UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS - CCET
COORDENAÇÃO DO CURSO DE ENGENHARIA DA COMPUTAÇÃO

EECP0017 - ENGENHARIA DE CONTROLE

Professor: ME. MARCIO MENDES CERQUEIRA

Aluno: ANDRE MOURA LIMA – 20240065366.

ATIVIDADE 02

Resolução e simulação em Python das questões Capítulo 3 e 4 as questões são B3.4, B3.6, B3.7, B3.10, 4.2 e B4.12 do Ogata 5. ed.

Acesso o link do repositório no GitHub: Aqui.

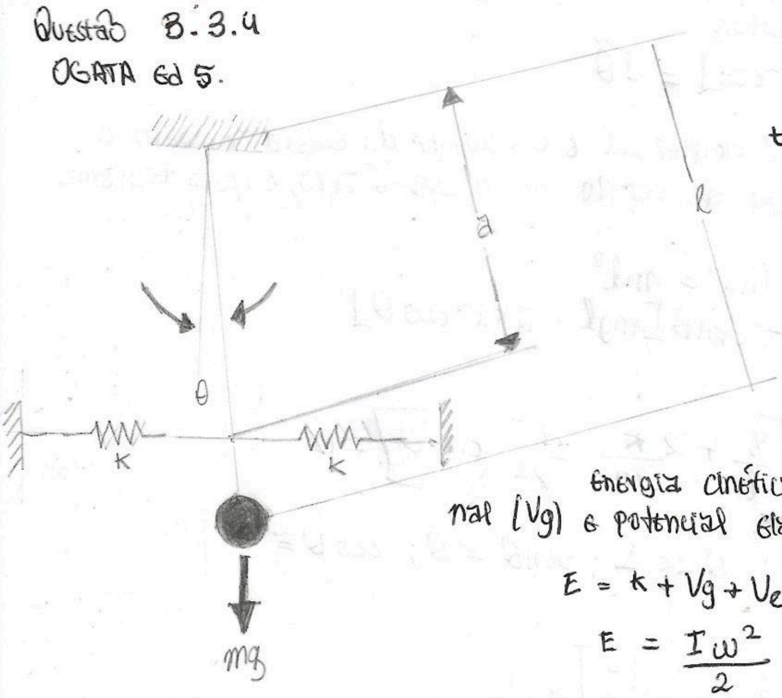
SÃO LUÍS - MA

2025.1

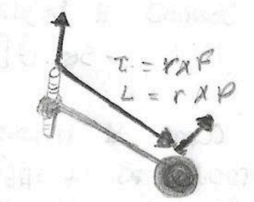
SUMÁRIO

B.3.4

Questão B.3.4
OGATA Ed 5.



torque
$\tau = \vec{r} \times \vec{F}$



2ª Lei de Newton para rotações
$\tau_{res} = I\alpha$ aceleração angular
$\alpha = \ddot{\theta}$ e I é o momento de inércia do sistema.

Energia cinética rotacional (K), potencial gravitacional (Vg) e potencial elástica (Uel)

$$E = K + V_g + U_{el}$$

$$E = \frac{I\omega^2}{2} + mgh + \frac{Kx^2}{2}$$

$$E = \frac{I\omega^2}{2} + mgh + \frac{Kx^2}{2}$$

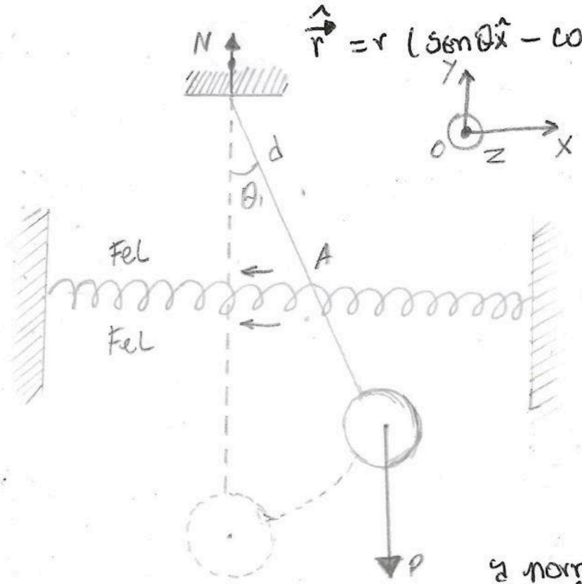
todas as forças externas, a energia se conserva (constante); logo, podemos dizer que

$$\frac{dE}{dt} = 0$$

$F = Kx = Ka\theta$
$\tau = F.\, a\, ka^2\theta$
$\tau m = -2ka^2\theta$
Pequenas oscilações
$\theta \approx \theta$ e $\cos\theta \approx 1$

Por torque
Peso da haste desprezível (ponto O)

$$\hat{\vec{r}} = r\,(\text{sen}\,\theta\hat{x} - \cos\theta\hat{y})$$



$$\tau_P = l(\text{sen}\,\theta\hat{x} - \cos\theta\hat{y}) \times (-mg\hat{y})$$
$$\tau_P = -mgl\,\text{sen}\,\theta\,\hat{z}$$

Cada mola exerce uma força elástica $F_{el} = kd\,\text{sen}\,\theta\hat{x}$, como são duas, multiplicamos por 2.

$$\tau_{el} = 2d(\text{sen}\,\theta\hat{x} - \cos\theta\hat{y}) \times kd\,\text{sen}\,\theta\hat{x}$$
$$\tau_{el} = -2kd^2\,\text{sen}\,\theta\cos\theta\,\hat{z}$$

a normal está sendo aplicada no ponto de rotação

$$\tau_N = 0$$
$$\tau_R = -\text{sen}\,\theta\,[mgl + 2kd^2\cos\theta]$$

Usando a Segunda Lei de Newton

$$-\operatorname{sen}\theta[mgL + 2kd^2\cos] = I\ddot{\theta}$$

Como a massa de haste é desprezível e o volume da esfera também o momento de inércia em relação ao centro de massa é zero, e, pelo teorema dos eixos paralelos.

$$I = I_{cm} + ML^2 = ml^2$$

$$mL^2\ddot{\theta} = -\operatorname{sen}\theta[mgl + 2kd^2\cos\theta]$$

Obtemos:

$$\ddot{\theta} + \operatorname{sen}\theta\left[\frac{g}{l} + 2\frac{k}{m}\frac{d^2}{l^2}\cos\theta\right] = 0$$

Para pequenas oscilações: $\theta << 1$; $\operatorname{sen}\theta \cong \theta$; $\cos\theta \cong 1$

Então:

$$\ddot{\theta} + \left[\frac{g}{l} + 2\frac{k}{m}\frac{d^2}{l^2}\right]\theta = 0$$

$$\ddot{\theta} + \left[\frac{2kd^2}{ml^2} + \frac{g}{l}\right]\theta = 0$$

Representação de espaço de estados $\underline{B.3.4}$

$$\ddot{\theta} + \left(\frac{2kd^2}{ml^2} + \frac{g}{l}\right)\theta = 0$$

Edo de 2ª ordem

$$\ddot{\theta} + w^2\theta = 0 \text{ com } w^2 = \frac{2kd^2}{ml^2} + \frac{g}{l}$$

$$x_1 = \theta, \; x_2 = \dot{\theta}$$

logo,
$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -w^2 x_1$$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -w^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$y = x_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

```
# Variáveis simbólicas
t = sp.symbols('t')                               # tempo
theta = sp.Function('theta')(t)                   # θ(t)
g, l, k, m, d = sp.symbols('g l k m d')   # constantes físicas

# Derivadas de theta
theta_dot = sp.diff(theta, t)
theta_ddot = sp.diff(theta, t, t)
```

Equação torque total:

```
# Torque resultante (projetado no eixo de rotação z)
tau_r = -sp.sin(theta) * (m * g * l + 2 * k * d**2 *
sp.cos(theta))
```

Aplicando a segunda Lei de Newton:

```
# Momento de inércia
I = m * l**2

# Equação de movimento rotacional
eq = sp.Eq(I * theta_ddot, tau_r)
```

Linearização (Pequenas Oscilações):

```
# Aproximações para pequenas oscilações
approx_eq = eq.subs({
    sp.sin(theta): theta,
    sp.cos(theta): 1
}).doit()

# Resolver isolando θ''(t)
theta_ddot_expr = sp.solve(approx_eq, theta_ddot)[0]
theta_ddot_expr
```

