Table of Contents

LAG COMPENSATOR DESIGN IN FREQUENCY DOMAIN 1
Example 1: System G0(s) = 1/[s(s+1)(0.5s+1)] 1
Step 1: Satisfy Steady-State Error Requirement 1
Step 2a: Bode Plot Analysis 1
Step 2b: Enhanced Bode Plot Analysis with Phase Margin Targeting 4
Step 2.5: Comprehensive Bode Analysis with Design Targets 6
Step 3: Lag Compensator Design 8
Design Validation Table 11
Step 4: Verification 12
Display Results 14

LAG COMPENSATOR DESIGN IN FREQUENCY DOMAIN

```
% 1. The primary function of the lag compensator is to provide attenuation
%    in the high-frequency range to give a system sufficient phase margin.

% 2. The phase-lag characteristic is of no consequence in lag compensation.

% 3. Lag compensator, being, basically, a low-pass filter, yields an
%    appreciable improvement in steady-state accuracy at the expense of
%    increasing the transient response time.

% 4. It raises order of the system by 1, and will suppress
%    high-frequency noise effects.

% We will look at two examples
```

Example 1: System G0(s) = 1/[s(s+1)(0.5s+1)]

```
% Design Requirements:
% 1. Steady-state error to ramp (□(∞))_ramp= ≤ 20%
% 2. Gain Margin K_g ≥ 10 dB
% 3. Phase Margin γ ≥ 40°
```

Step 1: Satisfy Steady-State Error Requirement

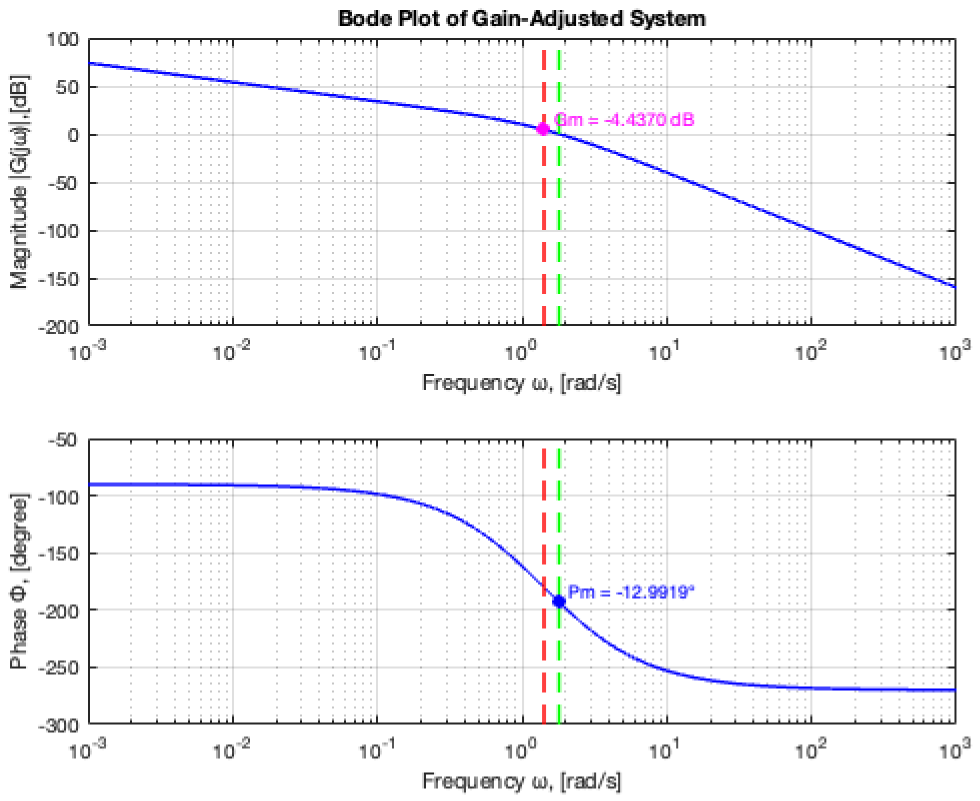
```
s = tf('s');
G0 = 1/(s*(s+1)*(0.5*s+1));
Kv_current = dcgain(s*G0);  % Kv_current = Limit s□0 (sGo(s)) = 1
Kv_required = 1/0.2;        % Kv_required = 1 / (□(∞))_ramp = 5
K = Kv_required/Kv_current; % K = Kv_required / Kv_current = 5
```

