Rede Brasileira de Calibração

Laboratório de Instrumentação

Engenharia de Controle e Automação

Prof. Agnaldo J. Rocha Reis

1. Contratante: Universidade Federal de Ouro Preto.
2. Contratado: Douglas Meneses Barbosa - 19.2.1021, Guilherme Iannini Dutra - 20.1.1272, Luana Cristina da Costa - 18.2.1022, Wellington Resende de Araújo Júnior - 19.1.1062.

Ouro Preto

Junho de 2022

Objetivo

Determinar os parâmetros de interesse (sensibilidade estática, constante de tempo, quociente de amortecimento e frequência natural não amortecida) a fim de se obter um modelo matemático representativo do sistema.

Obs: O trabalho foi desenvolvido na linguagem python.

Instrumentos de Primeira Ordem

Para determinar o comportamento dinâmico, iremos encontrar a constante de tempo (τ), aplicando uma entrada degrau e verificando o tempo necessário para que o valor de saída do instrumento alcance 63,2% do valor final. Os dados foram fornecidos pelo professor.

Utilizaremos as seguintes fórmulas:

$$q_0 = K \cdot q_{is} \cdot \left(1 - e^{\frac{-t}{\tau}}\right)$$

Em que,

q_0 é a resposta do sistema;
K é a sensibilidade estática;
q_{is} é o sinal de entrada (degrau);
t é o tempo;
τ é a constante de tempo do sistema.

$$\frac{q_o - K \cdot q_{is}}{K \cdot q_{is}} = -e^{\frac{-t}{\tau}}$$

$$1 - \frac{q_0}{K \cdot q_{is}} = e^{\frac{-t}{\tau}}$$

$$Z \overset{\Delta}{=} \log_e\left(1 - \frac{q_0}{K \cdot q_{is}}\right)$$

$$Z = \frac{-t}{\tau}$$

$$\frac{dZ}{dt} = \frac{-1}{\tau}$$

Código python disponível em [instrumentacao_parte1_1ordem.py]

Importação das bibliotecas:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importação dos dados:

```
path = "/content/dados1ordem.xlsx"
data_1ordem = pd.read_excel(path, sheet_name="Plan1")
```

Construção Tabela:

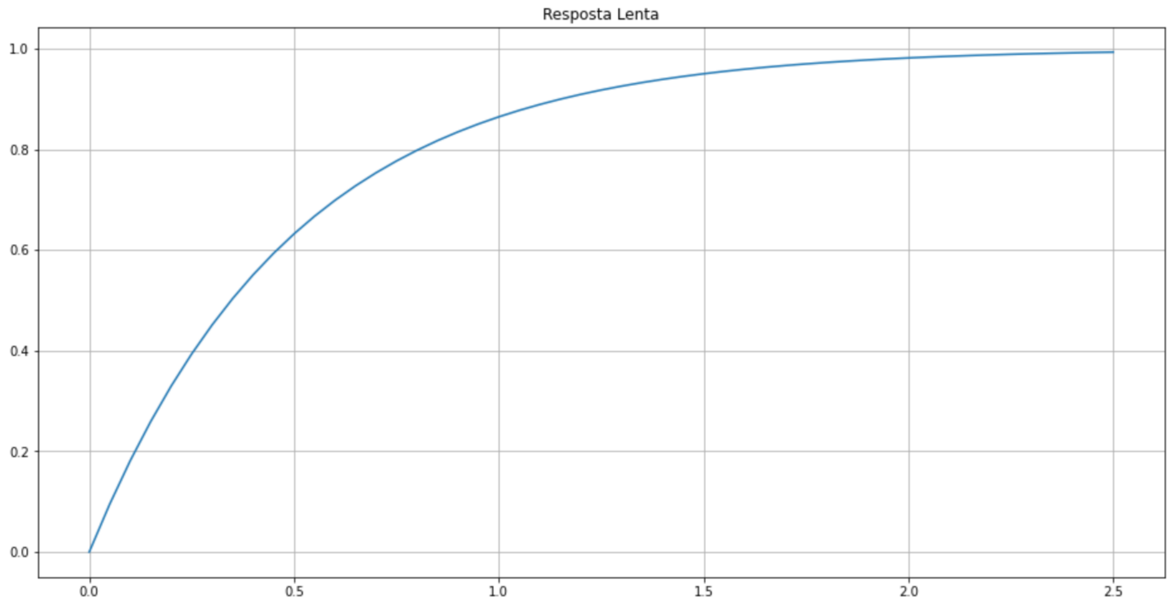
```
data_1ordem.columns = ["Tempo", "Lento", "Rápido"]
data_1ordem.head()
```

