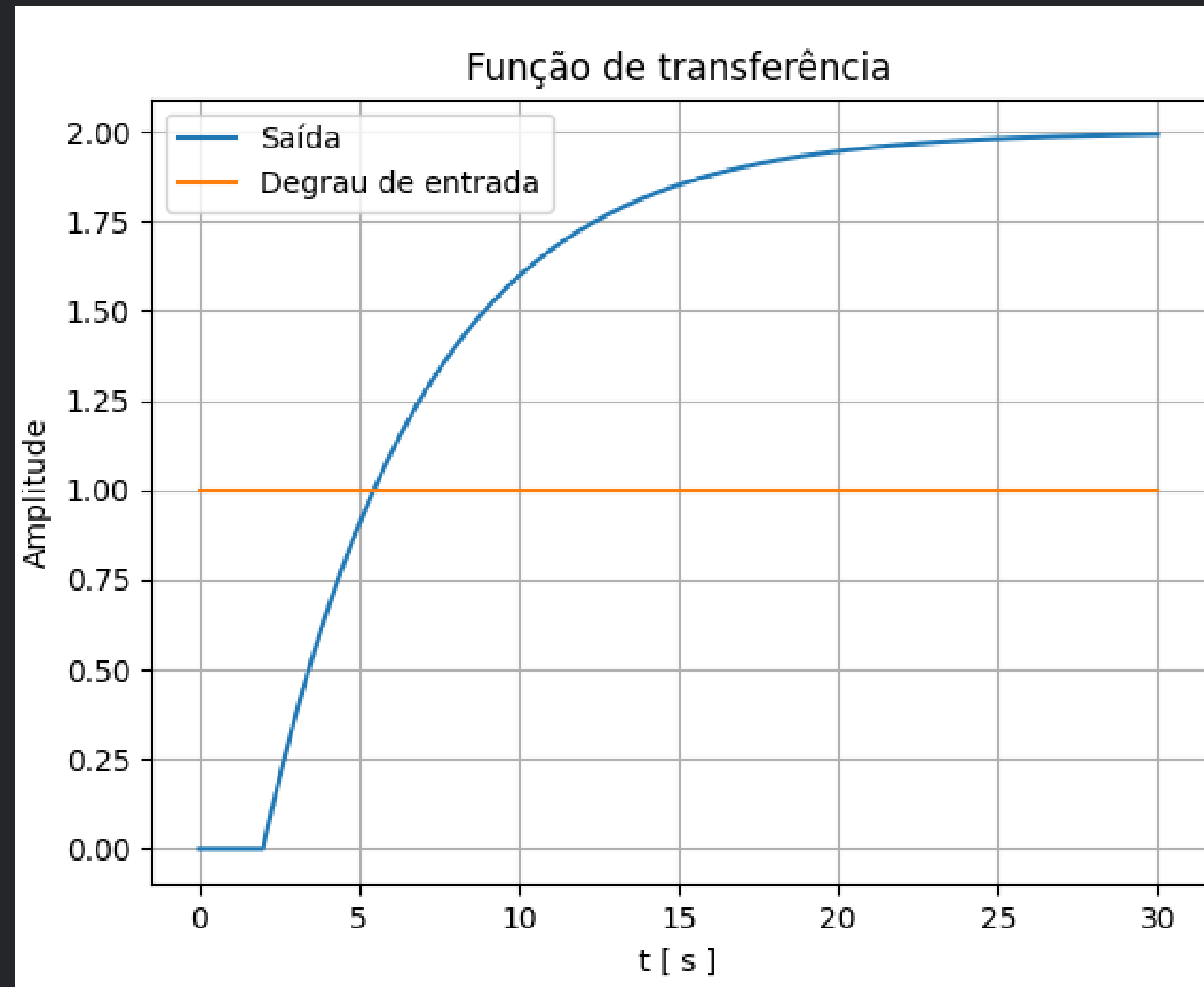


Trabalho 1

Sistemas Embarcados

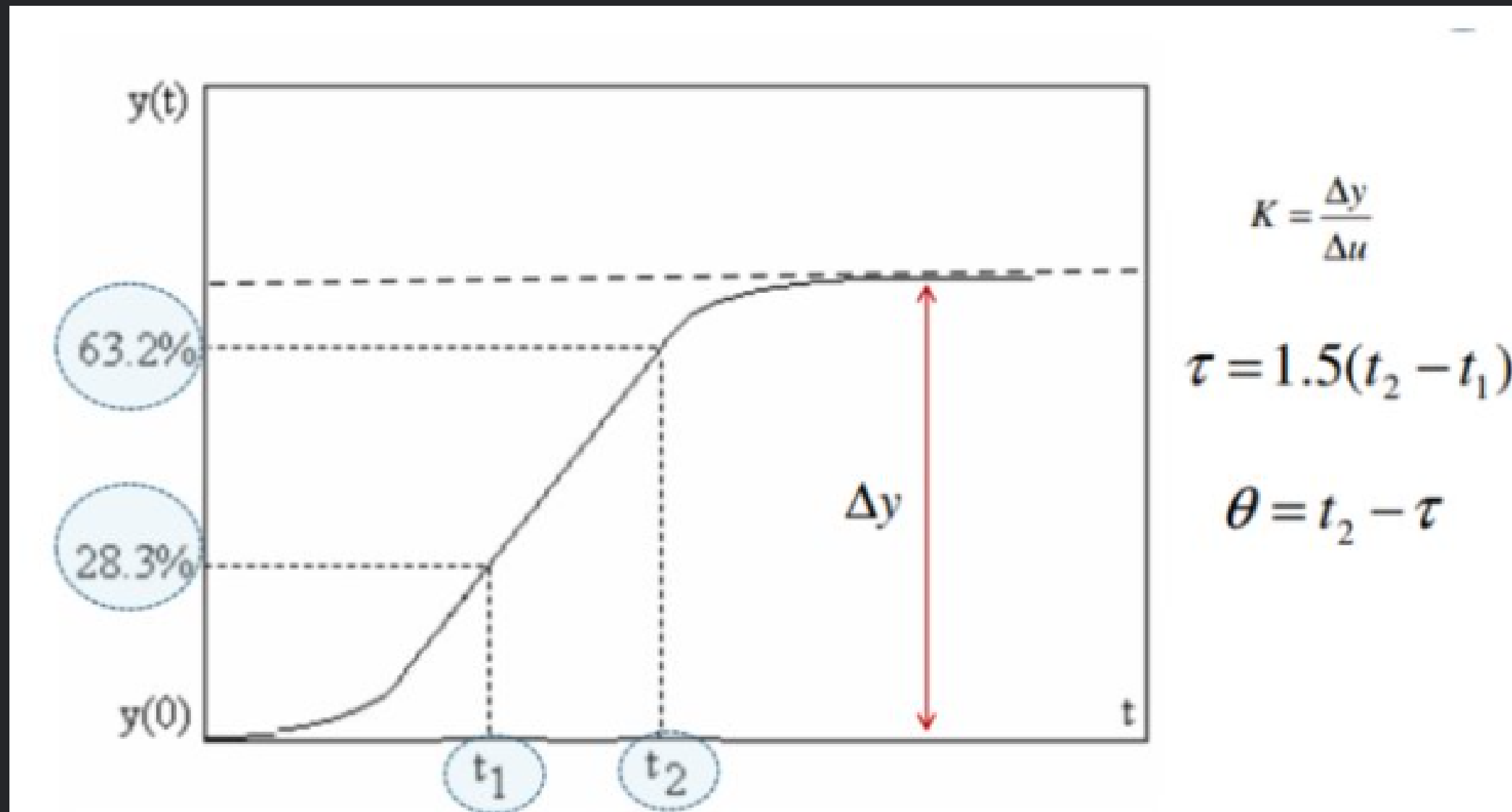
AMANDA SILVA GUIMARÃES
MANUELA GRIPP SILVA
MARIA LUIZA SILVA RAIMUNDO

Função de Transferência - Grupo 1



Valor degrau = 1
Valor saída = 2
Tempo atraso = 2s

Encontrando os valores de k , θ e τ



Através do Método Smith, encontramos os seguintes valores:

