

# Modelagem de Sistemas Dinâmicos - Trabalho N°1

Leonardo Soares da Costa Tanaka - DRE: 121067652

Engenharia de Controle e Automação/UFRJ

Rio de Janeiro, Brasil

## 1 Introdução

Considerando um sistema linear invariante no tempo de  $u(t)$ , saída  $y(t)$  e função de transferência dada por:

$$H(s) = \frac{100}{16} \frac{s^2 + 16}{s^2 + 0.2s + 100} \quad (1)$$

Com essa função de transferência, é possível obter os zeros e pólos do sistema utilizando a Fórmula de Bhaskara no numerador e denominador da função de transferência. Os valores dos zeros e pólos do sistema são:  $z_1 = 4j$ ,  $z_2 = -4j$ ,  $p_1 = -0,1 + 9,9995j$  e  $p_2 = -0,1 - 9,9995j$ .

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (2)$$

Plotando o Diagrama de Bode em Python com o seguinte código:

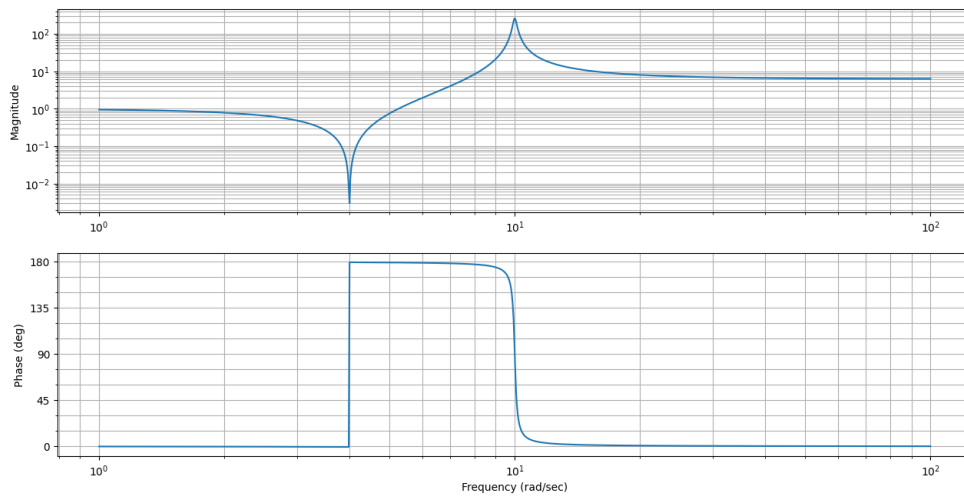


Figura 1: Diagrama de Bode

```

import control as ctrl
import matplotlib.pyplot as plt

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
H = ctrl.TransferFunction(num, den)

# 2. Plotar o diagrama de Bode
ctrl.bode_plot(H)
plt.show()

```

O diagrama de Bode é uma ferramenta muito útil, usada para analisar o comportamento de um sistema em frequências diferentes. Ele é usado para plotar a resposta em frequência de um sistema, mostrando como a amplitude e a fase de um sinal de entrada mudam em relação à frequência.

Então, é possível que observar pelo diagrama de Bode que haverá comportamento bem característicos nas frequências de 1 rad/s, 4 rad/s, 10 rad/s e 100 rad/s em suas magnitudes e fases, que são justamente as frequências dos cossenos escolhidos para entrada do sistema nas questões propostas.

Plotando o gráfico da resposta ao impulso em Python com o seguinte código:

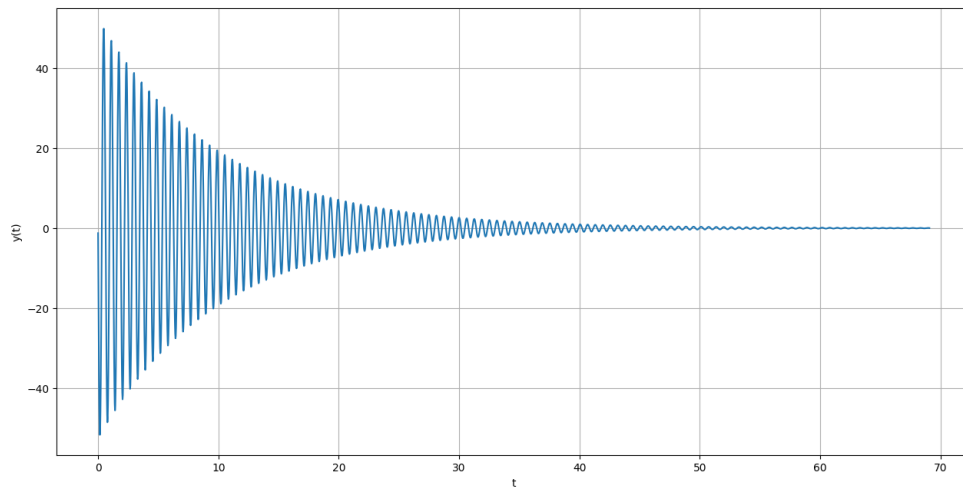


Figura 2: Resposta ao impulso

```

import numpy as np
import matplotlib.pyplot as plt
import control as ctrl

