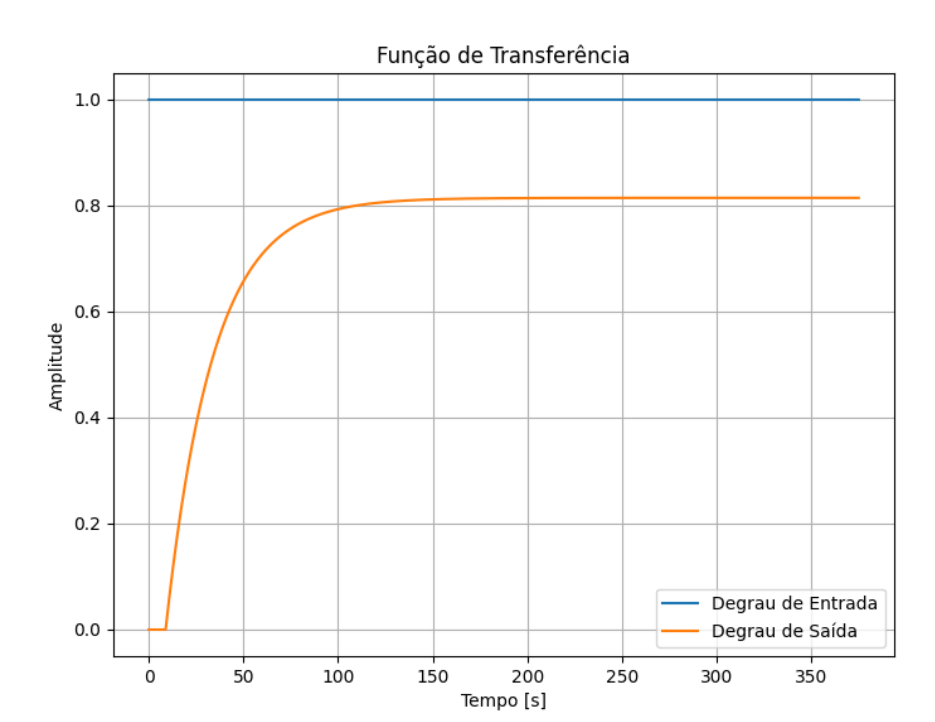
TRABALHO DE SISTEMAS EMBARCADOS (C213) CONTROLE PID

MATHEUS CAMARA CARVALHO

IGOR DA SILVA VILLAMARIM

GABRIEL DE SOUZA GEMELLE LEAL

FUNÇÃO DE TRANSFERÊNCIA



Valor do degrau: 1
Valor Máximo da Saída: 0,814
Tempo de Atraso: 8,85
Valor da constante de tempo: 25,049

MÉTODOS UTILIZADOS

Tabela 1: Parâmetros do PID para a Sintonia pela ZN Curva de Reação.

Controlador	K_p	T_i	T_d
PID	$\frac{1.2\tau}{k \cdot \theta}$	2θ	$\frac{\theta}{2}$

Tabela 5: Parâmetros do PID para a Sintonia Método de Cohen e Coon.

Controlador	K_p	T_i	T_d
PID	$\frac{\tau}{k \cdot \theta}\left(\frac{16\tau + 3\theta}{12\tau}\right)$	$\theta \cdot \frac{32 + {}^{6\theta}/_{\tau}}{13 + {}^{8\theta}/_{\tau}}$	$\frac{4\theta}{11 + {}^{2\theta}/_{\tau}}$

FUNÇÃO DE TRANSFERÊNCIA ZIEGLE-NICHOLS E COHEN E COON

```
Função de transferência do sistema (ZN):

18.4485s^2 + 4.1692s + 0.2355
-----------------------------
       25.0500s^2 + s

Kp (ZN): 4.169156316850236
Ti (ZN): 17.700000000000003
Td (ZN): 4.425000000000001
```

```
Função de transferência do sistema (CC):

14.9360s^2 + 4.9393s + 0.2589
-----------------------------
       25.0500s^2 + s

Kp (CC): 4.9392574628685 64
Ti (CC): 19.079568671963678
Td (CC): 3.0239386189258317
```