Temos:

$$\sin(\theta) \approx \theta, \quad \cos(\theta) \approx 1$$

$$\ddot{\theta} + \left(\frac{g}{l} + \frac{2kd^2}{ml^2}\right)\theta = 0$$

Portanto:

```
# Constante da equação
omega_squared = (g / l) + (2 * k * d**2) / (m * l**2)

# Forma final da EDO
final_eq = sp.Eq(sp.diff(theta, t, t) + omega_squared *
theta, 0)
final_eq
```

Resultado:

$$\ddot{\theta} + \left(\frac{g}{l} + \frac{2kd^2}{ml^2}\right)\theta = 0$$

Para representação em espaço de estado:

```
import sympy as sp

# Constantes simbólicas
g, l, k, m, d = sp.symbols('g l k m d')

# Estados
x1, x2 = sp.symbols('x1 x2')  # x1 = θ, x2 = θ'

# Derivadas dos estados
x1_dot = x2
omega_squared = (g / l) + (2 * k * d**2) / (m * l**2)
x2_dot = -omega_squared * x1
```

Forma Vetorial da Representação:

```
# Vetor de estado
X = sp.Matrix([x1, x2])

# Matriz A (dinâmica do sistema)
```

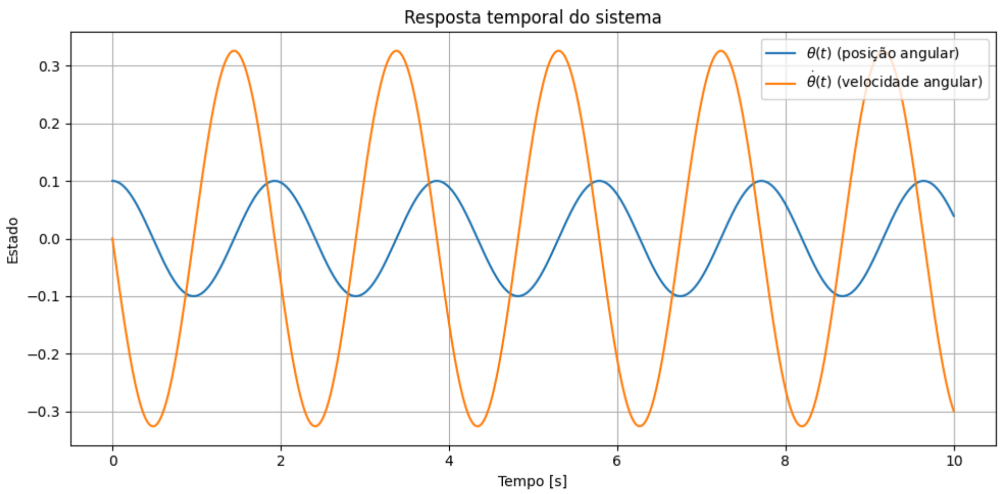
```
A = sp.Matrix([
    [0, 1],
    [-omega_squared, 0]
])

# Sem entrada (sistema autônomo)
B = sp.Matrix([
    [0],
    [0]
])

# Saída é a posição angular θ (x1)
C = sp.Matrix([[1, 0]])

# Sem entrada
D = sp.Matrix([[0]])
```

Chamando as variáveis temos:

```
A, B, C, D
```

```
(Matrix([
 [                          0, 1],
 [-2*d**2*k/(l**2*m) - g/l, 0]]),
 Matrix([
 [0],
 [0]]),
 Matrix([[1, 0]]),
 Matrix([[0]]))
```

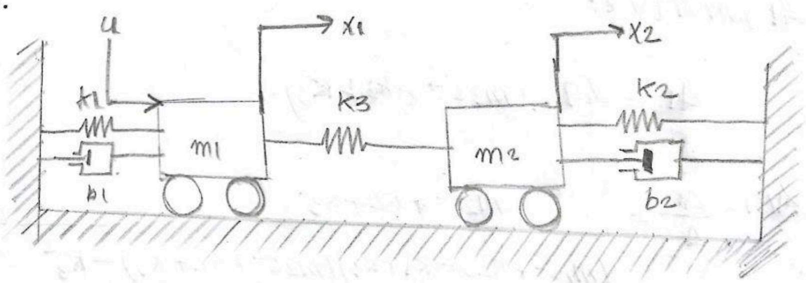
O que bateu com o resultado esperado do encontrado no livro do Ogata ed. 5. Com isso obtemos o resultado pico a pico:



O gráfico apresenta a resposta temporal de um sistema oscilatório, representando a evolução da posição angular $\theta(t)$ e da velocidade angular $\theta(t)$ ao longo do tempo. A curva azul mostra a posição angular com um comportamento senoidal, enquanto a curva laranja representa a velocidade angular, exibindo um padrão cossenoidal defasado em relação à posição. O sistema claramente possui dinâmica harmônica, com oscilações sustentadas e sem amortecimento visível, sugerindo um sistema conservativo, como um pêndulo ideal ou um sistema massa-mola sem atrito. As amplitudes permanecem constantes ao longo do tempo, indicando que não há perda de energia no sistema, e a frequência das oscilações também é constante, reforçando a característica de estabilidade periódica. Este tipo de resposta é típico de sistemas físicos com conservação de energia e sem forças dissipativas atuando.

B.3.6

Questão B.3.6
OGATA Ed 5.



Precisamos então encontrar a função que relaciona a posição de cada um dos blocos com a força u aplicada.

$$F_r = \sum F$$

$$m_1 \ddot{x}_1 = -k_1 x_1 - k_2 \dot{x}_1 - k_3 (x_1 - x_2) + u$$
$$m_2 \ddot{x}_2 = -k_2 x_2 - k_3 (x_2 - x_1)$$

$$m_1 \ddot{x}_1 + k_1 x_1 + k_2 \dot{x}_1 + k_3 x_1 - k_3 x_2 = u \quad (1)$$
$$m_2 \ddot{x}_2 + k_2 x_2 + k_3 x_2 - k_3 x_1 = 0 \quad (2)$$

Aplicando a transformada de Laplace

equação 1:

$$m_1 s^2 X_1(s) + k_1 X_1(s) + k_2 s X_1(s) + k_3 X_1(s) - k_3 X_2(s) = U(s)$$
$$(m_1 s^2 + k_2 s + k_1 + k_3) X_1(s) - k_3 X_2(s) = U(s) \quad (3)$$

equação 2:

$$m_2 s^2 X_2(s) + k_2 X_2(s) + k_3 X_2(s) - k_3 X_1(s) = 0$$
$$-k_3 X_1(s) + (m_2 s^2 + k_2 + k_3) X_2(s) = 0 \quad (4)$$

Podemos escrever:

$$\begin{bmatrix} m_1 s^2 + k_2 s + k_1 + k_3 & -k_3 \\ -k_3 & m_2 s^2 + k_2 + k_3 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} U(s) \\ 0 \end{bmatrix}$$

Para encontrar $\frac{X_1(s)}{U(s)}$: regra de Cramer

$$\Delta = (m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2$$

Determinante Δ_1 para $X_1(s)$ é:

$$\Delta_1 = U(s) \cdot (m_2 s^2 + k_2 + k_3)$$

$$X_1(s) = \frac{\Delta_1}{\Delta} = \frac{m_2 s^2 + k_2 + k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2} U(s)$$

FT: $\frac{X_1(s)}{U(s)}$

$$\frac{X_1(s)}{U(s)} = \frac{m_2 s^2 + k_2 + k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2}$$

Portanto: Usamos a equação (4)

$$X_2(s) = \frac{k_3}{(m_2 s^2 + k_2 + k_3)} X_1(s)$$

Substituindo $X_1(s)$.

$$\frac{X_2(s)}{U(s)} = \frac{k_3}{m_2 s^2 + k_2 + k_3} \cdot \frac{m_2 s^2 + k_2 + k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2}$$

Simplificando temos:

$$\frac{X_2(s)}{U(s)} = \frac{k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2}$$

Resultado:

$$\frac{X_1(s)}{U(s)} = \frac{m_2 s^2 + k_2 + k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2}$$

$$\frac{X_2(s)}{U(s)} = \frac{k_3}{(m_1 s^2 + k_2 s + k_1 + k_3)(m_2 s^2 + k_2 + k_3) - k_3^2}$$

Representação em diagrama de estados B.3.6

$$x_1 = x_1$$
$$x_2 = \dot{x}_1$$
$$x_3 = x_2$$
$$x_4 = \dot{x}_2$$

Apartir das equações originais

$$m_1\ddot{x}_1 + k_2\dot{x}_1 + (k_1+k_3)x_1 - k_3x_2 = u$$
$$m_2\ddot{x}_2 + k_2x_2 + k_3x_2 - k_3x_1 = 0$$