```

```

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
sys = ctrl.TransferFunction(num, den) # criar o objeto que representa o sistema

# 2. Calcular a resposta ao impulso
t, h = ctrl.impulse_response(sys)

# 3. Plotar do grafico da resposta ao impulso
plt.plot(t, h)
plt.xlabel('t(s)')
plt.ylabel('y(t)')
plt.grid()
plt.show()

```

Observar o gráfico de resposta ao impulso de um sistema é importante porque ele fornece informações sobre como o sistema responde a um impulso de entrada. No exemplo do trabalho, é possível perceber que a resposta ao impulso, é uma senoíde exponencialmente amortecida.

Por meio da tabela, foi calculado o Laplace do cosseno de frequência *omega* multiplicado pelo degrau unitário:

$$U(s) = \mathcal{L}\{\cos(\omega t)1(t)\} = \frac{s}{s^2 + \omega^2} \quad (3)$$

Calculando o Laplace da saída do sistema sem substituir o  $\omega$  da fórmula acima:

$$Y(s) = H(s)U(s) = \frac{100}{16} \frac{s^2 + 16}{s^2 + 0.2s + 100} \frac{s}{s^2 + \omega^2} = \frac{100}{16} \left[ \frac{k_1 s + k_2}{s^2 + 0.2s + 100} + \frac{k_3 s + k_4}{s^2 + \omega^2} \right] \quad (4)$$

$$s^3 + 16s = k_1 s^3 + k_2 s^2 + k_1 \omega^2 s + k_2 \omega^2 + k_3 s^3 + 0.2k_3 s^2 + 100k_3 s + k_4 s^2 + 0.2k_4 s + 100k_4 \quad (5)$$

$$\begin{cases} k_1 + k_3 = 1 \\ k_2 + 0.2k_3 + k_4 = 0 \\ \omega^2 k_1 + 100k_3 + 0.2k_4 = 16 \\ \omega^2 k_2 + 100k_4 = 0 \end{cases} \quad (6)$$

Foi feito isso para que não seja necessário repetir procedimentos no desenvolvimento do trabalho.

## 2 Desenvolvimento

1. Foi considerado uma entrada  $u(t) = \cos(t)1(t)$ . Foi obtido a  $y(t)$  por simulação numérica utilizando Python e as bibliotecas NumPy, Matplotlib e Control.

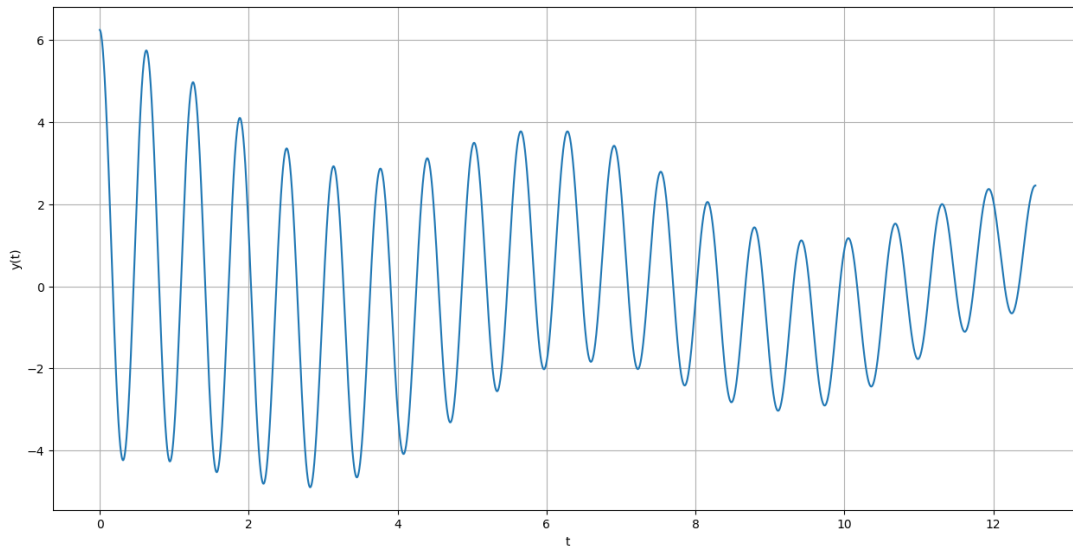


Figura 3: Resposta ao cosseno com frequência 1 multiplicado pelo degrau unitario

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

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
sys = ctrl.TransferFunction(num, den) # criar o objeto que representa o sistema

