SE 380 — HW 2

Bilal Khan
bilal2vec@gmail.com

November 12, 2023

Contents

1 1

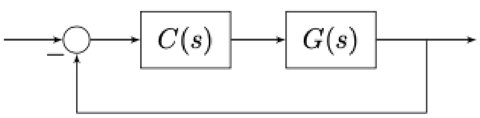


Figure 1: Control system in **1** and **2**

Consider the feedback control system where

$$G(s) = \frac{5}{(1+s)\left(1 + \frac{2\zeta}{\omega_n}s + \frac{s^2}{\omega_n^2}\right)}$$

With $\omega_n = 10$ and $\zeta = 0.1$.

Design a controller $C(s)$ such that the closed-loop system has the following specifications: The steady state error in response to a unit step function is $e_\infty \leq 0.05$, the gain margin $k_m \geq 10$dB, and the phase margin $\phi_m \geq 60°$.

We will design a controller with two parts: $C_1(s) = \mu/s^p$, to satisfy the steady state error requirement and we choose $p = 0$, and $C_2(s)$, to be a realizable controller that does not change the steady state behavior of the closed loop system to satisfy the gain and phase margin requirements.

$$\begin{aligned} y_\infty &= \frac{C_1(0)C_2(0)G(0)}{1 + C_1(0)C_2(0)C_3(0)} \\ &= \frac{\mu * 1 * 5}{1 + \mu * 1 * 5} \\ &= \frac{5\mu}{1 + 5\mu} \end{aligned}$$

For our steady state error requirement, we need $y_\infty \leq 0.05$, so manipulating this, we can see that we need $\mu \geq 3.8$. For simplicity, we choose $\mu = 4$.

For $C_2(s)$ to not change the steady state behavior of the closed loop system, we need $C_2(0) = 1$. We also need $C_2(s)$ to be realizable, so we choose to define it in the form $C_2(s) = \frac{L^\star(s)}{C_1(s)G(s)}$. The steady state gain of $G(s)$ is 5, so we need $L^\star(0) = 5$. For the controller to be realizable, it must have the same degree as $G(s)$, 3. For a phase margin of at least 60° and a gain margin of atleast 10 dB, we will choose an arbitrary crossover point ω_c for the magnitude path of the bode plot of the controller to hit zero such that these two requirements are satisfied. We will choose $\omega_c = 10^1$ for simplicity (Any choice of ω_c will work). For the margins to be appropriately large, it is safe to place one pole at least one decade before ω_c and the other two poles at least one decade after ω_c.

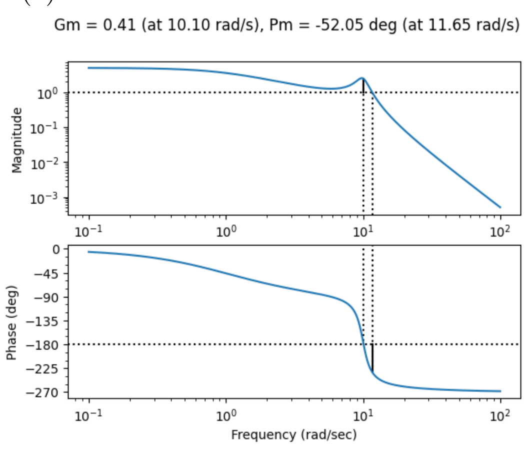
$$\begin{aligned}
C_2(s) &= \frac{L^\star(s)}{C_1(s)G(s)} \\
L^\star(s) &= \frac{5}{(1+\frac{s}{10^0})(1+\frac{s}{10^2})^2} \\
C(s) &= C_1(s)C_2(s) \\
&= C_1(s)L^\star(s)\frac{1}{G(s)} \\
&= 4.0\frac{5}{(1+\frac{s}{10^0})(1+\frac{s}{10^2})^2}\frac{(1+s)\left(1+\dfrac{2\zeta}{\omega_n}s+\dfrac{s^2}{\omega_n^2}\right)}{5} \\
&= 4.0\cdot\frac{(1+s)\left(1+\dfrac{2\zeta}{\omega_n}s+\dfrac{s^2}{\omega_n^2}\right)}{(1+\frac{s}{10^0})(1+\frac{s}{10^2})^2}
\end{aligned}$$

In code, we can verify that the controller satisfies the requirements:

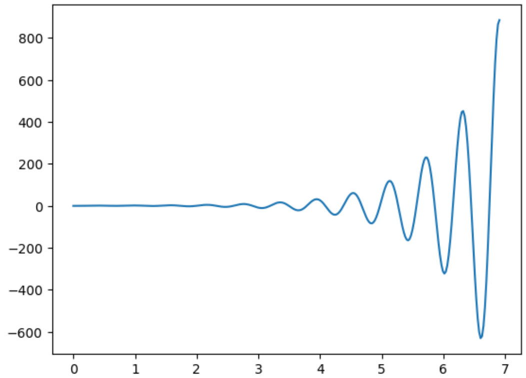
```
import math
import control as ct
import matplotlib.pyplot as plt

omega_n = 10
zeta = 0.1

s = ct.tf('s')
G = 5 / (1 + s) / (1 + 2 * zeta / omega_n * s + s**2 / omega_n**2)

# G(s) Bode plot
plt.figure()
ct.bode_plot(G, margins=True)
plt.show()

# G(s) step response
plt.figure()
t, y = ct.step_response(G / (1 + G))
```

```python
plt.plot(t, y)
plt.show()

mu = 4
C_1 = mu

pole_1 = 10**0
pole_2 = 10**2

L_star = 5 / ((1 + (s / pole_1)) * (1 + (s / pole_2))**2)
C_2 = L_star * (1 / G)
C = C_1 * C_2

# C(s) Bode plot
plt.figure()
ct.bode_plot(C * G, margins=True)
plt.show()

# C(s) step response
plt.figure()
t, y = ct.step_response((C * G) / (1 + (C * G)))
plt.plot(t, y)
plt.show()

# Steady State Output at end of step response
print(f"Steady State Output: {y[-1]}")
# Prints out
# Steady State Output: 0.9523809526136096
```

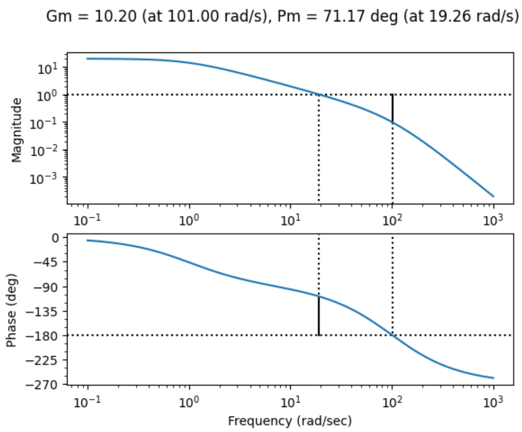
$G(s)$ Bode Plot



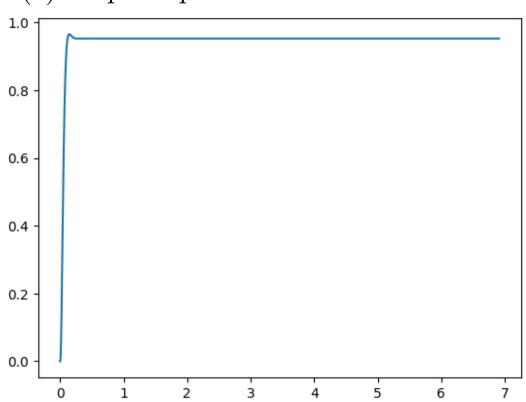
$G(s)$ Step Response



$C(s)$ Bode Plot



$C(s)$ Step Response



As we can see, the steady state error is less than 0.05 (the output is > 0.95 at the end of the step response once the oscillations have settled), the gain margin is greater than 10 dB, and the phase margin is greater than $60°$.

2 2

For the same feedback control system as part 1, design a controller $C(s)$ such athat the closed-loop system has the following specifications: No overshoot, the settling time at 1% is $T_s^{1\%} \leq 0.5\%$, and the steady-state error in response to a unit step function is $|e_\infty| = 0$.

To do this we will adapt the controller from part 1. The controller from part one consists of $C_1(s)$ which was used to control the steady-state error, and $C_2(s)$ which was used to make the system stable and satisfy the gain and phase margin requirements. We will keep $C_2(s)$ the same since we still want the controller to stabilize the system (satisfying the gain and phase margin requirements is an extra bonus to keep the system in a "safe" region of stability) and replace $C_1(s)$ with a controller that will make the system have no overshoot and a settling time of 0.5%. To do this we want to use a proportional-integral controller as it increases the steady state gain to zero error and only marginally reduces the phase margin.

$$mu = 2 \cdot \alpha$$
$$c = \frac{\mu}{\alpha}$$
$$T = \frac{10}{\omega_c}$$
$$C_3(s) = c + \frac{c}{T}\frac{1}{s}$$

We will keep $\alpha = 10$ and $T = 10/\omega_c$ using our crossover ω_c value from part 1 to keep the phase margin decline limited to $\approx 6\%$. We decide to set our increase of the steady state gain to $\mu = 2 \cdot \alpha$ instead of α since the value reached the settling time requirement too slowly.

