1 Lecture 0: Intro to Reaction Engineering

1. Reaction engineering

   Understanding, modeling, designing, using, controlling, analyzing, improving anything in which chemical reactions happen.

1. Reaction engineering applications
   (a) Traditional
      i. Industrial chemical/petroleum processes
      ii. Fine chemical/pharmaceutical processes
      iii. Emerging, eg biorefinergy, shale gas, http://cistar.us
   (b) Energy storage, batteries, fuel cells
   (c) Environmental systems
      i. Atmosphere, lake, bioreactor (water purification), catalytic convertor
   (d) Biological systems
      i. Cell, organ, body
   (e) Laboratory reactors - interrogate, quantify
   (f) Research - improved materials (catalysts), improved processes, understand limitations
      i. Sabatier plot, https://doi.org/10.1038/nchem.121

2. Course structure
   (a) Quantifying chemical reactions
      i. Stoichiometry
      ii. Thermodynamics - heat flow, direction, equilibrium
      iii. Kinetics - rates, mechanisms
   (b) Physical/chemical interactions
      i. Transport, mixing, diffusion resistance, ...
   (c) Chemical reactors
      i. Ideal 0 and 1-dimensional
      ii. Non-ideal
      iii. Non-isothermal
      iv. Non-steady state
      v. Multiphase
   (d) Chemical processes (beyond us)
   (e) Markets (beyond us)

2 Stoichiometry and reactions

1. Substances
2. Amounts
   (a) mass, moles, volumes
   (b) flow rates
3. compositions
   (a) amount/total amount
4. Reactions and stoichiometric coefficients
   (a) Advancements $n_j = \sum_i \nu_{ij}\xi_i$
   (b) Limiting reagents

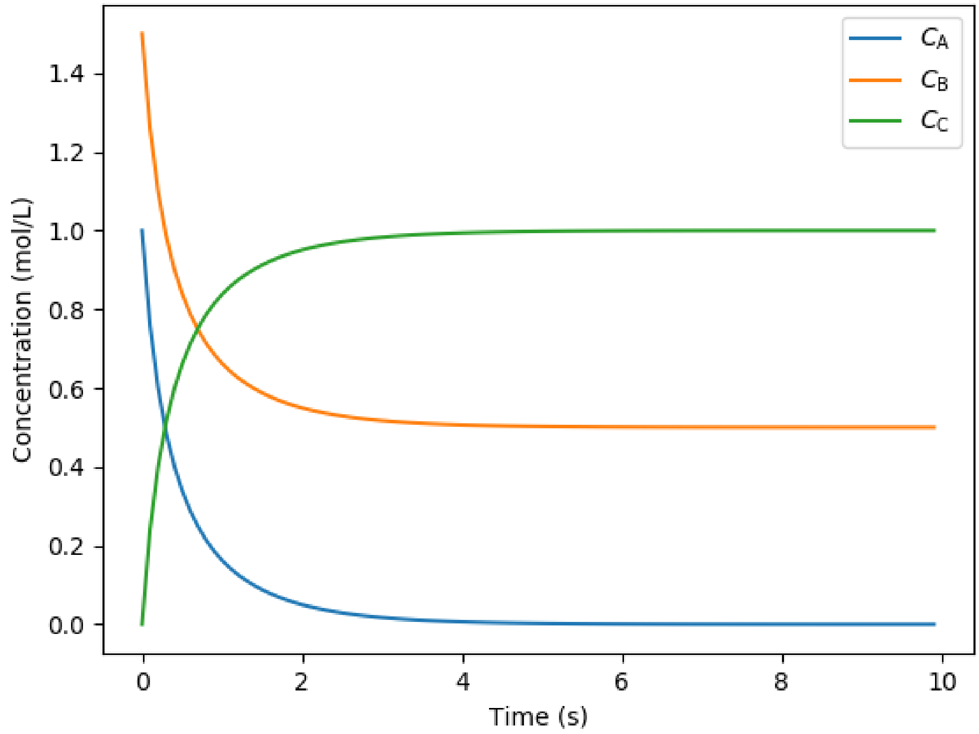
3 Chemical thermodynamics and equilibria

