Maynooth University
National University of Ireland Maynooth

Proportional Controller Design for a Feedback System

Assignment - 01

Parameters:

LIAM JAMES, GLENNON, 20344313, 127, 6, 7, 2

Date:

9/11/2023

(a) Task - Routh criterion

Using the Routh criterion, determine the range of the proportional gain k for which the feedback system is stable.

$$\frac{kGp}{1+kGpH} = \frac{\frac{127k}{(s+6)(s+7)}}{1+\frac{127k}{(s+6)(s+7)}\frac{1}{(s+2)}}$$

$$\frac{\frac{127k}{(s+6)(s+7)}}{1+\frac{127k}{(s+6)(s+7)(s+2)}}$$

To find the characteristic equation, we set the denominator equal to zero:

$$(s+6)(s+7)(s+2)+127k=0$$

Expanding this, we get the characteristic polynomial:

$$s^3+15s^2+68s+84+127k=0$$

Routh Criterion Table

s^3	1	68
s^2	15	84 + 127k
s^1	A1	A2
s^0	B1	

To be stable A1 and B2 need to be greater than zero

A1:

$$\frac{-1}{15}(84+127k)-(68)(15) = -8.47k+62.4$$

$$-8.47k+62.4>0$$

$$k<7.37$$

B1 :

$$84+127k>0$$

$$k>\frac{-84}{127}$$

$$k>-0.66$$

So according to the Routh criterion, the range of the proportional gain k for which the feedback system is stable values of k for a stable system are

$$-0.66>k>7.37$$

(b) Task - Nyquist plot

Using a nominal value of k = 1, draw the Nyquist plot (using Matlab) for the system and annotate the plot clearly with a range of frequency values. Determine the stability of the feedback system using the Nyquist stability criterion. Determine the maximum value of gain k for stability.

Draw the Nyquist plot - k = 1

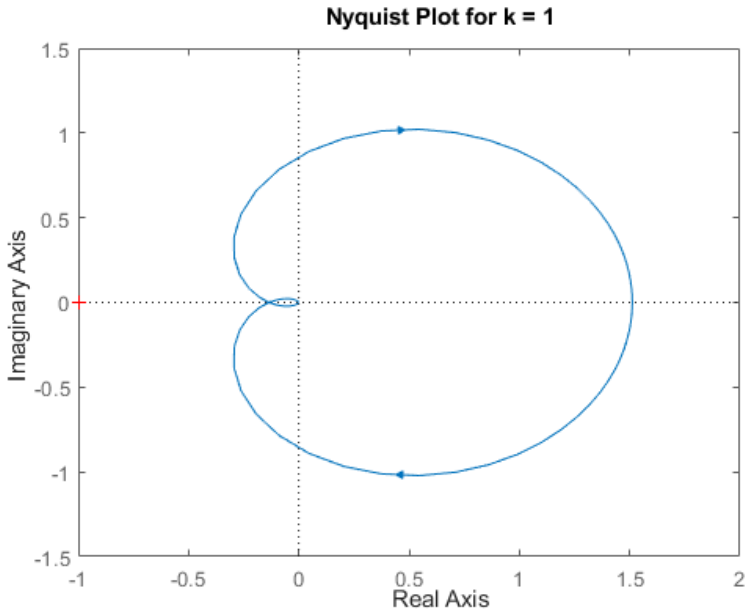


Fig b.1 Nyquist Plot for k = 1

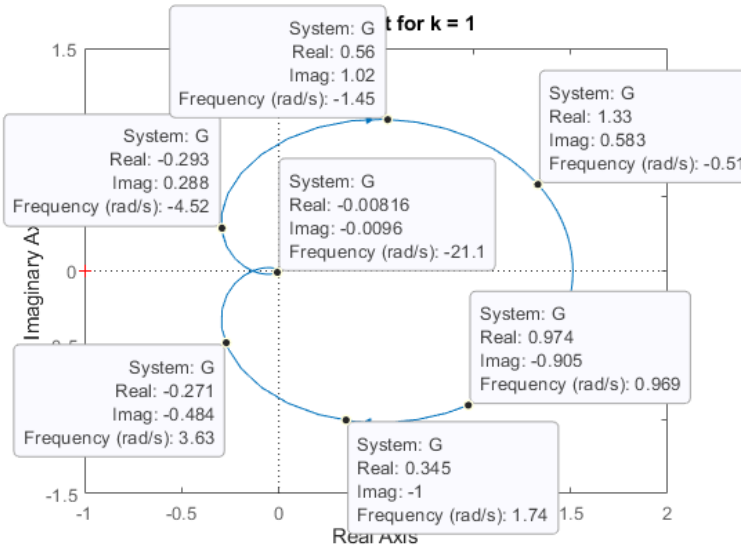


Fig b.1.1 Nyquist Plot for k = 1 with Points

Nyquist stability criterion

Poles = -6,-7,-2
P = 0, N = 0
Z = P (number of unstable poles) + N (number of encirclements)
Z = 0 Therefor the Nyquist stability criteria have been met

Bode plot

Determine the maximum value of gain k for stability.

