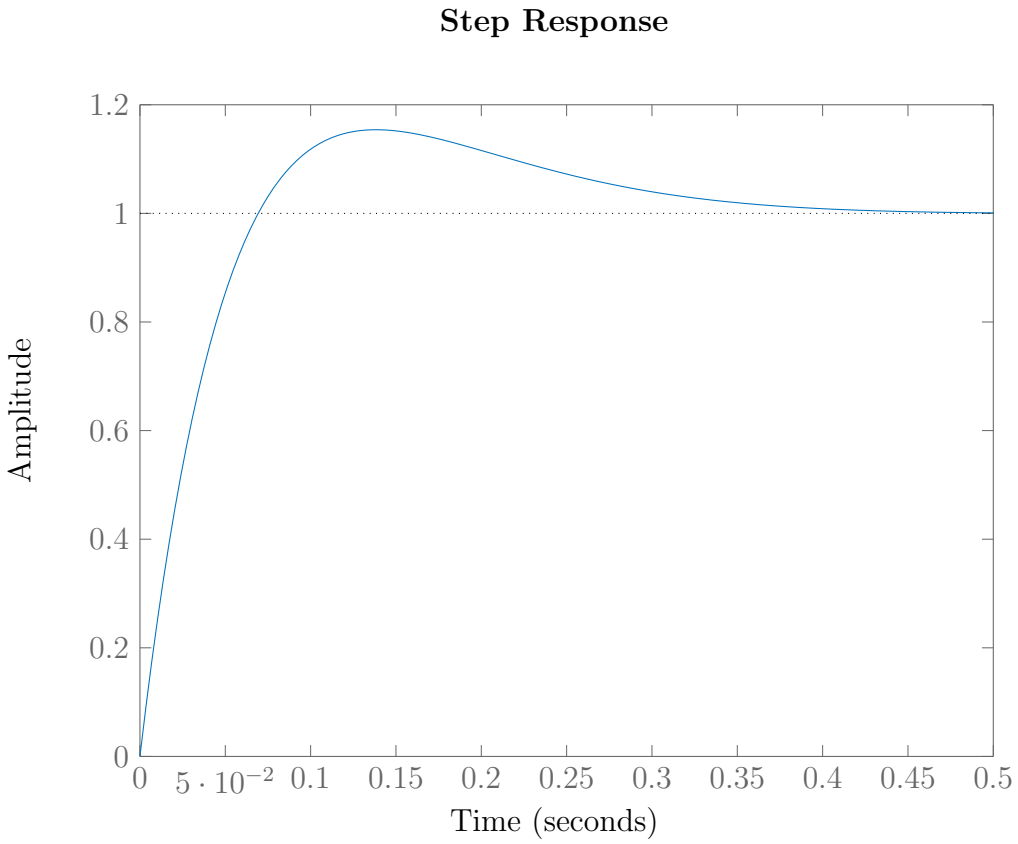

Figure 12: Step Response of the closed loop transfer function where the plant is given by Eq. (15) and controller Eq. (20). This graph was automatically generated from code using matlab (Appendix A.7)

Figure 12 shows the step response of the closed loop transfer function. The step response is shown to be within the specification of $\omega_0 = 15$ and $\xi = 0.9$.

The final solution to this question for the design of a PI controller is given by Eq. (20).