MÉTODOS DE IDENTIFICAÇÃO DE PLANTA E SELEÇÃO ATRAVÉS DE MENOR ERRO

```
Codeium: Refactor | Explain | Generate Docstring | X
def Smith(Step, Tempo, Saída):
    if not isinstance(Tempo, list) or not isinstance(Saída, list) or len(Tempo) < 1 or len(Saída) < 1:
        raise TypeError('Erro de Tipo: Os argumentos devem ser listas não vazias.')

    if not isinstance(Step, (int, float)):
        raise TypeError('Erro de Tipo: O argumento \'Step\' deve ser uma constante.')

    Saída = [x - Saída[0] for x in Saída]
    valorFinal = Saída[-1]
    k = valorFinal / Step

    t1 = 0
    t2 = 0
    for i in range(len(Saída)):
        if Saída[i] >= 0.283 * valorFinal and t1 == 0:
            t1 = Tempo[i]

        if Saída[i] >= 0.6321 * valorFinal:
            t2 = Tempo[i]
            break

    tau = 1.5 * (t2 - t1)
    Theta = t2 - tau
    identificacaoSmith = [k, tau, Theta]

    return identificacaoSmith
```

```
Codeium: Refactor | Explain | Generate Docstring | X
def Sundaresan(Step, Tempo, Saída):
    if not isinstance(Tempo, list) or not isinstance(Saída, list) or len(Tempo) < 1 or len(Saída) < 1:
        raise TypeError('Erro de Tipo: Os argumentos devem ser listas não vazias.')

    if not isinstance(Step, (int, float)):
        raise TypeError('Erro de Tipo: O argumento \'Step\' deve ser uma constante.')

    Saída = [x - Saída[0] for x in Saída]
    valorFinal = Saída[-1]
    k = valorFinal / Step

    t1 = 0
    t2 = 0
    for i in range(len(Saída)):
        if Saída[i] >= 0.353 * valorFinal and t1 == 0:
            t1 = Tempo[i]

        if Saída[i] >= 0.853 * valorFinal:
            t2 = Tempo[i]
            break

    tau = 2 / 3 * (t2 - t1)
    Theta = 1.3 * t1 - 0.29 * t2
    identificacaoSundaresan = [k, tau, Theta]

    return identificacaoSundaresan
```

CALCULANDO VALORES DE K, THETA E TAU

```
# Valores da função Smith
smith = Smith(np.mean(degrau), tempo, saida)
sys_smith = ctrl.TransferFunction([smith[0]], [smith[1], 1])
t, y = ctrl.step_response(sys_smith * AmplitudeDegrau, tempo)
saida_smith = y +  valorInicial
erro_smith = np.sqrt(np.mean((saida - saida_smith)**2))

# Valores da função Sundaresan
sundaresan = Sundaresan(np.mean(degrau), tempo, saida)
sys_sundaresan = ctrl.TransferFunction([sundaresan[0]], [sundaresan[1], 1])
t, y = ctrl.step_response(sys_sundaresan * AmplitudeDegrau, tempo)
saida_sundaresan = y +  valorInicial
erro_sundaresan = np.sqrt(np.mean((saida - saida_sundaresan)**2))
```

```
# Escolhendo através do erro
if erro_smith < erro_sundaresan:
    k = smith[0]
    tau = smith[1]
    theta = smith[2]
    print("O método escolhido foi smith!\n")

else :
    k = sundaresan[0]
    tau = sundaresan[1]
    theta = sundaresan[2]
    print("O método escolhido foi Sundaresan!\n")

print("Valor de K:", k)
print("Valor do atraso de transporte:", theta)
print("Valor da constante de tempo:", tau)
```

Para o nosso projeto o método escolhido foi o Smith, em razão de o valor do erro ter sido menor em relação ao erro do sundaresan. Utilizamos esse método para aumentar a eficiência tendo o erro como decisão.

CALCULANDO VALORES PARA ZIEGLER NICHOLS

```
# Ziegler-Nichols em malha aberta
Kp_zn = (1.2 * tau) / (k * theta)
Ti_zn = 2 * theta
Td_zn = theta / 2

# Criando o controlador PID com os parâmetros de Ziegler-Nichols
num_pid_zn = [Kp_zn * Td_zn, Kp_zn, Kp_zn / Ti_zn]
den_pid_zn = [1, 0]
PID_zn = ctrl.TransferFunction(num_pid_zn, den_pid_zn)

# Criando o sistema em série com os parâmetros de Ziegler-Nichols
sys_atraso = ctrl.tf([1], [tau, 1])
Cs_zn = ctrl.series(PID_zn, sys_atraso)

# Gerando a resposta ao degrau do sistema em malha fechada com Ziegler-Nichols
tempo_resposta_zn, resposta_zn = ctrl.step_response(ctrl.feedback(Cs_zn, 1))

# Calculando informações adicionais usando a função step_info
info_zn = ctrl.step_info(ctrl.feedback(Cs_zn, 1))
tempo_subida_zn = info_zn['RiseTime']
tempo_acomodacao_zn = info_zn['SettlingTime']
overshoot_zn = info_zn['Overshoot']

# Resultados de Ziegler-Nichols
print("\nResultados do Ziegler-Nichols:")
print("Tempo de Subida (ZN):", tempo_subida_zn)
print("Tempo de Acomodação (ZN):", tempo_acomodacao_zn)
print("Overshoot (ZN):", overshoot_zn)
```

