Modelagem de Sistemas Dinâmicos - Trabalho Nº1

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:

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

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 $p_2 = -0,1 - 9,9995j$.

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

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

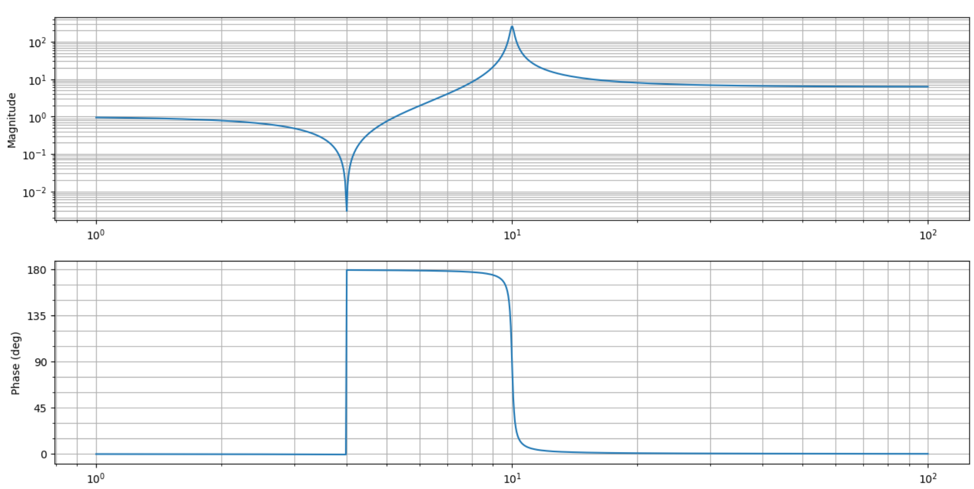

Figura 1: Diagrama de Bode

```
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.

Escrevendo a função de transferência com resposta em frequência:

$$H(jw) = H(s)|_{s=jw} = \frac{100}{16}\frac{(jw)^2+16}{(jw)^2+0.2jw+100} = \frac{100}{16}\frac{16-w^2}{0.2jw+100-w^2} \tag{3}$$

Calculando o módulo da função de trasnferência:

$$|H(jw)| = \frac{100}{16}\frac{\sqrt{(16-w^2)^2}}{\sqrt{(0.2)^2+(100-w^2)^2}} \tag{4}$$

Calculando a fase da função de transferência:

$$\angle H(jw) = arctan(0/(16-w^2)) - arctan(0.2w/(100-w^2)) \tag{5}$$

2 Questões

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.

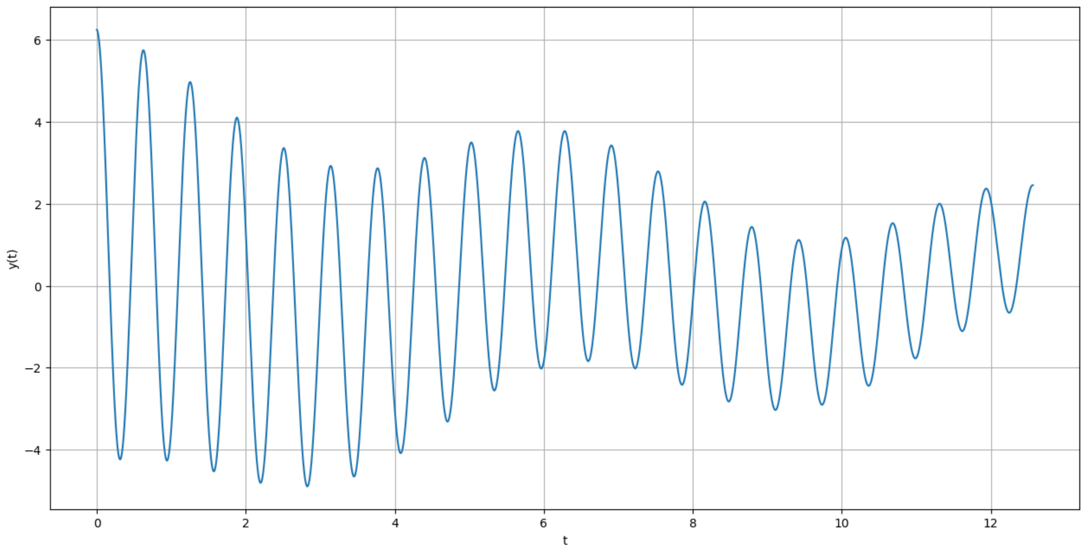

Figura 2: Resposta ao cosseno com frequência 1 multiplicado pelo degrau unitario

```
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:

$$y_{ss}(t) = |H(j)| \cdot cos(t + \angle H(j)) = \frac{100}{16} \frac{\sqrt{(16-1^2)^2}}{\sqrt{0.04+(100-1^2)^2}} \cdot cos(t + arctan(0/(16-1^2)) - arctan(0.2/(100-1^2))) =$$
$$= \frac{100}{16} \frac{15}{\sqrt{0.04+99^2}} \cdot cos(t + 0 - arctan(0.2/99.0002)) = \frac{100}{16} \frac{15}{99.000202} \cdot cos(t + 0.00202) = 0.946968 \cdot cos(t + 0.00202)$$

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.

