

# Modelagem de Sistemas Dinâmicos - Trabalho N<sup>o</sup>1

Leonardo Soares da Costa Tanaka - DRE: 121067652

Engenharia de Controle e Automação/UFRJ

Rio de Janeiro, Brasil

Maio de 2023

## 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:

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

Os valores dos zeros e pólos do sistema são:  $z_1 = 4j$ ,  $z_2 = -4j$ ,  $p_1 = -0.1 + 9.9995j$  e  $p_2 = -0.1 - 9.9995j$ .

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

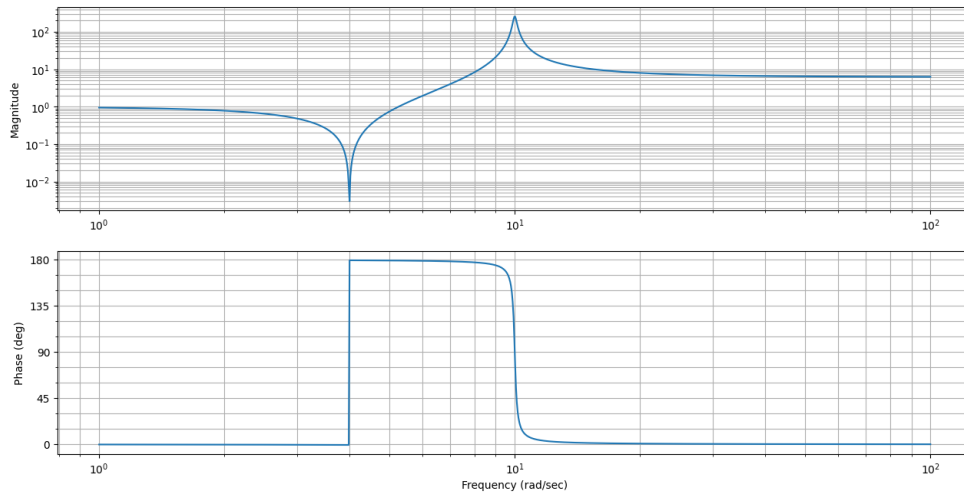


Figura 1: Diagrama de Bode

```

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 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:

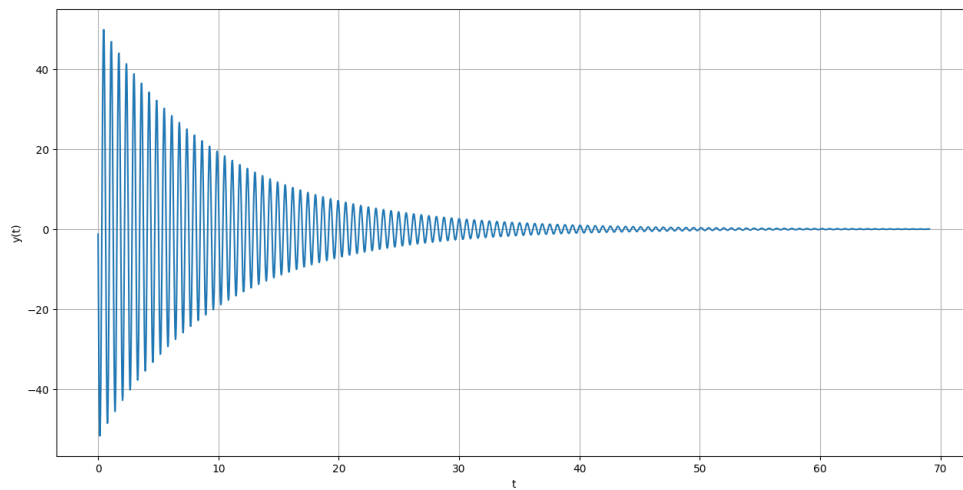


Figura 2: Resposta ao impulso

```

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:

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

Para determinar os valores de  $k_1$ ,  $k_2$ ,  $k_3$  e  $k_4$ , podemos igualar o numerador da função original aos numeradores das frações parciais:

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

Montando o sistema de equações resultante com a frequência  $\omega$  para ser utilizado no desenvolvimento:

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

## 2 Desenvolvimento

1. Foi considerado uma entrada  $u(t) = \cos(t)1(t)$ . Foi obtido a  $y(t)$  por simulação numérica. Solução analítica ( $\omega = 1$ ):

$$\begin{cases} k_1 + k_3 = 1 \\ k_2 + 0.2k_3 + k_4 = 0 \\ k_1 + 100k_3 + 0.2k_4 = 16 \\ k_2 + 100k_4 = 0 \end{cases} \quad (7)$$

