كليـــات التقنيـــة العليـــا
Higher Colleges of Technology

ELE 4623 Control Systems

Pitch Control of Variable Speed Wind Turbine

Team Members	ID
Alazar Wake	
Mahteb Negussie	
Nahom Megerssa	
Yetemegeta Sebisbe	

Submitted to: Dr. Kamal Abuqaaud
Submission Date: 10/05/2026

Table of Contents

1. Abstract 1
2. Introduction 2
3. Mathematical Modelling 3
4. Analysis of the Original System 4
   4.1. Unit Step Response (Question a): 4
   4.2. Time-Domain Expression and Output at t = 0.5 s (Question b and c): 5
   4.3. Step info values 6
      4.3.1. Maximum Overshoot and Peak Value (Question d): 6
      4.3.2. Peak time, Rise time and settling time (Question e): 7
   4.4. Steady state output (Question f): 7
      4.4.1. Steady State Error (Question g): 8
   4.5. Frequency Domain Analysis: Bode Plot (Question h) 8
      4.5.1. Phase Margin and Gain Margin (Question I): 9
   4.6. Stability Analysis (Question j): 10
   4.7. Root Locus Analysis (Questions k & l) 11
      4.7.1. Root locus stability analysis (Question l): 11
      4.7.2. Critical Stability Gain (Question m): 12
   4.8. Gain for Critically Damped Response (ζ = 1) (Question N): 13
   4.9. Gain for 5% Overshoot (Question O): 14
   4.10. Gain Compensator: PM + 10° (Question p): 15
      4.10.1. Original vs Gain-Compensated System (Question q): 16
5. Phase Lead Compensator (Question R1): 17
      5.1. Phase Lag Compensator (Question R2) 18
      5.1.1. Comparison: Original, Lead and Lag Compensators (Question s) 19
   5.2. PID Controller Ziegler-Nichols Method (Question T) 21
      5.2.1. Original vs PID (Question u) 23
6. Results and Engineering Discussion 24
   6.1. Summary of All Designs 24
7. Conclusion 27

References 28

1. Abstract

This report describes the design and analysis of classical control schemes for a variable speed wind turbine pitch controller system. This involves carrying out mathematical modeling for the pitch actuator and drive train components, determining the transfer function of the closed loop and analyzing its characteristics in the time domain as well as frequency domain. It is found that the uncompensated system has a phase margin of about 18.1°, gain margin of 5.47 dB and an overshoot of more than 60% during a step input.

Three controller designs have been considered here, these include:
(1) Gain compensation for increasing phase margin by 10 degrees, (2) Phase lead and phase lag controllers for obtaining a phase margin of 80 degrees and (3) Proportional Integral Derivative (PID) controller using Ziegler–Nichols tuning. Performance analysis was done using the MATLAB simulation software and the three control systems were analyzed using rise time, settling time, overshoot, percent error and margins as criteria of comparison. It is seen that phase-lead controller has the best performance in terms of transient behavior, lag controller improves steady state but slows down the response and the PID controller gives a good compromise between the two.

2. Introduction

Wind turbines are one of the popular renewable energy resources that has been researched in recent times. It harnesses the wind power to generate electricity. Figure 1 below shows typical components of a wind turbine system. Detail description of each sub-component can be read from literature [1]. To harvest wind kinetic energy efficiently, modern wind turbines are equipped with huge structural loads such as aerodynamic, gravitational, centrifugal, gyroscopic and operational loads. Increase in these loads on the wind turbine system causes mechanical damage which can result in the decrease in the lifespan and efficiency of the wind turbine [2,3]. The problem can be addressed with good control of the turbine rotor blades pitch angle. Hence, design and evaluation of appropriate pitch angle controller has been a topic of research in the literatures [1-4].

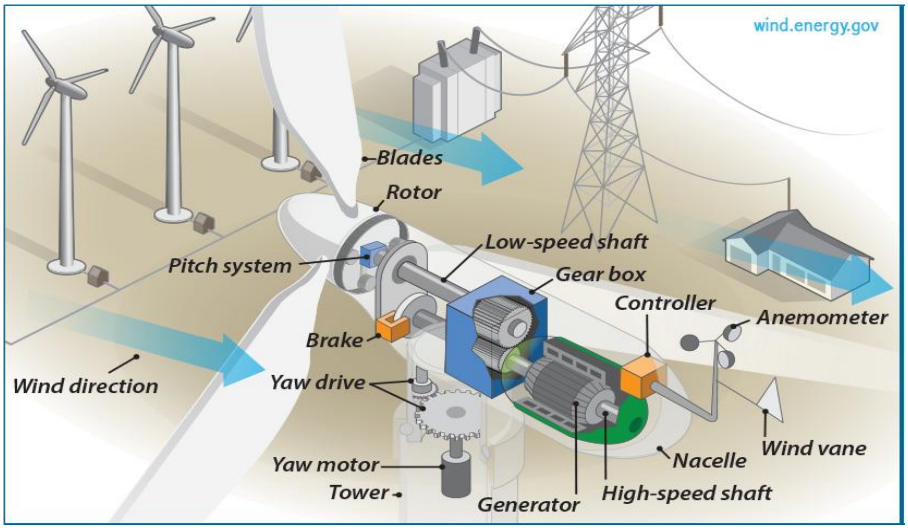


Figure 1: Inside a Wind Turbine [1]

3. Mathematical Modelling

3.1. System Description and Block Diagram

Figure 2 shows the block diagram of the wind turbine pitch control system. The system comprises three functional blocks in the forward path: a loop-shaping integrator with gain A, a pitch actuator G*p*(s) and a drive train G*d*(s). The drive train output is fed back in a unity inner loop before being cascaded with the actuator and loop-shaping term. The overall system uses unity negative feedback.

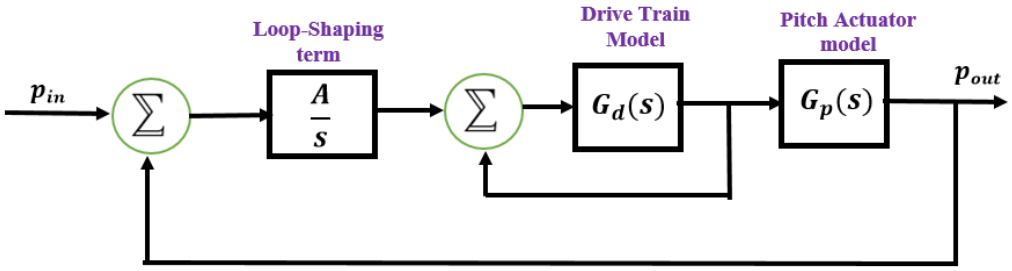


Figure 2: Block diagram of the pitch control system showing inner and outer feedback loops

$G_p(s) = \frac{1}{T_p s+1}$ (1) and $G_d(s) = \frac{1}{Js+B}$ (2)

The system parameters are:

Parameters	Description	Value
J	Wind turbine inertial $(kg.m^2)$	1
B	Damper coefficient $(N.\frac{m}{rad}/sec)$	1.5
T_p	Actuator time constant	0.2
A	Amplifier (nominal controller) gain	10

Table 1: Value of system parameter

4. Analysis of the Original System

4.1. Unit Step Response (Question a):

To find the unit step response, we first needed to find the transfer function of the system. First, $G_d(s)$ is in feedback orientation with unity feedback. As can be seen in figure 3 this resulted in Gin. Second, the total forward gain is the series combination of $\frac{A}{s}$, $G_d(s)$ and Gp(s). Finally, the overall transfer function is the feedback of the forward gain G and 1. The calculation is given below:

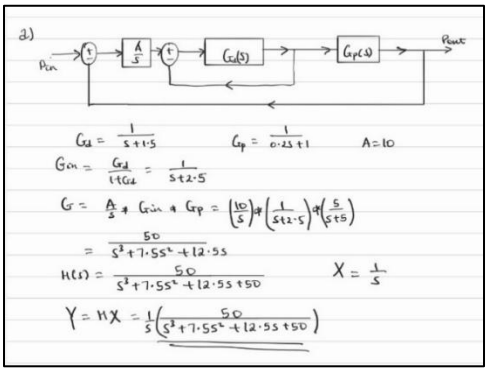


Figure 3: Transfer function calculation

The MATLAB code to get the transfer function is given below:

```
%% System Parameters
J  = 1;      % Wind turbine inertia (kg.m^2)
B  = 1.5;    % Damper coefficient (N.m.rad/sec)
Tp = 0.2;    % Actuator time constant
A  = 10;     % Amplifier (nominal controller) gain

Gp = tf(1, [Tp, 1]);                    % Pitch actuator: 1/(0.2s+1)
Gd = tf(1, [J,  B]);                    % Drive train:    1/(s+1.5)
LoopShaping = tf(A,  [1,  0]);          % Integrator+gain: 10/s
G_inner = feedback(Gd, 1);                % Inner feedback: Gd/(1+Gd)
G = series(LoopShaping, series(G_inner, Gp));  % Open-loop G(s)
H = feedback(G, 1)                    % Closed-loop H(s)
```

```
H =

          10
  ------------------------------
  0.2 s^3 + 1.5 s^2 + 2.5 s + 10
```

