Python Bootcamps 1, 2 and 3

- ✓ 1: Getting up to speed with Python
- ✓ 2: Learning to use Python to solve typical problem scenarios
- • 3: Detailed modeling of packed-bed and plug-flow reactors

Bootcamp 3 Outline

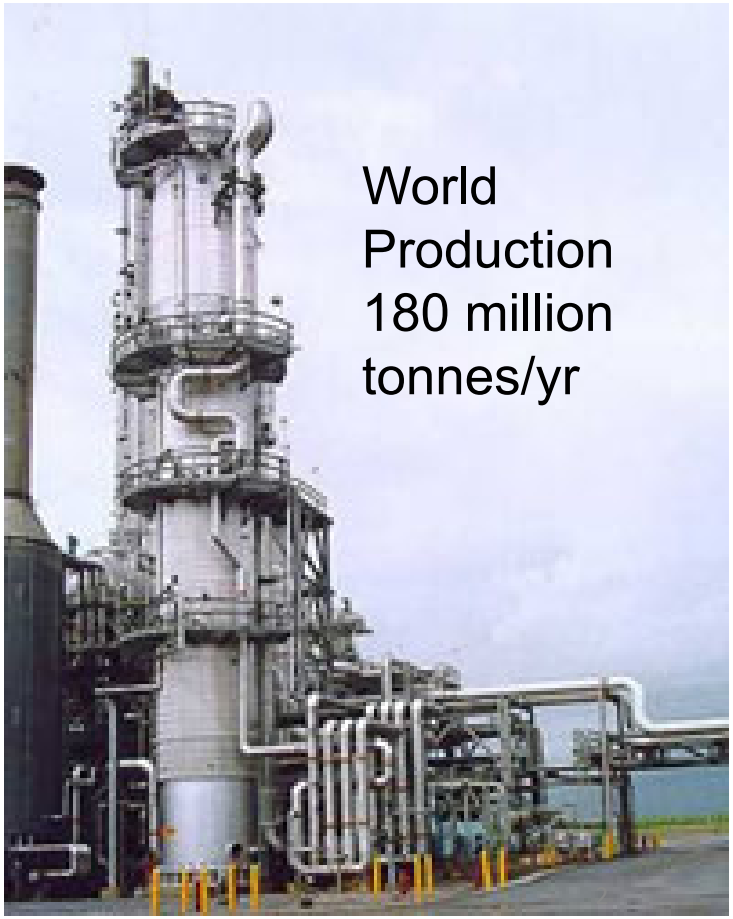
- Adiabatic, Packed-Bed, Plug-Flow Reactor
  - Ammonia Synthesis
- Tubular Reactor with Counter-current Heat Exchange
  - Acetone Cracking

Ordinary Differential Equation Models

Case Study 1

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis



World Production 180 million tonnes/yr

Ordinary Differential Equation Models

Case Study 1

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis

Reaction kinetics for main reaction $\frac{1}{2}N_2 + \frac{3}{2}H_2 \Leftrightarrow NH_3$

Forward reaction: $r_f = k_f \cdot p_{N_2}^{1/2} \cdot p_{H_2}^{3/2}$

$$k_f = k_{0f} \cdot e^{-\frac{E_f}{R \cdot T}} \qquad k_{0f} = 10{,}000 \ \frac{kmol}{m^3 s} \cdot \frac{1}{atm^2} \qquad E_f = 91{,}000 \ \frac{kJ}{kmol}$$

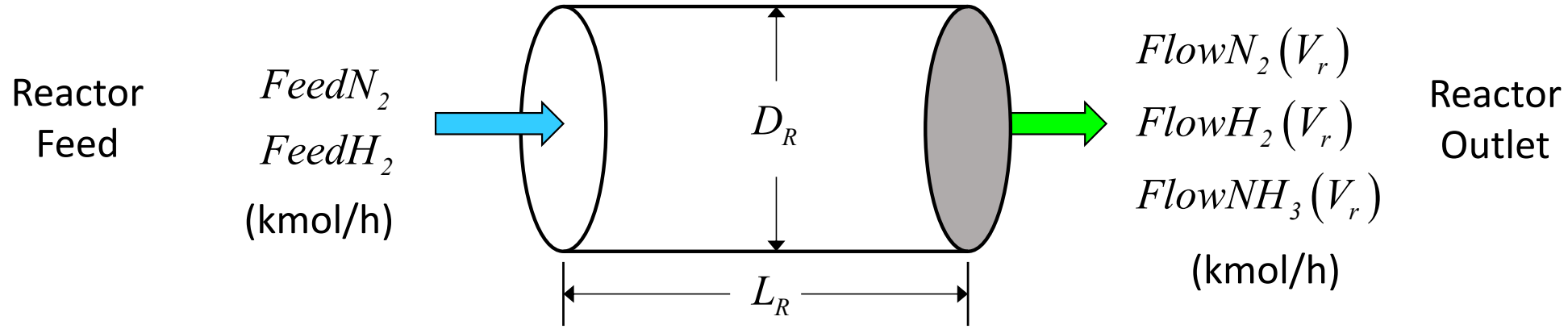
Reverse reaction: $r_r = k_r \cdot p_{NH_3}$

$$k_r = k_{0r} \cdot e^{-\frac{E_r}{R \cdot T}} \qquad k_{0r} = 1.3 \times 10^{10} \ \frac{kmol}{m^3 s} \cdot \frac{1}{atm} \qquad E_r = 141{,}000 \ \frac{kJ}{kmol}$$

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis



Differential Mole Balance on N_2

$$\frac{d[FlowN_2]}{dV} = (-r_f + r_r)\cdot\varepsilon$$

Note: $dV = A_r \cdot dz$

$$A_r = \pi\frac{D_r^2}{4} \qquad V_r = A_r \cdot L_r$$

Stoichiometric Balances on H_2 and NH_3

$$FlowH_2 = FeedH_2 - 3\cdot(FeedN_2 - FlowN_2)$$

$$FlowNH_3 = 2\cdot(FeedN_2 - FlowN_2)$$

dV is differential volume of empty reactor

ε is the void fraction of the packed bed

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis

Energy Balance

$$\frac{d}{dV}\left(\sum_i Flow_i \cdot H_i(T)\right) = 0$$

with constant heat capacity approximation

$$\frac{dT}{dV} \cong \frac{\left(r_f - r_r\right)\cdot\left(-\Delta H_{rxn}(T,P)\right)\cdot\varepsilon}{\left(\sum_i Flow_i \cdot C_{Pi}\right)}$$

pressure effect on enthalpy

$$H_i(T,P) = \int_{T_{ref}}^{T} C_{Pi}(T)\,dT + \int_{P_{ref}}^{P}\left[V - T\left(\frac{\partial V}{\partial T}\right)_P\right]dP + H_{fi}$$

$$\int_{T_{ref}}^{T} C_{Pi}(T)\,dT = \bar{C}_{Pi}(T)\cdot\left(T - T_{ref}\right)$$

$$\int_{P_{ref}}^{P}\left[V - T\left(\frac{\partial V}{\partial T}\right)_P\right]dP =$$

from eqn of state, analytically, or from P-V-T data. or using the Generalized Pitzer Correlation

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis

Pressure Drop – the Ergun equation for packed beds

$$\left[\frac{(P_0 - P_L)\cdot\rho}{G_0^2}\right]\cdot\left[\frac{D_P}{L}\right]\cdot\left[\frac{\varepsilon^3}{1-\varepsilon}\right] = 150\cdot\left[\frac{1-\varepsilon}{D_P\cdot G_0/\mu} + \frac{7}{4}\right]$$

written in terms of dimensionless groups

G_0 : mass flow rate per unit cross-sectional area of empty bed -- *constant with V*

Differential form:

$$\frac{dP}{dV} = \frac{1}{A_r}\cdot 150\cdot\left[\frac{1-\varepsilon}{D_P\cdot G_0/\mu} + \frac{7}{4}\right]\cdot\left[\frac{1-\varepsilon}{\varepsilon^3}\right]\cdot\left[\frac{G_0^2}{\rho\cdot D_P}\right]$$

P_0 : upstream pressure

P_L : downstream pressure at L

ρ : fluid density

G_0 : mass flux

D_P : effective particle diameter

ε : packing void fraction

μ : fluid viscosity

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis

Pressure Drop – the Ergun equation for packed beds

Fluid Density

$$\rho = \frac{\overline{MW}}{\tilde{V}} \qquad \overline{MW}: \text{avg molecular weight, } \frac{kg}{kmol} \qquad \tilde{V}: \text{specific volume, } \frac{m^3}{kmol}$$

$\tilde{V}$ from Peng-Robinson Equation of State

$$P = \frac{RT}{\tilde{V} - b_m} - \frac{a_m}{\tilde{V}\left(\tilde{V} + b_m\right) + b_m\left(\tilde{V} - b_m\right)}$$

Solve nonlinear, cubic equation for $\tilde{V}$

a_m, b_m : mixture coefficients

Ideal gas law approximation: $\rho = \frac{\overline{MW \cdot P}}{RT}$ 20% high at 150 atm

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis

Peng-Robinson EOS Mixture Coefficients

Coefficients for individual components

Units: K, kPa, kmol, kJ, m^3

$$a_i = 0.45724\frac{R^2T_c^2}{P_c}\left(1+m_i\left(1-\sqrt{\frac{T}{T_c}}\right)\right)^2 \qquad m_i = 0.37464 + 1.54226\,\omega_i - 0.26992\,\omega_i^2$$

$$b_i = 0.07780\frac{RT_c}{P_c}$$

k_{ij} : binary interactor factors

ω_i : acentric factor for component i

$\mathbf{x}$: mole fractions

Mixture coefficients

$$\mathbf{Q} = \sqrt{\mathbf{a}\cdot\mathbf{a}'}\otimes\left(1-\mathbf{K}\right) = \begin{bmatrix} 0 & k_{12}a_1a_2 & \cdots & k_{1n}a_1a_n \\ k_{12}a_1a_2 & 0 & k_{13}a_2a_3 & \vdots \\ \vdots & \vdots & \ddots & k_{n-1,n}a_{n-1}a_n \\ k_{1n}a_1a_n & \cdots & k_{n-1,n}a_{n-1}a_n & 0 \end{bmatrix} \qquad a_m = \mathbf{x}'\cdot\mathbf{Q}\cdot\mathbf{x}$$

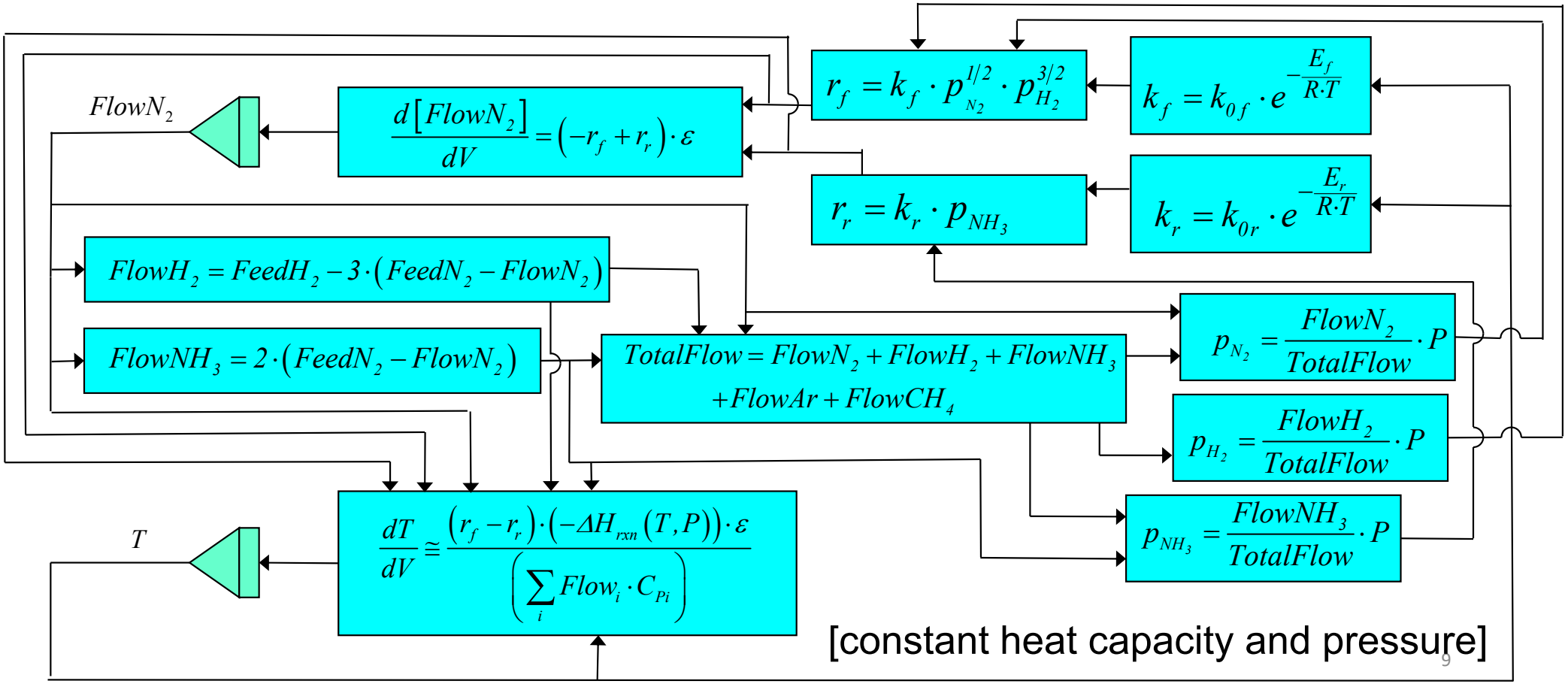
$$b_m = \mathbf{x}'\cdot\mathbf{b} = \sum_{i=1}^{n} x_i\cdot b_i$$

$\otimes$: item-by-item array multiplication

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model Information Diagram



Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

Ammonia.py

Header and basic data

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from HtCapFn import HtCap
from PBRsimplifiedFn import PBRsimplified
# basic data
Rgas = 8.314  # kJ/kmol/K
# molecular weights
MWN2 = 28.0134  # kg/kmol
MWH2 = 2.016
MWNH3 = 17.031
MWAr = 39.948
MWCH4 = 16.043
# heat capacity coefficients
# from fit of Hysys properties
# at 150 atm
CpCoef = np.array([[40.442,  -35.279,   46.918,  -19.41,  0],
                   [28.75,      1.86,     0,        0,     0],
                   [1088.5, -5691.4,  11787,   -10900,  3801.6],
                   [36.819,  -53.01,     63.973,  -27.022, 0],
                   [24.709,   50.297,     0,        0,     0]])
```