1. Chemical reactions $\sum_j \nu_j A_j = 0$
2. Thermodynamic potential differences
   (a) Standard states
   (b) Formation reactions
   (c) Reaction enthalpy $\Delta H^\circ(T) = \sum_j {}_j H_j^\circ(T) = \sum_j \nu_j H_{f,j}^\circ(T)$
   (d) Reaction entropy $\Delta S^\circ(T) = \sum_j {}_j S_j^\circ(T)$
3. Equilibrium-closed system
   (a) Free energy vs reaction advancement, $G(\xi, T) = \sum_j n_j\mu_j = \sum_j (n_{j0} + \nu_j\xi)\left(\mu_j^\circ(T) + RT\ln a(\xi, T)\right)$
   (b) Equilibrium $(\partial G/\partial\xi)_{T,P} = 0$
   (c) Equilibrium constants and algebraic solutions
   (d) Multiple reactions
4. Le'Chatlier principle - system at equilibrium responds to oppose any perturbation
   (a) Pressure, composition
   (b) Temperature: Gibbs-Helmholtz and van't Hoff
5. Equilibrium-open system
   (a) Reaction phase diagrams, see http://pubs.acs.org/doi/abs/10.1021/jacs.6b02651 for an example
   (b) Electrochemical reactions
6. The molecular interpretation
7. Non-ideal activities
8. Surface adsorption
   (a) Langmuir

4 Empirical kinetics

1. rates
2. reactor mass balance
3. rate expressions
4. rate orders
5. apparent orders
6. integrated rate expressions
7. temperature and Arrhenius expression

```
import numpy as np                #this lets up handles arrays of data
import matplotlib.pyplot as plt
from scipy.integrate import odeint, solve_ivp

def dCdt(C,t,k):
    dC_Adt = -k*C[0]*C[1]     # A + B -> C;  r = k CA CB
    dC_Bdt = -k*C[0]*C[1]
    dC_Cdt =  k*C[0]*C[1]
    dCdt = [dC_Adt,dC_Bdt,dC_Cdt]
    return dCdt

# initialize concentrations
C_0 = [1., 1.5, 0.]

# initialize k's
k = 2.

# Range of time to solve over
t = np.arange(0,10,0.1)
t_span = (0., 10.)

p = (k,) # turn parameters into a tuple
# Solve two ODEs with odeint
#C = solve_ivp(dCdt,t_span,C_0,p,method='LSODA')
C = odeint(dCdt,C_0,t,p)

C_A = C.transpose()[0] # Get C_A from C
C_B = C.transpose()[1] # Get C_B from C
C_C = C.transpose()[2]
plt.figure()
plt.plot(t,C_A,'-',label=r'$C_{\rm A}$')
plt.plot(t,C_B,'-',label=r'$C_{\rm B}$')
plt.plot(t,C_C,'-',label=r'$C_{\rm C}$')
plt.xlabel('Time (s)')
plt.ylabel('Concentration (mol/L)')
plt.legend()
plt.savefig('./conc.png')
```