Figure 4: Finding transfer function using MATLAB

The unit step response of the system H(s) was calculated using the MATLAB `step()` function as can be seen in Figure 4.

```
%% System Parameters
J  = 1;      % Wind turbine inertia (kg.m^2)
B  = 1.5;    % Damper coefficient (N.m.rad/sec)
Tp = 0.2;    % Actuator time constant
A  = 10;     % Amplifier (nominal controller) gain

Gp = tf(1, [Tp, 1]);                    % Pitch actuator: 1/(0.2s+1)
Gd = tf(1, [J,  B]);                    % Drive train:    1/(s+1.5)
LoopShaping = tf(A,  [1,  0]);          % Integrator+gain: 10/s
G_inner = feedback(Gd, 1);                % Inner feedback: Gd/(1+Gd)
G = series(LoopShaping, series(G_inner, Gp));  % Open-loop G(s)
H = feedback(G, 1)                    % Closed-loop H(s)

step(H)
```

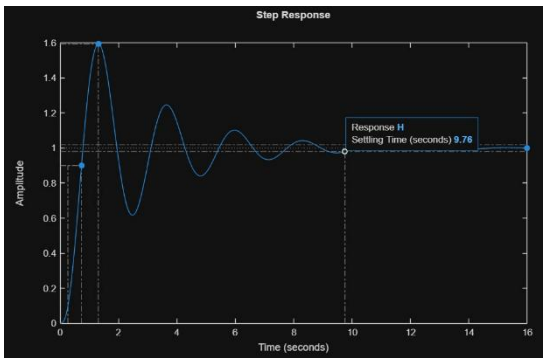


Figure 5: Step response

The step response reveals that the system is underdamped. The key performance metrics were extracted from MATLAB's `stepinfo()` and are summarized in Table 2 in Section 4.4.1.

4.2. Time-Domain Expression and Output at t = 0.5 s (Question b and c):

The closed-loop transfer function is H(s) = 50 / (s³ + 7.5s² + 12.5s + 50) as derived in section 4.1. The conversion to time domain is given below:

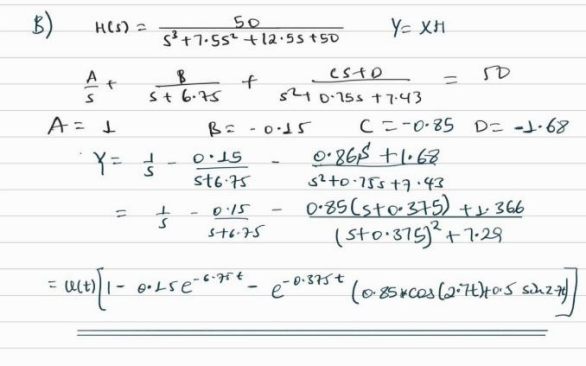


Figure 6: Time Domain calculation

The time-domain expression is obtained by using the inverse Laplace transform. In MATLAB, the Symbolic Math Toolbox is used:

```
syms s t
%% Transfer Functions
Gp_s = 1 / (Tp*s +1);
Gd_s = 1 / (J*s + B);
k_s = A / s;
Gdf_s = Gd_s/(Gd_s +1);

G_s = simplify(k_s * Gdf_s * Gp_s); % Open-loop G(s) (symbolic)
H_s = simplify(G_s/(G_s+1));  % Closed-loop H(s) (symbolic)

% Question (b): Time-domain output for unit step input
fprintf('(b) Time Domain Response\n')
Y_s = H_s * (1/s);          %<-- multiply by Laplace of unit step
Y_t = vpa(ilaplace(Y_s),4)  % Y(t) = L^-1{Y(s)}

% Manual equation from partial fractions (handwritten derivation)
y_manual = 1 - 0.15*exp(-6.75*t_vals) - exp(-0.375*t_vals) .* (0.85*cos(2.7*t_vals) ...
    + 0.5*sin(2.7*t_vals));
```

Figure 7: Time domain calculating using MATLAB

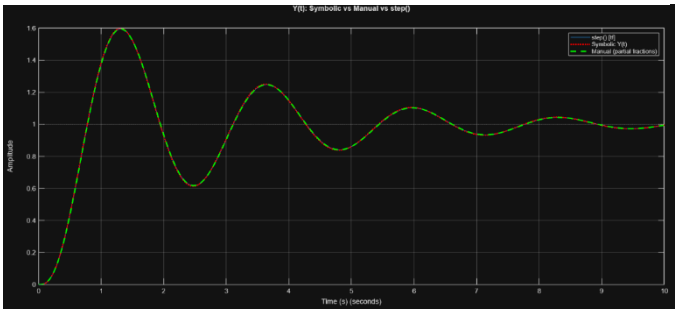


Figure 8: Time domain conversion and Time domain VS Step response

The exact time domain formula we calculated using MATLAB is:

$$Y(t)=1.0-0.155\,e^{-6.746t}+e^{-0.3771t}\cos(2.696t)\,(-0.4225+0.253\,i)+e^{-0.3771t}\cos(2.696t)\,(-0.4225-0.253\,i)+e^{-0.3771t}\sin(2.696t)\,(-0.253-0.4225\,i)+e^{-0.3771t}\sin(2.696t)\,(-0.253+0.4225\,i)$$

- The output pitch angle at t = 0.5 sec was calculated by substituting 0.5 in the time domain equation.

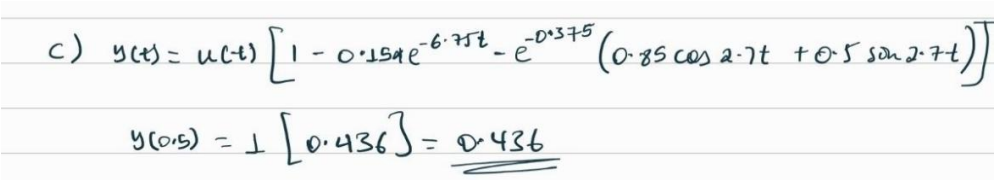


Figure 9: Pitch at 0.5 sec calculation

The output pitch angle at t = 0.5 sec was calculated by simply substituting t = 0.5 into the symbolic expression in MATLAB:

```
Y_t_05 = double(real(subs(Y_t, t, 0.5))) % Substitute 0.5 inplace of t, Y_t(0.5)

Y_t_05 = 0.4315
```

Figure 10: Pitch angle at t= 0.5 sec

4.3. Step info values

4.3.1. Maximum Overshoot and Peak Value (Question d):

Because it is a third-order system, the second-order performance formulas don't work. However, by removing the least dominant pole we can reach a second order system and apply the formulas.

In MATLAB, we used the function `stepinfo()` to obtain %OS, its peak value and much more.

```
stepinfo(H)

ans = struct with fields:
         RiseTime: 0.4647
    TransientTime: 9.7570
     SettlingTime: 9.7570
      SettlingMin: 0.6171
      SettlingMax: 1.5943
        Overshoot: 59.4283
       Undershoot: 0
             Peak: 1.5943
         PeakTime: 1.3107
```

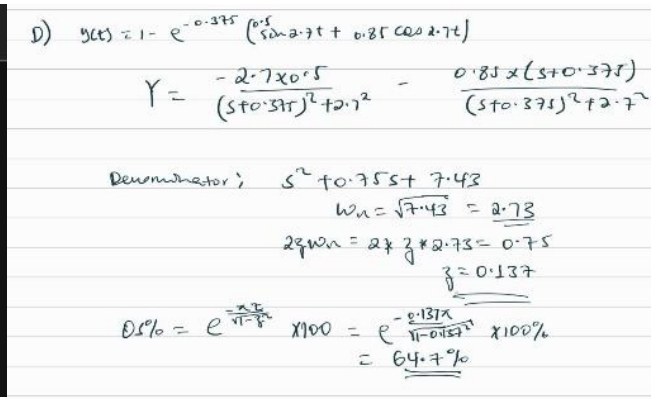


Figure 11: Overshoot and peak value calculation

4.3.2. Peak time, Rise time and settling time (Question e):

In MATLAB, we used the function `stepinfo()` to obtain Peak time, Rise time and settling time.

```
stepinfo(H)

ans = struct with fields:
         RiseTime: 0.4647
    TransientTime: 9.7570
     SettlingTime: 9.7570
      SettlingMin: 0.6171
      SettlingMax: 1.5943
         Overshoot: 59.4283
       Undershoot: 0
             Peak: 1.5943
         PeakTime: 1.3107
```

