Lista de Exercícios 1
Modelagem e Simulação de Processos COPPE PEQ COQ-791

Professor: Príamo Albuquerque Melo Junior
Aluno: Diego Telles Fernandes

11/12/2020

1 Sobre o modelo M6 do CSTR

Todos os algoritmos desenvolvidos para resolução desta lista estão disponíveis no repositório do GitHub e em anexo deste texto.

$$V\frac{dC}{dt} = q(C_f - C) - Vk(T)C$$
$$VCp\frac{dT}{dt} = qCp(T_f - T) + (-\Delta H)Vk(T)C - hA(T - T_C)$$

a) Objetivo: Calcular as curvas do Calor Gerado (Qg) e Removido (Qr) em função da Temperatura de forma a observar os estados estacionários do sistema quando Qg = Qr.

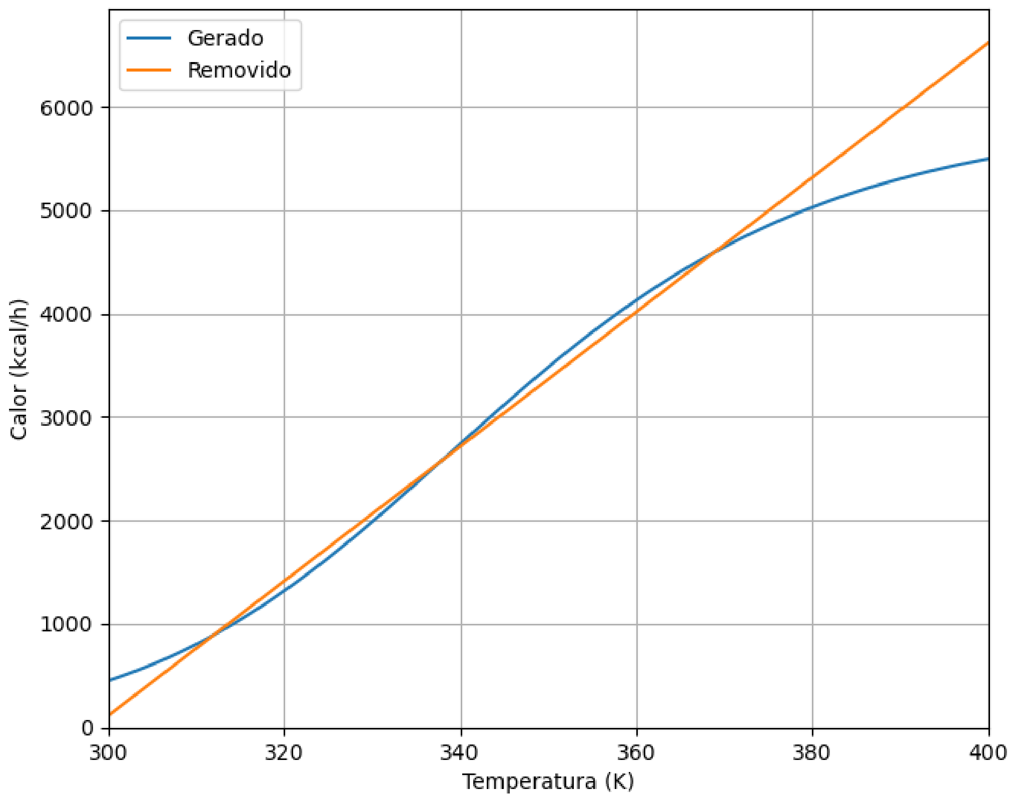
A partir do balando de massa podemos calcular a concentração no estado estacionário C_{EE} para uma dada Temperatura T_{EE} através de:

$$C_{EE} = \frac{qC_f}{q + Vk_0 exp\left(\frac{E}{RT_{EE}}\right)}$$

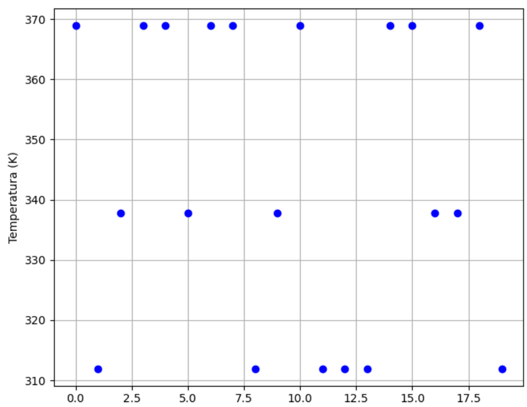
A partir do balaço de energia é possível calcular o Calor Gerado (Qg) e Removido (Qr) por:

$$Q_R = (qCp + Ah)T_{EE} - (qCpT_f + AhT_C)$$
$$Q_G = (-\Delta H)Vk_0 exp\left(\frac{E}{RT_{EE}}\right)$$

O sistema se encontra no estado estacionário (dC/dt = dT/dt = 0) e portanto a partir do balanço de massa é possível calcular a Concentração do Estado estacionário para uma dada temperatura. A partir do balanço de energia, também no estado estacionário, é possível calcular o calor gerado e removido do sistema. Como se pode observar acima, as curvas de Calor Removido e Calor Gerado calculado para um conjunto Temperaturas e Concentrações se tocam em três pontos o que indica três possíveis estados estacionários do sistema.



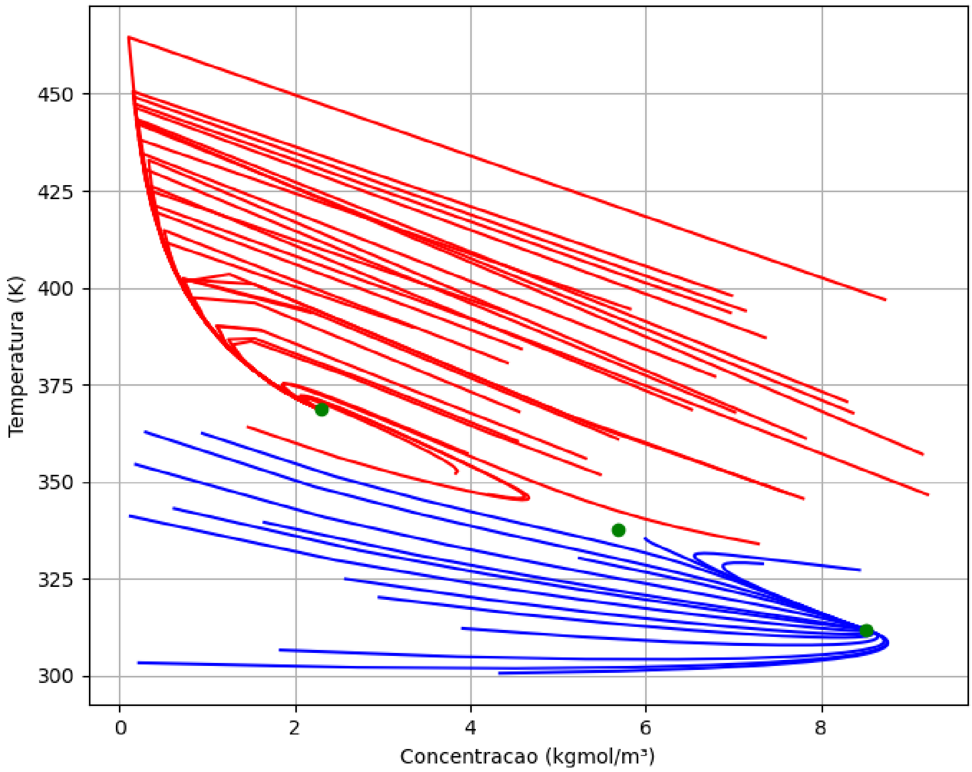
b) Objetivo: Encontrar as raízes do sistema de equações no Estado Estacionário para diversos chutes iniciais de Concentração e Temperatura de modo a observar que o sistema possui 3 possíveis Estados Estacionário e quais estes valores de Temperatura e Concentração.



