Terceiro Relatório PCO119

João Vitor Yukio Bordin Yamashita

September 17, 2022

1 Exercício 1

1. Temos a seguinte resposta em frequência:

```
import control
import numpy as np

num = np.array([200])
den = np.array([1, 10,100])

Gs = control.TransferFunction(num, den)

(mag, phase_rad, w) = control.bode_plot(Gs)

```

Listing 1: Diagrama de Bode

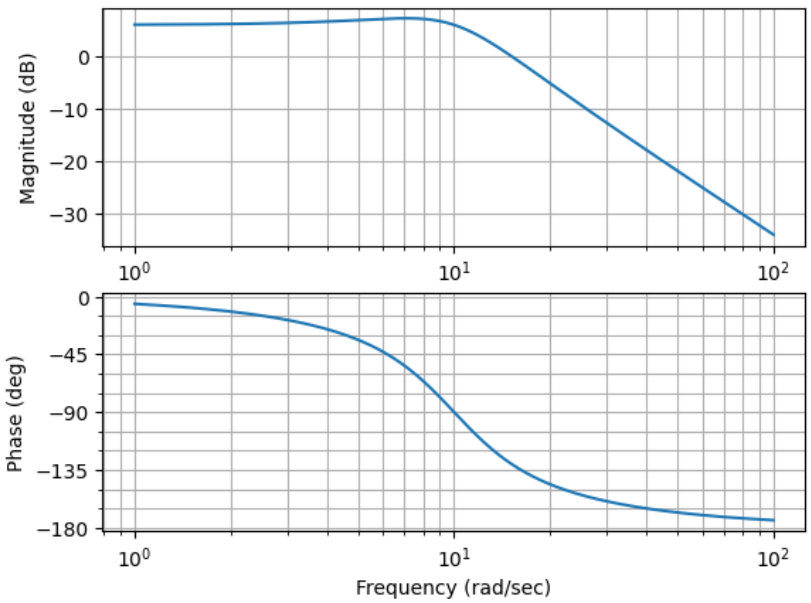

Figure 1: Diagrama de bode

2. A taxa de amostragem foi escolhida com base na largura de banda (10x a largura de banda) obtida pelo diagrama de bode acima:

```
(GM, PM, wg, wp) = control.margin(Gs)
#wp armazena a largura de banda
def rad2hz(rad: float)->float:
    return rad*0.159155
largBand = rad2hz(wp)

freqS = 10*largBand
```

```
    Ts = 1/freqS

    print(str(freqS) + ' Hz')
    print(str(Ts)+ " s")

    Output:
        24.15161071913653 Hz
        0.04140510592147172 s

```

Listing 2: Largura de banda

Para converter o modelo para z fazemos:

```
    Gz = control.sample_system(Gs, Ts, 'zoh')
    Gz

```

Listing 3: Transformada Z

$$\frac{0.148z + 0.1288}{z^2 - 1.523z + 0.661} \quad dt = 0.04140510592147172$$

Figure 2: Modelo em Z

3. Para a resposta em frequência de Z, fazemos:

```
    (mag, phase_rad, w) = control.bode_plot(Gs, label = 'Bode s')
    (mag1, phase_rad1, w1) = control.bode_plot(Gz, color = 'red', label = 'Bode z'
    )
    plt.legend()