Step 2a: Bode Plot Analysis

```
% Generating Bode Plot for the gain-adjusted transfer function and reading
% the current gain and phase margins as well as crossover frequencies.
% Variables:
% 1. w = frequency
% 2. Gm = Gain Margin , GmdB = Gain Margin in decibels
% 3. Pm = Phase Margin
% 4. Wcp = Phase Crossover Frequency
% 5. Wcg = Gain Crossover Frequency

G_scaled = K*G0; % Adjust the transfer function with gain from step 1.
w = logspace(-3,3,1000); % Adjust the frequency scale to designer's choice.


% Extract magnitude, phase, and frequency information from MATLAB's
% inbuilt bode generator function.

[mag,phase,wout] = bode(G_scaled,w);

mag = squeeze(mag);
phase = squeeze(phase);

% Calculate Stability Margins
[Gm,Pm,Wcp,Wcg] = margin(G_scaled);
GmdB = 20*log10(Gm);

figure;
% Plot Bode magnitude
subplot(2,1,1);
semilogx(wout,20*log10(mag),'b','LineWidth',1.5); grid on;
xlabel('Frequency ω, [rad/s]');
ylabel('Magnitude |G(jω)|,[dB]')
title('Bode Plot of Gain-Adjusted System')

% Plot Bode phase
subplot(2,1,2);
semilogx(wout, phase, 'b', 'LineWidth', 1.5); grid on;
xlabel('Frequency ω, [rad/s]');
ylabel('Phase Φ, [degree]');


% Draw vertical lines at Wcp and Wcg
subplot(2,1,1); % Magnitude plot
hold on;
yl = ylim;
plot([Wcp Wcp], yl, 'r--', 'LineWidth', 1.5); % Phase crossover (red)
plot([Wcg Wcg], yl, 'g--', 'LineWidth', 1.5); % Gain crossover (green)
hold off;

subplot(2,1,2); % Phase plot
hold on;
yl = ylim;
plot([Wcp Wcp], yl, 'r--', 'LineWidth', 1.5); % Phase crossover (red)
plot([Wcg Wcg], yl, 'g--', 'LineWidth', 1.5); % Gain crossover (green)
hold off;
```

```
% Annotate Gain Margin on magnitude plot
subplot(2,1,1);
hold on;
plot(Wcp, -GmdB, 'mo', 'MarkerFaceColor', 'm');
text(Wcp, -GmdB, sprintf('  Gm = %.4f dB', GmdB), 'Color', 'm', ...
    'VerticalAlignment', 'bottom', 'FontSize', 9);
hold off;

% Annotate Phase Margin on phase plot
subplot(2,1,2);
hold on;
plot(Wcg, -180+Pm, 'bo', 'MarkerFaceColor', 'b');
text(Wcg, -180+Pm, sprintf('  Pm = %.4f°', Pm), 'Color', 'b', ...
    'VerticalAlignment', 'bottom', 'FontSize', 9);
hold off;

% Display calculated values
disp(['Gain Margin (dB): ', num2str(GmdB)]);
disp(['Phase Margin (deg): ', num2str(Pm)]);
disp(['Phase crossover frequency (rad/s): ', num2str(Wcp)]);
disp(['Gain crossover frequency (rad/s): ', num2str(Wcg)]);
```

```
Warning: The closed-loop system is unstable.
Gain Margin (dB): -4.437
Phase Margin (deg): -12.9919
Phase crossover frequency (rad/s): 1.4142
Gain crossover frequency (rad/s): 1.802
```