Given the number of unstable poles = 0, then the number encirclements of -1 must be 0. If the number of the number encirclements of -1 becomes 1 then this system becomes unstable.
The Plots were checked to see when the number encirclements of -1 becomes 1. It was determined in matlab by using code to generate plots of all gains with an interval of 0.1. It was also previously known from the Routh Criterion calculations however it is good that both of these answers were equal as expected. See code appendix.(b.2)

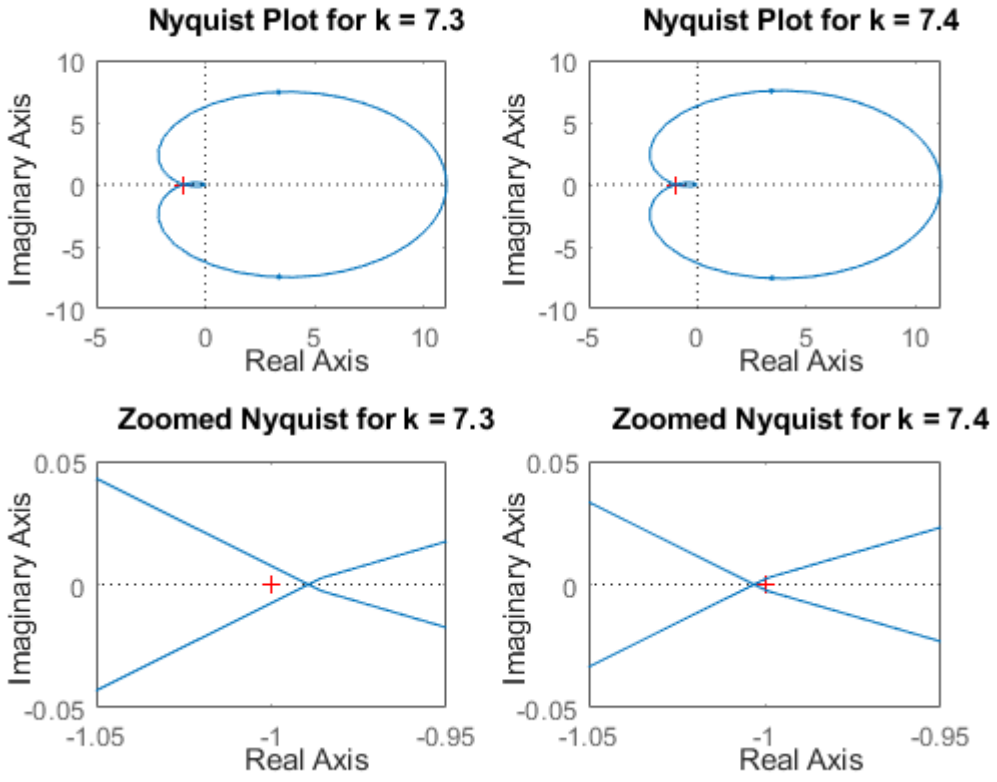


Fig b.3 Nyquist Plot for k = 7.3,7.4

(c) Task Bode Plot

Plot the Bode plot (gain and phase plots) for the feedback system, for a nominal value of k = 1. Use both the asymptotic approximation AND the Matlab calculation for the magnitude plot. The asymptotic approximation (determined manually) can be drawn using the line tool in the Matlab Figure environment. From the Matlab plot, determine the Gain Margin of the system and the frequency at this point.