```

Listing 4: Bode z

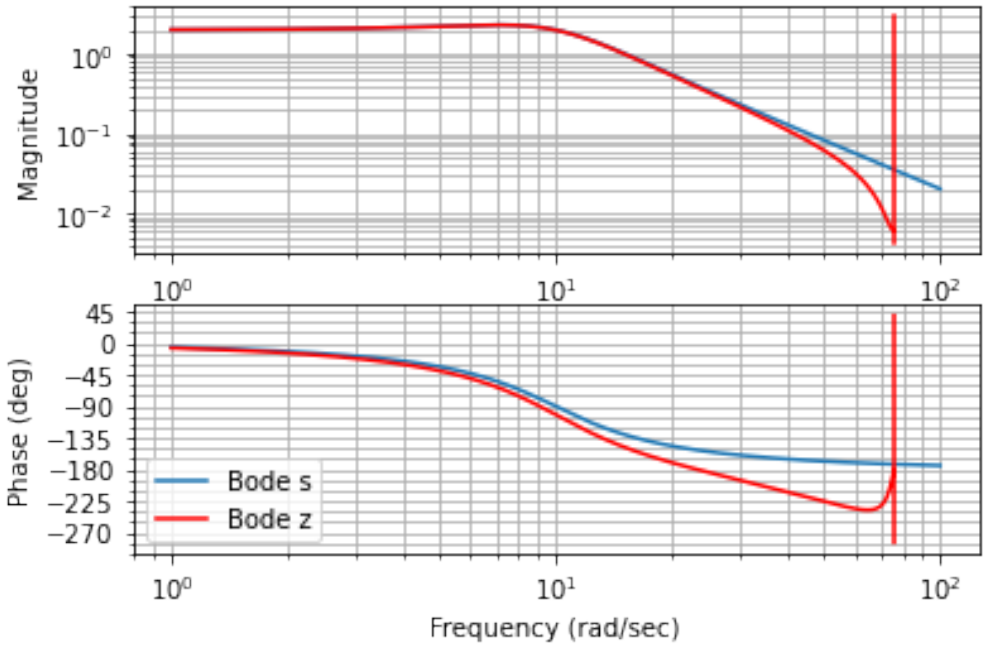

Figure 3: Diagrama de bode do modelo em Z

Podemos ver no diagrama em vermelho (diagrama de bode do sistema em z) que quanto mais a frequência do sinal se aproxima de metade da frequência de amostragem do sistema mais

degradado o sinal se torna (maior a atenuação e o atraso da fase), em casos que a fase pode ser menor do que -180 graus, tornando o sistema instável. Isso se deve ao fato do sinal se aproximar da frequência máxima estabelecida pelo teorema de Nyquist.

4. Para conversão do modelo em Z para W usamos um pacote chamado harold, da seguinte forma:

```
import harold

Gs1 = harold.Transfer(num, den)
Gz1 = harold.discretize(Gs1, Ts, method = 'zoh')

Gw1 = harold.undiscretize(Gz1, method = 'tustin')

Gw = control.TransferFunction(Gw1.num[0], Gw1.den[0])
Gw

```

Listing 5: Modelo W

$$\frac{-0.006017s^2 - 3.909s + 202.8}{s^2 + 10.29s + 101.4}$$

Figure 4: Modelo em W

Usamos a biblioteca para calcular os coeficientes, os resultados obtidos com essa biblioteca foram os mesmos usando o Octave.

5. Plotando os três diagramas conseguimos ver o efeito da discretização na resposta em frequência:

```
control.bode_plot(Gs, label = "S");
control.bode_plot(Gz, color = 'red', label = 'Z');
control.bode_plot(Gw, color = 'purple', label = 'W');
plt.legend()

