

## Resolução da Lista de Exercícios

**Francisco Davi Belo Rodrigues**

28 de outubro de 2025

### 1 Exercício 1

#### Enunciado e dados

Consideram-se dois tanques cilíndricos interligados em série. O tanque 1 recebe uma alimentação constante e descarrega no tanque 2, que por sua vez escoar para o ambiente. As vazões de saída de cada tanque dependem do nível interno segundo a relação empírica  $Q_i = k_i \sqrt{h_i}$ . Os parâmetros fornecidos são resumidos na Tabela 1.

Parâmetro	Valor
Vazão de alimentação $Q_0$	$20 \text{ m}^3 \text{ h}^{-1}$
Diâmetro do tanque 1 $D_1$	4 m
Diâmetro do tanque 2 $D_2$	3 m
Constante da válvula 1 $k_1$	$14 \text{ m}^{2.5} \text{ h}^{-1}$
Constante da válvula 2 $k_2$	$12 \text{ m}^{2.5} \text{ h}^{-1}$
Nível inicial no tanque 1 $h_1(0)$	3 m
Nível inicial no tanque 2 $h_2(0)$	2 m

Tabela 1: Dados operacionais da Questão 1.

#### Formulação do modelo

O modelo dinâmico é obtido a partir dos balanços de massa (ou volume, dado que a densidade é constante) em cada tanque. Para um volume de controle genérico com densidade constante  $\rho$ , tem-se

$$\frac{d(\rho V)}{dt} = \rho Q_{\text{in}} - \rho Q_{\text{out}},$$

o que conduz ao balanço volumétrico

$$\frac{dV}{dt} = Q_{\text{in}} - Q_{\text{out}}.$$

No tanque 1, o volume contido é  $V_1 = A_1 h_1$ , com  $A_1$  indicado na Eq. (1). O balanço volumétrico resulta em

$$A_1 \frac{dh_1}{dt} = Q_0 - Q_1,$$

em que  $Q_0$  é a vazão de alimentação e  $Q_1$  a vazão de saída do tanque 1. De modo análogo, para o tanque 2 obtém-se  $V_2 = A_2 h_2$  e

$$A_2 \frac{dh_2}{dt} = Q_1 - Q_2,$$

com  $Q_1$  proveniente do tanque 1 e  $Q_2$  a descarga para o ambiente.

As vazões de saída seguem a correlação empírica das válvulas e as áreas dos tanques são calculadas pela geometria cilíndrica. Mantendo  $t$  em horas, têm-se as equações numeradas finais:

$$A_i = \frac{\pi D_i^2}{4}, \quad i = 1, 2, \quad (1)$$

$$Q_1 = k_1 \sqrt{h_1}, \quad (2)$$

$$Q_2 = k_2 \sqrt{h_2}, \quad (3)$$

$$A_1 \frac{dh_1}{dt} = Q_0 - Q_1, \quad (4)$$

$$A_2 \frac{dh_2}{dt} = Q_1 - Q_2, \quad (5)$$

com condições iniciais  $h_1(0) = 3$  m e  $h_2(0) = 2$  m. Este conjunto de equações está pronto para utilização em ambientes de simulação como o EMSO, onde os parâmetros podem ser definidos separadamente sem substituição numérica antecipada.

## Resolução numérica

O sistema diferencial foi integrado em  $0 \leq t \leq 20$  h empregando o método Runge–Kutta de quarta/quinta ordem adaptativo (`solve_ivp` do SciPy) com passo máximo equivalente a 10 s após conversão interna de unidades no script de apoio. A implementação registra também as trajetórias discretizadas  $(t, h_1, h_2)$  em arquivo auxiliar para rastreabilidade.

```

1 import numpy as np
2 from math import pi
3 from scipy.integrate import solve_ivp
4 import matplotlib.pyplot as plt
5
6 Q0 = 20.0 # m^3/h
7 D1 = 4.0 # m
8 D2 = 3.0 # m
9 k1 = 14.0 # m^{2.5}/h
10 k2 = 12.0 # m^{2.5}/h
11 h1_0 = 3.0 # m
12 h2_0 = 2.0 # m
13
14 A1 = pi * (D1 ** 2) / 4.0
15 A2 = pi * (D2 ** 2) / 4.0
16
17 T_sim = 20.0
18
19 def model(t, y):
20     h1, h2 = y
21     q1 = k1 * np.sqrt(max(h1, 0.0))
22     q2 = k2 * np.sqrt(max(h2, 0.0))
23     dh1dt = (Q0 - q1) / A1
24     dh2dt = (q1 - q2) / A2
25     return [dh1dt, dh2dt]
26
27 sol = solve_ivp(
28     fun=model,
29     t_span=(0.0, T_sim),
30     y0=[h1_0, h2_0],
31     max_step=0.01,
32     dense_output=True
33 )
34

