Segundo Relatorio PCO119

João Vitor Yukio Bordin Yamashita

September 9, 2022

1 Exercício 1

1. É um sistema de segunda ordem.

2. Temos:

$$Y(z) = \frac{0.01781z + 0.01585}{z^2 - 1.679z + 0.7047}$$

Como queremos a resposta ao degrau, temos:

$$Ydeg(z) = \frac{z}{z-1}\frac{0.01781z + 0.01585}{z^2 - 1.679z + 0.7047}$$

$$\frac{Ydeg(z)}{z} = \frac{1}{z-1}\frac{0.01781z + 0.01585}{z^2 - 1.679z + 0.7047}$$

$$\frac{Ydeg(z)}{z} = \frac{0.01781z + 0.01585}{(z-1)(z - 0.83173791)(z - 0.84726209)}$$

Usando a função *residue* temos:

```
import control
import numpy as np

num2 = np.array([0.01781, 0.01585])
den2 = np.array([1, -2.679, 2.3837, -0.7047])
dt = 0.1

gz2= control.TransferFunction(num2, den2, dt)

import scipy
from scipy import signal

res = signal.residue(num2, den2)
res

Output:
(array([ 11.73879239, -13.04852002,   1.30972763]),
 array([0.83173791, 0.84726209, 1.        ]),
 array([], dtype=float64))

```

Listing 1: Cálculo dos resíduos

Portanto temos:

$$\frac{Ydeg(z)}{z} = \frac{11.73879239}{z - 0.83173791} - \frac{13.04852002}{z - 0.84726209} + \frac{1.30972763}{z-1} \tag{1}$$

De 1 temos:

$$Ydeg(z) = 11.73879239\frac{z}{z - 0.83173791} - 13.04852002\frac{z}{z - 0.84726209} + 1.30972763\frac{z}{z-1} \tag{2}$$

Usando a tabela de transformações em 2 obtemos:

$$Ydeg(z) = 11.73879239e^{0.83173791k} - 13.04852002e^{0.84726209k} + 1.30972763 \tag{3}$$

3. Simulando o resultado, obtemos:

```
    T2 = np.arange(0, 100)
    resultados = []
    for k in T2:
        yk = res[0][1]*(res[1][1])**k + res[0][2] + res[0][0]*(res[1][0])**k
        resultados.append(yk)
    plt.title("Resposta ao degrau do sistema com Ts = "+ str(dt) )
    plt.grid()
    plt.plot(T2*0.1, resultados)