Podemos isolar as derivadas da segunda ordem:

$$\ddot{x}_1 = \frac{1}{m_1}\left(-k_2\dot{x}_1 - (k_1+k_3)x_1 + k_3x_2 + u\right)$$
$$\ddot{x}_2 = \frac{1}{m_2}\left(-k_2x_2 - k_3x_2 + k_3x_1\right)$$

Agora, escrevemos o sistema em forma matricial

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{m_1}\left(-k_2x_2 - (k_1+k_3)x_1 + k_3x_3 + u\right)$$
$$\dot{x}_3 = x_4$$
$$\dot{x}_4 = \frac{1}{m_2}\left(-k_2x_3 - k_3x_3 + k_3x_1\right)$$

Em notação matricial

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Onde:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{k_1+k_3}{m_1} & -\frac{k_2}{m_1} & \frac{k_3}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_3}{m_2} & 0 & -\frac{k_2+k_3}{m_2} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix}$$

B.3.6

Saída como x_1 e x_2:

$$C=\begin{bmatrix}1&0&0&0\\0&0&1&0\end{bmatrix},\quad D=\begin{bmatrix}0\\0\end{bmatrix}$$

Representação em python:

Para resolver essa questão é necessário instalar no colab a biblioteca **!pip install control** que facilita resolver o método matemático correto com algumas funções embutidas e as outras bibliotecas também.

```
import numpy as np
import control as ct
from sympy import symbols

# Parâmetros do sistema como símbolos
m1, m2 = symbols('m1 m2')
k1, k2, k3 = symbols('k1 k2 k3')
s = symbols('s')  # Variável complexa da transformada de
Laplace

# Funções de Transferência
print("\n--- Funções de Transferência ---")
print(f"X1(s)/U(s) =")
print(f"({m2}*s^2 + {k2} + {k3}) / [({m1}*s^2 + {k2}*s + {k1}
+ {k3})*({m2}*s^2 + {k2} + {k3}) - {k3}^2]")

print(f"\nX2(s)/U(s) =")
print(f"{k3} / [({m1}*s^2 + {k2}*s + {k1} + {k3})*({m2}*s^2 +
{k2} + {k3}) - {k3}^2]")

# Espaço de Estado
print("\n--- Representação em Espaço de Estado ---")
print("Matriz A:")
print(f"[[0, 1, 0, 0]")
print(f" [{-(k1 + k3)/m1}, {-k2/m1}, {k3/m1}, 0]")
print(f" [0, 0, 0, 1]")
print(f" [{k3/m2}, 0, {-(k2 + k3)/m2}, 0]]")

print("\nMatriz B:")
print(f"[[0]")
print(f" [{1/m1}]")
print(f" [0]")
```

```
print(f"  [0]]")

print("\nMatriz C:")
print("[[1, 0, 0, 0]")
print(" [0, 0, 1, 0]]")

print("\nMatriz D:")
print("[[0]")
print(" [0]]")
```

O código realiza o cálculo e acha a função de transferência e em seguida ele acha também a representação de espaço de estados que pode ser representado por índices como o código posteriormente.

```
--- Funções de Transferência ---
X1(s)/U(s) =
(m2*s^2 + k2 + k3) / [(m1*s^2 + k2*s + k1 + k3)*(m2*s^2 + k2 + k3) - k3^2]

X2(s)/U(s) =
k3 / [(m1*s^2 + k2*s + k1 + k3)*(m2*s^2 + k2 + k3) - k3^2]

--- Representação em Espaço de Estado ---
Matriz A:
[[0, 1, 0, 0]
 [(-k1 - k3)/m1, -k2/m1, k3/m1, 0]
 [0, 0, 0, 1]
 [k3/m2, 0, (-k2 - k3)/m2, 0]]

Matriz B:
[[0]
 [1/m1]
 [0]
 [0]]

Matriz C:
[[1, 0, 0, 0]
 [0, 0, 1, 0]]

Matriz D:
[[0]
 [0]]
```

A resolução acima mostra exatamente o que foi encontrado onde a função de transferência é exposta deliberadamente para encontrar os espaços de estados da forma matricial.

Em índices seria:

```
import numpy as np
```

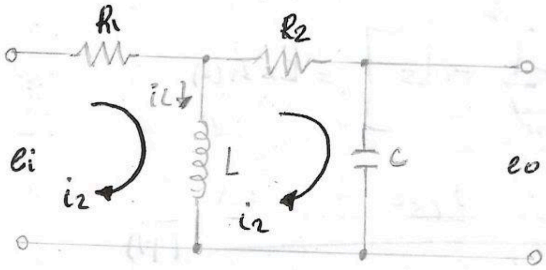
```python
import control as ct

# Parâmetros do sistema
m1, m2 = 1.0, 1.0
k1, k2, k3 = 1.0, 1.0, 1.0

# Funções de Transferência
print("\n--- Funções de Transferência ---")
num_X1 = [m2, 0, k2 + k3]
den_X1 = [m1*m2, 0, m1*(k2 + k3) + m2*(k1 + k3), 0, (k1 +
k3)*(k2 + k3) - k3**2]
sys_X1 = ct.TransferFunction(num_X1, den_X1)
print(f"X1(s)/U(s) = \n{sys_X1}\n")

num_X2 = [k3]
den_X2 = den_X1  # mesmo denominador
sys_X2 = ct.TransferFunction(num_X2, den_X2)
print(f"X2(s)/U(s) = \n{sys_X2}\n")

# Espaço de Estado
print("\n--- Representação em Espaço de Estado ---")
A = np.array([
    [0, 1, 0, 0],
    [-(k1 + k3)/m1, -k2/m1, k3/m1, 0],
    [0, 0, 0, 1],
    [k3/m2, 0, -(k2 + k3)/m2, 0]
])
B = np.array([[0], [1/m1], [0], [0]])
C = np.array([[1, 0, 0, 0], [0, 0, 1, 0]])
D = np.array([[0], [0]])

sys_ss = ct.ss(A, B, C, D)
print("Matriz A:\n", A)
print("\nMatriz B:\n", B)
print("\nMatriz C:\n", C)
print("\nMatriz D:\n", D)
```

Resultado:

```
--- Representação em Espaço de Estado ---
Matriz A:
 [[ 0.  1.  0.  0.]
 [-2. -1.  1.  0.]
 [ 0.  0.  0.  1.]
 [ 1.  0. -2.  0.]]

Matriz B:
 [[0.]
 [1.]
 [0.]
 [0.]]

Matriz C:
 [[1 0 0 0]
 [0 0 1 0]]
```

B.3.7

B.3.7



$$\sum v = 0$$
$$-e_i + V_{R_1} + V_L = 0$$
$$-V_L + V_{R2} + V_C = 0$$
$$V_C = e_o$$

Aplicando a lei de Kirchhoff das tensões, pois são mostradas no circuito as correntes das malhas.

A tensão no indutor e no capacitor são dadas por:

$$V_C = L\frac{d}{dt} i_L \qquad V_C = \frac{1}{C}\int_t i_C(t)\,dt$$

Aplicando a lei de Kirchhoff das correntes no nó superior (correntes que chegam = correntes que saem do nó), considerando a corrente no indutor:

$$\sum_{\text{chegando}} i = \sum_{\text{saindo}} i \Rightarrow i_1 = i_2 + i_L$$

$$R_1 i_1 + L\frac{d}{dt}[i_L] = R_1 i_1 + L\frac{d}{dt}[i_1 - i_2] = e_i \quad (I)$$

$$-L\frac{d}{dt}[i_1 - i_2] + R_2 i_2 + \frac{1}{C}\int_t [i_2]\,dt = 0$$

$$L\frac{d}{dt}[i_2 - i_1] + R_2 i_2 + \frac{1}{C}\int_t i_2\,dt = 0 \quad (II)$$

$$\frac{1}{C}\int_t i_2\,dt = e_o \quad (III)$$

Aplicando a transformada de Laplace com condições iniciais nulas:

$$R_1 I_1(s) + Ls[I_1(s) - I_2(s)] = E_i(s)$$
$$Ls[I_2(s) - I_1(s)] + R_2 I_2(s) + \frac{1}{sC} I_2(s) = 0$$

Objetivo é determinar uma única equação que tenha como variáveis no domínio s $E_i(s)$ e $E_o(s)$, ou seja, que não tenha $I_1(s)$ e $I_2(s)$.

