Quinto Relatório PCO119

João Vitor Yukio Bordin Yamashita

September 30, 2022

1 Questão 1

Todos os códigos usados nesse relatório podem ser encontrados no github da matéria[1].

1. Fazendo:

```python
import control
import numpy as np
import matplotlib.pyplot as plt

num = np.array([8])
den = np.array([1, 1.6, 4])

Gs = control.tf(num, den)

gm, pm, wcg, wcp = control.margin(Gs)
print('Margem de ganho: ', gm)
print('Margem de fase: ', str(pm) + ' graus')
print('Frequencia associada com a margem de ganho: ', str(wcg) + ' Rad/seg')
print('Frequencia associada com a margem de fase: ', str(wcp) + ' Rad/seg')

# Bode diagram
control.bode(Gs, dB=True, Hz=False, omega_limits=(0.1, 10), plot=True);

Output:
    Margem de ganho:  inf
    Margem de fase:  39.61231058053758 graus
    Frequencia associada com a margem de ganho:  nan Rad/seg
    Frequencia associada com a margem de fase:  3.1879476381760603 Rad/seg

```

Listing 1: Margens do sistema

Temos o seguinte Diagrama de Bode:

[1] https://github.com/JoaoYukio/PCO119/tree/main/Atividade%205

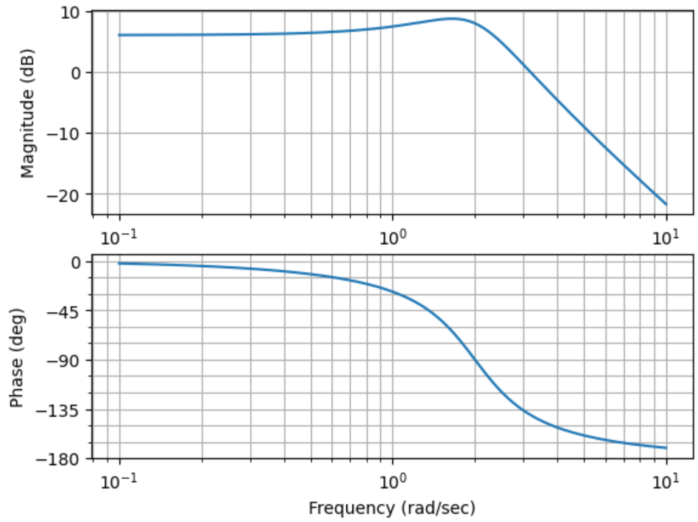

Figure 1: Diagrama de Bode do sistema

Podemos perceber do código acima que a margem de ganho é infinita, já que a fase nunca é menor do que -180, a margem de fase é de 39.61 graus a frequência de cruzamento é de 3.188 Rad/seg.

2. Não, como não temos um integrador no sistema e nem no compensador PD o sistema não conseguira um erro nulo. Podemos ver isso usando o teorema do valor final, considerando uma entrada do tipo degrau:

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s) = \lim_{s\to 0} s((K_p + K_d s)E(s)(\frac{8}{s + 1.6s + 4}))$$

Como E(s) = 1/s:

$$\lim_{t\to\infty} = \lim_{s\to 0} s((K_p + K_d s)\frac{1}{s}(\frac{8}{s + 1.6s + 4})) = \lim_{s\to 0}((K_p + K_d s)(\frac{8}{s + 1.6s + 4})) = K_p(\frac{8}{4}) = 2K_p$$

Portanto o erro para uma entrada do tipo degrau não será nulo, e para entradas de ordens superiores (rampa e parábola) teremos que ter mais integradores para garantir erro nulo. Esses integradores não estão presentes em controladores do tipo PD.

3. Temos a seguinte forma para um controlador avanço/atraso de fase:

$$C(s) = \frac{a_1 s + a_0}{b_1 s + 1}$$

Para o cálculo dos coeficientes vamos usar:

$$\varphi \triangleq \angle K(j\omega_u) = -180^\circ + \phi_m - \angle G(j\omega_u)$$

$$a_1 = \frac{1 - a_0 \left|G(j\omega_u)\right| \cos\varphi}{\omega_u \left|G(j\omega_u)\right| \sin\varphi} \qquad b_1 = \frac{\cos\varphi - a_0 \left|G(j\omega_u)\right|}{\omega_u \sin\varphi}$$

