

Lab Report # 2: Simulating Simple Mechanisms

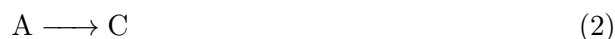
Max Huggins – UCA Department of Chemistry

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:



Reversible:



Consecutive:



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

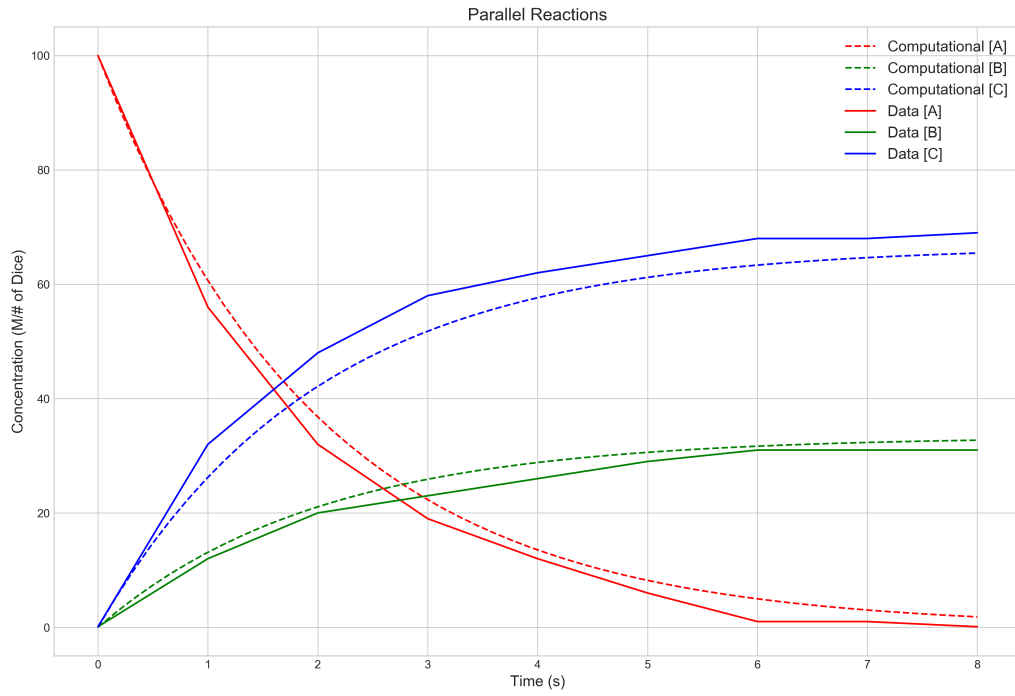


Figure 1: Concentration of species from parallel reactions as a function of time. Experimental and computed analytical results are plotted.

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](#).