```

Listing 2: Simulação do resultado

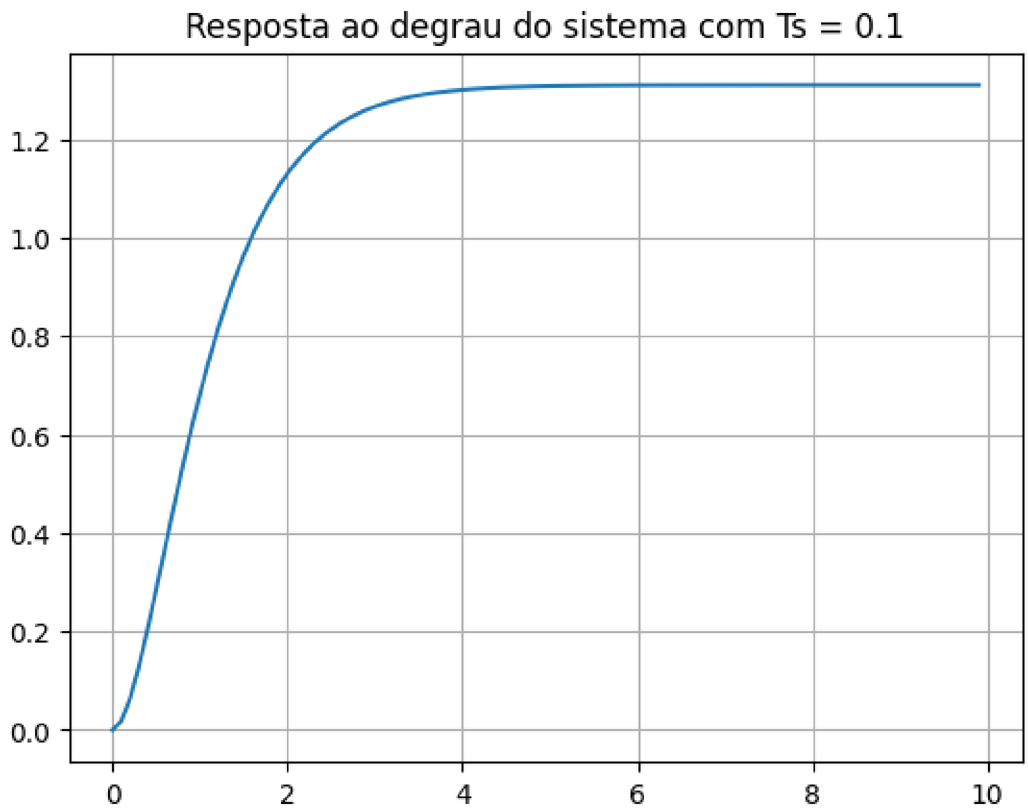

Figure 1: Simulação a partir dos calculos

Podemos comparar o resultado calculado e o simulado para confirmar os resultados:

```
    num = np.array([0.01781, 0.01585], dtype= float)
    den = np.array([1, -1.679, 0.7047])
    dt = 0.1

    gz= control.TransferFunction(num, den, dt)

    yout, T = control.step_response(gz)

    plt.title("Comparacao entre a simulacao e o resultado calculado")
    plt.grid()
    plt.plot(T2*0.1, resultados, '-', label = "Calculado")
    plt.plot(yout, T,':', label = "Simulado")
    plt.legend()

```

Listing 3: Comparacao dos resultados

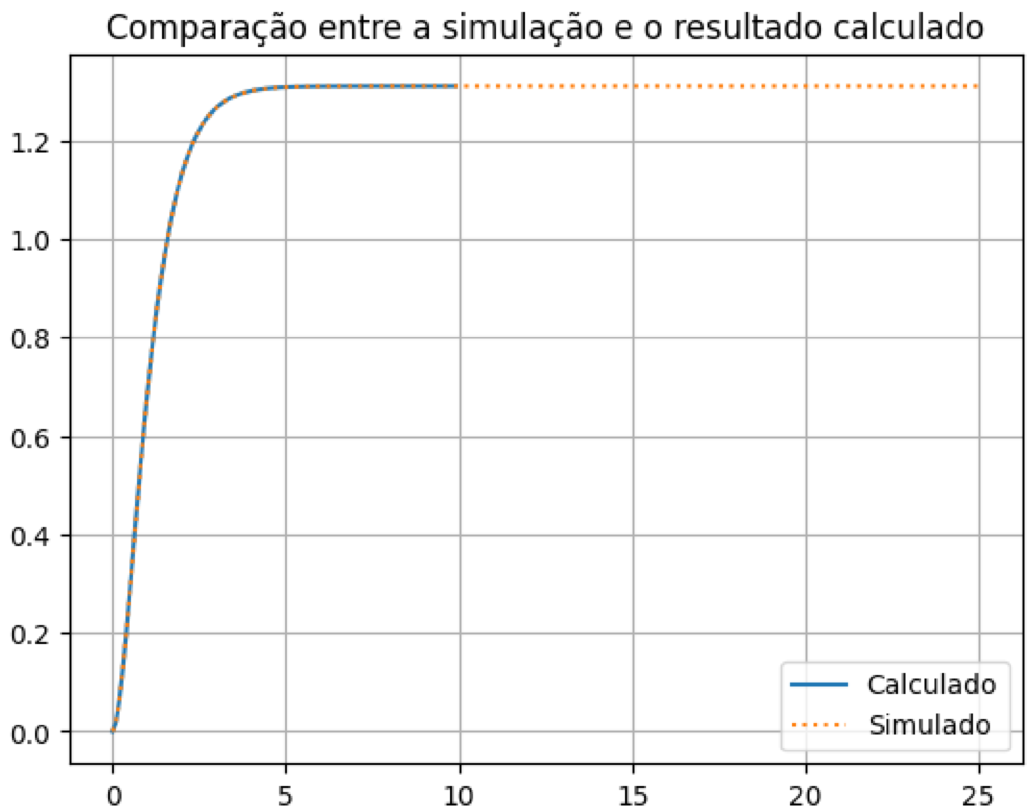

Figure 2: Comparação dos resultados

2 Exercício 2

Temos:

$$Y(z) = \frac{0.004865z + 0.004737}{z^2 - 1.914z + 0.9231} = \frac{0.004865z^{-1} + 0.004737z^{-2}}{1 - 1.914z^{-1} + 0.9231z^{-2}}$$

Com isso, podemos fazer:

$$Y(z) - 1.914z^{-1}Y(z) + 0.9231z^{-2}Y(z) = 0.004865z^{-1}E(z) + 0.004737z^{-2}E(z)$$