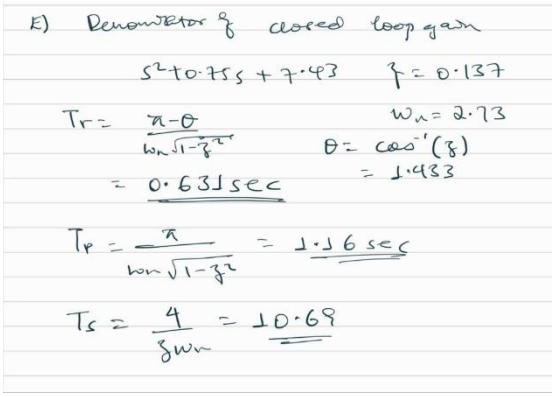


Figure 12: Peak time, Rise time and settling time MATLAB vs calculations

4.4. Steady state output (Question f):

The steady state output is determined using the Final Value Theorem as the system is stable:

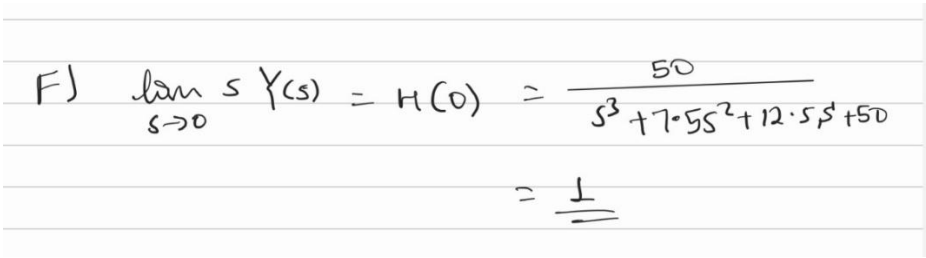


Figure 13: Steady state calculation

In MATLAB, steady state output was found by using the `dcgain(Tf)` function which is basically computing H (0) for it is a stable system.

```
yss = dcgain(H) % Steady State Output

yss = 1
```

Figure 14: Steady state computation through MATLAB

4.4.1. Steady State Error (Question g):

Because the system is a type 1 system the steady-state error is computed as follows:

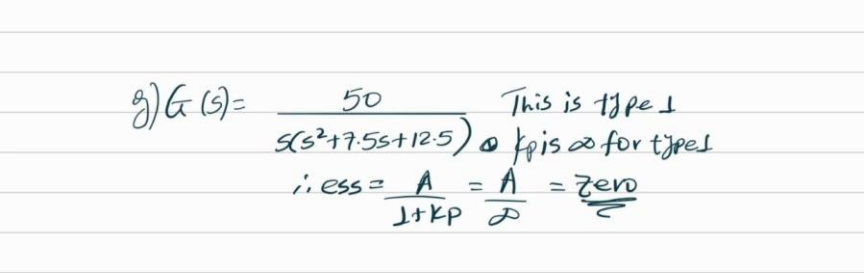


Figure 15: Steady State error computation

```
yss = dcgain(H)
ess = 1 - yss
```

```
yss = 1
ess = 0
```

Specification	Analytical	Simulation
Peak Time, Tp	1.17 s	1.31 s
Rise Time, Tr	0.64 s	0.46 s
Settling Time, Ts	10.5 s	9.75 s
Overshoot %OS	64.7%	59.43 %
Steady State	1	1

Table 2: Time-domain specifications. analytical vs. simulation

4.5. Frequency Domain Analysis: Bode Plot (Question h)

Bode plot for the open loop transfer function G(s) is created by means of MATLAB's built-in `margin()` function that can automatically determine the crossover frequency points.

```
figure(3);margin(G)
```

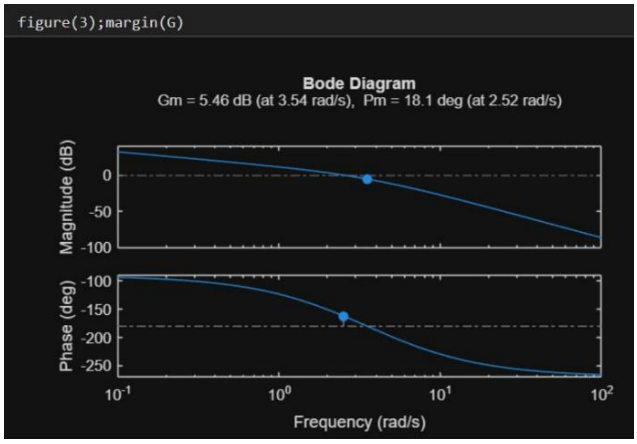


Figure 16: Bode plot

- Gain Crossover Frequency ≈ 2.52 rad/s: frequency at which |G(jω)| = 0 dB (unity gain).
- Phase Crossover Frequency ≈ 3.54 rad/s: frequency at which ∠G(jω) = −180°.

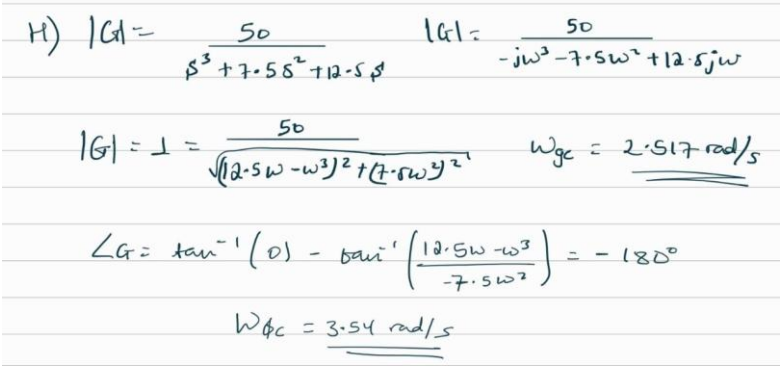


Figure 17: Gain and Phase crossover frequency calculation

4.5.1. Phase Margin and Gain Margin (Question I):

From our calculation and the Bode plot in section 4.5 we have the gain margin and the phase margin. In the bode plot, it can be seen that the PM is = 18.1^0 and the GM is = 5.46dB the calculation is as follows:

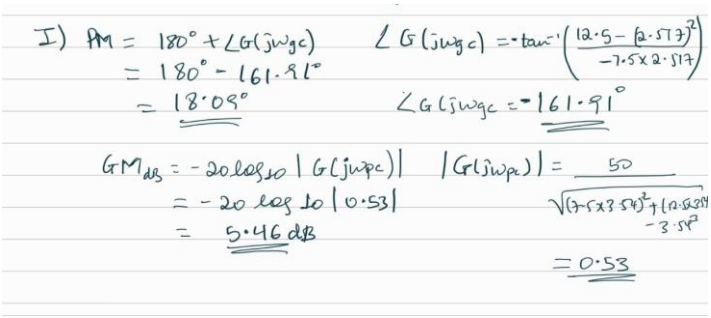


Figure 18: Phase margin and Gain margin calculation

These values are verified using `MATLAB`

```
[Gm, Pm, Wcg, Wcp] = margin(G)

Gm = 1.8750
Pm = 18.0777
Wcg = 3.5355
Wcp = 2.5174
```

Figure 19: Gm, Pm, Wcg and Wcp from MATLAB

4.6. Stability Analysis (Question j):

We can look at the stability in many ways, from the step response we can conclude the system is stable as it decays to 1 as time goes by. Also, we can comment on the stability by other two methods:

- **Closed-Loop Pole Locations:**

The closed-loop poles of H(s) are: $s_{1,2} = -0.38 \pm j2.69$ and $s_3 = -6.74$. All three poles have real negative parts, confirming that all poles lie in the Left-Half Plane (LHP). However this does not give us the full picture of stability.

- **Gain and Phase Margins:**

As PM = 18.1° > 0° and GM = 5.47 dB > 0 dB, the system is stable. However, since PM is less than 30°, the system is stable but has bad damping characteristics, leading to a big overshoot (approximately 64%) and a relatively long settling time of about 10.5 seconds. The system requires a compensator.

4.7. Root Locus Analysis (Questions k & l)

The root locus of G(s) with K as the variable parameter is shown in Figure 17.

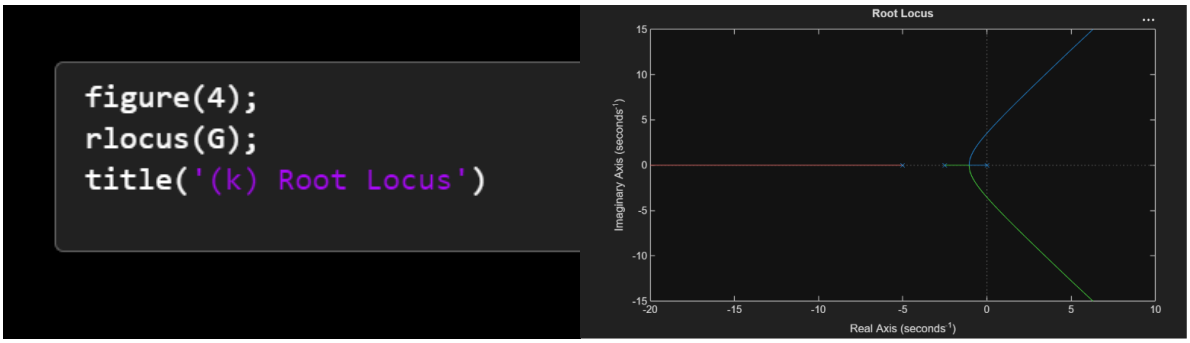


Figure 20: Root locus plot. Open-loop poles are at s = 0, −2.5, and −5.

