Lab Report # 2: Simulating Simple Mechanisms

Max Huggins – UCA Department of Chemistry

February 9, 2021

Abstract

The simulations of several types of reactions were performed using 100 dice. Three areas A, B, and C represent species. The number of dice inside of an area represented the concentration of the species. The trial of dice rolled is representative of the time. Parallel:

$$A \longrightarrow B \tag{1}$$

$$A \longrightarrow C \tag{2}$$

Reversible:

$$A \longleftrightarrow B \tag{3}$$

Consecutive:

$$A \longrightarrow B \tag{4}$$

$$B \longrightarrow C \tag{5}$$

reactions were analyzed. These chemical reactions proceed with purely probabilistic behavior-or rather, the laws of thermodynamics which explain the behavior of very large numbers.

1 Data

Here, several tables and plots will appear that hold the data of rolled dice. Unfortunately, it will appear somewhat raw.

1.1 Parallel Reactions

First, is the data of dice rolled for parallel reactions. This was produced from the following instructions:
1) All dice begin in A
2) For t=1, roll all dice in A. All that land on 1 go to B and those landing on 2 or 3 move to C. The rest stay in A
3) Repeat until all dice are in B or C
The recorded data is listed in table 1.1.
This data was plotted along with the theoretical results for concentration as a function of time. More discussion on this is placed in the calculations section of this paper, but the plots generated are shown in figure 1.

Parallel			
Time	A	B	C
0	100	0	0
1	56	12	32
2	32	20	48
3	19	23	58
4	12	26	62
5	6	29	65
6	1	31	68
7	1	31	68
8	0	31	69

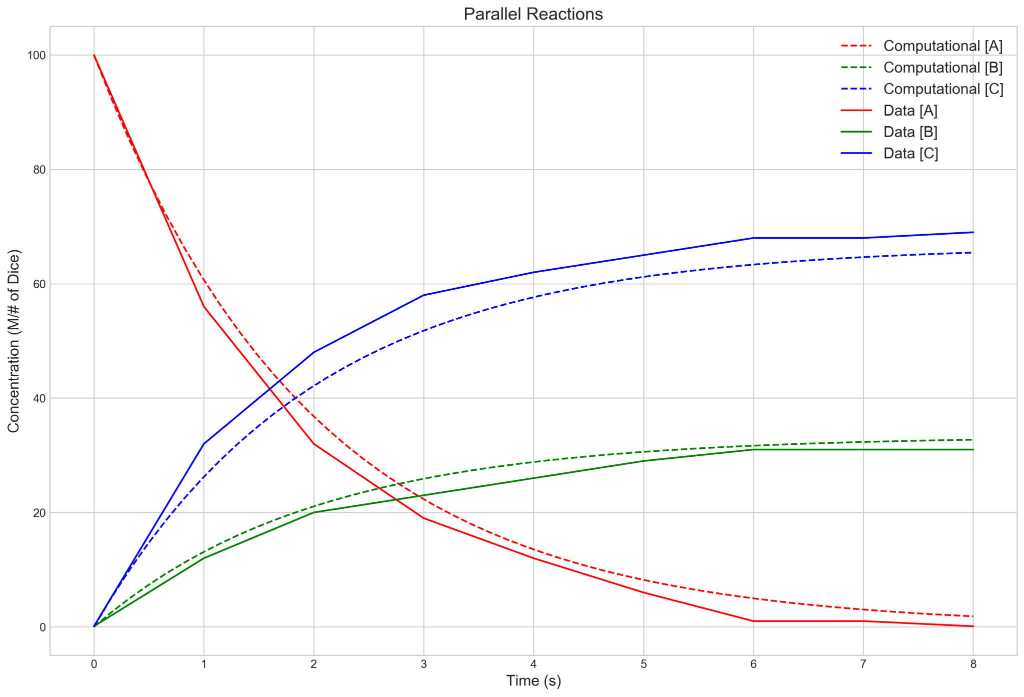

Figure 1: Concentration of species from parallel reactions as a function of time. Experimental and computed analytical results are plotted.

1.2 Reversible

The reversible reaction is modeled by the dice through these steps:
1) All dice begin in A
2) For t=1, roll all dice in A. Values 1 and 2 go to area B.
3) Roll all dice in A. Move values 1 and 2 to B. Roll all dice in B (excluding those just moved.) If a 6 is rolled, move it back to A.
4) Repeat step 3 until values are stable in A and B.
The corresponding values are recorded in table 1.2.

Reversible		
Time	A	B
0	100	0
1	67	33
2	58	42
3	48	52
4	39	61
5	40	60
6	37	63
7	38	62
8	39	61
9	38	62

Similarly to parallel reactions I have plotted the data in figure 2.