B.3.7

Segunda Equação.

$$I_2(s)\left[R_2+\frac{1}{sC}+Ls\right]=LsI_1(s)$$

$$I_2(s)=\frac{LCs^2}{LCs^2+R_2Cs+1} \quad (IV)$$

Substituindo (IV) na primeira equação

$$R_1I_1(s)+Ls\left[I_1(s)-\left(\frac{LCs^2}{LCs^2+R_2Cs+1}I_1(s)\right)\right]=E_i(s)$$

$$N=R_1\left[LCs^2+R_2Cs+1\right]+Ls\left[LCs^2+R_2Cs+1\right]-LsLCs^2$$
$$N=R_1LCs^2+R_1R_2Cs+R_1+L^2Cs^3+LR_2Cs^2+Ls-L^2Cs^3$$
$$N=LC(R_1+R_2)s^2+(R_1R_2C+L)+sR_1$$

$$I_1(s)\left[\frac{LC(R_1+R_2)s^2+(R_1R_2C+L)s+R_1}{LCs^2+R_2Cs+1}\right]=E_i(s) \quad V$$

Substituindo IV na segunda equação:

$$\frac{1}{sC}\left[\frac{LCs^2}{LCs^2+R_2Cs+1}I_1(s)\right]=E_0(s)$$

$$\frac{Ls}{LCs^2+R_2Cs+1}I_1(s)=E_0(s)$$

$$I_1(s)=E_0(s)\left(\frac{LCs^2+R_2Cs+1}{Ls}\right) \quad VI$$

Substituindo VI em V:

$$\left(\frac{LCs^2+R_2Cs+1}{Ls}\right)E_0(s)\left[\frac{LC(R_1+R_2)s^2+(R_1R_2C+L)s+R_1}{LCs^2+R_2Cs+1}\right]=E_i(s)$$

B.3.7
OGATA 6d5

Continuação

FT:

$$\frac{E_o(s)}{E_i(s)} = \frac{Ls}{LC(R_1+R_2)s^2 + (R_1R_2C+L)s + R_1}$$

Representação de Espaço de estado:

B.3.7

com:
$$\frac{E_o(s)}{E_i(s)} = \frac{Ls}{LC(R_1+R_2)s^2+(CR_1R_2+L)s+R_1}$$

$$a_2 = LC(R_1+R_2)$$
$$a_1 = CR_1R_2+L$$
$$a_0 = R_1$$
$$b_1 = L, b_0 = 0$$

$$\frac{E_o(s)}{E_i(s)} = \frac{b_1 s+b_0}{a_2 s^2+a_1 s+a_0}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{R_1}{CL(R_1-R_3)} & \frac{-CR_1R_2-L}{CL(R_1+R_2)} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & \frac{1}{C(R_1+R_2)} \end{bmatrix}$$

$$D = [0]$$

Representação em Python:

Importando dados da questão:

```
import sympy as sp

# Variáveis simbólicas
s = sp.symbols('s')
R1, R2, L, C = sp.symbols('R1 R2 L C')
Ei = sp.Function('Ei')(s)
Eo = sp.Function('Eo')(s)
I1 = sp.Function('I1')(s)
I2 = sp.Function('I2')(s)
```

Equações do Circuito no Domínio de Laplace:

```
# Primeira equação (malha 1)
eq1 = sp.Eq(R1 * I1 + L * s * (I1 - I2), Ei)

# Segunda equação (malha 2)
eq2 = sp.Eq(L * s * (I2 - I1) + R2 * I2 + (1 / (s * C)) * I2,
0)
```

Resolvendo o sistema para I1(s) e I2(s)

```
sol = sp.solve((eq1, eq2), (I1, I2), dict=True)[0]
I1_sol = sol[I1]
I2_sol = sol[I2]
```

Expressão para $E\ o\ (\ s\ )$ = 1 $s\ C\ \cdot\ I$ 2 (s) Eo(s)= sC 1 · I2(s)

```
Eo_expr = (1 / (s * C)) * I2_sol
```

Função de Transferência:

```
FT = sp.simplify(Eo_expr / Ei)
FT
```

Assim, obtivemos a solução por parte da questão se simplificarmos termos:

```
import sympy as sp

# Definindo as variáveis simbólicas
s = sp.symbols('s')
R1, R2, L, C = sp.symbols('R1 R2 L C')

# Numerador: Ls
numerador = L * s

# Denominador: LC(R1 + R2)s^2 + (CR1R2 + L)s + R1
denominador = L*C*(R1 + R2)*s**2 + (C*R1*R2 + L)*s + R1

# Função de transferência
G = numerador / denominador

# Simplificar (opcional, mas ajuda)
G_simplificada = sp.simplify(G)

# Exibir
sp.pretty_print(G_simplificada)
```

Assim, temos o calculado:

$$\frac{L \cdot s}{C \cdot L \cdot s^{2} \cdot (R_1 + R_2) + R_1 + s \cdot (C \cdot R_1 \cdot R_2 + L)}$$

Representação do espaço de estado:

```
import sympy as sp
from sympy import symbols, Matrix, init_printing, simplify

# Inicializar impressão bonita
init_printing()

# Definição das variáveis simbólicas
s = sp.symbols('s')
R1, R2, L, C = sp.symbols('R1 R2 L C')

# Coeficientes do denominador e numerador da função de
transferência
a2 = L * C * (R1 + R2)
a1 = C * R1 * R2 + L
a0 = R1  # Corrigido aqui

b1 = L
b0 = 0

# Construção das matrizes do espaço de estado (forma canônica
controlável)
A = Matrix([
    [0, 1],
    [-a0/a2, -a1/a2]
])

B = Matrix([
    [0],
    [1]
])

C_matrix = Matrix([
    [b0/a2, (b1 - b0 * a1/a2)/a2]
])

D = Matrix([
    [0]
])
```

```
# Exibição das matrizes
print("Matriz A:")
display(A)

print("Matriz B:")
display(B)

print("Matriz C:")
display(C_matrix)

print("Matriz D:")
display(D)
```

Logo a solução em estado é:

Matriz A:

$$\begin{bmatrix} 0 & 1 \\ -\frac{R_1}{CL(R_1+R_2)} & \frac{-CR_1R_2-L}{CL(R_1+R_2)} \end{bmatrix}$$

Matriz B:

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Matriz C:

$$\begin{bmatrix} 0 & \frac{1}{C(R_1+R_2)} \end{bmatrix}$$

Matriz D:

$$\begin{bmatrix} 0 \end{bmatrix}$$

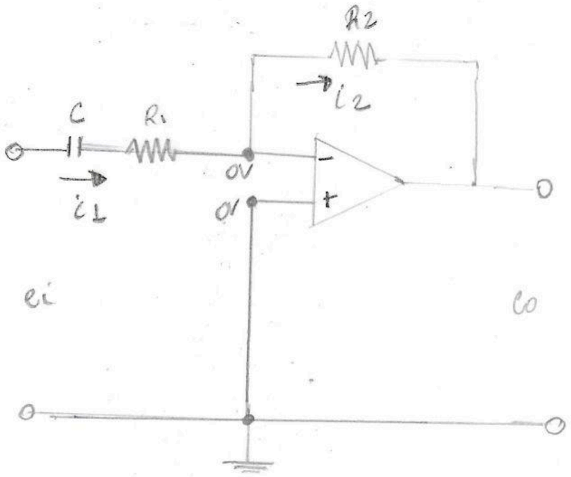
Obtivemos os resultados de cada solução assim extrair em Python para solução.

B.3.10

B.3.10
OGATA Ed 5

Obtenha a função de transferência $E_o(s)/E_i(s)$ do circuito:

inversor simples.



$$V_c = \frac{1}{c}\int i\,dt \qquad i_1 = i_2$$

$$V_c = \frac{1}{Cs}\cdot I(s) \qquad \frac{ei - 0}{\frac{1}{c} + R_1} = 0 - eo \Rightarrow \frac{\frac{ei}{1+R_1Cs}}{Cs} =$$

$$Z(e) = \frac{V(s)}{I(s)} = \frac{1}{Cs} \qquad -\frac{eo}{R_2} \rightarrow \frac{Cs \cdot ei}{1 + R_1Cs} = -\frac{eo}{R_2} \Rightarrow \frac{eo}{ei} =$$

$$-\frac{Cs \cdot R_2}{1 + R_1Cs}$$

Representação de espaço de Estado:

B.3.10

Simplificações para diagrama de Estado (Espaço de Estado).