Aplicando a propriedade da translação:

$$y(k) - 1.914y(k-1) + 0.9231y(k-2) = 0.004865e(k-1) + 0.004737e(k-2)$$

$$y(k) = 1.914y(k-1) - 0.9231y(k-2) + 0.004865e(k-1) + 0.004737e(k-2)$$

Para uma entrada do tipo degrau:

$$e(k) = 1 \forall k \geq 0$$

Por isso devemos calcular y(0) e y(1) manualmente:

$$y(0) = 0 - 0 + 0 + 0 = 0$$

$$y(1) = 0 - 0 + 0.004865 + 0 = 0.004865$$

```
    num = np.array([0.004865, 0.004737])
    den = np.array([1,-1.914,0.9231])
    dt = 0.1
    gz = control.TransferFunction(num, den, dt)

    yout, T = control.step_response(gz)
```

```python

T1 = np.arange(0, 17.5, 0.1)#Usando o mesmo dt!
y = np.zeros_like(T1)
y[0] = 0
y[1] = 0.004865
i = 2
for t in T1:
    if i == T1.size:
        break
    y[i] = 1.914*y[i-1]-0.9231*y[i-2] +0.004865 + 0.004737 # todos valores de 'e' sao iguais a 1
    i+=1

plt.grid()
plt.plot(yout, T,'+', label="Simulacao")
plt.plot(T1, y, color= '#F39C12', label = 'eq. diferencas')
plt.legend()

```

Listing 4: Equação a diferencas

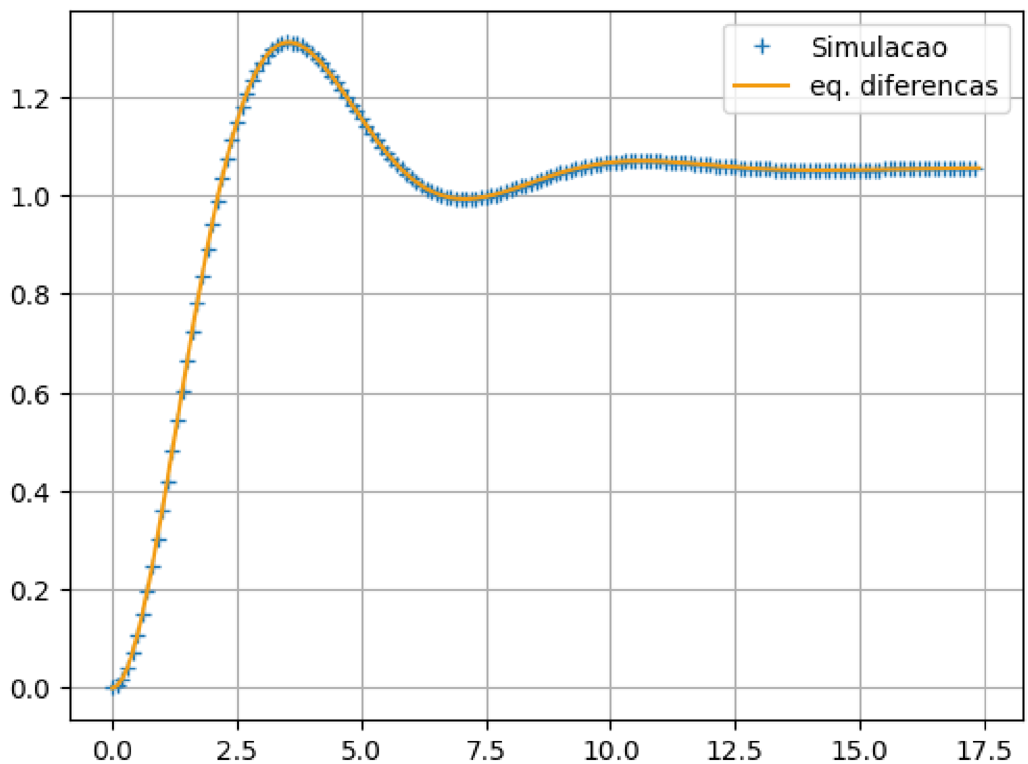

Figure 3: Comparação dos resultados

Podemos observar que ambos resultados são iguais.