CALCULANDO VALORES PARA COHEN E COON

```
# Cohen e Coon em malha aberta
Kp_cc = (tau / (k * theta)) * ((16 * tau + 3 * theta) / (12 * tau))
Ti_cc = theta * (32 + (6 * theta) / tau) / (13 + (8 * theta) / tau)
Td_cc = (4 * theta) / (11 + (2 * theta / tau))

# Criando o controlador PID com os parâmetros de Cohen e Coon
num_pid_cc = [Kp_cc * Td_cc, Kp_cc, Kp_cc / Ti_cc]
den_pid_cc = [1, 0]
PID_cc = ctrl.TransferFunction(num_pid_cc, den_pid_cc)

# Criando o sistema em série com os parâmetros de Cohen e Coon
Cs_cc = ctrl.series(PID_cc, sys_atraso)

# Gerando a resposta ao degrau do sistema em malha fechada com Cohen e Coon
tempo_resposta_cc, resposta_cc = ctrl.step_response(ctrl.feedback(Cs_cc, 1))

# Calculando informações adicionais usando a função step_info
info_cc = ctrl.step_info(ctrl.feedback(Cs_cc, 1))
tempo_subida_cc = info_cc['RiseTime']
tempo_acomodacao_cc = info_cc['SettlingTime']
overshoot_cc = info_cc['Overshoot']

# Resultados de Cohen e Coon
print("\nResultados de Cohen e Coon:")
print("Tempo de Subida (CC):", tempo_subida_cc)
print("Tempo de Acomodacao (CC):", tempo_acomodacao_cc)
print("Overshoot (CC):", overshoot_cc)
```

PARÂMETROS DO PID PARA OS MÉTODOS DE SINTONIA ESPECIFICADOS

Tabela 7: Parâmetros do PID para os métodos de Sintonia especificados.

Técnica	Método	K_p	T_i	T_d
1	Ziegler Nichols Malha Aberta	4.169	17.700	4.425
2	IMC	-	-	-
3	CHR sem Sobrevalor	-	-	-
4	CHR com Sobrevalor	-	-	-
5	Cohen e Coon	4.939	19.079	3.023
6	ITAE	-	-	-

CALCULANDO ERROS

```python
# Função para calcular o erro quadrático médio
Codeium: Refactor | Explain | Generate Docstring | ×
def calcular_erro_quadratico_medio(saida_real, saida_estimada):
    erro = np.sqrt(np.mean((saida_real - saida_estimada) ** 2))
    return erro

# Interpolação da resposta ao degrau do sistema para coincidir com os tempos dos dados reais
resposta_zn_interpolada = np.interp(tempo, tempo_resposta_zn, resposta_zn)
resposta_cc_interpolada = np.interp(tempo, tempo_resposta_cc, resposta_cc)
resposta_usuario_interpolada = np.interp(tempo, tempo_resposta_usuario, resposta_usuario)

# Calculando o erro para Ziegler-Nichols com os valores interpolados (malha fechada)
erro_zn = calcular_erro_quadratico_medio(saida, resposta_zn_interpolada)

# Calculando o erro para Cohen e Coon com os valores interpolados (malha fechada)
erro_cc = calcular_erro_quadratico_medio(saida, resposta_cc_interpolada)

# Calculando o erro para os parâmetros inseridos pelo usuário com os valores interpolados (malha fechada)
erro_usuario = calcular_erro_quadratico_medio(saida, resposta_usuario_interpolada)

# Calculando o erro para Ziegler-Nichols com os valores interpolados (malha aberta)
erro_zn_malha_aberta = calcular_erro_quadratico_medio(degrau, resposta_zn_interpolada)

# Calculando o erro para Cohen e Coon com os valores interpolados (malha aberta)
erro_cc_malha_aberta = calcular_erro_quadratico_medio(degrau, resposta_cc_interpolada)

# Calculando o erro para os parâmetros inseridos pelo usuário com os valores interpolados (malha aberta)
erro_usuario_malha_aberta = calcular_erro_quadratico_medio(degrau, resposta_usuario_interpolada)
```

Erro Quadrático Médio (RMSE):
-O RMSE é uma medida comum de quão bem um modelo de previsão ou estimativa se ajusta aos dados observados.
-Ele é calculado como a raiz quadrada da média dos quadrados dos erros entre os valores previstos pelo modelo e os valores observados.
-É frequentemente usado em problemas de regressão ou previsão, onde se deseja avaliar o quão bem um modelo está prevendo os valores reais.

A Interpolação é realizada para que a saída coincida com os tempos dos dados reais, para que não haja valores discrepantes

COMPARAÇÃO DOS ERROS MALHA ABERTA X MALHA FECHADA

Malha Fechada:
Erro Ziegler-Nichols: 0.280879499042471
Erro Cohen e Coon: 0.28577878742150886
Erro com parâmetros do usuário: 0.2776484652067941 5

Malha Aberta:
Erro Ziegler-Nichols: 0.056593101570233874
Erro Cohen e Coon: 0.06670258056145524
Erro com parâmetros do usuário: 0.16125387064678529