```

Listing 6: Bode W

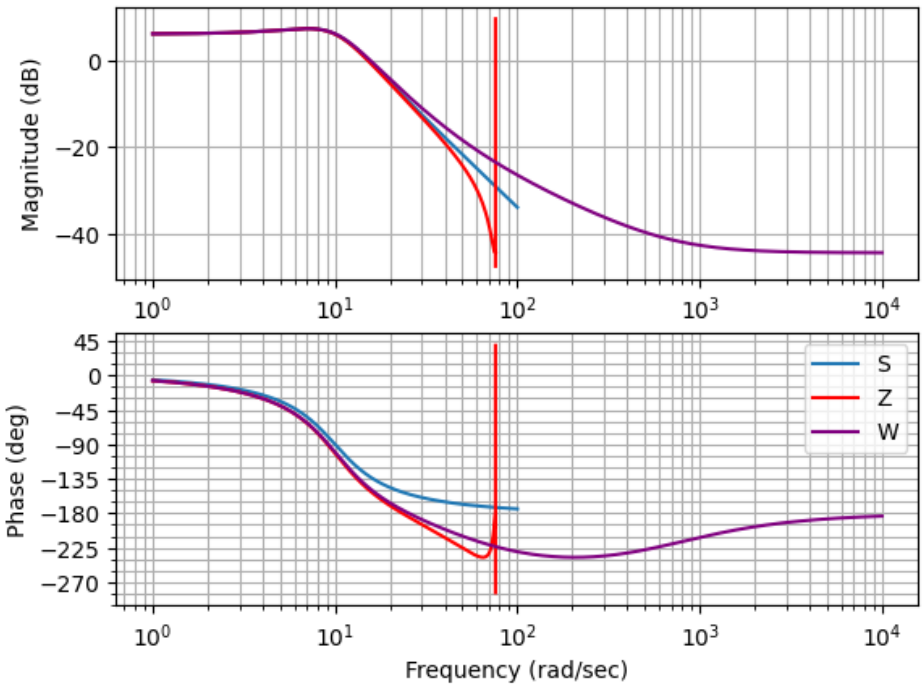

Figure 5: Diagrama de bode do modelo em W

Podemos ver que apesar do modelo em W estar no domínio contínuo ele ainda apresenta as deformações causadas pela discretização para frequências próximas da frequência de Nyquist.

Conseguimos perceber nesses três diagramas que, para baixas frequências, ambos sistemas respondem de formas muito semelhantes, mas quando as frequências se aproximam da frequência de Nyquist eles divergem.

2 Exercício 2

Para simular os sinais foi usado o método lsim do pacote matlab da biblioteca control, foi criada uma função que recebe o sistema, a frequência do sinal e o tempo de simulação:

```
    def plotSysResp(Gs, rad, tsim = 4):
        from control import matlab
        time = np.arange(0, tsim, 0.0001)
        sinal = []
        for t in time:
            sinal.append(np.sin(rad*t))
        plt.plot(time, sinal, label = 'Original')
        yout, T, xout = matlab.lsim(Gs, U = sinal, T = time)
        plt.plot(T, yout, label = 'Saida')
        plt.grid()
        plt.legend()
```

Listing 7: lsim

1. Para uma frequência de 1 rad/s:
   plotSysResp(Gs, 1, 10)

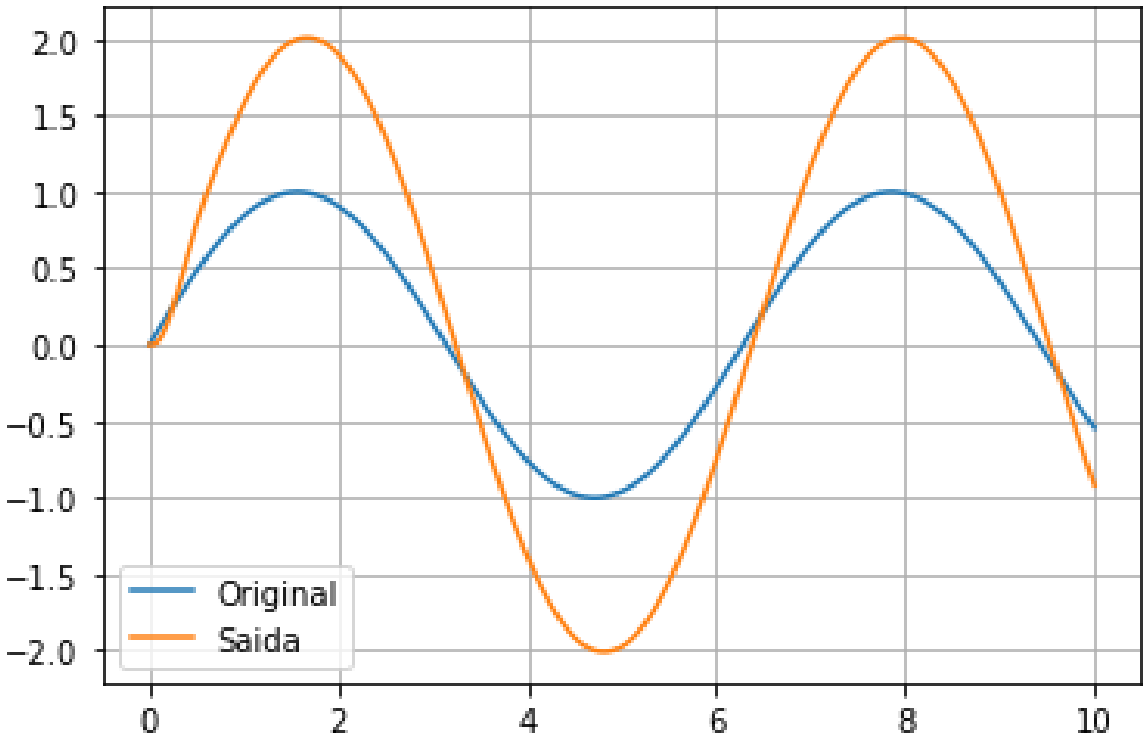

Figure 6: Resposta do sistema para um sinal de 1 rad/s

Podemos perceber na Figura 6 que o sistema não apresenta quase nenhuma defasagem e um

ganho de 2 vezes, conforme o observado no diagrama de Bode na Figura 3, onde o ganho está representado de forma escalar e não em dB.