```
# heat capacities at 350 degC
Tmid = 350  # degC
CpN2 = HtCap(1,Tmid,CpCoef)
CpH2 = HtCap(2,Tmid,CpCoef)
CpNH3 = HtCap(3,Tmid,CpCoef)
CpAr = HtCap(4,Tmid,CpCoef)
CpCH4 = HtCap(5,Tmid,CpCoef)
# reaction kinetics
#forward reaction
k0f = 3.6e7  # kmol/m3/h/atm
Ef = 9.1e4  # kJ/kmol
# reverse reaction
k0r = 4.68e13  # kmol/m3/h/atm
Er = 1.41e5  # kJ/kmol
# reactor parameters
Dr = 3  # diameter, m
Lr = 1  # length, m
Ar = np.pi*Dr**2/4  # x-sectional area, m2
Vr = Ar*Lr  # volume, m3
# catalyst particles and packing
Dp = 1.e-3  # particle diameter, m
eps = 0.4  # void fraction
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

```
# feed conditions
FeedN2 = 12348  # kmol/h
FeedH2 = 37044
FeedNH3 = 0
FeedAr = 12391
FeedCH4 = 5652
FeedP = 150  # atm
FeedT = 270  # degC
# initial conditions
y0 = np.array([FeedN2, FeedT])
# solution span and intervals
vspan = [0., Vr]
veval = np.linspace(0,Vr,200)
# solve model
result = solve_ivp(PBRsimplified,vspan,y0,t_eval=veval,
                        args=(FeedN2,FeedH2,FeedAr,FeedCH4,
                        FeedP,k0f,Ef,k0r,Er,eps,
                        CpN2,CpH2,CpNH3,CpAr,CpCH4))
vs = result.t
MflowN2 = result.y[0,:]
MflowH2 = FeedH2 - 3*(FeedN2 - MflowN2)
MflowNH3 = 2*(FeedN2 - MflowN2)
T = result.y[1,:]
# conversion, N2 basis, in %
Conv = (FeedN2 - MflowN2)/FeedN2 * 100
```

Solve differential equations
and
unpack the results

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

Create plots and display exit temperature and conversion

```
# create plots
# temperature profile
plt.figure()
plt.plot(vs,T,c='k')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Temperature - degC')
plt.title('Temperature Profile')
# molar flows profiles
plt.figure()
plt.plot(vs,MflowN2,c='b',label='N2')
plt.plot(vs,MflowH2,'m',label='H2')
plt.plot(vs,MflowNH3,'g',label='NH3')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Molar Flow Rate - kmol/h')
plt.title('Molar Flow Profiles')
plt.legend()
```

```
# conversion profile
plt.figure()
plt.plot(vs,Conv,c='k')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')
# print exit conditions
n = len(vs)
print('Exit conversion = {0:4.1f} %'.format(Conv[n-1]))
print('Exit temperature = {0:6.1f} degC'.format(T[n-1]))
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

PBRsimplified.py

```python
import numpy as np
from HtRxnFn import HtRxn
def PBRsimplified(v,y,FeedN2,FeedH2,FeedAr,FeedCH4,P, \
                  k0f,Ef,k0r,Er,eps,CpN2,CpH2,CpNH3,CpAr,CpCH4):
    Rgas = 8.314  # kJ/kmol/K
    # unpack dependent variables
    MflowN2 = y[0]
    T = y[1]
    # algebraic equations from stoichiometry for H2 and NH3
    MflowH2 = FeedH2 - 3*(FeedN2 - MflowN2)
    MflowNH3 = 2*(FeedN2 - MflowN2)
    # total molar flow
    TotFlow = MflowN2+MflowH2+MflowNH3+FeedAr+FeedCH4
    # partial pressures - atm
    PN2 = MflowN2/TotFlow*P
    PH2 = MflowH2/TotFlow*P
    PNH3 = MflowNH3/TotFlow*P
    # forward and reverse reaction rates
    rf = k0f*np.exp(-Ef/Rgas/(T+273.15))+PN2**0.5*PH2**1.5
    rr = k0r*np.exp(-Er/Rgas/(T+273.15))*PNH3
    # differential balance on N2 in kmol/h/m3
    dy = np.zeros(2)
    dy[0] = -(rf-rr)*eps
    # differential energy balance
    HtRx = HtRxn(T)
    dy[1] = (rf-rr)*(-HtRx)*eps/ \
            (MflowH2*CpN2+MflowH2*CpH2+MflowNH3*CpNH3 \
             +FeedAr*CpAr+FeedCH4*CpCH4)
    return dy
```

Function to compute derivatives of N_2 and temperature

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

HtCapFn.py

```
def HtCap(component,T,CpCoef):
    cmp = component-1
    a = CpCoef[cmp,0]
    b = CpCoef[cmp,1]
    c = CpCoef[cmp,2]
    d = CpCoef[cmp,3]
    e = CpCoef[cmp,4]
    TK = T + 273.15
    TK1 = TK/1000
    return a + b*TK1 + c*TK1**2 + d*TK1**3 + e*TK1**4
```

HtRxnFn.py

```
def HtRxn(T):
    aa = -1.9314e5
    bb = 4.8403e5
    cc = -9.944e5
    dd = 8.8054e5
    ee = -2.9078e5
    Tk1 = (T+273.15)/1000
    Hrx = aa + bb*Tk1 + cc*Tk1**2 + dd*Tk1**3 + ee*Tk1**4
    return Hrx
```

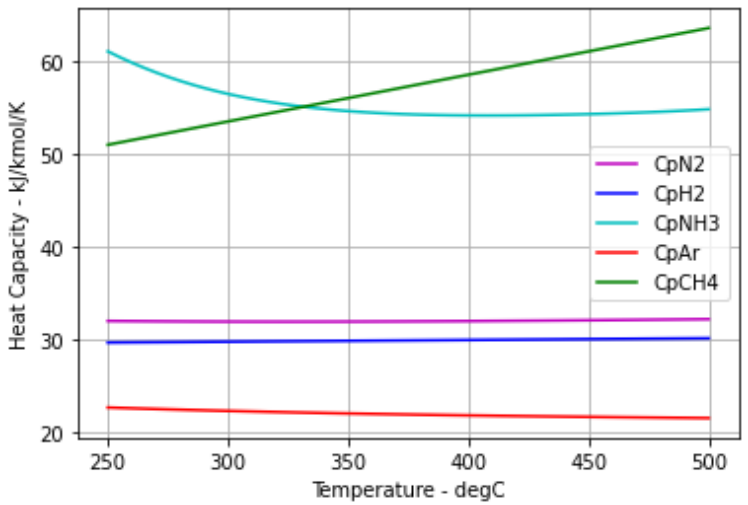
Supporting functions for heat capacity and heat of reaction

Ordinary Differential Equation Models

Ammonia Synthesis – Simplified Model

Heat Capacity Function Test

```
def HtCap(component,T,CpCoef):
    cmp = component-1
    a = CpCoef[cmp,0]
    b = CpCoef[cmp,1]
    c = CpCoef[cmp,2]
    d = CpCoef[cmp,3]
    e = CpCoef[cmp,4]
    TK = T + 273.15
    TK1 = TK/1000
    return a + b*TK1 + c*TK1**2 + d*TK1**3 + e*TK1**4
```



TestHtCap.py

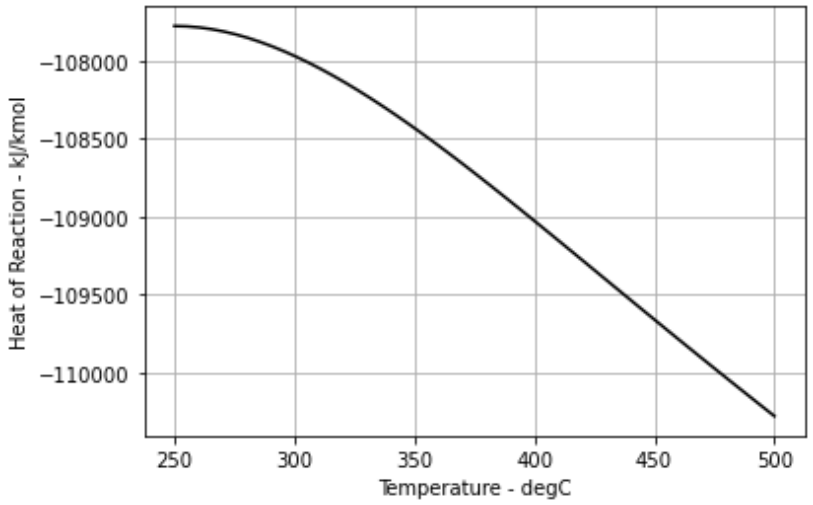
```
# Test HtCap function
import numpy as np
import matplotlib.pyplot as plt
from HtCapFn import HtCap

CpCoef = np.array([[40.442,  -35.279,   46.918,  -19.41,  0],
                   [28.75,     1.86,     0,        0,     0],
                   [1088.5, -5691.4,  11787,   -10900,  3801.6],
                   [36.819,  -53.01,    63.973,  -27.022, 0],
                   [24.709,   50.297,    0,        0,     0]])

T = np.linspace(250.,500.)
n = len(T)
CpN2 = []
CpH2 = []
CpNH3 = []
CpAr = []
CpCH4 = []
for i in range(n):
    CpN2.append(HtCap(1,T[i],CpCoef))
    CpH2.append(HtCap(2,T[i],CpCoef))
    CpNH3.append(HtCap(3,T[i],CpCoef))
    CpAr.append(HtCap(4,T[i],CpCoef))
    CpCH4.append(HtCap(5,T[i],CpCoef))

plt.plot(T,CpN2,c='m',label='CpN2')
plt.plot(T,CpH2,c='b',label='CpH2')
plt.plot(T,CpNH3,c='c',label='CpNH3')
plt.plot(T,CpAr,c='r',label='CpAr')
plt.plot(T,CpCH4,c='g',label='CpCH4')
plt.grid()
plt.xlabel('Temperature - degC')
plt.ylabel('Heat Capacity - kJ/kmol/K')
plt.legend()
```

Ordinary Differential Equation Models

Ammonia Synthesis – Simplified Model

Heat of Reaction Function Test

```
def HtRxn(T):
    aa = -1.9314e5
    bb = 4.8403e5
    cc = -9.944e5
    dd = 8.8054e5
    ee = -2.9078e5
    Tk1 = (T+273.15)/1000
    Hrx = aa + bb*Tk1 + cc*Tk1**2 + dd*Tk1**3 + ee*Tk1**4
    return Hrx
```



TestHtRxn.py

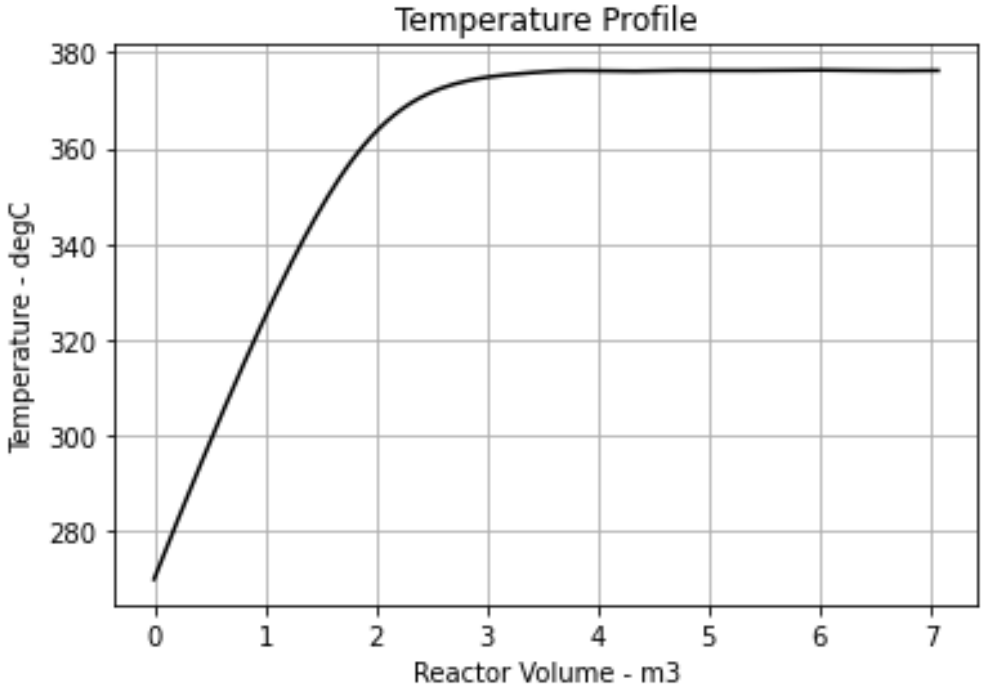
```
# Test HtRxn function
import numpy as np
import matplotlib.pyplot as plt
from HtRxnFn import HtRxn

T = np.linspace(250.,500.)
n = len(T)
HtRx = np.zeros(n)

for i in range(n):
    HtRx[i] = HtRxn(T[i])

