

INSTRUMENTATION AND CONTROL OF AN AUTOMATED TURRET WEAPON SYSTEM

Ryan Garry - 2024366

Control Systems and Instrumentation – ACDB5011

Wales Institute for Science and Art

University of Wales Trinity Saint David



Abstract

A control system was designed to control a hypothetical Automatic CIWS (Close-In-Weapon-System). This was assigned a response characterised by various randomly generated errors and represented as a standard 2ND Order transfer function in the frequency domain. Transfer Functions for the Open and Closed Loop systems were calculated and simulated through the use of Matlab simulation of a control system utilising a Proportional Integral Differential (PID) controller. Results from the simulation were then presented graphically and the results analysed and presented.

Results showed a slight increase in response time of the system as a result of the introduced controls, with increased stability at frequencies approaching the bounds of the system.

Introduction

In the control of processes of any kind, lag between input of a desired outcome and the resulting response from the system will occur. This can, and most likely will inevitably be introduced to any complex system by a variety of means. Factors such as inefficiencies in the interfaces between real-world physical systems and their controls, and the limitations of the mechanisms by which control of the equipment is measured and exerted, can all contribute to drift in measurement or resulting output.

Control systems must therefore be able to account for a wide variety of multiple simultaneous contributing factors, which in the case of a reasonably complex system, quickly become impossible to address on an individual basis.

For the purposes of designing control systems for complex processes, overall output can be measured and plotted in terms of its Amplitude Gain and Phase Gain. Thereby combining all errors into one complex conjugate that can be graphed either over periodic time intervals, in the case of discrete-time systems, or by continuous measurement in the case of continuous-time systems. This allows suitable systems to be developed theoretically for their effective control, by providing a manner in which feedback can be applied to the system, to produce predictably stable responses as close to the desired outcome as possible.

This report seeks to simulate this process through use of a Matlab mathematical model of a PID (Proportional Integral and Differential) Controller and Process to represent the system and controls, with an aim to producing a sufficient control solution to bring the system under control within 2 seconds, and within 5% overshoot of the desired outcome. Results from this exercise will be validated through the production and analysis of Matlab-produced plots, and calculations carried out alongside Matlab simulation.

To accomplish this, a Transfer function, illustrative of the Gain introduced to the system in the frequency domain in terms of amplitude and phase margin, will be calculated for the Plant process, in this case the angular direction of a CIWS Turret System, and the response of the system recorded. Pole-zero and Bode Plots will be used to determine points of instability within the system. Transfer functions will then be calculated to simulate the behaviour of the PID controller and from that the behaviour of the overall system with feedback incorporated. Further analysis will subsequently be conducted on graphical representations produced through the use of Matlab.

Method

The Forward Transfer Function of the CIWS, pre-control, represents the Gain of the process to be subject to control. This can be modelled in the form per the following transfer function, with values assigned for a, b, and c:

$$G(s) = \frac{1}{as^2 + bs + c}$$

Inputting assigned values for a, b, and c, and incorporating a feedback loop, the system can therefore be represented in block diagram format as a closed loop system with unknown controls as per Figure 1:

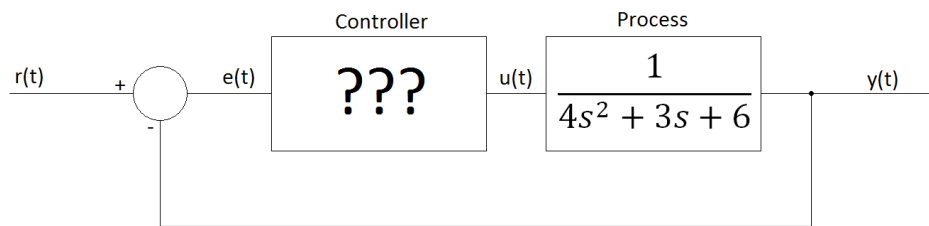


FIGURE 1: BLOC K DIAGRAM OF SYSTEM AND FEEDBACK LOOP

To determine the response produced, Simulink representations of this system was created in Matlab with a Step Input, PID controller and process (represented by a Continuous Transfer function component), with an Oscilloscope to provide measurements at various points of the process as per Figure 2 below:

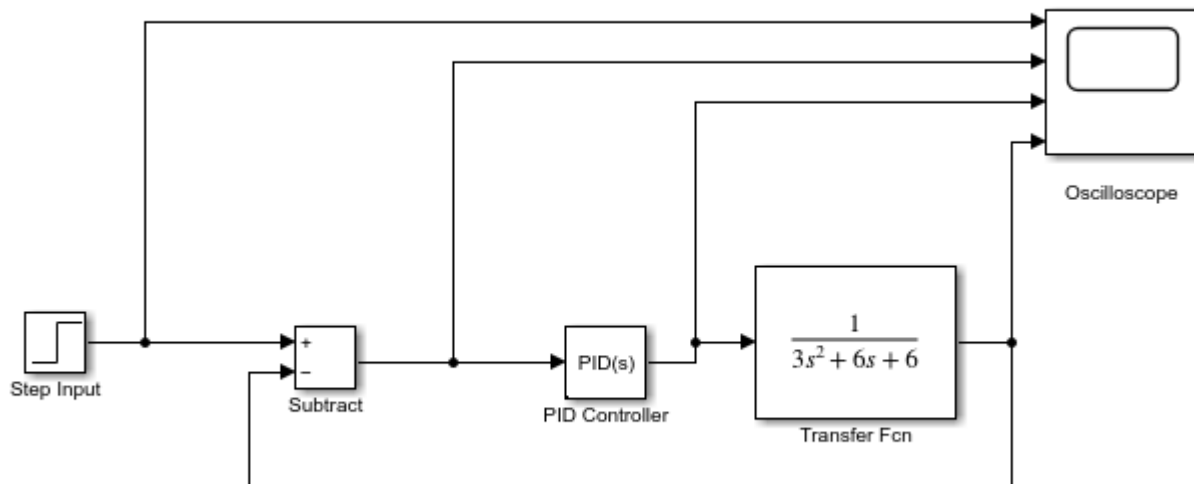


FIGURE 2: CIRCUIT 1 - SIMULINK MODEL OF CIWS CLOSED LOOP SYSTEM

A further Circuit was created for illustrative purposes to display the response of the process without any controls, as per figure 3:

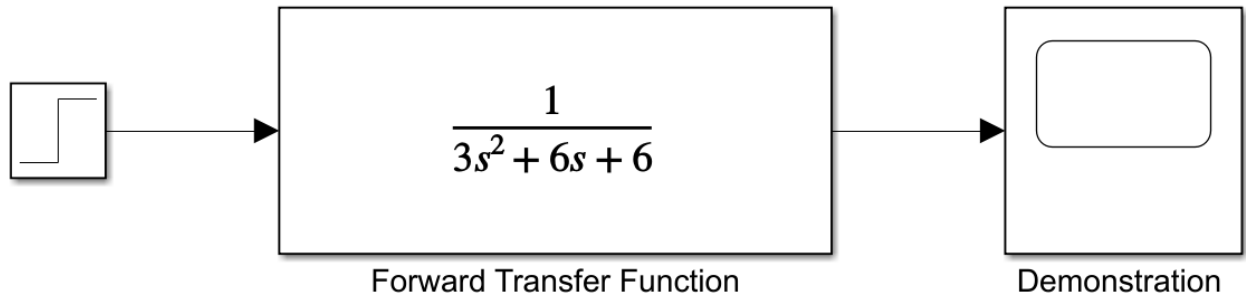


FIGURE 3: UNCONTROLLED SYSTEM

Using the Matlab PID Tuner App, set to the Transfer Function Based tuning method, values were obtained for the PID Proportional (Kp), Integral (Ki), and Derivative (Kd) Responses and Filter coefficient (N) for the controller, as per table 1 below. Figure 3 shows the remaining parameters for the Tuner App. A simulation was run from these values and measurements taken from the oscilloscope to ensure that the controlled response time was within parameters. These are presented below in the Results Section.