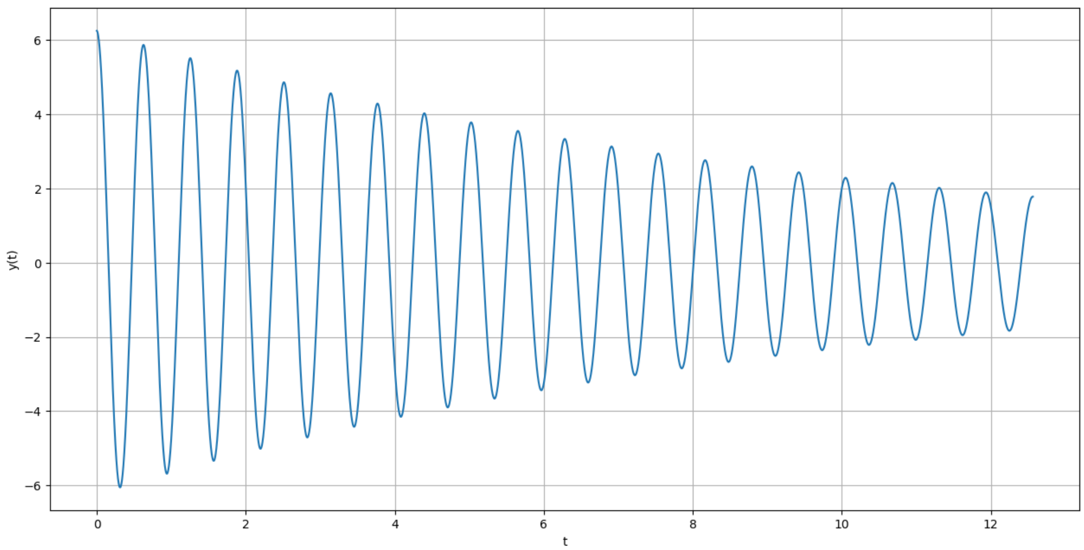

Figura 3: Resposta ao cosseno com frequência 4 multiplicado pelo degrau unitario

```
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:

$y_{ss}(t) = |H(j)| \cdot cos(4t + \angle H(j)) = \frac{100}{16}\frac{\sqrt{(16-4^2)^2}}{\sqrt{0.04+(100-4^2)^2}} \cdot cos(4t + arctan(0/(16-4^2)) - arctan(0.8/(100-4^2))) =$
$= \frac{100}{16}\frac{0}{\sqrt{0.04+84^2}} \cdot cos(4t + 0 - arctan(0.8/84)) = 0$

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.

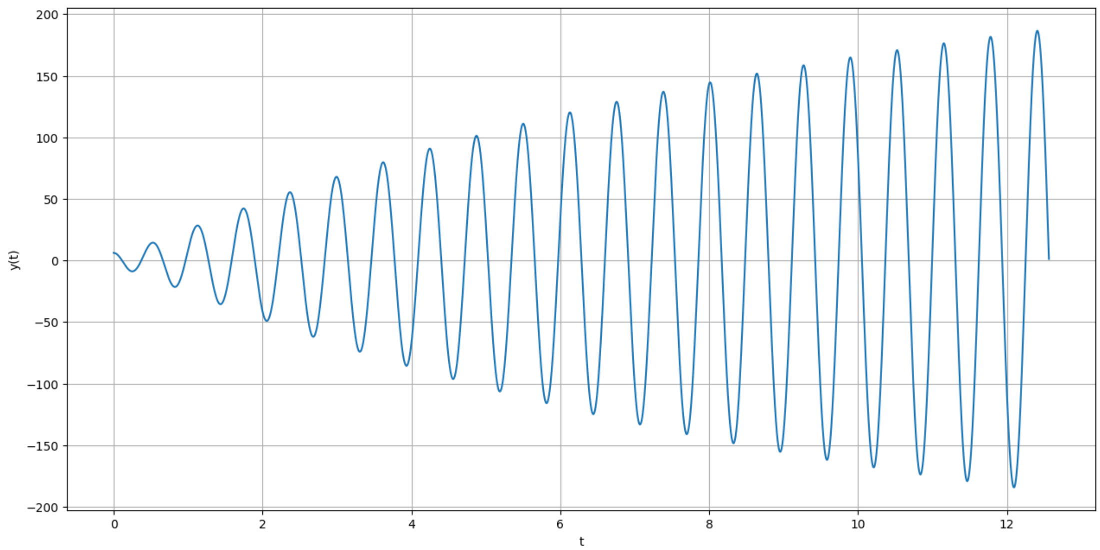

Figura 4: Resposta ao cosseno com frequência 10 multiplicado pelo degrau unitario

```
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:

$y_{ss}(t) = |H(j)| \cdot cos(10t + \angle H(j)) = \frac{100}{16} \frac{\sqrt{(16-10^2)^2}}{\sqrt{0.04+(100-10^2)^2}} \cdot cos(10t + arctan(0/(16-10^2)) - arctan(2/(100-10^2))) =$
$= \frac{100}{16} \frac{84}{\sqrt{0.04}} \cdot cos(10t + \pi - arctan(2/0)) = \frac{100}{16} \frac{84}{0.2} \cdot cos(10t + \pi - \pi/2) = 2625 \cdot cos(10t + \pi/2)$

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.