```

```

35 t_hours = np.linspace(0.0, T_sim, 100)
36 h1 = sol.sol(t_hours)[0]
37 h2 = sol.sol(t_hours)[1]
38
39 plt.figure(figsize=(6, 4))
40 plt.plot(t_hours, h1, label="h1 (m)")
41 plt.plot(t_hours, h2, label="h2 (m)")
42 plt.xlabel("Tempo (h)")
43 plt.ylabel("Nível (m)")
44 plt.legend()
45 plt.grid(True)
46 plt.tight_layout()
47 plt.savefig("figuras/questao1_niveis.png", dpi=300)
48
49 with open("figuras/questao1_niveis.dat", "w", encoding="utf-8") as f:
50     f.write("tempo_h h1_m h2_m\n")
51     for t, hv1, hv2 in zip(t_hours, h1, h2):
52         f.write(f"{t:.6f} {hv1:.6f} {hv2:.6f}\n")
53
54 print("h1 final:", h1[-1])
55 print("h2 final:", h2[-1])

```

Listing 1: Script Python utilizado para a integraçao numerica da Questao 1.

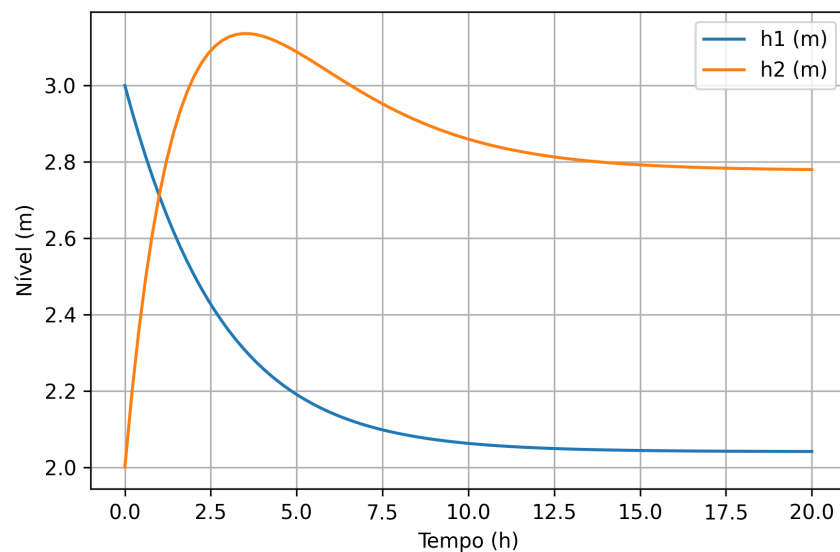


Figura 1: Perfis temporais simulados dos níveis  $h_1$  e  $h_2$  durante 20 horas.

## 2 Exercício 2

### Enunciado e dados

Um tanque de mistura perfeitamente agitado recebe duas correntes de entrada e tem uma corrente de saída. As correntes contêm três componentes (A, B e C) em diferentes proporções. A vazão de saída depende do nível de líquido no tanque segundo a relação  $F_3 = \rho_3 k \sqrt{h}$ , onde  $\rho_3$  é a densidade da corrente de saída. Os parâmetros fornecidos são resumidos na Tabela 2.