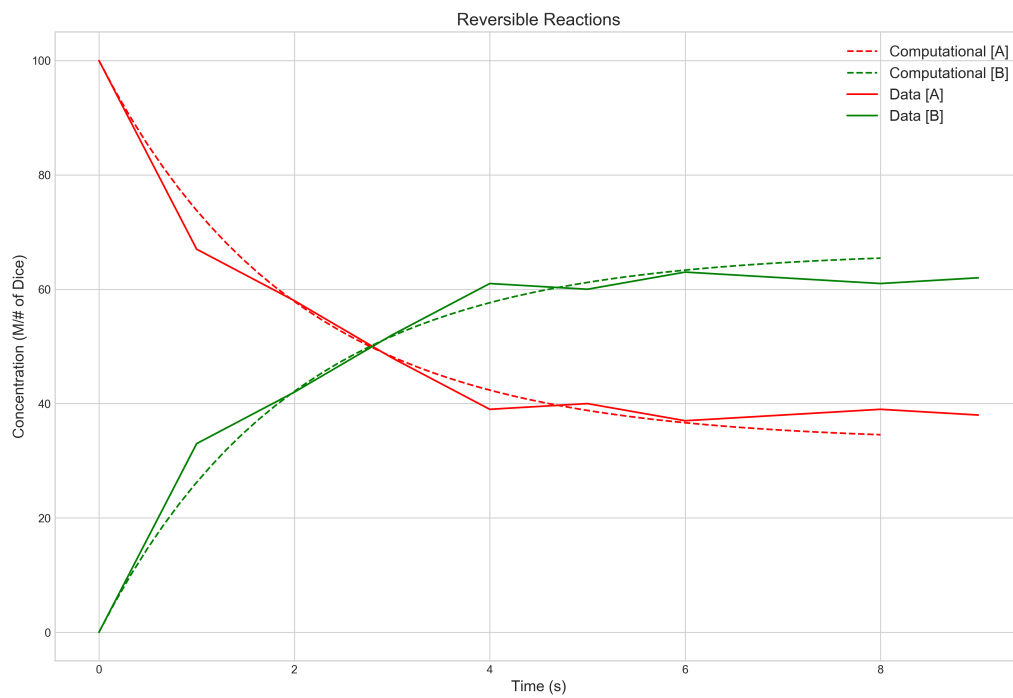


Figure 2: Concentration of species from reversible reactions as a function of time. Experimental and computed analytical results are plotted.

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				Consecutive Pt. 2				Pt. 2 Cont...				Pt. 2 Cont...			
Time	A	B	C	Time	A	B	C	Time	A	B	C	Time	A	B	C
0	100	0	0	0	100	0	0	25	2	0	98	50	2	0	98
1	17	83	0	1	82	18	0	26	2	0	98	51	2	0	98
2	2	84	14	2	70	14	16	27	2	0	98	52	1	1	98
3	0	66	34	3	60	11	29	28	2	0	98	53	1	0	99
4	0	57	43	4	49	13	38	29	2	0	98	54	1	0	99
5	0	46	54	5	43	7	50	30	2	0	98	55	0	1	99
6	0	37	63	6	33	11	56	31	2	0	98	56	0	0	100
7	0	30	70	7	27	7	66	32	2	0	98	57	0	0	100
8	0	22	78	8	18	12	70	33	2	0	98				
9	0	19	81	9	16	3	81	34	2	0	98				
10	0	16	84	10	13	4	83	35	2	0	98				
11	0	16	84	11	11	2	87	36	2	0	98				
12	0	12	88	12	10	1	89	37	2	0	98				
13	0	12	88	13	6	4	90	38	2	0	98				
14	0	11	89	14	6	1	93	39	2	0	98				
15	0	8	92	15	5	1	94	40	2	0	98				
16	0	6	94	16	4	1	95	41	2	0	98				
17	0	5	95	17	4	0	96	42	2	0	98				
18	0	5	95	18	4	0	96	43	2	0	98				
19	0	4	96	19	4	0	96	44	2	0	98				
20	0	2	98	20	3	1	96	45	2	0	98				
21	0	1	99	21	3	0	97	46	2	0	98				
22	0	1	99	22	2	1	97	47	2	0	98				
23	0	1	99	23	2	0	98	48	2	0	98				
24	0	0	100	24	2	0	98	49	2	0	98				

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

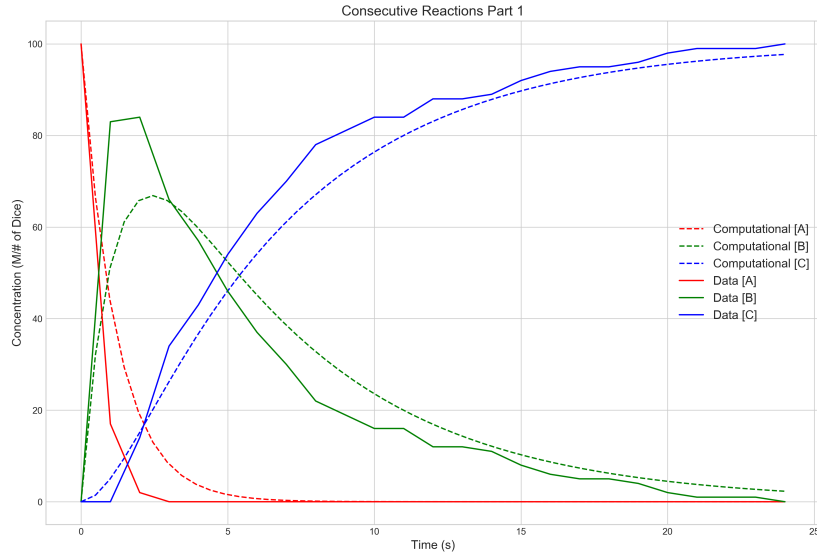


Figure 3: Concentration of species from consecutive reactions as a function of time. Experimental and computed analytical results are plotted.