	Tempo	Lento	Rapido
0	0.00	0.000000	0.000000
1	0.05	0.095167	0.153549
2	0.10	0.181277	0.283521
3	0.15	0.259192	0.393536
4	0.20	0.329692	0.486658

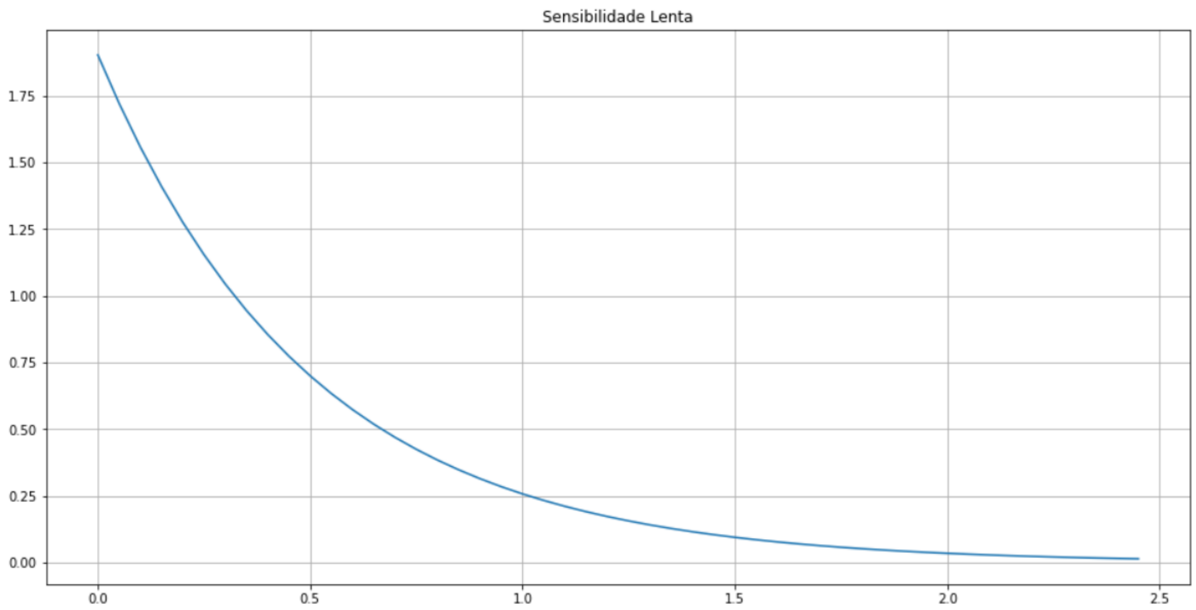
Sistema Lento

Gráficos obtidos:

```
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"],data_1ordem["Lento"])
plt.title("Resposta Lenta")
plt.grid()
plt.show()
```

