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 Lecture 1: 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 Lecture 2: 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

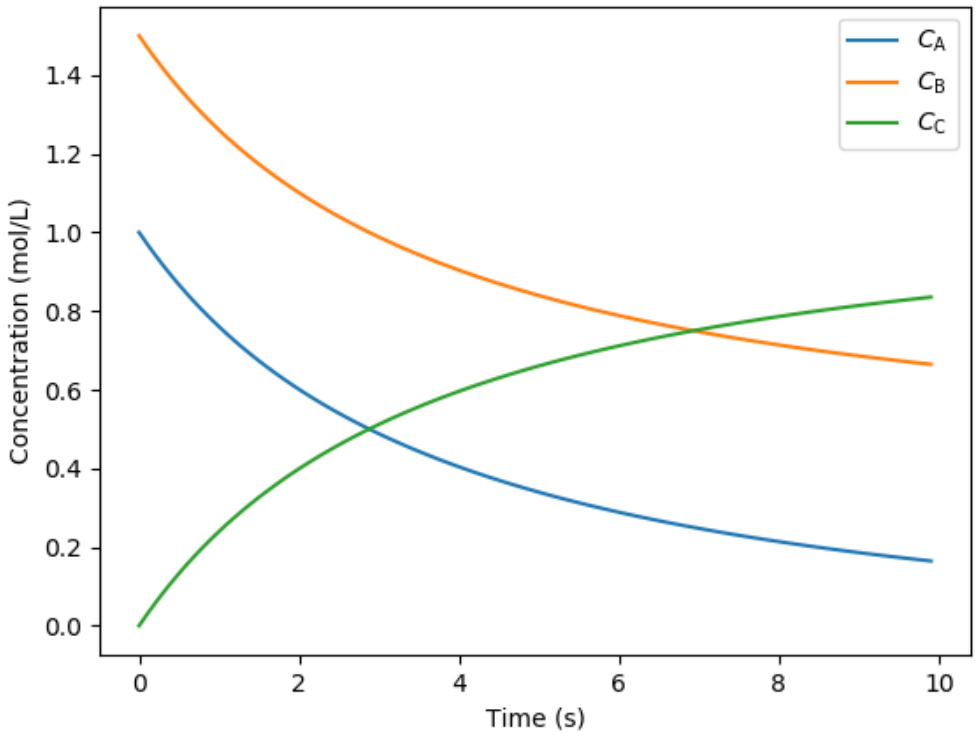
4 Lecture 3: Empirical kinetics

1. rates: number per unit time per unit something
2. reactor mass balance
3. rate expressions, Functions of T, P, composition C_i
4. rate orders
5. apparent orders
6. integrated rate expressions
7. temperature and Arrhenius expression, $k = Ae^{-E_a/k_BT}$
   (a) Arrhenius plot, $\ln k$ vs $1/T$

Table 1: Basic kinetic rate laws

	differential rate	integrated rate	half-life
First order	$r = kC_A$	$C_A = C_{A0}e^{-k\tau}$	$\ln 2/k$
Second order	$r = kC_A^2$	$1/C_A = 1/C_{A0} + k\tau$	$1/kC_{A0}$

```
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 = 0.2

# 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 Lecture 4: Analyzing reactor data

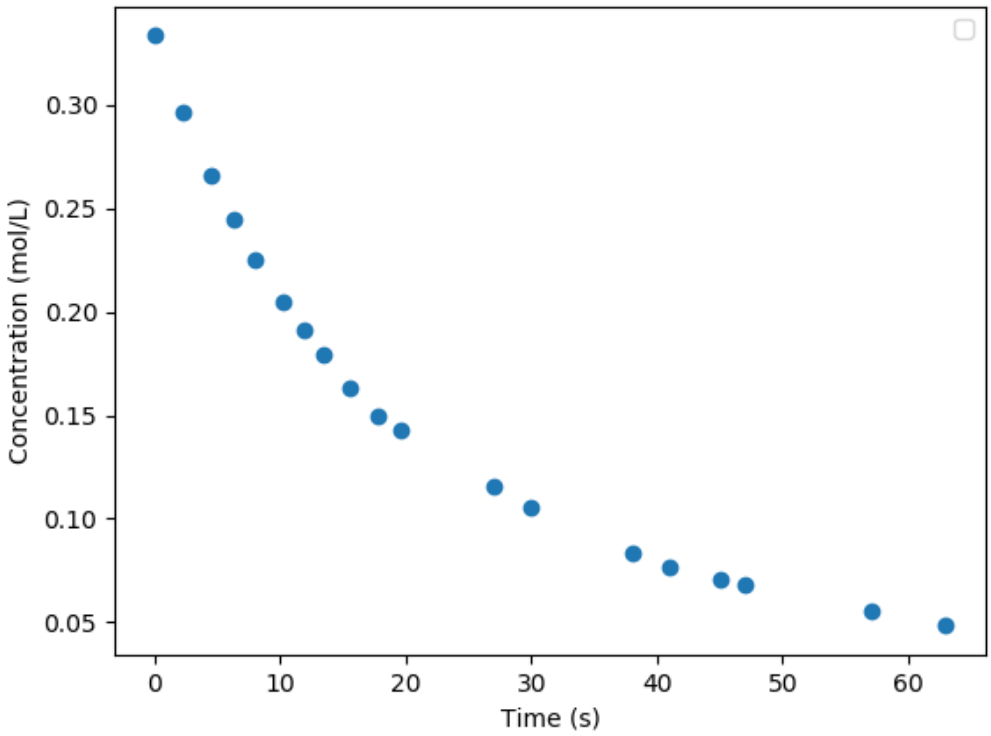
1. Differential methods
   (a) Measuring rates
2. Integral methods
3. Half-lives

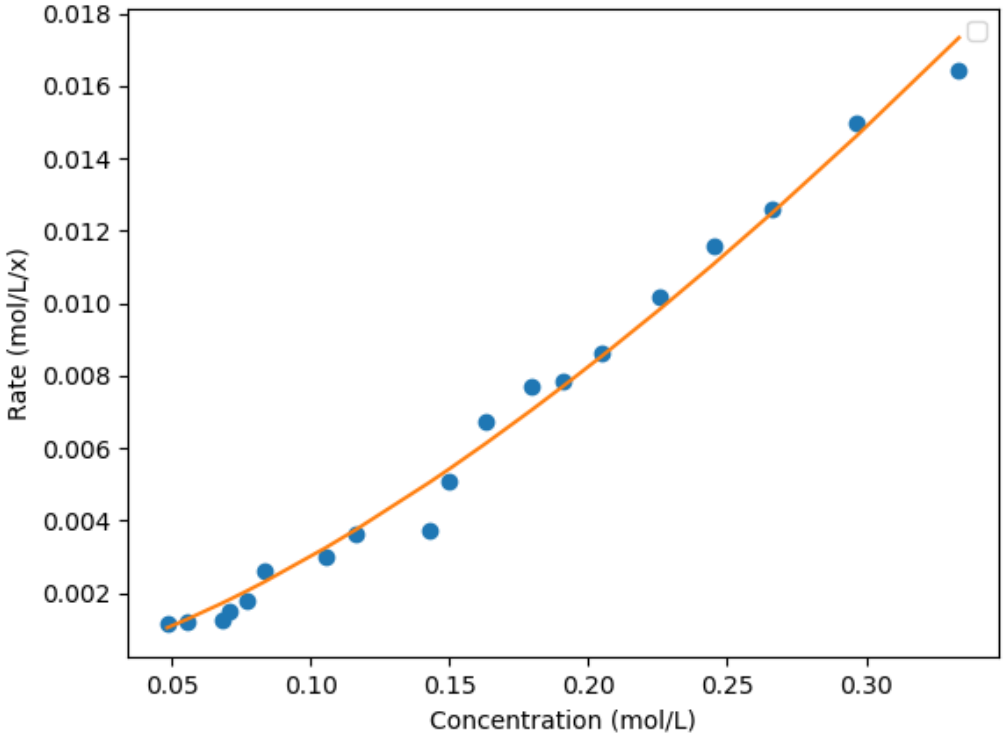
```
import numpy as np                #this lets up handles arrays of data
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