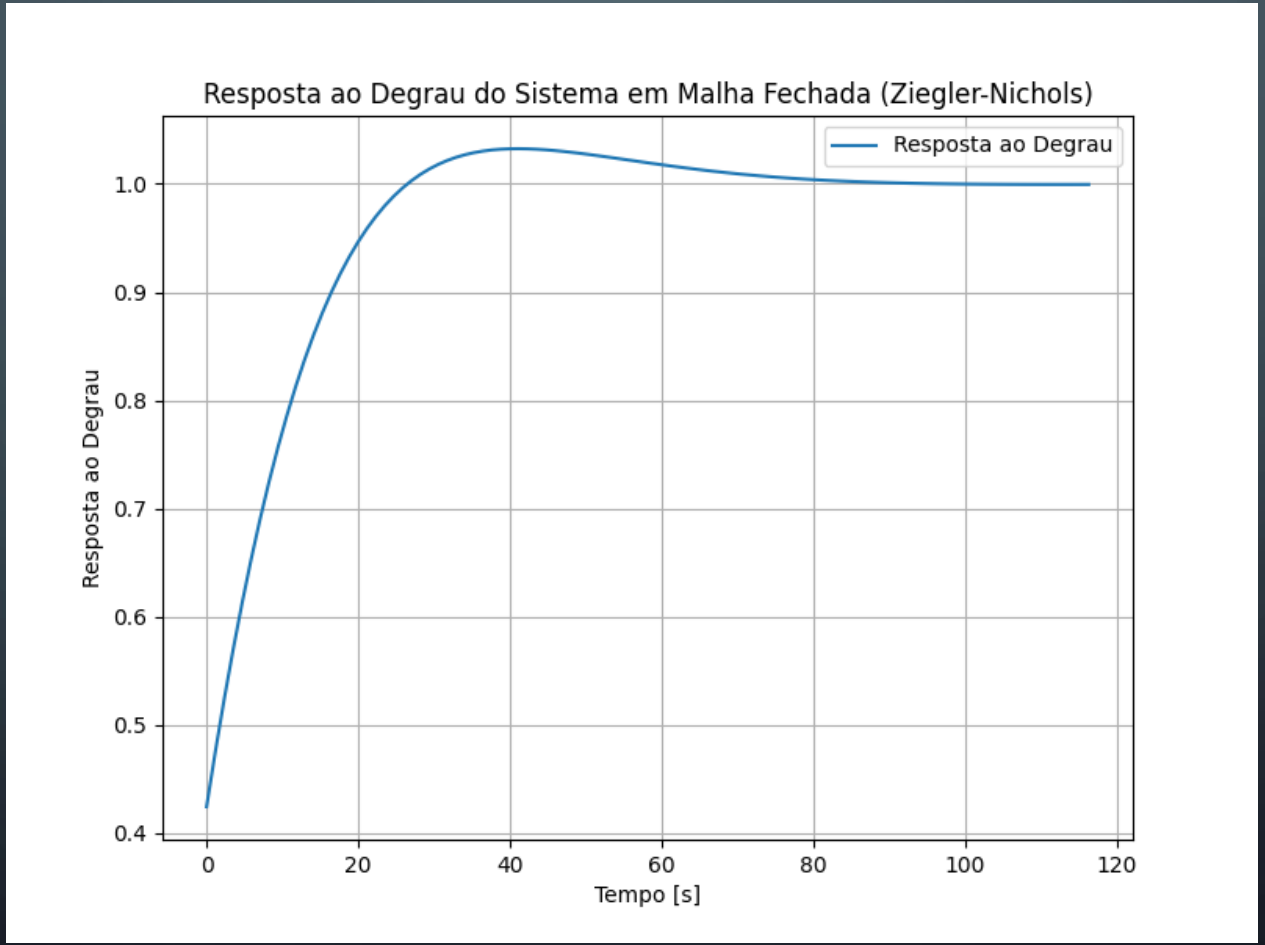
Obs: Os erros em malha fechada foram maiores do que os erros em malha aberta