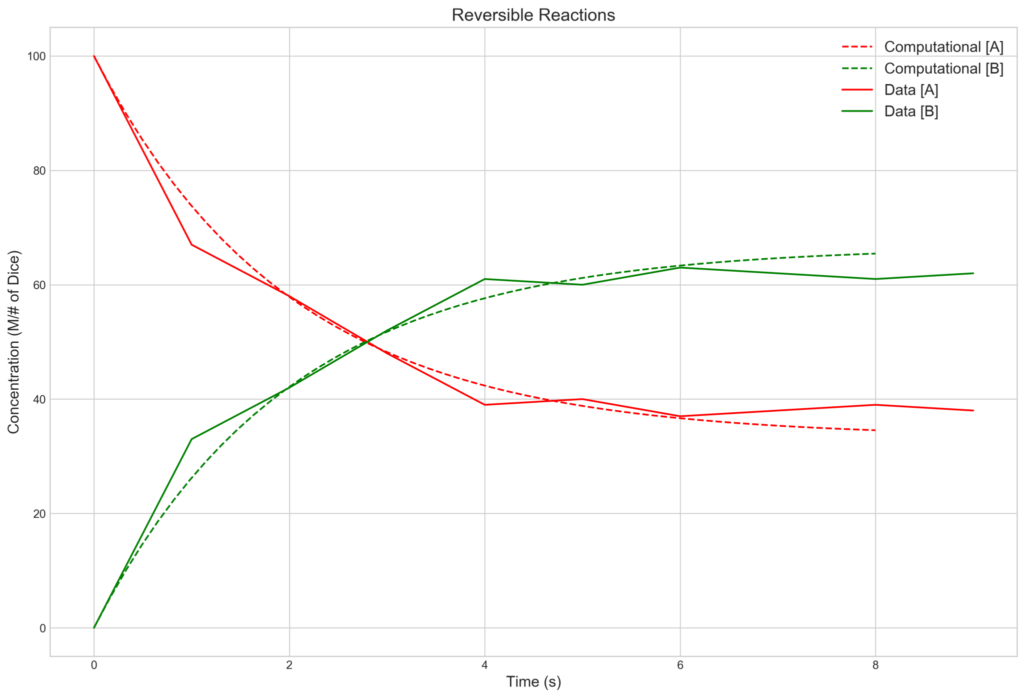

Figure 2: Concentration of species from reversible reactions as a function of time. Experimental and computed analytical results are plotted.

1.3 Consecutive Reactions

Consecutive reactions were simulated in the following manner using dice:
1) All dice begin in A
2) Roll all dice in A. Values between 1 and 5 move to area B. Sixes stay in A.
3) Roll all dice in B. If a 1 is rolled, it moves to C. Values between 2 and 6 stay in B. Now move all in A and move values 1-5 to B
4) Repeat until all dice are in C
A similar procedure is used, but with a "faster" second reaction. The data for consecutive reactions part 1 and 2 are recorded in table 1.3.

Consecutive Pt. 1			
Time	A	B	C
0	100	0	0
1	17	83	0
2	2	84	14
3	0	66	34
4	0	57	43
5	0	46	54
6	0	37	63
7	0	30	70
8	0	22	78
9	0	19	81
10	0	16	84
11	0	16	84
12	0	12	88
13	0	12	88
14	0	11	89
15	0	8	92
16	0	6	94
17	0	5	95
18	0	5	95
19	0	4	96
20	0	2	98
21	0	1	99
22	0	1	99
23	0	1	99
24	0	0	100

Consecutive Pt. 2			
Time	A	B	C
0	100	0	0
1	82	18	0
2	70	14	16
3	60	11	29
4	49	13	38
5	43	7	50
6	33	11	56
7	27	7	66
8	18	12	70
9	16	3	81
10	13	4	83
11	11	2	87
12	10	1	89
13	6	4	90
14	6	1	93
15	5	1	94
16	4	1	95
17	4	0	96
18	4	0	96
19	4	0	96
20	3	1	96
21	3	0	97
22	2	1	97
23	2	0	98
24	2	0	98
25	2	0	98
26	2	0	98
27	2	0	98
28	2	0	98
29	2	0	98
30	2	0	98
31	2	0	98
32	2	0	98
33	2	0	98
34	2	0	98
35	2	0	98
36	2	0	98
37	2	0	98
38	2	0	98
39	2	0	98
40	2	0	98
41	2	0	98
42	2	0	98
43	2	0	98
44	2	0	98
45	2	0	98
46	2	0	98
47	2	0	98
48	2	0	98
49	2	0	98
50	2	0	98
51	2	0	98
52	1	1	98
53	1	0	99
54	1	0	99
55	0	1	99
56	0	0	100
57	0	0	100

Figure 3 contains the plots for the consecutive reactions part 1 and figure 4 shows them for part 2.