TABLE 1: CONTROLLER PARAMETERS GENERATED BY MATLAB

Controller Parameter	Value
Proportional Response (Kp)	142.563721604951
Integral Response (Ki)	64.1103840053898
Dervative Response (Kd)	58.1132317796486
Filter Coefficient (N)	2227.18194606262

The transfer function of the PID controller (H) was then calculated using the following formula. This was multiplied by the transfer function of the Process (G), to give an Open Loop Transfer Function. The mathematical working for this and other work can be found in Annex A.

$$K_p + K_i \frac{1}{s} + K_d \frac{s}{\frac{1}{N}s + 1}$$

EQUATION 1: PID TRANSFER FUNCTION (H)

Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: **PID** Form: **Parallel**

Time domain:

☒ Continuous-time

☐ Discrete-time

Discrete-time settings

Sample time (-1 for inherited): **-1**

Compensator formula

$$P + I \frac{1}{s} + D \frac{N}{1 + N \frac{1}{s}}$$

Main Initialization Output Saturation Data Types State Attributes

Controller parameters

Source: **internal**

Proportional (P): **142.563721604951**

Integral (I): **64.1103840053898**

Derivative (D): **58.1132317796486**

☒ Use filtered derivative

Filter coefficient (N): **2227.18194606262**

Automated tuning

Select tuning method: **Transfer Function Based (PID Tuner App)** **Tune...**

☒ Enable zero-crossing detection

OK **Cancel** **Help** **Apply**

FIGURE 4: PID TUNER

From these results, and the initial inputs for a, b and c, Matlab code was generated to calculate a forward transfer function for the system's behaviour, as well as open and closed loop transfer functions for the system following the implementation of feedback. Bode Plots, Nyquist Plots and Nichols Charts were generated via Matlab from the figures calculated and will be presented and discussed below. A full transcript of the code used, with explanatory notes are presented in Annex B.

A Whole-System Transfer function was then calculated using the following formula:

$$STF = \frac{G}{1 + GH}$$

EQUATION 2: SYSTEM TRANSFER FUNCTION

The block diagram for this can be seen below in figure 5.

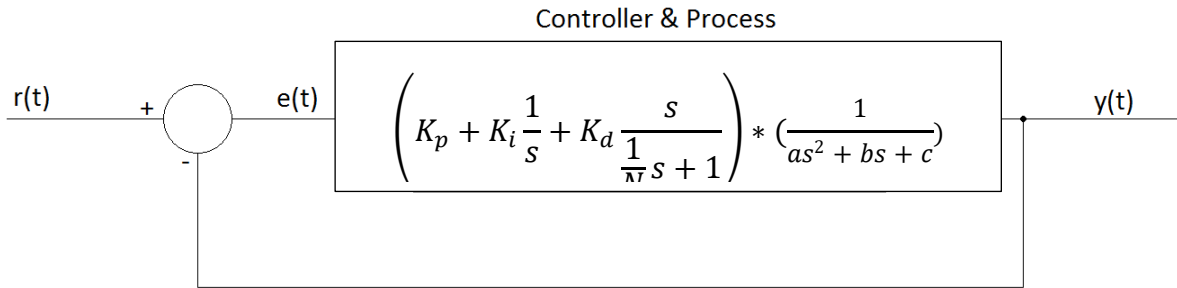


FIGURE 5: BLOCK DIAGRAM OF CONTROLLER AND PROCESS

From these equations, Matlab code was generated in order to validate results, and to generate graphed representations of impulse and step responses of the various transfer functions, as well as Bode Plots, Nyquist Plots and Nichols Charts of system behaviour. These will be presented and discussed later in this document.

A further simulated circuit was also constructed to demonstrate the above as per figure 5 below, with the two additional processes representing the Open Loop response of the system and the Closed Loop response in order to provide validation of results:

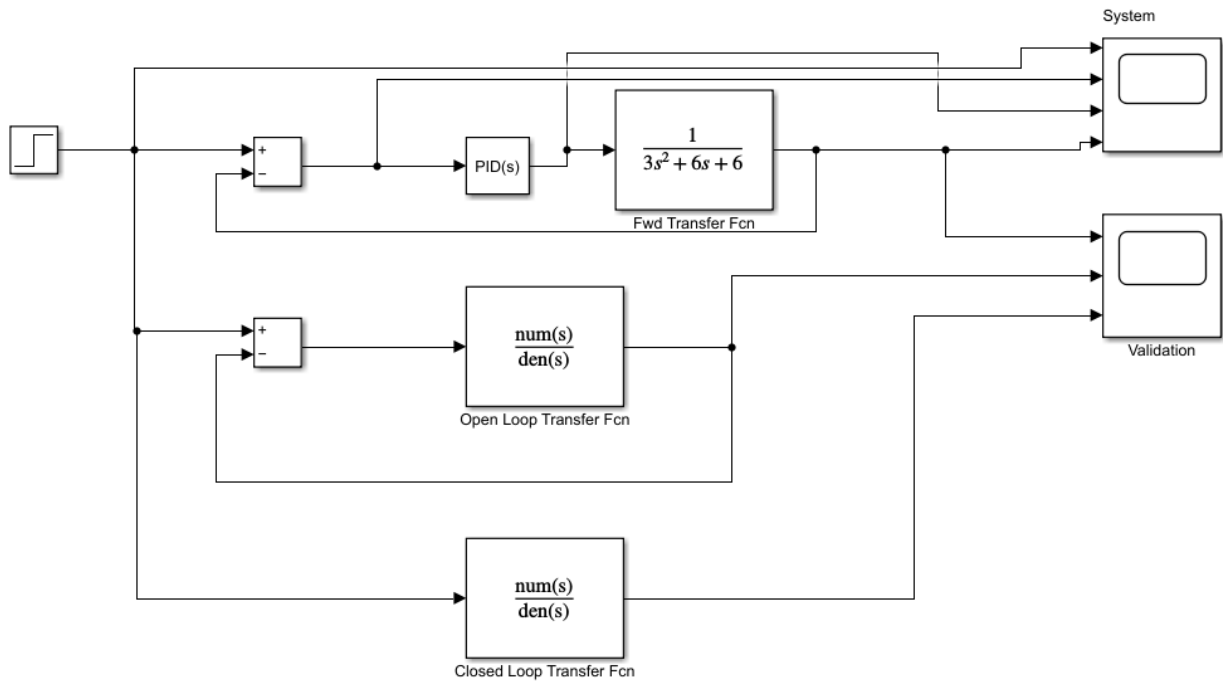


FIGURE 6: CIRCUIT 3 - DEMONSTRATION CIRCUIT

Results

Oscilloscope Readings:
Circuit 1: System Readings

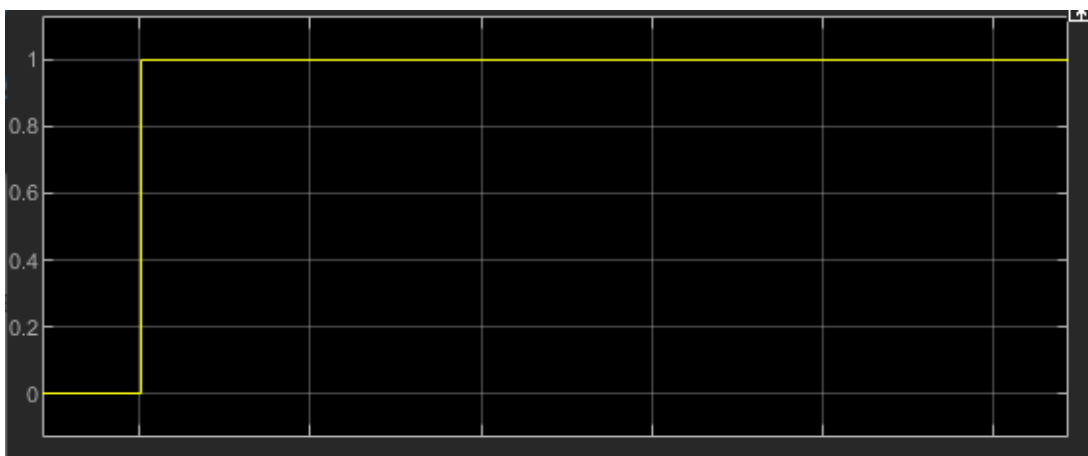


FIGURE 7: STEP INPUT

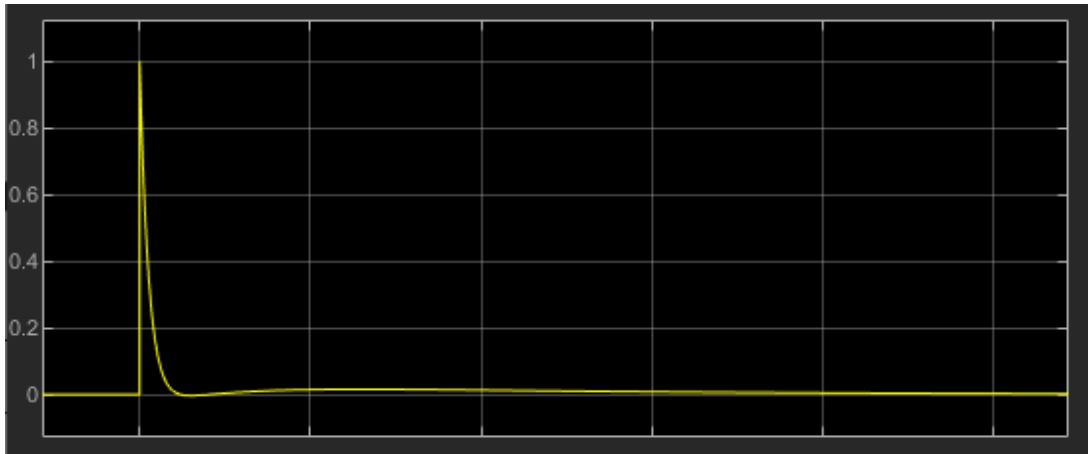


FIGURE 8: ERROR RESPONSE

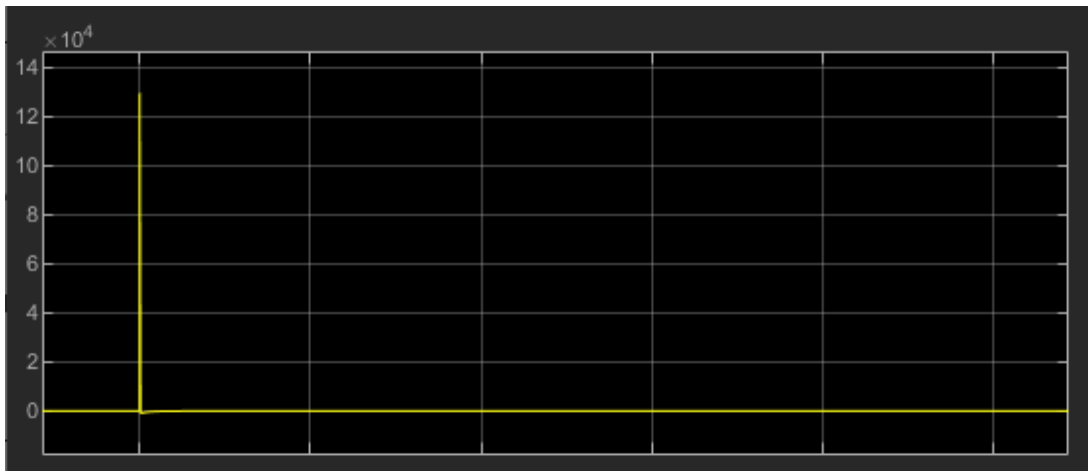


FIGURE 9: PID INPUT

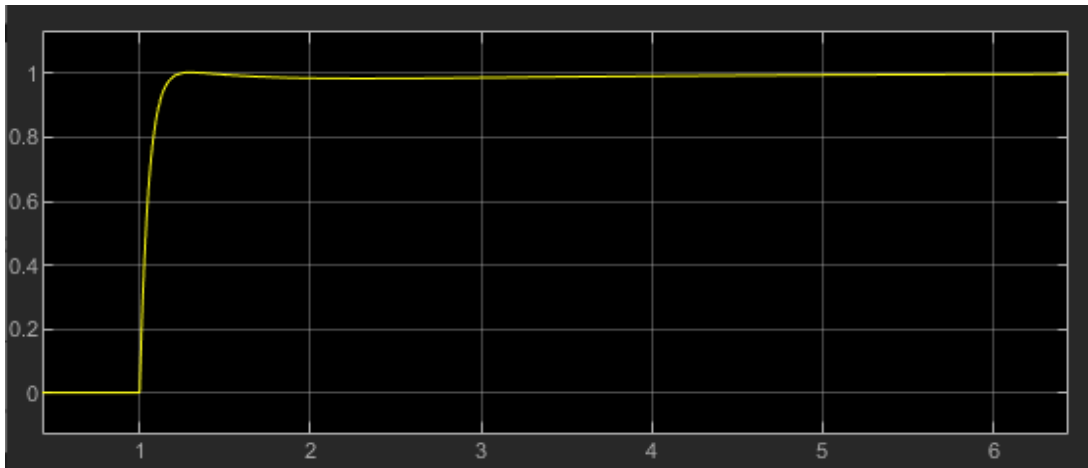


FIGURE 10: SYSTEM RESPONSE

Circuit 2 – Detail of System Responses:

Figures 11 – 14 below shows the responses of the Uncontrolled Plant, Open Loop Transfer function and Calculated Closed-Loop Transfer functions respectively plotted by the oscilloscope.

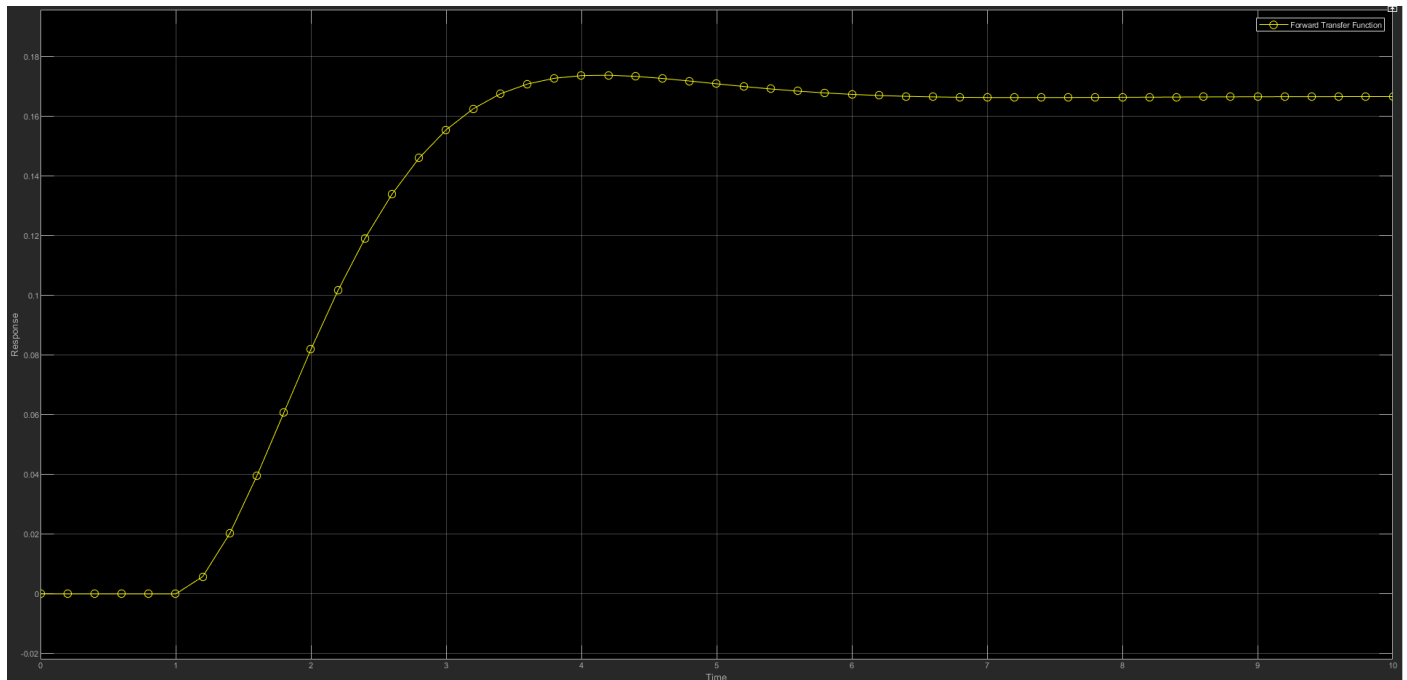


FIGURE 11: FORWARD TRANSFER FUNCTION RESPONSE (NO CONTROLS IMPLEMENTED)