# 2. Definir os valores de tempo para simulacao
t = np.linspace(0, 4*np.pi, 10000) # valores de tempo de 0 a 4*pi segundos

# 3. Definir o sinal de entrada como o cosseno multiplicado pelo degrau unitario
u = np.heaviside(t, 1) * np.cos(t)

# 4. Realizar a simulacao da resposta do sistema usando a funcao 'control.forced_response()'
t_out, yout= ctrl.forced_response(sys, T=t, U=u)
```

```
# 5. Plotar o grafico da resposta
plt.plot(t_out, yout)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

Solução analítica:

2. Foi considerado uma entrada  $u(t) = \cos(4t)1(t)$  (frequência de zero). Foi obtido a resposta  $y(t)$  por simulação numérica utilizando Python e as bibliotecas NumPy, Matplotlib e Control.

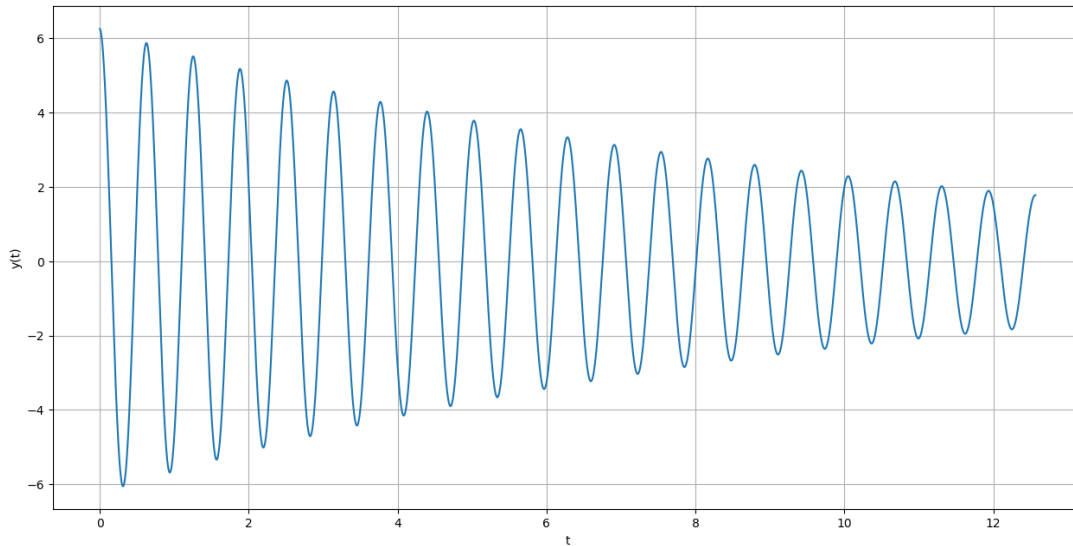


Figura 4: Resposta ao cosseno com frequência 4 multiplicado pelo degrau unitario

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

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
sys = ctrl.TransferFunction(num, den) # criar o objeto que representa o sistema

# 2. Definir os valores de tempo para simulacao
t = np.linspace(0, 4*np.pi, 10000) # valores de tempo de 0 a 4*pi segundos

# 3. Definir o sinal de entrada como o cosseno multiplicado pelo degrau unitario
u = np.heaviside(t, 1) * np.cos(4*t)

# 4. Realizar a simulacao da resposta do sistema usando a funcao 'control.forced_response()'
t_out, yout = ctrl.forced_response(sys, T=t, U=u)
```

```
# 5. Plotar o grafico da resposta
plt.plot(t_out, yout)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

Solução analítica:

3. Foi considerado uma entrada  $u(t) = \cos(10t)1(t)$ . Foi obtido a resposta  $y(t)$  por simulação numérica utilizando Python e as bibliotecas NumPy, Matplotlib e Control.