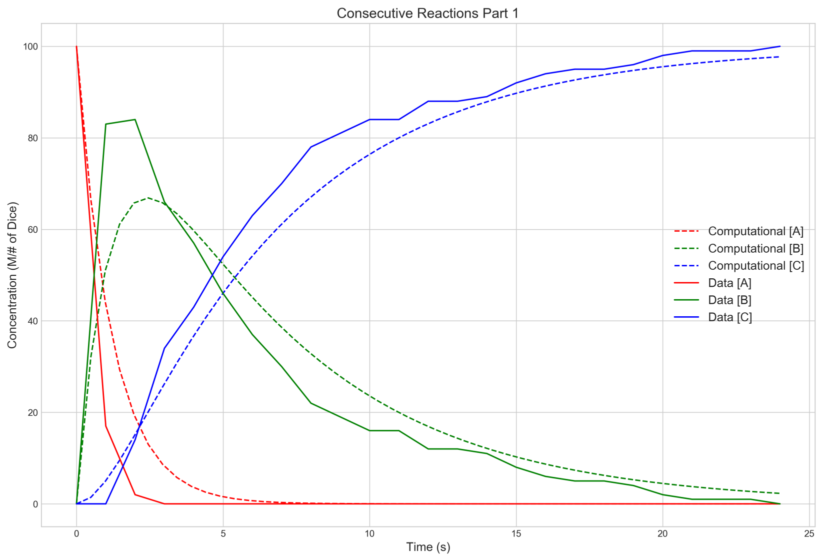

Figure 3: Concentration of species from consecutive reactions as a function of time. Experimental and computed analytical results are plotted.

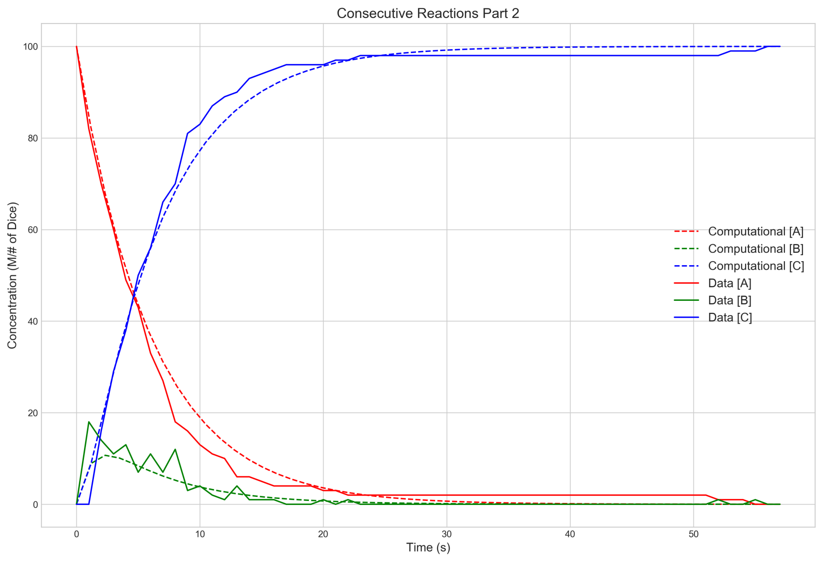

Figure 4: Concentration of species from consecutive reactions as a function of time. Experimental and computed analytical results are plotted.

2 Calculations

Quite simply, the tabulated data was determined through the probabilistic behavior of rolling dice. The data was then loaded into python and plotted along with computed analytical results of each type of reaction. The analytical results are obtained through the integration of the various rate laws that apply to each type of reaction. More information on how these analytical results were found in Atkins and Paula's text $PhysicalChemistry : Thermodynamics, Structure, andChange$ chapter 20E.
For parallel reactions the concentration of each species can be expressed as functions of time and are shown below:

$$[A] = [A]_0 e^{-k_T t} \tag{6}$$

$$[B] = \frac{k_1}{k_T}[A]_0(1 - e^{-k_T t}) \tag{7}$$

$$[C] = \frac{k_2}{k_T}[A]_0(1 - e^{-k_T t}) \tag{8}$$

Where all k_n values along with $[A]_0$ are constants. The variable t is time and [N] represents the concentration of some species, N. These functions were evaluated for the same time period as the amount of "seconds" the dice simulations occurred and were plotting along with the data points recorded. The results are available in the figures in the data section.
I will omit the equations used for the other reaction types as I feel they are redundant to the lab handout provided. However, I have provided the source code at the end of this report for the graphs I made. Note, all of the data that I pulled from excel for my python file is displayed in various tables around this report. In this script I have also included best fit data to attempt to determine the simulated k_n values. Fit lines were produced using the SciPy library and the best fit curves are included with the experimental data points for parallel, reversible, and consecutive respectively.

Values for $\frac{[B]}{[C]}$ and $\frac{k_1}{k_2}$ are shown in figure 9.