plt.plot(T,HtRx,c='k')
plt.grid()
plt.xlabel('Temperature - degC')
plt.ylabel('Heat of Reaction - kJ/kmol')
```

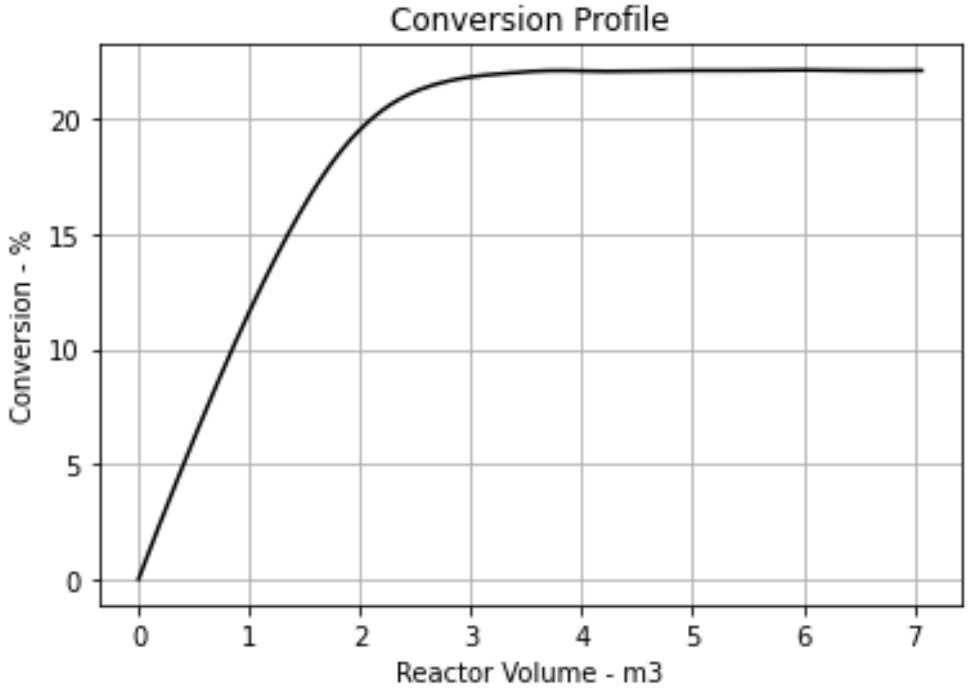
Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

Results



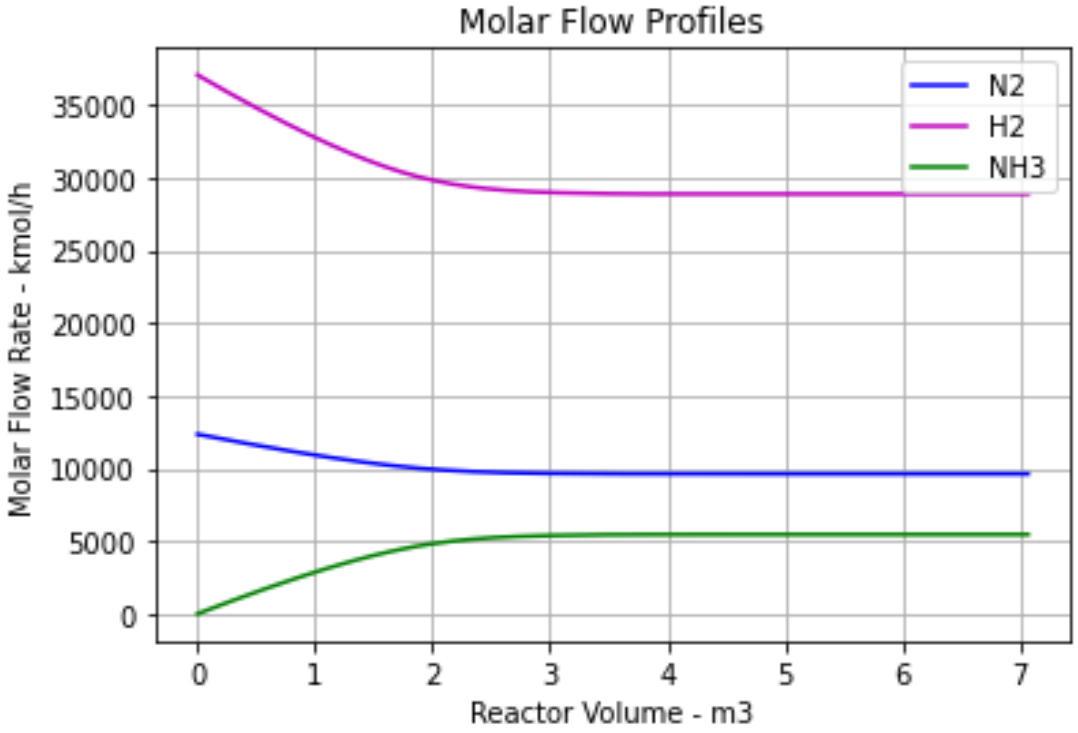


Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Simplified Model

Results



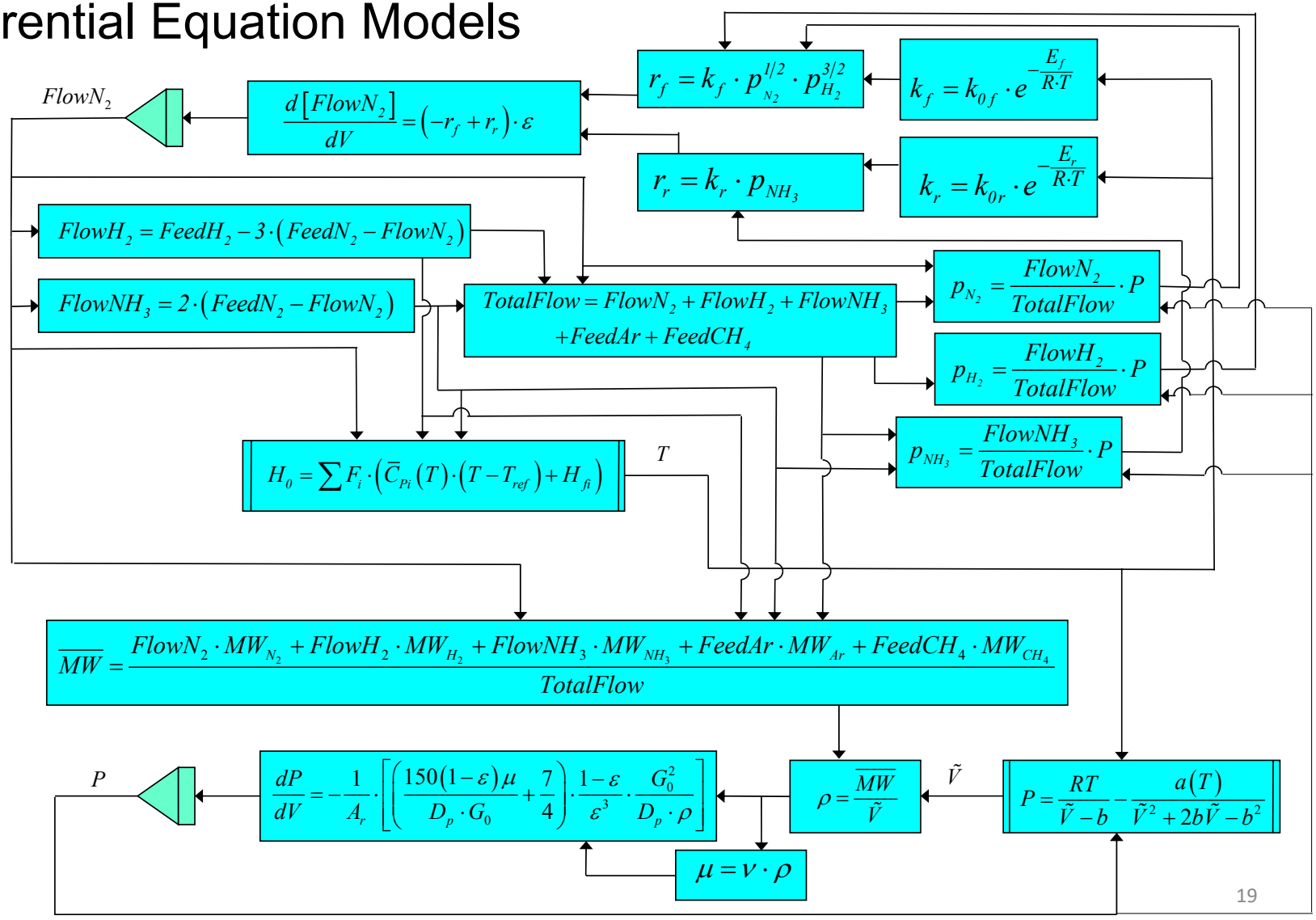

```
Exit conversion = 22.1 %
Exit temperature =  376.3 degC
```

Ordinary Differential Equation Models

Modeling and Simulation of a PBR

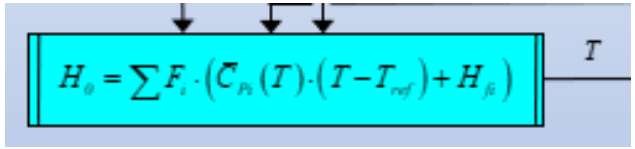
Ammonia Synthesis

Full Model Information Flow Diagram

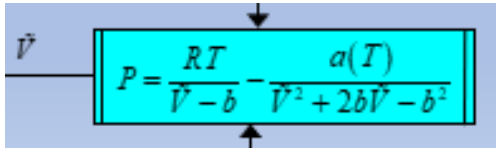


Ordinary Differential Equation Models

Numerical Technique to Solve the Implicit Relationships



$$H_0 - \sum F_i \cdot \left(\bar{C}_{Pi}(T) \cdot (T - T_{ref}) + H_{fi}\right) = 0$$



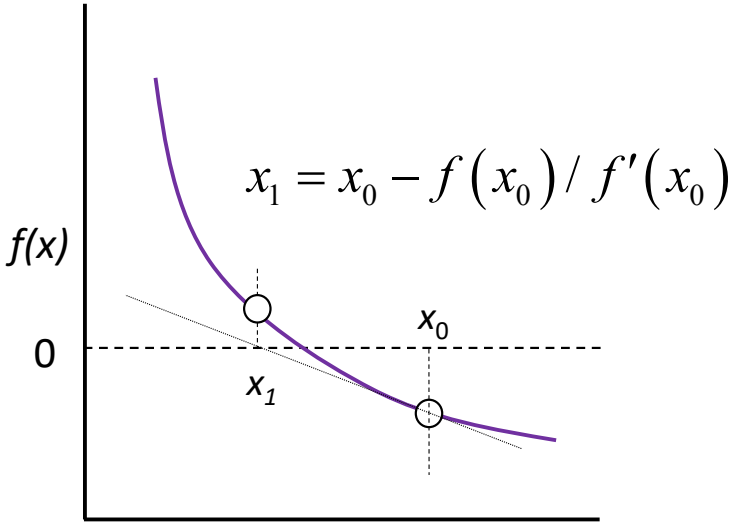
$$P - \frac{RT}{\tilde{V} - b} + \frac{a(T)}{\tilde{V}^2 + 2b\tilde{V} - b^2} = 0$$

Newton's Method to solve *f(x)* = 0

$$x_{k+1} = x_k - f(x_k) / f'(x_k)$$

An iterative method with a starting estimate x_0.

Convergence when $\left| \frac{x_{k+1} - x_k}{x_{k+1}} \right| < tol$



Ordinary Differential Equation Models

Numerical Technique to Solve the Implicit Relationships

The Modified Secant method based on Newton's method

When *f'(x)* is difficult or impossible to derive analytically:

$$f'(x) \cong \frac{f(x+\delta) - f(x)}{\delta}$$

δ : a small deviation, e.g., 1.e-7*x

Modified Secant Method is then

$$x_{k+1} = x_k - \delta f(x_k) / \left(f(x_k + \delta) - f(x_k) \right)$$

If the method diverges, it is also possible to include a decelerator factor (0 < *decel* < 1):

$$x_{k+1} = x_k - decel \cdot \delta f(x_k) / \left(f(x_k + \delta) - f(x_k) \right)$$

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

AmmoniaFull.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from HtCapFn import HtCap
from PBRFullFn import PBRFull
from FindTFn import findT
# basic data
Rgas = 8.314  # kJ/kmol/K
# molecular weights
MWN2 = 28.0134  # kg/kmol
MWH2 = 2.016
MWNH3 = 17.031
MWAr = 39.948
MWCH4 = 16.043
# heat capacity coefficients
# from fit of Hysys properties
# at 150 atm
CpCoef = np.array([[40.442,  -35.279,   46.918,  -19.41,  0],
                   [28.75,      1.86,     0,        0,     0],
                   [1088.5, -5691.4,  11787,   -10900,  3801.6],
                   [36.819,  -53.01,     63.973,  -27.022, 0],
                   [24.709,   50.297,     0,        0,     0]])
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

```
# reaction kinetics
#forward reaction
k0f = 3.6e7  # kmol/m3/h/atm
Ef = 9.1e4  # kJ/kmol
# reverse reaction
k0r = 4.68e13  # kmol/m3/h/atm
Er = 1.41e5  # kJ/kmol
# reactor parameters
Dr = 3  # diameter, m
Lr = 1  # length, m
Ar = np.pi*Dr**2/4  # x-sectional area, m2
Vr = Ar*Lr  # volume, m3
# catalyst particles and packing
Dp = 1.e-3  # particle diameter, m
eps = 0.4  # void fraction
# gas kinematic viscosity
nu = 5.075e-7  # m2/s
# feed conditions
FeedN2 = 12348  # kmol/h
FeedH2 = 37044
FeedNH3 = 0
FeedAr = 12391
FeedCH4 = 5652
FeedP = 150  # atm
FeedT = 270  # degC
```

```
# mass flux
M0 = FeedN2*MWN2+FeedH2*MWH2+FeedNH3*MWNH3+FeedAr*MWAr+FeedCH4*MWCH4
G0 = M0/Ar/3600  # kg/s/m2
# initial conditions
T0 = FeedT
H0 = (FeedN2*HtCap(1,T0,CpCoef)+FeedH2*HtCap(2,T0,CpCoef) \
      +FeedNH3*HtCap(3,T0,CpCoef)+FeedAr*HtCap(4,T0,CpCoef) \
      +FeedCH4+HtCap(5,T0,CpCoef))*T0
y0 = np.array([FeedN2, FeedP, H0])
# solution span and intervals
vspan = [0., Vr]
veval = np.linspace(0,Vr,200)
# solve model
result = solve_ivp(PBRFull,vspan,y0,t_eval=veval,
                           args=(FeedN2,FeedH2,FeedAr,FeedCH4,
                           k0f,Ef,k0r,Er,eps,nu,G0,Dp,Ar,
                           MWN2,MWH2,MWNH3,MWAr,MWCH4,CpCoef))
vs = result.t
MflowN2 = result.y[0,:]
MflowH2 = FeedH2 - 3*(FeedN2 - MflowN2)
MflowNH3 = 2*(FeedN2 - MflowN2)
P = result.y[1,:]
H = result.y[2,:]
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

```
# solve for T profile
n = len(vs)
T = np.zeros(n)
for i in range(n):
    Tmid = 350
    T[i] = findT(H[i],Tmid,MflowN2[i],MflowH2[i],MflowNH3[i] \
                 ,FeedAr,FeedCH4,CpCoef)
# conversion, N2 basis, in %
Conv = (FeedN2 - MflowN2)/FeedN2 * 100
# create plots
# temperature profile
plt.figure()
plt.plot(vs,T,c='k')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Temperature - degC')
plt.title('Temperature Profile')
# pressure profile
plt.figure()
plt.plot(vs,P,c='k')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Pressure - atm')
plt.title('Pressure Profile')
```

```
# molar flows profiles
plt.figure()
plt.plot(vs,MflowN2,c='b',label='N2')
plt.plot(vs,MflowH2,'m',label='H2')
plt.plot(vs,MflowNH3,'g',label='NH3')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Molar Flow Rate - kmol/h')
plt.title('Molar Flow Profiles')
plt.legend()
# conversion profile
plt.figure()
plt.plot(vs,Conv,c='k')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')
# print exit conditions
n = len(vs)
print('Exit conversion = {0:4.1f} %'.format(Conv[n-1]))
print('Exit temperature = {0:6.1f} degC'.format(T[n-1]))
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

PBRFullFn.py