$$x(t) = E_o(t)$$

$$\frac{dx}{dt} = \frac{-CR_2 \frac{dE_i(t)}{dt} - x(t)}{R_1 C}$$

$$\frac{E_o(s)}{E_i(s)} = \frac{N(s)}{D(s)} = \frac{-CR_2 s}{R_1 Cs + 1}$$

$$\dot{x} = ax + bu$$
$$y = cx + du$$

$$G(s) = \frac{-CR_2 s}{R_1 Cs + 1}$$

$$\dot{x} = -\frac{1}{R_1 C} x + -\frac{CR_2}{R_1 C} u$$

$$A = \left[-\frac{1}{R_1 C}\right], \quad B = \left[\frac{-CR_2}{R_1 C}\right], \quad C = [1], \quad D = [0]$$

Representação de Python:

```python
import sympy as sp

# Define símbolos
s, C, R1, R2 = sp.symbols('s C R1 R2')

# Define numerador e denominador simbolicamente
numerador = -C * s * R2
denominador = 1 + R1 * C * s

# Calcula a função de transferência
G = numerador / denominador

# Constrói a saída manualmente, usando substituição simbólica para o formato desejado
# Converte partes da função em string no formato que você quer (Cs·R2, R1Cs, etc.)
numerador_str = "-Cs·R₂"
denominador_str = "1 + R₁Cs"
linha = "-" * max(len(numerador_str), len(denominador_str)) * 2  # para centralizar melhor

# Imprime resultado no estilo desejado
print("\nFunção de Transferência Eo(s)/Ei(s):\n")
print(f"         {numerador_str}")
print(f"     {linha}")
print(f"        {denominador_str}")
```

Resultado:

```
Função de Transferência Eo(s)/Ei(s):

        -Cs·R₂
    ----------------
       1 + R₁Cs
```

A função de transferência para o circuito inversor simples é obtida com base nos cálculos do ogata ed. 5.

Representação de espaço de estados:

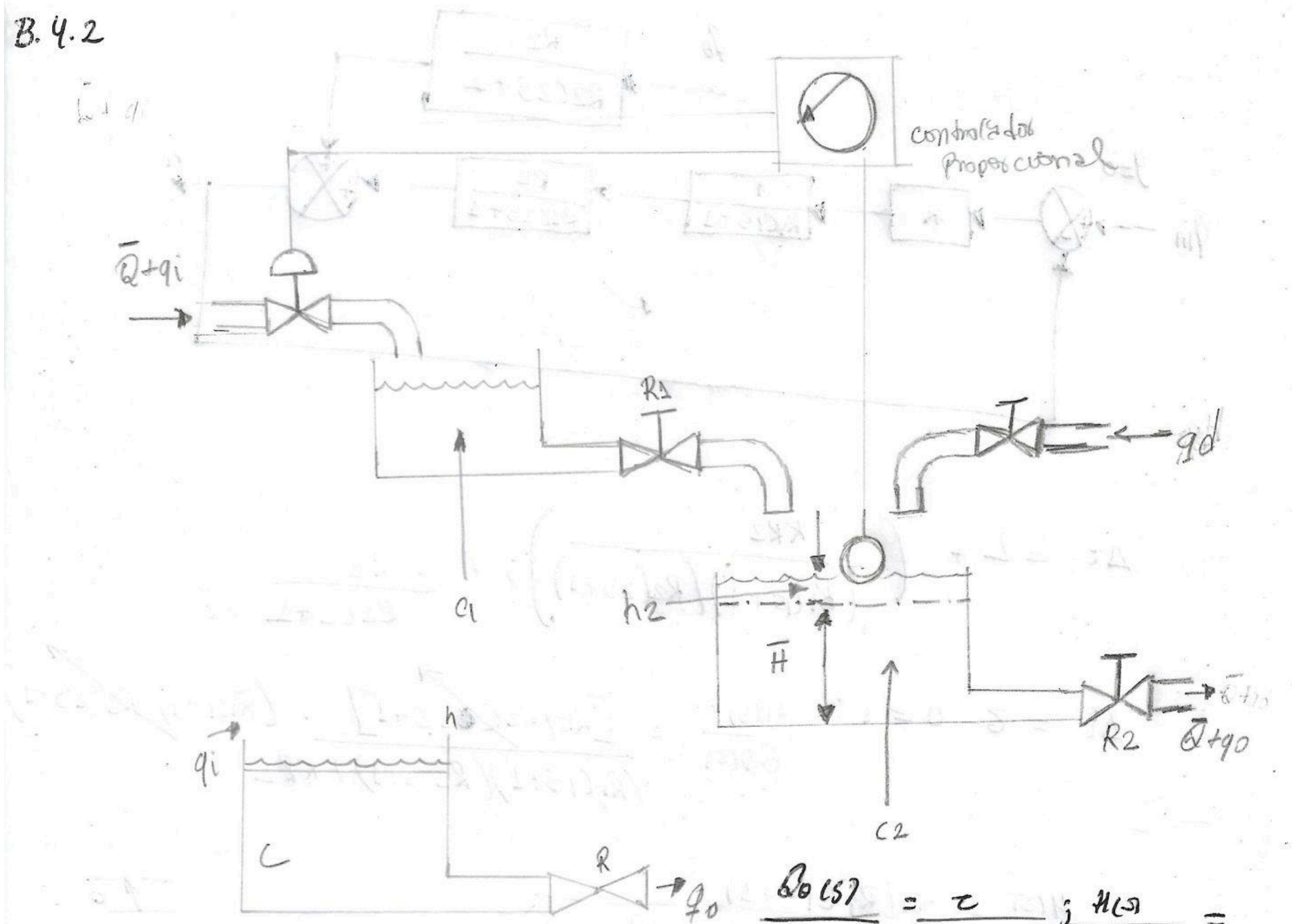
```
import sympy as sp

# Define símbolos
R1, R2, C = sp.symbols('R1 R2 C')

# Matrizes do sistema
A = sp.Matrix([[-1 / (R1 * C)]])
B = sp.Matrix([[-C * R2 / (R1 * C)]])
C_mat = sp.Matrix([[1]])
D = sp.Matrix([[0]])

# Impressão das matrizes
print("\nRepresentação em Espaço de Estado:\n")
print("A =")
sp.pprint(A)
print("\nB =")
sp.pprint(B)
print("\nC =")
sp.pprint(C_mat)
print("\nD =")
sp.pprint(D)
```

Resultado:

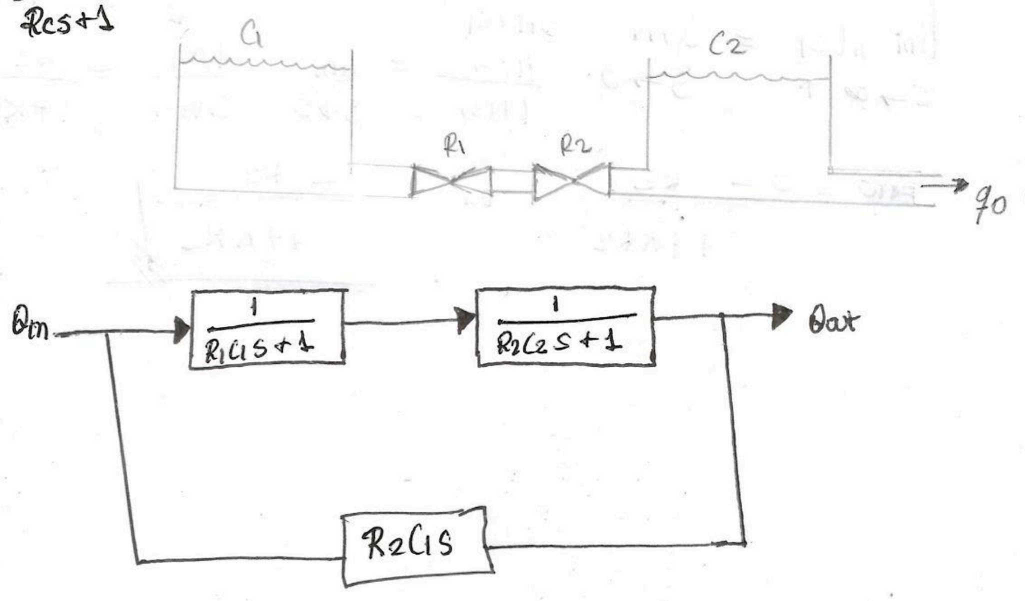
```
Representação em Espaço de Estado:

A =
⎡ -1 ⎤
⎢────⎥
⎣C⋅R₁⎦

B =
⎡-R₂ ⎤
⎢────⎥
⎣ R₁ ⎦

C =
[1]

D =
[0]
```