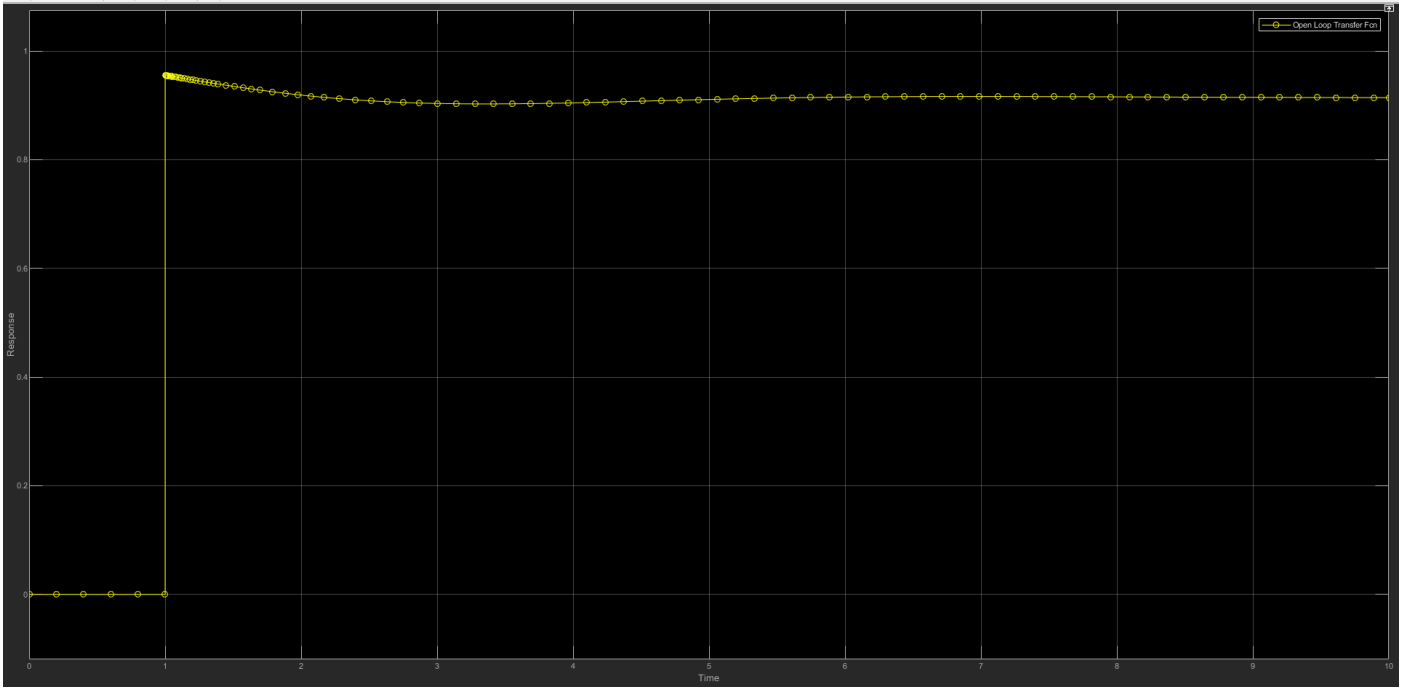


FIGURE 12: OPEN LOOP TRANSFER FUNCTION (CONTROLS WITHOUT FEEDBACK)

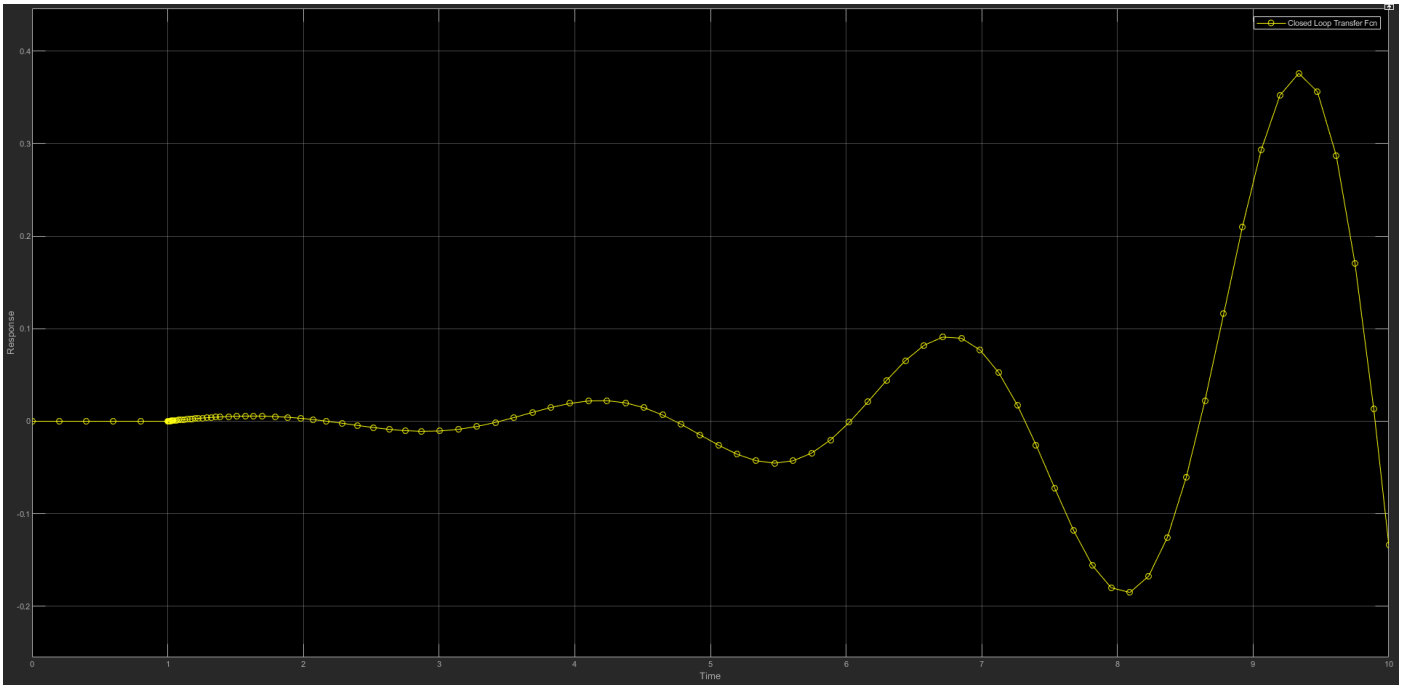


FIGURE 13: CALCULATED CLOSED LOOP TRANSFER FUNCTION – ASSUMED ERROR IN CALCULATIONS

Results of Matlab Analysis

Calculated Pole-Zero Coordinates

Coordinates and gain were calculated in Matlab using the pzmap and tz2pz functions for various system responses. These are presented below and are plotted using various methods later in this section:

Turret (pre-control transfer function)

z =

0

0

p =

-1.0000 + 1.0000i

-1.0000 - 1.0000i

k =

0.3333

Whole-System (forward Transfer function)