Obtendo a solução do sistema, por meio da utilização da calculadora científica:

$$k_1 = 0.8484; k_2 = -0.0306; k_3 = 0.1515; k_4 = 0.0003. \quad (8)$$

Substituindo  $k_1$ ,  $k_2$ ,  $k_3$  e  $k_4$  pelos valores encontrados na Transformada de Laplace da saída  $Y(s)$ :

$$Y(s) = \frac{100}{16} \left[ \frac{0.8484s - 0.0306}{(s + 0.1)^2 + 99.99} + \frac{0.1515s + 0.0003}{s^2 + 1^2} \right] \quad (9)$$

Organizando a função para facilitar a aplicação da Transformada Inversa de Laplace:

$$Y(s) = \frac{100}{16} \left[ \frac{0.8484(s + 0.1)}{(s + 0.1)^2 + 9.999^2} - \frac{0.0115 \cdot 9.999}{(s + 0.1)^2 + 9.999^2} + \frac{0.1515s}{s^2 + 1^2} + \frac{0.0003}{s^2 + 1^2} \right] \quad (10)$$

Aplicação da Transformada de Laplace de Laplace Inversa:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = \frac{100}{16} [0.8484e^{-0.1t}\cos(9.999t) - 0.0115e^{-0.1t}\sin(9.999t) + 0.1515\cos(t) + 0.0003\sin(t)] 1(t) \quad (11)$$

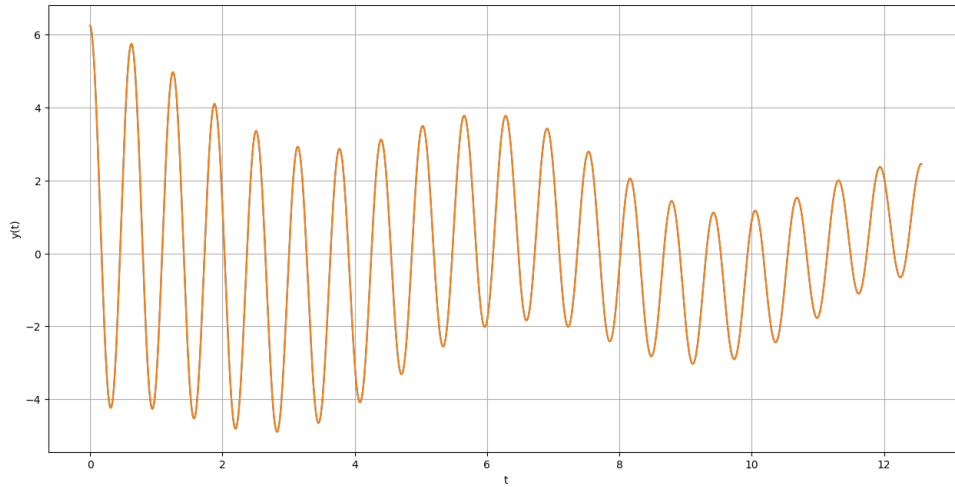


Figura 3: Resposta ao cosseno com frequência 1 multiplicado pelo degrau unitario

```
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. Resposta analitica
y_a = 100/16*(0.8484*np.exp(-0.1*t)*np.cos(9.999*t)-0.0115*np.exp(-0.1*t)*np.sin(9.999*t) +
            0.1515*np.cos(t) + 0.0003*np.sin(t))

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

(a) A resposta coincide com o esperado na solução analítica?

Sim, porque os valores obtidos graficamente da simulação numérica coincidiu bastante com a solução analítica. Podendo observar essa coincidência entre os métodos pela Figura 3 em que os dois traçados dos gráficos se sobressaem. Além disso, é possível observar o comportamento tanto de senoídes quanto de senoídes exponencialmente amortecidas.

Observando a Figura 4 a seguir, é possível observar que existe um erro quadrático considerável ao percorrer um intervalo de  $4\pi$  que é causado tanto pela aproximação numérica feita pela linguagem Python e biblioteca Numpy quanto pelo erro de arredondamento na solução analítica.