3 Exercício 3

1. Usando a função pzmap, temos:

```python
num1 = np.array([0.0001471, 0.0005194, 0.0001146])
den1 = np.array([1,-2.534,2.145,-0.6065])
dt1 = 0.1
gz1 = control.TransferFunction(num1,den1, dt1)

from control.matlab import pzmap
pzmap(gz1)
```

```
output:
(array([0.84843808+0.06824469j, 0.84843808-0.06824469j,
    0.83712385+0.j        ]),
array([-3.29445461+0.j, -0.23647673+0.j]))
```

Listing 5: Polos e zeros do primeiro item

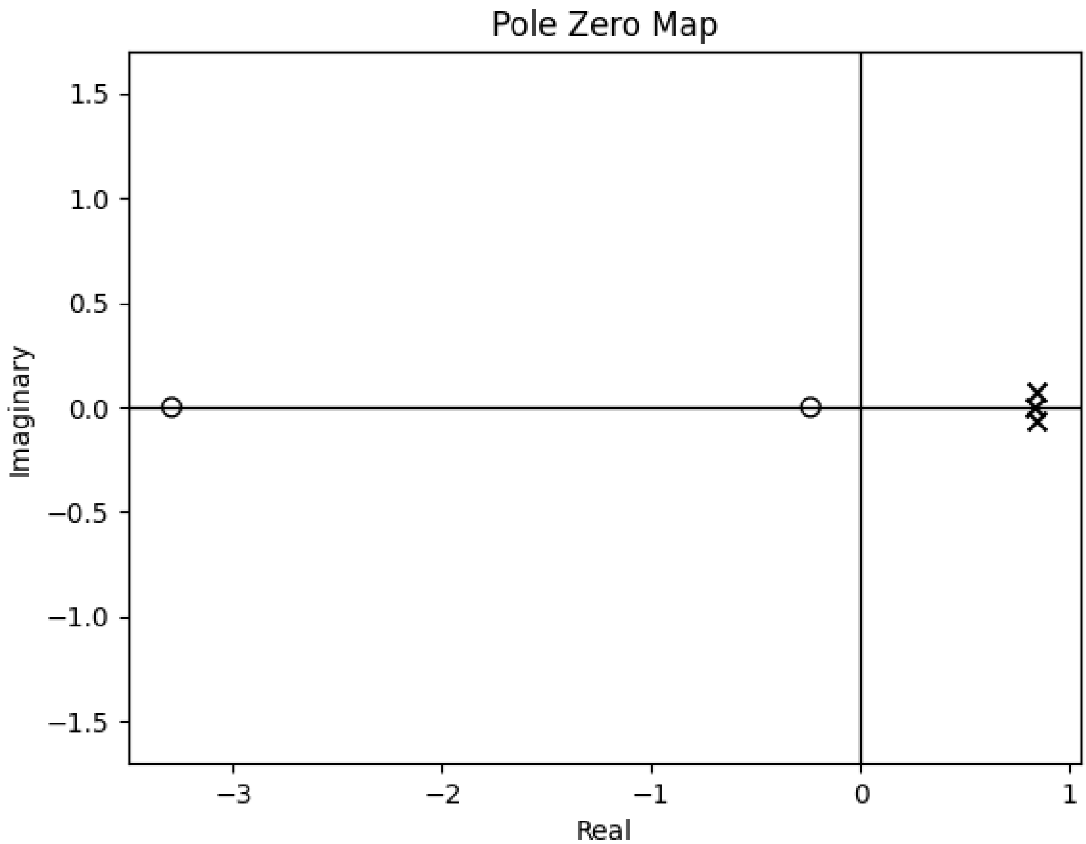

Figure 4: pzmap do primeiro item

```
poles1 = abs(control.poles(gz1))
poles1

Output:
array([0.85117831, 0.85117831, 0.83712385])
```

Listing 6: Polos e zeros do primeiro item

Podemos observar na saída acima e na Figura 4 que todos os polos estão dentro do círculo unitário, logo o sistema é estável. Simulando a resposta para uma entrada degrau podemos ver isso:

```
yout1, T1 = control.step_response(gz1)
plt.grid()
plt.title("Resposta do sistema 1 ao degrau")
plt.plot(yout1, T1)
```