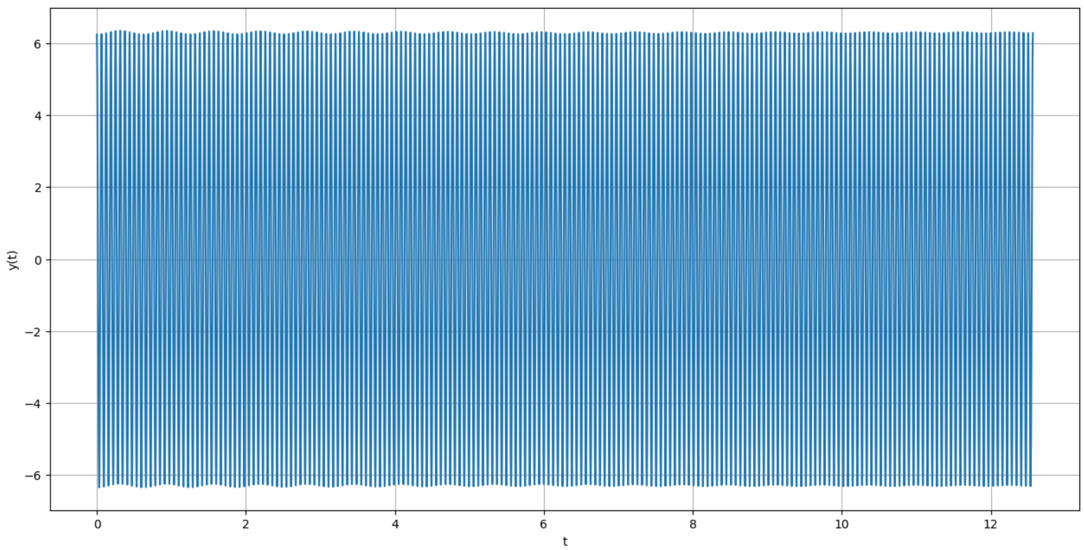

Figura 5: Resposta ao cosseno com frequência multiplicado pelo degrau unitario

```
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:

$y_{ss}(t) = |H(j)| \cdot cos(100t + \angle H(j)) = \frac{100}{16} \frac{\sqrt{(16-100^2)^2}}{\sqrt{0.04+(100-100^2)^2}} \cdot cos(100t + arctan(0/(16-100^2)) - arctan(20/(100-100^2))) =$
$= \frac{100}{16} \frac{9984}{\sqrt{0.04+(-9900)^2}} \cdot cos(100t + \pi - arctan(20/-9900)) = \frac{100}{16} \frac{9984}{9900} \cdot cos(100t + \pi - 0) = 6.303 \cdot cos(100t + \pi)$

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):

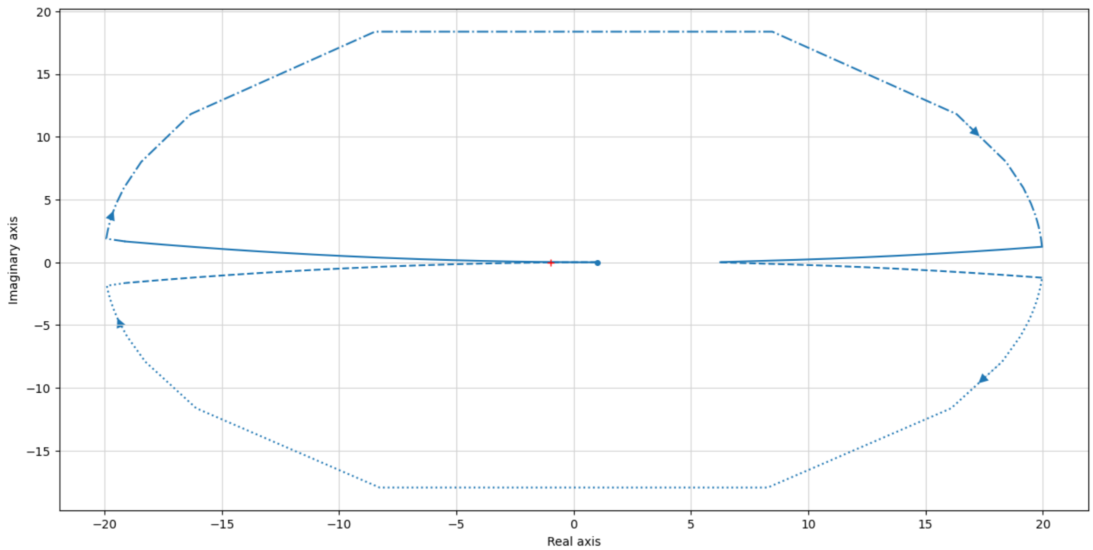

Figura 6: Diagrama de Nyquist

```
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, omega=None, plot=True)
plt.show()
```