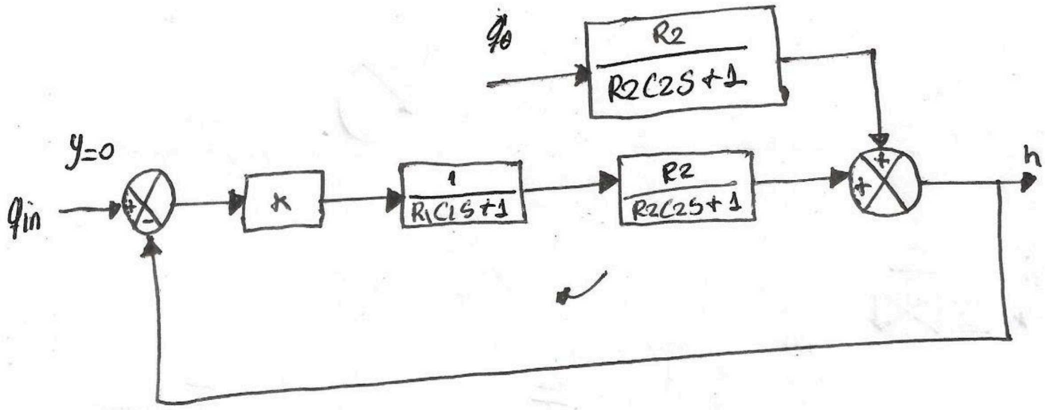
B.4.2

B. 4.2



$$\frac{Q_o(s)}{Q_{in}(s)} = \frac{\tau}{Rcs+1} \quad ; \quad \frac{H(s)}{Q_i(s)} = \frac{R}{Rcs+1}$$





$$\Delta t = L + \left(\frac{KR_2}{(R_1C_1s+1)(R_2C_2S+1)} \right) ; \quad P_d = \frac{R_2}{R_2C_2+1}$$

$$\Delta i = \tau - 0 = 1 \therefore \frac{H(s)}{Q_0(s)} = \frac{[R_2/R_2\cancel{C_2S+1}]}{(R_1C_1S+1)(R_2C_2+1)+KR_2} \cdot (R_1C_1+1)\cancel{(R_2C_2S+1)}$$

$$H(s) = \frac{R_2(R_1C_1S+1)}{(R_1C_1S+1)(R_2C_2S+1)+KR_2}$$

$$\lim_{t\to\infty} h(t) = \lim_{S\to 0} \frac{SH(s)}{\frac{H(s)}{1(s)}} = \lim_{S\to 0} \frac{H(s)}{Q_0(s)} = \frac{R_2}{1+KR_2}$$

$$\text{Erro} = 0 - \frac{R_2}{1+KR_2} \quad \text{ou} \quad \frac{-R_2}{1+KR_2}$$

Representação de espaço de estado:

B.4.2 Representação de diagrama estado...

$$Erro = -\frac{R_2}{1+KR_2}$$

$$G(s) = \frac{R_2}{s(\tau s+1)}$$

$$T(s) = \frac{KG(s)}{1+KG(s)} = \frac{K \cdot \frac{R_2}{s(\tau s+1)}}{1+K \cdot \frac{R_2}{s(\tau s+1)}} = \frac{KR_2}{s(\tau s+1)+KR_2}$$

$$T(s) = \frac{KR2}{\tau s^2+s+KR_2}$$

$$\tau \ddot{y} + \dot{y} + KR2y = KR2u$$

$x_1 = y$ (saída)
$x_2 = \dot{y}$ (derivada da saída)

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{1}{\tau}(-x_2 - KR2x_1 + KR2u)$$

$$y = x_1$$

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{KR_2}{\tau} & -\frac{1}{\tau} \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{KR2}{\tau} \end{bmatrix}, C = [1\ 0], D = 0$$

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, C = [1\ 0], D = [0]$$

Representação em python:

```
import sympy as sp


# Definir variáveis simbólicas
s = sp.symbols('s')
R1, R2, C1, C2, K = sp.symbols('R1 R2 C1 C2 K')

# H(s) = [R2(R1C1s + 1)] / [(R1C1s + 1)(R2C2s + 1) + KR2]
numerador = R2 * (R1 * C1 * s + 1)
denominador = (R1 * C1 * s + 1) * (R2 * C2 * s + 1) + K * R2
H_s = numerador / denominador

# Erro final: 0 - lim_{s -> 0} H(s)
lim_h = sp.limit(H_s, s, 0)
erro_final = -lim_h.simplify()

print("Erro final =", erro_final)
```

Resultado:

```
Erro final = -R2/(K*R2 + 1)
```

O resultado da maneira é exposta a solução para a questão.

Representação de espaços de estados:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import ss2tf, step

# Parâmetros do sistema
R2 = 1.0  # Valor de R2
K = 2.0   # Ganho do controlador
tau = 0.5 # Constante de tempo

# Representação matricial do espaço de estados
A = np.array([[0, 1],
```

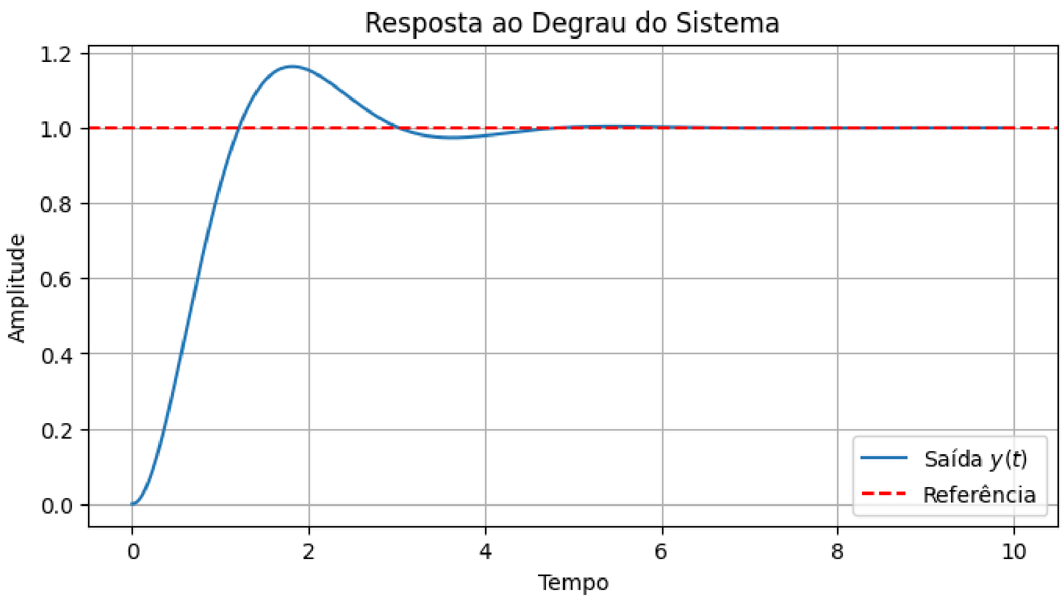
```
              [-K*R2/tau, -1/tau]])
B = np.array([[0],
              [K*R2/tau]])
C = np.array([[1, 0]])
D = np.array([[0]])

# Exibindo as matrizes
print("Matriz de Estado (A):")
print(A)
print("\nMatriz de Entrada (B):")
print(B)
print("\nMatriz de Saída (C):")
print(C)
print("\nMatriz de Transmissão Direta (D):")
print(D)

# Função de transferência a partir do espaço de estados
num, den = ss2tf(A, B, C, D)
print("\nFunção de Transferência:")
print(f"Numerador: {num}")
print(f"Denominador: {den}")

# Cálculo do erro final teórico
error_final = -R2 / (K * R2 + 1)
print(f"\nErro final teórico: {error_final}")

# Resposta ao degrau para verificação
t, y = step((A, B, C, D), T=np.linspace(0, 10, 1000))
steady_state_error = 1 - y[-1]  # Para degrau unitário
print(f"Erro final simulado: {steady_state_error}")

# Plot da resposta ao degrau
plt.figure(figsize=(8, 4))
plt.plot(t, y, label='Saída $y(t)$')
plt.axhline(y=1, color='r', linestyle='--',
label='Referência')
plt.xlabel('Tempo')
plt.ylabel('Amplitude')
plt.title('Resposta ao Degrau do Sistema')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

Resultados:

```
Matriz de Estado (A):
[[ 0.  1.]
 [-4. -2.]]

Matriz de Entrada (B):
[[0.]
 [4.]]

Matriz de Saída (C):
[[1 0]]

Matriz de Transmissão Direta (D):
[[0]]

Função de Transferência:
Numerador: [[0. 0. 4.]]
Denominador: [1. 2. 4.]

Erro final teórico: -0.3333333333333333
Erro final simulado: -2.4293994808077812e-05
```