$$\mathbf{k=2 \quad \theta=1.975 \quad \tau=4.995}$$

Encontrando os valores de k , Θ e τ

```
# valores obtidos pelo método de smith
y_max = max(saida) # valor máximo do sinal
d_Y = y_max - min(saida) # delta do sinal
d_u = 1 # valor do degrau unitário

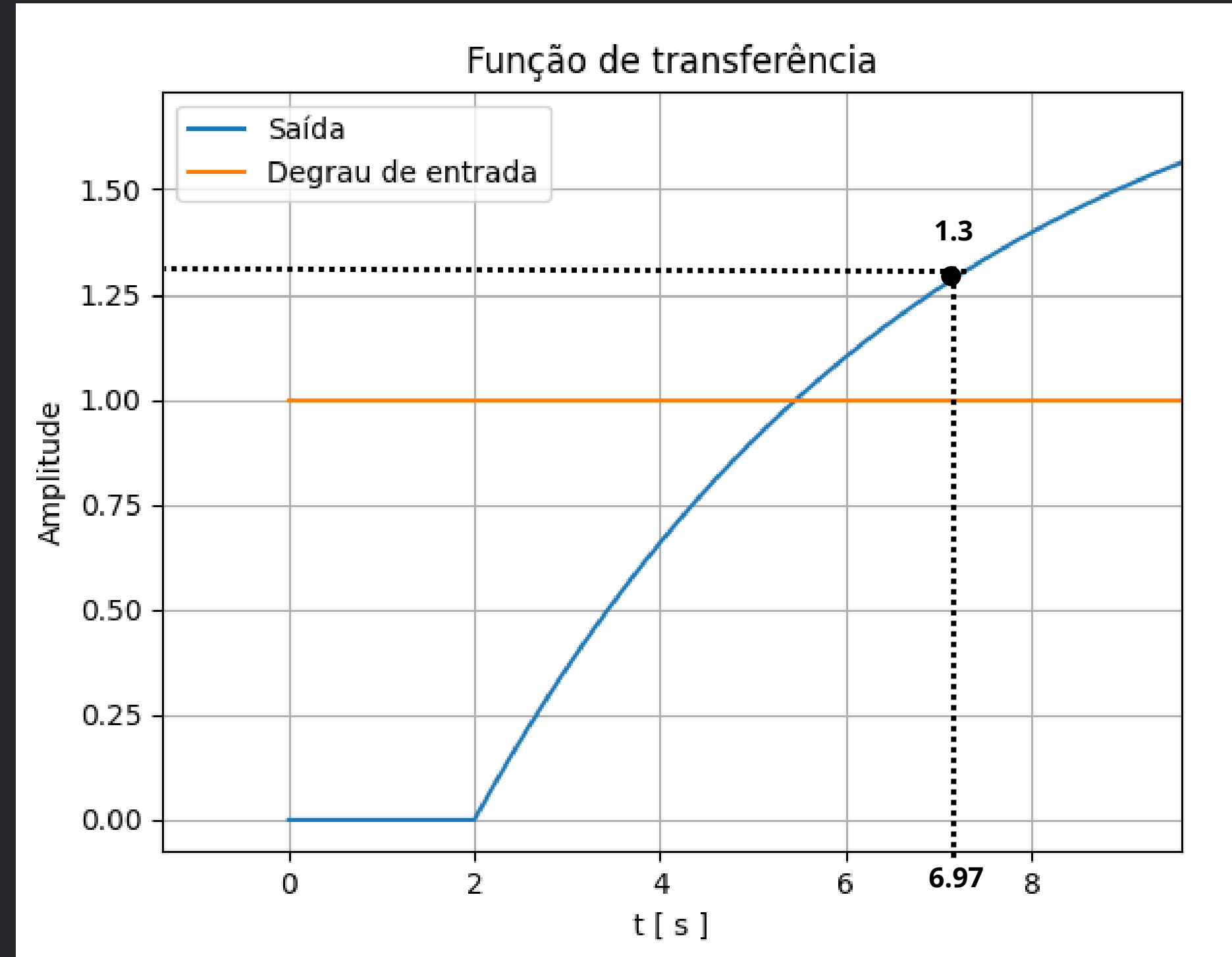
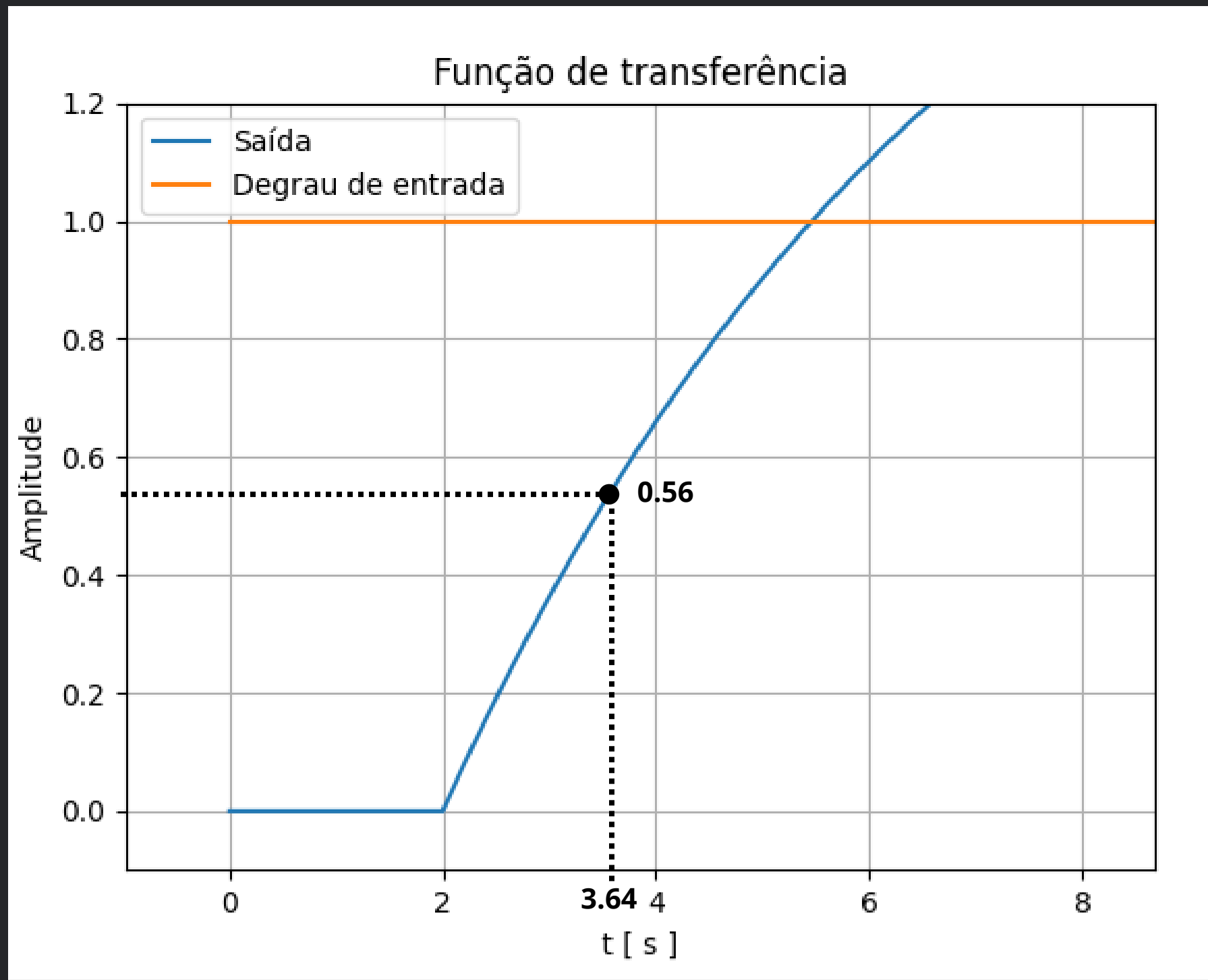
# valor do sinal em 28,3% para determinar t1 pelo gráfico
y_t1 = y_max * 0.283
t1 = 3.64

# valor do sinal em 63,2% para determinar t2 pelo gráfico
y_t2 = y_max * 0.632
t2 = 6.97

# encontrando os valores de k,  $\Theta$  e  $\tau$ 
k = d_Y/d_u # k = 2
tau = 1.5 * (t2 - t1) #  $\tau$  = 4.995
theta = t2 - tau #  $\Theta$  = 1.975

print( k, tau, theta)
```