```
# 7. Calculando o erro quadratico
y_error = (yout-y_a)**2
```

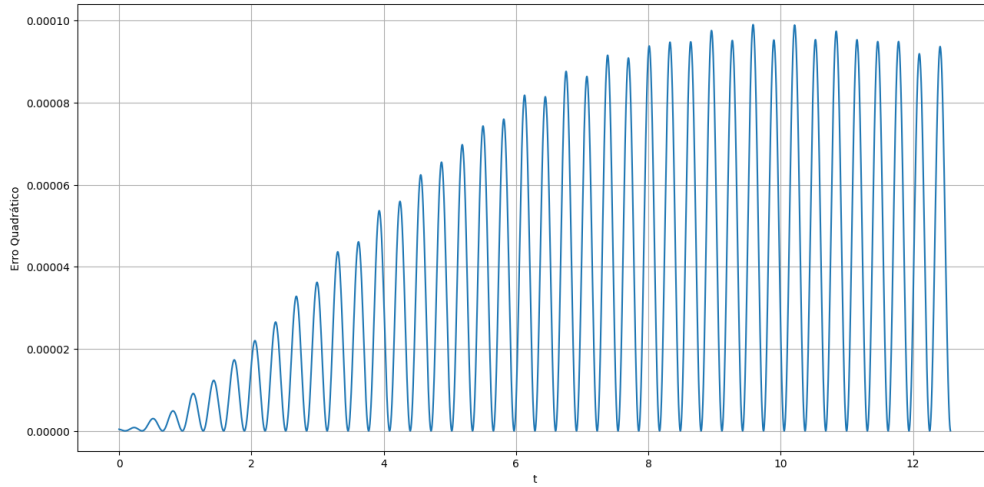


Figura 4: Erro quadrático ( $\omega = 1$ )

(b) Que frequências aparecem na resposta? A que estão associadas estas frequências?

Observando a Figura 3, as frequências que aparecem na resposta são aproximadamente 1 rad/s e 9.999 rad/s. A frequência 1 rad/s está relacionada a entrada que é um cosseno de frequência 1 rad/s. Além disso, a frequência 9.999 rad/s está relacionada a função de transferência do sistema que possui uma resposta ao impulso sendo uma senoíde exponencialmente amortecida de frequência 9.999 rad/s.

2. Foi considerado uma entrada  $u(t) = \cos(4t)1(t)$  (frequência de zero). Foi obtido a resposta  $y(t)$  por simulação numérica. Solução analítica ( $\omega = 4$ ):

$$\begin{cases} k_1 + k_3 = 1 \\ k_2 + 0.2k_3 + k_4 = 0 \\ 16k_1 + 100k_3 + 0.2k_4 = 16 \\ 16k_2 + 100k_4 = 0 \end{cases} \quad (12)$$

Obtendo a solução do sistema, por meio da utilização da calculadora científica:

$$k_1 = 1; k_2 = 0; k_3 = 0; k_4 = 0. \quad (13)$$

Substituindo  $k_1$ ,  $k_2$ ,  $k_3$  e  $k_4$  pelos valores encontrados na Transformada de Laplace da saída  $Y(s)$ . Organizando a função para facilitar a aplicação da Transformada Inversa de Laplace:

$$Y(s) = \frac{100}{16} \left[ \frac{s}{(s+0.1)^2 + 99.99} \right] = \frac{100}{16} \left[ \frac{(s+0.1)}{(s+0.1)^2 + 9.999^2} - \frac{0.01 \cdot 9.999}{(s+0.1)^2 + 9.999^2} \right] \quad (14)$$

Aplicação da Transformada de Laplace de Laplace Inversa:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = \frac{100}{16} [e^{-0.1t} \cos(9.999t) - 0.01e^{-0.1t} \sin(9.999t)] 1(t) \quad (15)$$

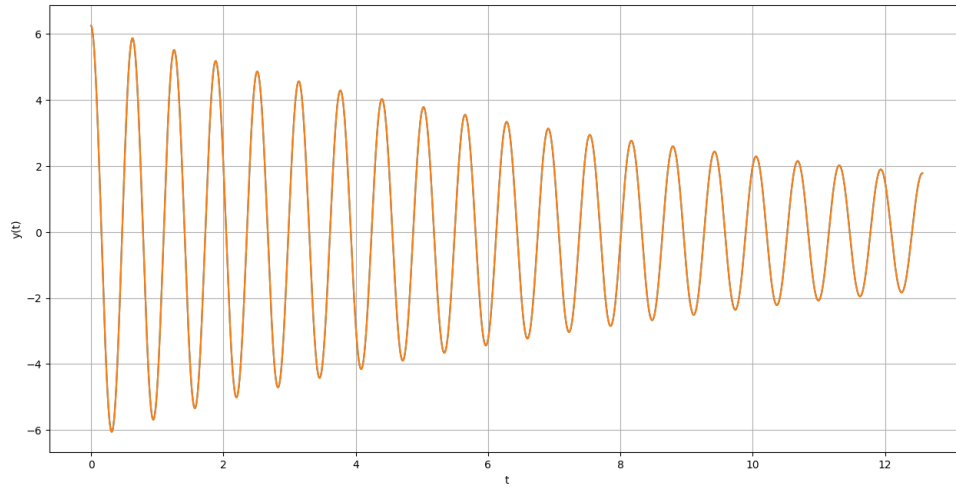


Figura 5: Resposta ao cosseno com frequência 4 multiplicado pelo degrau unitario

```
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. Resposta analitica
y_a = 100/16*(np.exp(-0.1*t)*np.cos(9.999*t)-0.01*np.exp(-0.1*t)*np.sin(9.999*t))
# 6. Plotar o grafico da resposta
plt.plot(t_out, yout)
plt.plot(t, y_a)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

(a) A resposta coincide com o esperado na solução analítica?

Sim, porque os valores obtidos graficamente da simulação numérica coincidiu bastante com a solução analítica. Podendo observar essa coincidência entre os métodos pela Figura 5 em que os dois traçados dos gráficos se sobressaem. Além disso, é possível observar o comportamento de uma senoíde exponencialmente amortecida.