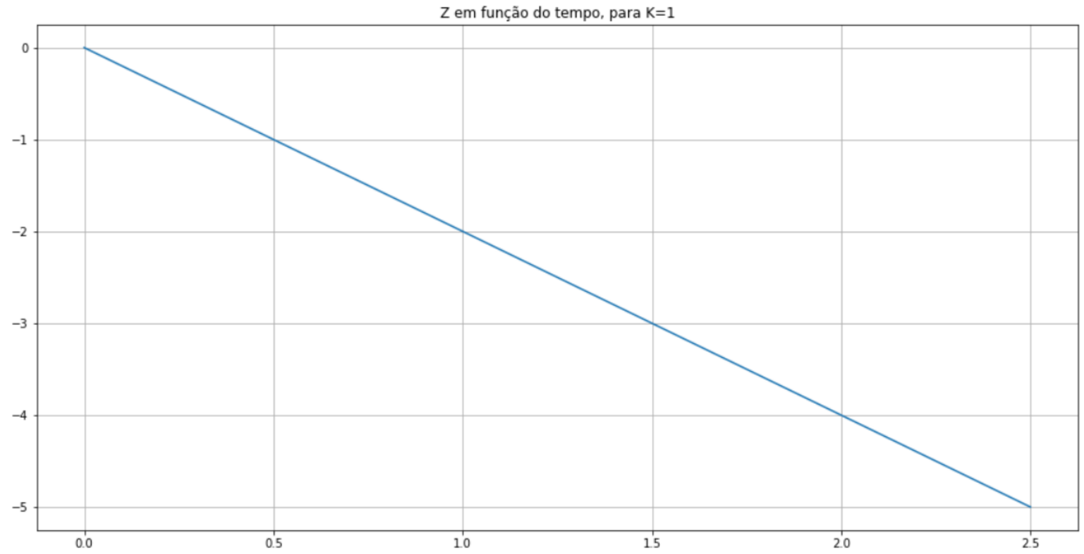



```
diff_lento = np.diff(data_1ordem["Lento"])/np.diff(data_1ordem["Tempo"])
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"][:50],diff_lento)
plt.title("Sensibilidade Lenta")
plt.grid()
plt.show()
```



Z em função do tempo s para K = 1:

```
K = 1
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"],np.log(1-(K*(data_1ordem["Lento"]))))
plt.title("Z em função do tempo, para K={}".format(K))
plt.grid()
plt.show()
```



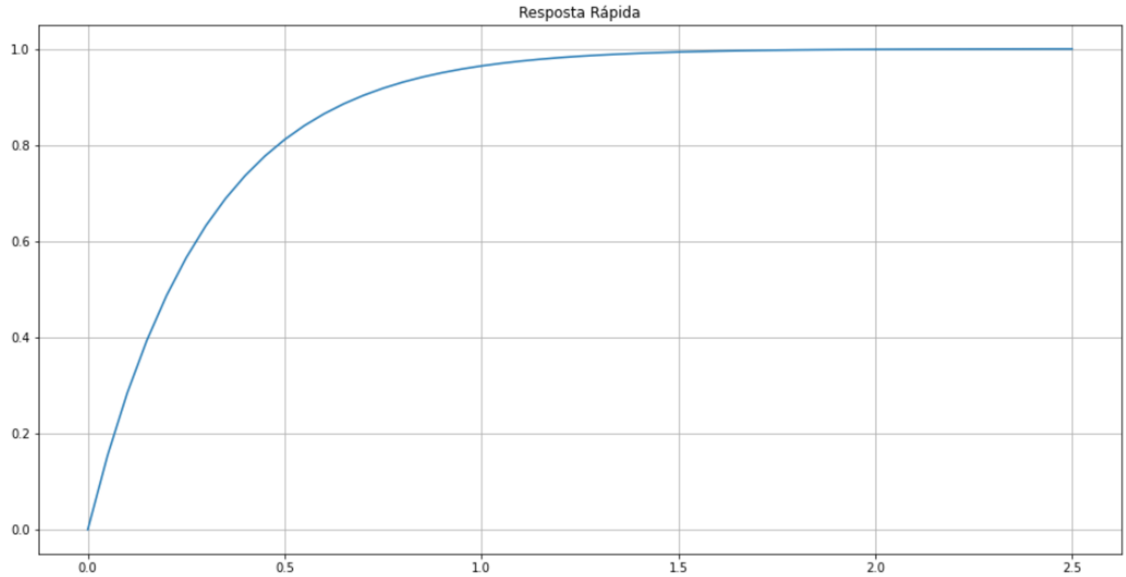

```
round(((-1/(np.diff(np.log(1-(K*(data_1ordem["Lento"]))))/np.diff(data_1orde
m["Tempo"])))).mean(),2)
print(Tal_lento)

Tal_lento = 0.5
```