2. Para uma frequência de 10 rad/s:
plotSysResp(Gs, 10, 3)

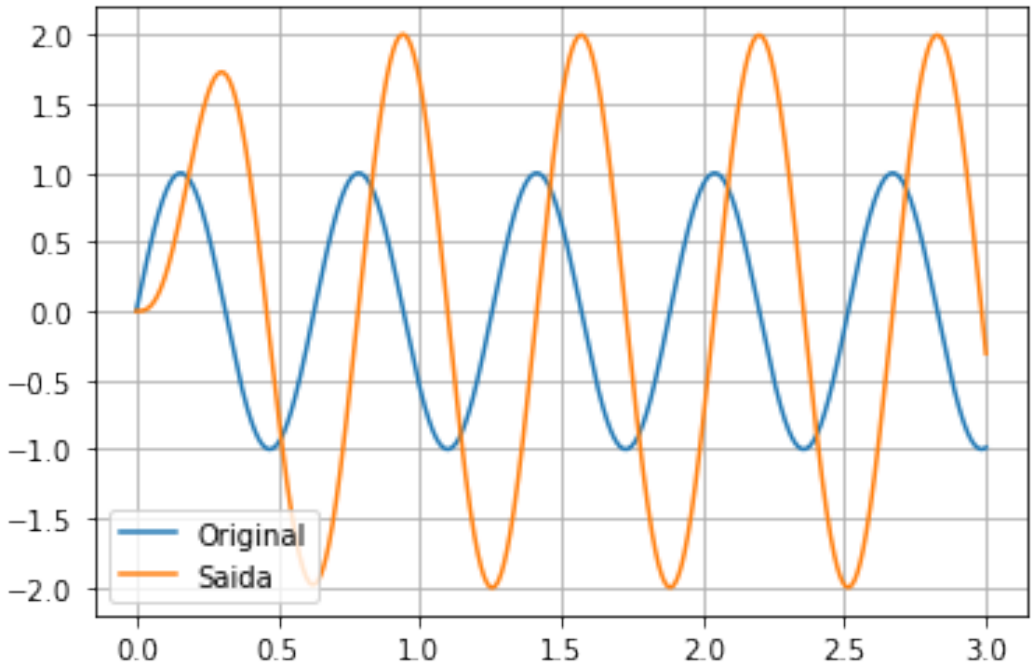

Figure 7: Resposta do sistema para um sinal de 10 rad/s

Na Figura 7 conseguimos perceber um atraso no sinal, mas nenhuma redução no ganho, esse comportamento era esperado, como podemos ver na Figura 3, onde o ganho para uma frequência de 10 rad/s é de 2 e a fase é de aproximadamente -90º.

Podemos calcular usando o atraso entre dois picos:

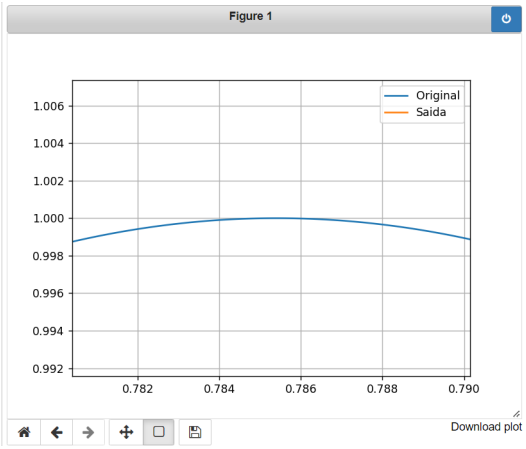

Figure 9: Pico do sinal de entrada

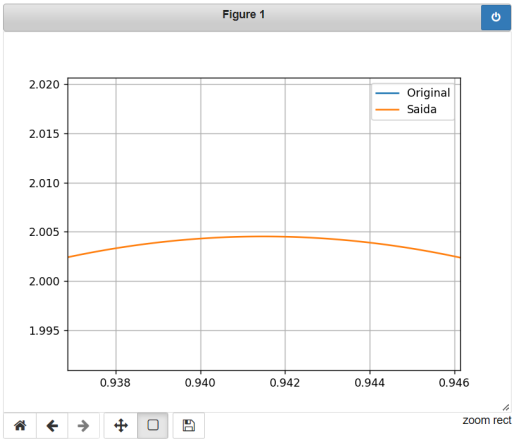

Figure 10: Pico do sinal de saída

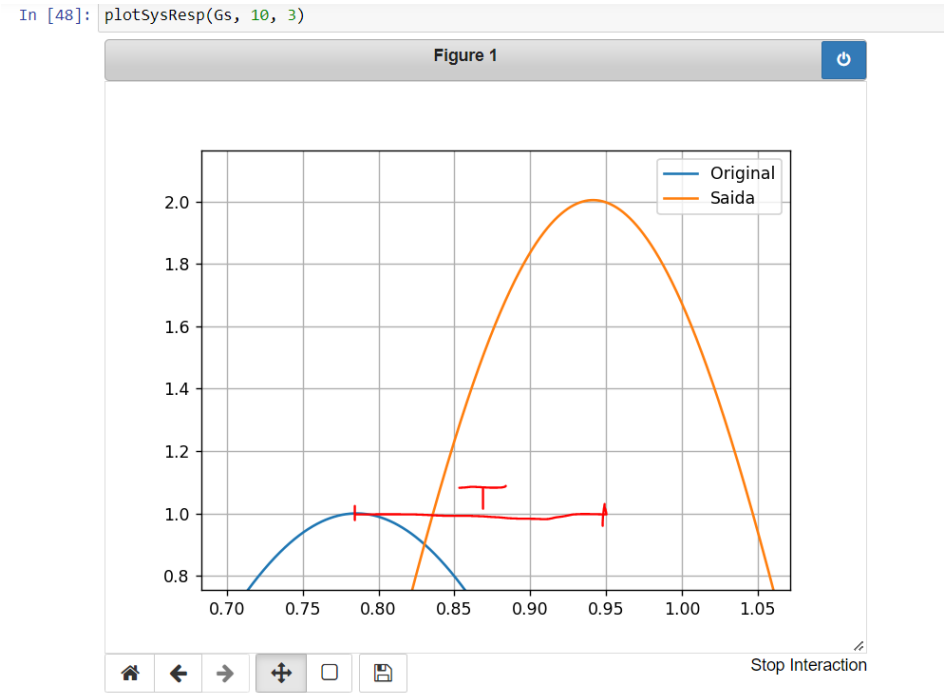

Figure 8: Atraso entre dois picos

Usando o zoom em um notebook jupyter, temos:

Calculando o atraso gerado, temos:

$$T_{atraso} = 0.786 - 0.942 = -0.156[s]$$

Como uma frequência de 10 rad/s equivale a 1.59154[Hz], o período da onda é de $1/1.59154 = 0.6283$[s], desta forma o atraso entre as duas ondas pode ser calculado fazendo:

$$Atraso_{graus} = \frac{-0.156}{0.6283} \times 360 = -89.384[\text{deg}]$$

Que confirma o esperado, um atraso de aproximadamente -90º.