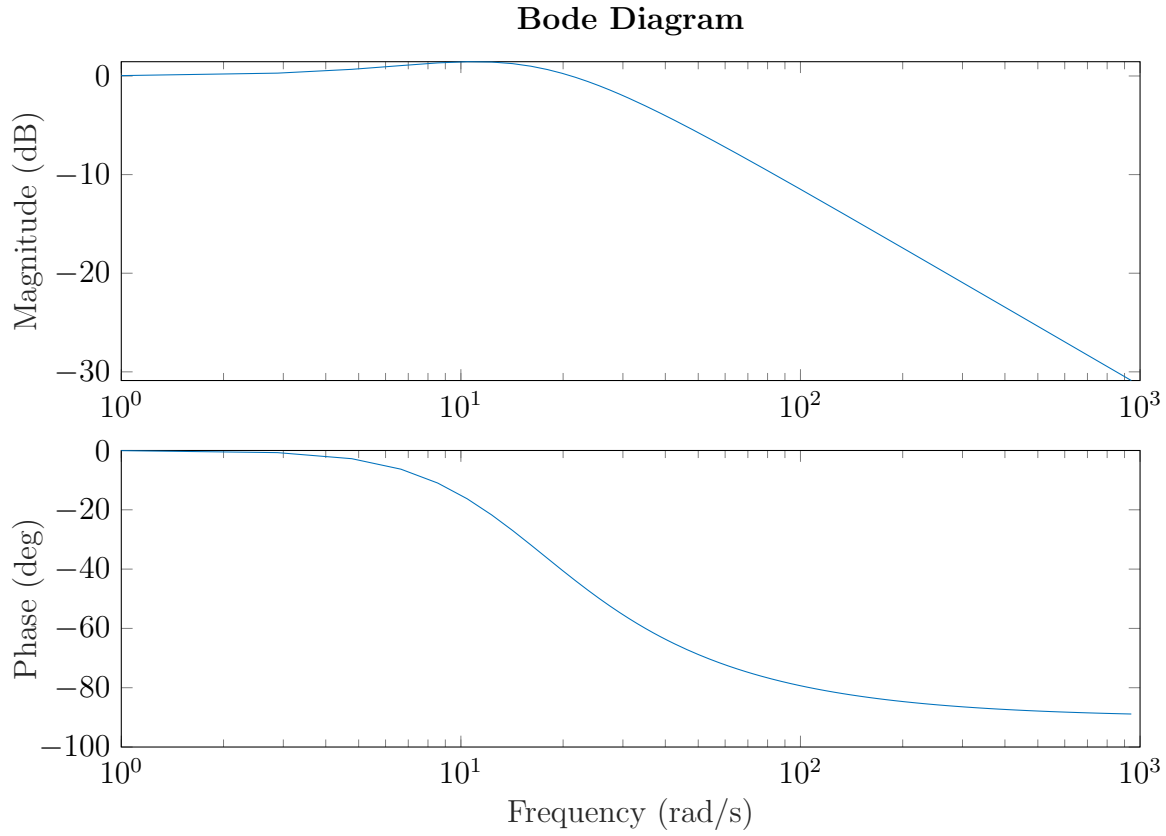

Figure 13: Bode Diagram of the closed loop transfer function where the plant is given by Eq. (15) and controller Eq. (20). This graph was automatically generated from code using matlab (Appendix A.7)

```
fb = bandwidth(CLTF) % calculate bandwidth
```

Figure 14: MATLAB code used to calculate the bandwidth of the closed loop system

Figure 13 shows the Bode Diagram for the closed loop transfer function. The bandwidth of the closed loop system is defined as when the gain is above -3dB. The bandwidth is calculated on matlabs using figure 14. The result is 34.785Hz.

4.2 2

$$\begin{aligned}\frac{\theta(s)}{V_{motor}(s)} &= \frac{15}{(s\ +\ 150)(s\ +\ 8)} \\ &= \frac{15/150}{(s\ +\ 8)} \\ &= \frac{0.1}{(s\ +\ 8)}\end{aligned} \tag{21}$$

Equation (21) shows how the transfer function for the motor can be simplified from a third order equation to a second order equation. The root at -150 is several times larger than -8, therefore the root at -8 can be considered dominant and the transfer function can be simplified to a second order equation.

$$\begin{aligned}\omega_0 &= \frac{-\ln(\frac{\%}{100})}{t \cdot \xi} \\ &= 8.69338\ rads^{-1}\end{aligned} \tag{22}$$

where: $\% = 2$
$t\ = 0.75\text{s}$
$\xi\ = 0.6$

Equation (22) shows the calculation to find the natural frequency of the system. The natural frequency is found by using the formula for the natural frequency of a second order system with damping ratio ξ and time constant t.