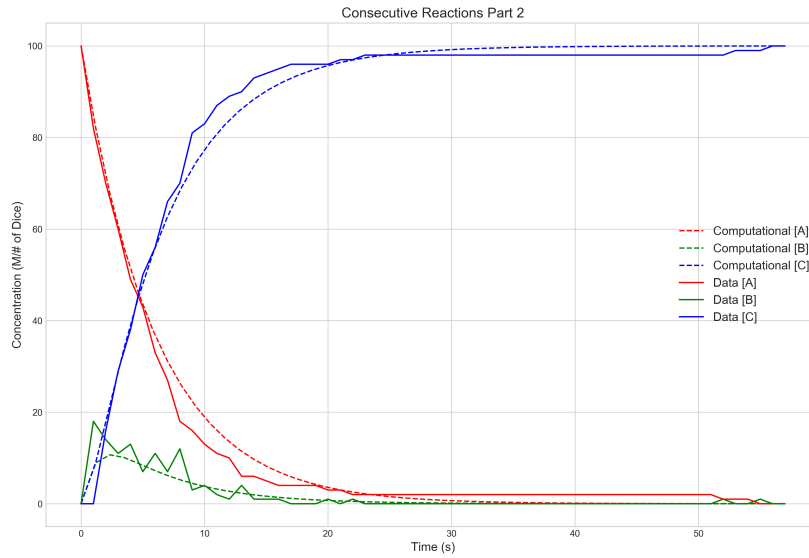


Figure 4: Concentration of species from consecutive reactions as a function of time. Experimental and computed analytical results are plotted.

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 *Physical Chemistry : Thermodynamics, Structure, and Change* chapter 20E.

For parallel reactions the concentration of each species can be expressed as functions of time and are shown below:

$$[A] = [A]_0 e^{-k_T t} \quad (6)$$

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

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

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.

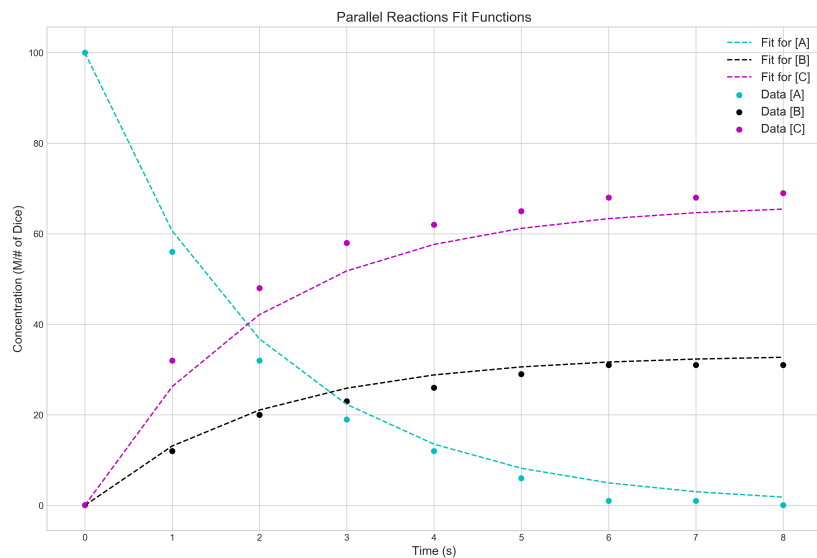


Figure 5: Best fit lines for parallel reaction simulation with dice.