Parâmetro	Valor
Vazão da corrente 1 $F_1$	10 kg min <sup>-1</sup>
Fração mássica de A na corrente 1 $x_{A1}$	0,6
Fração mássica de B na corrente 1 $x_{B1}$	0,0
Fração mássica de C na corrente 1 $x_{C1}$	0,4
Vazão da corrente 2 $F_2$	8 kg min <sup>-1</sup>
Fração mássica de A na corrente 2 $x_{A2}$	0,0
Fração mássica de B na corrente 2 $x_{B2}$	0,7
Fração mássica de C na corrente 2 $x_{C2}$	0,3
Densidade do componente A $\rho_A$	1200 kg m <sup>-3</sup>
Densidade do componente B $\rho_B$	1400 kg m <sup>-3</sup>
Densidade do componente C $\rho_C$	1000 kg m <sup>-3</sup>
Área da seção transversal $A$	0,2 m <sup>2</sup>
Parâmetro da válvula $k$	0,02 m <sup>2.5</sup> min <sup>-1</sup>
Massa inicial de A $m_{A0}$	20 kg
Massa inicial de B $m_{B0}$	20 kg
Massa inicial de C $m_{C0}$	40 kg

Tabela 2: Dados operacionais da Questão 2.

## Formulação do modelo

O modelo dinâmico é obtido a partir dos balanços de massa individuais para cada componente. Para um componente genérico  $i$  (A, B ou C), o balanço mássico no volume de controle resulta em

$$\frac{dm_i}{dt} = F_1 x_{i1} + F_2 x_{i2} - F_3 x_{i3},$$

onde  $m_i$  é a massa do componente  $i$  no tanque,  $x_{i1}$  e  $x_{i2}$  são as frações mássicas nas correntes de entrada, e  $x_{i3}$  é a fração mássica na corrente de saída.

Como o tanque é perfeitamente agitado, a composição de saída é igual à composição interna. Assim, a fração mássica de cada componente na corrente de saída é dada por

$$x_{i3} = \frac{m_i}{m_{\text{total}}}, \quad m_{\text{total}} = m_A + m_B + m_C. \quad (6)$$

Para calcular a densidade da mistura na corrente de saída, considera-se o comportamento de uma mistura ideal, na qual os volumes dos componentes puros são aditivos. Essa hipótese é adequada para misturas líquidas sem interações moleculares significativas (mistura ideal). Assim, o volume total da mistura é dado pela soma dos volumes individuais:

$$V_{\text{total}} = V_A + V_B + V_C.$$

Como cada volume parcial pode ser expresso em termos da massa e densidade do componente ( $V_i = m_i/\rho_i$ ), obtém-se

$$V_{\text{total}} = \frac{m_A}{\rho_A} + \frac{m_B}{\rho_B} + \frac{m_C}{\rho_C}.$$

Por outro lado, o volume total relaciona-se com a massa total e a densidade da mistura por  $V_{\text{total}} = m_{\text{total}}/\rho_3$ . Igualando as duas expressões, tem-se

$$\frac{m_{\text{total}}}{\rho_3} = \frac{m_A}{\rho_A} + \frac{m_B}{\rho_B} + \frac{m_C}{\rho_C}.$$

Dividindo ambos os lados por  $m_{\text{total}}$  e utilizando a definição de fração mássica  $x_i = m_i/m_{\text{total}}$ , chega-se a

$$\frac{1}{\rho_3} = \frac{x_A}{\rho_A} + \frac{x_B}{\rho_B} + \frac{x_C}{\rho_C},$$

de onde resulta a densidade da mistura na corrente de saída:

$$\rho_3 = \frac{1}{\frac{x_A}{\rho_A} + \frac{x_B}{\rho_B} + \frac{x_C}{\rho_C}}. \quad (7)$$

O volume total no tanque é obtido pela soma dos volumes de cada componente:

$$V = \frac{m_A}{\rho_A} + \frac{m_B}{\rho_B} + \frac{m_C}{\rho_C}, \quad (8)$$

e o nível de líquido é

$$h = \frac{V}{A}. \quad (9)$$

A vazão de saída é calculada por

$$F_3 = \rho_3 k \sqrt{h}. \quad (10)$$

O sistema de equações diferenciais finais é

$$\frac{dm_A}{dt} = F_1 x_{A1} + F_2 x_{A2} - F_3 x_A, \quad (11)$$

$$\frac{dm_B}{dt} = F_1 x_{B1} + F_2 x_{B2} - F_3 x_B, \quad (12)$$

$$\frac{dm_C}{dt} = F_1 x_{C1} + F_2 x_{C2} - F_3 x_C, \quad (13)$$

com condições iniciais  $m_A(0) = 20$  kg,  $m_B(0) = 20$  kg e  $m_C(0) = 40$  kg.