COMPARAÇÃO DAS CONCLUSÕES

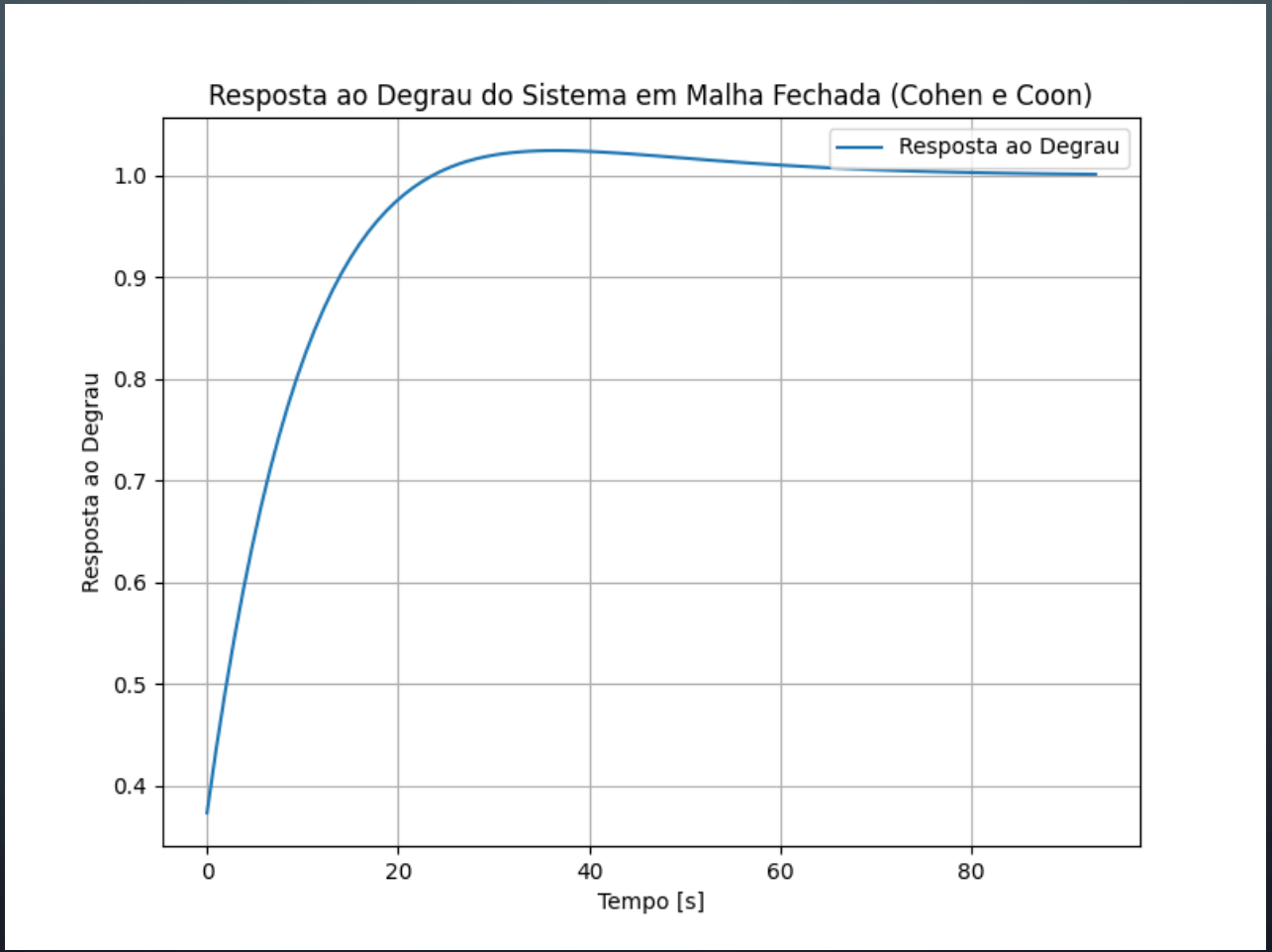
```
Resultados do Ziegler-Nichols:
Tempo de Subida (ZN): 17.614802807995112
Tempo de Acomodação (ZN): 58.716009359983715
Overshoot (ZN): 3.25963818749635

Resultados de Cohen e Coon:
Tempo de Subida (CC): 14.092853484911865
Tempo de Acomodacao (CC): 46.03665471737876
Overshoot (CC): 2.402143852236338
```

Cohen e Coon apresentou valores menores em todos parâmetros, sendo tempo de subida (14,09), Tempo de acomodação (46,03) e Overshoot (2,40)

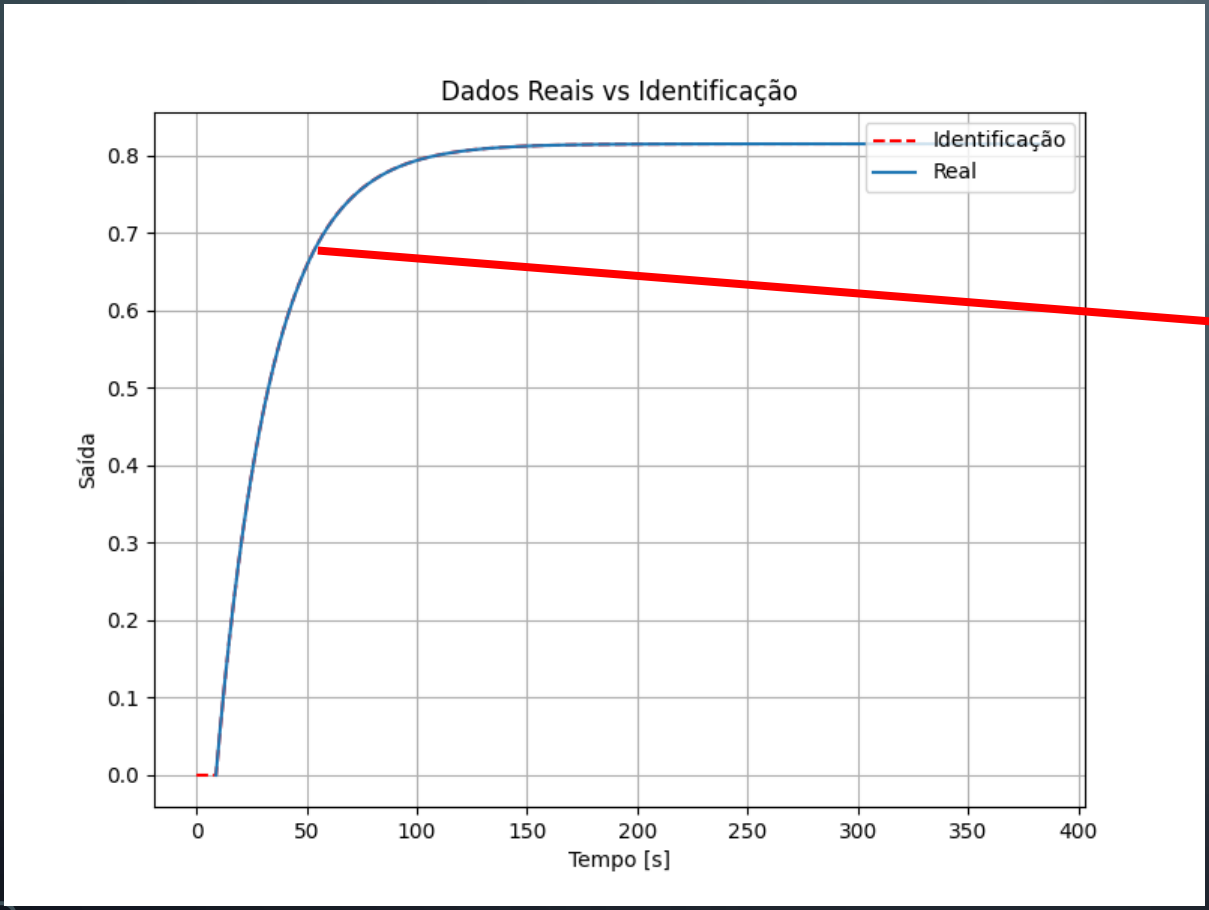
RESPOSTA AO DEGRAU (ZIEGLE-NICHOLS)