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')
```

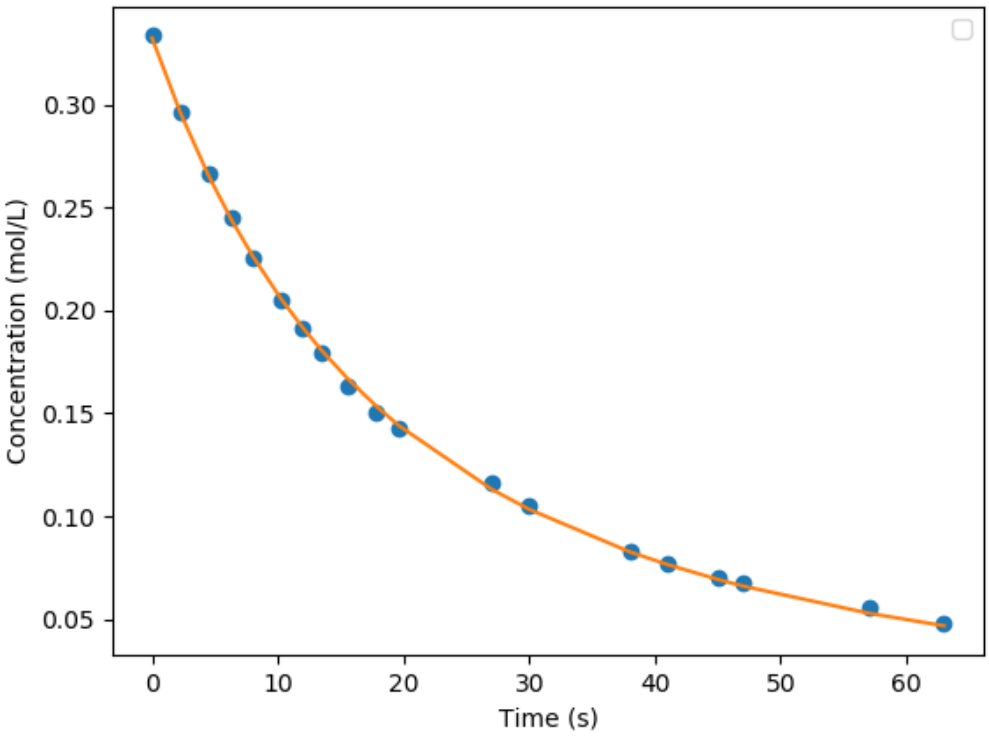