Step 2b: Enhanced Bode Plot Analysis with Phase Margin Targeting

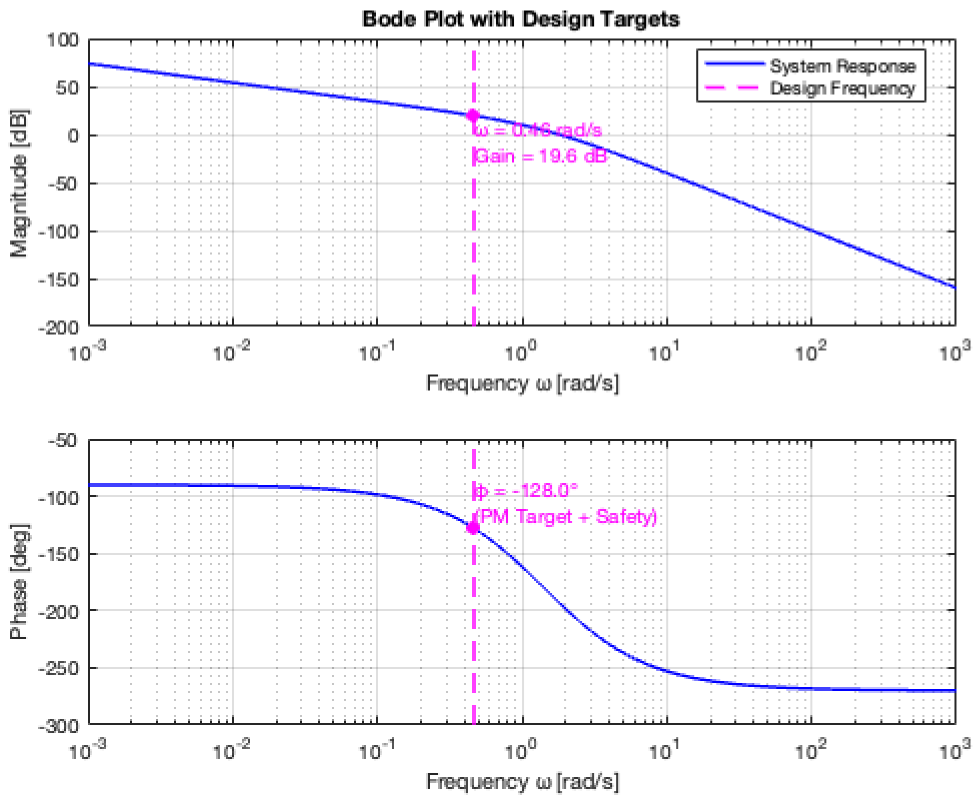
This section now includes explicit identification of the required phase margin frequency point for compensator design

```
% System definition
G_scaled = K*G0;  % Gain-adjusted system from Step 1
w = logspace(-3,3,1000);  % Extended frequency range for better resolution

% Bode data extraction
[mag,phase,wout] = bode(G_scaled,w);
mag = squeeze(mag);
phase = squeeze(phase);

% Calculate stability margins
[Gm,Pm,Wcp,Wcg] = margin(G_scaled);
GmdB = 20*log10(Gm);

% Phase margin design parameters
required_PM = 40;         % Design specification
safety_margin = 12;       % Maximum recommended safety factor
target_phase = -180 + required_PM + safety_margin;  % -128° phase target

% Find frequency where phase crosses target value
```

```
target_idx = find(phase <= target_phase, 1);
w_target = interp1(phase(target_idx-1:target_idx),
wout(target_idx-1:target_idx), target_phase);

% Find corresponding gain at target frequency
mag_target = interp1(wout, mag, w_target);
mag_target_dB = 20*log10(mag_target);

% Bode Plot Visualization with Design Targets
figure;

% Magnitude Plot
subplot(2,1,1);
semilogx(wout, 20*log10(mag), 'b', 'LineWidth', 1.5);
hold on;
plot([w_target w_target], ylim, 'm--', 'LineWidth', 1.5);  % Target frequency
line
plot(w_target, mag_target_dB, 'mo', 'MarkerFaceColor', 'm');  % Gain at target
grid on;
title('Bode Plot with Design Targets');
xlabel('Frequency ω [rad/s]');
ylabel('Magnitude [dB]');
legend('System Response', 'Design Frequency', 'Location', 'Best');

% Phase Plot
subplot(2,1,2);
semilogx(wout, phase, 'b', 'LineWidth', 1.5);
hold on;
plot([w_target w_target], ylim, 'm--', 'LineWidth', 1.5);  % Target frequency
line
plot(w_target, target_phase, 'mo', 'MarkerFaceColor', 'm');  % Phase at target
grid on;
xlabel('Frequency ω [rad/s]');
ylabel('Phase [deg]');

% Annotations and Margin Display
% Add phase margin target labels
subplot(2,1,1);
text(w_target, mag_target_dB, sprintf('ω = %.2f rad/s\nGain = %.1f dB',...
    w_target, mag_target_dB), 'VerticalAlignment','top', 'Color','m');

subplot(2,1,2);
text(w_target, target_phase, sprintf('φ = %.1f°\n(PM Target + Safety)',...
    target_phase), 'VerticalAlignment','bottom', 'Color','m');

% Display key parameters
disp(['Required Phase Margin (+safety): ',
num2str(required_PM+safety_margin), '°']);
disp(['Target Frequency (ω_target): ', num2str(w_target), ' rad/s']);
disp(['Current Gain at ω_target: ', num2str(mag_target_dB), ' dB']);
```

```
Warning: The closed-loop system is unstable.
Required Phase Margin (+safety): 52°
```

```
Target Frequency (ω_target): 0.46464 rad/s
Current Gain at ω_target: 19.5601 dB
```