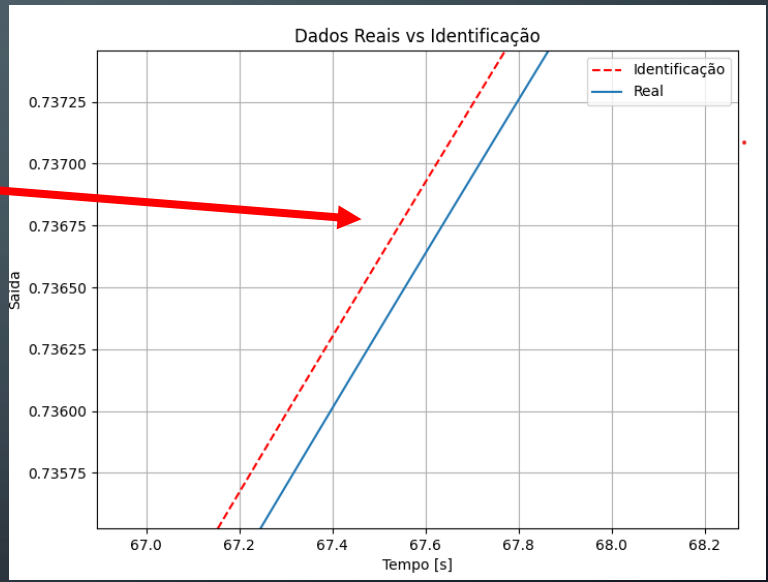


RESPOSTA AO DEGRAU (COHEN E COON)