Observando a Figura 6 a seguir, é possível observar que existe um erro quadrático considerável ao percorrer um intervalo de  $4\pi$  que é causado tanto pela aproximação numérica feita pela linguagem Python e biblioteca Numpy quanto pelo erro de arredondamento na solução analítica.

```
# 7. Calculando o erro quadratico
y_error = (yout-y_a)**2
```



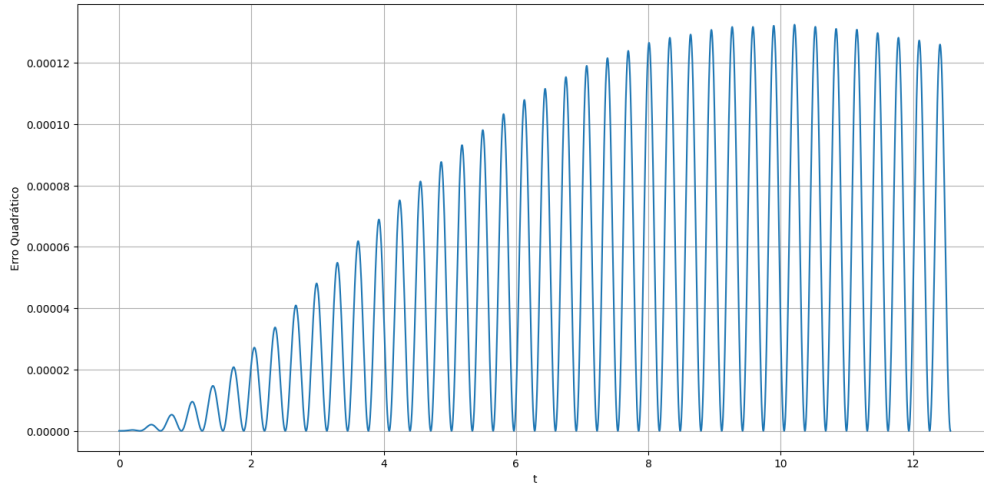


Figura 6: Erro quadrático ( $\omega = 4$ )

(b) Que frequências aparecem na resposta? A que estão associadas estas frequências?

Observando a Figura 5, a única frequência que aparece na resposta é aproximadamente 9.999 rad/s. A frequência 9.999 rad/s está relacionada a função de transferência do sistema que possui uma resposta ao impulso sendo uma senoíde exponencialmente amortecida.

(c) Aparece a frequência 4 rad/s na resposta? Caso negativo, porque?

Não aparece a frequência 4 rad/s na resposta. Porque, os zeros da função de transferência afetam a resposta em frequência do sistema, anulando a amplitude da resposta nessa frequência específica. Quando a entrada é um cosseno com frequência igual a 4 rad/s, a função de transferência do sistema irá multiplicar essa entrada pelos fatores  $(s - 4j)$  e  $(s + 4j)$ . Como ambos os zeros estão localizados em  $\pm 4j$  na parte imaginária do plano complexo, esses fatores serão iguais a zero na frequência de 4 rad/s, o que significa que a resposta do sistema a essa frequência será nula.

3. Foi considerado uma entrada  $u(t) = \cos(10t)1(t)$ . Foi obtido a resposta  $y(t)$  por simulação numérica. Solução analítica ( $\omega = 10$ ):

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

Obtendo a solução do sistema, por meio da utilização da calculadora científica:

$$k_1 = 1; k_2 = 420; k_3 = 0; k_4 = -420. \quad (17)$$

Substituindo  $k_1$ ,  $k_2$ ,  $k_3$  e  $k_4$  pelos valores encontrados na Transformada de Laplace da saída  $Y(s)$ :

$$Y(s) = \frac{100}{16} \left[ \frac{s + 420}{(s + 0.1)^2 + 99.99} - \frac{420}{s^2 + 10^2} \right] \quad (18)$$

Organizando a função para facilitar a aplicação da Transformada Inversa de Laplace:

$$Y(s) = \frac{100}{16} \left[ \frac{(s + 0.1)}{(s + 0.1)^2 + 9.999^2} + \frac{41.9941 \cdot 9.999}{(s + 0.1)^2 + 9.999^2} - \frac{42 \cdot 10}{s^2 + 10^2} \right] \quad (19)$$

Aplicação da Transformada de Laplace de Laplace Inversa:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = \frac{100}{16} \left[ e^{-0.1t} \cos(9.999t) + 41.9941 e^{-0.1t} \sin(9.999t) - 42 \sin(10t) \right] 1(t) \quad (20)$$