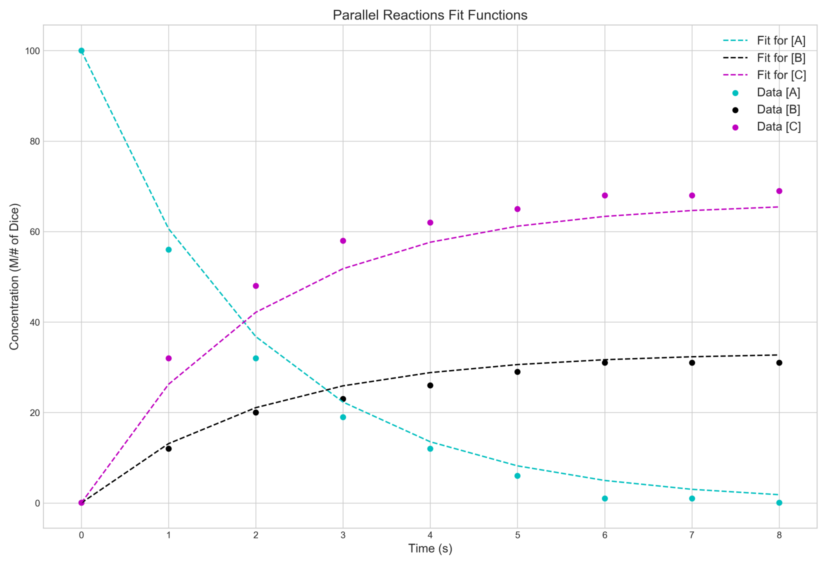

Figure 5: Best fit lines for parallel reaction simulation with dice.

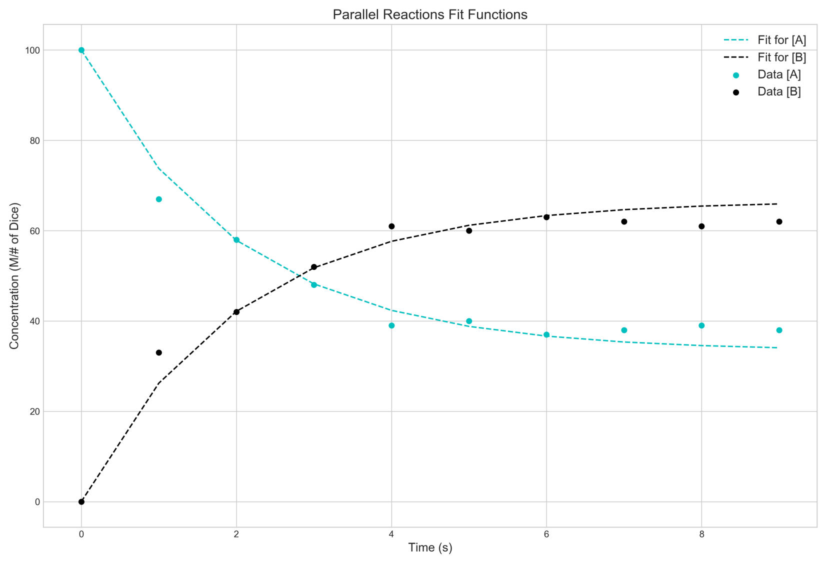

Figure 6: Best fit lines for reversible reaction simulation with dice.

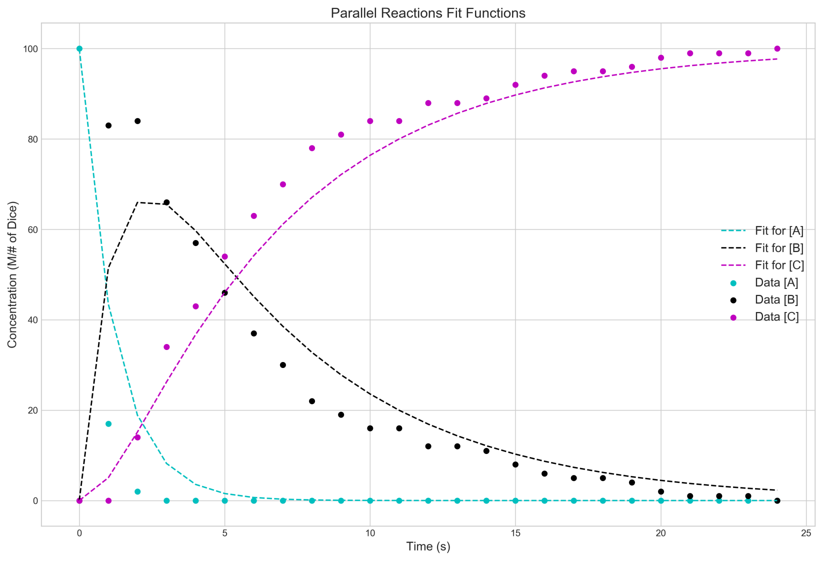

Figure 7: Best fit lines for consecutive reaction simulation with dice.