$$\begin{aligned}0 &= s^2 + 2\omega_0\xi s\ + \omega_0^2 \\ \therefore s &= -5.216028 \pm j6.954704\end{aligned} \tag{23}$$

where: $\omega_0 = 8.69338$
$\xi\ = 0.6$

Equation (23) shows the calculation for the approximate position of the dominant poles based on the values of ω_0 and ξ.

$$G_c(s) = \frac{k(s+a)}{s} \qquad G_p(s) = \frac{0.1}{s+8} \tag{24}$$

$$\begin{aligned} 0 &= 1 \ + G_p(s) \cdot G_c(s) \\ &= s^2 + 2\omega_0 \xi s \ + \omega_0^2 \\ &= s^2 \ + (8 + 0.1k)\, s + 0.1ka \end{aligned} \tag{25}$$

The plant and general model for a PI controller are given by Eq. (24). Setting the denominator of the system to zero, we re-arrange the controller to fit the general equation Eq. (25).

$$\begin{aligned} 2\omega_0 \xi &= 8 + 0.1k \\ \therefore k &= 24.32 \end{aligned} \tag{26}$$

where: $\omega_0 = 8.6934$
$\xi \ = 0.6$

$$\begin{aligned} \omega_0^2 &= 0.1ka \\ \therefore a &= 31.0752 \end{aligned} \tag{27}$$

where: $\omega_0 = 8.6934$
$k \ = 24.32$

Equation (26)-(27) shows the calculation to find the constants of proportionality and integration k and a by comparing coefficients using Eq. (25) such that the system meets the specification set out by ω_0 and ξ.

$$\begin{aligned} G_p(s) &= \frac{15}{(s \ + \ 150)\,(s \ + \ 8)} \\ &= \frac{15}{s^2 + 158s + 1200} \end{aligned} \tag{28}$$

$$\begin{aligned} G_c(s) &= \frac{k\,(s+a)}{s} \\ &= \frac{24.32\,(s + 31.075)}{s} \\ &= \frac{24.32s + 755.7}{s} \end{aligned} \tag{29}$$

where: $k = 24.32$
$a = 31.075$

$$\begin{aligned} G(s) &= G_p(s) \cdot G_c(s) \\ &= \frac{364.8 + 11340}{s^3 + 158s^2 + 1200s} \end{aligned} \tag{30}$$

where: $G_p(s) = \left(\frac{15}{s^2+158s+1200}\right)$
$G_c(s) = \left(\frac{24.32s+755.7}{s}\right)$

$$\begin{aligned} H(s) &= \frac{G_p(s) \cdot G_c(s)}{1 + G_p(s) \cdot G_c(s)} \\ &= \frac{364.8s + 11340}{s^3 + 158s^2 + 1565 + 11340} \end{aligned} \tag{31}$$

where: $G_p(s) = \left(\frac{15}{s^2+158s+1200}\right)$
$G_c(s) = \left(\frac{24.32s+755.7}{s}\right)$

Equations (28)-(31) shows the plant, controller, forward path and closed loops transfer functions as designed for the motor system.