Below is given the calculations for the angel in which the poles breakaway, the breakaway point and the asymptote

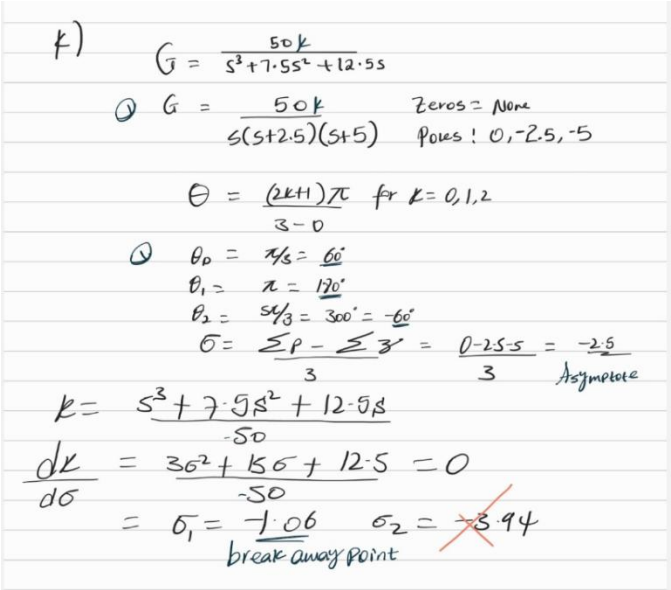


Figure 21: Root locus calculations

4.7.1. Root locus stability analysis (Question l):

- At K = 0, the closed-loop poles coincide with the open-loop poles: s = 0, −2.5, −5 (all in the LHP stable start).
- As K increases, two branches break away from the real axis, because there is no zeros between them, between s = 0 and s = −2.5 and progressively move toward the jω-axis.

- At K = Kcr ≈ 1.875, the complex branches cross the imaginary axis the system becomes marginally stable.

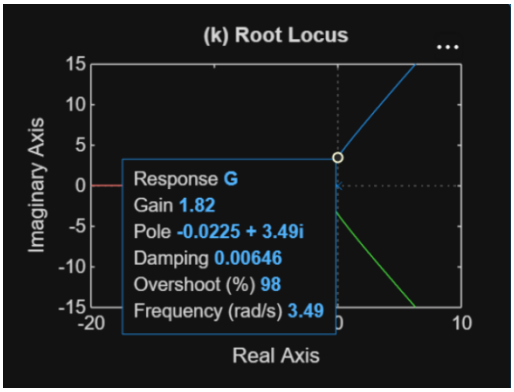


Figure 22: Gain at the imaginary axis

- For K > Kcr, the poles enter the Right-Half Plane and the system becomes unstable. For this reason the system is stable for 0 < K < 1.875.
- The calculations for the critical gain is given below:

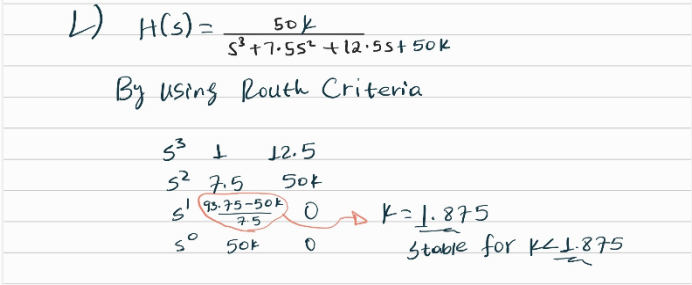


Figure 23: Routh criteria calculation

4.7.2. Critical Stability Gain (Question m):

This can be derived from the Routh–Hurwitz criterion applied to the characteristic equation $s^3 + 7.5s^2 + 12.5s + 50K = 0$:

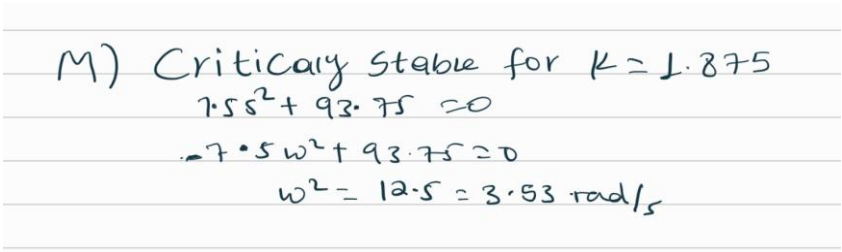


Figure 24: Frequency of the undamped system

At K = 1.875, the closed-loop poles are on the jω-axis at $\pm j \cdot \sqrt{12.5} = \pm j3.54$ rad/s and they produce an undamped oscillation at W = 3.53 rad/s. The step response is shown in Figure 22.

```
figure(5);
Hcr = feedback(1.875*G, 1);
step(Hcr)
```

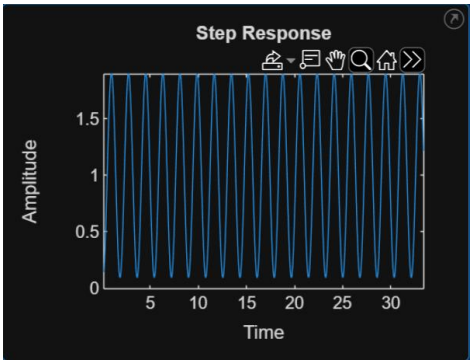


Figure 25: Step response at the critical gain K = 1.875

4.8. Gain for Critically Damped Response (ζ = 1) (Question N):

The point in the root locus of the transfer function where it is said to be d=critically damped is when the two poles meet just before breakaway. At this point the poles are real and equal, giving a critically damped response (ζ = 1). The breakaway point is -1.06 rad/s.

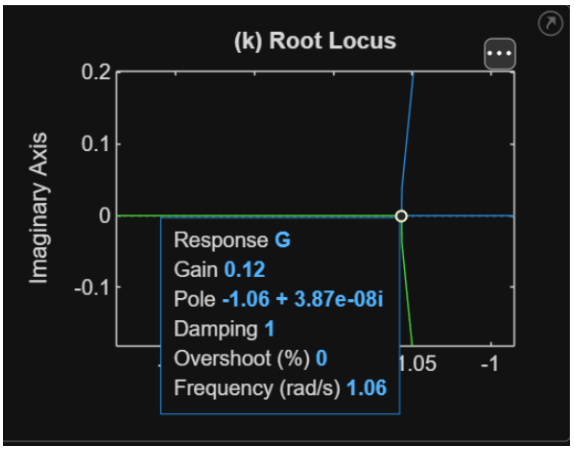


Figure 26: Gain at the point of break away

The root s = −1.06 lies between open-loop poles at 0 and −2.5 (on the real-axis portion of the locus), confirming it is the breakaway point.

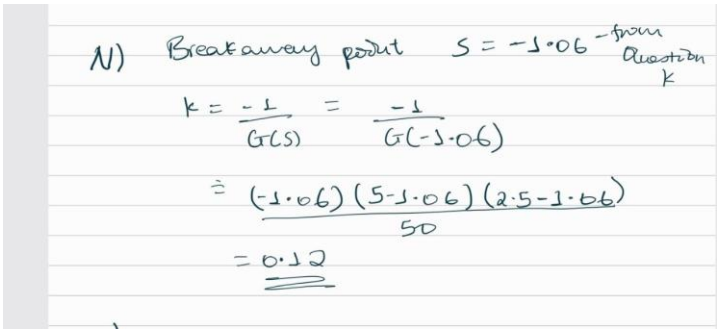


Figure 27: Breakaway gain calculation

The step response at K = 0.12 is shown in Figure 25. The response is critically damped no overshoot, with the fastest possible settling for a purely real-pole system.

```
figure(6);
Hcr_g = feedback(0.12*G, 1);
step(Hcr_g)
```

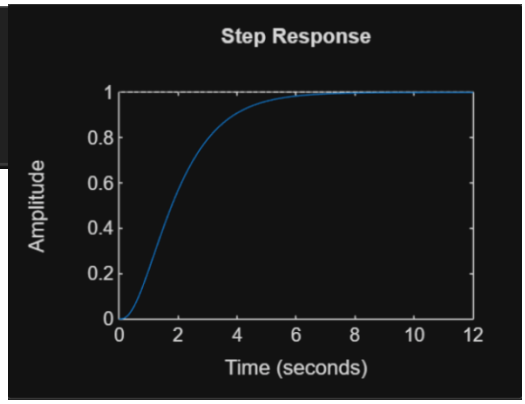
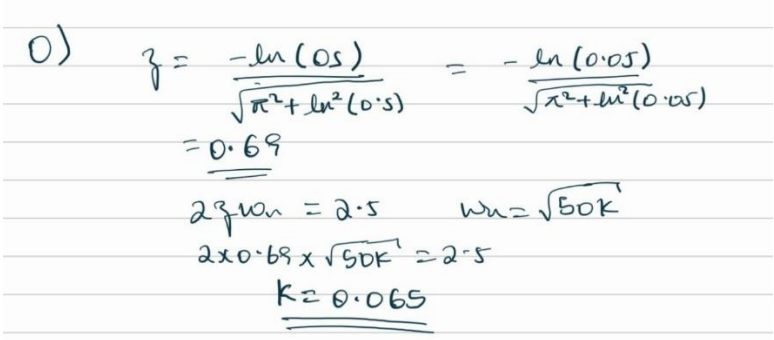


Figure 28: Step response at K = 0.12 Critically damped response

4.9. Gain for 5% Overshoot (Question O):

For the sake of computation complexity, we dropped the pole furthest from the imaginary axis as it decays faster than the rest. After that we used the formular for overshoot percentage on the resulting second order transfer function.