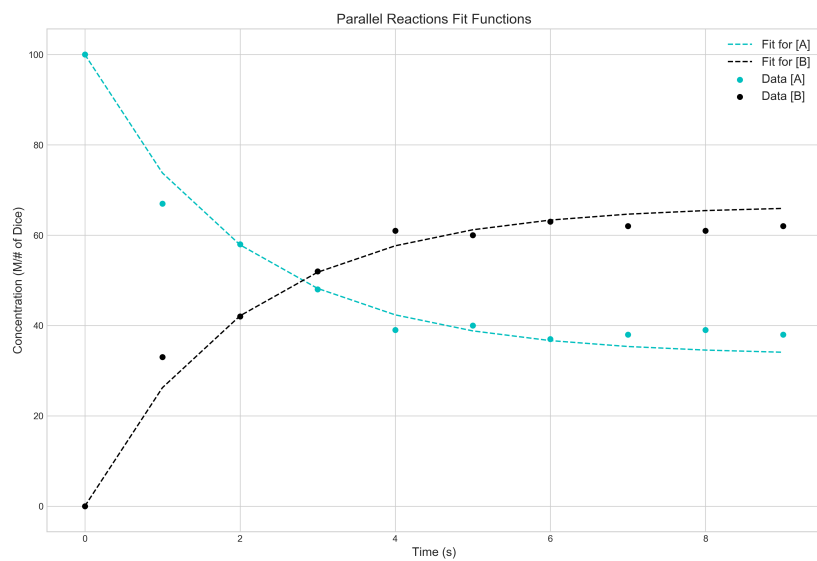


Figure 6: Best fit lines for reversible reaction simulation with dice.

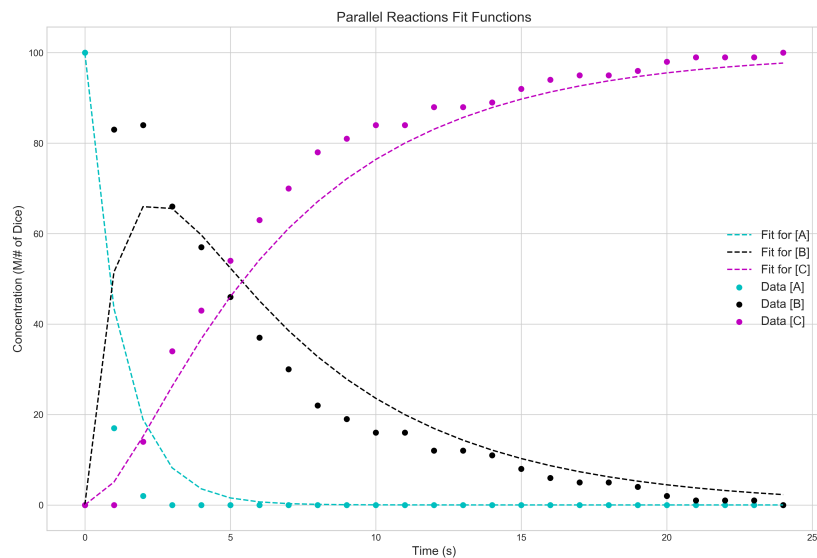


Figure 7: Best fit lines for consecutive reaction simulation with dice.

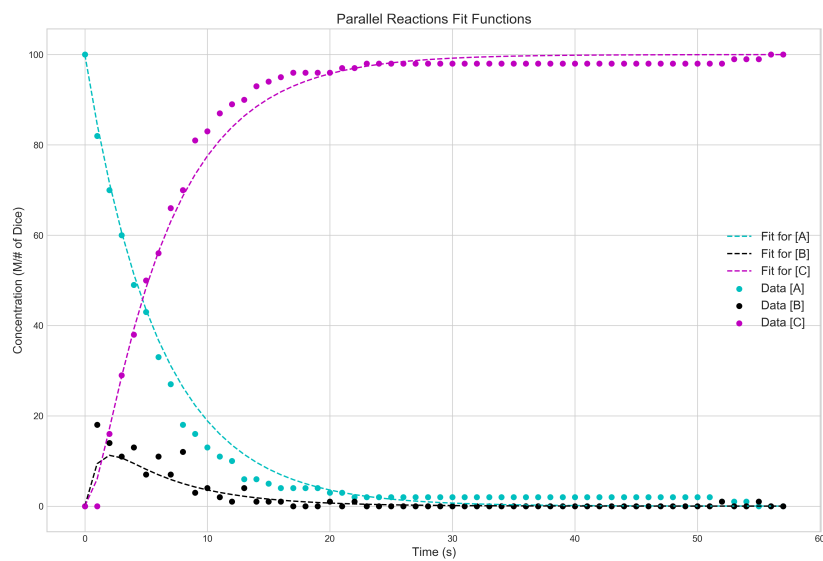


Figure 8: Best fit lines for consecutive reaction simulation with dice.