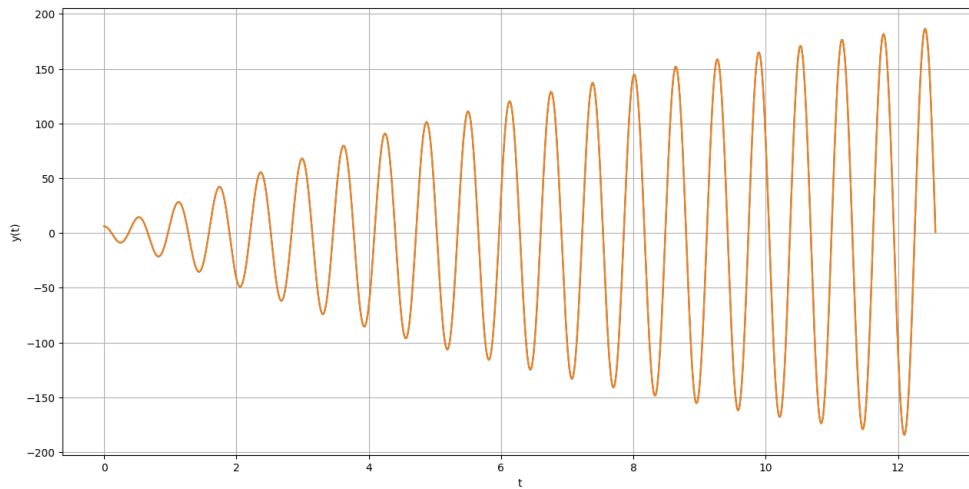


Figura 7: Resposta ao cosseno com frequência 10 multiplicado pelo degrau unitario

```
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. Resposta analitica
y_a = 100/16*(np.exp(-0.1*t)*np.cos(9.999*t)+41.9941*np.exp(-0.1*t)*np.sin(9.999*t)-
            42*np.sin(10*t))

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

(a) A resposta coincide com o esperado na solução analítica?

Sim, porque os valores obtidos graficamente da simulação numérica coincidiu bastante com a solução analítica. Podendo observar essa coincidência entre os métodos pela Figura 7 em que os dois traçados dos gráficos se sobressaem.

Observando a Figura 8 a seguir, é possível observar que existe um erro quadrático 10 vezes maior que os demais ao percorrer um intervalo de  $4\pi$  que é causado tanto pela aproximação numérica feita pela linguagem Python e biblioteca Numpy quanto pelo erro de arredondamento na solução analítica.