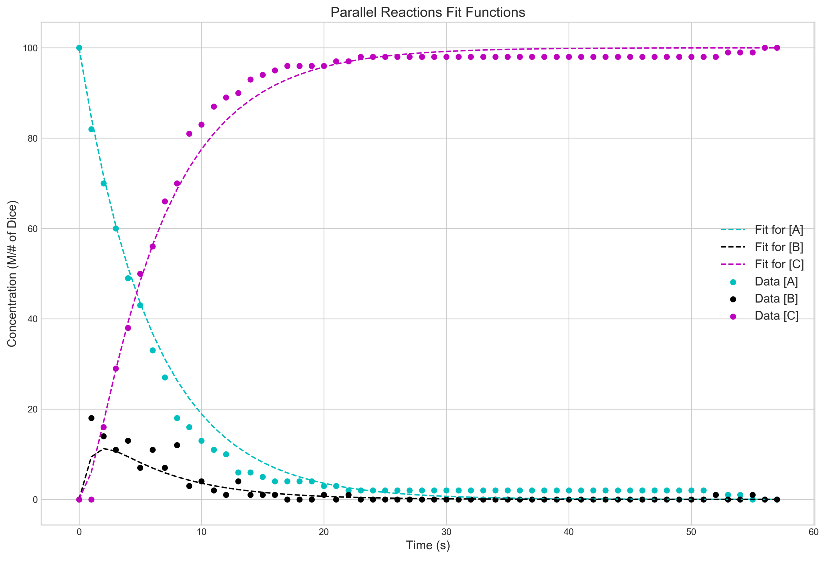

Figure 8: Best fit lines for consecutive reaction simulation with dice.

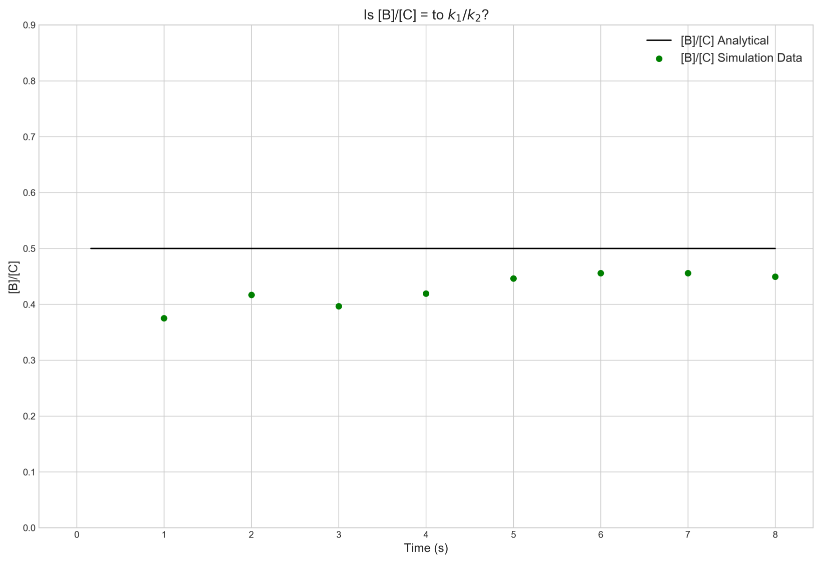

Figure 9: Concentration ratios and k constant ratios.

Branching ratios?

$$K_{eq} = \left.\frac{[B]}{[A]}\right|_{eq} \tag{9}$$

Taking average of last three values for reversible reactions:

$$\frac{[B]}{[A]} = \frac{62+61+62}{38+39+38} = 1.609 \tag{10}$$

Where k_1 and k_2 was found to be 2 using fit data and supposed values. This is off by approximately 20%.
Here are the analytical results for the concentration functions of A, B, and C.

$$[A] = [A]_0 e^{-k_1 t} \tag{11}$$

$$[B] = \frac{k_1}{k_2 - k_1}[A]_0(e^{-k_1 t} - e^{-k_2 t}) \tag{12}$$

$$[C] = [A]_0[1 - \frac{k_1}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t}) - e^{-k_1 t}] \tag{13}$$

When $k_1 > k_2$ we cannot make any assumptions, but when $k_1 >> k_2$ it is fair to say the coefficient: $\frac{k_1}{k_2 - k_1} \approx -1$ because a relatively small number subtracted from a relatively large number simply leaves the relatively large number. This brings about:

$$[B] \approx -[A]_0(e^{-k_1 t} - e^{-k_2 t}) = [A]_0(e^{-k_2 t} - e^{-k_1 t}) \tag{14}$$

As well as,

$$[C] \approx [A]_0[1 - (-1)(e^{-k_1 t} - e^{-k_2 t}) - e^{-k_1 t}] = [A]_0[1 - e^{-k_2 t})] \tag{15}$$

Now we will show:

$$[C] \approx [A]_0 - [A] - [B] \tag{16}$$

Substituting equations 11 and 14 into 16:

$$[C] \approx [A]_0 - [A]_0(e^{-k2t} - e^{-k_1 t}) - [A]_0 e^{-k_1 t} \tag{17}$$

Factoring $[A]_0$,

$$[C] \approx [A]_0(1 - (e^{-k_2 t} - e^{-k_1 t}) - e^{-k_1 t}) = [A]_0(1 - e^{-k_2 t}) \tag{18}$$

It can be seen equation 15 and 18 are equivalent.