Listing 7: Resposta ao degrau

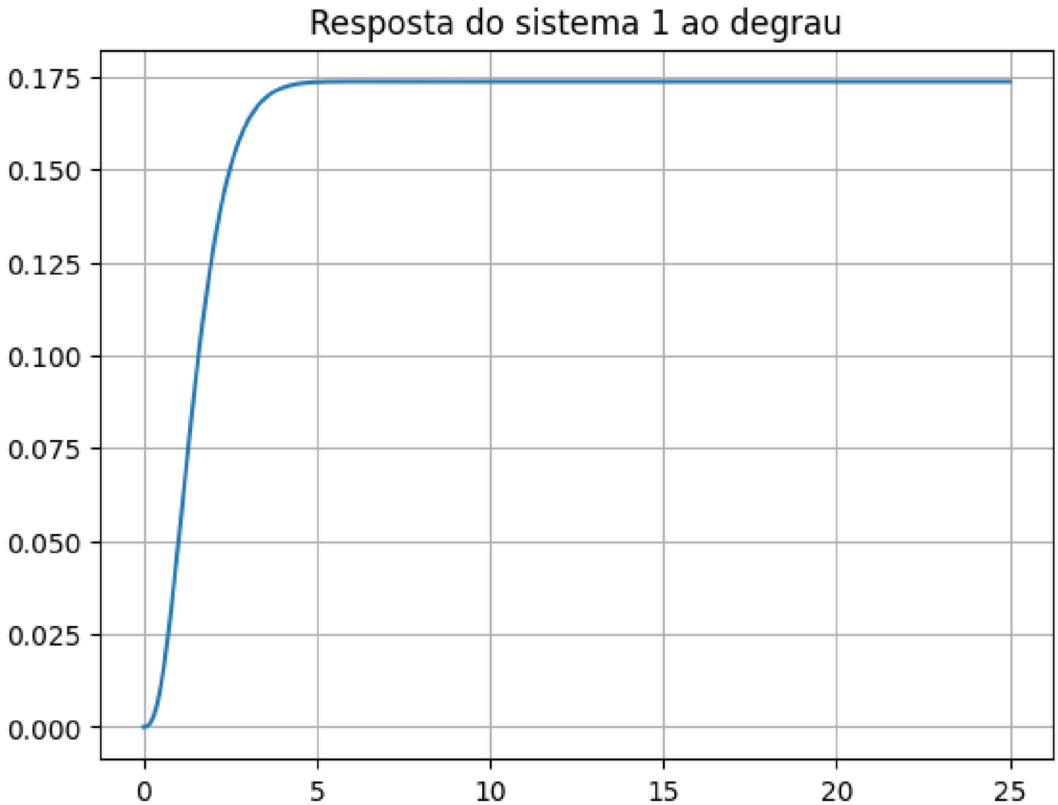

Figure 5: Resposta ao degrau do primeiro item

2. Para o segundo item temos:

```
    num2 = np.array([0.0001548,0.0005752,0.0001332])
    den2 = np.array([1,-2.734,2.471,-0.7408])
    dt2 = 0.1
    gz2 = control.TransferFunction(num2, den2, dt2)
    pzmap(gz2)

```

Listing 8: Item 2

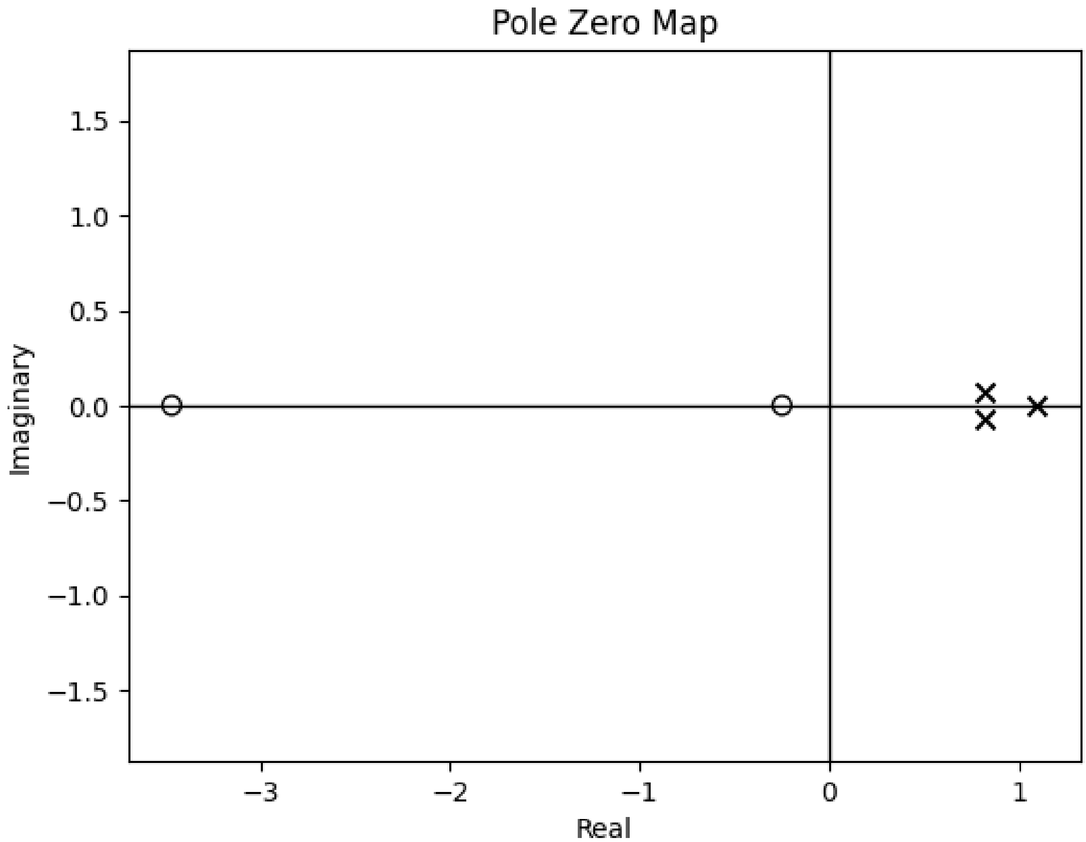

Figure 6: pzmap do segundo item

```
poles2 = abs(control.poles(gz2))
poles2

Output:
    array([1.09812074, 0.82134466, 0.82134466])
```