Encontrando os valores de k , Θ e τ



Plotando a estimativa

```
# importando os dados
mat = loadmat('TransferFunction1.mat')

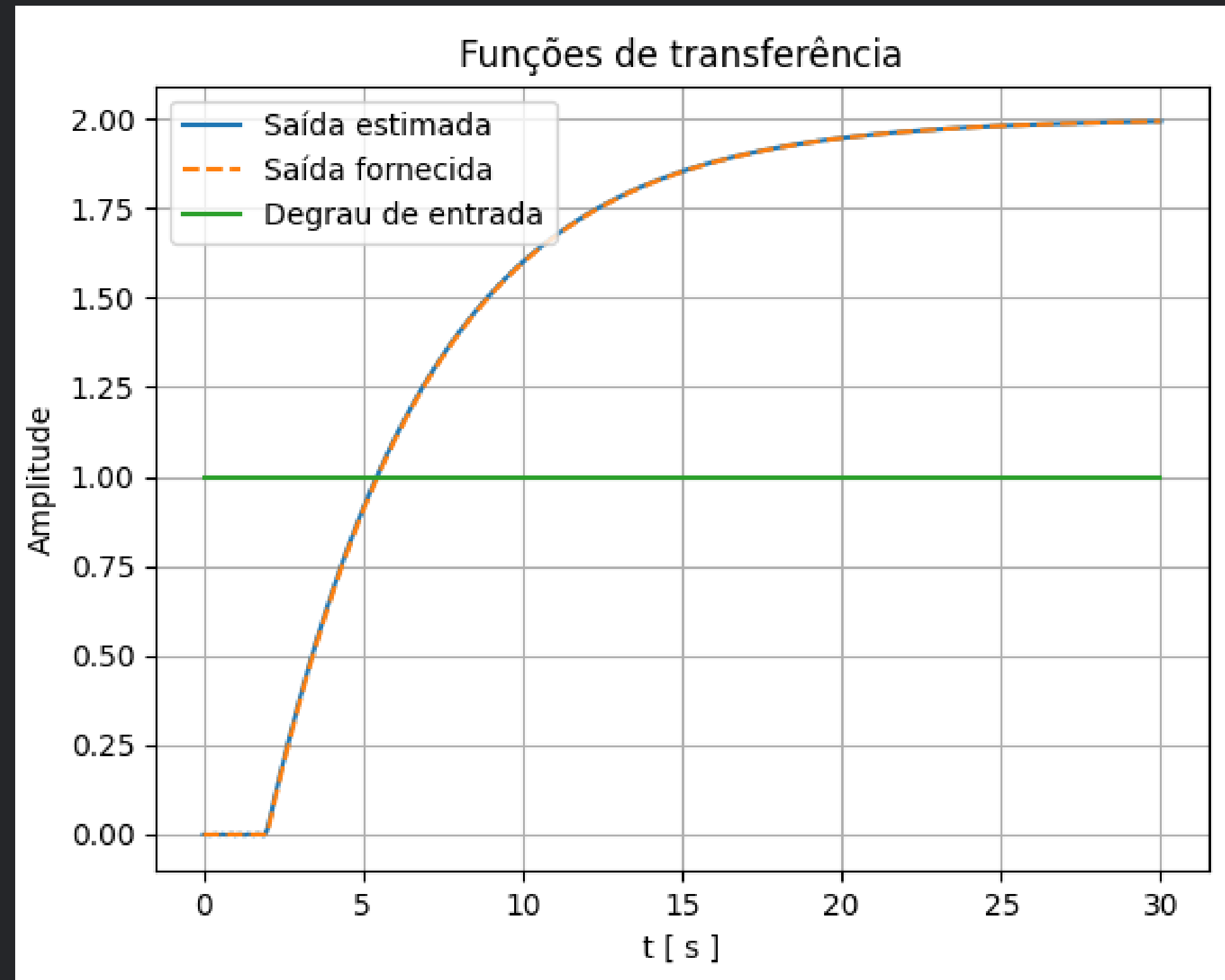
# simulação da função de transferência
degrau = mat.get('degrau')
saida = mat.get('saida')
t1 = mat.get('t')

# definindo as variáveis da função de transferência
k = 2
tau = 4.995
theta = 1.975