Plot the Bode plot (gain and phase plots) for the feedback system

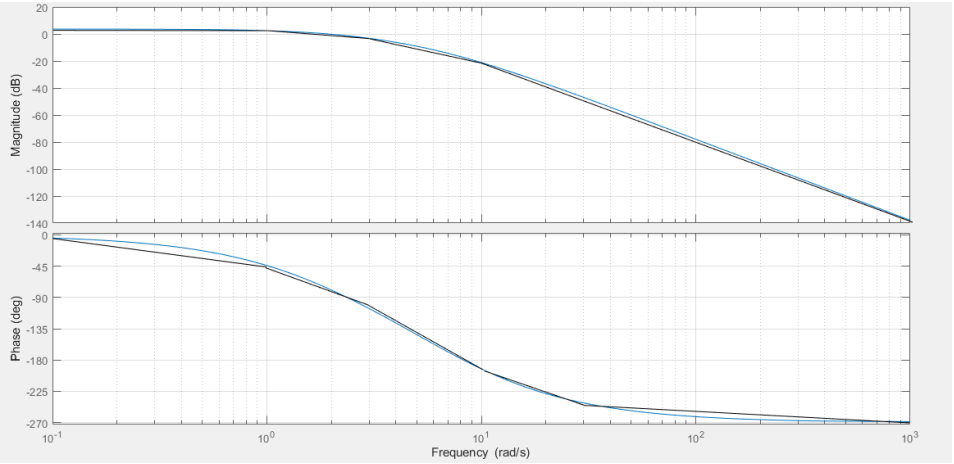
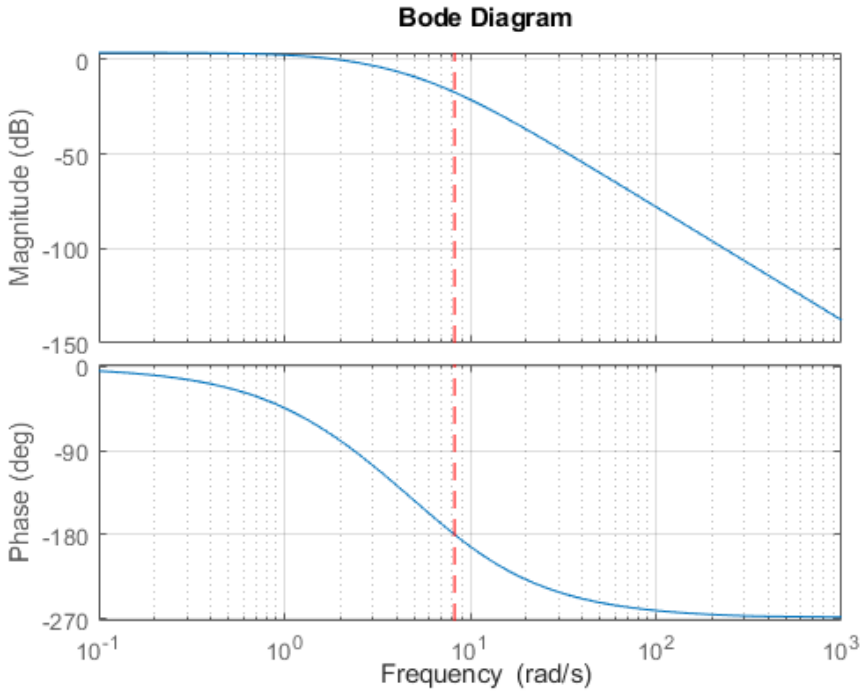


Fig c.1 Straight-line Bode Approximation



Gain Margin: 17.349452 dB at frequency 8.246215 rad/s

Fig c.2 Bode

For k = 1,Gain Margin = 17.35dB,Frequency phase passes through -180 = 8.25 rad/s

(d) Task Gain Margin

Determine the value of k to achieve a gain margin of 2 for the feedback system. Using Matlab, plot both the Nyquist AND Bode plot for your calculated k value and confirm that the result is correct in both cases (i.e. the Gain Margin should be approximately 2dB on both plots).