Listing 9: Polos e zeros

Podemos ver que o segundo sistema tem um polo fora do círculo unitário, logo, é instável. Como podemos ver na simulação da resposta ao degrau:

```
yout2, T2 = control.step_response(gz2) #Sistema instavel, |polo| > 1

plt.grid()
plt.title("Resposta ao degrau do segundo sistema sistema")
plt.plot(yout2, T2, color='orange')
```

Listing 10: Simulacao degrau item 2

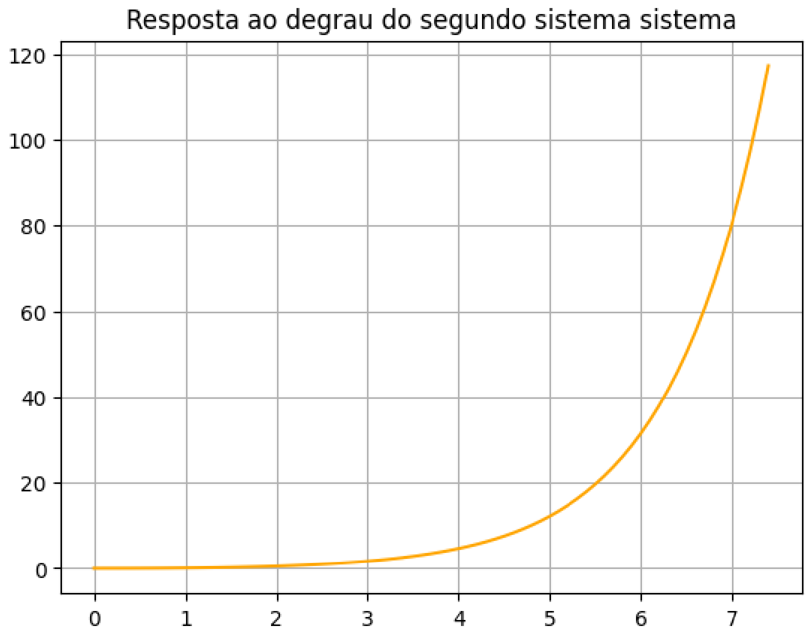

Figure 7: Resposta ao degrau do segundo sistema

4 Exercício 4

1. Usando o diagrama de bode:

```
    num = np.array([1])
    den = np.array([1, 0.8, 1])

    gs = control.TransferFunction(num, den)
    (mag, phase, w) = control.bode_plot(gs)