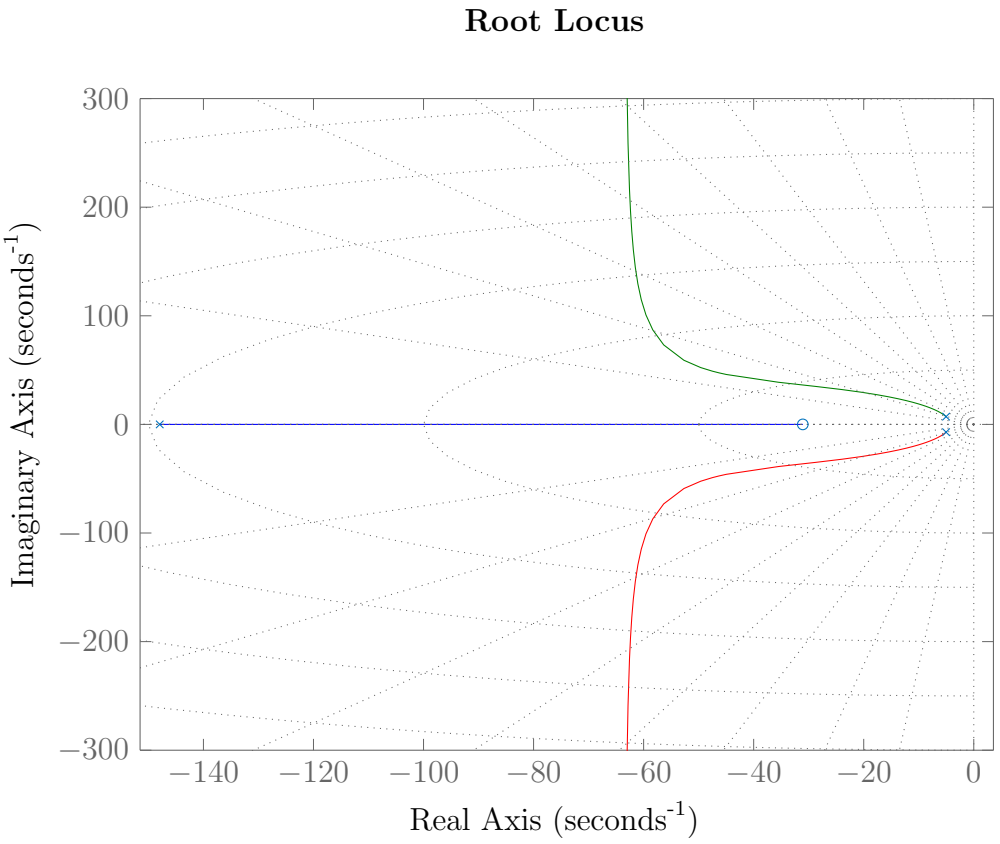

Figure 15: Root Locus plot of the closed loop transfer function (31). This graph was automatically generated from code using matlab (Appendix A.8)

Figure 15 shows the dominant poles passing through -5.03 ± 7.16 which is close to the value of $-5.216028 \pm j6.954704$ calculated in Eq. (23). Therefore the system will behave as designed. The calculations simplifies the third order equation to a second order equation, therefore is only an approximation and we would not expect the roots of the actual system to be exactly the same as the calculated values.