```
21
22 delta_t = np.ediff1d(t)        # finite difference between adjacent points
23 delta_C = np.ediff1d(C_Br2)
24
25 grad_t = np.gradient(t)            # second order approximation to gradient, allowing for unequal step size
26 grad_C = np.gradient(C_Br2)
27 rate = -np.divide(grad_C,grad_t)
28
29 plt.figure()
30 plt.plot(C_Br2,rate,'o')
31 plt.xlabel('Concentration (mol/L)')
32 plt.ylabel('Rate (mol/L/x)')
33 plt.legend()
34
35 popt, pcov = curve_fit(differential, C_Br2, rate )
36
37 print('k = {0:f}, alpha={1:f}'.format(popt[0],popt[1]))
38
39 model = differential(C_Br2,popt[0],popt[1])
40 plt.plot(C_Br2,model,'-')
41
42 plt.savefig('./xylene-rate.png')
43
44 difference_array = np. subtract(rate, model)
45 squared_array = np. square(difference_array)
46 mse = squared_array. mean()
47 print(mse)
48
49 # Suggests order of 1.5
50 popt1, pcov1 = curve_fit(integral, t, C_Br2)
51 print('k = {0:f}'.format(popt[1]))
52
53 model1 = integral(t, popt1[0], popt1[1])
54
55 plt.figure()
56 plt.plot(t,C_Br2,'o')
57 plt.plot(t,model1,'-')
58 plt.xlabel('Time (s)')
59 plt.ylabel('Concentration (mol/L)')
60 plt.legend()
61 plt.savefig('./xylene-int-model.png')
```

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





6 Lecture 5: Molecular chemical kinetics

6.0.1 Reaction mechanisms

1. Elementary steps and molecularity
2. Ozone decomposition, rate second-order at high P_{O_2}, first-order at low P_{O_2}

$$\begin{array}{l} \underline{2O_3 \longrightarrow 3O_2} \\ O_3 \xrightarrow{k_1} O_2 + O \\ O_2 + O \xrightarrow{k_{-1}} O_3 \\ O + O_3 \xrightarrow{k_2} 2O_2 \end{array}$$

3. Detailed balance and microscopic reversibility
4. Equilibrium requirement $K_{eq}(T) = k_f(T)/k_r(T)$
5. Reversibility $r_{\text{net}} = r_f(1-\beta)$, $\beta = Q/K_c = \exp(-\Delta G(T, c_j)/RT)$