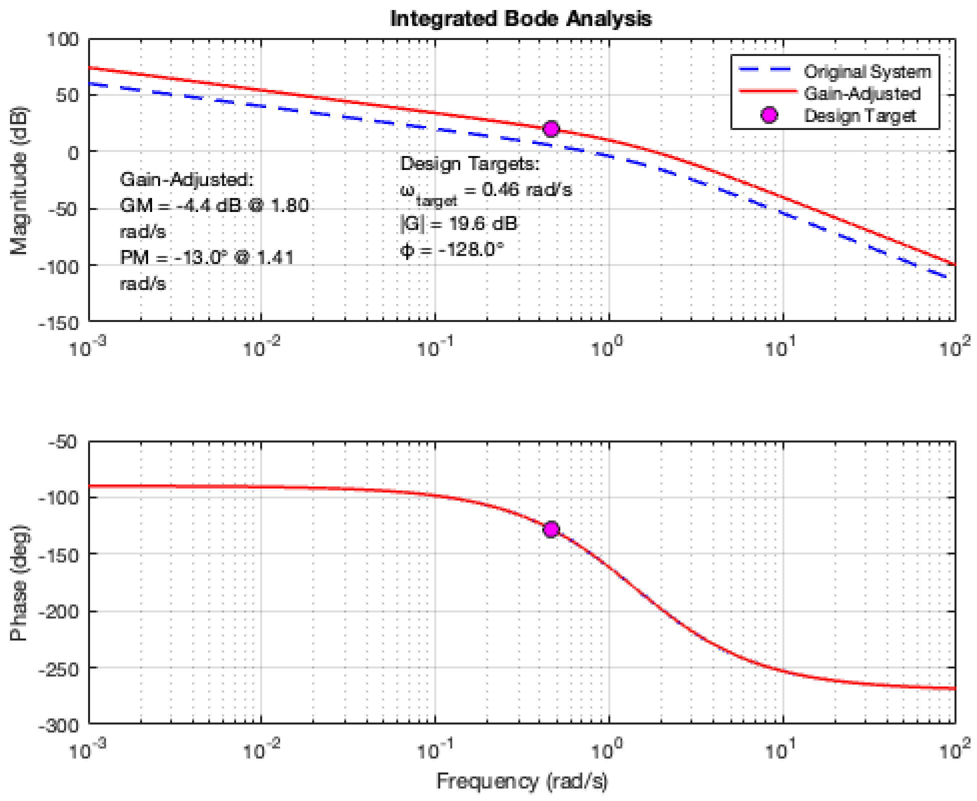
Step 2.5: Comprehensive Bode Analysis with Design Targets

Combines original system, gain-adjusted system, and design targets

```
% System definitions
%G0 = tf(1, conv([1 0], conv([1 1], [0.5 1]))); % Original system
G_scaled = 5*G0; % Gain-adjusted system from Step 1

% Frequency vector
w = logspace(-3, 2, 1000);

% Get Bode data for both systems
[mag0, phase0, ~] = bode(G0, w);
[magS, phaseS, wout] = bode(G_scaled, w);
mag0 = squeeze(mag0); phase0 = squeeze(phase0);
magS = squeeze(magS); phaseS = squeeze(phaseS);

% Calculate margins for both systems
[Gm0, Pm0, Wcp0, Wcg0] = margin(G0);
[GmS, PmS, WcpS, WcgS] = margin(G_scaled);
```

```
% Design parameters
required_PM = 40;
safety_margin = 12;
target_phase = -180 + required_PM + safety_margin;

% Find target frequency for gain-adjusted system
target_idx = find(phaseS <= target_phase, 1);
w_target = interp1(phaseS(target_idx-1:target_idx),
wout(target_idx-1:target_idx), target_phase);
mag_target = interp1(wout, magS, w_target);

% Create integrated visualization
figure;
tiledlayout(2,1);

% Magnitude plot
nexttile;
semilogx(w, 20*log10(mag0), 'b--', 'LineWidth', 1.5); hold on;
semilogx(wout, 20*log10(magS), 'r-', 'LineWidth', 1.5);
plot(w_target, 20*log10(mag_target), 'ko', 'MarkerFaceColor', 'm',
'MarkerSize', 8);
grid on;
title('Integrated Bode Analysis');
ylabel('Magnitude (dB)');
legend('Original System', 'Gain-Adjusted', 'Design Target',
'Location','northeast');

% Phase plot
nexttile;
semilogx(w, phase0, 'b--', 'LineWidth', 1.5); hold on;
semilogx(wout, phaseS, 'r-', 'LineWidth', 1.5);
plot(w_target, target_phase, 'ko', 'MarkerFaceColor', 'm', 'MarkerSize', 8);
grid on;
ylabel('Phase (deg)');
xlabel('Frequency (rad/s)');

% Add stability margin annotations

annotation('textbox', [0.15 0.5 0.2 0.28], 'String', ...
    sprintf('Gain-Adjusted:\nGM = %.1f dB @ %.2f rad/s\nPM = %.1f° @ %.2f rad/
s',...
    20*log10(GmS), WcgS, PmS, WcpS), 'EdgeColor','none');

annotation('textbox', [0.4 0.7 0.1 0.1], 'String', ...
    sprintf('Design Targets:\nω_{target} = %.2f rad/s\n|G| = %.1f dB\nϕ =
%.1f°',...
    w_target, 20*log10(mag_target), target_phase), 'EdgeColor','none');
hold off;
```

Warning: The closed-loop system is unstable.