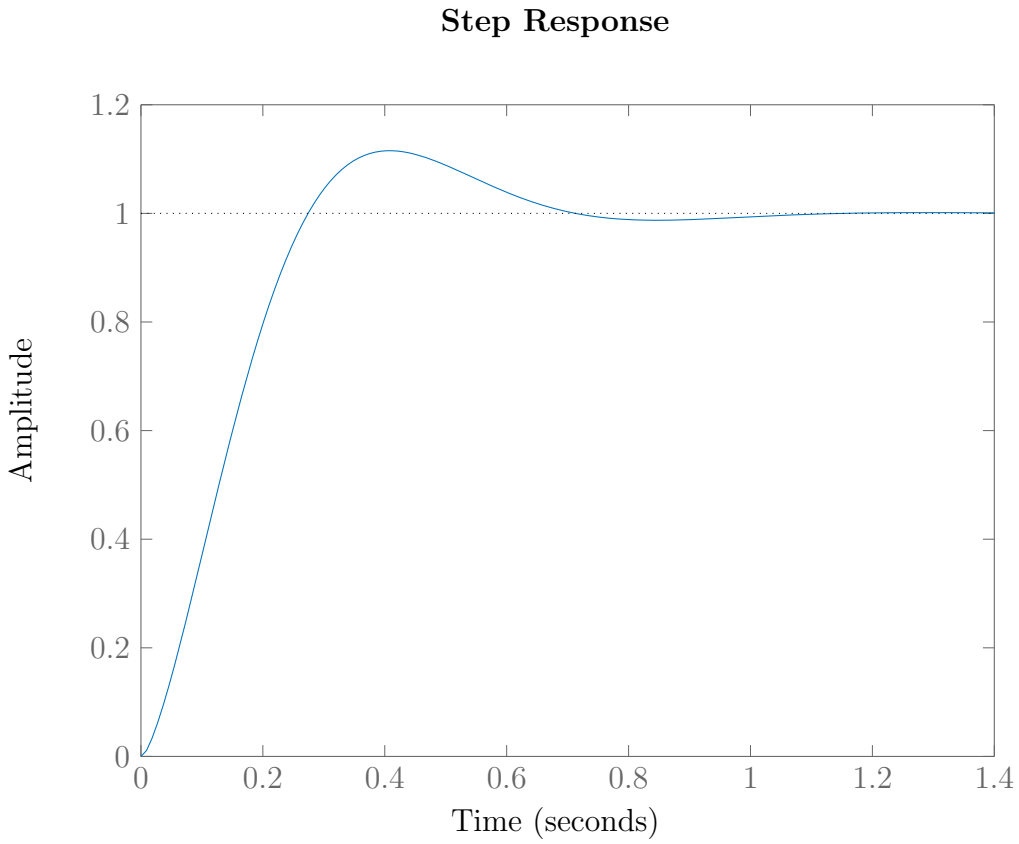

Figure 16: Step Response of the closed loop transfer function (31). This graph was automatically generated from code using matlab (Appendix A.8)

Figure 16 shows the amplitude at 750ms to be 0.993, therefore is within the specification of 2% by 750ms.

5 Summary and Conclusions

The aim of this lab was to complete a set of exercises to demonstrate the understanding of the concepts of control systems. The exercises were based on the material covered in the lectures and the textbook. Five exercises and two questions completed, proving the understanding of the concepts of control systems, and the ability to apply them to a real world problem.

Matlab was used to complete the exercises and design questions. The ability to use Matlab was demonstrated by the through the use of commands and functions required to produce the graphs for this document.

To conclude the lab was completed successfully, and the graphs produced were accurate and correct.

Appendices

A Matlab Code

A.1 Ex1: (3.1)

```
Gp = tf([20], [1 20]); % plant transfer function
step(Gp); % Plot the unity step response
axis([0 0.45 0 1.1]); % Set the axis limits

cleanfigure; % cleanfigure
matlab2tikz('Output/ex1.tex'); % convert to tex
```

A.2 Ex2: (3.2)

```
figure; %new figure
hold on; %hold on to the figure

Gp = tf([1], [1 1]); % plant transfer function
step(Gp, 'b'); % Plot the unity step response

Gp = tf([3], [1 3]); % plant transfer function
step(Gp, 'r'); % Plot the unity step response

Gp = tf([12], [1 12]); % plant transfer function
step(Gp, 'g'); % Plot the unity step response

Gp = tf([20], [1 20]); % plant transfer function
step(Gp, 'k'); % Plot the unity step response

axis([0 6 0 1.1]); % Set the axis limits
cleanfigure; %clean figure
matlab2tikz('Output/ex2.tex'); %convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure; %new figure

Gp = tf([20], [1 -20]); % plant transfer function
step(Gp); % Plot the unity step response

axis([0 1 0 500000]); % change axis
```

```
cleanfigure; %clean figure
matlab2tikz('Output/ex2_2.tex'); %convert to tex
```

A.3 Ex3: (3.3)

```
figure; % new figure
hold on; % hold on to the figure

PlotPlantResponse(1, 'b'); %plot response of dampend plant
PlotPlantResponse(10, 'r'); %plot response of dampend plant
PlotPlantResponse(20, 'g'); %plot response of dampend plant
PlotPlantResponse(1000000, 'k'); %plot response of dampend plant
axis([0 6 0 1.1]); % set axis limits

cleanfigure; % clean figure
matlab2tikz('Output/ex3.tex'); % convert to tex

function [] = PlotPlantResponse(a, colour)
    Gp = tf([10], [1 10]); % define plant
    Temp = tf([a], [1 a]); % this is a/(s+a) for a=1
    G = Gp * Temp; % Multiply them both together
    step(G, colour); % Plot the unity step response
end
```