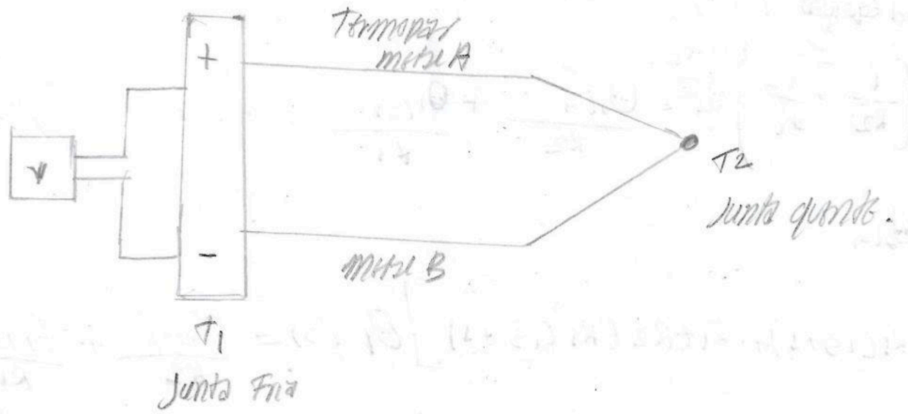
Assim, podemos concluir através da resposta ao degrau do termopar.



O gráfico mostra a resposta ao degrau de um sistema dinâmico, destacando o comportamento transitório da saída $y(t)$ frente a uma entrada de referência unitária (1). Observa-se que a resposta apresenta um sobresinal significativo, atingindo um pico acima de 1,1 antes de se estabilizar em torno do valor de regime. Esse comportamento é típico de sistemas de segunda ordem subamortecidos, indicando a presença de oscilações transitórias antes da estabilização. Após cerca de 6 segundos, a saída converge para a referência com um erro desprezível, evidenciando boa estabilidade e resposta rápida, apesar do pequeno excesso inicial. O uso da linha tracejada vermelha como referência facilita a visualização do erro de regime e da qualidade da resposta do sistema.

B.4.12

B.4.12



θ_0 = temperatura
θ_1 = temperatura do termopar
θ_2 = temperatura do poço térmico
R_1, R_2: Resistências térmicas do termopar
C_1, C_2 : capacitâncias térmicas do termopar e do poço térmico
h_1, h_2 : taxas de transferência de calor.

Sistema
Termopar
Resistência térmica : R_1
Capacitância térmica : C_1
constante de tempo $\tau_1 = R_1C_1 = 2s$
Poço Térmico (contém, protege, o termopar)
Resistência térmica : R_2
capacitância térmica : C_2
constante de tempo : $\tau_2 = R_2C_2 = 30s$.

$$\frac{C_1}{C_2} = \frac{m_1}{m_2} = \frac{8g}{40g} = \frac{1}{5} \Rightarrow C_1 = \frac{C_2}{5}$$

termopar (θ_1):

$$C_1 \frac{d\theta_1}{dt} = h_1 = \frac{\theta_2 - \theta_1}{R_1}$$

$$R_1C_1 \frac{d\theta_1}{dt} + \theta_1 = \theta_2 . \quad [1]$$

Rearranjando

$$C_2 \frac{d\theta_2}{dt} + \left(\frac{1}{R_2} + \frac{1}{R_1}\right)\theta_2 = \frac{\theta_0}{R_2} + \frac{\theta_1}{R_1} . \quad [2]$$

Transformada de Laplace e funções de transferência

$$R_1C_1 s\,\theta_1(s) + \theta_1(s) = \theta_2(s) \rightarrow \theta_2(s) = (R_1C_1 s + 1)\,\theta_1(s).$$

Expressão 2

$\Theta_2(s)$ da equação

$$C_2 s\Theta_2(s) + \left(\frac{1}{R_2} + \frac{1}{R_1}\right)\Theta_2 = \frac{\Theta_0(s)}{R_2} + \frac{\Theta_1(s)}{R_1}$$

Substituindo $\Theta_2(s)$

$$\left[C_2 s(R_1C_1s+1) + R_1 + R_2(R_1C_1s+1)\right]\Theta_1(s) = \frac{\Theta_0(s)}{R_2} + \frac{\Theta_1(s)}{R_1}$$

$\frac{\Theta_1(s)}{\Theta_0(s)}$, obtemos

$$\frac{\Theta_1(s)}{\Theta_0(s)} = \frac{1}{R_1C_1R_2C_2s^2 + (R_1C_1 + R_2C_2 + R_2C_1)s + 1}$$

Substituindo temos:

$R_1C_1 = 2s$, $R_2C_2 = 30s$

R_2C_1: como $C_1 = \frac{C_2}{5}$

$$R_2C_1 = R_2\left(\frac{C_2}{5}\right) = \frac{R_2C_2}{5} = \frac{30}{5} = 6s.$$

$$(2)(30)s^2 + (2+30+6)s + 1 = 60s^2 + 38s + 1$$

$$60s^2 + 38s + 1 = 0$$

$$s = \frac{-38 \pm \sqrt{38^2 - 4\cdot 60\cdot 1}}{2\cdot 60} = \frac{-38 \pm \sqrt{1444 - 240}}{120} =$$

$$\frac{-38 \pm 34.7}{120}$$

$$s_1 = \frac{-38 + 34.7}{120} \approx -0,0275 \rightarrow \tau_2 = \frac{1}{|s_1|} \approx \underline{36,365}$$

$$s_2 = \frac{-38 - 34.7}{120} \approx 0.606 \rightarrow \tau_1 = \frac{1}{|s_2|} \approx \underline{1.655}$$

$$(1.65s+1)(36.36s+1) \approx \tau_1 = 1.655$$

$$\underline{\tau_2 = 36,365}$$

Representação de espaço de estado B.4.12

Termopar

$$C_1 \frac{d\theta_1}{dt} = \frac{\theta_2 - \theta_1}{R_1} \rightarrow \frac{d\theta_1}{dt} \quad \frac{1}{R_1C_1}(\theta_2 - \theta_1)$$

$R_1C_1 = 2s$: $$\frac{d\theta_1}{dt} = \frac{1}{2}\theta_1 + \frac{1}{2}\theta_2 . \quad [1]$$

$$C_2 \cdot \frac{d\theta_2}{dt} = \frac{\theta_0 - \theta_2}{R_2} - \frac{\theta_2 - \theta_1}{R_1}$$

$R_2C_2 = 30s$ e $R_2C_1 = 6s$ ($C_1 = C_2$ (s).

$$\frac{d\theta_2}{dt} = \frac{1}{30}\theta_0 - \frac{1}{30}\theta_2 - \frac{1}{6}\theta_2 + \frac{1}{6}\theta_1 .$$

$$\frac{d\theta_2}{dt} = \frac{1}{6}\theta_1 - \frac{1}{5}\theta_2 + \frac{1}{30}\theta_0 \cdot \quad [2]$$

Definição de variáveis de estado

$$x_1 = \theta_1, x_2 = \theta_2$$

$$\begin{cases} \dot{x}_1 = -\frac{1}{2}x_1 + \frac{1}{2}x_2, \\ \dot{x}_2 = \frac{1}{6}x_1 - \frac{1}{5}x_2 + \frac{1}{30}u, \text{ (onde } u = \theta_0) \end{cases}$$

forma matricial

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{6} & -\frac{1}{5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{30} \end{bmatrix} u .$$

$$y = [1 \ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\dot{x} = Ax + Bu . \quad y = Cx$$

$$A = \begin{bmatrix} -0.5 & 0.5 \\ 0.1667 & -0.2 \end{bmatrix} ; \quad B = \begin{bmatrix} 0 \\ 0.0333 \end{bmatrix} . \quad C = [1 0] .$$

Polos: τ_1 e τ_2

$\det[sI - A] = 0$:

$s^2 + 0.7s + 0.0167 = 0 \rightarrow s \approx -0.0275, -0{,}606,$

$\tau_1 = 1{,}65s$ e $\tau_2 = 36.36s.$

$$\dot{x} = \begin{bmatrix} -0.5 & 0.5 \\ 0.1667 & -0.2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.0333 \end{bmatrix} u,$$

$$y = [1 \ 0]\, x,$$

Onde $x = [\theta_1, \theta_2]^T$, $u = \theta_0$ e $y = \theta_1$.