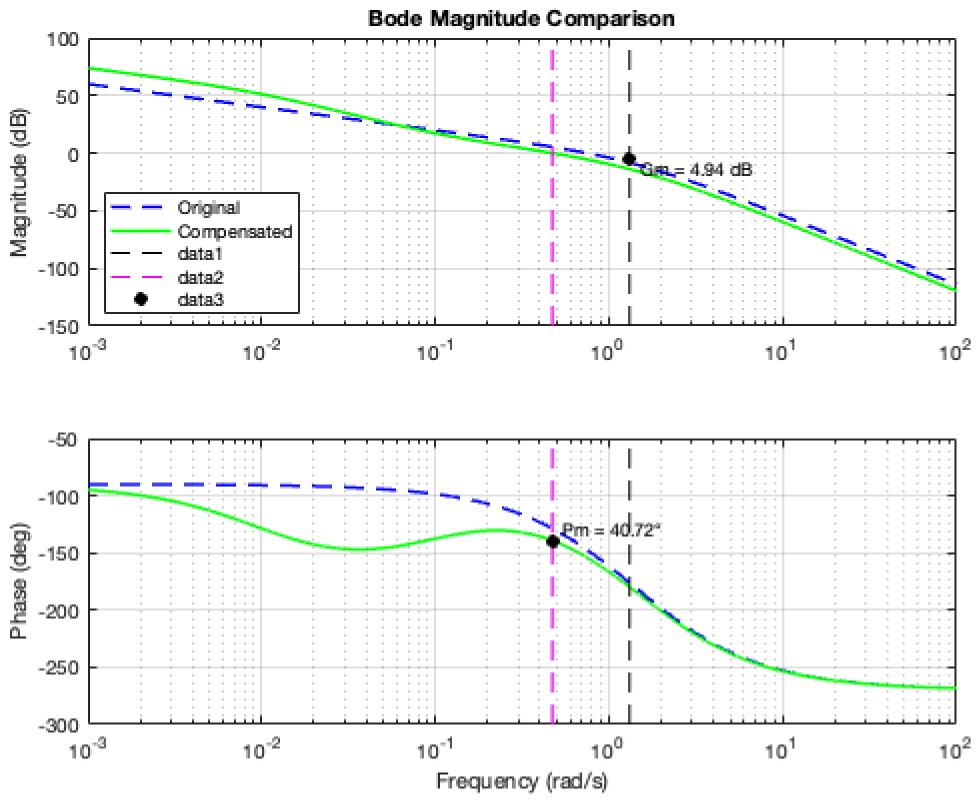


Step 3: Lag Compensator Design

```
% Compensator TF: Gc(s) = Kc*(1+T*s)/(1+β*T*s)
% Gc = Kc*(1 + T*s)/(1 + β*T*s);

% Find attenuation required at target frequency (ω_target)
attenuation_needed = -mag_target_dB;

% Calculate β using attenuation requirement
beta = 10^(-attenuation_needed/20);  % Solve 20log_{10}(1/β) = |
attenuation_needed|

z = 0.1; % zero of the lag compensator (z)
T = 1/z; % Time constant of the lag compensator.

% Calculate compensator gain Kc
Kc = K/beta;  % Maintains steady-state error constant

% Construct lag compensator transfer function [PDF Eq.1]
numerator = [T 1];
denominator = [beta*T 1];
C_lag = K * tf(numerator, denominator);

% Display Design Parameters
disp('===== LAG COMPENSATOR PARAMETERS =====');
```

```matlab
disp(['Target Frequency (ω_gcn): ' num2str(w_target) ' rad/s']);
disp(['Required Attenuation: ' num2str(attenuation_needed) ' dB']);
disp(['Calculated β: ' num2str(beta)]);
disp(['Time Constant T: ' num2str(T) ' seconds']);
disp(['Compensator Gain Kc: ' num2str(Kc)]);
disp(' ');
disp('Transfer Function: C_lag(s) = ');
C_lag
disp('===================================================');

% System definitions
G_comp = C_lag * G0; % Compensated system

% Calculate stability margins for all systems
[Gm0, Pm0, Wcp0, Wcg0] = margin(G0);
[GmC, PmC, WcpC, WcgC] = margin(G_comp);

[mag0, phase0, w0] = bode(G0, w);
[magC, phaseC, wout] = bode(G_comp, w);

mag0 = squeeze(mag0); phase0 = squeeze(phase0);
magC = squeeze(magC); phaseC = squeeze(phaseC);

figure;
% Magnitude plot
subplot(2,1,1);
semilogx(w, 20*log10(mag0), 'b--', 'LineWidth', 1.5); hold on;
semilogx(wout, 20*log10(magC), 'g-', 'LineWidth', 1.5);
grid on;
title('Bode Magnitude Comparison');
ylabel('Magnitude (dB)');
legend('Original', 'Compensated', 'Location', 'southwest');

% Add compensated system margins
yl = ylim;
plot([WcpC WcpC], yl, 'k--', 'LineWidth', 1.2); % Phase crossover
plot([WcgC WcgC], yl, 'm--', 'LineWidth', 1.2); % Gain crossover
plot(WcpC, -GmC, 'ko', 'MarkerFaceColor', 'k');
text(WcpC, -GmC, sprintf('  Gm = %.2f dB', GmC),...
    'Color', 'k', 'VerticalAlignment', 'top', 'FontSize', 9);
hold off;

% Phase plot
subplot(2,1,2);
semilogx(wout, phase0, 'b--', 'LineWidth', 1.5); hold on;
semilogx(wout, phaseC, 'g-', 'LineWidth', 1.5);
grid on;
xlabel('Frequency (rad/s)');