   (a) A $\rightleftharpoons$ B example
6. Collision theory
   (a) A + B $\rightarrow$ products

(b) rate proportional to A/B collision frequency z_{AB} weighted by fraction of collisions with energy $> E_a$

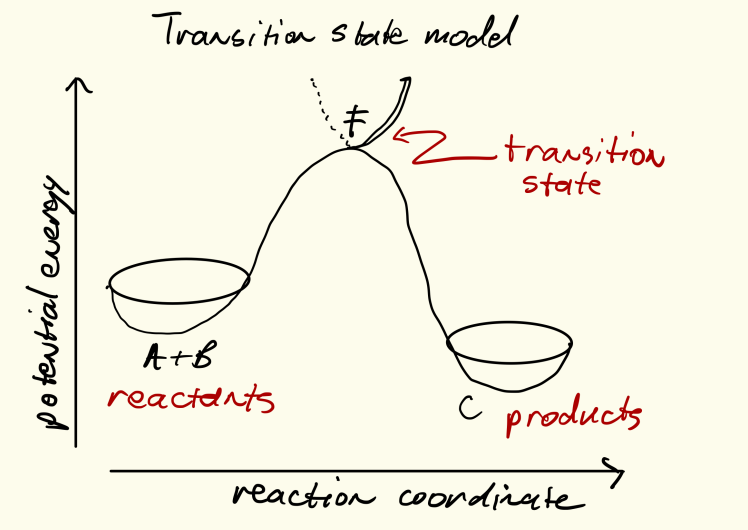
$$r = kC_AC_B, k = \left(\frac{8k_BT}{\pi\mu}\right)^{1/2} \sigma_{AB}N_{av}e^{-E_a/k_BT}$$

(c) upper bound on real rates

6.0.2 Transition state theory (TST)

1. Assumptions
   (a) Existence of reaction coordinate (PES)
   (b) Existence of dividing surface
   (c) Equilibrium between reactants and "transition state"
   (d) Harmonic approximation for transition state
2. rate proportional to concentration of "activated complex" over reactants times crossing frequency

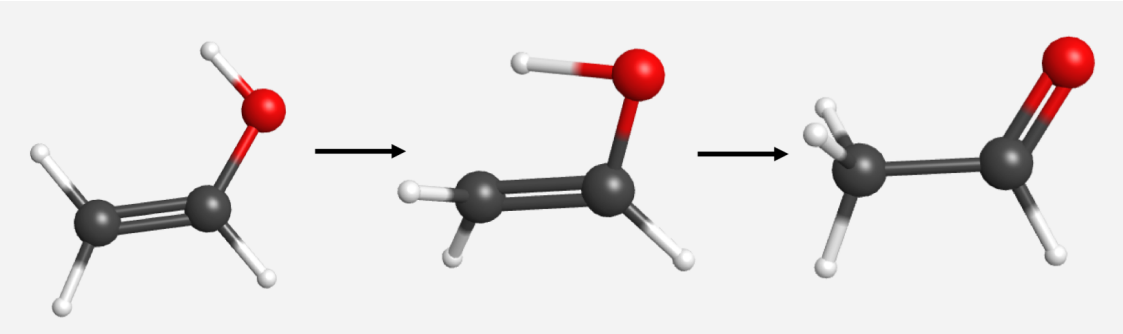
$$\begin{aligned} r &= kC_AC_B \\ &= k^\ddagger C_{AB}^\ddagger \\ &= \nu^\ddagger K^\ddagger C_AC_B \\ &= \nu^\ddagger \frac{k_BT}{h\nu^\ddagger}\bar{K}^\ddagger(T)C_AC_B \\ &= \frac{k_BT}{h}\frac{q^\ddagger(T)}{q_A(T)q_B(T)}e^{-\Delta E(0)/k_BT}C_AC_B \end{aligned}$$



6.0.3 Locating transition states computationally

1. Reactants/products are minima on potential energy surface
2. Transition state is first order saddle point. Unique point on pathway from reactant to product valley

3. vinyl alcohol to acetaldehyde example
4. https://www.webmo.net



6.0.4 Thermodynamic connection

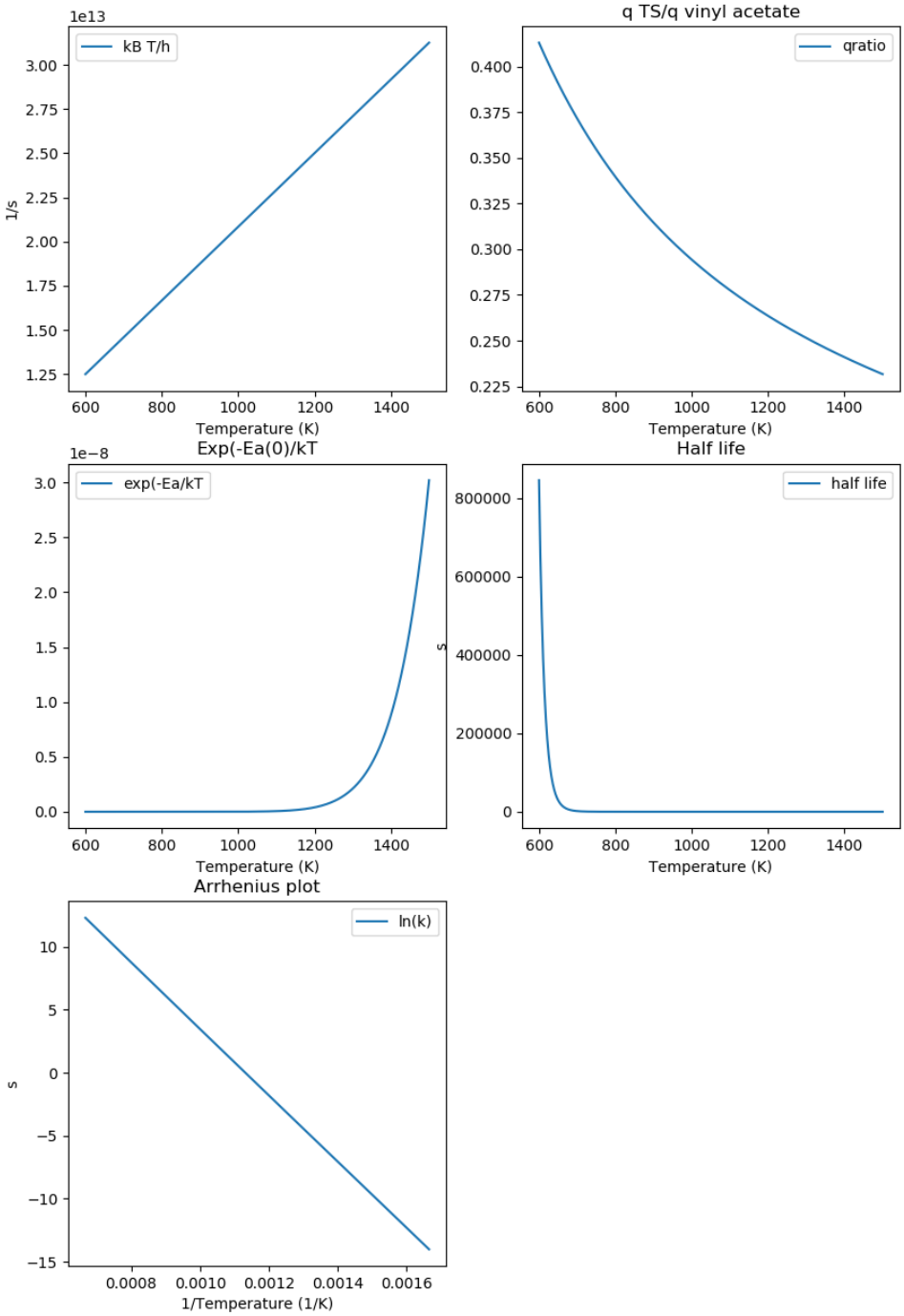
1. Relate activated complex equilibrium constant to activation free energy (isochoric standard state)

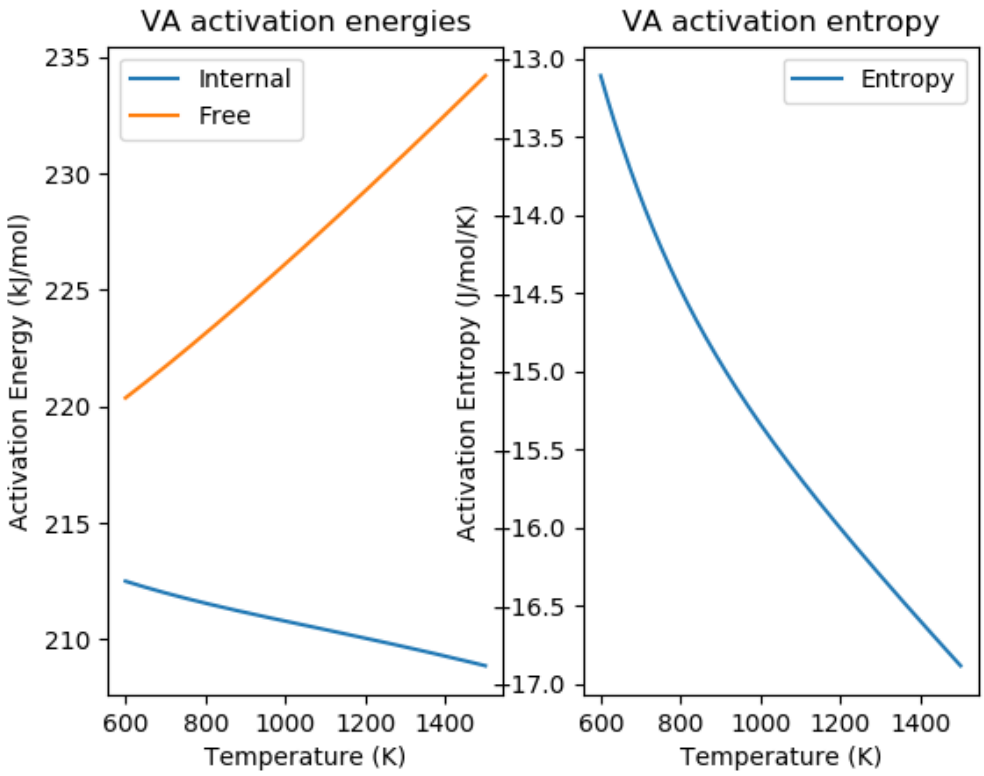
$$\bar{K}^{\ddagger}(T) = e^{-\Delta A^{\circ\ddagger}(T)/kT} = e^{-\Delta U^{\circ\ddagger}(T)/k_B T} e^{\Delta S^{\circ\ddagger}(T)/k_B}$$

2. Compare to Arrhenius expression

$$E_a = \Delta U^{\circ\ddagger}(T) + kT, A = \frac{k_B T}{h} e^1 e^{\Delta S^{\circ\ddagger}(T)/k_B}$$