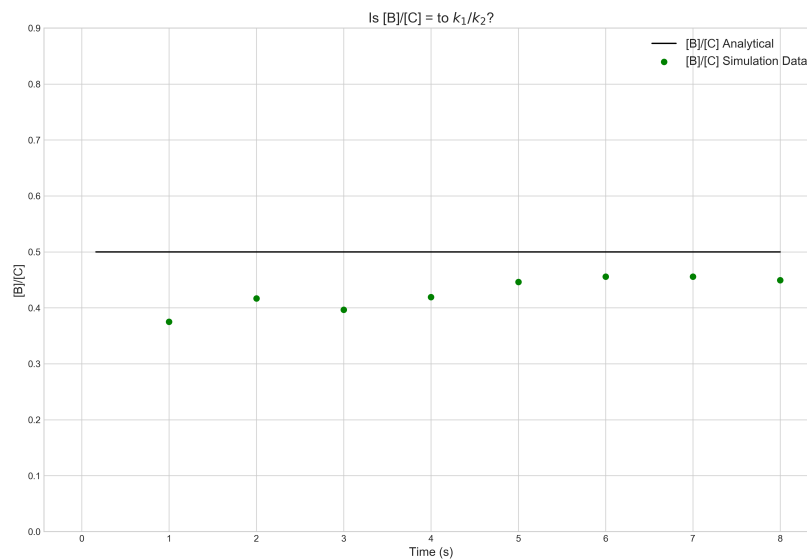


Figure 9: Concentration ratios and k constant ratios.

Branching ratios?

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

Taking average of last three values for reversible reactions:

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

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.

$$[A] = [A]_0 e^{-k_1 t} \quad (11)$$

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

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

When $k_1 > k_2$ we cannot make any assumptions, but when $k_1 \gg k_2$ 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:

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

As well as,

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

Now we will show:

$$[C] \approx [A]_0 - [A] - [B] \quad (16)$$

Substituting equations 11 and 14 into 16:

$$[C] \approx [A]_0 - [A]_0 (e^{-k_1 t} - e^{-k_2 t}) - [A]_0 e^{-k_1 t} \quad (17)$$

Factoring $[A]_0$,

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

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.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Sep 17 09:41:40 2019
4
5 @author: maxhu
6 """
7 #=====Modules=====
8 import matplotlib.pyplot as plt
9 import numpy as np
10 import pandas as pd
11 from scipy.optimize import curve_fit
12 plt.style.use('seaborn-whitegrid')
13 #=====Constants=====
14
15 #=====Parallel=====
16 A_0 = 100
17 k_1p = 1/6
18 k_2p = 2/6
19 k_T = .5
20 #=====Reversible=====
21 k_1r = 2/6
22 k_2r = 1/6
23 #=====Consecutive=pt.1=====
24 k_1c1 = 5/6
25 k_2c1 = 1/6
26 #=====Consecutive=pt.2=====
27 k_1c2 = 1/6
28 k_2c2 = 5/6
29
30 #=====Functions=====
31
32 #=====Parallel=====
33
34 def AP(t):
35     return A_0 * np.exp(-k_T * t)
36 def BP(t):
37     return (k_1p / k_T) * A_0 * (1 - np.exp(-k_T * t))
38 def CP(t):
39     return (k_2p / k_T) * A_0 * (1 - np.exp(-k_T * t))
40
41 #=====Reversible=====
42
43 def AR(t):
44     num = A_0 * (k_2r + k_1r * np.exp((-k_1r - k_2r) * t))
45     den = k_1r + k_2r
46     return num / den
47 def BR(t):
48     num = k_1r * A_0 * (1 - np.exp((-k_1r - k_2r) * t))
49     den = k_1r + k_2r
50     return num / den
51
52 #=====Consecutive=pt.1=====
```