In Figure below, there is a significant difference between calculated and MATLAB. The calculated uses approximation from the dominant poles while the MATLAB is most accurate one.

```
% Approximate Calculation Using Dominant Poles
G_dom = tf(10,[1 2.5 0])
zeta_5 = -log(0.05) / sqrt(pi^2 + log(0.05)^2);
wn_5   = 2.5 / (2*zeta_5);
K_5os  = wn_5^2 / 10;

H_5os = feedback(K_5os * G, 1);
S_5os   = stepinfo(H_5os);
```

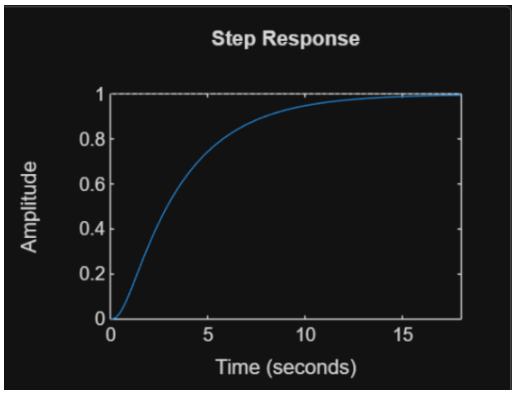


Figure 29: Step response for gain = 0.0656 for the original transfer function

By using root locus we can obtain the most accurate gain for 5% OS.

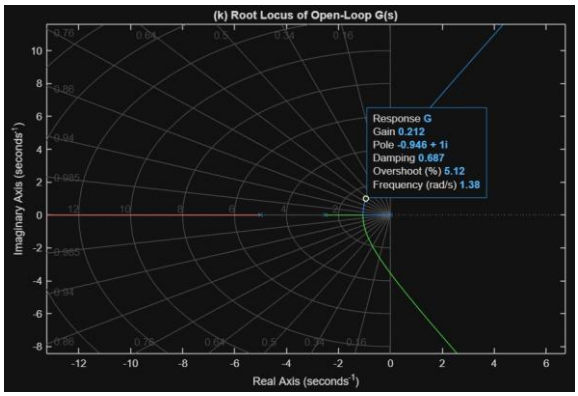
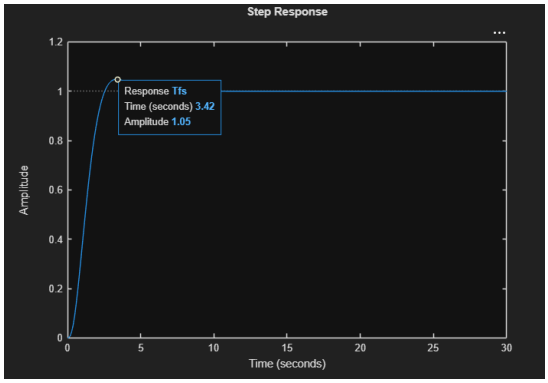


Figure 30: 5% OS gain

```
Gds = tf([1], [1 1.5]);
Gps = tf([1], [0.2 1]);
G_in_s = feedback(Gds,1);
LoopShaping_s = tf([10],[1 0]);
G_fd_s = series(Gps,series(G_in_s, LoopShaping_s));

Tfs = feedback(0.212*G_fd_s, 1);
figure(1);
step(Tfs)%question O
```

Figure 31: Actual 5% OS response



4.10. Gain Compensator: PM + 10° (Question p):

- Target PM = PM_current + 10° = 18.1° + 10° = 28.1°
- Required phase at the new crossover: ∠G(jω_new) = −180° + 28.1° = −151.9°

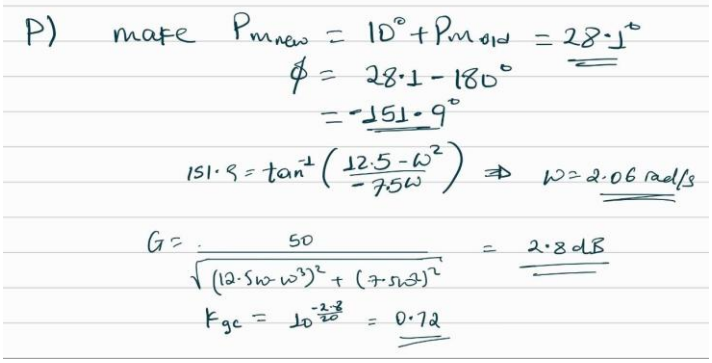


Figure 32: Calculation for 10 degree compensation

```
%% Question (p): Gain Compensator Design

PM_p = Pm + 10              % New target PM = current PM + 10°

% Find corresponding phase angle on Bode plot
Y_p = -180 + PM_p           % Target phase = -180° + PM_target

[~, idx_p] = min(abs(phase - Y_p));   % Index where phase ≈ Y_p

% Read magnitude at that frequency (dB)
mag_dB_p = 20 * log10(mag(idx_p))     % |G(jω)| in dB at target frequency

%Compute gain K to bring magnitude to 0 dB at that point
K_gc = 10^(-mag_dB_p / 20)            % K = 10^(-|G|_dB / 20)

H_gc = feedback(K_gc * G, 1)          % Closed-loop with gain compensator
```

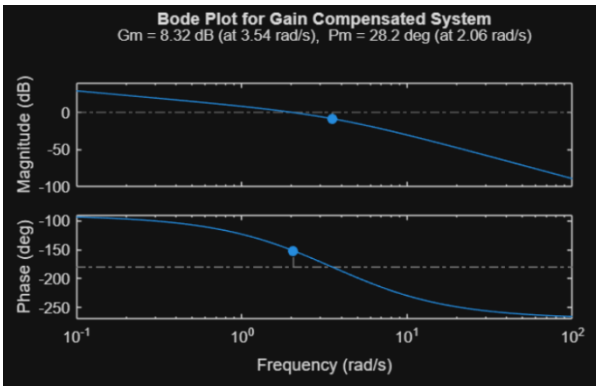


Figure 33: 10 degree compensated system

4.10.1. Original vs Gain-Compensated System (Question q):

Table 3 compares the key performance metrics of the original system (K = 1) with the gain-compensated system (K = 0.72). Figure 31 shows the corresponding step responses.

Specification	Original (K=1)	Gain Comp (K=0.72)
Rise Time (s)	0.465	0.57
Settling Time (s)	9.757	7.234
Overshoot (%)	59.428%	45.087%
Steady-State Error	0	≈ 0
Phase Margin (deg)	18.1°	28.1°
Gain Margin (dB)	5.47 dB	8.26 dB

Table 3: Performance comparison: Original vs. Gain-Compensated system.

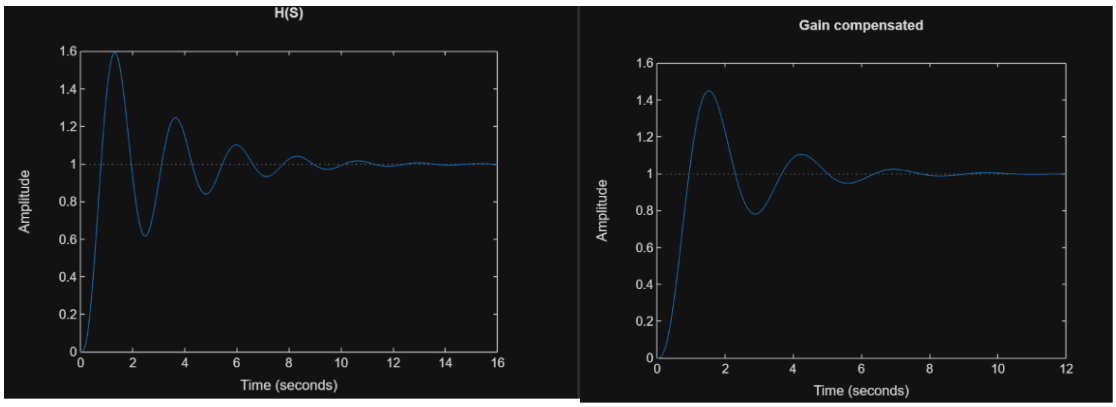


Figure 34: Step response comparison.