```

Listing 11: Bode

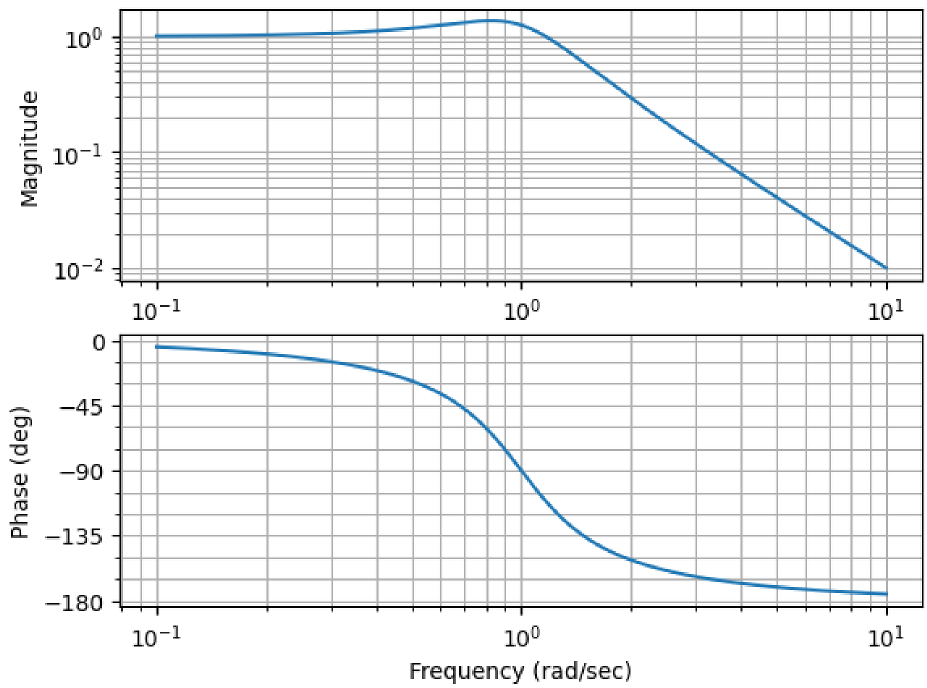

Figure 8: Diagrama de Bode

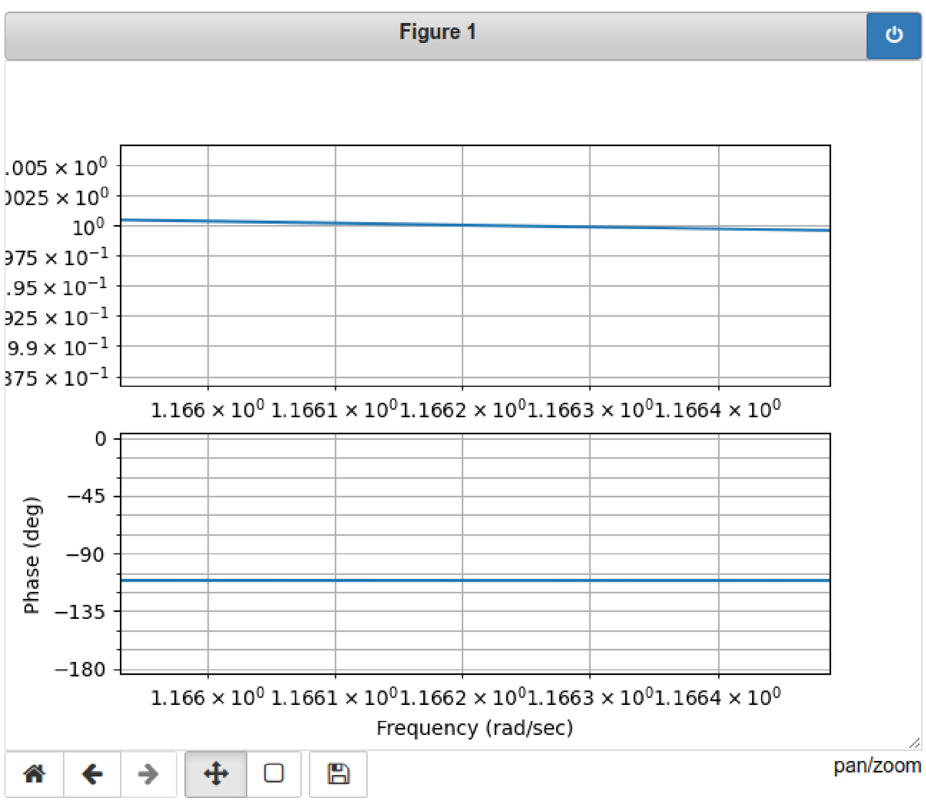

Figure 9: Diagrama de Bode na frequência com ganho unitário

Podemos ver que a frequência onde temos ganho unitário é aproximadamente 1.662[rad/sec] = 0.185606561 [Hz], que é nossa largura de banda, seguindo o aconselhado nos *slides*, vamos usar

10x a largura de banda como frequência de amostragem.

```
#1 rad/s = 0.159155 Hz

#Freq ganho unitario, 1.1662 -> figura bode_ganUn.png

freqS = 10*(1.1662*0.159155)
freqS
Ts = 1/freqS

```

Listing 12: Freq. amostragem

Também podemos usar o tempo de subida do sistema para escolher o período de amostragem, o valor recomendado nos *slides* é de 0.3 vezes o tempo de subida.

```
control.step_info(gs)

Output:
{'RiseTime': 1.5699443815868492,
 'SettlingTime': 8.54747496641729,
 'SettlingMin': 0.9356617613694419,
 'SettlingMax': 1.2533621089794214,
 'Overshoot': 25.336210897942134,
 'Undershoot': 0,
 'Peak': 1.2533621089794214,
 'PeakTime': 3.488765292415221,
 'SteadyStateValue': 1.0}

```

Listing 13: Tempo de subida

Logo o período de amostragem seria de 1.5699443815868492 * 0.3 = 0.47098331447605474 s

2. Usando a frequência de amostragem calculada no primeiro método, temos:

```
gz = control.sample_system(gs, Ts, method = 'zoh')
gz