3. Para uma frequência de 100 rad/s:
plotSysResp(Gs, 100, 0.2)

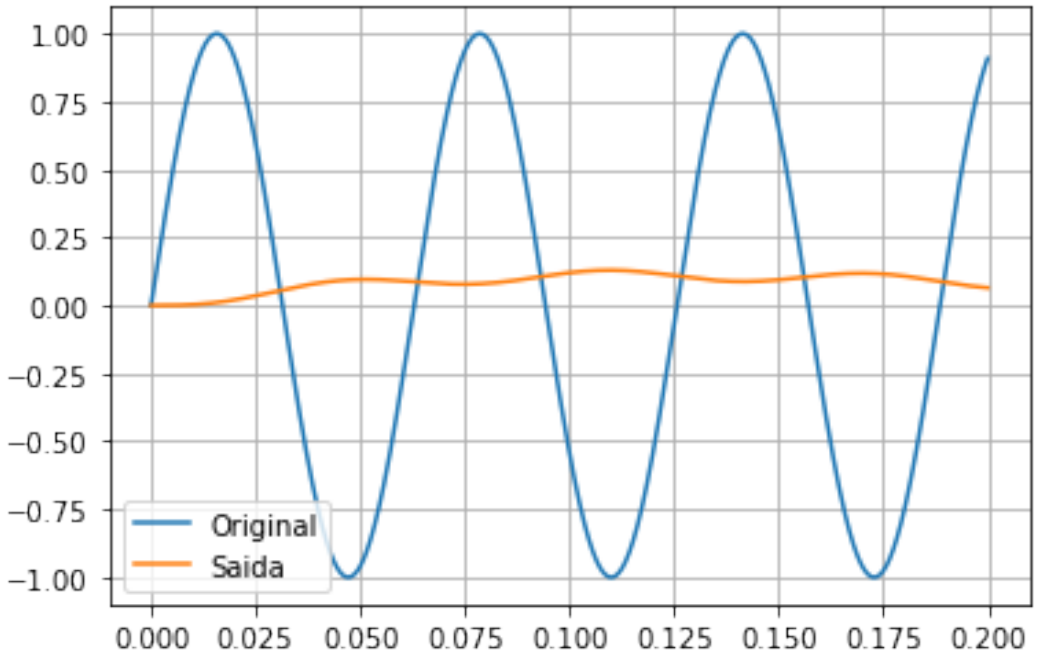

Figure 11: Resposta do sistema para um sinal de 100 rad/s

Podemos perceber que o sinal foi basicamente perdido, novamente, esse era o comportamento esperado, considerando que para essa frequência temos um ganho próximo de 0.01 e um defasamento de -180º.

3 Exercício 3

Temos uma atenuação de:

$$-20\frac{dB}{dec}$$

A partir da frequência do polo do sistema, como podemos observar abaixo:

```
1    #Sistema primeira ordem
2    Gs1 = control.TransferFunction(1, [10, 1])
3    (mag1, phase_rad1, w1) = control.bode_plot(Gs1)
4    #Cai 20 dB por decada a partir de 10^-1
```