```

53
54 def AC1(t):
55     return A_0 * np.exp(-k_1c1 * t)
56 def BC1(t):
57     num = k_1c1 * A_0 * (np.exp(-k_1c1 * t) - np.exp(-k_2c1 * t))
58     den = k_2c1 - k_1c1
59     return num / den
60 def CC1(t):
61     return A_0 * (1 - (k_1c1 / (k_2c1 - k_1c1))
62                  * (np.exp(-k_1c1 * t) - np.exp(-k_2c1 * t))
63                  - np.exp(-k_1c1 * t))
64
65 #====Consecutive=pt.2==#
66
67 def AC2(t):
68     return A_0 * np.exp(-k_1c2 * t)
69 def BC2(t):
70     return (k_1c2 / k_2c2) * A_0 * (np.exp(-k_1c2 * t) - np.exp(-k_2c2 * t))
71 def CC2(t):
72     return A_0 - A_0 * np.exp(-k_1c2 * t) - (k_1c2 / k_2c2) * A_0 * (np.exp(
73         -k_1c2 * t) - np.exp(-k_2c2 * t))
74
75 #====Fitting====#
76 def AP_FIT(x, c):
77     return 100 * np.exp(-c*x)
78
79 def BP_FIT(x, a, c):
80     return a * 100 * (1 - np.exp(-c*x))
81
82 def CP_FIT(x, a, c):
83     return a * 100 * (1 - np.exp(-c*x))
84 #====#
85 def AR_FIT(x, a, b):
86     return (100 / (a + b)) * (b + a * np.exp((-a - b) * x))
87
88 def BR_FIT(x, a, b, c):
89     return (a * 100 * (1 - np.exp((-a - b) * x))) / (a + b)
90 #====#
91 def AC_FIT(x, a):
92     return 100 * np.exp(-a*x)
93
94 def BC_FIT(x, b):
95     return (0.83333333 / (b - 0.83333333)) * 100 * (
96         np.exp(-0.83333333 * x) - np.exp(-b * x))
97
98 def CC_FIT(x, b):
99     return 100 * (1 - (0.83333333 / (b - 0.83333333)) * (
100         np.exp(-0.83333333 * x) - np.exp(-b * x)) -
101         np.exp(-0.83333333 * x))
102 #====#
103 def AC_FIT2(x, a):
104     return 100 * np.exp(-a*x)
105
106 def BC_FIT2(x, b):

```

```

107     return (.16666667 / (b - .16666667)) * 100 * (
108         np.exp(-.16666667 * x) - np.exp(-b * x))
109
110 def CC_FIT2(x, b):
111     return 100 * (1 - (.16666667 / (b - .16666667)) * (
112         np.exp(-.16666667 * x) - np.exp(-b * x)) -
113         np.exp(-.16666667 * x))
114
115 #=====Data=====#
116
117 Par_data = pd.read_excel (r'C:\Users\maxhu\Documents\PCHEM\Lab 2\Lab2.xlsx',
118                          sheet_name='Parallel')
119 Rev_data = pd.read_excel (r'C:\Users\maxhu\Documents\PCHEM\Lab 2\Lab2.xlsx',
120                          sheet_name='Reversible')
121 Cons1_data = pd.read_excel (r'C:\Users\maxhu\Documents\PCHEM\Lab 2\Lab2.xlsx',
122                          sheet_name='Consecutive pt. 1')
123 Cons2_data = pd.read_excel (r'C:\Users\maxhu\Documents\PCHEM\Lab 2\Lab2.xlsx',
124                          sheet_name='Consecutive pt. 2')
125
126 #=====Making some lists with my data=====#
127
128 A_p = pd.DataFrame(Par_data, columns= ['A'])
129 B_p = pd.DataFrame(Par_data, columns= ['B'])
130 C_p = pd.DataFrame(Par_data, columns= ['C'])
131
132 A_p = A_p.values.tolist()
133 B_p = B_p.values.tolist()
134 C_p = C_p.values.tolist()
135
136 A_p = [val for sublist in A_p for val in sublist]
137 B_p = [val for sublist in B_p for val in sublist]
138 C_p = [val for sublist in C_p for val in sublist]
139
140 #=====#
141
142 A_r = pd.DataFrame(Rev_data, columns= ['A'])
143 B_r = pd.DataFrame(Rev_data, columns= ['B'])
144
145 A_r = A_r.values.tolist()
146 B_r = B_r.values.tolist()
147
148 A_r = [val for sublist in A_r for val in sublist]
149 B_r = [val for sublist in B_r for val in sublist]
150
151 #=====#
152
153 A_c1 = pd.DataFrame(Cons1_data, columns= ['A'])
154 B_c1 = pd.DataFrame(Cons1_data, columns= ['B'])
155 C_c1 = pd.DataFrame(Cons1_data, columns= ['C'])
156
157 A_c1 = A_c1.values.tolist()
158 B_c1 = B_c1.values.tolist()
159 C_c1 = C_c1.values.tolist()
160

```