## Resolução numérica

O sistema diferencial foi integrado em  $0 \leq t \leq 60$  min empregando o método Runge–Kutta de quarta/-quinta ordem adaptativo (`solve_ivp` do SciPy) com passo máximo equivalente a 0,01 min. A implementação registra também as trajetórias discretizadas ( $t, h, x_A, x_B, x_C$ ) em arquivo auxiliar para rastreabilidade.

```

1 import numpy as np
2 from scipy.integrate import solve_ivp
3 import matplotlib.pyplot as plt
4
5 F1 = 10.0
6 xA1 = 0.6
7 xB1 = 0.0
8 xC1 = 0.4
9 F2 = 8.0
10 xA2 = 0.0
11 xB2 = 0.7
12 xC2 = 0.3
13 rhoA = 1200.0
14 rhoB = 1400.0
15 rhoC = 1000.0
16 A = 0.2
17 k = 0.02
18 mA0 = 20.0
19 mB0 = 20.0
20 mC0 = 40.0
21
22 T_sim = 60.0
23
24 def model(t, y):
25     mA, mB, mC = y
26     m_total = mA + mB + mC

```

```

27     xA = mA / m_total
28     xB = mB / m_total
29     xC = mC / m_total
30     rho3 = 1.0 / (xA / rhoA + xB / rhoB + xC / rhoC)
31     V = mA / rhoA + mB / rhoB + mC / rhoC
32     h = V / A
33     F3 = rho3 * k * np.sqrt(max(h, 0.0))
34     dmAdt = F1 * xA1 + F2 * xA2 - F3 * xA
35     dmBdt = F1 * xB1 + F2 * xB2 - F3 * xB
36     dmCdt = F1 * xC1 + F2 * xC2 - F3 * xC
37     return [dmAdt, dmBdt, dmCdt]
38
39 sol = solve_ivp(
40     fun=model,
41     t_span=(0.0, T_sim),
42     y0=[mA0, mB0, mC0],
43     max_step=0.01,
44     dense_output=True
45 )
46
47 t_min = np.linspace(0.0, T_sim, 200)
48 mA = sol.sol(t_min)[0]
49 mB = sol.sol(t_min)[1]
50 mC = sol.sol(t_min)[2]
51
52 m_total = mA + mB + mC
53 xA = mA / m_total
54 xB = mB / m_total
55 xC = mC / m_total
56 V = mA / rhoA + mB / rhoB + mC / rhoC
57 h = V / A
58
59 fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(6, 6))
60
61 ax1.plot(t_min, h, label="h (m)", color='blue')
62 ax1.set_xlabel("Tempo (min)")
63 ax1.set_ylabel("Nível (m)")
64 ax1.legend()
65 ax1.grid(True)
66
67 ax2.plot(t_min, xA, label="xA", color='red')
68 ax2.plot(t_min, xB, label="xB", color='green')
69 ax2.plot(t_min, xC, label="xC", color='purple')
70 ax2.set_xlabel("Tempo (min)")
71 ax2.set_ylabel("Fração mássica")
72 ax2.legend()
73 ax2.grid(True)
74
75 plt.tight_layout()
76 plt.savefig("figuras/questao2_tanque.png", dpi=300)
77
78 with open("figuras/questao2_tanque.dat", "w", encoding="utf-8") as f:
79     f.write("tempo_min h_m xA xB xC\n")
80     for t, hval, xa, xb, xc in zip(t_min, h, xA, xB, xC):
81         f.write(f"{t:.6f} {hval:.6f} {xa:.6f} {xb:.6f} {xc:.6f}\n")
82
83 print("h final:", h[-1])
84 print("xA final:", xA[-1])
85 print("xB final:", xB[-1])

```

```
86 print("xC final:", xC[-1])
```

Listing 2: Script Python utilizado para a integraçao numerica da Questao 2.