$$2dB = 10^{\frac{2}{10}} = 1.259$$

$$\sum_{i=1}^{n} x_i^2$$

$$k = 5.86$$

Nyquist plot

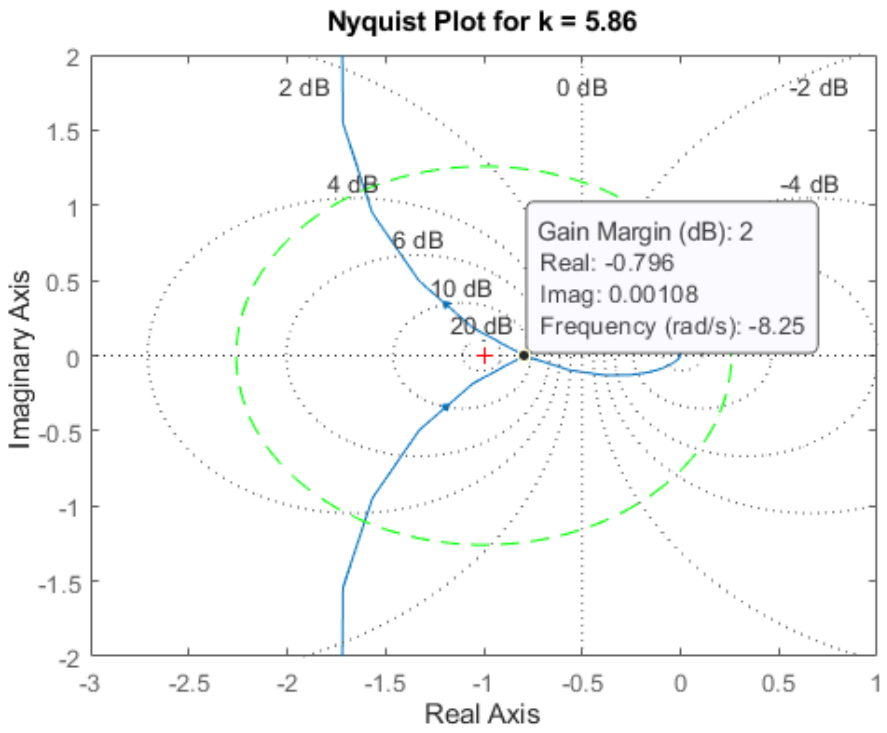


Fig d.1 Nyquist Plot for k = 5.86

Bode plot

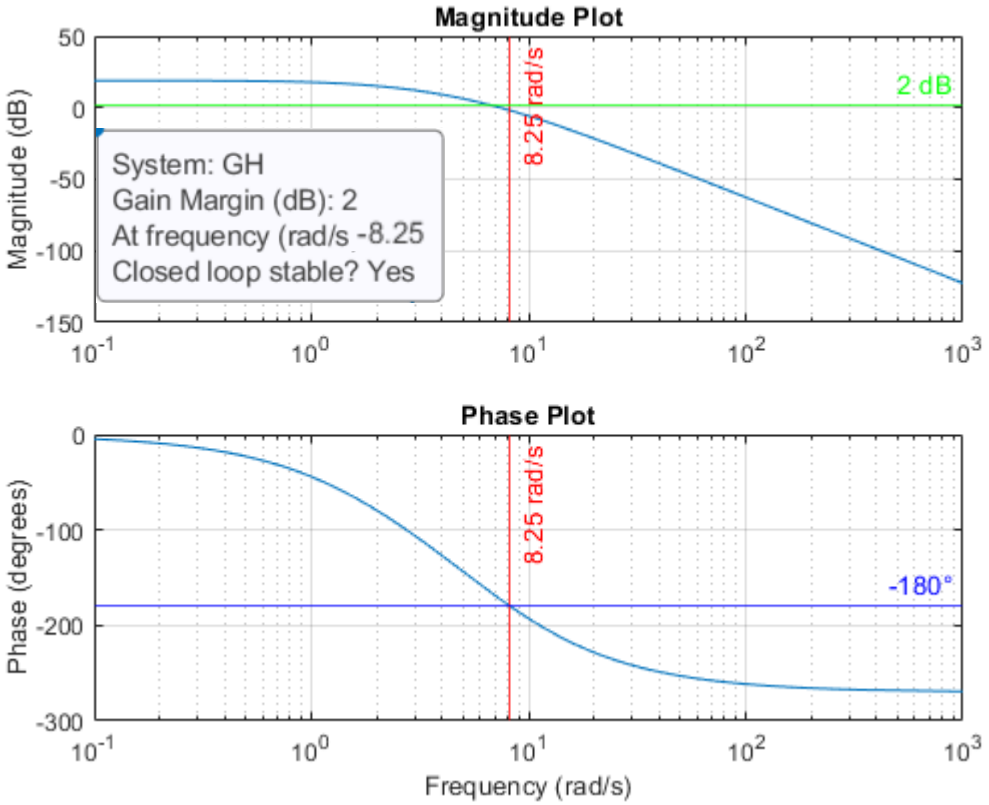


Fig d.2 Bode Plot for k = 5.86

(e) Task Root Locus

Sketch the root locus diagram for the feedback system for k as a variable. Confirm your sketch by using Matlab, and a nominal value of k = 1, to generate the root locus. Annotate the diagram with a range of values of k at various (important) points.