DADOS REAIS VS IDENTIFICAÇÃO

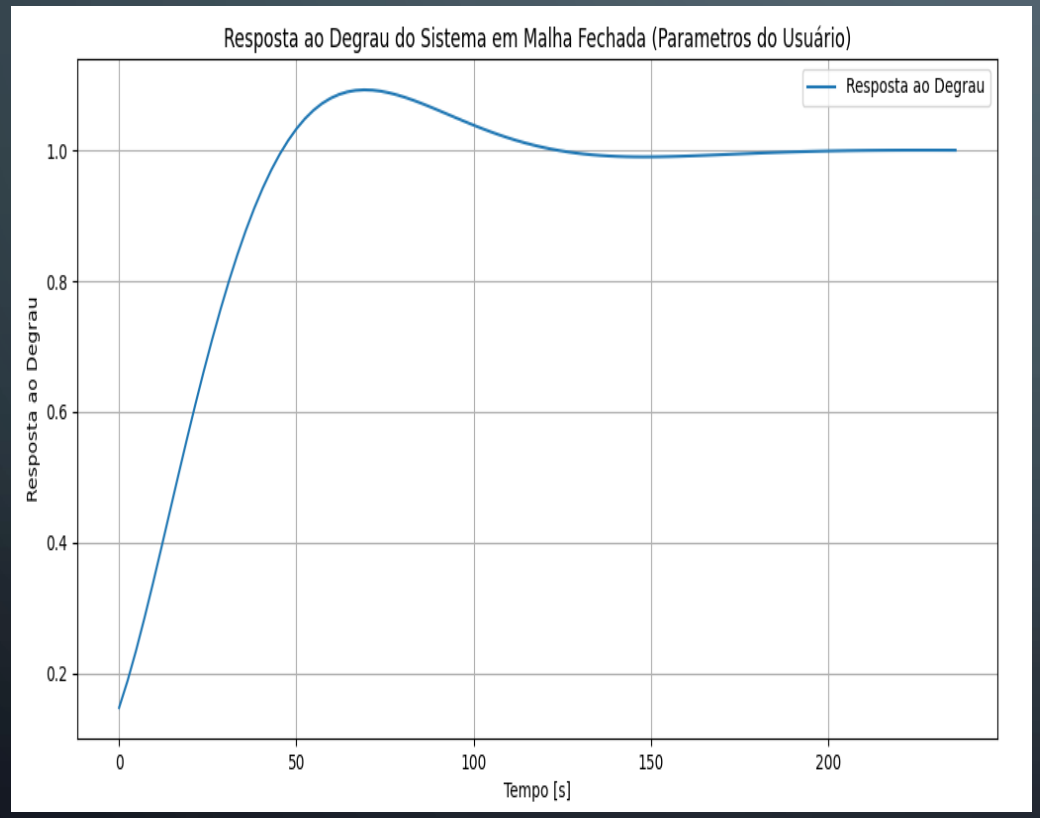


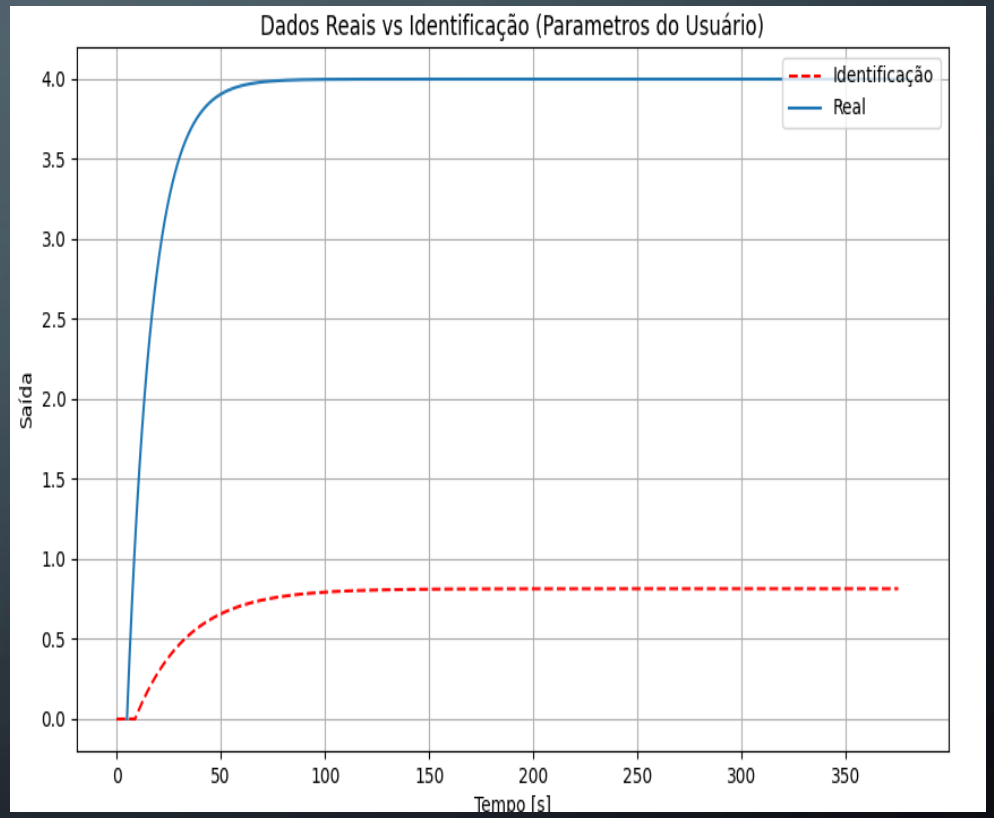


PARÂMETROS ESCOLHIDOS PELO USUÁRIO: ZIEGLER NICHOLS

```
Digite o método desejado (Ziegler-Nichols (zn) ou Cohen e Coon (co)): zn
Digite o valor de K: 4
Digite o valor de tau: 12
Digite o valor de theta: 5
Digite o valor do setpoint: 0.5
```