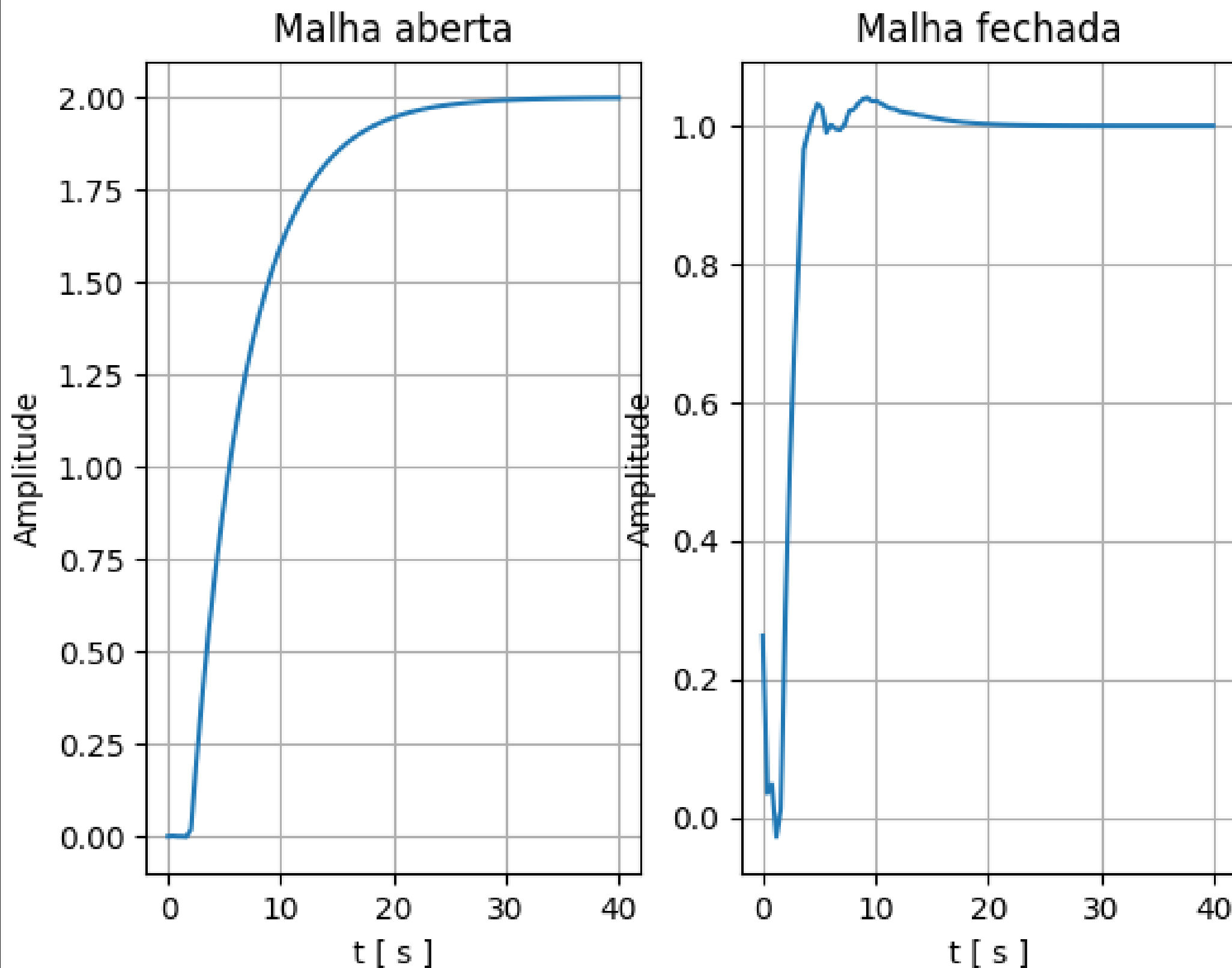
# construindo a função de transferência
num = np.array([k])
den = np.array([tau, 1])
H = cnt.tf(num, den)
n_pade = 20
(num_pade, den_pade) = cnt.pade(theta, n_pade)
H_pade = cnt.tf(num_pade, den_pade)
Hs = cnt.series(H, H_pade)

# simulação da função de transferência estimada
time, y = cnt.step_response(1*Hs, T=t1)
```