Figure 2: Coeficientes do compensador

Fazemos:

```
from math import pi
a0 = 0.8
```

```python
magJW, phase, omega = control.bode(Gs, dB =True,omega = 2) #Omega = novaFreqCorte
#Temos
phi = -180 + 60 - (phase[0] * (180 / math.pi))
SatJW = 2j
absGsJW = abs(8 / ((SatJW)**2 + 1.6*SatJW +4))
# Calculo do a1
from math import cos, pi, sin

a_1 = (1 - a0*absGsJW*cos(phi*(pi/180)))/((2)*absGsJW*sin(phi*(pi/180)))

#Calculando b1
b_1 = (cos(phi*(pi/180)) - a0*abs(magJW))/((2)*sin(phi*(pi/180)))
b_1 = b_1[0]

#Compensador
Cav = control.tf([a_1, a0], [b_1, 1])
#Sistema controlado
SisCont = control.series(Gs, Cav)

# Bode diagram
control.bode(SisCont, dB=True, Hz=False, omega_limits=(0.1, 10), plot=True);

gm1, pm1, wcg1, wcp1 = control.margin(SisCont)
print('Margem de ganho: ', gm1)
print('Margem de fase: ', str(pm1) + ' graus')
print('Frequencia associada com a margem de ganho: ', str(wcg1) + ' Rad/seg')
print('Frequencia associada com a margem de fase: ', str(wcp1) + ' Rad/seg')

Output:
    Margem de ganho:  19.15777306599095
    Margem de fase:  60.000000102839806 graus
    Frequencia associada com a margem de ganho:  6.707239221934248 Rad/seg
    Frequencia associada com a margem de fase:  2.000000000000002 Rad/seg

```

Listing 2: Coeficientes do compensador

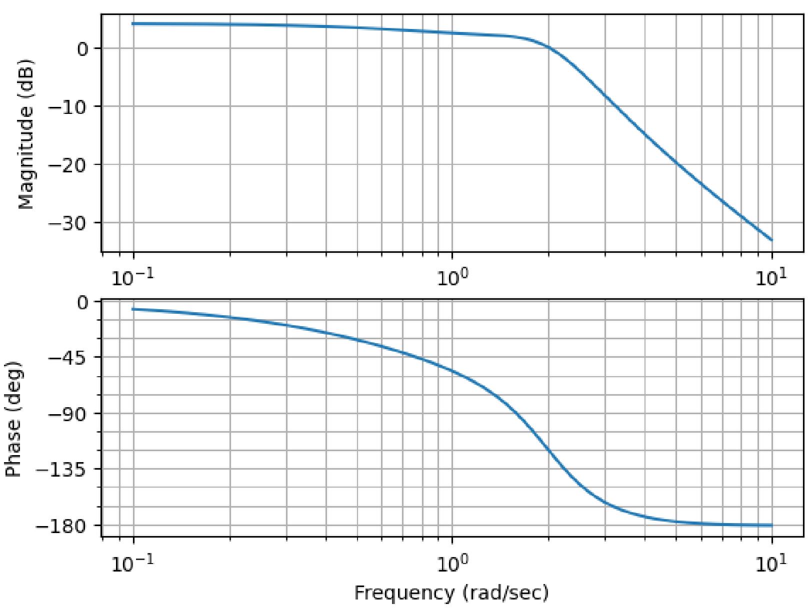

Figure 3: Diagrama de Bode do sistema controlado

4. Podemos ver que a margem de fase e frequência de corte foram atingidas. O compensador teve

a seguinte função de transferência:

$$C(s) = \frac{1.159s + 1.8}{3.634s + 1}$$

5.

```
T, yout = control.step_response(control.feedback(SisCont))
T2, yout2 = control.step_response(control.feedback(Gs), T = T)
plt.plot(T, yout, label = 'Sinal com compensador')
plt.plot(T2, yout2, color = 'orange', label =  'Sinal sem compensador')
plt.legend()
plt.grid()