```
import numpy as np
from HtRxnPFn import HtRxnP
from SpecVolFn import SpecVol
from FindTFn import findT
def PBRFull(v,y,FeedN2,FeedH2,FeedAr,FeedCH4,
            k0f,Ef,k0r,Er,eps,nu,G0,Dp,Ar,
            MWN2,MWH2,MWNH3,MWAr,MWCH4,CpCoef):
    Rgas = 8.314  # kJ/kmol/K
    # unpack dependent variables
    MflowN2 = y[0]
    P = y[1]
    H = y[2]
    # algebraic equations from stoichiometry for H2 and NH3
    MflowH2 = FeedH2 - 3*(FeedN2 - MflowN2)
    MflowNH3 = 2*(FeedN2 - MflowN2)
    # total molar flow
    TotFlow = MflowN2+MflowH2+MflowNH3+FeedAr+FeedCH4
    # partial pressures - atm
    PN2 = MflowN2/TotFlow*P
    PH2 = MflowH2/TotFlow*P
    PNH3 = MflowNH3/TotFlow*P
    # find T from H
    T = 350
    T = findT(H,T,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4,CpCoef)
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

```
# forward and reverse reaction rates
rf = k0f*np.exp(-Ef/Rgas/(T+273.15))+PN2**0.5*PH2**1.5
rr = k0r*np.exp(-Er/Rgas/(T+273.15))*PNH3
# differential balance on N2 in kmol/h/m3
dy = np.zeros(3)
dy[0] = -(rf-rr)*eps
# average molecular weight
MWavg = (MflowN2*MWN2+MflowH2*MWH2+MflowNH3*MWNH3 \
         +FeedAr*MWAr+FeedCH4*MWCH4)/TotFlow
# gas density
z = np.array([MflowN2, MflowH2, MflowNH3, FeedAr, FeedCH4])/TotFlow
w = np.array([ 0.039, -0.216, 0.25, 0.001, 0.011])
Tc = np.array([126.2, 33.19, 405.65, 150.86, 190.564])
Pc = np.array([3394, 1297, 11277, 4870, 4641])
K = np.array([[0, -0.036, 0.222, 0, 0.036],
              [-0.036, 0, 0, 0, 0.202],
              [0.222, 0, 0, 0, 0],
              [0, 0, 0, 0, 0.023],
              [0.036, 0.202, 0, 0.023, 0]])
SV = SpecVol(T,P,z,w,Tc,Pc,K)
RhoGas = MWavg/SV
# gas viscosity from mu and RhoGas
mu = nu * RhoGas
```

```
# differential Ergun equation, dP/dV in atm/m3
dy[1] = -(150*(1-eps)/Dp/G0*mu+7/4)*(1-eps)/eps**3/Dp \
        /RhoGas*G0**2/Ar/101325
    # differential energy balance
HtRx = HtRxnP(T,P)
dy[2] = (rf-rr)*(-HtRx)*eps
return dy
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

FindTFn.py

```
from HtCapFn import HtCap
def fH(H,T,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4,CpCoef):
    result = H - (MflowN2*HtCap(1,T,CpCoef)+MflowH2*HtCap(2,T,CpCoef) \
                  +MflowNH3*HtCap(3,T,CpCoef)+FeedAr*HtCap(4,T,CpCoef) \
                  +FeedCH4*HtCap(5,T,CpCoef))*T
    return result

def findT(H,T1,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4,CpCoef):
    tol = 1.e-7
    while True:
        T2 = T1 + 0.1
        Tnew = T1 - 0.1*fH(H,T1,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4 \
               ,CpCoef) / \
              (fH(H,T2,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4,CpCoef) \
               - fH(H,T1,MflowN2,MflowH2,MflowNH3,FeedAr,FeedCH4,CpCoef))
        if abs((Tnew-T1)/Tnew) < tol: break
        T1 = Tnew
    return Tnew
```

Modified Secant method

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

HtCapFn.py

```
def HtCap(component,T,CpCoef):
    cmp = component-1
    a = CpCoef[cmp,0]
    b = CpCoef[cmp,1]
    c = CpCoef[cmp,2]
    d = CpCoef[cmp,3]
    e = CpCoef[cmp,4]
    TK = T + 273.15
    TK1 = TK/1000
    return a + b*TK1 + c*TK1**2 + d*TK1**3 + e*TK1**4
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

SpecVolFn.py

```
import numpy as np
Rgas = 8.314  # kJ/kmol/K
def PR(V,T,P,am,bm):
    er = P - (Rgas*T/(V-bm) - am/(V**2+2*bm*V-bm**2))
    return er

def SpecVol(T,P,z,w,Tc,Pc,K):
    Tk = T + 273.15
    PkPa = P * 101.325
    m = 0.37464 + 1.54226*w - 0.26992*w**2
    alpha = (1 + m*(1-np.sqrt(Tk/Tc)))**2
    a = 0.45724*Rgas**2*Tc**2/Pc*alpha
    b = 0.0788*Rgas*Tc/Pc
    Q = (a*a)**0.5*(1-K)
    Qz = Q.dot(z)
    am = z.dot(Qz)
    bm = z.dot(b)
    V1 = Rgas*Tk/PkPa
    tol = 1.e-6
    while True:
        delta = 0.001*V1
        V2 = V1 + delta
        Vnew = V1 - delta*PR(V1,Tk,PkPa,am,bm) \
            / (PR(V2,Tk,PkPa,am,bm)-PR(V1,Tk,PkPa,am,bm))
        if abs((Vnew-V1)/Vnew) < tol: break
        V1 = Vnew
    return Vnew
```

Modified Secant method

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

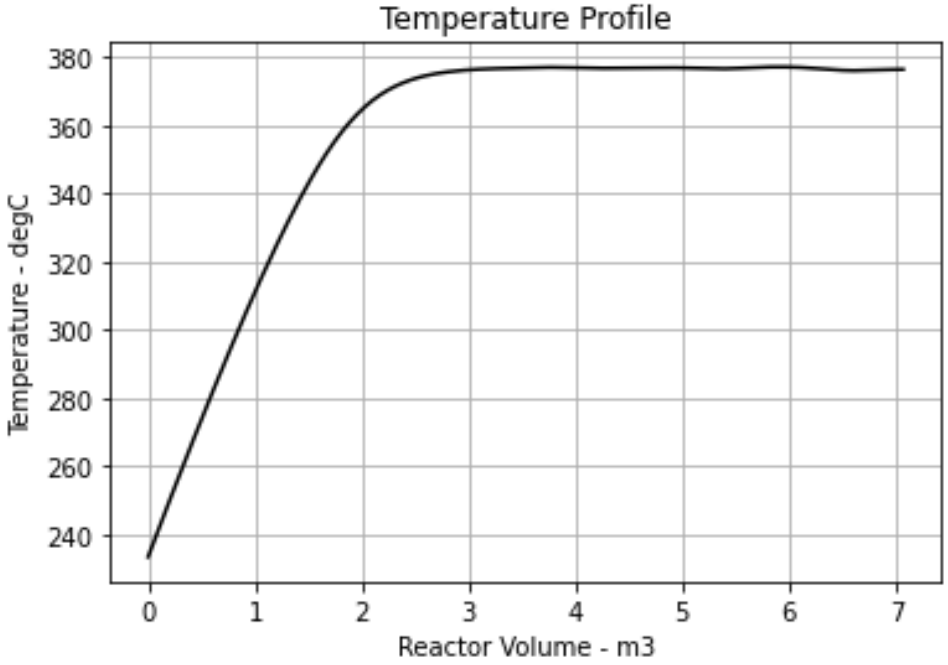
HtRxnPFn.py

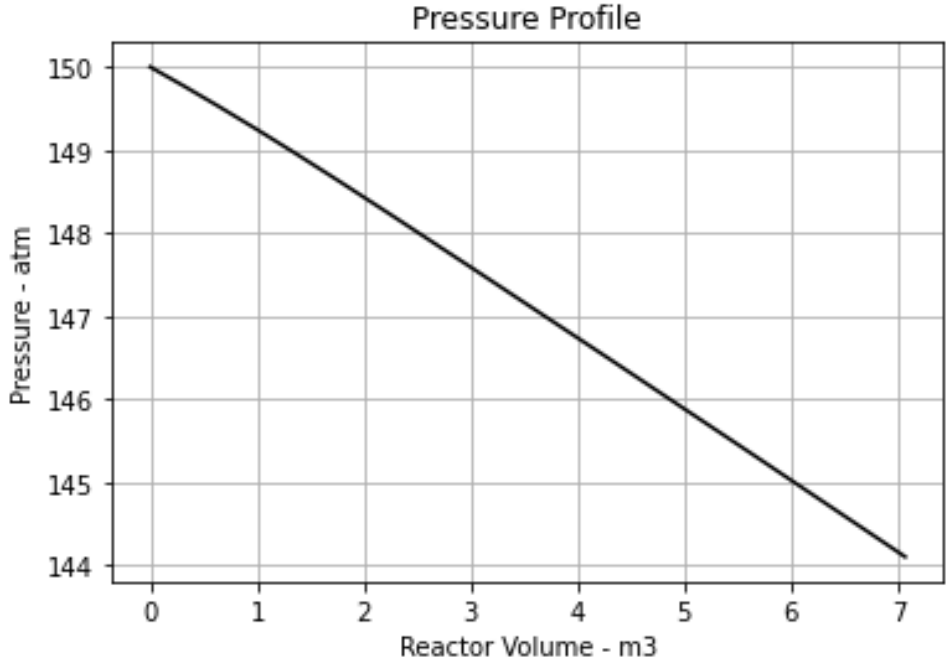
```
Rgas = 8.314  # kJ/kmol/K
def HtRxnP(T,P):
    import numpy as np
    Rgas = 8.314  # kJ/kmol/K
    aa = -1.9314e5
    bb = 4.8403e5
    cc = -9.944e5
    dd = 8.8054e5
    ee = -2.9078e5
    Tk = T + 273.15
    Tk1 = Tk/1000
    Pk = P*101.325
    HrxT = aa + bb*Tk1 + cc*Tk1**2 + dd*Tk1**3 + ee*Tk1**4
    Tc = np.array([405.6, 126.2, 33.2])
    Pc = np.array([112.5, 33.5, 12.8])
    Omega = np.array([0.25, 0.04, 0.0])
    Tr = Tk / Tc
    Pck = Pc*101.325
    B0 = 0.1445 - 0.33/Tr - 0.1385/Tr**2 - 0.0121/Tr**3
    B1 = 0.073 + 0.46/Tr - 0.5/Tr**2 - 0.097/Tr**3 - 0.0073/Tr**8
    dB0 = 0.33/Tr**2 + 0.277/Tr**3 + 0.0363/Tr**4
    dB1 = -0.46/Tr**2 + 1./Tr**3 + 0.291/Tr**4 + 0.0584/Tr**9
    H = Rgas*Tk*(1-Pk)/Pck*((dB0-B0/Tr)+Omega*(dB1-B1/Tr))
    Hrx = HrxT + np.array([2, -1, 3]).dot(H)
    return Hrx
```

Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model

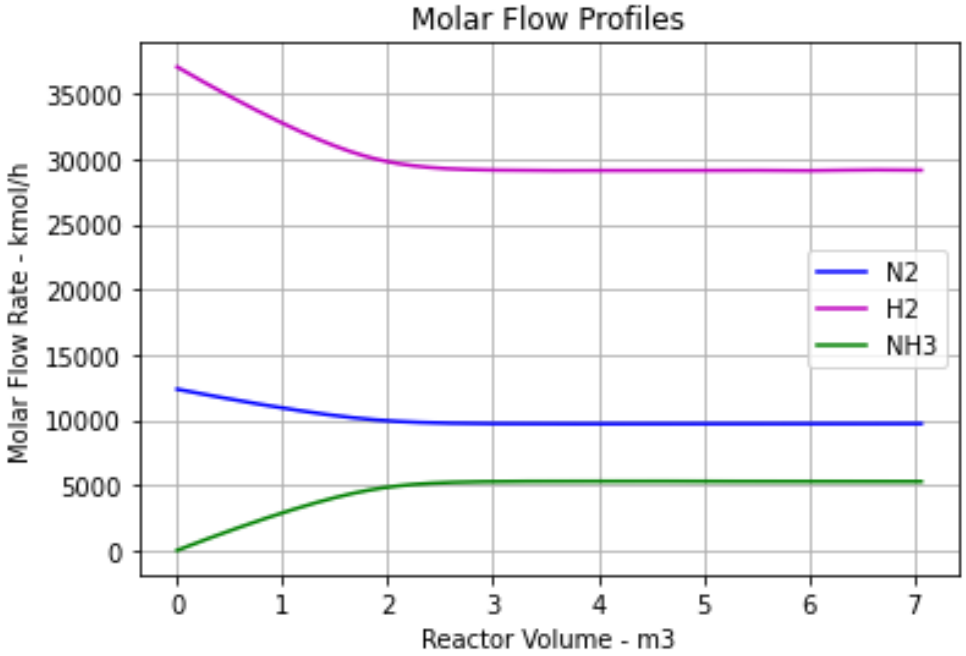


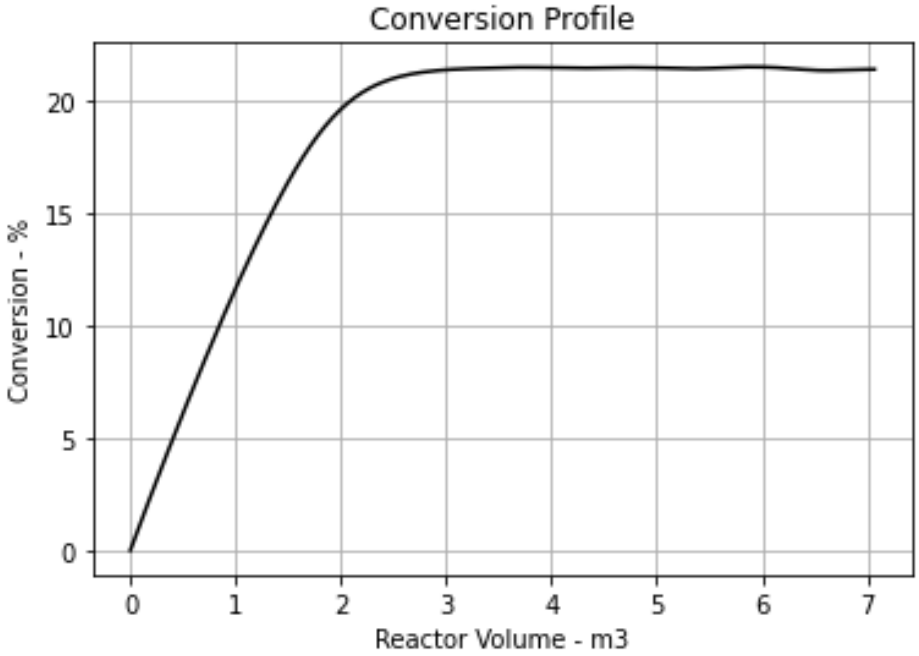


Ordinary Differential Equation Models

Modeling and Simulation of a Plug-flow, Packed-bed Reactor (PBR)

Ammonia Synthesis – Full Model






```
Exit conversion = 21.4 %
Exit temperature =  376.4 degC
```

compared to simplified model results

```
Exit conversion = 22.1 %
Exit temperature =  376.3 degC
```

Case Study 2

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

$$CH_3COCH_3 \Rightarrow CH_2CO + CH_4$$

Feed: 7850 kg/hr
7.85 kg/hr per tube
0.135 kmol/hr

Inlet temperature: 1035 K
Inlet pressure: 162 kPa (1.6 atm)

Counter-current heat transfer
Air: 90 T/hr
Inlet temperature: 1250 K

Reactor: 1000 1" Sch 40 tubes
Total volume: 2 m^3
Tube ID: 26.7 mm
Tube length: 3.57 m

Assume $\Delta P \cong 0$

adapted from
Fogler, H. Scott, **Elements of Chemical Reaction Engineering**, 4th Edition, Prentice-Hall, 2006, p. 504.