ylabel('Phase (deg)');

% Add compensated system margins
yl = ylim;
plot([WcpC WcpC], yl, 'k--', 'LineWidth', 1.2); % Phase crossover
plot([WcgC WcgC], yl, 'm--', 'LineWidth', 1.2); % Gain crossover
plot(WcgC, -180+PmC, 'ko', 'MarkerFaceColor', 'k');
text(WcgC, -180+PmC, sprintf('  Pm = %.2f°', PmC),...
    'Color', 'k', 'VerticalAlignment', 'bottom', 'FontSize', 9);
hold off;

% Display margin information
disp('=== Compensated System Margins ===');
disp(['Gain Margin: ', num2str(GmC), ' dB @ ', num2str(WcpC), ' rad/s']);
disp(['Phase Margin: ', num2str(PmC), '° @ ', num2str(WcgC), ' rad/s']);
```

```
===== LAG COMPENSATOR PARAMETERS =====
Target Frequency (ω_gcn): 0.46464 rad/s
Required Attenuation: -19.5601 dB
Calculated β: 9.5061
Time Constant T: 10 seconds
Compensator Gain Kc: 0.52598

Transfer Function: C_lag(s) =

C_lag =

   50 s + 5
  -----------
  95.06 s + 1

Continuous-time transfer function.
===================================================
=== Compensated System Margins ===
Gain Margin: 4.9354 dB @ 1.3159 rad/s
Phase Margin: 40.7241° @ 0.47285 rad/s
```



Design Validation Table

Calculate steady-state error constants

```
Kv_original = dcgain(s*G0);
Kv_scaled = dcgain(s*G_scaled);
Kv_comp = dcgain(s*G_comp);

% Precompute status indicators
pm_status = cell(1,3);
pm_status(:) = {'Fail'};
pm_status(PmC >= 40) = {'Pass'};

gm_status = cell(1,3);
gm_status(:) = {'Fail'};
gm_status(20*log10(GmC) >= 10) = {'Pass'};

sse_status = cell(1,3);
sse_status(:) = {'Fail'};
sse_status(1/Kv_comp <= 0.2) = {'Pass'};

disp('=== System Comparison Table ===');
fprintf('%-25s | %-8s | %-8s | %-8s | %-8s | %-8s\n',...
    'Parameter', 'Original', 'Gain-Adj', 'Compensated', 'Req', 'Status');
fprintf('%-25s | %-8.2f | %-8.2f | %-8.2f | %-8.2f | %-8s\n',...
    'Phase Margin (°)', Pm0, PmS, PmC, 40, pm_status{1});
```

```
fprintf('%-25s | %-8.2f | %-8.2f | %-8.2f | %-8.2f | %-8s\n',...
    'Gain Margin (dB)', 20*log10(Gm0), 20*log10(GmS), 20*log10(GmC), 10,
gm_status{1});
fprintf('%-25s | %-8.2f | %-8.2f | %-8.2f | %-8.2f | %-8s\n',...
    'Steady-State Error', 1/Kv_original, 1/Kv_scaled, 1/Kv_comp, 0.2,
sse_status{1});
fprintf('%-25s | %-8.2f | %-8.2f | %-8.2f | %-8s | %-8s\n',...
    'Bandwidth (rad/s)', bandwidth(feedback(G0,1)),...
    bandwidth(feedback(G_scaled,1)), bandwidth(feedback(G_comp,1)),...
    'N/A', 'N/A');
```