```
Vinyl alcohol to TS  216 kJ/mol
Delta Uddagger (1000 K) =  211 kJ/mol
Delta Addagger (1000 K) =  226 kJ/mol
Delta Sddagger (1000 K) = -1480 J/mol K
```



6.0.5 Bimolecular reaction

1. Diels-Alder example

```
import numpy as np

kB = 8.61733e-5       # eV /K
h = 4.13566766e-15  # eV s
eVtokJ = 96.485332
Nav = 6.022e23       # Avogadro's number
R0 = kB * eVtokJ * 1000.        # gas constant in J/mol K

A = 9.2e6 # liter/mole/second
Ea = 99. # kJ/mole

T = 500.   # K


# 1 M standard state
deltaUdd = Ea - R0 * T /1000    # kJ /mol

SS = 1.0  # mol/liter
deltaSdd = R0 * ( np.log(A/(1./SS)) - np.log(kB * T / h) - 1.)

deltaAdd = deltaUdd - T * deltaSdd/1000.

print('1 M standard state, 500 K:')
print('Delta Udd ={:4.0f} kJ/mol    Delta Sdd = {:4.0f} J/mol K   Delta Add = {:4.0f} kJ/mol'.format(deltaUdd,deltaSdd,deltaAdd))

# 1 bar standard state
P0 = 1.0e5    # 1 bar = 10^5 Pa = 10^5 J/m^3
P0 = P0 / 1e3  # J/l
```



```
SS = P0/(R0 * T)   #  J/mol / J/l  = mol/liter
print('1 bar = {:5.3f} mol/liter standard state, 500 K'.format(SS))

deltaHdd = Ea - 2* R0 * T /1000   # kJ /mol
deltaSdd = R0 * ( np.log(A/(1./SS)) - np.log(kB * T / h) - 2.)

deltaGdd = deltaHdd - T * deltaSdd/1000.
print('Delta Hdd ={:4.0f} kJ/mol    Delta Sdd = {:4.0f} J/mol K   Delta Gdd = {:4.0f} kJ/mol'.format(deltaHdd,deltaSdd,deltaGdd)
```