Pole-Zero Coordinates:

p =

-1.0000 + 1.0000i

-1.0000 - 1.0000i

z =

0×1 empty double column vector

Closed Loop Transfer Function

z =

-1.8834 + 0.0000i

-1.0010 + 1.0005i

-1.0010 - 1.0005i

-0.5833 + 0.0000i

p =

1.0e+03 *

0.0000 + 0.0000i

-2.2271 + 0.0000i

-0.0010 + 0.0010i

-0.0010 - 0.0010i

-0.0010 + 0.0010i

-0.0010 - 0.0010i

k =

4.3182e+04

Frequency Response Plots

Pole Zero Plots

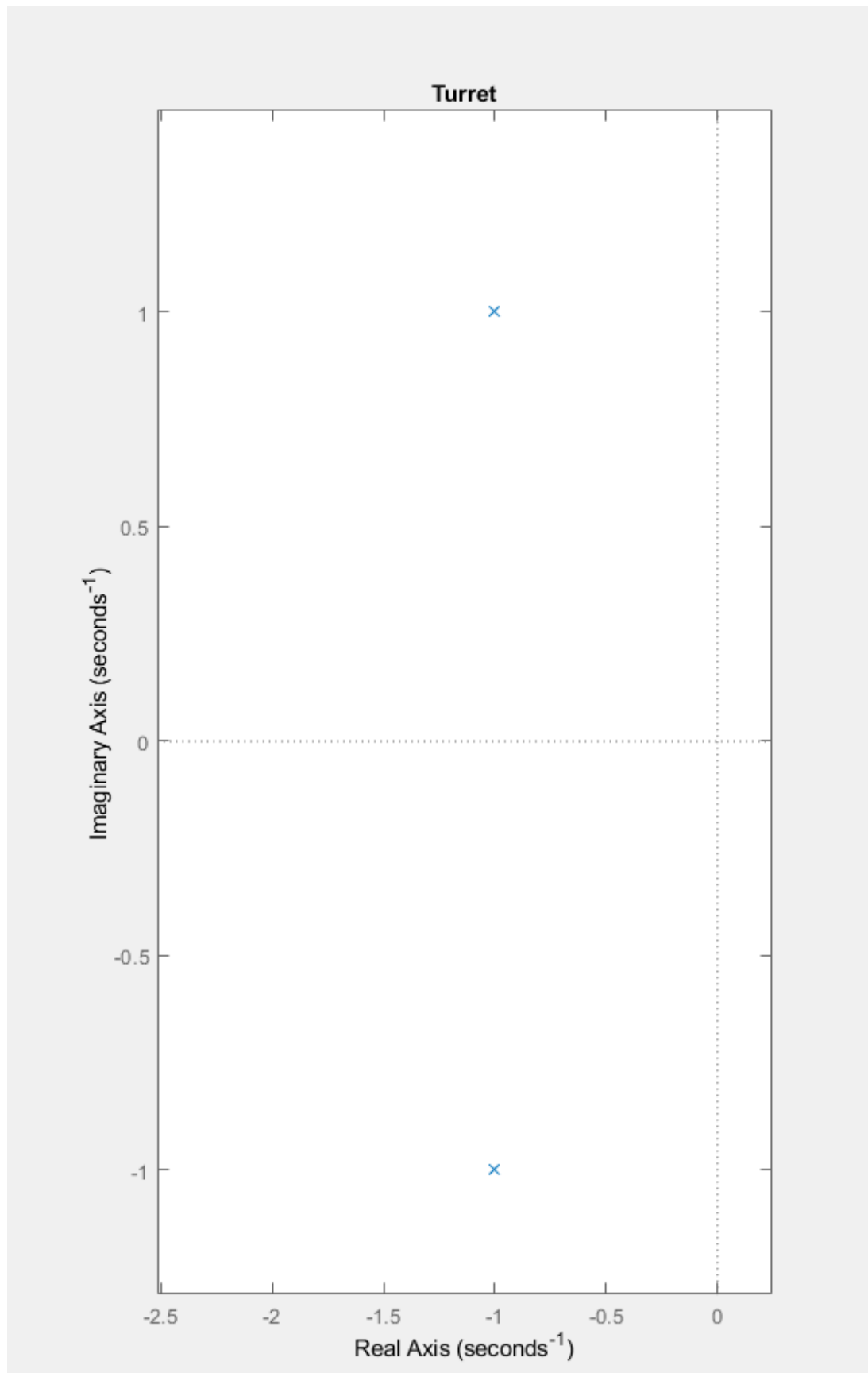


FIGURE 14: POLE-ZERO PLOT OF TURRET RESPONSE

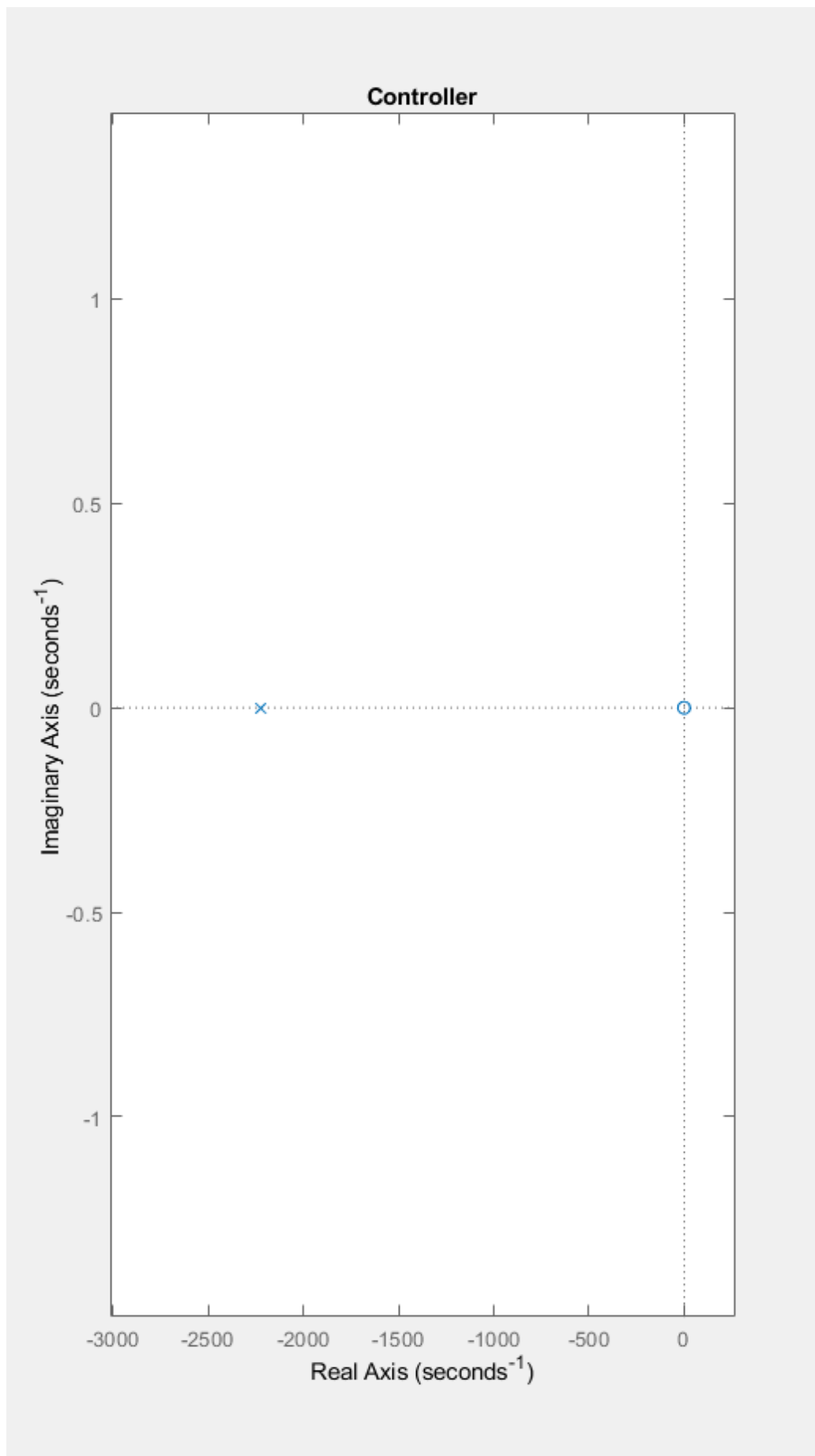


FIGURE 15: POLE-ZERO PLOT OF CONTROLLER RESPONSE

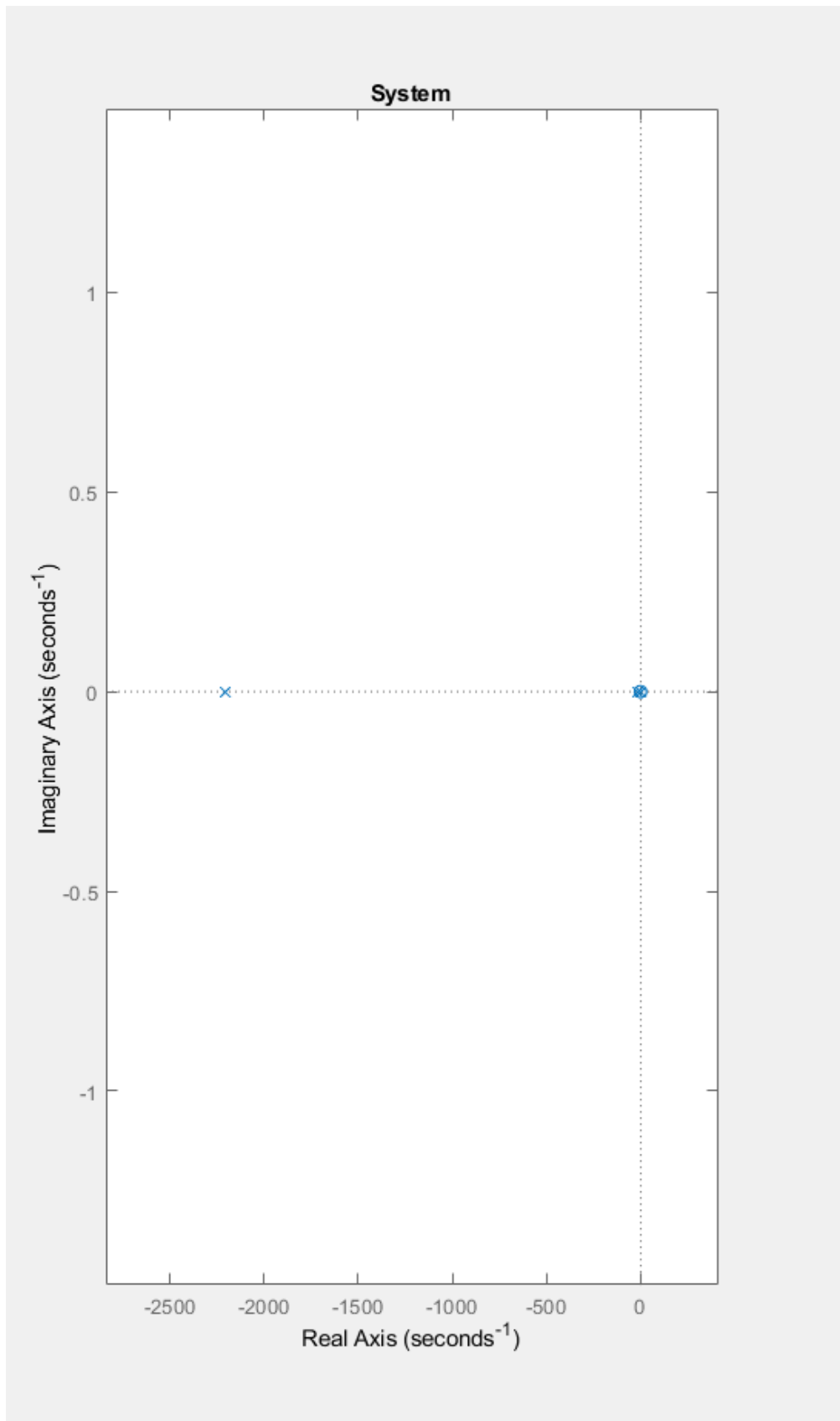


FIGURE 16: POLE-ZERO PLOTS OF SYSTEM RESPONSE

Bode Diagrams

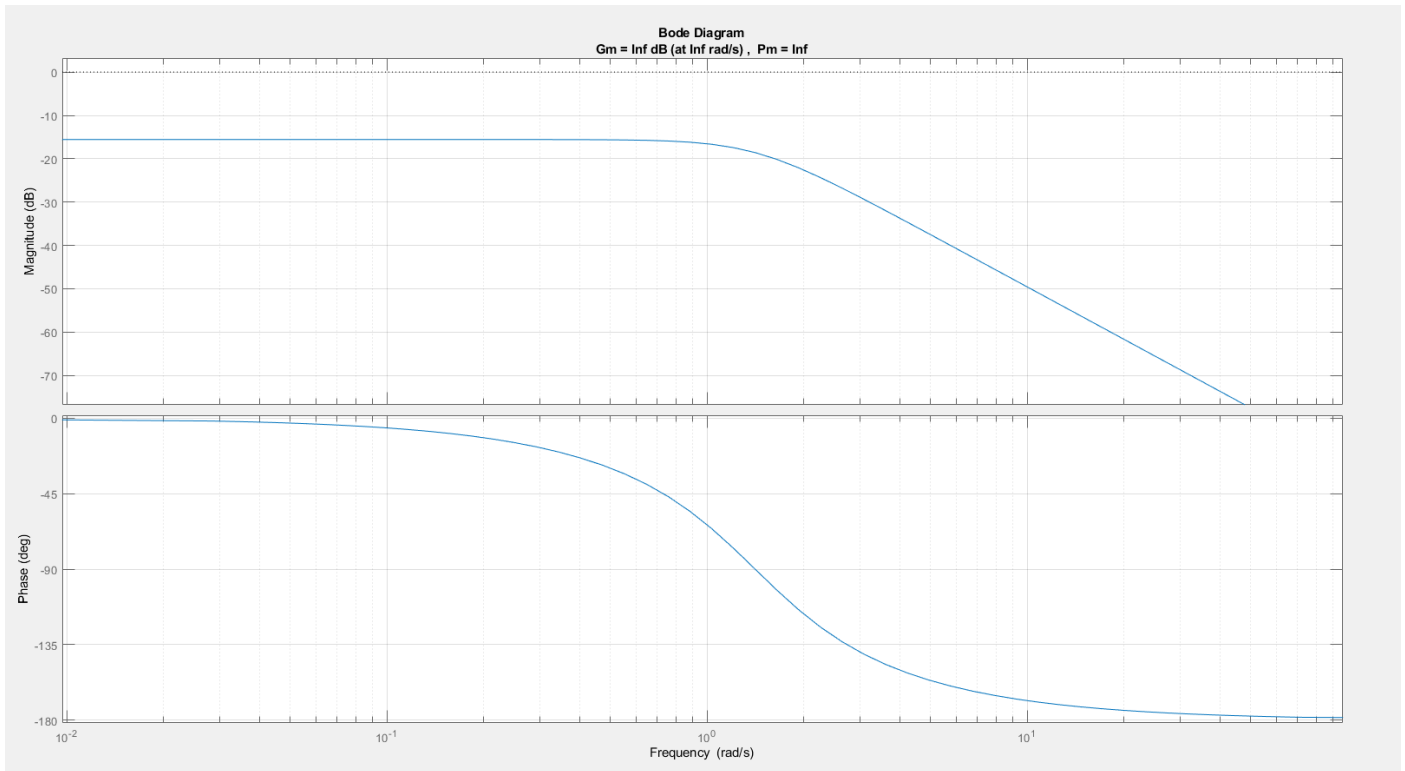


FIGURE 17: BODE PLOT OF UNCONTROLLED SYSTEM RESPONSE

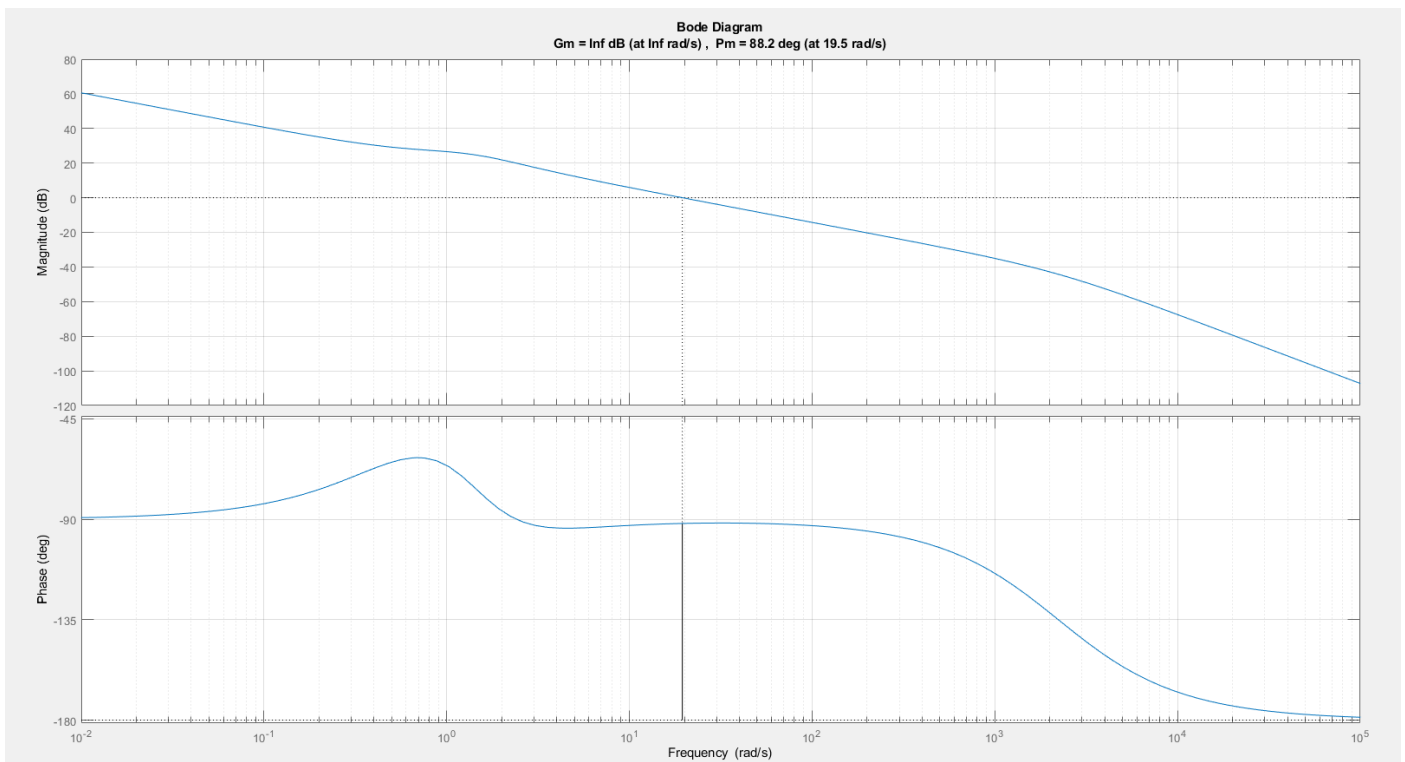


FIGURE 18: BODE PLOT OF WHOLE SYSTEM RESPONSE

Nichols Chart

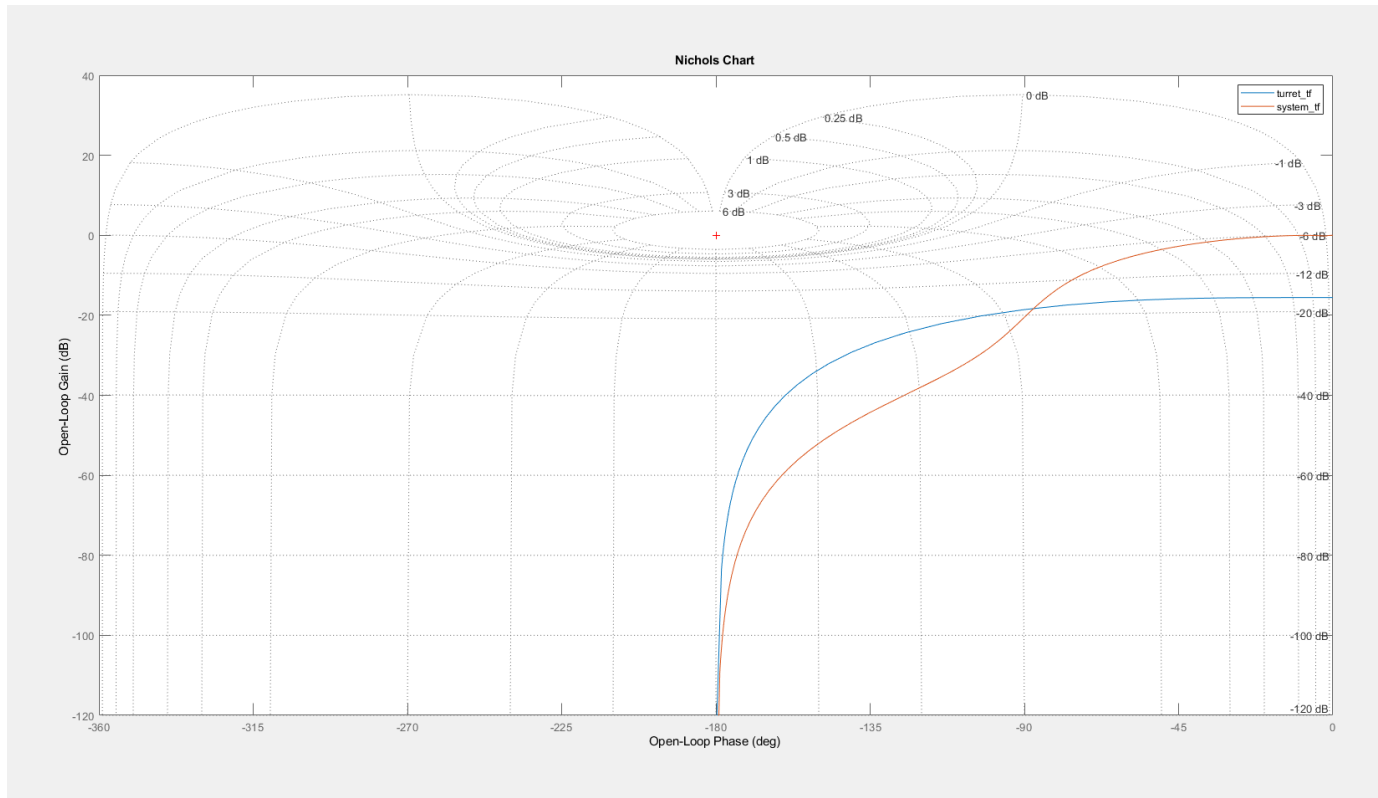