Plotando estimada e fornecida



Erro na função



Método Modelo Interno - IMC

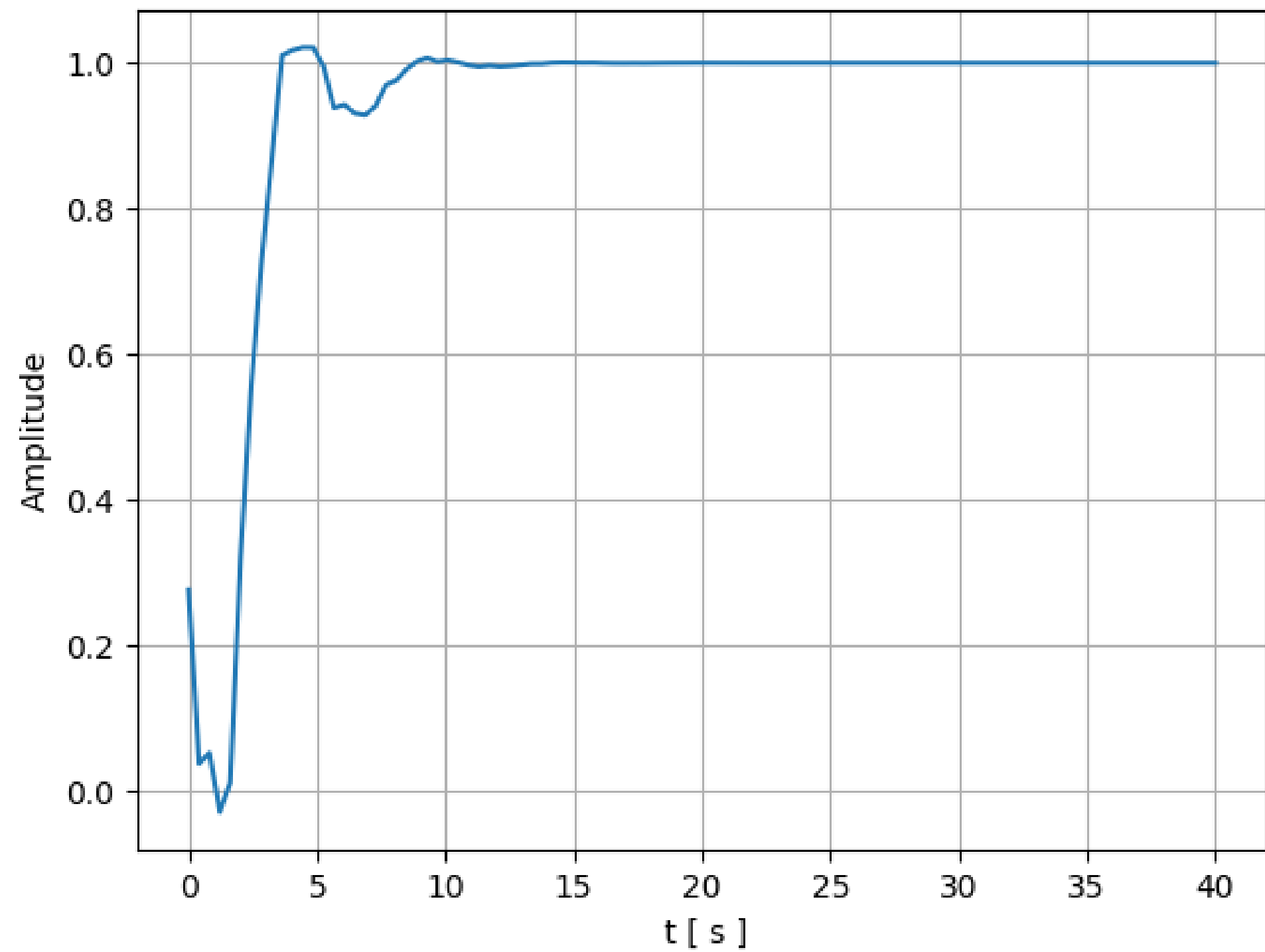
- Foi proposto por Rivera et al (1986).
- Utiliza um modelo interno do processo, que utiliza a função de transferência da planta para determinar os ajustes dos parâmetros PID.
- Seu uso presume-se um processo de baixa ordem sem atraso de resposta.
- A velocidade de resposta depende de um parâmetro λ . K_p , T_i e T_d tornam-se funções deste parâmetro.
- Quanto menor o valor de λ , mais rápida é a resposta e melhor é o desempenho. Porém, quanto mais baixo for λ , mais sensível o processo será às perturbações.
- Para um controlador PID, sugere-se que $\lambda / \theta > 0.8$.

```
# definindo as variáveis da função de transferência
k = 2
tau = 4.995
theta = 1.975

# controlador PID - IMC
lambda = 1.6 # lambda > 1.58
kp = ((2 * tau + theta) / (k * (2 * lambda + theta)))
ti = tau + (theta / 2)
td = (tau * theta) / (2 * tau + theta)

# construindo a função de transferência da planta
num = np.array([k])
den = np.array([tau, 1])
H = cnt.tf(num, den)
n_pade = 20
(num_pade, den_pade) = cnt.pade(theta, n_pade)
H_pade = cnt.tf(num_pade, den_pade)
Hs = cnt.series(H, H_pade)
```