RESPOSTA AO DEGRAU (PARÂMETROS DO USUÁRIO): ZIEGLER NICHOLS

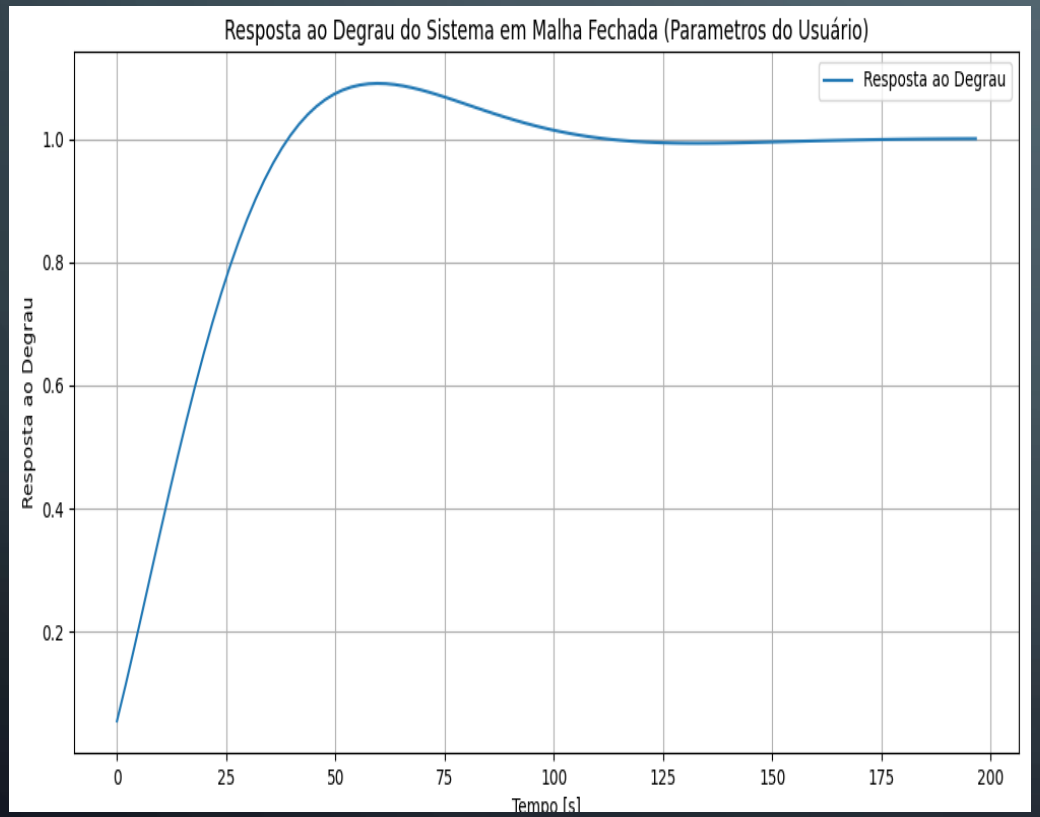


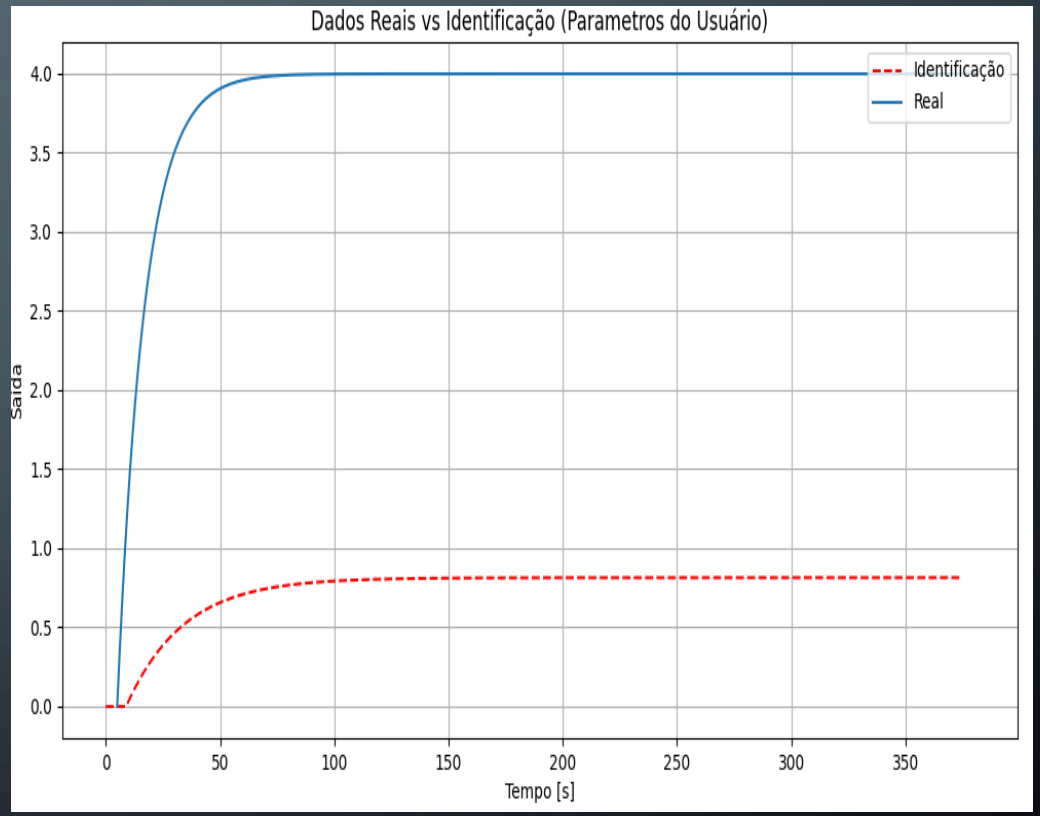


PARÂMETROS ESCOLHIDOS PELO USUÁRIO: COHEN E COON

```
Digite o método desejado (Ziegler-Nichols (zn) ou Cohen e Coon (co)): co
Digite o valor de K: 4
Digite o valor de tau: 12
Digite o valor de theta: 5
Digite o valor do setpoint: 0.5
```

RESPOSTA AO DEGRAU (PARÂMETROS DO USUÁRIO): COHEN E COON





OBRIGADO PELA ATENÇÃO!