The gain compensator reduces the loop gain, which shifts the gain crossover to a lower frequency and increases the phase margin. However, this comes at the cost of a slower response (increased rise and settling time) since the bandwidth is reduced.

5. Phase Lead Compensator (Question R1):

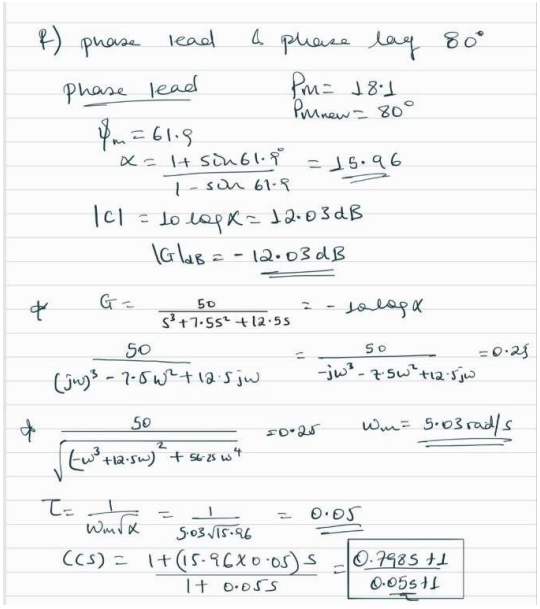


Figure 35: Phase lead compensator calculations

A phase-lead compensator adds positive phase at the gain crossover frequency, directly improving the phase margin.

```
PM_W = 80; % Desired phase margin  [degrees]
[mag, phase, W_out] = bode(G);

mag = squeeze(mag); % Convert 3-D array → column vector (linear)
mag_dB = 20*log10(mag); % Convert to dB

phase = squeeze(phase); % Phase in degrees

phi = PM_W - Pm  % ϕ = PM_desired − PM_current  [degrees]

alpha = (1+sind(phi))/(1-sind(phi)) % Calculate alpha, alpha > 1 for lead

G_dB = -10*log10(alpha) % Target magnitude at W_m (compensator adds −G_dB correction)

[~,index_w] = min(abs(mag_dB - G_dB)); % Find frequency W_m where |G(jω)|_dB ≈ G_dB

W_m = W_out(index_w % New gain crossover frequency  [rad/s]

tau = 1/(W_m * sqrt(alpha)) % τ = 1 / (W_m · √α)

G_lead = tf([alpha*tau 1],[tau 1]) % C_lead(s) = (ατs+1)/(τs+1)

Gc_lead = series(G,G_lead) % Open-loop with lead: G·C_lead

margin(Gc_lead)
[Gm_l,Pm_l,  Wcg_l, Wcp_l] = margin(Gc_lead); % Store compensated margins
```

Figure 36: Phase lead compensator MATLAB implementation

The resulting compensator from MATLAB is Gc_lead = $\frac{1+0.768s}{1+0.04802s}$, yielding an achieved phase margin of approximately 44.3°.

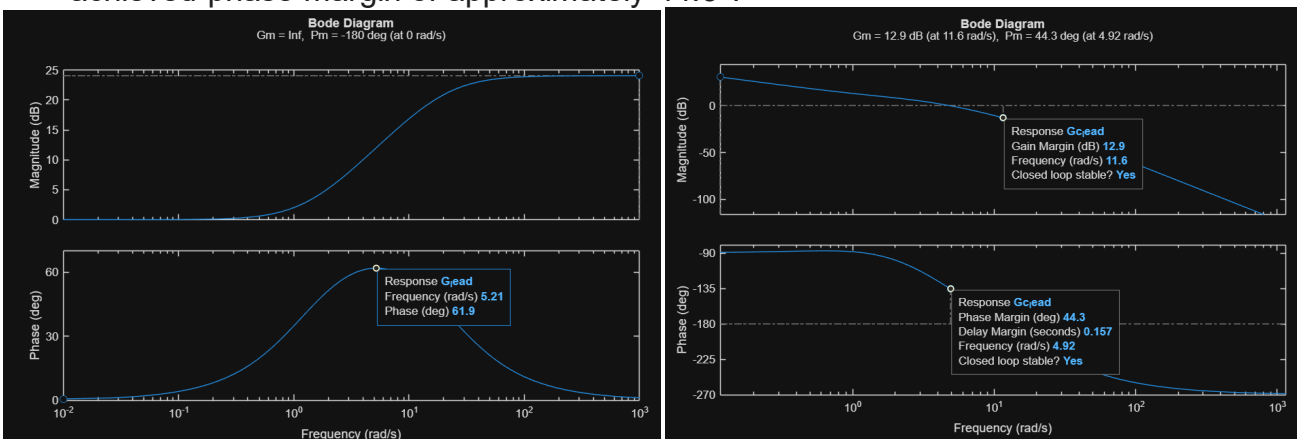


Figure 37: phase lead compensator Bode plot and phase lead compensated system

5.1. Phase Lag Compensator (Question R2)

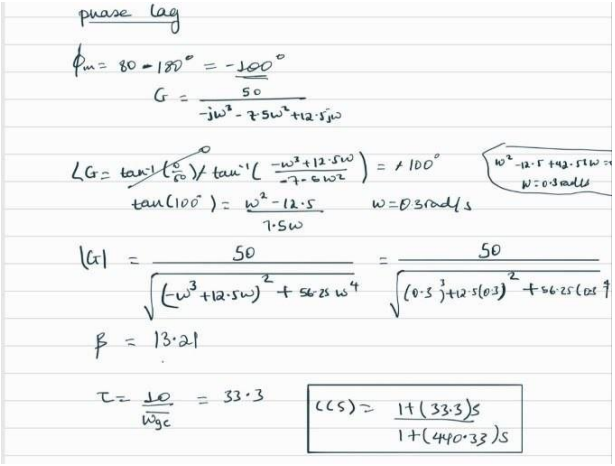


Figure 38: Phase lag calculations

A phase-lag compensator attenuates the open-loop gain at high frequencies without directly adding phase. By shifting the gain crossover to a lower frequency (where the original phase is less negative), the phase margin is improved indirectly.

```
pk = -180 + PM_W % Target phase value  [degrees]

[~, index_lag] = min(abs(phase - pk));

W_s = W_out(index_lag) % Desired new gain crossover frequency  [rad/s]

tau_lag = 10/W_s % τ_lag = 10 / W_s  (zero at 1/τ_lag ≪ W_s)

G_lag_dB = 20 * log10(mag(index_lag)) % |G(jW_s)|  in dB

Beta = 10^(G_lag_dB/20) % β = 10^(|G|_dB / 20);  pole at 1/(β·τ_lag)

G_lag = tf([tau_lag 1],[Beta * tau_lag 1])   % C_lag(s) = (τs+1)/(βτs+1)

Gc_lag = series(G,G_lag) % Open-loop with lag: G·C_lag

margin(Gc_lag)
% Calculate the phase margins for the lead and lag compensators

[Gm_lg,Pm_lg,  Wcg_lg, Wcp_lg] = margin(Gc_lag); % Store compensated margins
```

Figure 39: Phase Lag MATLAB implementation

The resulting compensator from MATLAB is Gc_lag = $\frac{1+36.41s}{1+526.3s}$,, yielding an achieved phase margin of approximately 75.3°.

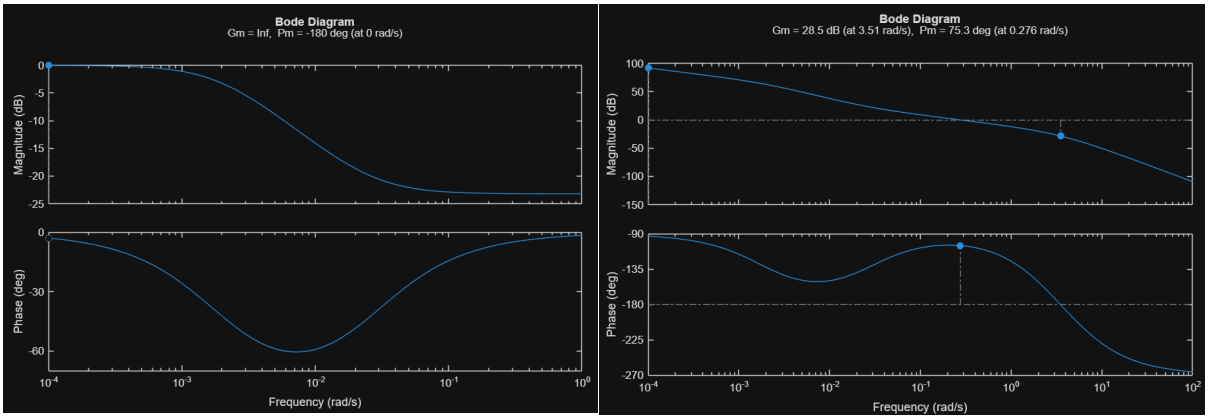


Figure 40: Phase Lag Bode plot and Phase Lag compensated system

5.1.1. Comparison: Original, Lead and Lag Compensators (Question s)

Table 4 compares all three systems. Figure 14 overlays the three step responses.