```
# 7. Calculando o erro quadratico
y_error = (yout-y_a)**2
```

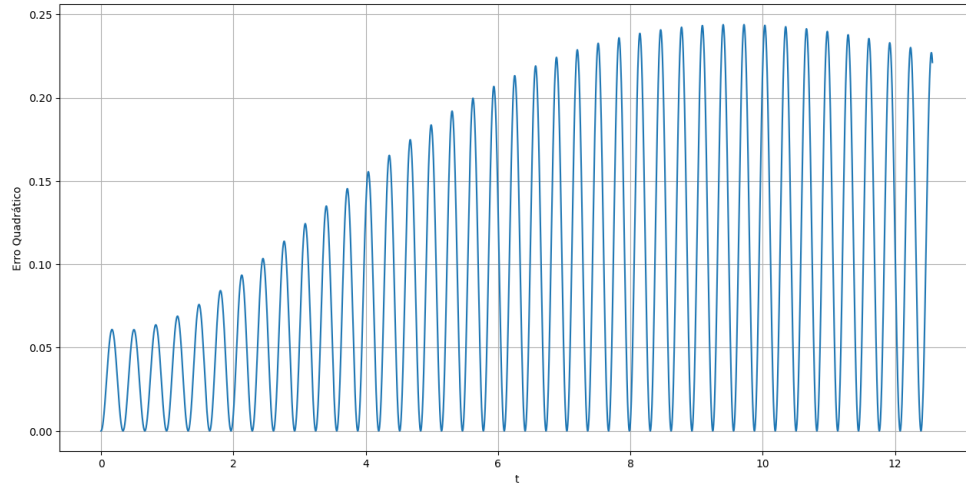


Figura 8: Erro quadrático ( $\omega = 10$ )

(b) Que frequências aparecem na resposta? A que estão associadas estas frequências?

As frequências que aparecem na resposta são aproximadamente 10 rad/s e 9.999 rad/s. A frequência 10 rad/s está relacionada a entrada que é um cosseno de frequência 10 rad/s. Além disso, a frequência 9.999 rad/s está relacionada a função de transferência do sistema que possui uma resposta ao impulso sendo uma senoíde exponencialmente amortecida. Porém, é muito difícil distinguir as duas frequências pelo fato de terem valores muito próximos e estarem em fase. Mesmo que seja aumentado o intervalo de amostragem ainda seria muito difícil de visualizar.

4. Foi considerado uma entrada  $u(t) = \cos(100t)1(t)$ . Foi obtido a resposta  $y(t)$  por simulação numérica. Solução analítica ( $\omega = 100$ ):

$$\begin{cases} k_1 + k_3 = 1 \\ k_2 + 0.2k_3 + k_4 = 0 \\ 10000k_1 + 100k_3 + 0.2k_4 = 16 \\ 10000k_2 + 100k_4 = 0 \end{cases} \quad (21)$$

Obtendo a solução do sistema, por meio da utilização da calculadora científica:

$$k_1 = -0.0085; k_2 = 0.0020; k_3 = 1.0085; k_4 = -0.2037. \quad (22)$$

Substituindo  $k_1$ ,  $k_2$ ,  $k_3$  e  $k_4$  pelos valores encontrados na Transformada de Laplace da saída  $Y(s)$ :

$$Y(s) = \frac{100}{16} \left[ \frac{-0.0085s + 0.0020}{(s + 0.1)^2 + 99.99} + \frac{1.0085s - 0.2037}{s^2 + 100^2} \right] \quad (23)$$

Organizando a função para facilitar a aplicação da Transformada Inversa de Laplace:

$$Y(s) = \frac{100}{16} \left[ -\frac{0.0085(s + 0.1)}{(s + 0.1)^2 + 9.999^2} + \frac{0.0010 \cdot 9.999}{(s + 0.1)^2 + 9.999^2} + \frac{1.0085s}{s^2 + 100^2} - \frac{0.0002 \cdot 100}{s^2 + 100^2} \right] \quad (24)$$

Aplicação da Transformada de Laplace de Laplace Inversa:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = \frac{100}{16} \left[ -0.0085e^{-0.1t}\cos(9.999t) + 0.0010e^{-0.1t}\sin(9.999t) + 1.0085\cos(100t) - 0.0002\sin(100t) \right] 1(t) \quad (25)$$

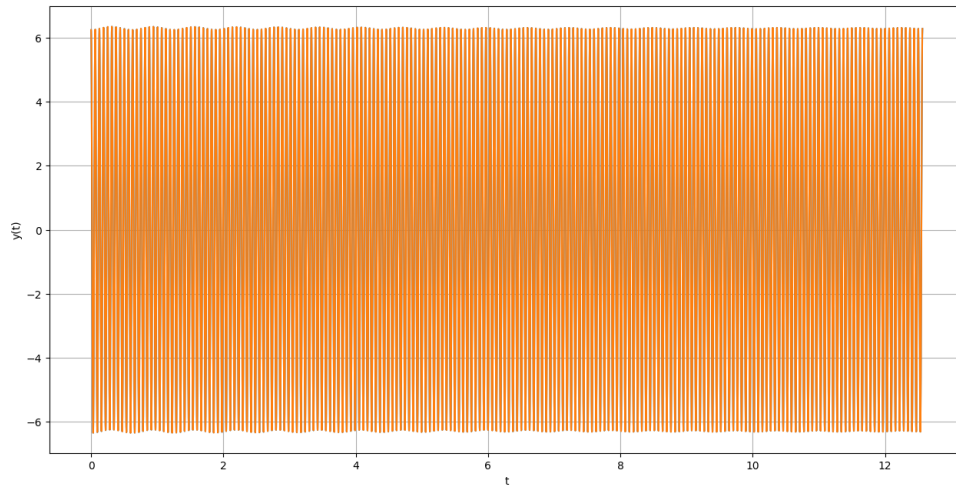


Figura 9: Resposta ao cosseno com frequência multiplicado pelo degrau unitario

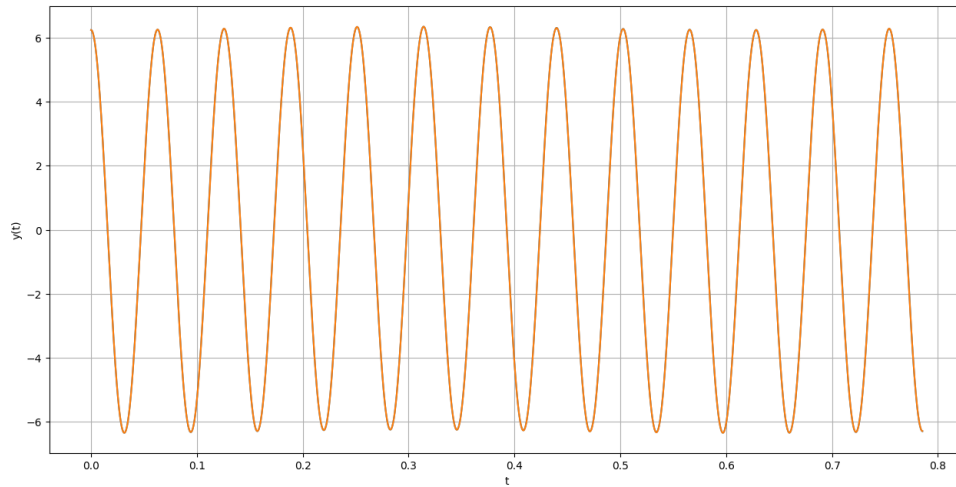


Figura 10: Resposta com um intervalo menor