A.4 Ex4a: (3.4)

```
figure; % new figure
hold on;

PlotPlantResponse(0.2, 'r'); %plot response of dampend plant
PlotPlantResponse(0.6, 'g'); %plot response of dampend plant
PlotPlantResponse(1.2, 'b'); %plot response of dampend plant
PlotPlantResponse(1.6, 'k'); %plot response of dampend plant
PlotPlantResponse(3.0, 'm'); %plot response of dampend plant

cleanfigure; %cleanfigure
matlab2tikz('Output/ex4a.tex'); %convert to tex

function [] = PlotPlantResponse(a, colour)
    time = 0:0.01:6; %time vector
```

```
    Gp = tf([10], [1 a, 10]); % plant transfer function
    step(Gp, time, colour); % Plot the unity step response
end
```

A.5 Ex4b: (3.5)

```
figure; % new figure
hold on; % hold on to the figure

PlotPlantResponse(5, 'r'); % plot response of dampend plant
PlotPlantResponse(7, 'g'); % plot response of dampend plant
PlotPlantResponse(10, 'b'); % plot response of dampend plant
PlotPlantResponse(25, 'k'); % plot response of dampend plant
PlotPlantResponse(35, 'm'); % plot response of dampend plant

cleanfigure; % clean figure
matlab2tikz('Output/ex4b.tex'); % convert to tex

figure; % new figure
hold on; % hold on to the figure
sgrid on; % turn on the grid
pzmapResponse(5, 'r'); % plot response of dampend plant
pzmapResponse(7, 'g'); % plot response of dampend plant
pzmapResponse(10, 'b'); % plot response of dampend plant
pzmapResponse(25, 'k'); % plot response of dampend plant
pzmapResponse(35, 'm'); % plot response of dampend plant
h = findobj(gca, 'type', 'line'); % get all lines
set(h, 'markersize', 10, 'linewidth', 1.2); % set line width
axis([-40 0 -9 9]); % set axis

cleanfigure; % clean figure
matlab2tikz('Output/ex4b_1.tex'); % convert to tex

function [] = PlotPlantResponse(a, colour)
    Gp = tf([100], [1 12 100]);
    Temp = tf([a], [1 a]); % this is a/(s+a) for a=1
    G = Gp * Temp; % Multiply them both together
    step(G, colour); % Plot the unity step response
end

function [] = pzmapResponse(a, colour)
```

```
    Gp = tf([100], [1 12 100]);
    Temp = tf([a], [1 a]); % this is a/(s+a) for a=1
    G = Gp * Temp; % Multiply them both together
    pzplot(G, colour); % Plot the unity step response
end
```

A.6 Ex5: (3.6)

```
Gp = tf([10], [1 10]); % plant transfer function

Kp = 1; % set constatnt of proportonality to 1
Gc = tf([Kp]); % controller transfer function
G = Gc * Gp; % forward path transfer function

rlocus(G); % Plot root locus of forward path transfer function
sgrid; % apply grid of s plane

cleanfigure; % clean figure
matlab2tikz('Output/ex5.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Kp = 1; % set constatnt of proportonality to 1
Ki = 205; % set constatnt of integration to 205

Gc = tf([Kp Kp * Ki], [1 0]); % controller transfer function
G = G * Gc; % forward path transfer function

figure; % create new figure
rlocus(G); % Plot the unity step response
sgrid; % apply grid of s plane

cleanfigure; % clean figure
matlab2tikz('Output/ex5_1.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Kp = 39.1; % set constatnt of proportonality to 38.9
Ki = 205; % set constatnt of integration to 205

Gc = tf([Kp Kp * Ki], [1 0]); % controller transfer function
```

```
G = G * Gc; % forward path transfer function
CLTF = feedback(G, 1) % closed loop transfer function

figure; % create new figure
step(CLTF); % Plot the unity step response

cleanfigure; % clean figure
matlab2tikz('Output/ex5_2.tex'); % convert to tex
```