Table 4: Performance comparison: Original, Phase-Lead, and Phase-Lag compensators.

Specification	Original	Lead	Lag
Rise Time (s)	0.46472	0.25769	5.3935
Settling Time (s)	9.757	1.4587	60.843
Overshoot (%)	59.428	20.252	6.8876
Steady-State Error	0	5.55e-16	-0.006
Phase Margin (deg)	18.078°	44.331°	75.253°
Gain Margin (dB)	5.46	12.933	28.526

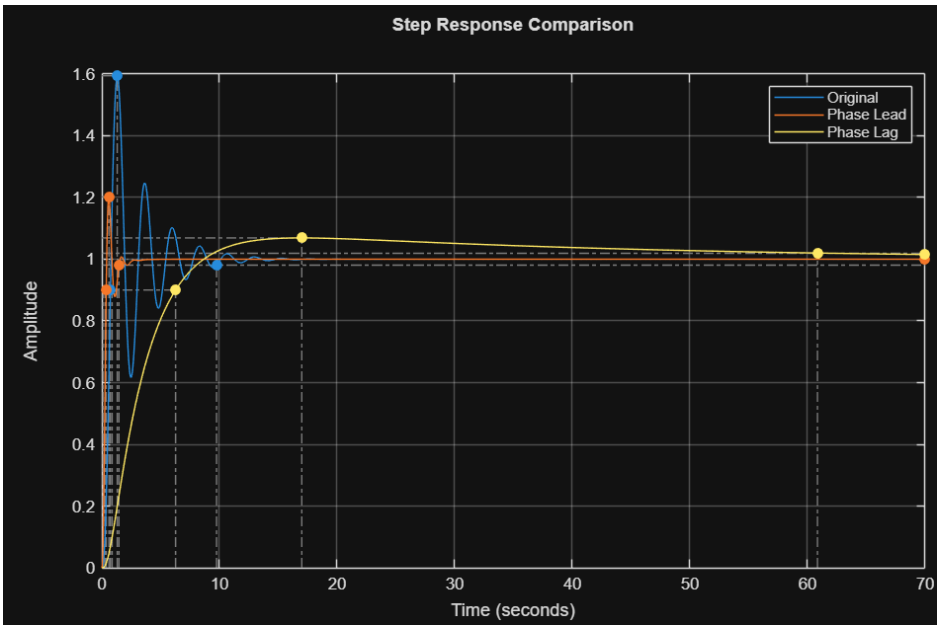


Figure 14. Step response comparison of all three designs. The lead compensator achieves faster transient response; the lag compensator improves stability at the cost of slower speed.

Key observations

- **Phase-Lead Compensator:** The lead compensator improved transient performance significantly, it reduced overshoot from 59.4% to 20.3% and settling time from 9.76 s to 1.46 s. However, the 80° phase margin target was not met, only 44.3° was achieved. Even though we even tried the standard 6° correction factor it failed to achieve our desired phase margin, actually reducing the it below the predicted value. This shows the limitation of phase lead design and suggests the compensator alone is insufficient for this system.

- **Phase-Lag Compensator:** The lag compensator achieved a phase margin of 75.3°, close to the 80° target. While stability improved and overshoot dropped to 6.9%, the transient response showed significance slowdown, settling time grew from 9.76 s to 60.8 s. This is consistent with lag compensator theory and highlights the fundamental trade-off: improved stability and steady-state behavior at the expense of speed

5.2. PID Controller Ziegler-Nichols Method (Question T)

This is because a PID controller offers all three effects, i.e. Proportional control (reducing rise time), Integral control (removing steady state error) and Derivative control (overshoot reduction).

Step 1: Determine Kcr and Pcr

The critical gain Kcr is the gain where the system becomes marginally stable (poles on jω axis).

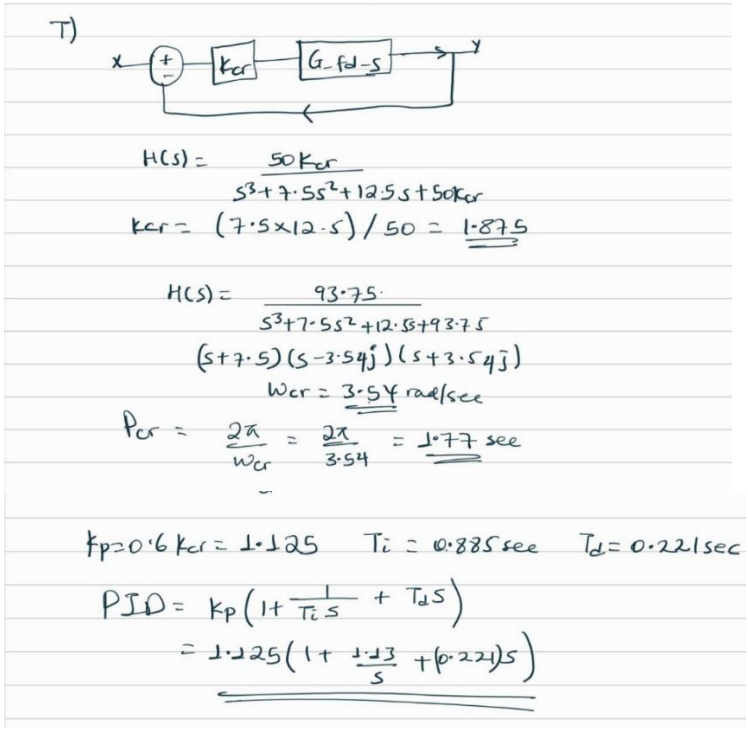


Figure 41: PID parameter calculation

Step 2: Apply Ziegler–Nichols PID Table

Parameter	Formula	Calculation	Value
Kp	0.6·Kcr	0.6 × 1.875	1.125
Ti	0.5·Pcr	0.5 × 1.775	0.888 s
Td	0.125·Pcr	0.125 × 1.775	0.222 s

Table 5: Ziegler–Nichols PID tuning parameters.

Step 3: PID Transfer Function

The PID controller is implemented in MATLAB using two methods:

```
1) C_PID = tf([Kp*Td Kp Kp/Ti],[1 0]);
2) C_PID_2 = pid(Kp,ki,Kd);
```

The achieved stability margins of the PID-controlled system are PM = 24.8° and GM = inf gain.

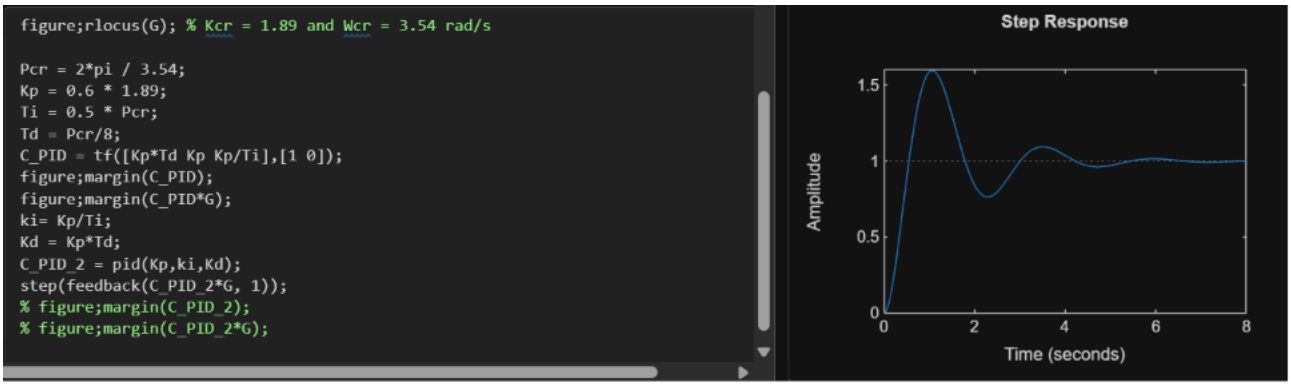


Figure 42: Step response of the PID-controlled system using Ziegler–Nichols tuning.