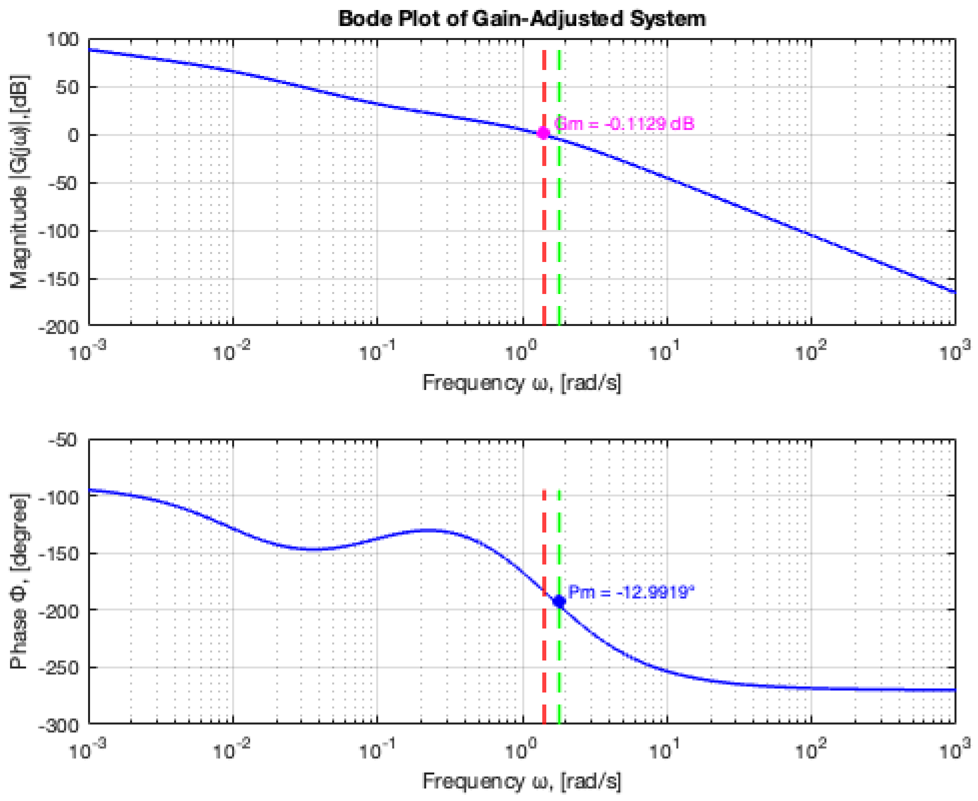
```
=== System Comparison Table ===
Parameter                 | Original | Gain-Adj | Compensated | Req      |
Status
Phase Margin (°)          | 32.61    | -12.99   | 40.72    | 40.00    |
Pass
Gain Margin (dB)          | 9.54     | -4.44    | 13.87    | 10.00    |
Pass
Steady-State Error        | 1.00     | 0.20     | 0.20     | 0.20     |
Pass
Bandwidth (rad/s)         | 1.26     | 2.52     | 0.85     | N/A      |
N/A
```

Step 4: Verification

```
G_comp = C_lag * G_scaled;
w = logspace(-3,3,1000);

% Stability Margins
[Gm_comp,Pm_comp,~,~] = margin(G_comp);

[mag,phase,wout] = bode(G_comp,w);

mag = squeeze(mag);
phase = squeeze(phase);

% Calculate Stability Margins
GmdB = 20*log10(Gm_comp);

figure;
% Plot Bode magnitude
subplot(2,1,1);
semilogx(wout,20*log10(mag),'b','LineWidth',1.5); grid on;
xlabel('Frequency ω, [rad/s]');
ylabel('Magnitude |G(jω)|,[dB]')
title('Bode Plot of Gain-Adjusted System')

% Plot Bode phase
subplot(2,1,2);
semilogx(wout, phase, 'b', 'LineWidth', 1.5); grid on;
xlabel('Frequency ω, [rad/s]');
ylabel('Phase Φ, [degree]');
```

```
% Draw vertical lines at Wcp and Wcg
subplot(2,1,1); % Magnitude plot
hold on;
yl = ylim;
plot([Wcp Wcp], yl, 'r--', 'LineWidth', 1.5); % Phase crossover (red)
plot([Wcg Wcg], yl, 'g--', 'LineWidth', 1.5); % Gain crossover (green)
hold off;

subplot(2,1,2); % Phase plot
hold on;
yl = ylim;
plot([Wcp Wcp], yl, 'r--', 'LineWidth', 1.5); % Phase crossover (red)
plot([Wcg Wcg], yl, 'g--', 'LineWidth', 1.5); % Gain crossover (green)
hold off;

% Annotate Gain Margin on magnitude plot
subplot(2,1,1);
hold on;
plot(Wcp, -GmdB, 'mo', 'MarkerFaceColor', 'm');
text(Wcp, -GmdB, sprintf('  Gm = %.4f dB', GmdB), 'Color', 'm', ...
    'VerticalAlignment', 'bottom', 'FontSize', 9);
hold off;

% Annotate Phase Margin on phase plot
subplot(2,1,2);
hold on;
plot(Wcg, -180+Pm, 'bo', 'MarkerFaceColor', 'b');
text(Wcg, -180+Pm, sprintf('  Pm = %.4f°', Pm), 'Color', 'b', ...
    'VerticalAlignment', 'bottom', 'FontSize', 9);
hold off;
```

```
Warning: The closed-loop system is unstable.
```