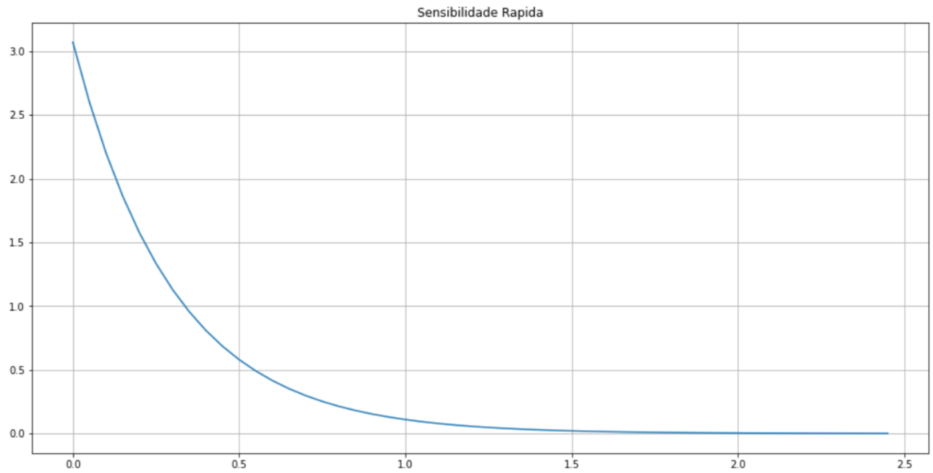
Sistema Rápido

Gráficos obtidos:

```
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"],data_1ordem["Rapido"])
plt.title("Resposta Rápida")
plt.grid()
plt.show()
```

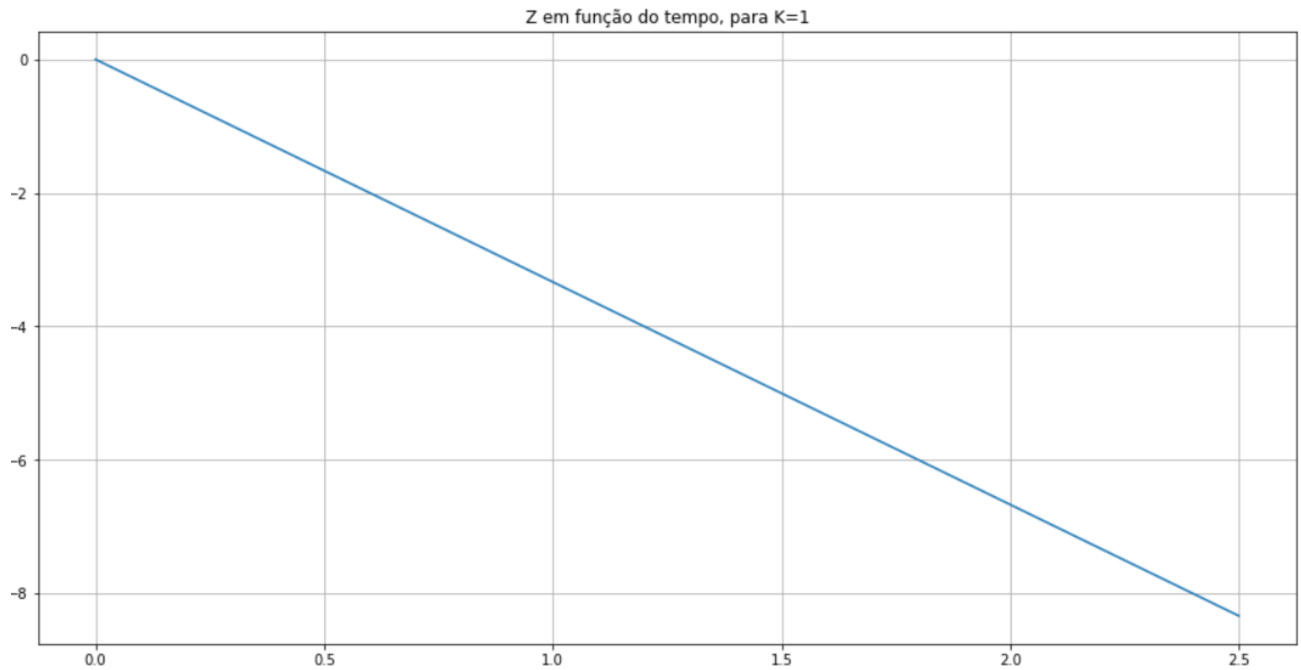



```
diff_lento = np.diff(data_1ordem["Rapido"])/np.diff(data_1ordem["Tempo"])
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"][:50],diff_lento)
plt.title("Sensibilidade Rapida")
plt.grid()
plt.show()
```