```

161 A_c1 = [val for sublist in A_c1 for val in sublist]
162 B_c1 = [val for sublist in B_c1 for val in sublist]
163 C_c1 = [val for sublist in C_c1 for val in sublist]
164
165 #=====
166
167 A_c2 = pd.DataFrame(Cons2_data, columns= ['A'])
168 B_c2 = pd.DataFrame(Cons2_data, columns= ['B'])
169 C_c2 = pd.DataFrame(Cons2_data, columns= ['C'])
170
171 A_c2 = A_c2.values.tolist()
172 B_c2 = B_c2.values.tolist()
173 C_c2 = C_c2.values.tolist()
174
175 A_c2 = [val for sublist in A_c2 for val in sublist]
176 B_c2 = [val for sublist in B_c2 for val in sublist]
177 C_c2 = [val for sublist in C_c2 for val in sublist]
178
179
180 #=====Plotting=====
181
182 #=====Parallel=====
183 t_values = np.linspace(0, 8)
184 t_data = np.linspace(0, 8, 9)
185
186 fig = plt.figure(1, figsize=(15, 10))
187 my_fig = fig.add_subplot(111)
188 my_fig.set_title('Parallel Reactions', fontsize = '15')
189 #Computational Values
190 plt.plot(t_values, AP(t_values), color = 'r', label = 'Computational [A]',
191          linestyle = '—')
192 plt.plot(t_values, BP(t_values), color = 'g', label = 'Computational [B]',
193          linestyle = '—')
194 plt.plot(t_values, CP(t_values), color = 'b', label = 'Computational [C]',
195          linestyle = '—')
196 #Actual Data
197 plt.plot(t_data, A_p, color = 'r', label = 'Data [A]')
198 plt.plot(t_data, B_p, color = 'g', label = 'Data [B]')
199 plt.plot(t_data, C_p, color = 'b', label = 'Data [C]')
200
201 my_fig.set_xlabel('Time (s)', fontsize = '13')
202 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
203
204 plt.legend(loc='best', fontsize = '13')
205 plt.savefig('PCHEML2PARALLEL', dpi=300)
206
207 #=====Reversible=====
208
209 t_values = np.linspace(0, 8)
210 t_data = np.linspace(0, 9, 10)
211
212 fig = plt.figure(2, figsize=(15, 10))
213 my_fig = fig.add_subplot(111)
214 my_fig.set_title('Reversible Reactions', fontsize = '15')

```

```

215 #Computational Values
216 plt.plot(t_values, AR(t_values), color = 'r', label = 'Computational [A]',
217          linestyle = '—')
218 plt.plot(t_values, BR(t_values), color = 'g', label = 'Computational [B]',
219          linestyle = '—')
220 #Actual Data
221 plt.plot(t_data, A_r, color = 'r', label = 'Data [A]')
222 plt.plot(t_data, B_r, color = 'g', label = 'Data [B]')
223
224 my_fig.set_xlabel('Time (s)', fontsize = '13')
225 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
226
227 plt.legend(loc='best', fontsize = '13')
228 plt.savefig('PCHEML2REVERSIBLE', dpi=300)
229
230 #=====Consecutive 1=====#
231
232 t_values = np.linspace(0, 24)
233 t_data = np.linspace(0, 24, 25)
234
235 fig = plt.figure(3, figsize=(15, 10))
236 my_fig = fig.add_subplot(111)
237 my_fig.set_title('Consecutive Reactions Part 1', fontsize = '15')
238 #Computational Values
239 plt.plot(t_values, AC1(t_values), color = 'r', label = 'Computational [A]',
240          linestyle = '—')
241 plt.plot(t_values, BC1(t_values), color = 'g', label = 'Computational [B]',
242          linestyle = '—')
243 plt.plot(t_values, CC1(t_values), color = 'b', label = 'Computational [C]',
244          linestyle = '—')
245 #Actual Data
246 plt.plot(t_data, A_c1, color = 'r', label = 'Data [A]')
247 plt.plot(t_data, B_c1, color = 'g', label = 'Data [B]')
248 plt.plot(t_data, C_c1, color = 'b', label = 'Data [C]')
249
250 my_fig.set_xlabel('Time (s)', fontsize = '13')
251 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
252
253 plt.legend(loc='best', fontsize = '13')
254 plt.savefig('PCHEML2CONSECUTIVE1', dpi=300)
255
256 #=====Consecutive 2=====#
257
258 t_values = np.linspace(0, 57)
259 t_data = np.linspace(0, 57, 58)
260
261 fig = plt.figure(4, figsize=(15, 10))
262 my_fig = fig.add_subplot(111)
263 my_fig.set_title('Consecutive Reactions Part 2', fontsize = '15')
264 #Computational Values
265 plt.plot(t_values, AC2(t_values), color = 'r', label = 'Computational [A]',
266          linestyle = '—')
267 plt.plot(t_values, BC2(t_values), color = 'g', label = 'Computational [B]',
268          linestyle = '—')

```