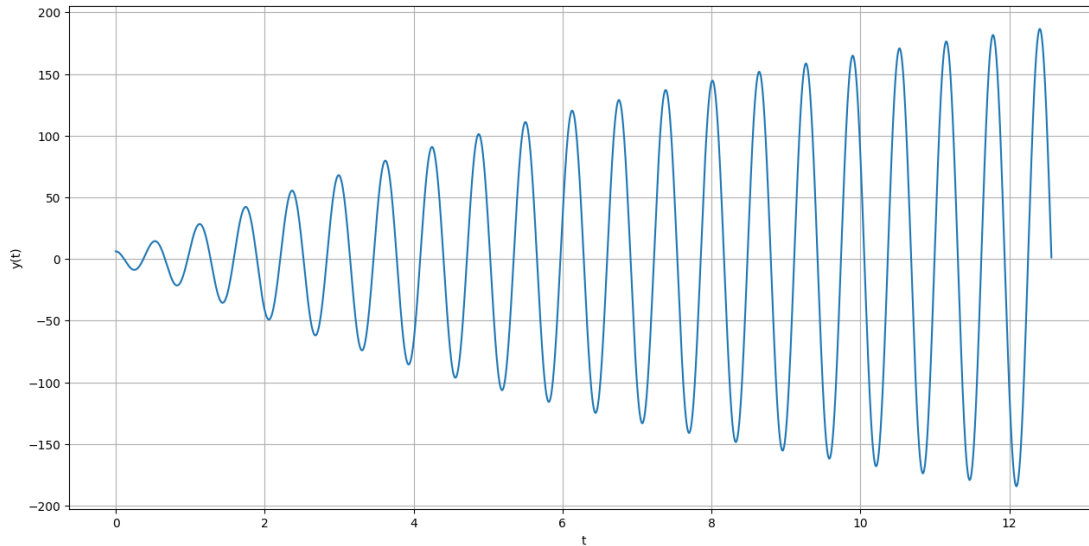


Figura 5: Resposta ao cosseno com frequência 10 multiplicado pelo degrau unitario

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

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
sys = ctrl.TransferFunction(num, den) # criar o objeto que representa o sistema

# 2. Definir os valores de tempo para simulacao
t = np.linspace(0, 4*np.pi, 10000) # valores de tempo de 0 a 4*pi segundos

# 3. Definir o sinal de entrada como o cosseno multiplicado pelo degrau unitario
u = np.heaviside(t, 1) * np.cos(10*t)

# 4. Realizar a simulacao da resposta do sistema usando a funcao 'control.forced_response()'
t_out, yout = ctrl.forced_response(sys, T=t, U=u)
```



```
# 5. Plotar o grafico da resposta
plt.plot(t_out, yout)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

Solução analítica:

4. Foi considerado uma entrada  $u(t) = \cos(100t)1(t)$ . Foi obtido a resposta  $y(t)$  por simulação numérica utilizando Python e as bibliotecas NumPy, Matplotlib e Control.

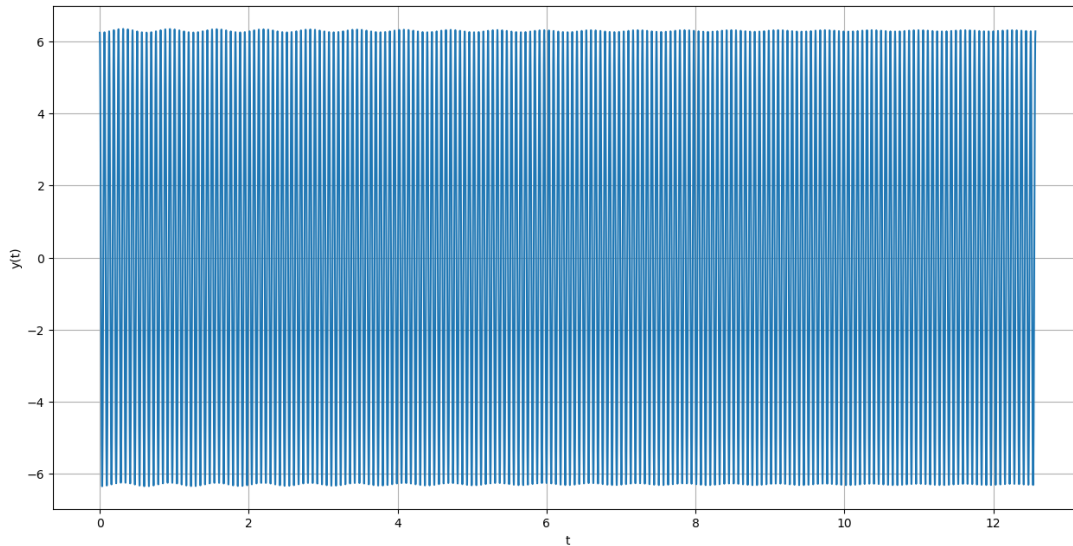


Figura 6: Resposta ao cosseno com frequência multiplicado pelo degrau unitario

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

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
sys = ctrl.TransferFunction(num, den) # criar o objeto que representa o sistema