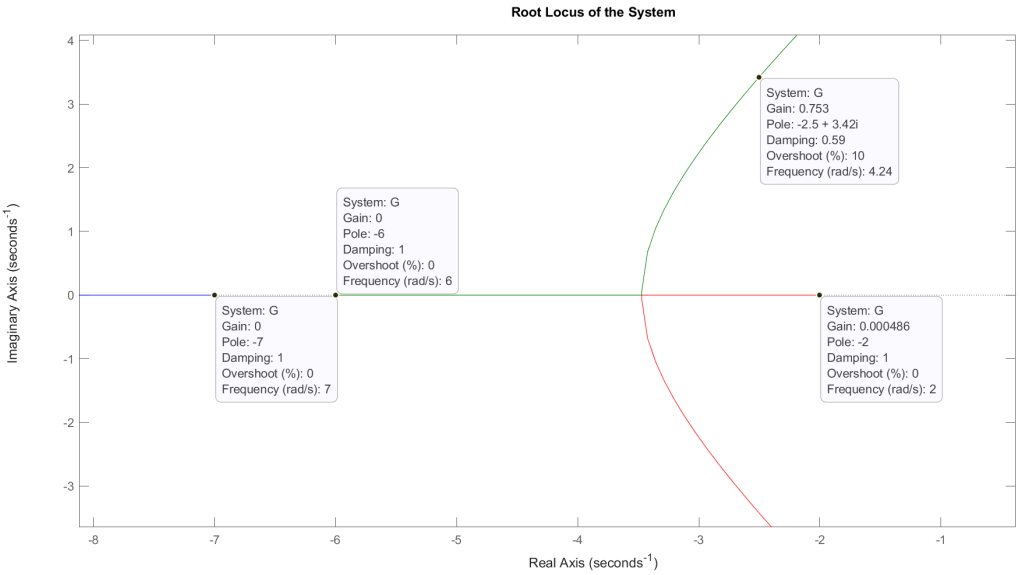


Fig e.1 Root Locus

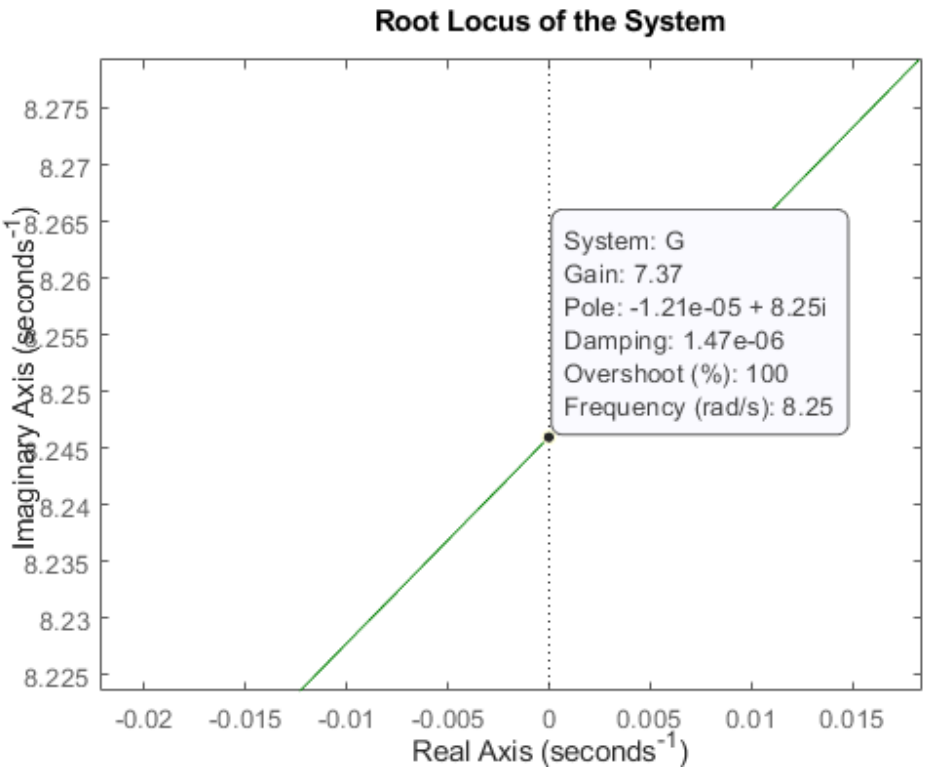


Fig e.1 Root Locus zoomed

(f) Task Design Specifications

Determine the value of k required to meet the following design specifications:

- 10% overshoot (obtain the value of ζ for this overshoot)

- Settling time of less than $10s$

Using the latter, determine the range of frequency n that applies. Estimate the settling time of the system for your chosen k value. Can we improve on this settling time in the given set-up? Justify your answer.

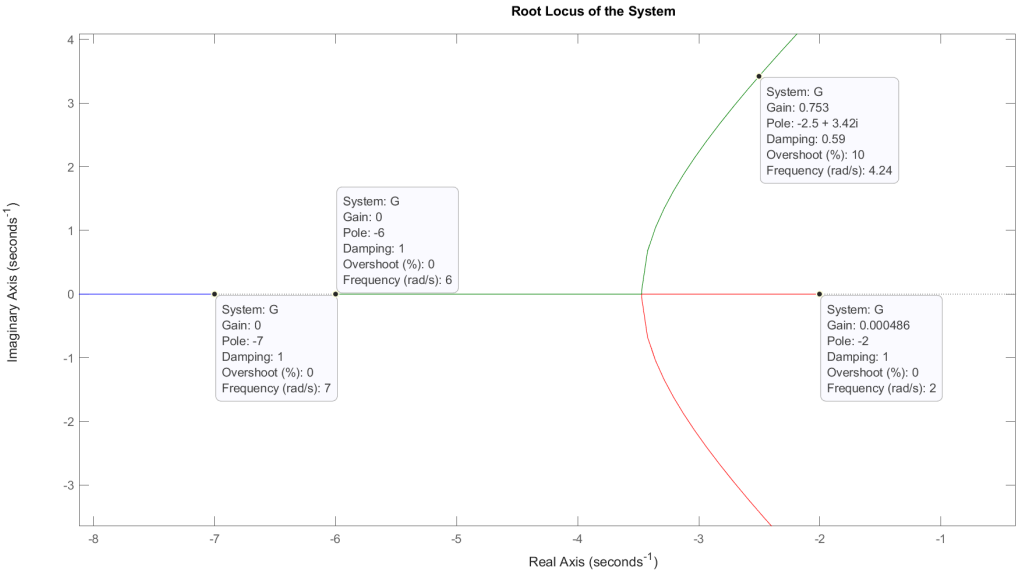


Fig f.1 Root Locus

It is clear from the root locus that the gain k is 0.753 when there's a 10% overshoot