Como se pode observar, para diversos chutes aleatórios de Temperatura e Concentração o algorítimo de resolução do sistema no estado estacionário sempre encontra um dos três pontos. São eles:

T(K)	C(kgmol/m³)
312	8,5
338	5,7
369	2,3

c) Objetivo: Resolver o sistema de forma dinâmica para valores aleatórios de Concentração e Temperaturas e inciais e apresentar um plano de fases que indique o caminho do sistema até atingir o Estado Estacionário.



Como se pode observar parece haver uma fronteira ao qual valores abaixo dela o sistema sempre converge para o estado estacionário de menor temperatura (curvas azuis) e acima dela o sistema sempre converge para o estado estacionário de maior temperatura (curvas vermelhas). O estado estacionário de temperatura intermediária não é estável e nenhuma curva converge para este ponto.

d) Objetivo: construir 2 o diagramas de soluções estacionárias do reator. O primeiro tendo como parâmetro a Temperatura de entrada da camisa (T_C) e o segundo a Temperatura de entrada da carga (T_f) para vários valores de q/v.

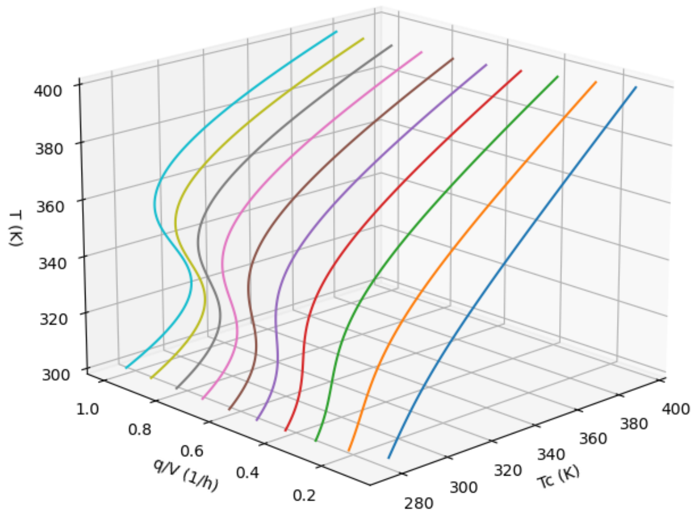
No estado estacionário podemos calcular a Temperatura da camisa (T_C) para um dado par T_{EE} e C_{EE} da seguinte forma:

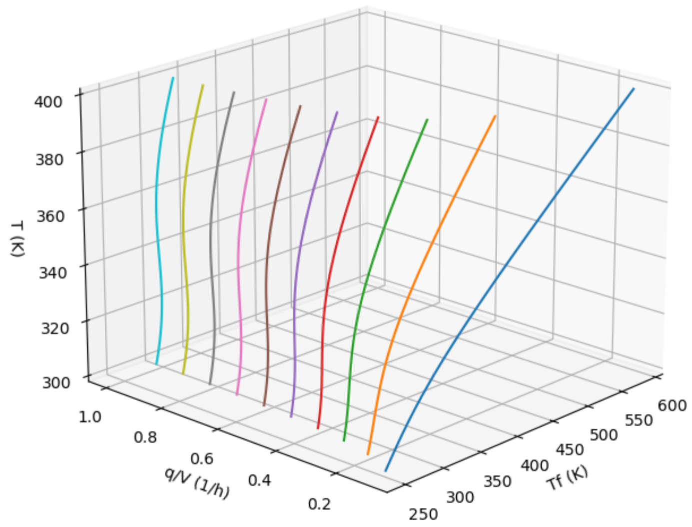
$$T_C = T_{EE} + \frac{qCp(T_{EE} - T_f) - (-\Delta H)Vk_0 exp\left(\frac{E}{RT_{EE}}\right)C_{EE}}{Ah}$$