Case Study 2

Tubular Reactor with Counter-current Heat Exchange

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Basic data:

$$r_A = -k \cdot C_A \qquad ln(k) = 42.529 - \frac{34222}{T}$$

r_A : reaction rate of acetone, $\frac{kmol}{hr \cdot m^3}$

C_A : concentration of acetone, $\frac{kmol}{m^3}$

k : rate parameter, $1/hr$

T : temperature, K

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Basic data: Heat capacity

Acetone: $C_{PA} = 6.8132 + 278.6 \cdot Tk - 156.28 \cdot Tk^2 + 34.76 \cdot Tk^3$ $\frac{kJ}{kmol \cdot K}$ $Tk = \frac{T[K]}{1000}$

Ketene: $C_{PK} = 18.909 + 143.56 \cdot Tk - 130.23 \cdot Tk^2 + 66.526 \cdot Tk^3 - 14.112 \cdot Tk^4$

Methane: $C_{PM} = -0.7030 + 108.48 \cdot Tk - 42.522 \cdot Tk^2 + 5.8628 \cdot Tk^3 + 0.67857 \cdot \frac{1}{Tk^2}$

$$\bar{C}_{PA}(T) = \frac{\int_{T_{ref}}^{T} C_{PA}(T) \cdot dT}{T - T_{ref}} = 1000 \cdot \frac{\int_{Tk_{ref}}^{Tk} C_{PA}(tk) \cdot d(tk)}{Tk - Tk_{ref}}$$

Heat of reaction

$$\Delta H_{rxn}(25°C) = 80,770 \ \frac{kJ}{kmol}$$

endothermic

$$\Delta H_{rxn}(T) = \Delta H_{rxn}(25°C) - \Delta H_A(T) + \Delta H_K(T) + \Delta H_M(T)$$

$$\Delta H_i(T) = 1000 \cdot \int_{Tk_{ref}}^{Tk} C_{Pi}(tk)\, d(tk)$$

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Feed concentration:

$$C_{AF} = \frac{n}{V} = \frac{P}{R \cdot T} = \frac{162\left[kPa\right]}{8.314\left[\frac{kPa \cdot m^3}{kmol \cdot K}\right] \cdot 1035\left[K\right]} = 0.018 \ \frac{kmol}{m^3}$$

Reactor balances:

$$\frac{dF_A}{dV} = r_A = -k \cdot C_A \qquad C_A = \frac{F_A}{F_T} \cdot C_{AF} \cdot \frac{T_F}{T}$$

$$F_K = F_M = F_{AF} - F_A \qquad F_T = F_A + F_K + F_M$$

$$\frac{d\dot{H}}{dV} = UA\left(T_a - T\right)$$

$$\dot{H} = \dot{H}_A + \dot{H}_K + \dot{H}_M$$

$$\dot{H}_A = F_A \cdot \left(\overline{C}_{PA}\left(T\right) \cdot \left(T - T_{ref}\right) + H_{fA}\right) \ \ldots$$

Air energy balance:

$$\frac{d\dot{H}_a}{d\left(-V\right)} = UA\left(T - T_a\right)$$

counter-current

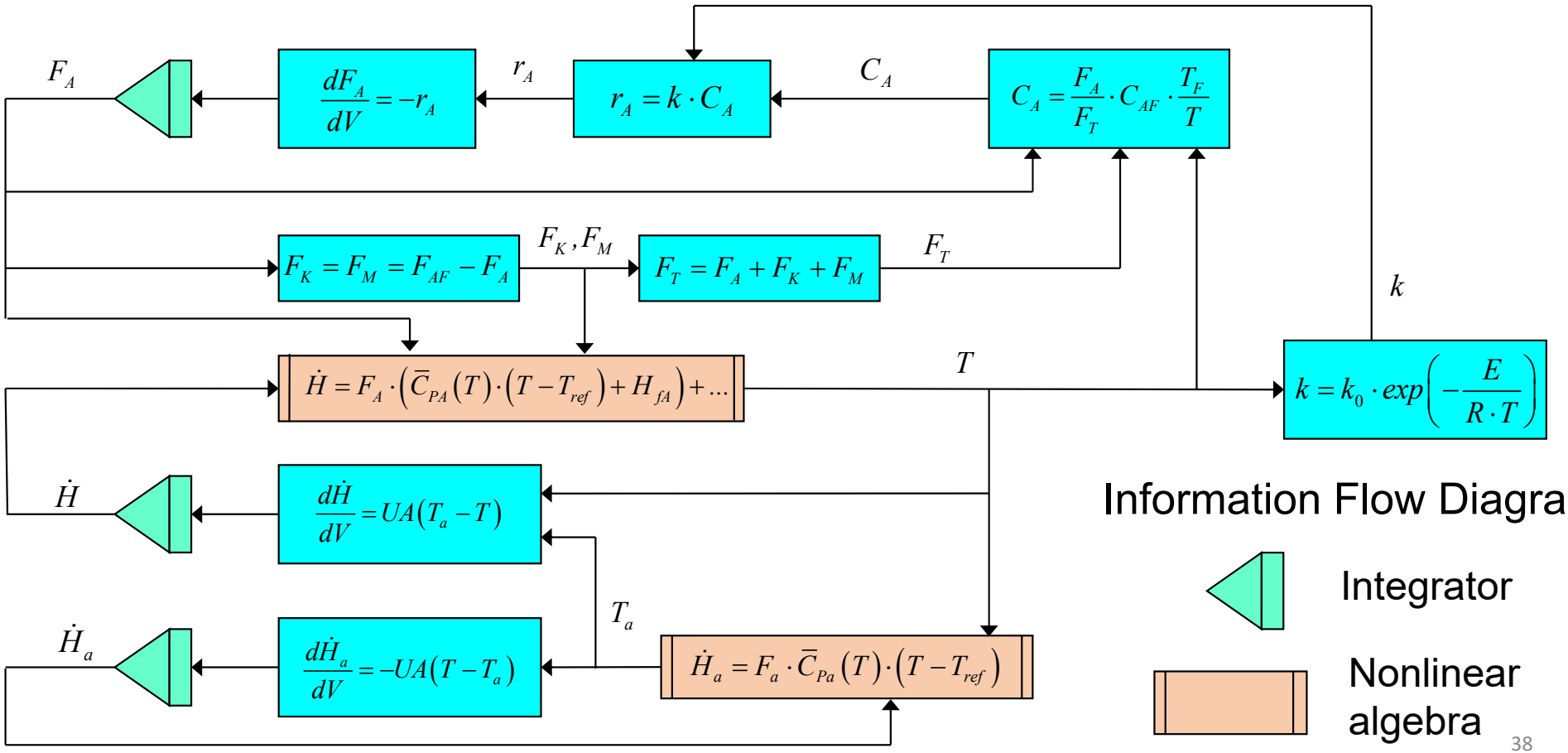
$$\dot{H}_a = F_a \cdot \overline{C}_{Pa}\left(T\right) \cdot \left(T - T_{ref}\right)$$

Note:

$$H_{fa} = 0$$

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene – full model



Information Flow Diagram

Integrator

Nonlinear algebra

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplification of the enthalpy balance

$$\frac{d\dot{H}}{dV} = UA\left(T_a - T\right)$$

$$\frac{d\dot{H}}{dV} = \frac{d\sum\left(F_i H_i\right)}{dV} = \sum \frac{dF_i}{dV} H_i + \sum F_i \frac{dH_i}{dV}$$

$$\frac{dF_i}{dV} = r_i = \nu_i \cdot \left(-r_A\right)$$

$$\frac{dH_i}{dV} = C_{Pi} \frac{dT}{dV} \quad \textit{assuming constant heat capacity}$$

$$\frac{dH_i}{dV} = \left(-r_A\right)\sum \nu_i H_i + \frac{dT}{dV}\sum F_i C_{Pi}$$

$$\sum \nu_i H_i = \Delta H_{rx} \quad \nu_i : \textit{stoichiometric coefficients}$$

$$\frac{dT}{dV} = \frac{r_A \cdot \Delta H_{rx} + UA\left(T_a - T\right)}{\sum F_i C_{Pi}}$$

$$\frac{d\dot{H}_a}{d\left(-V\right)} = UA\left(T - T_a\right)$$

assuming constant heat capacity

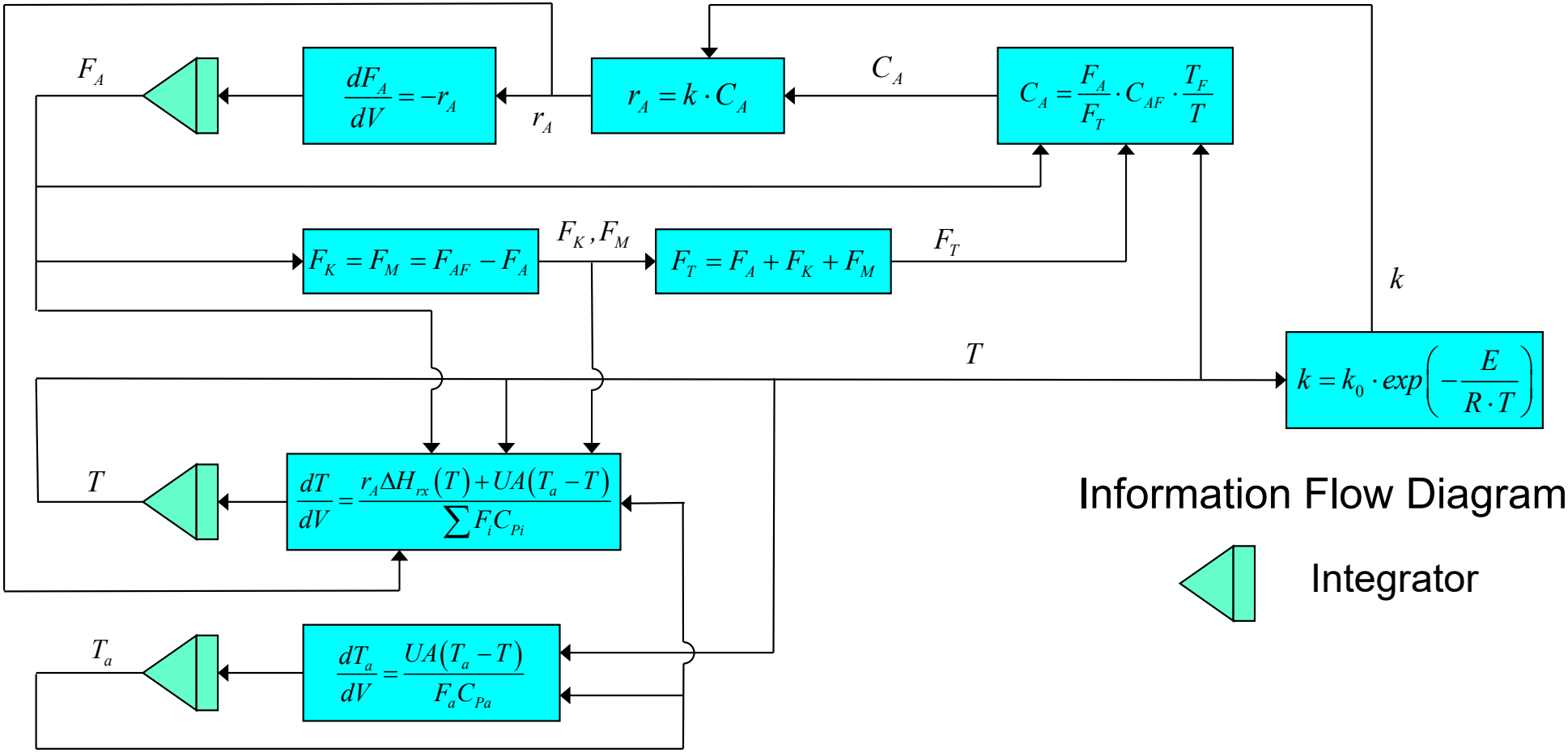
and molar flow rate

$$\frac{d\dot{H}_a}{dV} = F_a \cdot C_{Pa} \cdot \frac{dT}{dV}$$

$$\frac{dT_a}{dV} = \frac{UA\left(T_a - T\right)}{F_a C_{Pa}}$$

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene - simplified model



Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solution Strategy

Estimate final air temperature at v = 0

→ Solve model from v = 0 to v = Vr

Determine air temperature at v = Vr from solution

If air temperature at v = Vr meets spec ⟶ done!

Adjust final air temperature at v = 0

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model **AcetonePFR_Simplified.py**

```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRsimplifiedFn import AcetonePFR
import matplotlib.pyplot as plt

Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
Tref = 298.15  # K

MWA = 58.08  # kg/kmol
MWAir = 28.96

NoTubes = 1000
TotalVolume = 2  # m3

VolPerTube = TotalVolume/NoTubes
TubeID = 26.7e-3  # m3
TubeXC = np.pi*TubeID**2/4
TubeLength = VolPerTube/TubeXC

U = 400  # kJ/m2/hr/K
A = 4/TubeID  # m2/m3
```

```
ReactorMassFeed = 7850  # kg/hr
FAM = ReactorMassFeed/NoTubes
FAF = FAM/MWA

TF = 1035  # K
P = 162  # kPa
CAF = P/Rgas/TF  # kmol/m3

AirFeed = 11088*8  # kg/hr
FaM = AirFeed/NoTubes
Fa = FaM/MWAir
TaF = 1250  # K
Ta0 = 1117.7  # K - estimate

vspan = [0., VolPerTube]
veval = np.linspace(0., VolPerTube, 200)

y0 = [ FAF, TF, Ta0 ]

result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                   method='LSODA',args=(FAF,CAF,TF,Fa,U,A))
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model

```
vs = result.t
n = len(vs)
FAout = result.y[0,:]
FKout = FAF - FAout
FMout = FAF - FAout
Tout = result.y[1,:]
Taout = result.y[2,:]

Conv = (FAF-FAout)/FAF * 100

plt.figure()
plt.plot(vs,FAout,c='k',label='Acetone')
plt.plot(vs,FKout,c='g',label='Ketene')
plt.grid()
plt.ylim(0, 0.14)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Molar Flow Rate - kmol/hr')
plt.title('Molar Flow Rates')
plt.legend()
```

```
plt.figure()
plt.plot(vs,Tout,c='b',label='Reaction')
plt.plot(vs,Taout,c='r',label='Air')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Temperature - K')
plt.title('Temperature Profiles')
plt.legend()

plt.figure()
plt.plot(vs,Conv,c='g')
plt.grid()
plt.ylim(0, 100)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')

print('\nAir Entry Temperature = {0:7.1f} K'.format(Taout[n-1]))
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model

AcetonePFRsimplifiedFn.py

```
import numpy as np
from HtRxnAcetoneFn import HtRxn

def AcetonePFR(v,y,FAF,CAF,TF,Fa,U,A):
    Rgas = 8.314
    CpA = 163.89
    CpK = 84.65
    CpM = 71.79
    CpAir = 33.44
    lnk0 = 42.529
    k0 = np.exp(lnk0)
    E = 284522
    FA = y[0]
    T = y[1]
    Ta = y[2]
    FK = FAF - FA
    FM = FAF - FA
    FT = FA + FK + FM
    CA = FA/FT*CAF*TF/T
    rA = k0*np.exp(-E/Rgas/T)*CA
    dy = np.zeros(3)
    dy[0] = -rA
    SumCp = FA*CpA + FK*CpK + FM*CpM
    dy[1] = (rA*(-HtRxn(T))+U*A*(Ta-T))/SumCp
    dy[2] = U*A*(Ta-T)/Fa/CpAir
    return dy
```