FIGURE 19: NICHOLS PLOT OF TURRET TF AND WHOLE SYSTEM TF

Discussion

System response plots and simulated circuits

The system response plots at figures 8-10 illustrate the behaviour of the system over time, in terms of the error response, PID Input, and the combined system response. As can be seen, as the error response becomes 1 at the point of input change as a result of the systems inability to instantly react. The PID input immediately rises rapidly to a high level, and drops just as rapidly as the error response approaches zero, and subsequently allows the natural system response to provide damping to the increased output, as seen by the slight rounding off and subsequent decrease at the peak of the system response plot in figure 10. At this point the system response effectively stabilises at under 2 seconds with a gain of approximately -0.03.

Figures 11-13 were generated through Simulink simulation of the system, by inputting the transfer functions generated through manual calculations. The accuracy of the calculations cannot be verified and it is thought likely that the response of the Whole System shown in figure 13 is the result of an error in calculations.

Frequency Response Plots

Pole-Zero diagrams and Root Locus Plot

The Poles and Zeros on a Pole-zero Plot show the frequencies of s where the numerator and denominators of the component transfer functions are infinite and zero respectively. At the systems Zeroes and Poles, with the denominator or numerator at zero, the value of the transfer function becomes infinity or zero respectively.

For factorable equations, this can be demonstrated by factorising the components of the transfer function, and performing a reverse Laplace transform. For non-factorable equations, such as the transfer function used here, the completing the square method can be used.