De maneira análoga, quando T_C é fixo podemos calcular a Temperatura de alimentação que originaria aquele estado estacionário como sendo:

$$T_f = T_{EE} + \frac{hA(T_{EE} - T_C) - (-\Delta H)Vk_0 exp\left(\frac{E}{RT_{EE}}\right)C_{EE}}{qCp}$$

Como se pode observar a medida que os valores de q/v aumentam a curva adquire maior ”sinuosidade”o indica uma presença de mais de 1 estado estacionário do sistema.





2 CÓDIGO FONTE

a) Algoritmo:

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

# Parâmetros do modelo
q = 0.1             # m3/h
V = 0.1             # m3
k0 = 9703*3600      # 1/h
DH = 5960           # kcal/kgmol
E = 11843           # kcal/kgmol
Cp = 500            # kcal/m3/K
hA = 15             # kcal/h/K
R = 1.987           # kcal/kgmol/K
Tc = 298.5          # K
Tf = 298.15         # K
Cf = 10             # kgmol/m3


# Objetivo: Calcular as curvas do Calor Gerado (Qg) e Remoovido (Qr) em
# função da Temperatura de forma a observar os 3 estados estacionários
# do sistma quando Qg = Qr

# No estado estacionário dC/dt = dT/dt = 0 a Concentração do Estado estácionário
# pode ser calculada de forma analítica para uma dada Temperatura

# Gerando vetor de Temperaturas no Estado Estacionário no domínio dado (300 < TEE <
TEE = np.linspace(300,400,num = 101)            # K

# Calculando a Concentração do Estado estácionário
CEE = q*Cf/(q + V*k0*np.exp(-E/(R*TEE)))        # kgmol/m3

# Calculando Calor Gerado (Qg) e Remoovido (Qr) para TEE e CEE
Qg = DH*V*k0*np.exp(-E/(R*TEE))*CEE             # kcal/h
Qr = (q*Cp + hA)*TEE - (q*Cp*Tf + hA*Tc)        # kcal/h

# Plotando gráfico
plt.plot(TEE,Qg,label='Gerado')
plt.plot(TEE,Qr,label='Removido')
plt.xlabel('Temperatura (K)')
plt.ylabel('Calor (kcal/h)')
plt.xlim(left=300,right=400)
plt.ylim(bottom=0)
plt.legend()
plt.grid()
plt.show()
```

b) Algoritmo:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import fsolve

# Parâmetros do modelo
q = 0.1          # m3/h
V = 0.1          # m3
k0 = 9703*3600   # 1/h
DH = 5960        # kcal/kgmol
E = 11843        # kcal/kgmol
Cp = 500         # kcal/m3/K
hA = 15          # kcal/h/K
R = 1.987        # kcal/kgmol/K
Tc = 298.5       # K
Tf = 298.15      # K
Cf = 10          # kgmol/m3

# Objetivo: Encontrar as raízes do sistema de equações no Estado Estacionário
# para dirversos chutes iniciais de C e T de modo a observar que o sistema
# possui 3 possíveis Estados Estacionário

# Definindo função que representa o sistema de equações no estado estacionário
# ao qual as raízes serão calculadas. O parâmetro x dessa função é um vetor
# de duas posições cuja primeira represeta a temperatura e a segunda a concentração
def func(x):
    return [q*(Cf-x[1]) - V*k0*np.exp(-E/(R*x[0]))*x[1],
            q*Cp*(Tf - x[0]) + DH*V*k0*np.exp(-E/(R*x[0]))*x[1] - hA*(x[0] - Tc)]

# Definindo loop do numero de vezes que este sistema será resolvido
for i in range(20):

    # Definindo chute inicial de Tempeteratura de forma aleatória dentro do domínio
    # proposto de 300 a 400K
    TEE = np.array([np.random.rand()*100 + 300])    # K

    # Definindo chute inicial de Concentração de forma aleatória dentro do domínio
    # proposto de 0 a 10 kgmol/m3
    CEE = np.array([np.random.rand()*10])           # kgmol/m3

    # Encontrando as raízes do sistema a partir do chute inicial proposto
    root = fsolve(func, [TEE, CEE])

    # Criando e atualizando matriz de raízes encontradas
    # (número de análises nas linhas, raiz da Temperatura na 1ª Coluna e raiz da
    # Concentração na 2ª)
    if i == 0:
        Mroot = np.array([root])
    else:
        Mroot = np.vstack((Mroot, root))

# Mostra matriz com raízes no terminal
print("Temperatura (K)", "Concentração (kgmol/m3)")
print(Mroot)

# Plotando gráfico
plt.plot(Mroot[:,0], 'bo')
plt.ylabel('Temperatura (K)')
plt.grid()
plt.show()
```