```
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. Resposta analitica
y_a = 100/16*(-0.0085*np.exp(-0.1*t)*np.cos(9.999*t)+0.0010*np.exp(-0.1*t)*np.sin(9.999*t)+
    1.0085*np.cos(100*t)-0.0002*np.sin(100*t))
# 6. Plotar o grafico da resposta
plt.plot(t_out, yout)
plt.plot(t, y_a)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.show()
```

(a) A resposta coincide com o esperado na solução analítica?

Sim, porque os valores obtidos graficamente da simulação numérica coincidiu bastante com a solução analítica. Podendo observar essa coincidência entre os métodos pelas figuras 9 e 10 em que os dois traçados dos gráficos se sobressaem. Além disso, é possível observar o comportamento tanto de senoídes quanto de senoídes exponencialmente amortecidas.

Observando a Figura 11 a seguir, é possível observar que existe um erro quadrático considerável ao percorrer um intervalo de  $4\pi$  que é causado tanto pela aproximação numérica feita pela linguagem Python e biblioteca Numpy quanto pelo erro de arredondamento na solução analítica.

```
# 7. Calculando o erro quadratico
y_error = (yout-y_a)**2
```

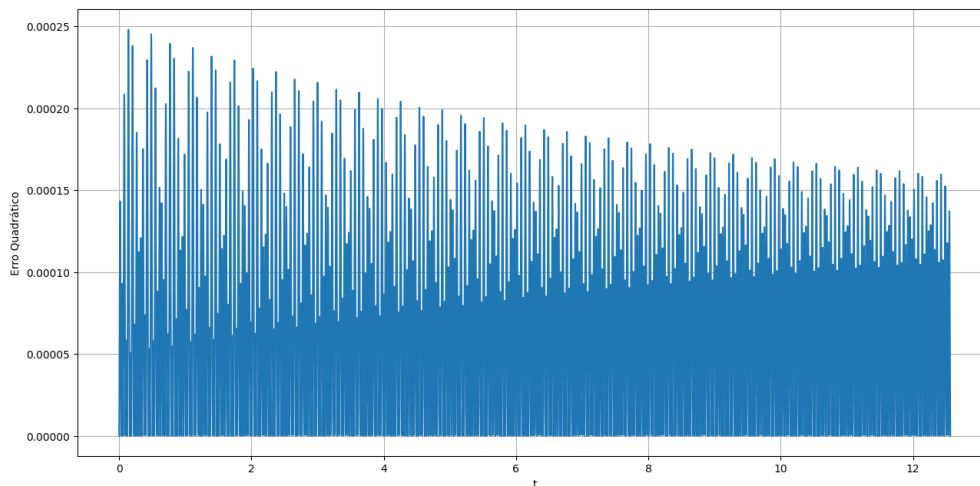


Figura 11: Erro quadrático ( $\omega = 100$ )

(b) Que frequências aparecem na resposta? A que estão associadas estas frequências?

Observando as figuras 9 e 10, as frequências que aparecem na resposta são aproximadamente 100 rad/s e 9.999 rad/s. A frequência 100 rad/s está relacionada a entrada que é um cosseno de frequência 1. Além disso, a frequência 9.999 rad/s está relacionada a função de transferência do sistema que possui uma resposta ao impulso sendo uma senoíde exponencialmente amortecida. Só é possível observar essas duas frequências, porque foram utilizados dois gráficos com intervalos diferentes de amostragem

### 3 Conclusão

A conclusão é que os pólos e zeros de um sistema linear têm um impacto significativo em sua resposta. A localização dos pólos determina a estabilidade do sistema e a rapidez com que ele responde a uma entrada. Se os pólos estiverem na parte direita do plano complexo, o sistema é instável e não pode ser usado em aplicações práticas. Por outro lado, se os pólos estiverem na parte esquerda do plano complexo, o sistema é estável e pode ser utilizado em aplicações práticas. Esse é o caso sistema linear do trabalho que possui dois pólos complexos conjugados com parte real negativa ( $p_1 = -0.1 + 9.9995j$  e  $p_2 = -0.1 - 9.9995j$ ).

Além disso, 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. Por outro lado, os zeros 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 (o que ocorre na questão 3(c) e é explicado nesse tópico), enquanto um zero próximo a uma frequência específica pode reduzir a amplitude da resposta do sistema a essa frequência.

Portanto, entender a localização dos pólos e zeros em um sistema linear é crucial para projetar sistemas estáveis e prever sua resposta a diferentes entradas.



## 4 Extras

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

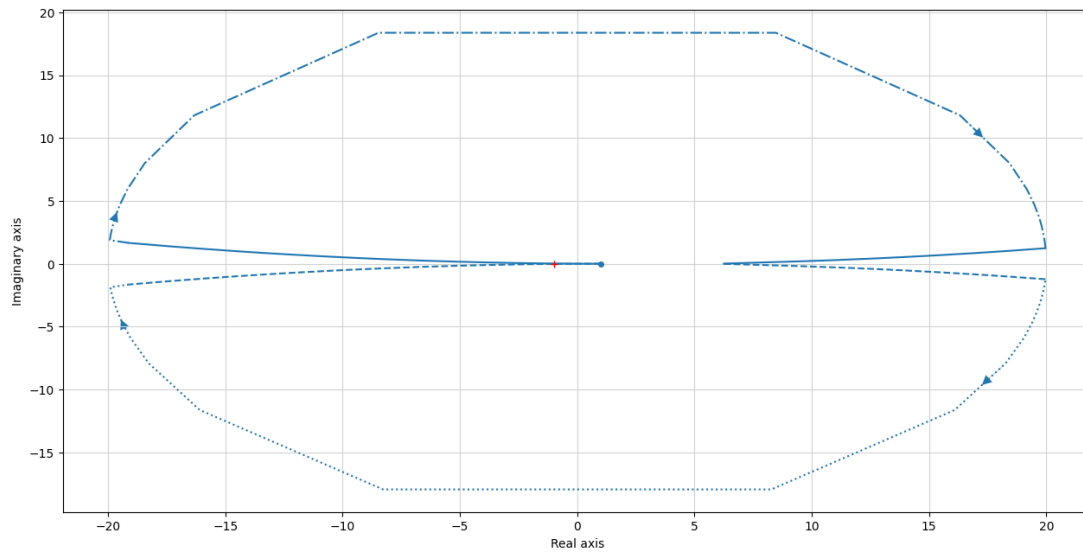


Figura 12: Diagrama de Nyquist