$$10 = 100e\frac{-pi}{\sqrt{1-^2}} \qquad = 0.59$$

$$Ts = \frac{4}{wn} < 10 \qquad Ts = \frac{4}{(0.59)(10)} < 10$$

$$0.678 < wn \qquad Ts = \frac{4}{(0.59)(4.42)} = 1.53$$

This cant be improved given the damping ratio has to be 0.59 to achieve a 10% overshoot, and the k value of 0.753 dictates the natural frequency to be 4.24 rad/s.

(g) Task Simulink

Simulate the feedback system in Simulink and compare the step responses obtained using the design from (d) to that obtained from the design in (f). Plot both response on the same set of axes. Using the in-built Figure functionality, obtain the peak overshoot for the response from the design in (f). Is it 10% as specified in (f)? Comment, with justification, on your findings.

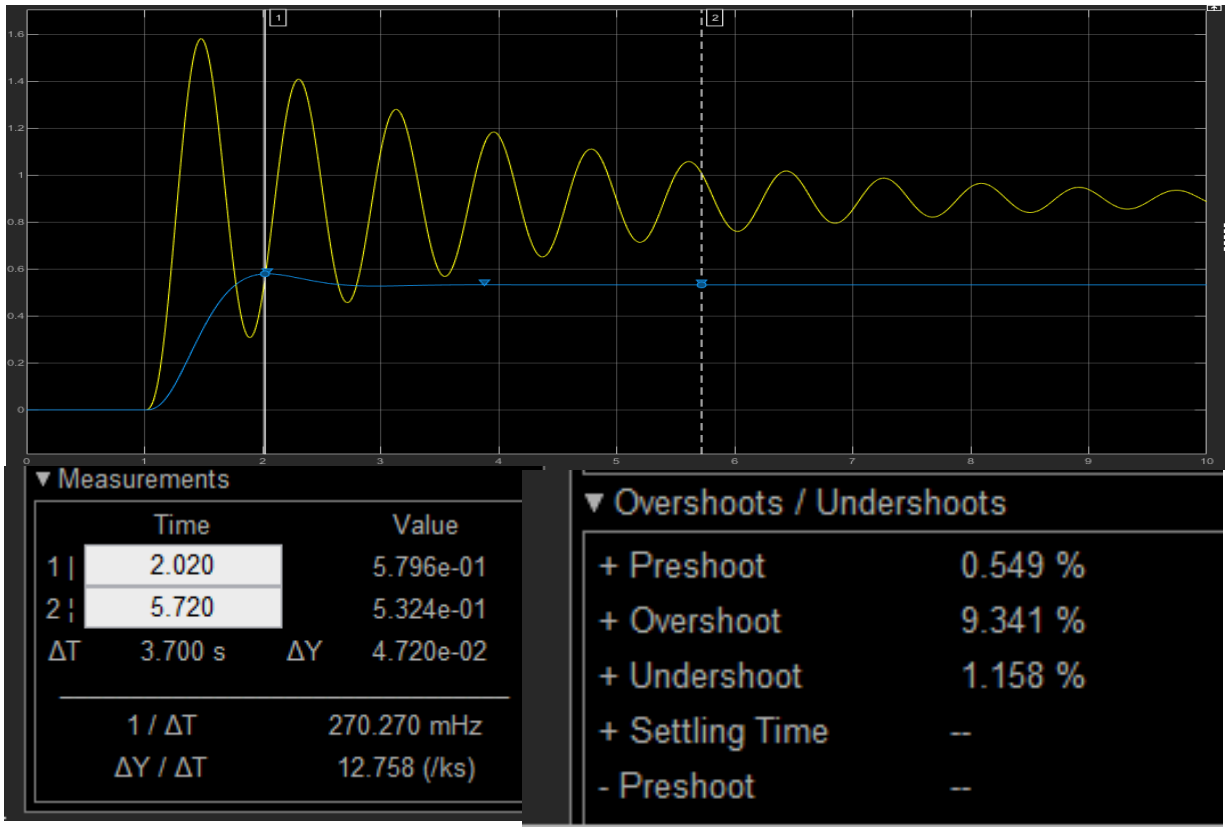


Fig g.1 Scope Out

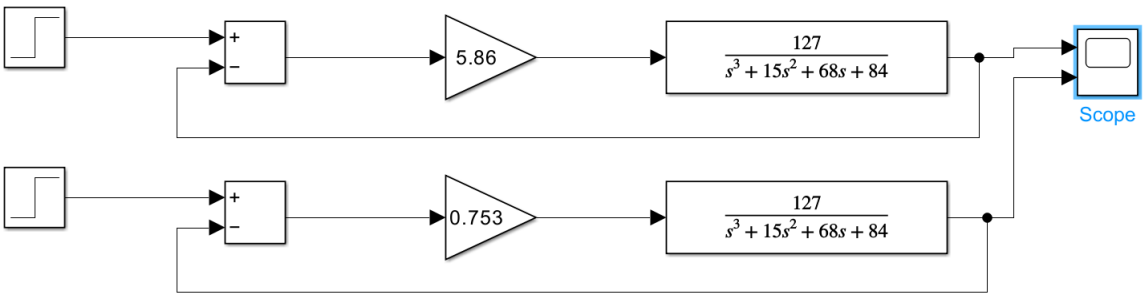


Fig g.2 Simulink Model

Approximately 9.34 % overshoot slightly under what was expected from root locus.
That discrepancy probably comes from stacking rounding errors and floating point calculations.

$$10 = 100e\frac{-pi}{\sqrt{1-^2}}$$

For a damping ratio of 0.59, this formula predicts an overshoot close to 10%. However, the actual system behaviour can vary slightly due to factors like model approximations, non-linearities, or external disturbances..

(h) Task Conclusion and Comparison

Write a brief conclusion, comparing frequency domain (Nyquist, Bode) design to design using transient response specifications and root locus.

In control system design, various methods are employed to analyse and design systems to meet specific performance criteria. Among these methods, frequency domain analysis (such as Nyquist and Bode plots) and time domain analysis (using transient response specifications and root locus) are widely used. Each method offers unique insights and tools for control system design, and they often complement each other. Here's a brief comparison:

Frequency Domain Analysis (Nyquist, Bode Plots)

Insight into Stability and Margins: Frequency domain methods, particularly Nyquist and Bode plots, are excellent for assessing system stability and robustness. They provide clear information about gain and phase margins, which are crucial for understanding how close the system is to instability.
Frequency Response: These methods are adept at analysing how a system responds to different frequencies, making them ideal for systems where frequency characteristics are of primary concern, such as in filters or vibration analysis.