```
1 M standard state, 500 K:
Delta Udd =  95 kJ/mol    Delta Sdd = -124 J/mol K   Delta Add =  157 kJ/mol
1 bar = 0.024 mol/liter standard state, 500 K
Delta Hdd =  91 kJ/mol    Delta Sdd = -164 J/mol K   Delta Gdd =  172 kJ/mol
```

6.0.6 Correlations across reactions

1. early vs late transition states
2. Brønsted-Evans-Polyani relationship

$$E_a = \alpha \Delta H + \beta$$

3. linear free energy relationships between similar reactions (substituent effects)

$$\ln(k_1/k_1') \propto \ln(K_1/K_1')$$

4. compensation effect linear correlation across catalysts for the same reaction

$$\Delta H^{\circ\ddagger} \propto \Delta S^{\circ\ddagger}$$

7 Lecture 6: Reaction networks

7.1 Simple and visible

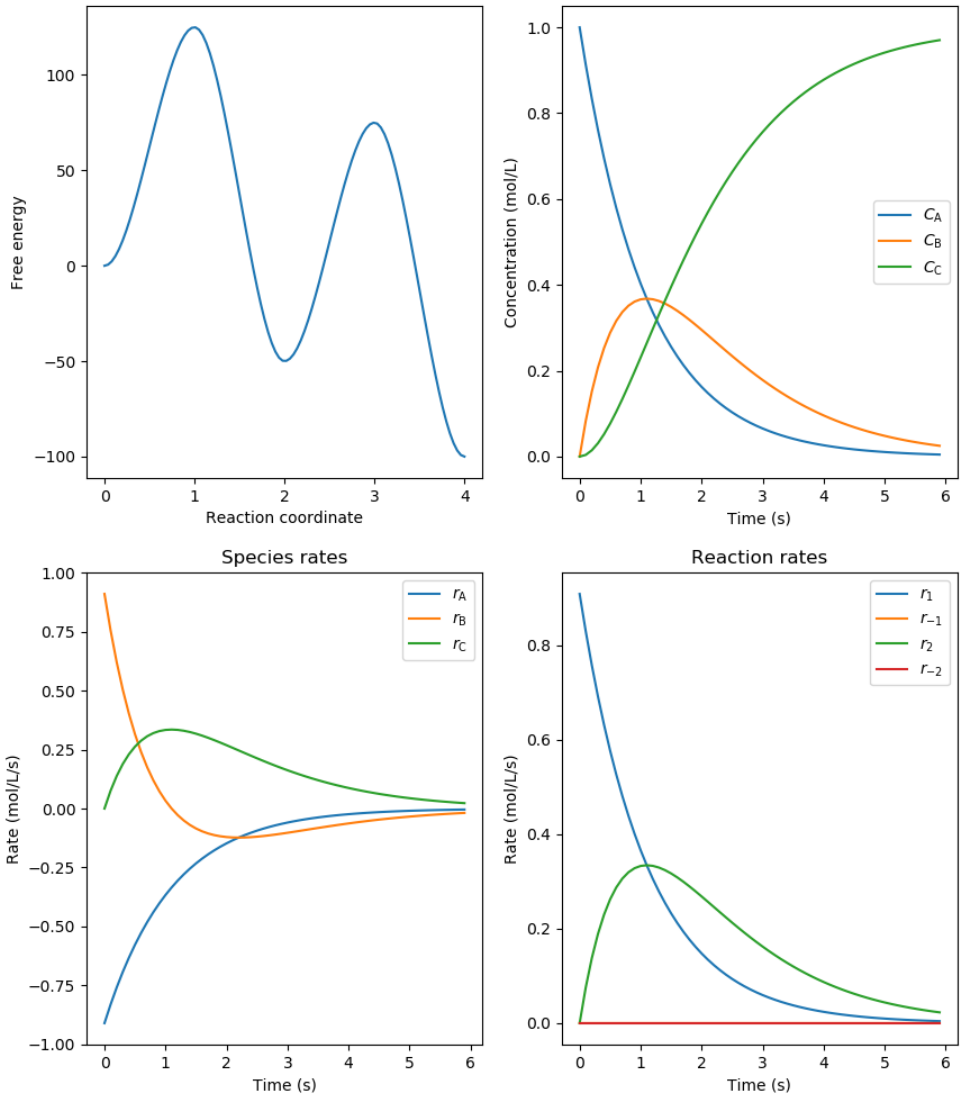
1. Open vs closed (catalyzed) networks
2. A $\longleftrightarrow$ B $\longleftrightarrow$ C reaction
3. free energy surface
4. Characterizations
   (a) instantaneous selectivity
   (b) overall selectivity
   (c) yield
5. species rates vs reaction rates
   (a) Vector representations of rates
6. Rate determining and degree of rate control
7. Equilibrium limited


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

kB = 8.61733e-5      # eV /K
h = 4.13566766e-15  # eV s

def dCdt(C,t,k):
    dC_Adt = -k[0]*C[0]+k[1]*C[1]     # A <->  B <-> C;  r = k CA CB
    dC_Bdt =  k[0]*C[0]-k[1]*C[1] -k[2]*C[1]+k[3]*C[2]
    dC_Cdt =  k[2]*C[1]-k[3]*C[2]
    dCdt = [dC_Adt,dC_Bdt,dC_Cdt]
    return dCdt

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

R0 = 8.314
T = 500.

# initialize k's
deltaGtot = -100.0; deltaG1 = -50.0; deltaG1d = 125. ; deltaG2d = 125.

deltaG2 = deltaGtot - deltaG1
deltaGn1d = deltaG1d-deltaG1
deltaGn2d = deltaG2d-deltaG2

p1 = BPoly.from_derivatives([0, 1, 2, 3, 4], [[0, 0], [deltaG1d, 0], [deltaG1, 0], [deltaG1+deltaG2d,0], [deltaGtot,0]])

x=np.linspace(0,4,100)
y=p1(x)

plt.figure(figsize=(10,12))
plt.subplot(2,2,1)
plt.plot(x,y)
plt.ylabel('Free energy')
plt.xlabel('Reaction coordinate')

k = (kB*T/h)*np.exp(-np.array([deltaG1d,deltaGn1d,deltaG2d,deltaGn2d])*1000./(R0*T))
print('T = {:4.1f}'.format(T))
print('Delta G1 = {:4.1f}  deltaG1dagger = {:4.1f}  delta G2 = {:4.1f}   deltaG2d = {:4.1f} kJ/mol'.format(deltaG1,deltaG1d,delt
print('k1 = {:6.3e}     k-1 = {:6.3e}         k2 = {:6.3e}      k-2 = {:6.3e} /s'.format(k[0],k[1],k[2],k[3]))

# Range of time to solve over
t = np.arange(0,6,0.1)

p = (k,) # turn parameters into a tuple
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.subplot(2,2,2)
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.subplot(2,2,3)
plt.plot(t,-k[0]*C_A+k[1]*C_B,label=r'$r_{\rm A}$')
plt.plot(t,k[0]*C_A-k[1]*C_B-k[2]*C_B + k[3]*C_C,label=r'$r_{\rm B}$')
plt.plot(t,k[2]*C_B-k[3]*C_C,label=r'$r_{\rm C}$')
plt.xlabel('Time (s)')
plt.ylabel('Rate (mol/L/s)')
```



```
68 plt.title('Species rates')
69 plt.legend()
70
71 plt.subplot(2,2,4)
72 plt.plot(t,k[0]*C_A,label=r'$r_1$')
73 plt.plot(t,k[1]*C_B,label=r'$r_{-1}$')
74 plt.plot(t,k[2]*C_B,label=r'$r_2$')
75 plt.plot(t,k[3]*C_C,label=r'$r_{-2}$')
76 plt.xlabel('Time (s)')
77 plt.ylabel('Rate (mol/L/s)')
78 plt.title('Reaction rates')
79 plt.legend()
80
81 plt.savefig('./Images/ABC-rxn.png')
```

```
T = 500.0
Delta G1 = -50.0  deltaG1dagger = 125.0  delta G2 = -50.0   deltaG2d = 125.0 kJ/mol
k1 = 9.092e-01     k-1 = 5.433e-06         k2 = 9.092e-01      k-2 = 5.433e-06 /s
```

7.2 Simply and invisible

1. Lindemann-Hinshelwood model for first order reactions
2. Irreversible steps
3. Pseudo-steady state approximation
   (a) A $\longrightarrow$ B $\longrightarrow$ C in limit $k_2 > k_1$