```
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()
```

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

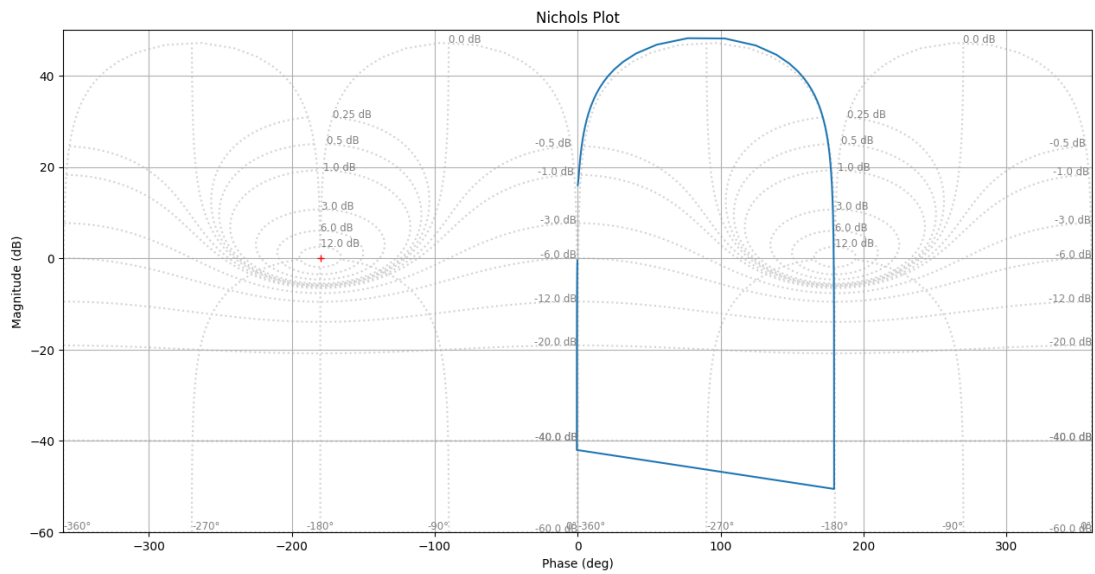


Figura 13: Diagrama de Nichols

```
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 Nichols
ctrl.nichols_plot(H)
plt.grid()
plt.show()
```

## Referências

- [1] Documentação do Python Control. Disponível em: <https://python-control.readthedocs.io/en/0.9.3.post2/>. Acesso em: 06 de maio de 2023.
- [2] Documentação do NumPy. Disponível em: <https://numpy.org/doc/stable/>. Acesso em: 06 de maio de 2023.
- [3] Documentação do Matplotlib. Disponível em: <https://matplotlib.org/stable/api/index>. Acesso em: 06 de maio de 2023.
- [4] LATHI, B.P. *Linear Systems and Signals*. 3rd ed. Oxford: Oxford University Press, 2018.
- [5] GOMES, A.C.N. Notas de Aula - Sinais e Sistemas (COE350). Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2022. Disponível em: <http://coep.ufrj.br/~afel/notas.pdf>. Acesso em: 06 de maio de 2023.