Time Domain Analysis (Root Locus)

Root Locus Technique: The root locus technique is particularly powerful for understanding the effects of varying system parameters (like gain) on pole locations and hence on system stability and transient response.

Conclusion

- Complementary Nature: Both frequency domain and time domain methods are complementary. While frequency domain methods provide a clear picture of system behaviour across a range of frequencies, time domain methods excel in predicting the time response of a system to various inputs.
- Design Integration: In practice, a robust control system design often integrates both approaches. For instance, one might use Bode plots to ensure adequate margins for stability and then use root locus or transient response analysis to fine-tune the system for desired time-domain specifications.

In summary, both frequency domain and time domain analyses are fundamental to control system design, each offering unique and valuable perspectives. The integration of both approaches often leads to a more comprehensive and robust system design.

Code Apendix.

```
% Define the transfer function

s = tf('s');

G = 127/((s+6)*(s+7)*(s+2));

% Plot the Nyquist plot

figure;

nyquist(G);

title('Nyquist Plot for k = 1');
```

Code b.1

```
% Define the range of k

k = 0:0.1:10; % Adjust the range and step size as needed

% Loop over k values

for i = 1:length(k)

   % Define the transfer function for each k

   Gs = tf(127*k(i), conv([1, 6], conv([1, 7], [1, 2])));

   % Plot the Nyquist diagram

   figure;

   nyquist(Gs);

   title(['Nyquist Plot for k = ', num2str(k(i))]);

end
```

Code b.2

```
% Define the transfer function for different k values

s = tf('s');

G1 = (127*7.3)/((s+6)*(s+7)*(s+2));

G2 = (127*7.4)/((s+6)*(s+7)*(s+2));

% Create a 2x2 subplot

figure;

% Nyquist plot for k = 7.3

subplot(2,2,1);

nyquist(G1);

title('Nyquist Plot for k = 7.3');

% Nyquist plot for k = 7.4

subplot(2,2,2);

nyquist(G2);

title('Nyquist Plot for k = 7.4');

% Zoomed Nyquist plot for k = 7.3

subplot(2,2,3);

nyquist(G1);

title('Zoomed Nyquist for k = 7.3');

axis([-1.05 -0.95 -0.05 0.05]);

% Zoomed Nyquist plot for k = 7.4

subplot(2,2,4);

nyquist(G2);

title('Zoomed Nyquist for k = 7.4');

axis([-1.05 -0.95 -0.05 0.05]);
```