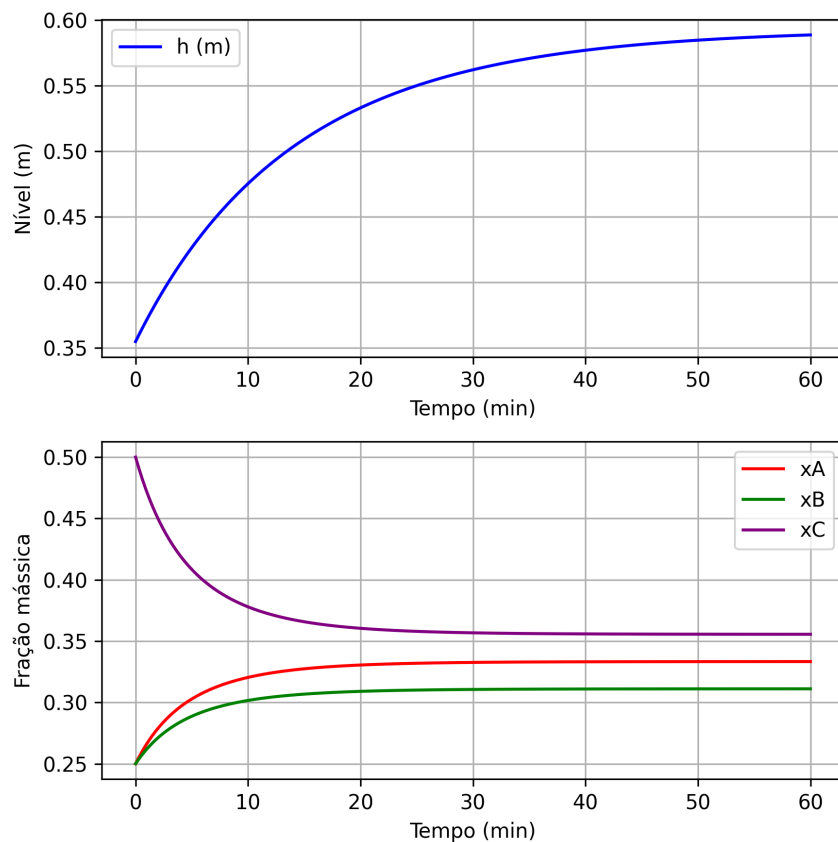
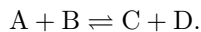


Figura 2: Perfis temporais simulados do nível  $h$  e das frações mássicas  $x_A$ ,  $x_B$  e  $x_C$  durante 60 minutos.

### 3 Exercício 3

#### Enunciado e dados

Um reator do tipo BSTR (batelada) é utilizado para executar a seguinte reação reversível:



As taxas  $[\text{mol} \cdot \text{L}^{-1} \cdot \text{min}^{-1}]$  das reações direta e inversa podem ser estimadas segundo as seguintes equações:

$$r_d = k_d C_A^{1,1} C_B^{1,4}, \quad (14)$$

$$r_i = k_i C_C^{1,31} C_D^{1,2}, \quad (15)$$

em que  $C_A$ ,  $C_B$ ,  $C_C$  e  $C_D$  são as concentrações  $[\text{mol} \cdot \text{L}^{-1}]$ ,  $T$  é a temperatura  $[\text{K}]$ , e as constantes cinéticas  $k_d$  e  $k_i$  são dadas por

$$k_d = 50\,000 \exp\left(-\frac{4\,000}{T}\right), \quad (16)$$

$$k_i = 30\,000 \exp\left(-\frac{5\,000}{T}\right). \quad (17)$$

Inicialmente são colocados no reator somente os reagentes A e B, com concentrações de  $0,5 \text{ mol L}^{-1}$  e  $0,8 \text{ mol L}^{-1}$ , respectivamente. O reator apresenta um sistema de ajuste de temperatura que inicialmente está fixado em 400 K. Essa temperatura é mantida constante durante os primeiros 5 minutos de batelada. Passado esse tempo, inicia-se uma rampa de diminuição da temperatura de forma que ao final da batelada a temperatura alcance o valor de 350 K.

## Formulação do modelo

O modelo dinâmico é obtido a partir dos balanços de massa para cada componente no reator em batelada. Para um reator em batelada, não há termos de entrada ou saída de massa, logo o balanço de massa para cada componente é dado por

$$\frac{dC_i}{dt} = r_i,$$

onde  $C_i$  é a concentração do componente  $i$  e  $r_i$  é a taxa líquida de geração/consumo do respectivo componente.

Para a reação reversível  $A + B \rightleftharpoons C + D$ , as velocidades das reações direta e inversa são dadas em (14) e (15), e a taxa líquida de reação é

$$r_{\text{net}} = r_d - r_i, \quad (18)$$

onde  $r_d$  é a velocidade da reação direta e  $r_i$  a da reação inversa.