References

[1] Atkins, P. W., & Paula, J. D. (2014). Physical chemistry: thermodynamics, structure, and change. New York: W.H. Freeman.

Python script for plotting.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 17 09:41:40 2019

@author: maxhu
"""
#==============================Modules==========================================#
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
plt.style.use('seaborn-whitegrid')
#============================Constants==========================================#

#===Parallel==========#
A_0 = 100
k_1p = 1/6
k_2p = 2/6
k_T = .5
#====Reversible======#
k_1r = 2/6
k_2r = 1/6
#====Consecutive=pt.1==#
k_1c1 = 5/6
k_2c1 = 1/6
#====Consecutive=pt.2==#
k_1c2 = 1/6
k_2c2 = 5/6

#==============================Functions========================================#

#===Parallel==========#

def AP(t):
    return A_0 * np.exp(-k_T * t)
def BP(t):
    return (k_1p / k_T) * A_0 * (1 - np.exp(-k_T * t))
def CP(t):
    return (k_2p / k_T) * A_0 * (1 - np.exp(-k_T * t))

#====Reversible======#

def AR(t):
    num = A_0 * (k_2r + k_1r * np.exp((-k_1r - k_2r) * t))
    den = k_1r + k_2r
    return num / den
def BR(t):
    num = k_1r * A_0 * (1 - np.exp((-k_1r - k_2r) * t))
    den = k_1r + k_2r
    return num / den

#====Consecutive=pt.1==#
```

```

def AC1(t):
    return A_0 * np.exp(-k_1c1 * t)
def BC1(t):
    num = k_1c1 * A_0 * (np.exp(-k_1c1 * t) - np.exp(-k_2c1 * t))
    den = k_2c1 - k_1c1
    return num / den
def CC1(t):
    return A_0 * (1 - (k_1c1 / (k_2c1 - k_1c1))
                  * (np.exp(-k_1c1 * t) - np.exp(-k_2c1 * t))
                  - np.exp(-k_1c1 * t))

#===Consecutive=pt.2==#

def AC2(t):
    return A_0 * np.exp(-k_1c2 * t)
def BC2(t):
    return (k_1c2 / k_2c2) * A_0 * (np.exp(-k_1c2 * t) - np.exp(-k_2c2 * t))
def CC2(t):
    return A_0 - A_0 * np.exp(-k_1c2 * t) - (k_1c2 / k_2c2) * A_0 * (np.exp(
            -k_1c2 * t) - np.exp(-k_2c2 * t))

#===Fitting===========#
def AP_FIT(x, c):
    return 100 * np.exp(-c*x)

def BP_FIT(x, a, c):
    return a *100 * (1 - np.exp(-c*x))

def CP_FIT(x, a, c):
    return a *100 * (1 - np.exp(-c*x))
#=====================#
def AR_FIT(x, a, b):
    return (100 / (a + b)) * (b + a * np.exp((-a - b) * x))

def BR_FIT(x, a, b, c):
    return (a * 100 * (1 - np.exp((-a - b) * x))) / (a + b)
#=====================#
def AC_FIT(x, a):
    return 100 * np.exp(-a*x)

def BC_FIT(x, b):
    return (0.83333333 / (b - 0.83333333)) * 100 * (
            np.exp(-0.83333333 * x) - np.exp(-b * x))

def CC_FIT(x, b):
    return 100 * (1 - (0.83333333 / (b - 0.83333333)) * (
            np.exp(-0.83333333 * x) - np.exp(-b * x)) -
        np.exp(-0.83333333 * x))
#=====================#
def AC_FIT2(x, a):
    return 100 * np.exp(-a*x)

def BC_FIT2(x, b):
    return (.16666667 / (b - .16666667)) * 100 * (
            np.exp(-.16666667 * x) - np.exp(-b * x))

def CC_FIT2(x, b):
    return 100 * (1 - (.16666667 / (b - .16666667)) * (
            np.exp(-.16666667 * x) - np.exp(-b * x)) -
        np.exp(-.16666667 * x))

#====================================Data====================================#

Par_data = pd.read_excel (r'C:\Users\maxhu\Documents\PChem\Lab 2\Lab2.xlsx',
                      sheet_name='Parallel')
Rev_data = pd.read_excel (r'C:\Users\maxhu\Documents\PChem\Lab 2\Lab2.xlsx',
                      sheet_name='Reversible')
Cons1_data = pd.read_excel (r'C:\Users\maxhu\Documents\PChem\Lab 2\Lab2.xlsx',
                      sheet_name='Consecutive pt. 1')
Cons2_data = pd.read_excel (r'C:\Users\maxhu\Documents\PChem\Lab 2\Lab2.xlsx',
                      sheet_name='Consecutive pt. 2')