Representação em python:

```
import numpy as np
from scipy import signal

# Dados do problema
tau1 = 2.0     # Constante de tempo do termopar (R1*C1) em
segundos
tau2 = 30.0    # Constante de tempo do poço térmico (R2*C2) em
segundos
m1 = 8.0       # Massa do termopar em gramas
m2 = 40.0      # Massa do poço térmico em gramas

# Passo 1: Relação entre capacitâncias térmicas (C) e massas
(m)
# Como os calores específicos são iguais, C1/C2 = m1/m2
C1_C2_ratio = m1 / m2
print(f"Passo 1: Razão C1/C2 = m1/m2 = {C1_C2_ratio:.3f}")

# Passo 2: Cálculo de R2*C1
R2C1 = tau2 * C1_C2_ratio
print(f"\nPasso 2: R2*C1 = R2*C2 * (C1/C2) = {tau2} *
{C1_C2_ratio:.3f} = {R2C1:.3f} s")

# Passo 3: Equação característica do sistema
# Denominador da função de transferência: R1C1*R2C2*s² +
(R1C1 + R2C2 + R2C1)*s + 1
a = tau1 * tau2
b = tau1 + tau2 + R2C1
c = 1

print(f"\nPasso 3: Equação característica:")
print(f"{a:.1f}s² + {b:.1f}s + {c} = 0")

# Passo 4: Resolver a equação quadrática para encontrar as
raízes
roots = np.roots([a, b, c])
print(f"\nPasso 4: Raízes da equação característica:")
print(f"s1 = {roots[0]:.4f} 1/s")
```

```
print(f"s2 = {roots[1]:.4f} 1/s")

# Passo 5: Calcular as constantes de tempo (T = -1/s para s <
0)
T1 = -1/roots[1]
T2 = -1/roots[0]
print(f"\nPasso 5: Constantes de tempo do sistema:")
print(f"T1 = {T1:.3f} s")
print(f"T2 = {T2:.3f} s")

# Passo 6: Fatoração do polinômio
# Verificação usando a função transferência
num = [1]
den = [a, b, c]
system = signal.TransferFunction(num, den)

# Resultados finais
print("\nResultado Final:")
print(f"As constantes de tempo do sistema combinado são:")
print(f"T1 = {T1:.3f} s (resposta mais rápida - termopar)")
print(f"T2 = {T2:.3f} s (resposta mais lenta - poço
térmico)")

# Verificação adicional
print("\nVerificação:")
print("Polos do sistema:", roots)
print("Que correspondem às constantes de tempo T1 e T2
calculadas")
```

Resultado:

```
Passo 1: Razão C1/C2 = m1/m2 = 0.200

Passo 2: R2*C1 = R2*C2 * (C1/C2) = 30.0 * 0.200 = 6.000 s

Passo 3: Equação característica:
60.0s² + 38.0s + 1 = 0

Passo 4: Raízes da equação característica:
s1 = -0.6058 1/s
s2 = -0.0275 1/s

Passo 5: Constantes de tempo do sistema:
T1 = 36.349 s
T2 = 1.651 s

Resultado Final:
As constantes de tempo do sistema combinado são:
T1 = 36.349 s (resposta mais rápida - termopar)
T2 = 1.651 s (resposta mais lenta - poço térmico)

Verificação:
Polos do sistema: [-0.60582253 -0.02751081]
Que correspondem às constantes de tempo T1 e T2 calculadas
```

Podemos verificar que o cálculo encontrados que eu realizei manuscrito é aproximado ao encontrado no python, o que podemos ter uma pequena discrepância para **T1 = 1.65s e T2 = 36.365s** no python obtivemos para **T1 = 1.651s e T2 = 36.349s.**

Representação de espaços de estados:

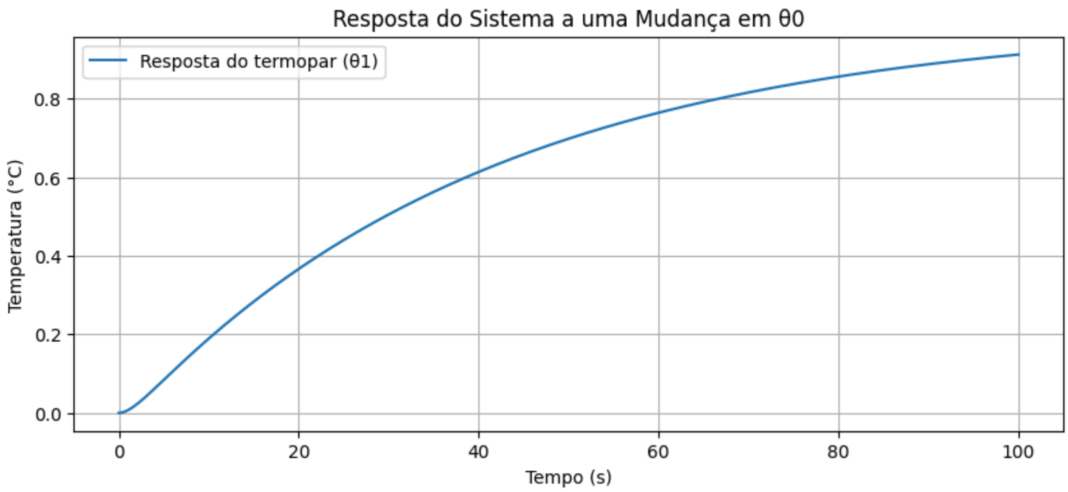
```
import numpy as np
from scipy import signal

# Parâmetros do sistema
tau1 = 2.0     # R1*C1 = 2 s
tau2 = 30.0    # R2*C2 = 30 s
m1, m2 = 8.0, 40.0  # Massas (g)
C1_C2 = m1 / m2      # C1/C2 = 0.2
R2C1 = tau2 * C1_C2  # R2*C1 = 6 s

# Matrizes do espaço de estados
A = np.array([
    [-1/tau1, 1/tau1],
    [1/(R2C1), -1/tau2 - 1/R2C1]
])
B = np.array([[0], [1/tau2]])
C = np.array([[1, 0]])  # Saída = θ1
D = np.array([[0]])

# 1. Cálculo dos polos (autovalores de A)
polos = np.linalg.eigvals(A)
constantes_tempo = -1/np.real(polos)

# 2. Saída matricial formatada
print("="*50)
print("Representação em Espaço de Estados")
print("="*50)
print("\nMatriz A (Dinâmica do Sistema):")
print(A)
print("\nMatriz B (Entrada):")
print(B)
```

```
print("\nMatriz C (Saída):")
print(C)
print("\nMatriz D (Transmissão Direta):")
print(D)

print("\n" + "="*50)
print("Análise do Sistema")
print("="*50)
print("\nPolos do Sistema:")
print(polos)
print("\nConstantes de Tempo Correspondentes:")
print(constantes_tempo)

# 3. Resposta ao degrau (opcional)
t = np.linspace(0, 100, 1000)
u = np.ones_like(t)   # Entrada degrau θ0 = 1
sys = signal.lti(A, B, C, D)
_, y, _ = signal.lsim(sys, U=u, T=t)

# 4. Plot (opcional)
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 4))
plt.plot(t, y, label='Resposta do termopar (θ1)')
plt.xlabel('Tempo (s)')
plt.ylabel('Temperatura (°C)')
plt.title('Resposta do Sistema a uma Mudança em θ0')
plt.grid(True)
plt.legend()
plt.show()
```

Resultados:

```
==================================================
Representação em Espaço de Estados
==================================================

Matriz A (Dinâmica do Sistema):
[[-0.5         0.5       ]
 [ 0.16666667 -0.2       ]]

Matriz B (Entrada):
[[0.        ]
 [0.03333333]]

Matriz C (Saída):
[[1 0]]

Matriz D (Transmissão Direta):
[[0]]

==================================================
Análise do Sistema
==================================================

Polos do Sistema:
[-0.67532035 -0.02467965]

Constantes de Tempo Correspondentes:
[ 1.4807787 40.5192213]
```

Podemos confirmar is[...]tema termopar.



O gráfico apresenta a resposta dinâmica do sistema térmico a uma mudança na entrada θ_0, indicando a evolução da temperatura (θ_1) registrada pelo termopar ao longo do tempo. Observa-se que a temperatura aumenta de forma contínua e suave, partindo de 0 °C e aproximando-se de um valor de regime próximo a 0,9 °C após cerca de 100 segundos. Esse comportamento sugere que o sistema possui uma dinâmica de primeira ordem com uma constante de tempo moderada, apresentando uma resposta estável e sem oscilações, típica de sistemas térmicos onde a dissipação de calor ocorre de forma gradual. O gráfico também evidencia que o sistema atinge aproximadamente 90% de sua resposta final em torno de 90 segundos, o que permite uma estimativa do tempo de estabilização.