$$r_B \approx 0$$

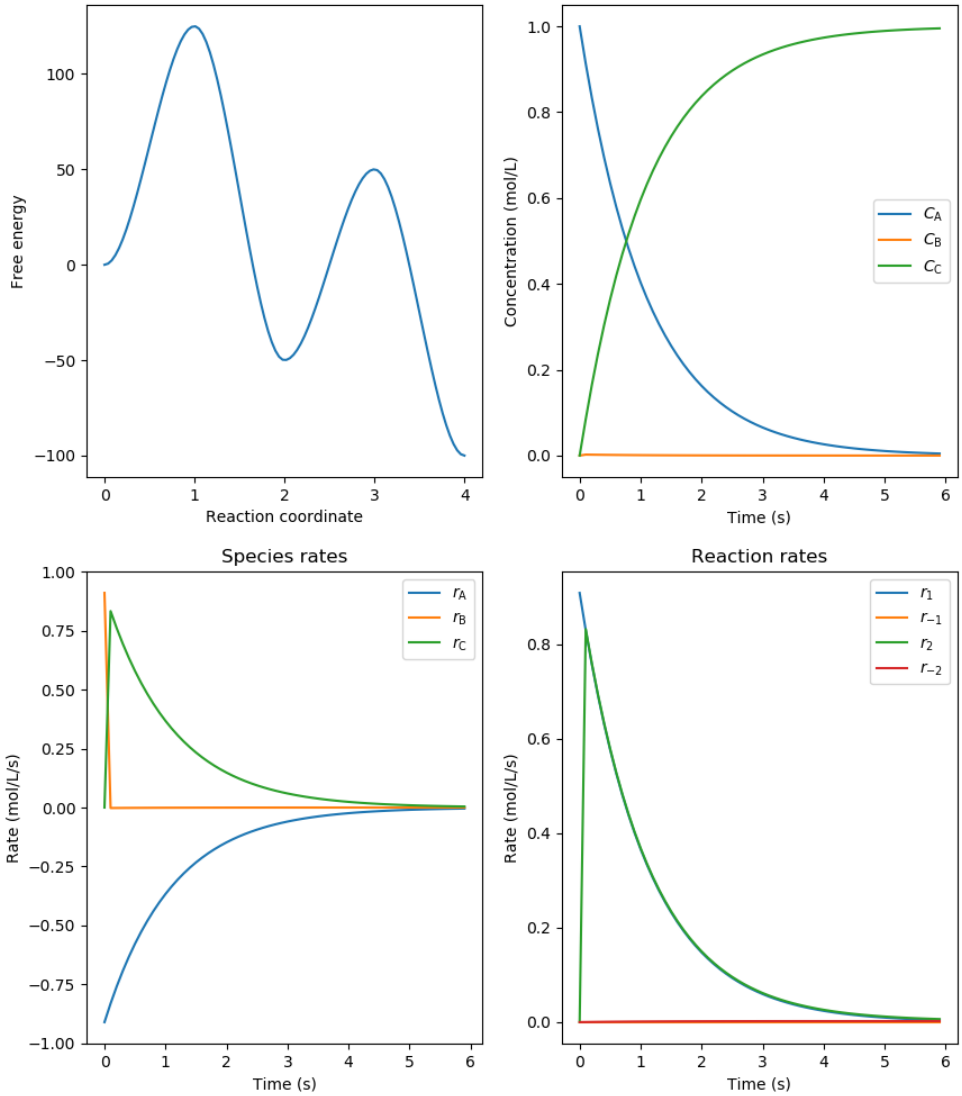
$$C_{\mathrm{B,max}} \approx 0$$

$$C_A = C_{A0} e^{-k_1 t}$$

$$C_B = C_{A0} \frac{k_1}{k_2 - k_1} \left(e^{-k_1 t} - e^{-k_2 t} \right)$$

(a) B intermediate → reactive intermediate

```
T = 500.0
Delta G1 = -50.0  deltaG1dagger = 125.0  delta G2 = -50.0   deltaG2d = 100.0 kJ/mol
k1 = 9.092e-01     k-1 = 5.433e-06         k2 = 3.720e+02     k-2 = 2.222e-03 /s
```



7.3 Simplifying reaction networks

1. Simplifications helpful when parameters of full model are unknown or to fit to experimental observations
2. $2O_3 \longrightarrow 3O_2$ PSSA
   (a) limiting behaviors
3. CO chlorination (closed)

$$Cl_2 \overset{h\nu}{\rightleftharpoons} 2Cl\cdot$$

$$CO + Cl\cdot \rightleftharpoons COCl\cdot$$

$$COCl\cdot + Cl_2 \rightleftharpoons COCl_2 + Cl\cdot$$

4. Ethane pyrolysis $C_2H_6 \longrightarrow C_2H_4 + H_2$ (open)

	A (s^{-1})	E_a ($kJ\,mol^{-1}$)
$C_2H_6 \longrightarrow 2CH_3$	1.0×10^{17}	356
$CH_3 + C_2H_6 \longrightarrow CH_4 + C_2H_5$	2.0×10^{11}	44
$C_2H_5 \longrightarrow C_2H_4 + H$	3.0×10^{14}	165
$H + C_2H_6 \longrightarrow H_2 + C_2H_5$	3.4×10^{12}	28
$H + C_2H_5 \longrightarrow C_2H_6$	1.6×10^{13}	0