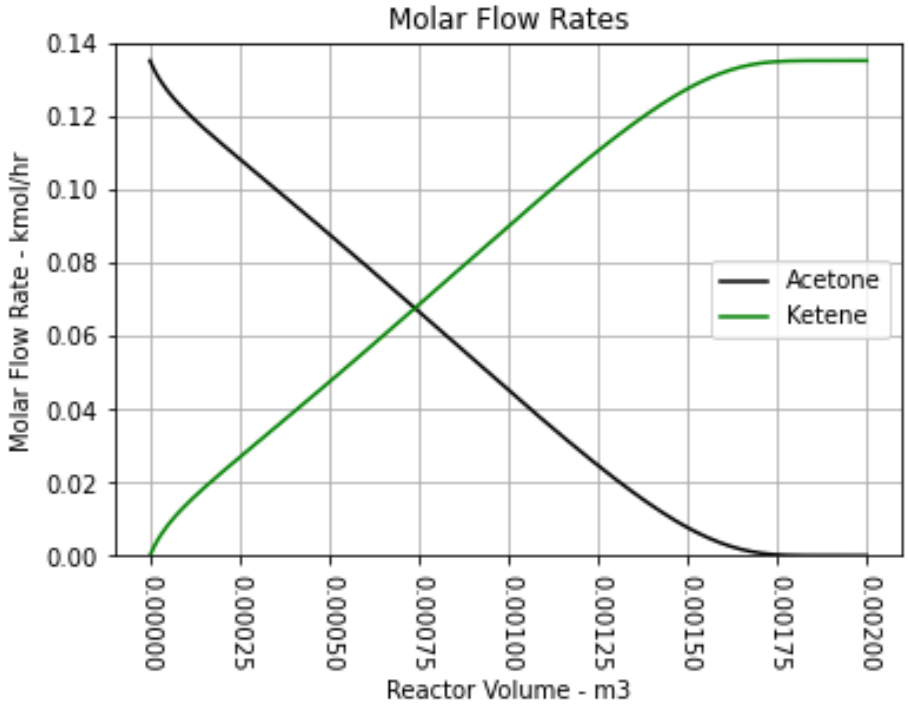
HtRxnAcetoneFn.py

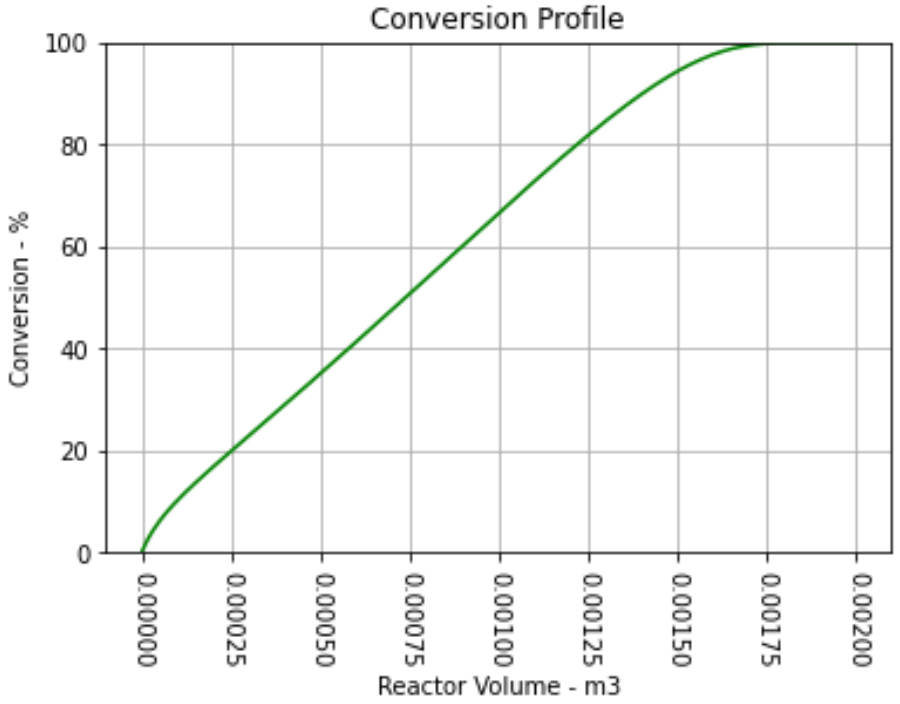
```
def HtRxn(T):
    Hx0 = 80.77e3
    Tk = T/1000
    Tk0 = (25+273.15)/1000
    # acetone
    HA = (6.8132e-3*Tk+0.2786/2*Tk**2-0.15628/3*Tk**3 \
          +0.03476/4*Tk**4 - (6.8132e-3*Tk0+0.2786/2*Tk0**2 \
          -0.15628/3*Tk0**3+0.03476/4*Tk0**4))*1000
    # ketene
    HK = 18.909*Tk+143.56/2*Tk**2-130.23/3*Tk**3 \
         +66.526/4*Tk**4-14.112/5*Tk**5 \
         - (18.909*Tk0+143.56/2*Tk0**2-130.23/3*Tk0**3 \
             +66.526/4*Tk0**4-14.112/5*Tk0**5)
    # methane
    HM = -0.703028*Tk+108.4773/2*Tk**2-42.52157/3*Tk**3 \
         +5.862788/4*Tk**4-0.678565/Tk \
         -(-0.703028*Tk0+108.4773/2*Tk0**2-42.52157/3*Tk0**3 \
              +5.862788/4*Tk0**4-0.678565/Tk0)
    hx = Hx0 + (-HA+HK+HM)*1000
    return hx
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model

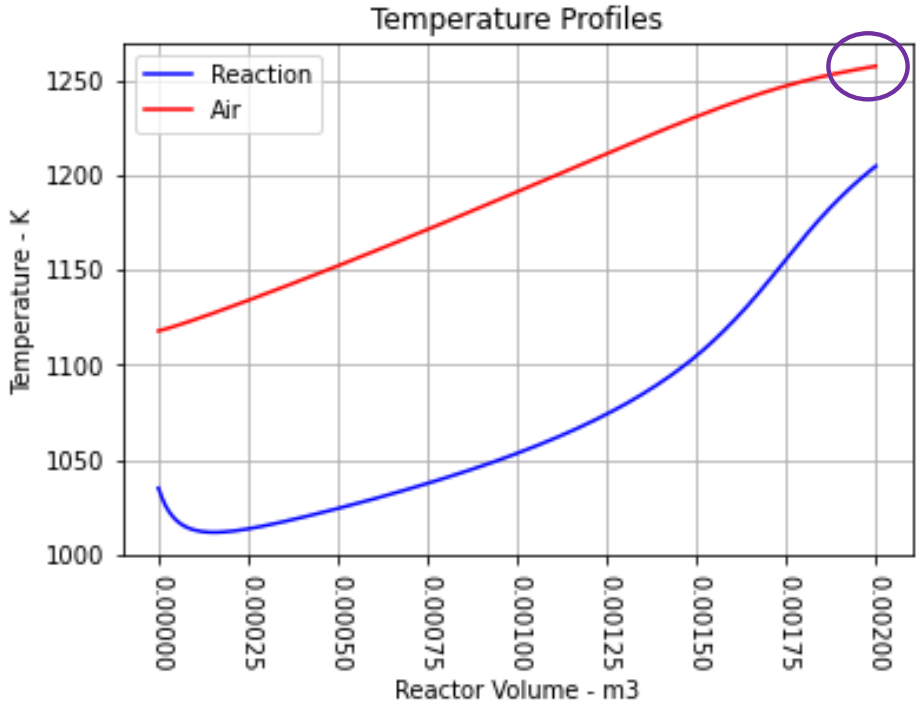




Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model



Air entry temperature of 1250K not quite met

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model – Solve two-point boundary value problem

Solv2PBV.py

```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRFn import AcetonePFR
import matplotlib.pyplot as plt
from scipy.optimize import brentq
from SolvAcetoneFn import SolvAcetone

Ta0_soln = brentq(SolvAcetone,1100,1150)

Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
Tref = 298.15  # K

MWA = 58.08  # kg/kmol
MWAir = 28.96

NoTubes = 1000
TotalVolume = 2  # m3

VolPerTube = TotalVolume/NoTubes
TubeID = 26.7e-3  # m3
TubeXC = np.pi*TubeID**2/4
TubeLength = VolPerTube/TubeXC

U = 400  # kJ/m2/hr/K
A = 4/TubeID  # m2/m3

ReactorMassFeed = 7850  # kg/hr
FAM = ReactorMassFeed/NoTubes
FAF = FAM/MWA

TF = 1035  # K
P = 162  # kPa
CAF = P/Rgas/TF  # kmol/m3

AirFeed = 11088*8  # kg/hr
FaM = AirFeed/NoTubes
Fa = FaM/MWAir
TaF = 1250  # K
Ta0 = Ta0_soln  # K - estimate

vspan = [0., VolPerTube]
veval = np.linspace(0., VolPerTube, 200)

y0 = [ FAF, TF, Ta0 ]
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model – Solve two-point boundary value problem

```
result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                   method='LSODA',args=(FAF,CAF,TF,Fa,U,A))

vs = result.t
n = len(vs)
FAout = result.y[0,:]
FKout = FAF - FAout
FMout = FAF - FAout
Tout = result.y[1,:]
Taout = result.y[2,:]

Conv = (FAF-FAout)/FAF * 100

plt.figure()
plt.plot(vs,FAout,c='k',label='Acetone')
plt.plot(vs,FKout,c='g',label='Ketene')
plt.grid()
plt.ylim(0, 0.14)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Molar Flow Rate - kmol/hr')
plt.title('Molar Flow Rates')
plt.legend()
```

```
plt.figure()
plt.plot(vs,Tout,c='b',label='Reaction')
plt.plot(vs,Taout,c='r',label='Air')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Temperature - K')
plt.title('Temperature Profiles')
plt.legend()

plt.figure()
plt.plot(vs,Conv,c='g')
plt.grid()
plt.ylim(0, 100)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')

print('\nAir Entry Temperature = {0:7.1f} K'.format(Taout[n-1]))
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model – Solve two-point boundary value problem

```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRFn import AcetonePFR

def SolvAcetone(Ta0):
    Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
    Tref = 298.15  # K

    MWA = 58.08  # kg/kmol
    MWAir = 28.96

    NoTubes = 1000
    TotalVolume = 2  # m3

    VolPerTube = TotalVolume/NoTubes
    TubeID = 26.7e-3  # m3
    TubeXC = np.pi*TubeID**2/4
    TubeLength = VolPerTube/TubeXC

    U = 400  # kJ/m2/hr/K
    A = 4/TubeID  # m2/m3

    ReactorMassFeed = 7850  # kg/hr
    FAM = ReactorMassFeed/NoTubes
    FAF = FAM/MWA
```

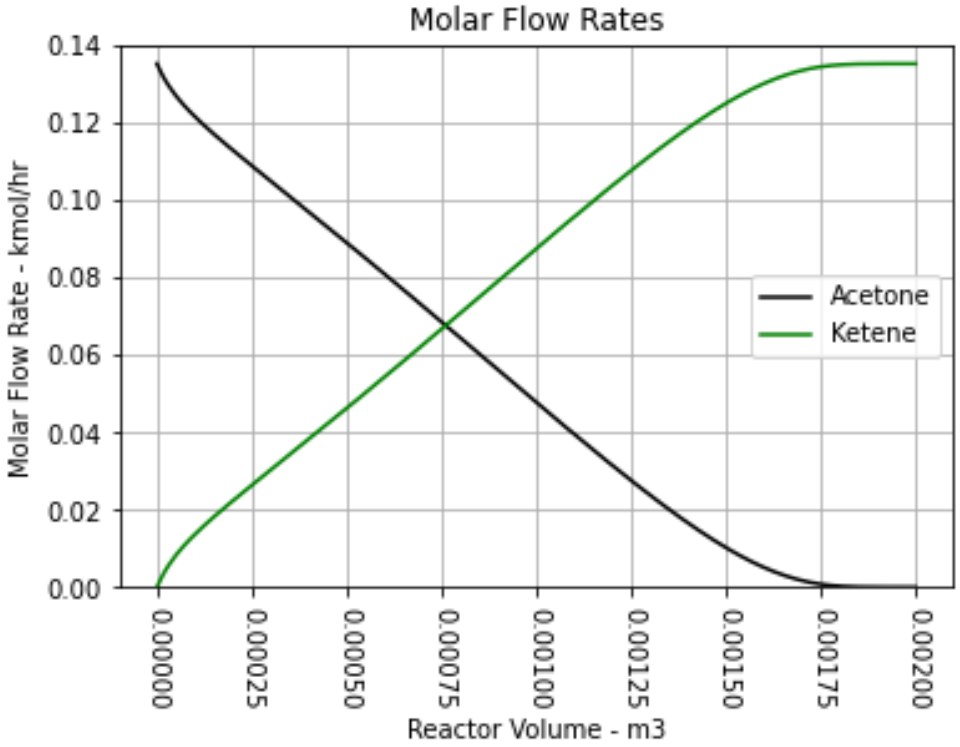
SolvAcetoneFn.py

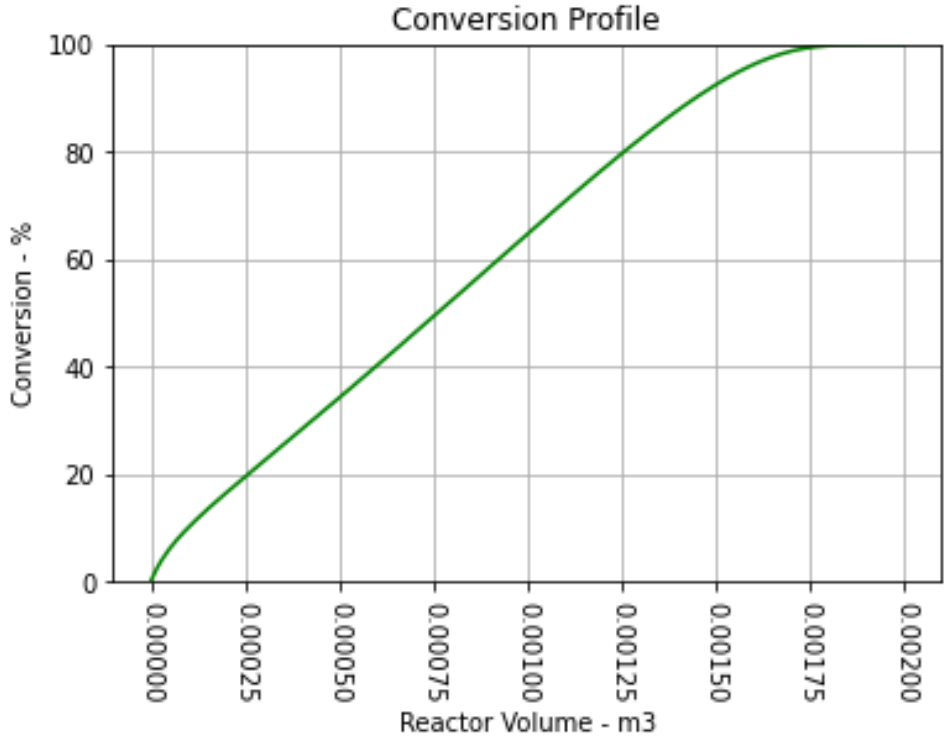
```
    TF = 1035  # K
    P = 162  # kPa
    CAF = P/Rgas/TF  # kmol/m3