Code b.3

```
% Define the transfer function with k = 1

k = 1;

Gs = tf(127*k, conv([1, 6], conv([1, 7], [1, 2])));

figure;

bode(Gs);

grid on;

% Determine the Gain Margin and the frequency

[gainMargin, phaseMargin, gmFreq, pmFreq] = margin(Gs);

fprintf('Gain Margin: %f dB at frequency %f rad/s\n', 20*log10(gainMargin), gmFreq);
```

Code c.1

```
% Define the transfer function

s = tf('s');

k = 5.86;

G = (127*k)/((s+6)*(s+7)*(s+2));

% Plot the Nyquist plot

figure;

nyquist(G);

title('Nyquist Plot for k = 5.86');

grid on;

hold on;

% Add a point at (-1,0)

plot(-1, 0, 'ro', 'MarkerFaceColor', 'r');

text(-1, 0, '(-1,0)', 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');

% Calculate the radius for 2 dB gain

gain_2dB = 10^(2/20); % Convert 2 dB to linear scale

% Create a circle centered at (-1,0) with the calculated radius

theta = linspace(0, 2*pi, 100);

x_circle = -1 + gain_2dB*cos(theta);

y_circle = gain_2dB*sin(theta);

plot(x_circle, y_circle, 'g--');

text(-1 + gain_2dB/2, gain_2dB/2, '2 dB Circle', 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'left');

axis equal;

xlim([-3 1]);

ylim([-2 2]);
```

Code d.1

```
% Define the transfer function with k = 1
```

```
k = 1;

Gs = tf(127*k, conv([1, 6], conv([1, 7], [1, 2])));

% Plot the Bode plot

figure;

bode(Gs);

grid on;

% Determine the Gain Margin and the frequency

[gainMargin, phaseMargin, gmFreq, pmFreq] = margin(Gs);

fprintf('Gain Margin: %f dB at frequency %f rad/s\n', 20*log10(gainMargin), gmFreq);

% Define the transfer function

s = tf('s');

k = 1;

G = (127*k)/((s+6)*(s+7)*(s+2));

% Plot the Bode plot

h = bodeplot(G);

setoptions(h, 'FreqUnits', 'rad/s'); % Set frequency units to rad/s

title('Bode Plot for k = 1');

grid on;

% Enable data cursor mode

datacursormode on;

% Modify the data cursor text

dcm_obj = datacursormode(gcf);

set(dcm_obj, 'UpdateFcn', @myupdatefcn);

% Custom update function for data cursor

function txt = myupdatefcn(~, event_obj)

   % Obtain clicked position

   pos = get(event_obj, 'Position');

   freq = pos(1);

   mag_db = pos(2);

   % Display the information

   txt = {['Frequency: ', num2str(freq), ' rad/s'], ...

          ['Magnitude: ', num2str(mag_db), ' dB']};

end
```

Code d.2

```
% Define the transfer function
```

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

G = 127/((s+6)*(s+7)*(s+2));

% Plot the root locus

figure;

rlocus(G);

title('Root Locus of the System');

% Use rlocfind to select a point on the root locus plot

[k, poles] = rlocfind(G);

% Display the selected gain and pole location

disp(['Selected Gain: ', num2str(k)]);

disp(['Pole Location: ', num2str(poles)]);

% Calculate and display damping ratio, overshoot, and natural frequency

for i = 1:length(poles)

   zeta = -real(poles(i))/abs(poles(i));

   wn = abs(poles(i));

   PO = exp((-pi * zeta) / sqrt(1 - zeta^2)) * 100;

   disp(['Pole ', num2str(i), ':']);

   disp(['  Damping Ratio: ', num2str(zeta)]);

   disp(['  Natural Frequency: ', num2str(wn), ' rad/s']);

   disp(['  Percent Overshoot: ', num2str(PO), '%']);

end
```

Code e.1

```
% Define the transfer function
```

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

G = 127/((s+6)*(s+7)*(s+2));

% Plot the root locus

figure;

rlocus(G);

title('Root Locus of the System');

% Use rlocfind to select a point on the root locus plot

[k, poles] = rlocfind(G);

% Display the selected gain and pole location

disp(['Selected Gain: ', num2str(k)]);

disp(['Pole Location: ', num2str(poles)]);

% Calculate and display damping ratio, overshoot, and natural frequency

for i = 1:length(poles)

   zeta = -real(poles(i))/abs(poles(i));

   wn = abs(poles(i));

   PO = exp((-pi * zeta) / sqrt(1 - zeta^2)) * 100;

   disp(['Pole ', num2str(i), ':']);

   disp(['  Damping Ratio: ', num2str(zeta)]);

   disp(['  Natural Frequency: ', num2str(wn), ' rad/s']);

   disp(['  Percent Overshoot: ', num2str(PO), '%']);

end
```

Code f.1