Listing 8: Atenuacao 1 ordem

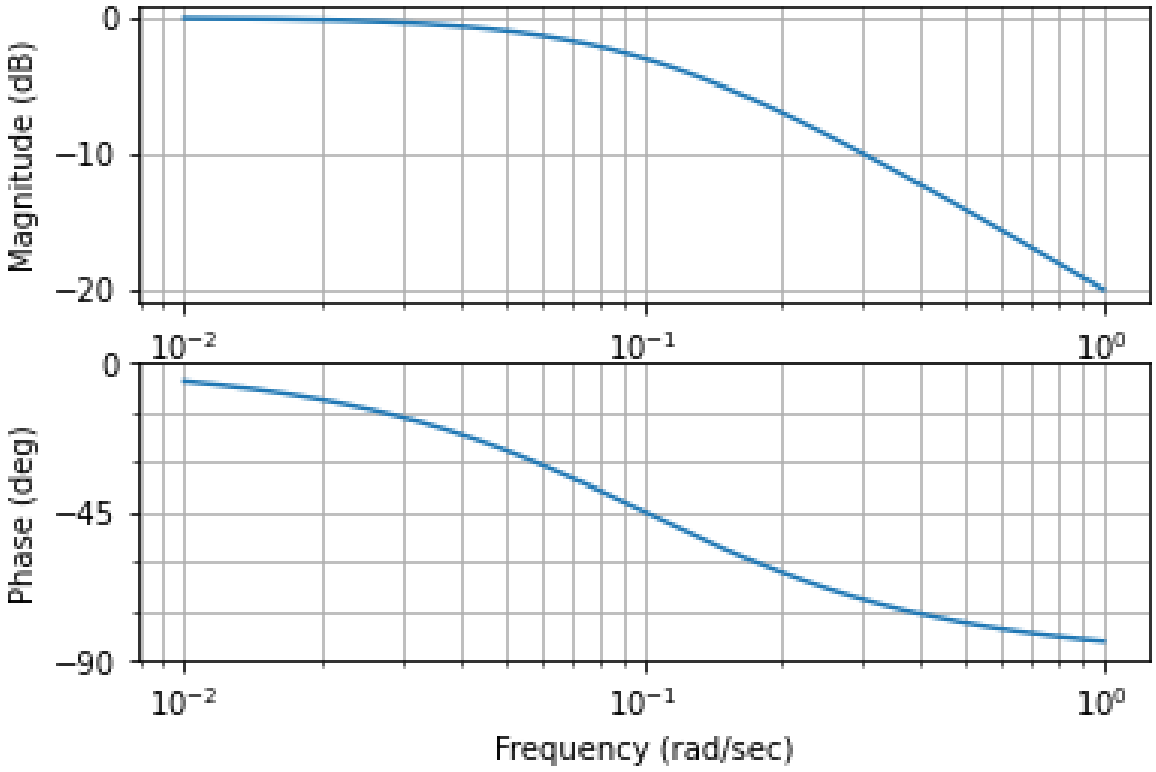

Figure 12: Atenuação para um sistema de primeira ordem

4 Exercício 4

Para um sistema subamortecido temos uma atenuação de:

$$-40\frac{dB}{dec}$$

A partir da frequência natural do sistema, como podemos observar abaixo:

```
    wn = 10
    coefAmort = 0.5
    num2 = wn**2
    den2 = [1, 2*coefAmort*wn, wn**2]
    Gs2 = control.TransferFunction(num2, den2)
    (mag2, phase_rad2, w2) = control.bode_plot(Gs2, dB=True)
    #cai 40db por decada a partir de wn
    #modulo em 10rad/s = (2*0.5)**-1 = 1
```

Listing 9: Atenuacao 2 ordem

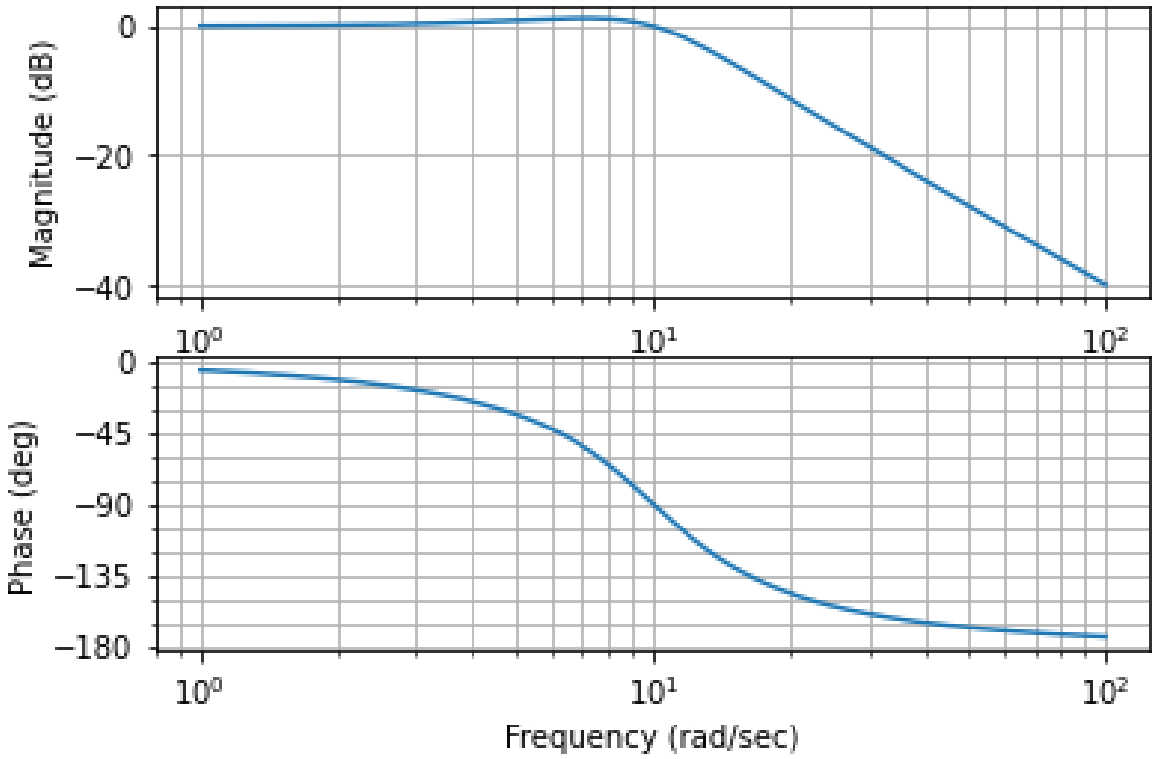

Figure 13: Atenuação para um sistema de segunda ordem