Z em função do tempo s para K = 1:

```
K = 1
plt.figure(figsize=(16,8))
plt.plot(data_1ordem["Tempo"],np.log(1-(K*(data_1ordem["Rapido"]))))
plt.title("Z em função do tempo, para K={}".format(K))
plt.grid()
plt.show()
```




```
Tal_rapido =
round(((-1/(np.diff(np.log(1-(K*(data_1ordem["Rapido"]))))/np.diff(data_1ord
em["Tempo"])))).mean(),2)
print(Tal_rapido)

Tal_rápido = 0.3
```

Instrumentos de Segunda Ordem

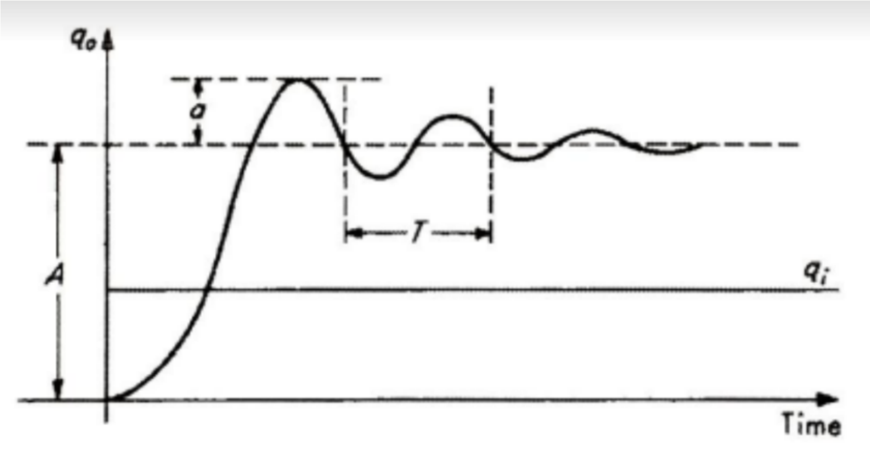
Criar uma rotina que permita que os parâmetros quociente de amortecimento (ζ) e frequência natural não amortecida (ωn) sejam estimados, com base nos dados e gráficos fornecidos pelo professor.

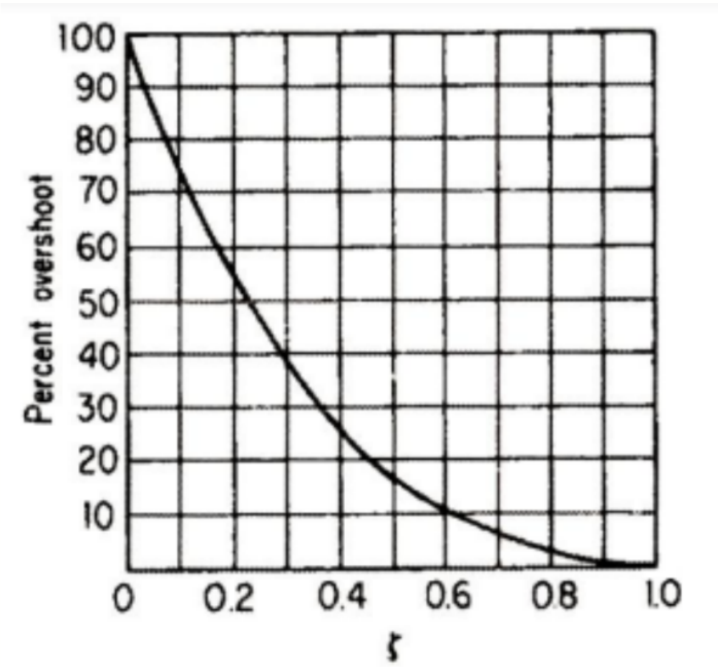
Fórmulas:

$$\zeta = \sqrt{\frac{1}{\left[\frac{\pi}{\log_e \frac{a}{A}}\right]^2 + 1}}$$

$$\omega n = \frac{2\pi}{T \cdot \sqrt{1 - \zeta^2}}$$

Gráficos:





Código python disponível em [instrumentacao_parte2_2ordem.py]()

Carregamento de dados:

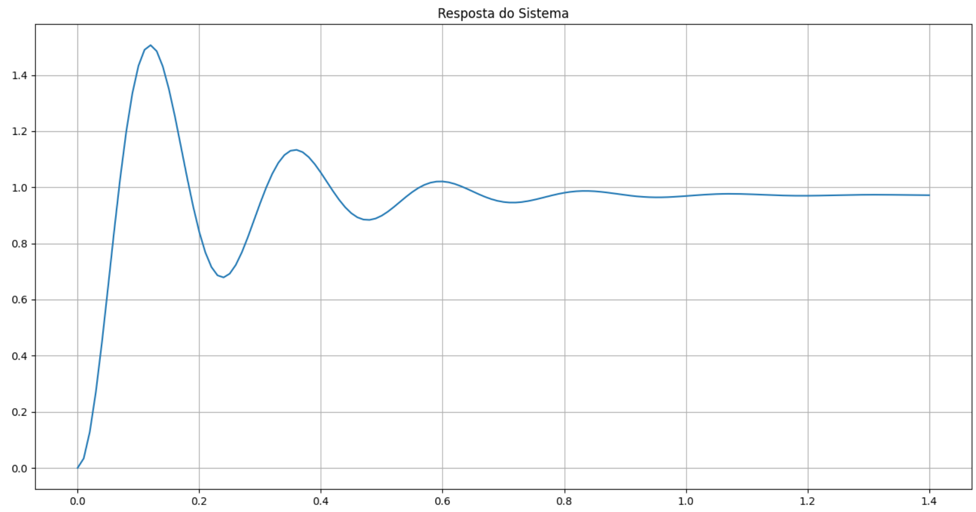
```
dados = pd.read_excel("dados2aordem.xlsx")
```

Transformando dados em lista:

```
tempo = dados["tempo"].tolist()
qo = dados["qo"].tolist()
```

Gráfico do sistema:

```
plt.figure(figsize=(16,8))
plt.plot(tempo,qo)
plt.title("Resposta do Sistema")
plt.grid()
plt.show()
```



Encontrando **A**:

```
A = statistics.mode(round(dados["qo"], 2))

# o valor de "A" foi encontrado obtendo-se a moda dos valores da lista qo. Ou seja, pegamos o valor que mais se repete. Arredondamos os valores com 2 casas decimais para isso.
```

Encontrando o valor máximo da lista qo:

```
valor_max = max(qo)
```

Encontrando **a:**

```
a = valor_max - A

# o valor de "a" foi encontrado pegando o maior valor da lista qo e subtraindo o valor de "A".
```

Uma função foi criada para encontrar **T**:

```
def encontrar_T(qo, tempo, A):

    i = qo.index(valor_max)  #iniciamos i na posição do maior valor
    marcador_de_tempo = []
    while qo[i] >= A:
        i = i + 1
    marcador_de_tempo.append(tempo[i])  # marcamos o tempo de início
    while qo[i] <= A:
        i = i + 1
    while qo[i] >= A:
        i = i + 1
    marcador_de_tempo.append(tempo[i])  # marcamos o tempo final
    return marcador_de_tempo[1] - marcador_de_tempo[0]
```

Encontrando **T**:

```
T = encontrar_T(qo, tempo, A)
```

Encontrando **ζ:**

```
ζ = math.sqrt(1 / ((math.pi/np.log(a/A))**2 + 1))
```

Encontrando **wn:**

```
ωn = 2*math.pi / (T*math.sqrt(1 - ζ**2))
```

Resultados encontrados:

```
A = 0.97
Valor máximo = 1.50651956243529
a = 0.5365195624352901
T = 0.24
ζ = 0.18523862371958566
ωn = 26.640999373920554
```

Conclusão:

Há coerência nos resultados estimados obtidos para a constante de tempo no sistema de primeira ordem e o quociente de amortecimento (ζ) e Frequência natural não amortecida (ωn). Portanto, o modelo construído em Python é funcional.