```

Listing 14: Metodo ZOH

$$\frac{0.1234z + 0.1068}{z^2 - 1.42z + 0.6498} \quad dt = 0.5387740576692223$$

Figure 10: Transformada usando ZOH

3. Para o método de Tustin:

```
gz2 = control.sample_system(gs, Ts, method = 'tustin')
gz2

```

Listing 15: Metodo Tustin

$$\frac{0.05634z^2 + 0.1127z + 0.05634}{z^2 - 1.44z + 0.6654} \quad dt = 0.5387740576692223$$

Figure 11: Transformada usando Tustin

4. Os modelos são diferentes, isso se deve ao fato do ZOH ser uma transformação literal do sistema de amostragem, já o método de Tustin usa uma aproximação. O método de Tustin é menos preciso mas é mais simples, já o ZOH é preciso mas é mais complexo. Podemos ver na figura abaixo uma comparação da resposta em S e dos dois modelos para uma entrada degrau, podemos ver que o modelo de Tustin é mais impreciso.

```
    yT , T = control.step_response(gs)
    yout1, T1 = control.step_response(gz)
    yout3, T3 = control.step_response(gz2)

    plt.figure(0)
    plt.grid()
    plt.step(yout1, T1, 'r', label = "ZOH")
    plt.step(yout3, T3, color =  'orange', label = "Tustin")
    plt.plot(yT, T, label = "Continuo")
    plt.legend(loc = 'right')
    plt.show()

```

Listing 16: Resposta dos tres sistemas

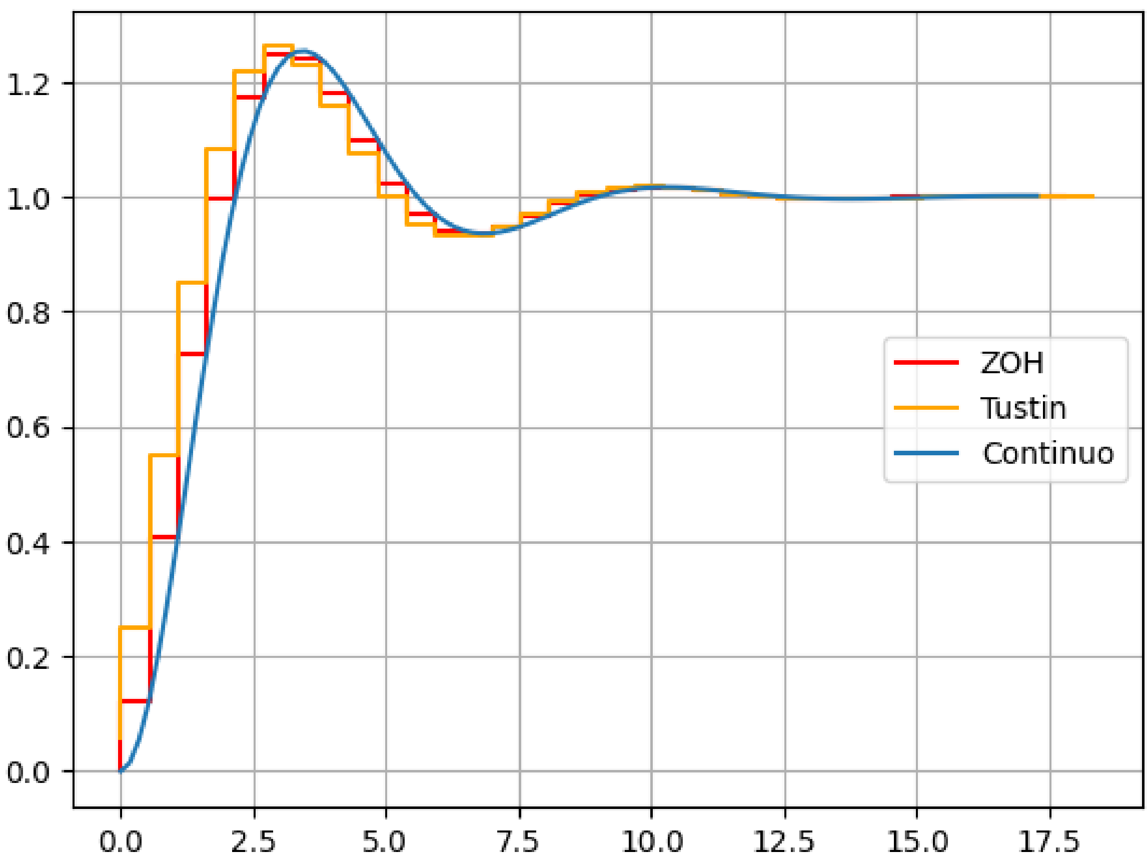

Figure 12: Resposta dos três modelos