#=====Making some lists with my data=====#

A_p = pd.DataFrame(Par_data, columns= ['A'])
B_p = pd.DataFrame(Par_data, columns= ['B'])
C_p = pd.DataFrame(Par_data, columns= ['C'])

A_p = A_p.values.tolist()
B_p = B_p.values.tolist()
C_p = C_p.values.tolist()

A_p  = [val for sublist in A_p for val in sublist]
B_p  = [val for sublist in B_p for val in sublist]
C_p  = [val for sublist in C_p for val in sublist]

#==========#

A_r = pd.DataFrame(Rev_data, columns= ['A'])
B_r = pd.DataFrame(Rev_data, columns= ['B'])

A_r = A_r.values.tolist()
B_r = B_r.values.tolist()

A_r  = [val for sublist in A_r for val in sublist]
B_r  = [val for sublist in B_r for val in sublist]

#==========#

A_c1 = pd.DataFrame(Cons1_data, columns= ['A'])
B_c1 = pd.DataFrame(Cons1_data, columns= ['B'])
C_c1 = pd.DataFrame(Cons1_data, columns= ['C'])

A_c1 = A_c1.values.tolist()
B_c1 = B_c1.values.tolist()
C_c1 = C_c1.values.tolist()

```

```python
A_c1  = [val for sublist in A_c1 for val in sublist]
B_c1  = [val for sublist in B_c1 for val in sublist]
C_c1  = [val for sublist in C_c1 for val in sublist]

#==========#

A_c2 = pd.DataFrame(Cons2_data, columns= ['A'])
B_c2 = pd.DataFrame(Cons2_data, columns= ['B'])
C_c2 = pd.DataFrame(Cons2_data, columns= ['C'])

A_c2 = A_c2.values.tolist()
B_c2 = B_c2.values.tolist()
C_c2 = C_c2.values.tolist()

A_c2  = [val for sublist in A_c2 for val in sublist]
B_c2  = [val for sublist in B_c2 for val in sublist]
C_c2  = [val for sublist in C_c2 for val in sublist]


#==============================Plotting==========================================#

#======Parallel=========#
t_values = np.linspace(0, 8)
t_data = np.linspace(0, 8, 9)

fig = plt.figure(1,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Parallel Reactions', fontsize = '15')
#Computational Values
plt.plot(t_values, AP(t_values), color = 'r', label = 'Computational [A]',
         linestyle = '--')
plt.plot(t_values, BP(t_values), color = 'g', label = 'Computational [B]',
         linestyle = '--')
plt.plot(t_values, CP(t_values), color = 'b', label = 'Computational [C]',
         linestyle = '--')
#Actual Data
plt.plot(t_data, A_p, color = 'r', label = 'Data [A]')
plt.plot(t_data, B_p, color = 'g', label = 'Data [B]')
plt.plot(t_data, C_p, color = 'b', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2PARALLEL', dpi=300)

#=====Reversible========#

t_values = np.linspace(0, 8)
t_data = np.linspace(0, 9, 10)

fig = plt.figure(2,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Reversible Reactions', fontsize = '15')
```

```
#Computational Values
plt.plot(t_values, AR(t_values), color = 'r', label = 'Computational [A]',
         linestyle = '--')
plt.plot(t_values, BR(t_values), color = 'g', label = 'Computational [B]',
         linestyle = '--')
#Actual Data
plt.plot(t_data, A_r, color = 'r', label = 'Data [A]')
plt.plot(t_data, B_r, color = 'g', label = 'Data [B]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2REVERSIBLE', dpi=300)

#=====Consecutive 1=====#

t_values = np.linspace(0, 24)
t_data = np.linspace(0, 24, 25)

fig = plt.figure(3,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Consecutive Reactions Part 1', fontsize = '15')
#Computational Values
plt.plot(t_values, AC1(t_values), color = 'r', label = 'Computational [A]',
         linestyle = '--')
plt.plot(t_values, BC1(t_values), color = 'g', label = 'Computational [B]',
         linestyle = '--')
plt.plot(t_values, CC1(t_values), color = 'b', label = 'Computational [C]',
         linestyle = '--')
#Actual Data
plt.plot(t_data, A_c1, color = 'r', label = 'Data [A]')
plt.plot(t_data, B_c1, color = 'g', label = 'Data [B]')
plt.plot(t_data, C_c1, color = 'b', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2CONSECUTIVE1', dpi=300)

#=====Consecutive 2======#

t_values = np.linspace(0, 57)
t_data = np.linspace(0, 57, 58)

fig = plt.figure(4,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Consecutive Reactions Part 2', fontsize = '15')
#Computational Values
plt.plot(t_values, AC2(t_values), color = 'r', label = 'Computational [A]',
         linestyle = '--')
plt.plot(t_values, BC2(t_values), color = 'g', label = 'Computational [B]',
         linestyle = '--')
```