A.7 Q4.1: (4.1)

```
Gp = tf([6], [15 1]); % plant transfer function

Kp = 1; % set constatnt of proportonality to 1
Ki = 8.354; % set constatnt of integration to 8.354

Gc = tf([Kp * 1 Kp * Ki], [1 0]); % controller transfer function
G = Gp * Gc; % forward path transfer function

figure; % create new figure
rlocus(G); % Plot the unity step response
sgrid; % apply grid of s plane

cleanfigure; % clean figure
matlab2tikz('Output/Q4_1.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Kp = 67.333; % set constatnt of proportonality to 67.333
Ki = 8.354; % set constatnt of integration to 8.354

Gc = tf([Kp * 1 Kp * Ki], [1 0]); % controller transfer function
G = Gp * Gc; % forward path transfer function
CLTF = feedback(G, 1); % closed loop transfer function

figure; % create new figure
step(CLTF); % Plot the unity step response

cleanfigure; % clean figure
matlab2tikz('Output/Q4_1_1.tex'); % convert to tex
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w = linspace(1, 300 * pi, 500); % create frequency vector
[a, b, c] = bode(CLTF, w); % calculate bode plot

figure; % create new figure
set(gcf, 'Position', [105 100 650 200]); % set figure size
semilogx(w, squeeze(b)); % plot magnitude

xlabel('Frequency (rad/s)'); % set x label
ylabel('Phase (deg)'); % set y label

cleanfigure; % clean figure
matlab2tikz('Output/Q4_1_3.tex'); % convert to tex

figure; % create new figure
set(gcf, 'Position', [100 100 650 200]); % set figure size
semilogx(w, mag2db(squeeze(a))); % plot magnitude

title('Bode Diagram'); % set title
ylabel('Magnitude (dB)'); % set y label

cleanfigure; % clean figure
matlab2tikz('Output/Q4_1_2.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fb = bandwidth(CLTF) % calculate bandwidth
```

A.8 Q4.2: (4.2)

```
Gp = tf([15], [1 150]) * tf([1], [1 8]); % plant transfer function

Kp = 24.32; % set constatnt of proportonality to 67.333
Ki = 31.075; % set constatnt of integration to 8.354

Gc = tf([Kp * 1 Kp * Ki], [1 0]) % controller transfer function
G = Gp * Gc; % forward path transfer function
CLTF = feedback(G, 1); % closed loop transfer function

figure; % create new figure
```

```
rlocus(CLTF); % Plot the unity step response
sgrid; % apply grid of s plane

cleanfigure; % clean figure
matlab2tikz('Output/Q4_2.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure; % create new figure
step(CLTF); % Plot the unity step response

cleanfigure; % clean figure
matlab2tikz('Output/Q4_2_1.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w = linspace(1, 300 * pi, 1000); % create frequency vector
[a, b, c] = bode(CLTF, w); % calculate bode plot

figure; % create new figure
set(gcf, 'Position', [105 100 650 200]); % set figure size
semilogx(w, squeeze(b)); % plot magnitude

xlabel('Frequency (rad/s)'); % set x label
ylabel('Phase (deg)'); % set y label

cleanfigure; % clean figure
matlab2tikz('Output/Q4_2_3.tex'); % convert to tex

figure; % create new figure
set(gcf, 'Position', [100 100 650 200]); % set figure size
semilogx(w, mag2db(squeeze(a))); % plot magnitude

title('Bode Diagram'); % set title
ylabel('Magnitude (dB)'); % set y label

cleanfigure; % clean figure
matlab2tikz('Output/Q4_2_2.tex'); % convert to tex

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
fb = bandwidth(CLTF) % calculate bandwidth
```