In this way, the poles and zeros of a system define the bounds of its behaviour. Plotting these in terms of Phase angle and amplitude gain against frequency, can illustrate when the behaviour of the system throughout a frequency range approaches its boundaries, and therefore its relative stability at the various frequencies.

In figure 14 the response of the pre-control turret is shown. As per the calculated coordinates shown, the system does not contain any zeros, but two poles at $-1 + 1i$ and $+1 - 1i$. This is likely to result in an overly damped system, which can be seen in the pre-control system response in figure 11, which stabilises short of the required output. As the poles are situated on the negative side of the real time axis (horizontal), the system responses will decrease over time, and the system is effectively stable to begin with, albeit operating outside of required parameters.

Bode Plot Analysis

Phase difference responses can show how Closed Loop systems will behave. If the gain response of a system has any 0dB and -180° -phase frequencies, then the system will oscillate. This can be seen illustrated by a dotted line crossing the upper of the two plots in each diagram at 0dB Amplitude and on the lower at -180° . In systems where at certain frequencies the system response reaches both 0dB and -180° (the “crossover frequencies”), a vertical line is drawn (as seen on figure 19, showing the Bode plots of the whole system response). The gain and phase margins of the system will then be shown in the range of Y-axis values between the points at which the system response meets the crossover frequencies for both.

Figure 18 is a bode plot of the uncontrolled turret response. As can be seen in the Pole-Zero plot (figure 14), the system does not reach 0dB amplitude gain, and therefore can be considered a stable system.

In both it's pre- and post-control responses, the system acts as a low pass filter, and exhibits a lower amplitude response at lower frequencies.

Post-control, the effect of the PID controller can be seen providing a damping effect to the system response between $f=10^{-1}$ to $f=10^0$. This results in a more stable decrease in Amplitude as frequencies decrease.

As the line slopes downwards toward the higher frequency ranges, the system response has two more poles than zeros. This is also visible in the Pole-Zero plot of the Turret response in figure 14.

Nichols Plot

The Nichols plot shown in figure 19 illustrates the response of the pre- and post-control system. This diagram maps directly from the bode plot and illustrates gain and phase together along the “M” and “N” circles of the grid.

Frequency ranges are not immediately apparent from this diagram however it does illustrate the effect of the PID response on the overall system response. As can be seen, the system response shows a greater margin from the critical point at 0dB and -180° , and results in a lower negative gain in the output.

References/Bibliography

<https://uk.mathworks.com/help/>

ANNEX B – MATLAB CODE EXAMPLES

Base Values:

Assigned Forward Transfer Function Values

```
a=3;  
b=6;  
c=6;
```

Controller parameters from PID tuning block in Simulink

```
Kp=142.563721604951;  
Kd=58.1132317796486;  
N=2227.18194606262;  
Ki=64.1103840053898;
```

Numerators and Denominators of Transfer functions

These equations represent the various components of Transfer functions as a function of s . They are set out in decreasing powers of s down to s^0 and represent numerators and denominators in the Laplace transform

```
% Numerator coefficients of transfer functions  
N1=Kp+ (N*Kd) ;  
N2= (Ki+ (N*Kp) ) ;  
N3=N*Ki;  
  
% Denominator coefficients of closed loop transfer function  
D1=a;  
D2= (a*N) +b;  
D3=Kp+ (N*Kd) + (b*N) +c;  
D4= (N*Kp) +Ki+ (c*N) ;  
D5=N*Ki;
```


Equations

Calculating Transfer Functions

% Create transfer function to represent only the turret response:

```
turret_tf = tf(1,[a b c]);
```

% create transfer function of controller response:

```
Controller_tf = tf([N1 N2 N3], [1 N 0]);
```

%AND/OR.. to double check it.. using rational expression

%Specify s as a tf object

```
s = tf('s')
```

%Calculation of Controller_tf (H) as rational expression as per Annex A.

```
Controller_tf_2 = Kp + (Ki*(1/s)) + Kd*(s/(((1/N)*s)+1))
```

```
figure;
```

```
subplot(1,2,1)
```

```
impz(Controller_tf);
```

```
subplot(1,2,2)
```

```
impz(Controller_tf_2);
```

```
title('Controller TF Validation');
```

% Create closed loop transfer function of whole system response (inc. feedback)

```
system_tf=tf([N1 N2 N3],[D1 D2 D3 D4 D5]);
```

%AND/OR.. to double check it.. using rational expression

%Specify s as a tf object

```
s = tf('s')
```

%Calculation of system_tf (H) as rational expression as per Annex A.

```
system_tf = (1/(a*s^2+b*s+c))/1+(Kp + (Ki*(1/s)) + Kd*(s/(((1/N)*s)+1))) * (1/(a*s^2+b*s+c))*1
```

Results

%Returns Pole-zero coordinates for turret transfer function.

```
[p,z] = pzmap(turret_tf)
```

%Returns pole-zero and gain of the system. Important to note that b and a are numerator and denominator signifiers in Matlab and employing this method will overwrite prior values entered for a and b (3 and 6 respectively).

```
b = [1]
```

```
a = [3 6 6]
```

```
[b,a] = eqtflength(b,a);
```

```
[z,p,k] = tf2zp(b,a);
```

Creating Plots and Graphs

Transfer Function Outputs

Outputs from the impulse and step responses of the various transfer functions are presented via the following code:

```
%Turret TF transfer function Impulse and Step Response
```

```
subplot(3,2,1)
```

```
step(turret_tf)
```

```
subplot(3,2,2)
```

```
impulse(turret_tf);
```

```
%Controller transfer function Impulse and Step Response
```

```
subplot(3,2,3)
```

```
step(Controller_tf)
```

```
subplot(3,2,4)
```

```
impulse(Controller_tf);
```

```
%System transfer function Impulse and Step Response
```

```
subplot(3,2,5)
```

```
step(system_tf)
```

```
subplot(3,3,6)
```

```
impulse(system_tf);
```

Nyquist Plot

Creates Nyquist plot of the forward transfer function for the Plant process, calculated by Matlab from simulation base values.

```
% Nyquist plot - This HAS to exclude the feedback as it will be a measure of  
% System stability - how much gain / phase shift needed to make instable  
figure;  
subplot(2,2,1);  
nyquist(system_tf);  
title('Nyquist Plot of forward TF');
```

Bode Plots

Creates Bode Plots of the Open Loop Transfer Function.

```
% Create Bode plot of forward transfer function  
subplot(2,2,2);  
margin(System_tf);  
%title('Bode plot of System TF')
```

Pole-Zero Plots

Closed-Loop Transfer System and Uncontrolled Plant

```
% Plot Pole-Zero diagram of closed loop system  
subplot(2,2,3);  
pzplot(system_tf);  
title('Pole-Zero plot of closed loop TF');
```

```
% Plot pole-zero diagram of turret (plant) transfer function  
subplot(2,2,4);  
pzplot(turret_tf);  
title('Pole-Zero plot of Turret Function');
```

pole-zero diagram of turret (plant), controller and Closed Loop transfer function

```
% create new figure
figure;

% with 1 row of 3 graphs, and with the first graph of these....
subplot(1,3,1);
% plot the pole-zero diagram of the plant on its own
pzplot(turret_tf);
title('Turret');

% in the next graph
subplot(1,3,2);
% plot the pole-zero diagram of the controller on its own
pzplot(Controller_tf);
title('Controller');

% and in the last graph
subplot(1,3,3);
% plot the pole-zero diagram of entire system with feedback
pzplot(system_tf);
title('System');
```

Nichols Chart

```
%This produces a gridlined Nichols Chart displaying pre and post control system
figure;
subplot (1,1,1);
nichols(turret_tf,system_tf);
ngrid
```

Root Locus Diagram

Transformation into Time Domain

```
delta = (b.^2)-(4*a*c);
if(delta < 0)
disp("Delta < 0 The equation does not have a real root");
elseif (delta == 0)
```

```
disp('The equation has only one real roots');  
disp(-b./(2*a));  
else  
disp('The equation has two real roots');  
disp((-b+sqrt(delta))./(2*a));  
disp((-b-sqrt(delta))./(2*a));
```