```
plt.plot(t_values, CC2(t_values), color = 'b', label = 'Computational [C]',
         linestyle = '--')
#Actual Data
plt.plot(t_data, A_c2, color = 'r', label = 'Data [A]')
plt.plot(t_data, B_c2, color = 'g', label = 'Data [B]')
plt.plot(t_data, C_c2, color = 'b', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2CONSECUTIVE2', dpi=300)

#==============================================================================#

#===========================Plotting===========================================#

#=====Parallel========#
t_values = np.linspace(0, 8)
t_data = np.linspace(0, 8, 9)
#Fits
popt, pcov = curve_fit(AP_FIT, t_data, AP(t_data))

fig = plt.figure(5,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
#Computational Values
plt.plot(t_data, AP_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
         linestyle = '--')
print(popt)
popt, pcov = curve_fit(BP_FIT, t_data, BP(t_data))

plt.plot(t_data, BP_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
         linestyle = '--')
print(popt)

popt, pcov = curve_fit(CP_FIT, t_data, CP(t_data))

plt.plot(t_data, CP_FIT(t_data, *popt), color = 'm', label = 'Fit for [C]',
         linestyle = '--')
print(popt)

#Actual Data
plt.scatter(t_data, A_p, color = 'c', label = 'Data [A]')
plt.scatter(t_data, B_p, color = 'k', label = 'Data [B]')
plt.scatter(t_data, C_p, color = 'm', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2PARALLELFITS', dpi=300)

#=====Reversible=======#
```

```

t_values = np.linspace(0, 8)
t_data = np.linspace(0, 9, 10)

popt, pcov = curve_fit(AR_FIT, t_data, AR(t_data))

fig = plt.figure(6,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
#Computational Values
plt.plot(t_data, AR_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
          linestyle = '--')
print(popt)
popt, pcov = curve_fit(BR_FIT, t_data, BR(t_data))

plt.plot(t_data, BR_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
          linestyle = '--')
print(popt)

#Actual Data
plt.scatter(t_data, A_r, color = 'c', label = 'Data [A]')
plt.scatter(t_data, B_r, color = 'k', label = 'Data [B]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2REVERSIBLEFITS', dpi=300)

#=====Consecutive 1=====#

t_values = np.linspace(0, 24)
t_data = np.linspace(0, 24, 25)

popt, pcov = curve_fit(AC_FIT, t_data, AC1(t_data), maxfev=4000)

fig = plt.figure(7,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
#Computational Values
plt.plot(t_data, AC_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
          linestyle = '--')
print(popt)
popt, pcov = curve_fit(BC_FIT, t_data, BC1(t_data), maxfev=4000)

plt.plot(t_data, BC_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
          linestyle = '--')
print(popt)

popt, pcov = curve_fit(CC_FIT, t_data, CC1(t_data), maxfev=2000)

plt.plot(t_data, CC_FIT(t_data, *popt), color = 'm', label = 'Fit for [C]',
          linestyle = '--')
print(popt)
```

```python


#Actual Data
plt.scatter(t_data, A_c1, color = 'c', label = 'Data [A]')
plt.scatter(t_data, B_c1, color = 'k', label = 'Data [B]')
plt.scatter(t_data, C_c1, color = 'm', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2CONSECUTIVE1FITS', dpi=300)

#=====Consecutive 2=====#

t_values = np.linspace(0, 57)
t_data = np.linspace(0, 57, 58)

popt, pcov = curve_fit(AC_FIT2, t_data, AC2(t_data), maxfev=4000)

fig = plt.figure(8,figsize=(15, 10))
my_fig = fig.add_subplot(111)
my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
#Computational Values
plt.plot(t_data, AC_FIT2(t_data, *popt), color = 'c', label = 'Fit for [A]',
         linestyle = '--')
print(popt)
popt, pcov = curve_fit(BC_FIT2, t_data, BC2(t_data), maxfev=4000)

plt.plot(t_data, BC_FIT2(t_data, *popt), color = 'k', label = 'Fit for [B]',
         linestyle = '--')
print(popt)

popt, pcov = curve_fit(CC_FIT2, t_data, CC2(t_data), maxfev=2000)

plt.plot(t_data, CC_FIT2(t_data, *popt), color = 'm', label = 'Fit for [C]',
         linestyle = '--')
print(popt)


#Actual Data
plt.scatter(t_data, A_c2, color = 'c', label = 'Data [A]')
plt.scatter(t_data, B_c2, color = 'k', label = 'Data [B]')
plt.scatter(t_data, C_c2, color = 'm', label = 'Data [C]')

my_fig.set_xlabel('Time (s)', fontsize = '13')
my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')

plt.legend(loc='best', fontsize = '13')
plt.savefig('PCHEML2CONSECUTIVE2FITS', dpi=300)
```