c) Algoritmo:

```
# Importando Bibliotecas Python
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import solve_ivp

# Parâmetros do modelo
q = 0.1           # m3/h
V = 0.1           # m3
k0 = 9703*3600    # 1/h
DH = 5960         # kcal/kgmol
E = 11843         # kcal/kgmol
Cp = 500          # kcal/m3/K
hA = 15           # kcal/h/K
R = 1.987         # kcal/kgmol/K
Tc = 298.5        # K
Tf = 298.15       # K
Cf = 10           # kgmol/m3

# Objetivo: Resolver o sistema de forma dinâmica para valores aleatórios de
#Concentração e Tempertauras e inciais e apresentar um plano de fases que
# indique o caminho do sistema até atingir o Estado Estacionário

# Vetor das coordenadas do Estado Estaionário obtido nos exercícios anteriores
CEE = np.array([2.29464183, 5.6865874, 8.50357778])
TEE = np.array([368.88297641, 337.78144478, 311.95180991])

# Definindo função que representa o sistema de equações diferenciais.
# O parâmetro y dessa função é um vetor de duas posições cuja primeira
# represeta a concentração e a segunda a temperatura

def CSTR(t, y): # Declaração da função para uso dosolve_ivp

    # Decompondo o parâmetro y
    C, T = y

    # Calculando a velocidade específica da reação
    k = k0*np.exp(-E/(R*T))

    # Calculando os termos diferenciais no ponto (C, T)
    dCdt = q*(Cf - C)/V - k*C
    dTdt = q*(Tf - T)/V + DH*k*C/Cp - hA*(T - Tc)/(V*Cp)

    return dCdt, dTdt



# Definindo loop do numero de vezes que este sistema será resolvido
for i in range(50):

    # Definindo chute inicial de Tempeteratura de forma aleatória dentro
    # do domínio proposto de 300 a 400K
    T0 = np.random.rand()*100 + 300  # K

    # Definindo chute inicial de Concentração de forma aleatória dentro
    # do domínio proposto de 0 a 10 kgmol/m3
    C0 = np.random.rand()*10         # kgmol/m3

    # Definindo intervalo de tempo que o solver utilizará para resolver as EDOS
    t_span = np.array([0, 30])
```

```python
    # Definindo vetor de condições iniciais
    y0 = np.array([C0, T0])

    # Utilizando o pacote solve_ivp para resolução do sistema de equações
    # diferencias e armazenando o resultados nas variáveis C, T e t
    sol = solve_ivp(CSTR, t_span, y0, t_eval=np.linspace(0, 30, 101))
    C, T = sol.y
    t = sol.t

    # Identificando para qual estado estacionário o sistema convergiu
    DT = np.array([abs(T[len(C)-1] - TEE[0]), abs(T[len(C)-1] - TEE[1]),
        abs(T[len(C)-1] - TEE[2])])
    minidx = np.where(DT == np.min(DT))

    # Definindo esquema de cores em função do EE que o sistema convergiu
    # (azul para a menor, preto para a intermediária e vermelho para a maior T)
    if (minidx[0][0] == 0):
        corLabel = 'r-'
    elif (minidx[0][0] == 1):
        corLabel = 'k-'
    else:
        corLabel = 'b-'

    # Carregando esta curva no gráfico
    plt.plot(C, T, corLabel)

# Inserindo os pontos de estado estacionário em verde, configurando plotando
plt.plot(CEE, TEE, 'go')
plt.xlabel('Concentracao (kgmol/m³)')
plt.ylabel('Temperatura (K)')
plt.grid()
plt.show()
```