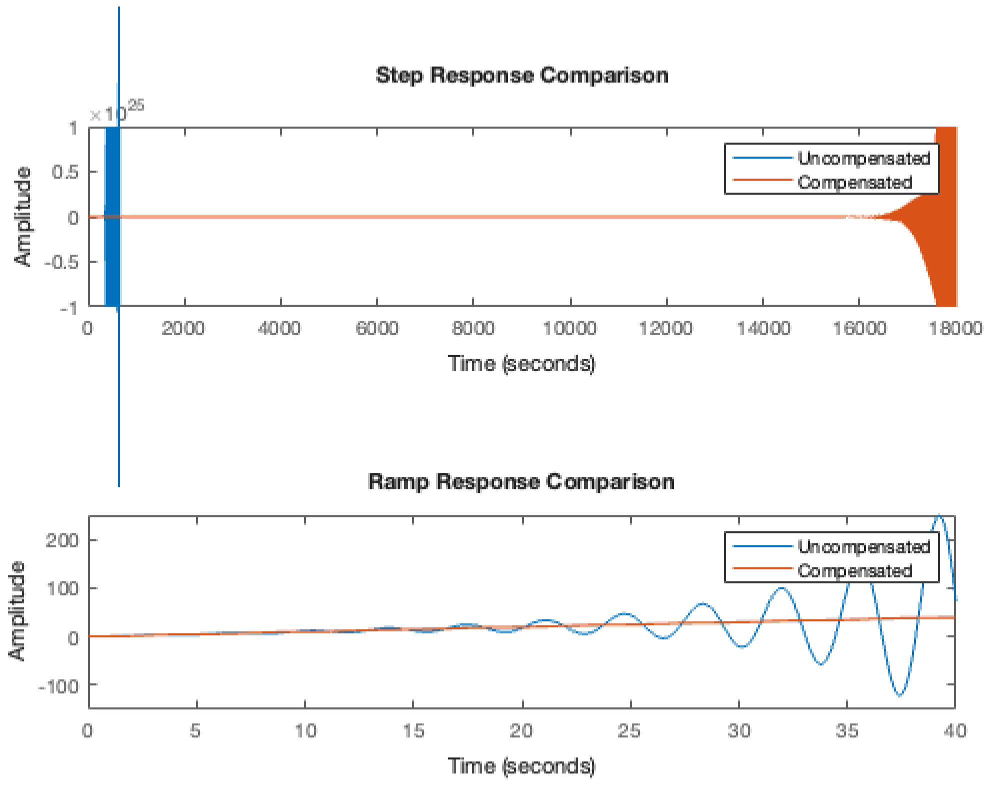
Display Results

Time Responses

```
figure
subplot(2,1,1)
step(feedback(G_scaled,1), feedback(G_comp,1))
legend('Uncompensated','Compensated')
title('Step Response Comparison')

subplot(2,1,2)
t = 0:0.1:40;
ramp = t;
lsim(feedback(G_scaled,1), feedback(G_comp,1), ramp, t)
legend('Uncompensated','Compensated')
title('Ramp Response Comparison')


fprintf('Example 1 Results:\n');
fprintf('Original Phase Margin: %.2f°\n', Pm);
fprintf('Compensated Phase Margin: %.2f°\n', Pm_comp);
fprintf('Steady-State Error Reduction: %.4f to %.4f\n',...
    1/Kv_current, 1/(Kv_current*beta));
```

```
Example 1 Results:
Original Phase Margin: -12.99°
```

```
Compensated Phase Margin: -0.33°
Steady-State Error Reduction: 1.0000 to 0.1052
```



Published with MATLAB® R2024b