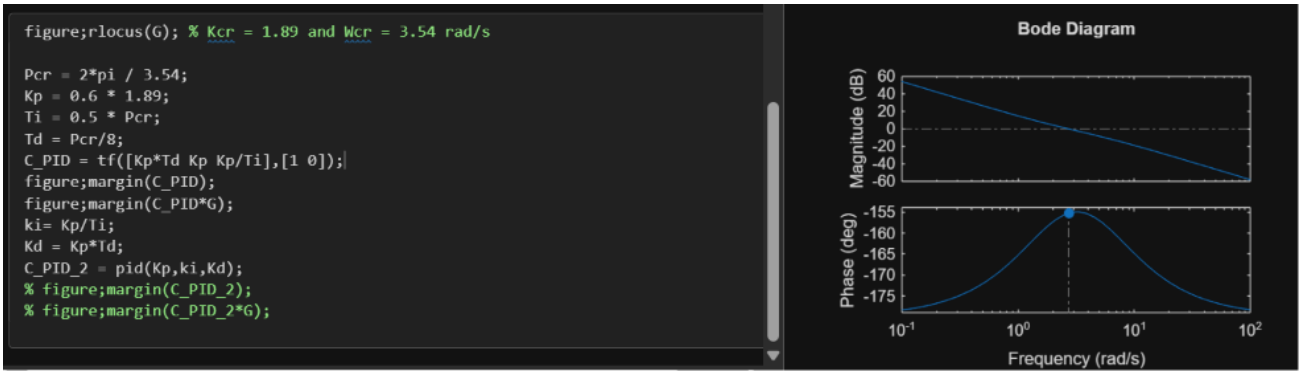


Figure 43: Bode plot of the PID open-loop system Gc_PID·G(s).

5.2.1. Original vs PID (Question u)

Specification	Original	PID (ZN)
Rise Time (s)	0.4647	0.3753
Settling Time (s)	9.7570	5.1420
Overshoot (%)	59.4283	59.5470
Steady-State Error	0	0
Phase Margin (deg)	18.1°	24.8
Gain Margin (dB)	5.47	inf

Table 6: Performance comparison: Original vs. PID Controller.

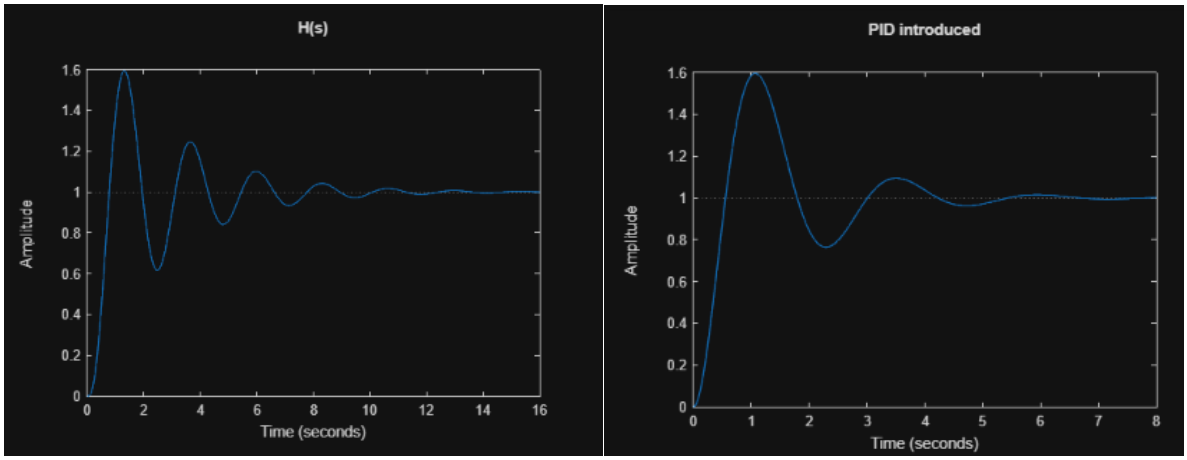


Figure 44: Step response comparison of the original and PID-controlled systems.

6. Results and Engineering Discussion

6.1. Summary of All Designs

Table 7 consolidates all key metrics across the original and compensated systems for direct comparison.

Specification	Original	Gain Comp	Lead	Lag	PID
Rise Time (s)	0.46472	0.57	0.25769	5.3935	0.3753
Settling Time (s)	9.757	7.234	1.4587	60.843	5.1420
Overshoot (%)	59.428	45.087%	20.252	6.8876	59.5470
SS Error	0	≈ 0	5.55e-16	-0.006	0
PM (deg)	18.078°	28.1°	44.331°	75.253°	24.8
GM (dB)	5.46	8.26 dB	12.933	28.526	inf

Table 7: Full performance summary across all designs.

Theory vs Simulation

Dominant Pole Approximation Results were compared against results from MATLAB simulation and showed very good agreement on all major parameters. The approximated value of 64.7% was a very good estimate of the simulated value of 59.4% in section 4.3.1. The difference is due to the fact that the third closed-loop pole is not taken into account resulting in a second order approximation. Time domain predictions for rise time, peak time and settling time are each about 10-15% less than those predicted by the simulation. This is a reasonable tolerance, given the approximate nature of this method.

There was a good agreement between the analytical check using the Routh-Hurwitz method, and the numeric margin() function regarding the Gain Margin. Therefore, there appears to be a good basis to believe that Dominant Pole Analysis can provide a useful, easy-to-use method for predicting the behavior of multi-order systems (i.e., greater than two) without having to solve them numerically.

Trade-offs in Engineering

- ### Gain Compensation
  The gain compensation has provided a means to reduce overshoot and improve stability margins through simply reducing the overall gain. As such, this provides the simplest way to implement a design. Since gain compensation does not add phase to the loop, the potential for improvement is therefore limited. In addition, since we have decreased the gain, we will decrease our bandwidth (i.e., slow down our transient response). We should think of gain compensation as a "first-step" or conservative approach rather than a final answer.

- ### Lead Phase Compensation
  This method gave us the smallest rise-time and settling-time. Lead compensation produces a direct injection of phase at the crossover frequency that produces improvements in both overshoot and stability margins. Our primary trade off is that we did not achieve the desired phase margin in one design cycle due to the shift in crossover frequency caused by insertion of the compensator.

- **Lag Compensator**

This compensator provided the best stability margins and the smallest overshoot among all the compensators. However, this is done at the expense of an extremely long settling time, such that the closed loop system will be much slower than the original system. Although conceptually reasonable, such extreme slow down may make this design impractical in practice when used in pitch angle control.

- **PID Controller**

The ZN-tuned PID provided a very reasonable design with fast settling compared to the original system, no steady-state error, and adequate gain margin. However, the downside to this design is its high overshoot, which is typical of any Ziegler-Nichols PID tuning because of the emphasis on disturbance rejection over minimizing overshoots. Since the original system already provides zero steady-state error due to being Type 1, the only additional benefit from using PID in this case is improved transient performance..

Limitations

- The time-invariant system that was analyzed in all parts of this paper lacks nonlinearities, actuator saturation effects, and time-varying wind disturbances which could affect the performance in the real world.
- The dominant pole approximation entails some error by ignoring the third closed-loop pole. Even though this approximation worked fine in our case, in other cases, it could lead to bigger errors depending on how close to each other the poles were.
- The Ziegler-Nichols method gives just an initial guess of PID parameters, and always results in high gain values. Tuning of the controller would be necessary to obtain better results.
- The phase-lead compensation is affected by any shift in the crossover frequency. Therefore, the calculated phase margin cannot be considered accurate enough since an iterative procedure would be needed.
- Very large delay times of a lag compensator would not be suitable for controlling aerodynamic processes.

7. Conclusion

The objective of this study has been a comprehensive analysis along with a compensator design for the pitch angle control system of the wind turbines. From the subsystem models, the transfer functions have been derived both in the time domain and the frequency domain. It was found that the system without any compensation had no steady state errors but suffered from considerable overshoot and inadequate phase margin necessitating some kind of compensation.

A total of four compensations methods have been analyzed for comparison, including gain compensation, phase lead compensator, phase lag compensator, and the PID controller. These compensators address some of the limitations of the initial system through different methodologies and compromises. In terms of transients speed, the lead compensator is the most appropriate one while the lag compensator provides better stability margins. The PID controller provides a good compromise of all parameters. None of these systems can optimize all the criteria together. This demonstrates the nature of control system design where every performance enhancement is associated with a compromise somewhere else.

In terms of the wind turbine pitch angle control system, the appropriate solution will be a well-tuned PID controller or perhaps an improved version of the lead compensation scheme.

References

[1] S. Baburajan, "Pitch Control of Wind Turbine through PID, Fuzzy and Adaptive Fuzzy-PID Controllers," M.S. thesis, Dept. Electrical Engineering, Rochester Institute of Technology, NY, 2017.

[2] National Instruments, "Wind Turbine Control Methods," White Paper, 2008. [Online]. Available: https://www.ni.com/en-lb/innovations/white-papers/08/wind-turbine-control-methods.html

[3] "Wind Turbine Adaptive Control," DSCL, University of Connecticut. [Online]. Available: https://dscl.uconn.edu/blade-adaptive-control/

[4] T. Corke, "Wind Turbine Control," University of Notre Dame. [Online]. Available: https://www3.nd.edu/~tcorke/w.WindTurbineCourse/WindTurbineControl_Presentation.pdf

[5] K. Ogata, Modern Control Engineering, 5th ed. Prentice Hall, 2010.

[6] R. C. Dorf and R. H. Bishop, Modern Control Systems, 13th ed. Pearson, 2017.