```

Listing 3: Resposta ao impulso

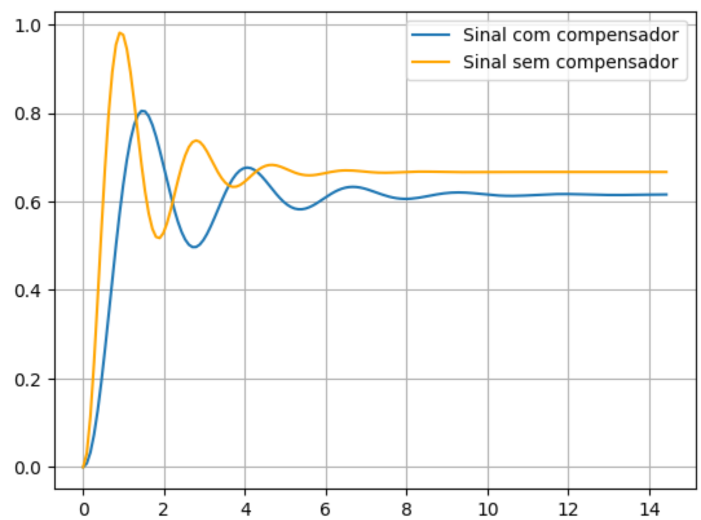

Figure 4: Resposta ao degrau do dois sistemas

Para verificar os valores de *overshoot* e tempo de acomodação fazemos:

```
control.step_info(control.feedback(SisCont))

Output:
    {'RiseTime': 0.6311559792196466,
     'SettlingTime': 7.032880911304633,
     'SettlingMin': 0.4969946997279903,
     'SettlingMax': 0.8046060576621104,
     'Overshoot': 30.74848437009292,
     'Undershoot': 0,
     'Peak': 0.8046060576621104,
     'PeakTime': 1.442642238216335,
     'SteadyStateValue': 0.6153846153846154}

```

Listing 4: Resposta temporal

Podemos ver que o valor do *overshoot* não foi o esperado (aprox. 10%), fazendo um ajuste no valor de a0 podemos aproximar para o valor esperado:

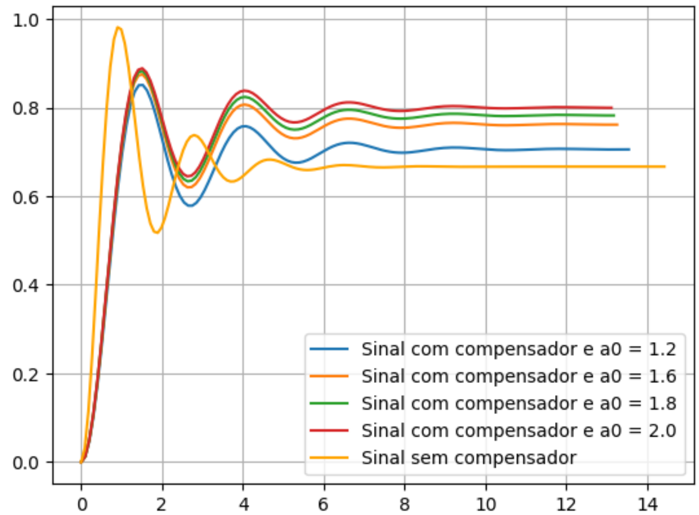

Figure 5: Resposta ao degrau em função da variação de a0

O valor do *overshoot* em função de a0 pode ser visto na tabela abaixo:

a_0	**OV**
0.8	30%
1.2	20%
1.6	14%
1.8	12%
2	11%

Table 1: Influência de a0 no *overshoot* (valores obtidos por simulação)

Podemos ver que o valor esperado para o *overshoot* converge para o esperado a medida que aumentamos o valor de a0. Além disso o erro é reduzido, já que com o aumento de a0 estamos aproximando um polo para origem, para um valor grande (a0 = 50) podemos ver que o erro do sistema tende a 0.