    AirFeed = 11088*8  # kg/hr
    FaM = AirFeed/NoTubes
    Fa = FaM/MWAir
    TaF = 1250  # K

    vspan = [0., VolPerTube]
    veval = np.linspace(0., VolPerTube, 200)

    y0 = [ FAF, TF, Ta0 ]

    result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                       method='LSODA',args=(FAF,CAF,TF,Fa,U,A))

    vs = result.t
    n = len(vs)
    Taout = result.y[2,:]
    return Taout[n-1]-TaF
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model – Solve two-point boundary value problem - results

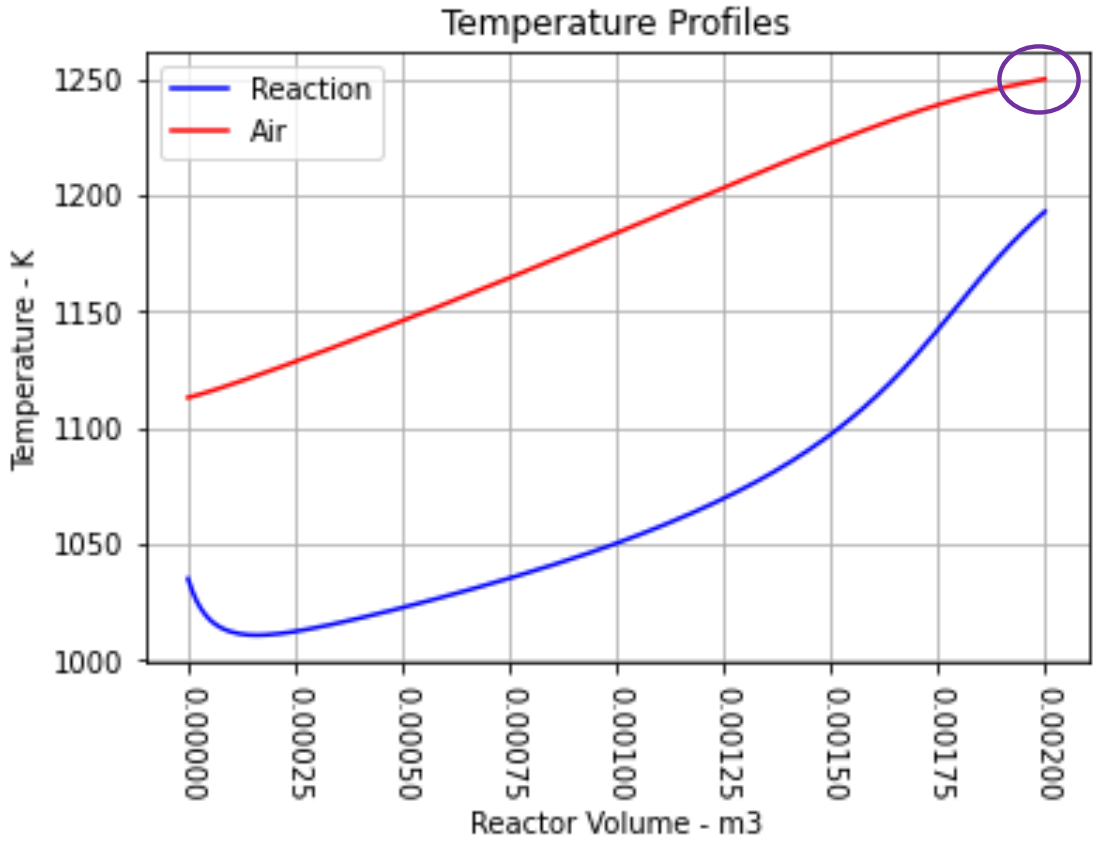




Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Simplified model – Solve two-point boundary value problem - results



Air entry temperature of 1250K now met

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

AcetonePFR_Full.py

```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRFn import AcetonePFR
import matplotlib.pyplot as plt
from CpFn import CpAavg,CpAiravg
from findTFn import findT,findTa

Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
Tref = 298.15  # K

MWA = 58.08  # kg/kmol
MWAir = 28.96

NoTubes = 1000
TotalVolume = 2  # m3

VolPerTube = TotalVolume/NoTubes
TubeID = 26.7e-3  # m3
TubeXC = np.pi*TubeID**2/4
TubeLength = VolPerTube/TubeXC

U = 400  # kJ/m2/hr/K
A = 4/TubeID  # m2/m3
```

```
ReactorMassFeed = 7850  # kg/hr
FAM = ReactorMassFeed/NoTubes
FAF = FAM/MWA

TF = 1035  # K
P = 162  # kPa
CAF = P/Rgas/TF  # kmol/m3

AirFeed = 11088*8  # kg/hr
FaM = AirFeed/NoTubes
Fa = FaM/MWAir
TaF = 1250  # K
Ta0 = 1117.7  # K - estimate

# initial conditions
HfA = -216.67*1000  # acetone heat of formation, kJ/kmol
H0 = FAF*(CpAavg(TF)*(TF-Tref)+HfA)
Ha0 = Fa*CpAiravg(Ta0)*(Ta0-Tref)

vspan = [0., VolPerTube]
veval = np.linspace(0., VolPerTube, 200)

y0 = [ FAF, H0, Ha0 ]
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

```
result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                   method='LSODA',args=(FAF,CAF,TF,Fa,Ta0,U,A))

vs = result.t
n = len(vs)
FAout = result.y[0,:]
FKout = FAF - FAout
FMout = FAF - FAout

Tout = np.zeros(n)
Hout = result.y[1,:]
for i in range(n):
    Tout[i] = findT(Hout[i],TF,FAout[i],FKout[i],FMout[i])

Haout = result.y[2,:]
Taout = np.zeros(n)
for i in range(n):
    Taout[i] = findTa(Haout[i],Ta0,Fa)

Conv = (FAF-FAout)/FAF * 100
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

```
plt.figure()
plt.plot(vs,FAout,c='k',label='Acetone')
plt.plot(vs,FKout,c='g',label='Ketene')
plt.grid()
plt.ylim(0, 0.14)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Molar Flow Rate - kmol/hr')
plt.title('Molar Flow Rates')
plt.legend()

plt.figure()
plt.plot(vs,Tout,c='b',label='Reaction')
plt.plot(vs,Taout,c='r',label='Air')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Temperature - K')
plt.title('Temperature Profiles')
plt.legend()
```

```
plt.figure()
plt.plot(vs,Conv,c='g')
plt.grid()
plt.ylim(0, 100)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')

print('\nAir Entry Temperature = {0:7.1f} K'.format(Taout[n-1]))
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

AcetonePFRFn.py

```
import numpy as np
from findTFn import findT,findTa

def AcetonePFR(v,y,FAF,CAF,TF,Fa,Ta0,U,A):
    Rgas = 8.314
    lnk0 = 42.529
    k0 = np.exp(lnk0)
    E = 284522
    FA = y[0]
    H = y[1]
    Ha = y[2]
    FK = FAF - FA
    FM = FAF - FA
    FT = FA + FK + FM
    T = findT(H,TF,FA,FK,FM)
    CA = FA/FT*CAF*TF/T
    rA = k0*np.exp(-E/Rgas/T)*CA
    Ta = findTa(Ha,Ta0,Fa)
    dy = np.zeros(3)
    dy[0] = -rA
    dy[1] = U*A*(Ta-T)
    dy[2] = U*A*(Ta-T)
    return dy
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

```
from CpFn import CpAavg,CpKavg,CpMavg,CpAiravg

def fH(H,T,FA,FK,FM):
    Tref = 298.15
    HfA = -216.67e3
    HfK = -61.09e3
    HfM = -74.81e3
    HAt = FA*(CpAavg(T)*(T-Tref)+HfA)
    HKt = FK*(CpKavg(T)*(T-Tref)+HfK)
    HMt = FM*(CpMavg(T)*(T-Tref)+HfM)
    Hout = H - (HAt+HKt+HMt)
    return Hout

def findT(H,TF,FA,FK,FM):
    T1 = TF
    tol = 1.e-6
    while True:
        T2 = T1 + 0.01
        Tnew = T1 - 0.01*fH(H,T1,FA,FK,FM) \
            / (fH(H,T2,FA,FK,FM)-fH(H,T1,FA,FK,FM))
        if abs((Tnew-T1)/Tnew) < tol: break
        T1 = Tnew
    return Tnew
```

findTFn.py

```
def fHa(Ha,Ta,Fa):
    Tref = 298.15
    Cp = CpAiravg(Ta)
    Haout = Ha - Fa*Cp*(Ta-Tref)
    return Haout

def findTa(Ha,Ta0,Fa):
    Ta1 = Ta0
    tol = 1.e-6
    while True:
        Ta2 = Ta1 + 0.01
        Tanew = Ta1 - 0.01*fHa(Ha,Ta1,Fa) \
            / (fHa(Ha,Ta2,Fa)-fHa(Ha,Ta1,Fa))
        if abs((Tanew-Ta1)/Tanew) < tol: break
        Ta1 = Tanew
    return Tanew
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Full model

CpFn.py

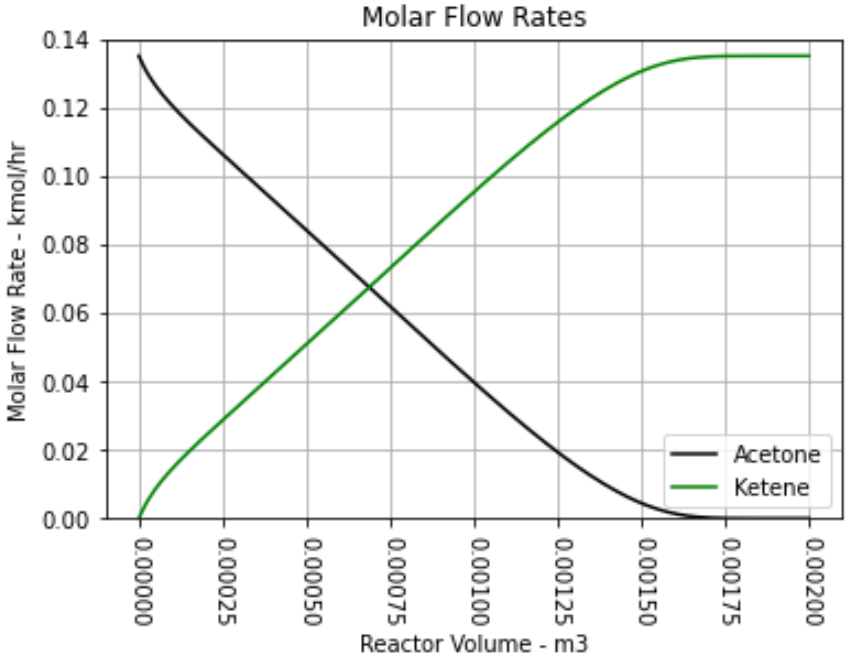
```
def CpAavg(T):
    a = 6.8132
    b = 278.6
    c = -156.28
    d = 34.76
    Tref = 298.15
    Trefk = Tref/1000
    Tk = T/1000
    CpT = a*Tk + b/2*Tk**2 + c/3*Tk**3 + d/4*Tk**4
    CpTref = a*Trefk + b/2*Trefk**2 + c/3*Trefk**3 + d/4*Trefk**4
    cpav = (CpT-CpTref)/(Tk-Trefk)
    return cpav

def CpKavg(T):
    a = 18.909
    b = 143.56
    c = -130.23
    d = 66.526
    e = -14.112
    Tref = 198.15
    Trefk = Tref/1000
    Tk = T/1000
    CpT = a*Tk + b/2*Tk**2 + c/3*Tk**3 + d/4*Tk**4 + e/5*Tk**5
    CpTref = a*Trefk + b/2*Trefk**2 + c/3*Trefk**3  \
        + d/4*Trefk**4 + e/5*Trefk**5
    cpav = (CpT-CpTref)/(Tk-Trefk)
    return cpav
```

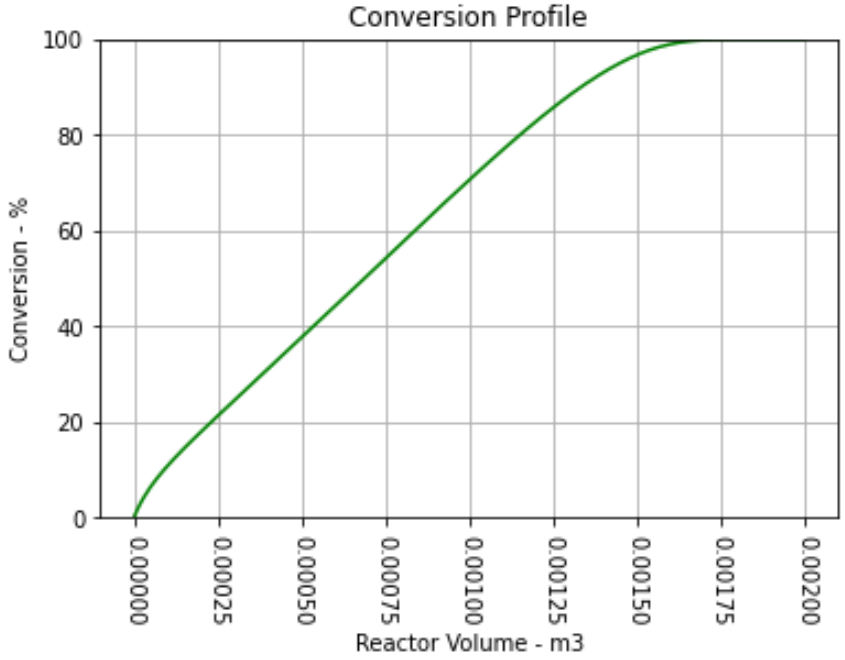
```
def CpMavg(T):
    a = -0.703029
    b = 108.4773
    c = -42.52157
    d = 5.862788
    e = 0.678565
    Tref = 198.15
    Trefk = Tref/1000
    Tk = T/1000
    CpT = a*Tk + b/2*Tk**2 + c/3*Tk**3 + d/4*Tk**4 + e/5*Tk**5
    CpTref = a*Trefk + b/2*Trefk**2 + c/3*Trefk**3  \
        + d/4*Trefk**4 + e/Trefk
    cpav = (CpT-CpTref)/(Tk-Trefk)
    return cpav

def CpAiravg(T):
    a = 28.09
    b = 0.001965
    c = 0.000004799
    d = -0.000000001965
    Tref = 298.15
    CpT = a*T + b/2*T**2 + c/3*T**3 + d/4*T**4
    CpTref = a*Tref + b/2*Tref**2 + c/3*Tref**3 + d/4*Tref**4
    cpav = (CpT-CpTref)/(T-Tref)
    return cpav
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Results for estimate of exit air temperature