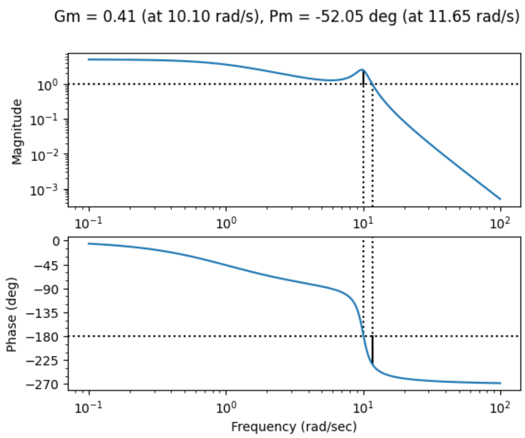
We can verify that the controller satisfies the requirements in code:

```
import math
import numpy as np
import control as ct
import matplotlib.pyplot as plt

omega_n = 10
zeta = 0.1

s = ct.tf('s')
G = 5 / (1 + s) / (1 + 2 * zeta / omega_n * s + s**2 / omega_n**2)

# G(s) Bode plot
plt.figure()
ct.bode_plot(G, margins=True)
plt.show()

# G(s) step response
plt.figure()
t, y = ct.step_response(G / (1 + G))
plt.plot(t, y)
plt.show()

omega_c = 10**1

# We don't rely on C_1 to control the steady-state error
C_1 = 1

pole_1 = 10**0
```

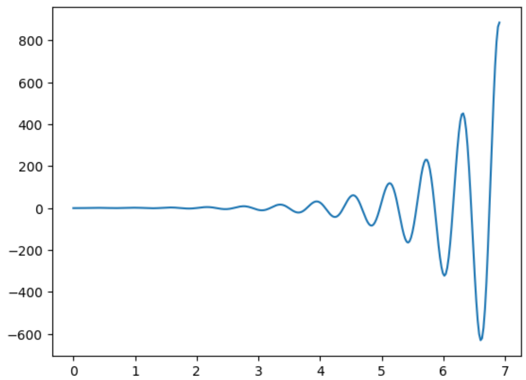
```python
pole_2 = 10**2
L_star = 5 / ((1 + (s / pole_1)) * (1 + (s / pole_2))**2)
C_2 = L_star * (1 / G)

alpha = 10
mu = 2 * alpha  # the two here is a tunable parameter to control how much to increase steady-s
T = 10 / omega_c

c = mu / alpha
C_3 = c + (c / T) * (1 / s)

C = C_1 * C_2 * C_3

# C(s) Bode plot
plt.figure()
ct.bode_plot(C * G, margins=True)
plt.show()

# C(s) step response
plt.figure()
t, y = ct.step_response((C * G) / (1 + (C * G)))
plt.plot(t, y)
plt.show()

# time idx at t = 0.5
idx = np.argmin(np.abs(t - 0.5))

print(f'Overshoot: {1 - max(y)}')
print(f"Max error after settling point (t = 0.5): {max(np.abs(y[idx:] - 1))}")
print(f"Steady-state error: {np.abs(y[-1] - 1)}")

# Prints out:
# Overshoot: -2.8851916411554157e-08
# Max error after settling point (t = 0.5): 0.0019246920997199046
# Steady-state error: 2.1373960601422937e-08
```

As we can see the overshoot is zero (modulo floating point rounding issues at $\approx 1e-8$), the error after $t = 0.5$ stays at less that 0.1% satisfying the settling time requirement, and the steady state error is zero (neglibly small at $\approx 1e-8$).

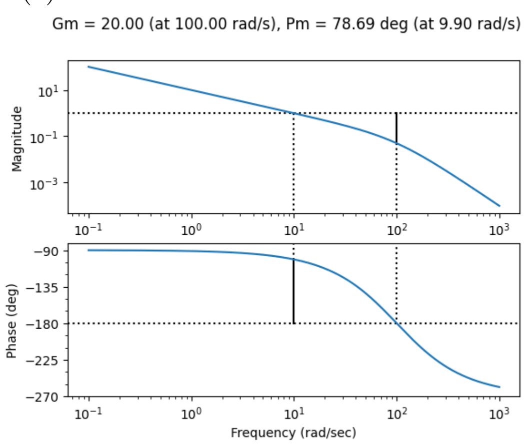
$G(s)$ Bode Plot



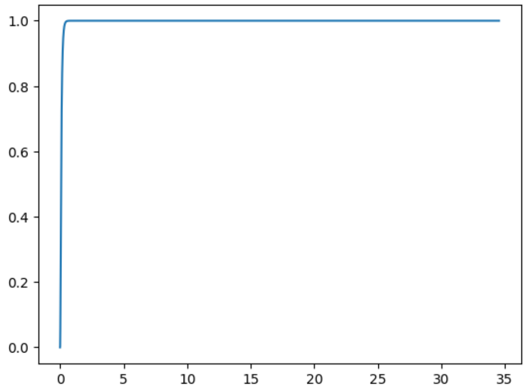
$G(s)$ Step Response



$C(s)$ Bode Plot



$C(s)$ Step Response


1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35