d) Algoritmo:

```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import fsolve

# Parâmetros do modelo
V = 0.1            # m3
k0 = 9703*3600     # 1/h
DH = 5960          # kcal/kgmol
E = 11843          # kcal/kgmol
Cp = 500           # kcal/m3/K
hA = 15            # kcal/h/K
R = 1.987          # kcal/kgmol/K
Tc_fixo = 298.5         # K
Tf_fixo = 298.15        # K
Cf = 10            # kgmol/m3


# Objetivo: construir 2 o diagramas de soluções estacionárias do reator.
# O primeiro tendo como parâmetro a Temperatura de entrada da camisa (Tc)
# e o segundo a Temperatura de entrada da carga

# Criando vetor de vazões de entrada do sistema
q = np.linspace(0.01, 0.1, 10)


##############################################
#### 1 - Tc Como parâmetro independente ####
##############################################

# Criando área de plotagem
fig = plt.figure(1)
ax = fig.gca(projection='3d')

# Definindo loop do numero de curvas de q/v
for idx in range(10):

    # Criando vetor de temperaturas do estado estacionário do sistema
    TEE = np.linspace(300, 400, 101)

    # Calculando vetor de velocidade específica da reação
    k = k0*np.exp(-E/(R*TEE))

    # Calculando vetor de concentrações do estado estacionário do sistema
    CEE = q[idx]*Cf/(q[idx] + V*k)

    # Calculando vetor de Tc para cada respectivo par TEE e CEE
    Tc = TEE + (q[idx]*Cp*(TEE-Tf_fixo) - DH*V*k*CEE)/hA

    # Criando vetor de q/v
    qplot = np.ones(101)*q[idx]/V

    # Carregando esta curva no gráfico 1
    ax.plot(Tc, qplot, TEE)

# configurando grafico 1
ax.set_xlabel('Tc (K)')
ax.set_ylabel('q/V (1/h)')
```

```
ax.set_zlabel('T (K)')

##############################################
#### 1 - Tf Como parâmetro independente ####
##############################################

# Criando área de plotagem
fig = plt.figure(2)
ax = fig.gca(projection='3d')

# Definindo loop do numero de curvas de q/v
for idx in range(10):

    # Criando vetor de temperaturas do estado estacionário do sistema
    TEE = np.linspace(300, 400, 101)

    # Calculando vetor de velocidade específica da reação
    k = k0*np.exp(-E/(R*TEE))

    # Calculando vetor de concentrações do estado estacionário do sistema
    CEE = q[idx]*Cf/(q[idx] + V*k)

    # Calculando vetor de Tf para cada respectivo par TEE e CEE
    Tf = TEE + (hA*(TEE-Tc_fixo) - DH*V*k*CEE)/(q[idx]*Cp)

    # Criando vetor de q/v
    qplot = np.ones(101)*q[idx]/V

    # Carregando esta curva no gráfico 2
    ax.plot(Tf, qplot, TEE)

# configurando grafico 2
ax.set_xlabel('Tf (K)')
ax.set_ylabel('q/V (1/h)')
ax.set_zlabel('T (K)')

# Plotando ambos os gráficos
plt.show()
```