1. Chain reactions
   (a) Polymerization

7.4 Complex reaction networks

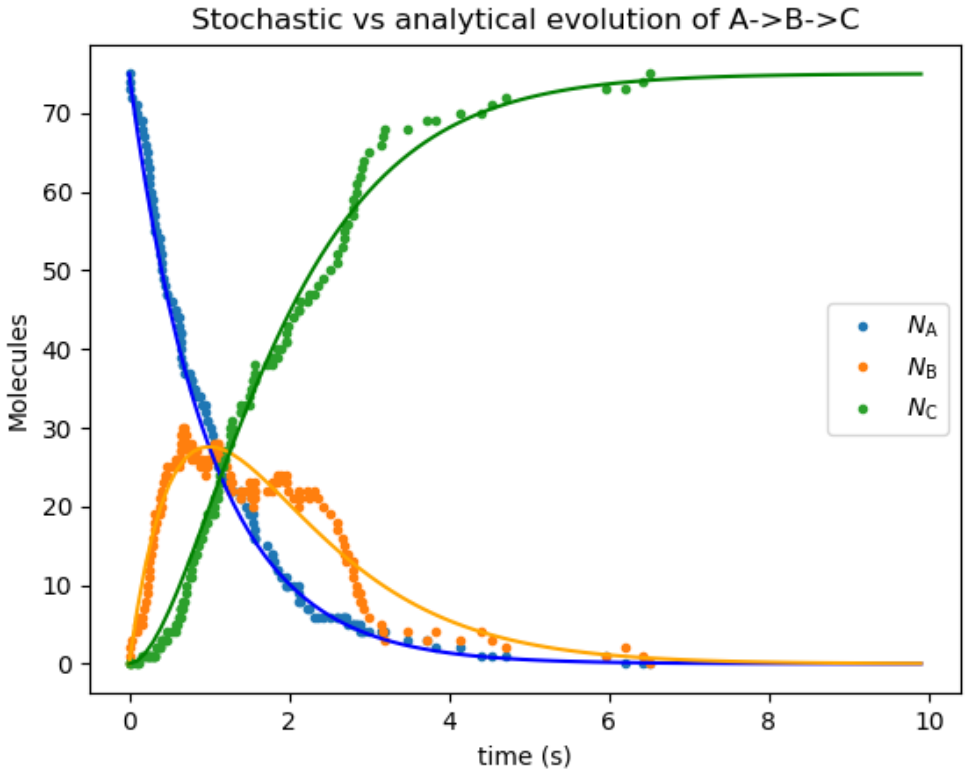
1. Ethane pyrolysis redux, doi:10.1021/jp206503d

7.5 Stochastic solutions

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint, solve_ivp

Nav = 6.022e23
l_to_nm3 = 1e21

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

Ntot = 75  # set total number of particles

CA0 = 1.0  # set initial concentrations, mol/l
CB0 = 0.0
CC0 = 0.0

k = np.array([1.0,1.0])   # /second

Ctot = CA0 + CB0 + CC0
```



```
24
25  Volume = Ntot/Ctot # number/mol/liter
26  side = Volume**(1/3)
27
28  na = Ntot *(CA0/Ctot); nb = Ntot *(CB0/Ctot); nc = Ntot *(CC0/Ctot); tnow = 0.
29
30  NA = np.array([na])
31  NB = np.array([nb])
32  NC = np.array([nc])
33  t = np.array([tnow])
34
35  while na > 0 or nb > 0:  # While there is still A and B left, there will be reaction occurring.
36      p1 = np.random.random_sample()  # Get random number from 0 to 1.
37      p2 = np.random.random_sample()
38      r1 = k[0] * na  # Definition for the reaction mechanism.
39      r2 = k[1] * nb
40      if p2 < (r1/(r1+r2)):  # Monte Carlo step
41          na-=1;  nb+=1
42      else:
43          nb-=1;  nc+=1
44
45      dt = -np.log(p1) / (r1 + r2)  # This is the time interval until the next reaction to occur.
46      tnow+= dt  # To refresh the time for next circulation.
47
48      NA = np.append(NA,na)  # inelegant and wasteful
49      NB = np.append(NB,nb)
50      NC = np.append(NC,nc)
51      t  = np.append(t,tnow)
52
53  #    open('KMC_data.txt','w+').write('\n'.join('%f, %i, %i, %i' %x for x in data))
54      # Record data of this circulation to the .txt file.
55
56  # analytical solution
57  C_0 = [CA0, CB0, CC0]
58  ta = np.arange(0,10,0.1)
59  p = (k,) # turn parameters into a tuple
60  C = odeint(dCdt,C_0,ta,p)
61
62  C_A = C.transpose()[0] # Get C_A from C
63  C_B = C.transpose()[1] # Get C_B from C
64  C_C = C.transpose()[2]
65
66  # Plot with the data acquired above.
67  plt.figure()
68  plt.plot(t, NA, 'o', markersize = '3', label=r'$N_{\rm A}$')  # Configure for the first line.
69  plt.plot(t, NB, 'o', markersize = '3', label=r'$N_{\rm B}$')
70  plt.plot(t, NC, 'o', markersize = '3', label=r'$N_{\rm C}$')
71  plt.plot(ta,C_A*Volume,'-',color='blue')
72  plt.plot(ta,C_B*Volume,'-',color='orange')
73  plt.plot(ta,C_C*Volume,'-',color='green')
74  plt.legend()
75  plt.title('Stochastic vs analytical evolution of A->B->C')
76  plt.xlabel('time (s)')  # Labeling the axis.
77  plt.ylabel('Molecules')
78  plt.savefig('./Images/kMC.png')
```

8 Heterogeneous reactions

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

8.0.1 Heterogeneous reactions and catalysis

1. molecule-surface collisions
2. surface reactions
3. Ammonia oxidation, $NH_3 + O_2 \longrightarrow NO + N_2$, doi:10.1021/acscatal.8b04251

9 Liquid-phase reactions

9.0.1 Diffusion-controlled reactions

1. Intermediate complex
2. Steady-state approximation

Table 2: Equilibrium and Rate Constants

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

$$K_{eq}(T) = e^{\Delta S^\circ(T)/k_B} e^{-\Delta H^\circ(T)/k_B T}$$

$$K_c(T) = \left(\frac{1}{c^\circ}\right)^{\nu_c+\nu_d-\nu_a-\nu_b} \frac{(q_c/V)^{\nu_c}(q_d/V)^{\nu_d}}{(q_a/V)^{\nu_a}(q_b/V)^{\nu_b}} e^{-\Delta E(0)\beta}$$

$$K_p(T) = \left(\frac{k_B T}{P^\circ}\right)^{\nu_c+\nu_d-\nu_a-\nu_b} \frac{(q_c/V)^{\nu_c}(q_d/V)^{\nu_d}}{(q_a/V)^{\nu_a}(q_b/V)^{\nu_b}} e^{-\Delta E(0)\beta}$$

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

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

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

3. Diffusion-controlled limit ($k_D = 4\pi(r_A + r_B)D_{AB}$)
4. Reaction-controlled limit ($k_{app} = (k_D/k_{-D})k_r$)