Os balanços de massa para cada componente seguem a estequiometria da reação:

$$\frac{dC_A}{dt} = -r_{\text{net}}, \quad (19)$$

$$\frac{dC_B}{dt} = -r_{\text{net}}, \quad (20)$$

$$\frac{dC_C}{dt} = r_{\text{net}}, \quad (21)$$

$$\frac{dC_D}{dt} = r_{\text{net}}, \quad (22)$$

com condições iniciais

$$C_A(0) = 0,5 \text{ mol L}^{-1}, \quad C_B(0) = 0,8 \text{ mol L}^{-1}, \quad C_C(0) = 0 \text{ mol L}^{-1}, \quad C_D(0) = 0 \text{ mol L}^{-1}.$$

A temperatura do reator segue o perfil de operação especificado:

$$T(t) = \begin{cases} 400 \text{ K}, & \text{se } t \leq 5 \text{ min}, \\ 400 - \frac{50}{10}(t - 5) \text{ K}, & \text{se } t > 5 \text{ min}, \end{cases} \quad (23)$$

ou seja, permanece em 400 K nos primeiros 5 min e decresce linearmente até 350 K ao final da batelada (15 min).

A conversão do reagente A é definida como

$$X_A = \frac{C_{A0} - C_A}{C_{A0}}, \quad (24)$$

onde  $C_{A0} = 0,5 \text{ mol L}^{-1}$  é a concentração inicial de A.

## Resolução numérica

O sistema de equações ordinais foi integrado em  $0 \leq t \leq 15$  min empregando o método Runge-Kutta de quarta/quinta ordem adaptativo (`solve_ivp` do SciPy) com passo máximo equivalente a 0,01 min. A implementação registra também as trajetórias discretizadas ( $t, T, C_A, C_B, C_C, C_D, X_A$ ) em arquivo auxiliar para rastreabilidade.



```

1 import numpy as np
2 from scipy.integrate import solve_ivp
3 import matplotlib.pyplot as plt
4
5 CAO = 0.5
6 CBO = 0.8
7 CCO = 0.0
8 CDO = 0.0
9
10 T_sim = 15.0
11
12 def model(t, y):
13     CA, CB, CC, CD = y
14     if t <= 5.0:
15         T = 400.0
16     else:
17         T = 400.0 - (400.0 - 350.0) * (t - 5.0) / (T_sim - 5.0)
18     # constantes de velocidade direta e inversa atualizadas
19     kd = 50000.0 * np.exp(-4000.0 / T)
20     ki = 30000.0 * np.exp(-5000.0 / T)
21     # taxas com os expoentes especificados
22     rd = kd * (CA ** 1.1) * (CB ** 1.4)
23     ri = ki * (CC ** 1.31) * (CD ** 1.2)
24     r_net = rd - ri
25     dCAdt = -r_net
26     dCBdt = -r_net
27     dCCdt = r_net
28     dCDdt = r_net
29     return [dCAdt, dCBdt, dCCdt, dCDdt]
30
31 sol = solve_ivp(
32     fun=model,
33     t_span=(0.0, T_sim),
34     y0=[CAO, CBO, CCO, CDO],
35     max_step=0.01,
36     dense_output=True
37 )
38
39 t_min = np.linspace(0.0, T_sim, 300)
40 CA = sol.sol(t_min)[0]
41 CB = sol.sol(t_min)[1]
42 CC = sol.sol(t_min)[2]
43 CD = sol.sol(t_min)[3]
44
45 T = np.where(t_min <= 5.0, 400.0, 400.0 - (400.0 - 350.0) * (t_min - 5.0) / (T_sim - 5.0))
46 XA = (CAO - CA) / CAO
47
48 fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(6, 8))
49
50 ax1.plot(t_min, T, label="T (K)", color='red')
51 ax1.set_xlabel("Tempo (min)")
52 ax1.set_ylabel("Temperatura (K)")
53 ax1.legend()
54 ax1.grid(True)
55
56 ax2.plot(t_min, CA, label="CA", color='blue')
57 ax2.plot(t_min, CB, label="CB", color='green')
58 ax2.plot(t_min, CC, label="CC", color='orange')