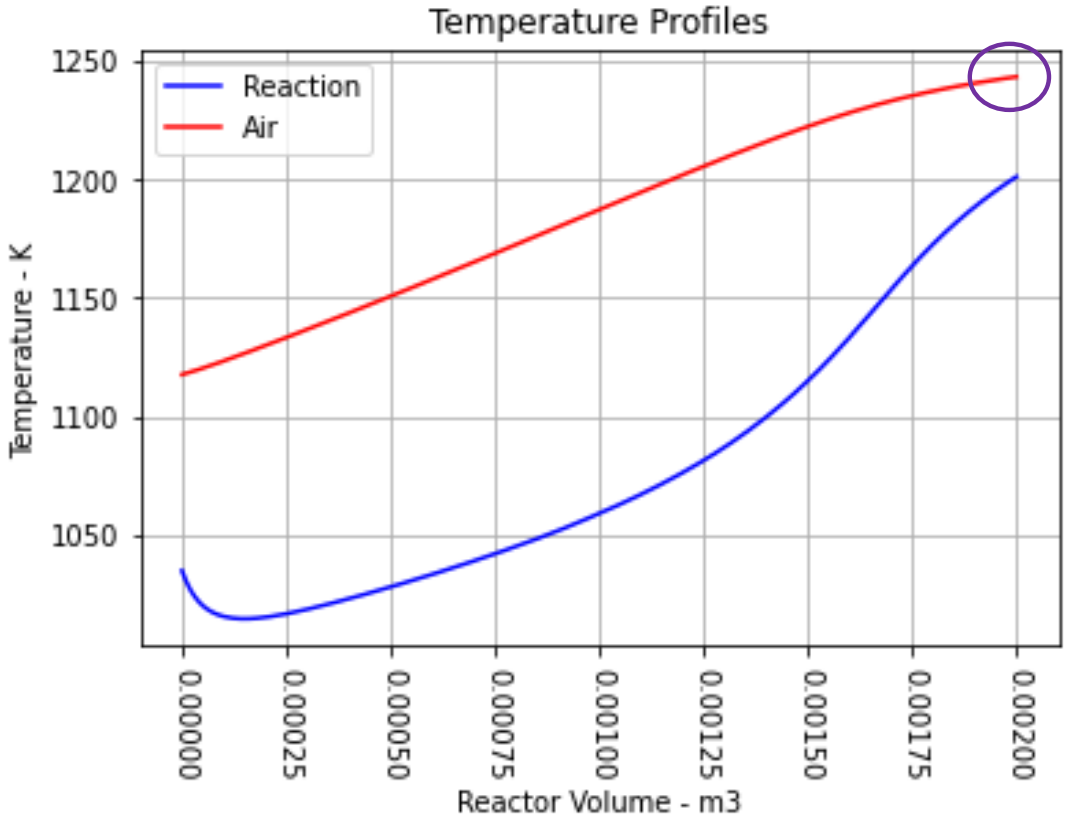




Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Results for estimate of exit air temperature – 1117.7 K



Air entry temperature of 1250K not quite met

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solve the two-point boundary value problem – full model

Solv2PBV_Full.py

```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRFn import AcetonePFR
import matplotlib.pyplot as plt
from scipy.optimize import brentq
from SolvAcetoneFullFn import SolvAcetone
from CpFn import CpAavg,CpAiravg
from findTFn import findT,findTa

Ta0_soln = brentq(SolvAcetone,1100,1150)

Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
Tref = 298.15  # K

MWA = 58.08  # kg/kmol
MWAir = 28.96

NoTubes = 1000
TotalVolume = 2  # m3

VolPerTube = TotalVolume/NoTubes
TubeID = 26.7e-3  # m3
TubeXC = np.pi*TubeID**2/4
TubeLength = VolPerTube/TubeXC

U = 400  # kJ/m2/hr/K
A = 4/TubeID  # m2/m3

ReactorMassFeed = 7850  # kg/hr
FAM = ReactorMassFeed/NoTubes
FAF = FAM/MWA

TF = 1035  # K
P = 162  # kPa
CAF = P/Rgas/TF  # kmol/m3

AirFeed = 11088*8  # kg/hr
FaM = AirFeed/NoTubes
Fa = FaM/MWAir
TaF = 1250  # K
Ta0 = Ta0_soln  # K - estimate

# initial conditions
HfA = -216.67*1000  # acetone heat of formation, kJ/kmol
H0 = FAF*(CpAavg(TF)*(TF-Tref)+HfA)
Ha0 = Fa*CpAiravg(Ta0)*(Ta0-Tref)

vspan = [0., VolPerTube]
veval = np.linspace(0., VolPerTube, 200)

y0 = [ FAF, H0, Ha0 ]
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solve the two-point boundary value problem – full model

```
result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                   method='LSODA',args=(FAF,CAF,TF,Fa,Ta0,U,A))

vs = result.t
n = len(vs)
FAout = result.y[0,:]
FKout = FAF - FAout
FMout = FAF - FAout

Tout = np.zeros(n)
Hout = result.y[1,:]
for i in range(n):
    Tout[i] = findT(Hout[i],TF,FAout[i],FKout[i],FMout[i])

Haout = result.y[2,:]
Taout = np.zeros(n)
for i in range(n):
    Taout[i] = findTa(Haout[i],Ta0,Fa)

Conv = (FAF-FAout)/FAF * 100
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solve the two-point boundary value problem – full model

```
plt.figure()
plt.plot(vs,FAout,c='k',label='Acetone')
plt.plot(vs,FKout,c='g',label='Ketene')
plt.grid()
plt.ylim(0, 0.14)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Molar Flow Rate - kmol/hr')
plt.title('Molar Flow Rates')
plt.legend()

plt.figure()
plt.plot(vs,Tout,c='b',label='Reaction')
plt.plot(vs,Taout,c='r',label='Air')
plt.grid()
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Temperature - K')
plt.title('Temperature Profiles')
plt.legend()
```

```
plt.figure()
plt.plot(vs,Conv,c='g')
plt.grid()
plt.ylim(0, 100)
plt.xlabel('Reactor Volume - m3')
plt.tick_params(axis='x',labelrotation=-90)
plt.ylabel('Conversion - %')
plt.title('Conversion Profile')

print('\nAir Entry Temperature = {0:7.1f} K'.format(Taout[n-1]))
```

Tubular Reactor with Counter-current Heat Exchange

Solve the two-point boundary value problem – full model

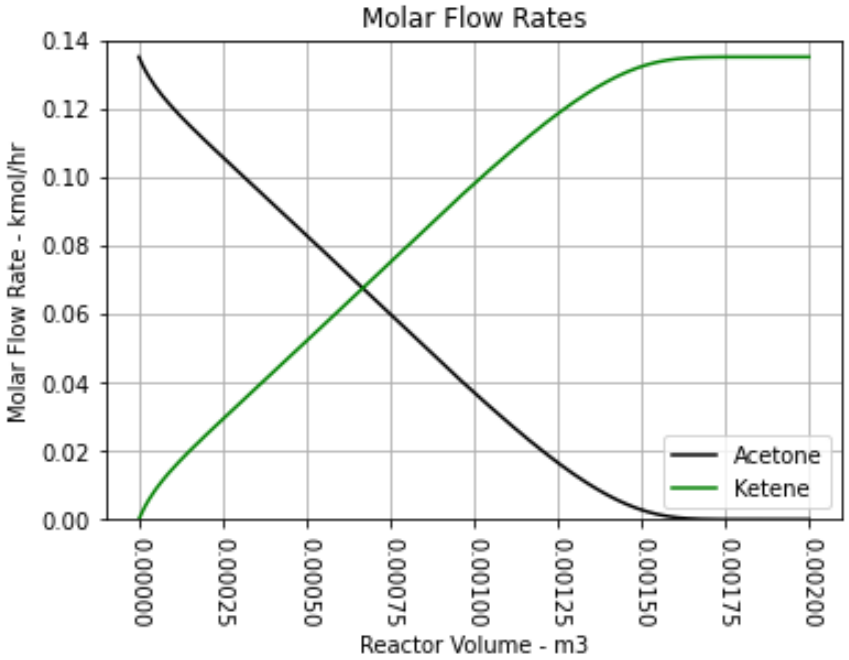
SolvAcetoneFullFn.py

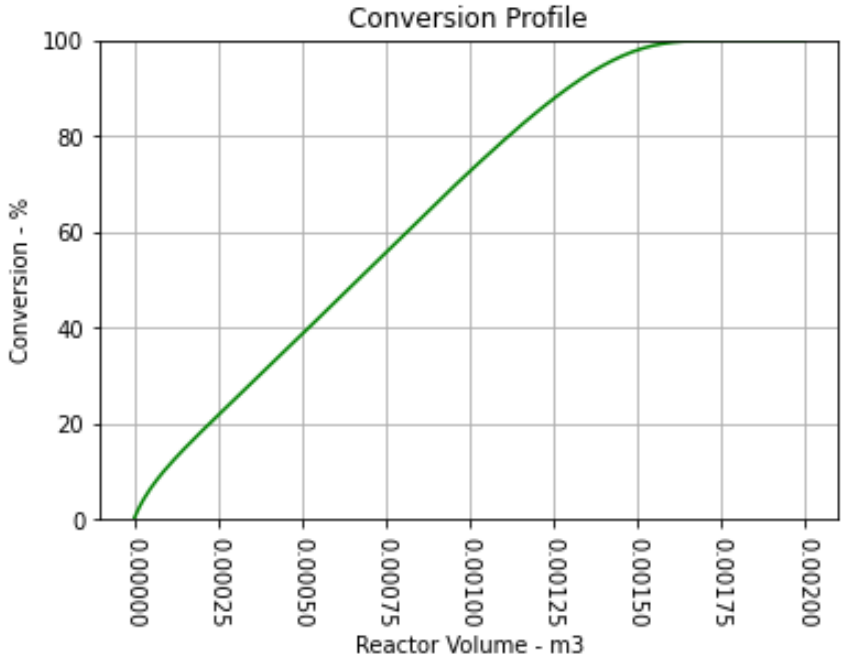
```
import numpy as np
from scipy.integrate import solve_ivp
from AcetonePFRFn import AcetonePFR
from CpFn import CpAavg,CpAiravg
from findTFn import findTa

def SolvAcetone(Ta0):
    Rgas = 8.314  # kJ/kmol/K also m3*kPa/kmol/K
    Tref = 298.15  # K

    MWA = 58.08  # kg/kmol
    MWAir = 28.96

    NoTubes = 1000
    TotalVolume = 2  # m3

    VolPerTube = TotalVolume/NoTubes
    TubeID = 26.7e-3  # m3

    U = 400  # kJ/m2/hr/K
    A = 4/TubeID  # m2/m3

    ReactorMassFeed = 7850  # kg/hr
    FAM = ReactorMassFeed/NoTubes
    FAF = FAM/MWA

    TF = 1035  # K
    P = 162  # kPa
    CAF = P/Rgas/TF  # kmol/m3

    AirFeed = 11088*8  # kg/hr
    FaM = AirFeed/NoTubes
    Fa = FaM/MWAir
    TaF = 1250  # K

    # initial conditions
    HfA = -216.67*1000  # acetone heat of formation, kJ/kmol
    H0 = FAF*(CpAavg(TF)*(TF-Tref)+HfA)
    Ha0 = Fa*CpAiravg(Ta0)*(Ta0-Tref)

    vspan = [0., VolPerTube]
    veval = np.linspace(0., VolPerTube, 200)

    y0 = [ FAF, H0, Ha0 ]

    result = solve_ivp(AcetonePFR,vspan,y0,t_eval=veval, \
                       method='LSODA',args=(FAF,CAF,TF,Fa,Ta0,U,A))

    vs = result.t
    n = len(vs)

    Haout = result.y[2,:]
    Taout = np.zeros(n)
    for i in range(n):
        Taout[i] = findTa(Haout[i],Ta0,Fa)

    return Taout[n-1]-TaF
```

Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solve the two-point boundary value problem – full model

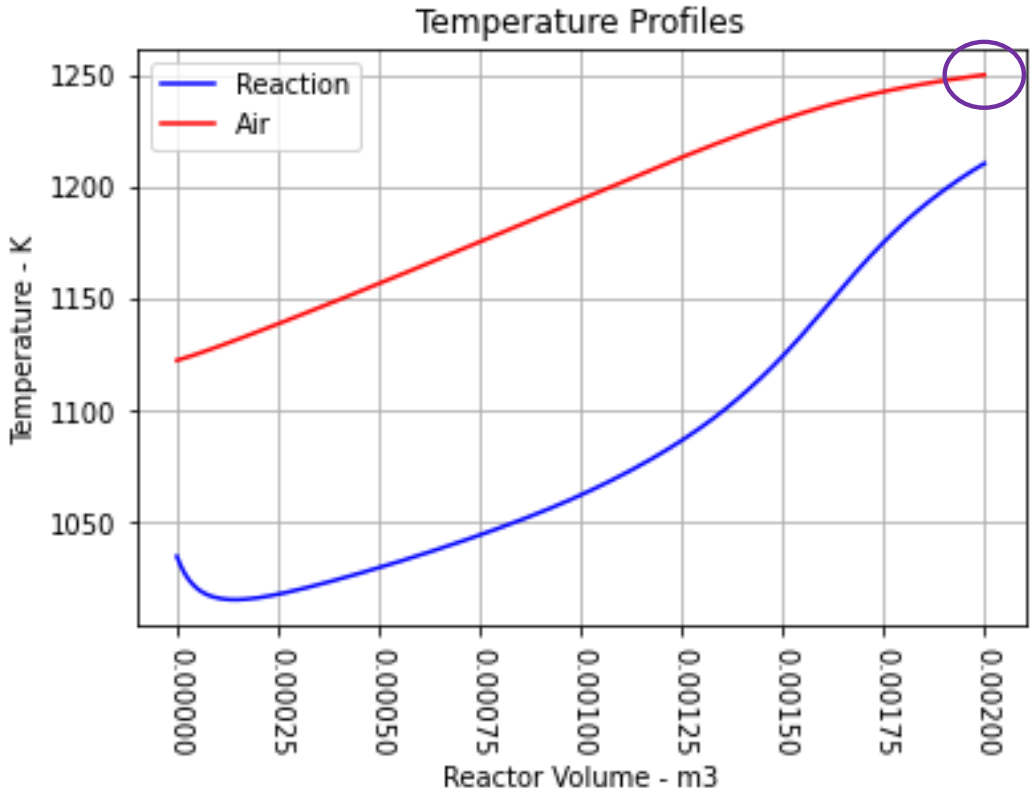




Tubular Reactor with Counter-current Heat Exchange

Example: Vapor-phase cracking of acetone to ketene

Solve the two-point boundary value problem – full model



Air entry temperature of 1250K now met

Python Bootcamps 1, 2 and 3

- ✓ 1: Getting up to speed with Python
- ✓ 2: Learning to use Python to solve typical problem scenarios
- ✓ 3: Detailed modeling of packed-bed and plug-flow reactors

References:

Applied Numerical Methods with Python
Steven C. Chapra and David E. Clough
McGraw-Hill, 2022.

Introduction to Engineering and Scientific Computing with Python
David E. Clough and Steven C. Chapra
CRC Press, Taylor & Francis, 2023.

Elements of Chemical Reaction Engineering, 4th Edition
Fogler, H. Scott,, Prentice-Hall, 2006.

Contact **David.Clough@Colorado.edu** for follow-up assistance.

"Prof. Clough, may I be excused? My brain is full."