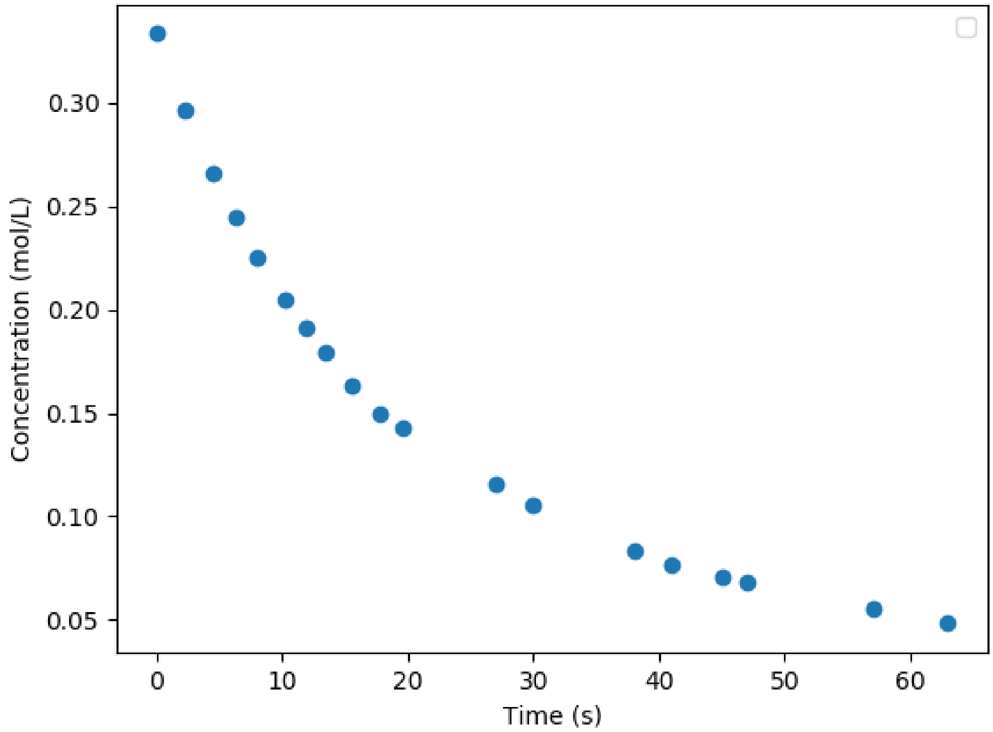
5 Analyzing reactor data

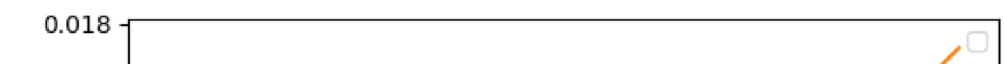
1. Differential methods
   (a) Measuring rates
2. Integral methods
3. Half-lives

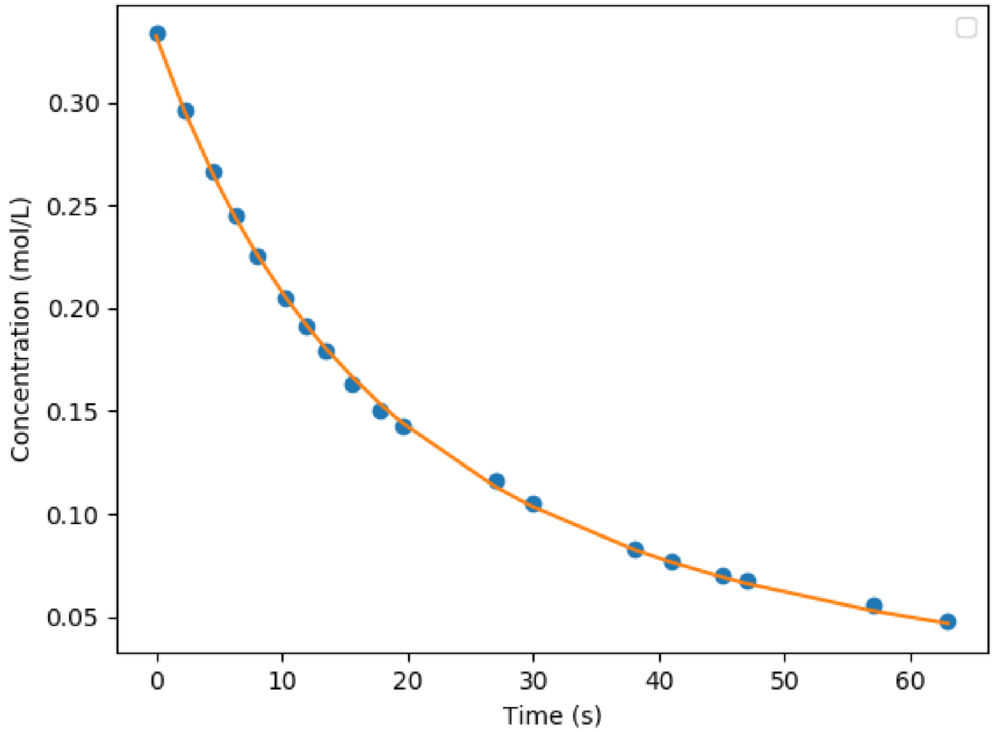
```
import numpy as np                #this lets up handles arrays of data
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def differential(x, k, alpha):
    return k*x**alpha

def integral(t, a, b):
    return (2*a/(2+a*b*t))**2

t = np.array([0.00, 2.25, 4.50, 6.33, 8.00, 10.25, 12.00, 13.50, 15.60, 17.85, 19.60, 27.00, 30.00, 38.00, 41.00, 45.00, 47.00,

C_Br2 = np.array([0.3335, 0.2965, 0.2660, 0.2450, 0.2255, 0.2050, 0.1910, 0.1794, 0.1632, 0.1500, 0.1429, 0.1160, 0.1053, 0.0830

plt.figure()
plt.plot(t,C_Br2,'o')
plt.xlabel('Time (s)')
plt.ylabel('Concentration (mol/L)')
plt.legend()
plt.savefig('./xylene-conc.png')
```

```

delta_t = np.ediff1d(t)        # finite difference between adjacent points
delta_C = np.ediff1d(C_Br2)

grad_t = np.gradient(t)            # second order approximation to gradient, allowing for unequal step size
grad_C = np.gradient(C_Br2)
rate = -np.divide(grad_C,grad_t)

plt.figure()
plt.plot(C_Br2,rate,'o')
plt.xlabel('Concentration (mol/L)')
plt.ylabel('Rate (mol/L/x)')
plt.legend()

popt, pcov = curve_fit(differential, C_Br2, rate )

print('k = {0:f}, alpha={1:f}'.format(popt[0],popt[1]))

model = differential(C_Br2,popt[0],popt[1])
plt.plot(C_Br2,model,'-')

plt.savefig('./xylene-rate.png')

difference_array = np. subtract(rate, model)
squared_array = np. square(difference_array)
mse = squared_array. mean()
print(mse)

# Suggests order of 1.5
popt1, pcov1 = curve_fit(integral, t, C_Br2)
print('k = {0:f}'.format(popt[1]))

model1 = integral(t, popt1[0], popt1[1])

plt.figure()
plt.plot(t,C_Br2,'o')
plt.plot(t,model1,'-')
plt.xlabel('Time (s)')
plt.ylabel('Concentration (mol/L)')
plt.legend()
plt.savefig('./xylene-int-model.png')
```

```
k = 0.085277, alpha=1.450860
2.4942019231742367e-07
k = 1.450860
```