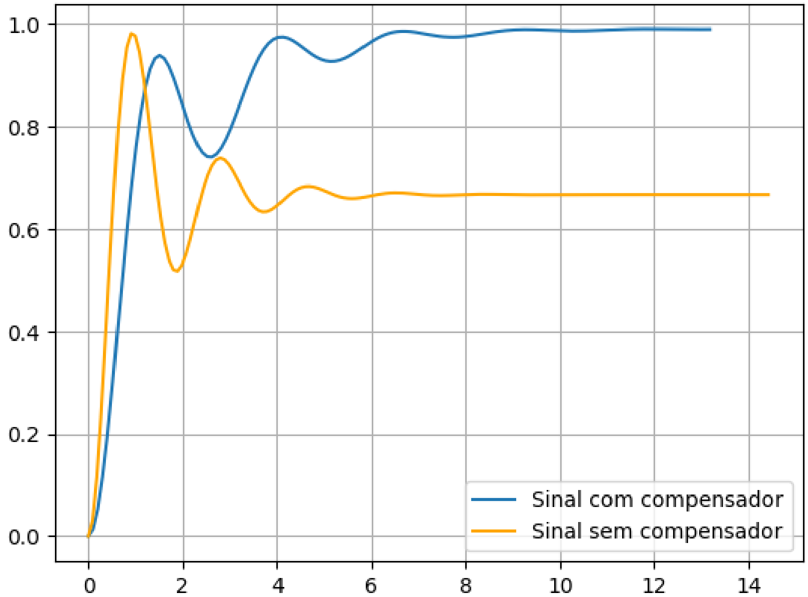

Figure 6: Redução do erro para um valor grande de a0

Isso pode ser útil caso a resposta para um a0 grande não viole nenhuma restrição do sistema e a margem de ganho suporte o valor de a0, caso contrário o sistema se torna instável. Podemos ver a variação do lugar das raízes na imagem abaixo, na esquerda temos a0 = 0.8 e na direita temos a0 = 50:

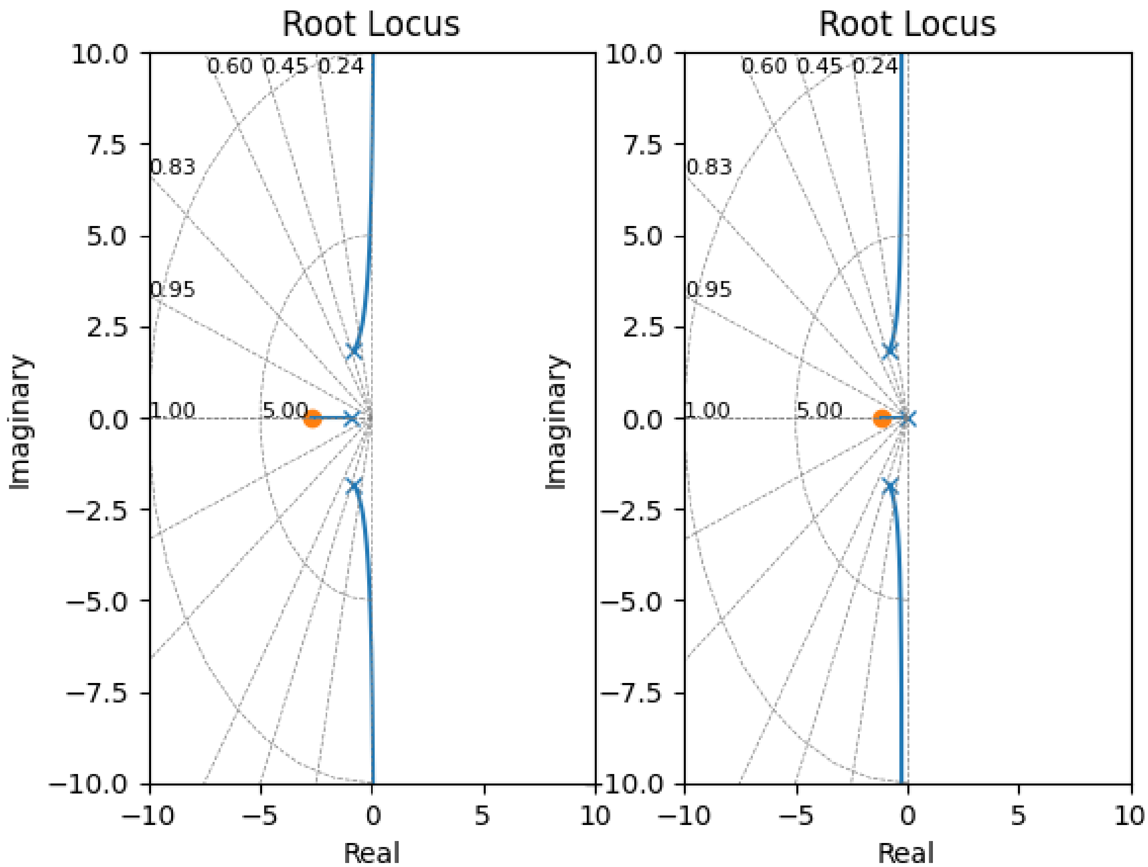

Figure 7: Na esquerda temos a0 = 0.8 e na direita temos a0 = 50

Todos os códigos usados nesse relatório podem ser encontrados no github da matéria[2].

[2] https://github.com/JoaoYukio/PCO119/tree/main/Atividade%205