Controlle PID - IMC



Método Cohen e Coon - CC

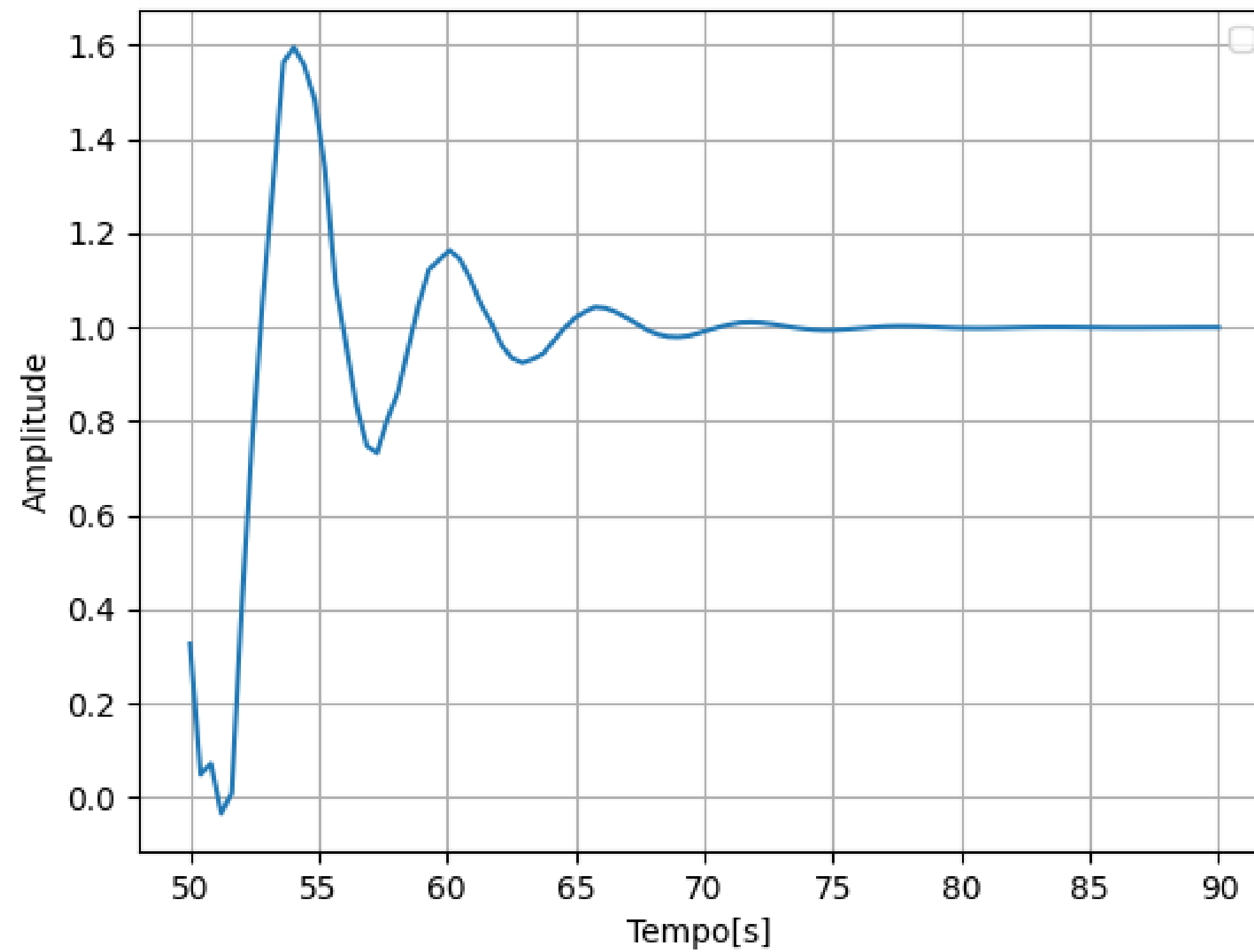
- Desenvolvido por George Z. Cohen e John M. Coon em 1953.
- É especialmente mais útil quando se lida com sistemas de processos industriais que possuem uma resposta mais lenta.
- É um método que não requer a indução de oscilações no sistema. Em vez disso, ele se baseia na análise da resposta do sistema a uma perturbação degrau e usa essa resposta para calcular os parâmetros do PID.
- Mais adequado para sistemas nos quais a estabilidade é uma preocupação crítica e sistemas de primeira ordem com tempo morto.
- Pode não ser a melhor escolha para sistemas de alta ordem ou que possuam características muito complexas.

```
# definindo as variáveis da função de transferência
k = 2
tau = 4.995
theta = 1.975

# controlador PID - CC
ke = (1/k)*(tau/theta)*((4/3+(1/4*(theta/tau))))
ti = theta*((32+6*(theta/tau))/(13+(8*(theta/tau))))
td = theta*((4/(11+(2*(theta/tau)))))

# construindo a função de transferência da planta
num = np.array([k])
den = np.array([tau, 1])
H_cc = cnt.tf(num, den)
n_pade = 20
(num_pade, den_pade) = cnt.pade(theta, n_pade)
H_pade = cnt.tf(num_pade, den_pade)
Hs = cnt.series(H_cc, H_pade)
```

Controle PID - CC



IMC vs CC

IMC Malha Aberta:

- Usa modelo interno no controlador.
- Simplifica dinâmica do sistema.
- Indicado com modelo preciso e perturbações conhecidas.

IMC Malha Fechada:

- Usa modelo interno para melhorar desempenho com realimentação.
- Rápida resposta e robustez a perturbações.
- Ideal quando feedback é crucial.

Cohen e Coon Malha Aberta:

- Sintonia PID clássica.
- Ajustes iterativos nos parâmetros.
- Aplicável com conhecimento limitado do sistema.

Cohen e Coon Malha Fechada:

- Sintonia PID considerando realimentação.
- Enfoca estabilidade e resposta transitória.
- Útil para otimizar desempenho sob feedback.

Interface - IMC

Escolha um método do controlador PID:

[1] - Método IMC

[2] - Método CC

[3] - Entrar com os parâmetros do PID

[4] - Sair

Escolha uma opção: 1

Kp calculado por IMC = 1.1650438169425512

Ti calculado por IMC = 5.9825

Td calculado por IMC = 0.8244985374007523

Interface - CC

Escolha um método do controlador PID:

[1] - Método IMC

[2] - Método CC

[3] - Entrar com os parâmetros do PID

[4] - Sair

Escolha uma opção: 2

Kp calculado por CC = 1.8110759493670885

Ti calculado por CC = 4.2000009289651329

Td calculado por CC = 0.6700144324645556

Escolha entre IMC e Cohen e Coon:

- IMC com modelo conhecido e perturbações compreendidas.
- Cohen e Coon quando modelo é incerto ou complexo.

Conclusão:

- Escolha depende da natureza do sistema, modelo conhecido, e requisitos de desempenho em malha aberta ou fechada.
- Ambos métodos oferecem abordagens distintas, proporcionando flexibilidade no projeto de controle.

Obrigada!