```

```

59 ax2.plot(t_min, CD, label="CD", color='purple')
60 ax2.set_xlabel("Tempo (min)")
61 ax2.set_ylabel("Concentração (mol/L)")
62 ax2.legend()
63 ax2.grid(True)
64
65 ax3.plot(t_min, XA, label="XA", color='black')
66 ax3.set_xlabel("Tempo (min)")
67 ax3.set_ylabel("Conversão de A")
68 ax3.legend()
69 ax3.grid(True)
70
71 plt.tight_layout()
72 plt.savefig("figuras/questao3_reator.png", dpi=300)
73
74 with open("figuras/questao3_reator.dat", "w", encoding="utf-8") as f:
75     f.write("tempo_min T_K CA CB CC CD XA\n")
76     for t, temp, ca, cb, cc, cd, xa in zip(t_min, T, CA, CB, CC, CD, XA):
77         f.write(f"{t:.6f} {temp:.6f} {ca:.6f} {cb:.6f} {cc:.6f} {cd:.6f} {xa:.6f}\n")
78
79 print("T final:", T[-1])
80 print("CA final:", CA[-1])
81 print("CB final:", CB[-1])
82 print("CC final:", CC[-1])
83 print("CD final:", CD[-1])
84 print("XA final:", XA[-1])

```

Listing 3: Script Python utilizado para a integração numérica da Questão 3.

## Referências

[1] Autor, *Título do Livro ou Artigo*, Editora, Ano.

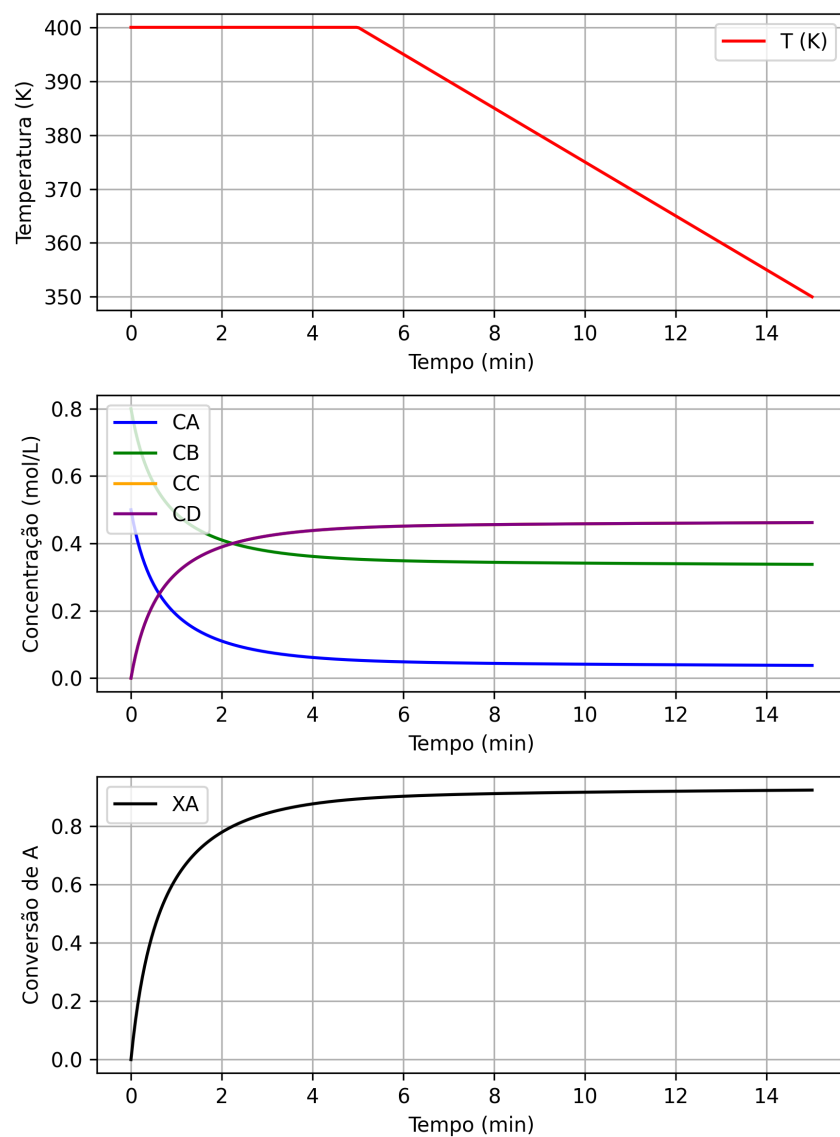


Figura 3: Perfis temporais simulados da temperatura, concentrações e conversão do reagente A durante 15 minutos de batelada.