# 2. Definir os valores de tempo para simulacao
t = np.linspace(0, 4*np.pi, 10000) # valores de tempo de 0 a 4*pi segundos

# 3. Definir o sinal de entrada como o cosseno multiplicado pelo degrau unitario
u = np.heaviside(t, 1) * np.cos(100*t)

# 4. Realizar a simulacao da resposta do sistema usando a funcao 'control.forced_response()'
t_out, yout = ctrl.forced_response(sys, T=t, U=u)
```

```
# 5. Plotar o grafico da resposta
plt.plot(t_out , yout)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

Solução analítica:

### 3 Conclusão

A resposta de um sistema linear é afetada pela localização de seus pólos e zeros. Os pólos determinam a estabilidade e a forma como o sistema responde às diferentes entradas. Em geral, se o sistema tem pólos na parte direita do plano complexo (parte real positiva), o sistema é instável e não pode ser utilizado em aplicações práticas.

Por outro lado, se os pólos estão na parte esquerda do plano complexo (parte real negativa), o sistema é estável e pode ser usado em aplicações práticas. A localização dos pólos também afeta a rapidez com que o sistema responde a uma entrada. Quanto mais longe os pólos estiverem do eixo imaginário, mais rápido será a resposta do sistema.

Os zeros, por outro lado, afetam a forma como o sistema responde a diferentes frequências de entrada. Um zero em uma frequência específica anula a resposta do sistema a essa frequência, enquanto um zero próximo a uma frequência específica pode reduzir a amplitude da resposta do sistema a essa frequência.

## 4 Extras

Plotando o Diagrama de Nyquist em Python com o seguinte código (Feito por curiosidade):

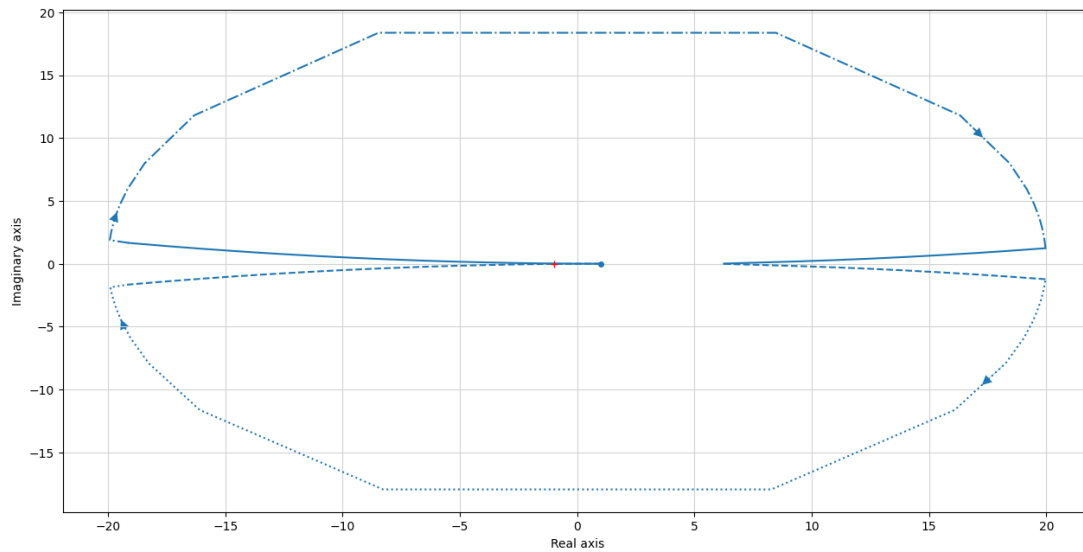


Figura 7: Diagrama de Nyquist

```
import control as ctrl
import matplotlib.pyplot as plt

# 1. Definir a funcao de transferencia do sistema
num = [100, 0, 1600] # numerador da funcao de transferencia
den = [16, 3.2, 1600] # denominador da funcao de transferencia
H = ctrl.TransferFunction(num, den)

# 2. Plotar o diagrama de Nyquist
ctrl.nyquist_plot(H)
plt.show()
```