Concentration (mol/L)
0.30
0.25
0.20
0.15
0.10
0.05
0
10
20
30
40
50
60
Time (s)


Rate (mol/L/x)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.05
0.10
0.15
0.20
0.25
0.30
Concentration (mol/L)



6 Molecular basis

1. reaction pathway, detailed balance
2. bimolecular, collision theory, TST
3. unimolecular reactions

7 Mechanisms

1. simple reaction network
2. free energy surface
3. QSSA
4. Pre-equilibrium
5. Selectivity
6. Rate control

Table 1: Equilibrium and Rate Constants

Equilibrium Constants a A + b B $\rightleftharpoons$ c C + d D

$$\begin{aligned}
K_{eq}(T) &= e^{\Delta S^\circ(T,V)/k_B} e^{-\Delta H^\circ(T,V)/k_B T} \\
K_c(T) &= \left(\frac{1}{c^\circ}\right)^{\nu_c+\nu_d-\nu_a-\nu_b} \frac{(q_c/V)^{\nu_c}(q_d/V)^{\nu_d}}{(q_a/V)^{\nu_a}(q_b/V)^{\nu_b}} e^{-\Delta E(0)\beta} \\
K_p(T) &= \left(\frac{k_B T}{P^\circ}\right)^{\nu_c+\nu_d-\nu_a-\nu_b} \frac{(q_c/V)^{\nu_c}(q_d/V)^{\nu_d}}{(q_a/V)^{\nu_a}(q_b/V)^{\nu_b}} e^{-\Delta E(0)\beta}
\end{aligned}$$

Unimolecular Reaction $[A] \rightleftharpoons [\mathrm{A}]^\ddagger \rightarrow C$

$$k(T) = \nu^\ddagger \bar{K}^\ddagger = \frac{k_B T}{h} \frac{\bar{q}_\ddagger(T)/V}{q_A(T)/V} e^{-\Delta E^\ddagger(0)\beta}$$

$$E_a = \Delta H^{\circ\ddagger} + k_B T \quad A = e^1 \frac{k_B T}{h} e^{\Delta S^{\circ\ddagger}}$$

Bimolecular Reaction $\mathrm{A} + \mathrm{B} \rightleftharpoons [\mathrm{AB}]^\ddagger \rightarrow \mathrm{C}$

$$k(T) = \nu^\ddagger \bar{K}^\ddagger = \frac{k_B T}{h} \frac{q_\ddagger(T)/V}{(q_A(T)/V)(q_B(T)/V)} \left(\frac{1}{c^\circ}\right)^{-1} e^{-\Delta E^\ddagger(0)\beta}$$

$$E_a = \Delta H^{\circ\ddagger} + 2k_B T \quad A = e^2 \frac{k_B T}{h} e^{\Delta S^{\circ\ddagger}}$$

8 Heterogeneous reactions

1. adsorption, L-H
2. TPD
3. catalysis
4. Sabatier analysis

9 Liquid-phase reactions