```

269 plt.plot(t_values, CC2(t_values), color = 'b', label = 'Computational [C]',
270          linestyle = '—')
271 #Actual Data
272 plt.plot(t_data, A_c2, color = 'r', label = 'Data [A]')
273 plt.plot(t_data, B_c2, color = 'g', label = 'Data [B]')
274 plt.plot(t_data, C_c2, color = 'b', label = 'Data [C]')
275
276 my_fig.set_xlabel('Time (s)', fontsize = '13')
277 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
278
279 plt.legend(loc='best', fontsize = '13')
280 plt.savefig('PCHEML2CONSECUTIVE2', dpi=300)
281
282 #=====
283
284 #=====Plotting=====
285
286 #=====Parallel=====
287 t_values = np.linspace(0, 8)
288 t_data = np.linspace(0, 8, 9)
289 #Fits
290 popt, pcov = curve_fit(AP_FIT, t_data, AP(t_data))
291
292 fig = plt.figure(5, figsize=(15, 10))
293 my_fig = fig.add_subplot(111)
294 my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
295 #Computational Values
296 plt.plot(t_data, AP_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
297          linestyle = '—')
298 print(popt)
299 popt, pcov = curve_fit(BP_FIT, t_data, BP(t_data))
300
301 plt.plot(t_data, BP_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
302          linestyle = '—')
303 print(popt)
304
305 popt, pcov = curve_fit(CP_FIT, t_data, CP(t_data))
306
307 plt.plot(t_data, CP_FIT(t_data, *popt), color = 'm', label = 'Fit for [C]',
308          linestyle = '—')
309 print(popt)
310
311 #Actual Data
312 plt.scatter(t_data, A_p, color = 'c', label = 'Data [A]')
313 plt.scatter(t_data, B_p, color = 'k', label = 'Data [B]')
314 plt.scatter(t_data, C_p, color = 'm', label = 'Data [C]')
315
316 my_fig.set_xlabel('Time (s)', fontsize = '13')
317 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
318
319 plt.legend(loc='best', fontsize = '13')
320 plt.savefig('PCHEML2PARALLELFITS', dpi=300)
321
322 #=====Reversible=====

```

```

323
324 t_values = np.linspace(0, 8)
325 t_data = np.linspace(0, 9, 10)
326
327 popt, pcov = curve_fit(AR_FIT, t_data, AR(t_data))
328
329 fig = plt.figure(6, figsize=(15, 10))
330 my_fig = fig.add_subplot(111)
331 my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
332 #Computational Values
333 plt.plot(t_data, AR_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
334          linestyle = '—')
335 print(popt)
336 popt, pcov = curve_fit(BR_FIT, t_data, BR(t_data))
337
338 plt.plot(t_data, BR_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
339          linestyle = '—')
340 print(popt)
341
342 #Actual Data
343 plt.scatter(t_data, A_r, color = 'c', label = 'Data [A]')
344 plt.scatter(t_data, B_r, color = 'k', label = 'Data [B]')
345
346 my_fig.set_xlabel('Time (s)', fontsize = '13')
347 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
348
349 plt.legend(loc='best', fontsize = '13')
350 plt.savefig('PCHEML2REVERSIBLEFITS', dpi=300)
351
352 #=====Consecutive 1=====
353
354 t_values = np.linspace(0, 24)
355 t_data = np.linspace(0, 24, 25)
356
357 popt, pcov = curve_fit(AC_FIT, t_data, AC1(t_data), maxfev=4000)
358
359 fig = plt.figure(7, figsize=(15, 10))
360 my_fig = fig.add_subplot(111)
361 my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
362 #Computational Values
363 plt.plot(t_data, AC_FIT(t_data, *popt), color = 'c', label = 'Fit for [A]',
364          linestyle = '—')
365 print(popt)
366 popt, pcov = curve_fit(BC_FIT, t_data, BC1(t_data), maxfev=4000)
367
368 plt.plot(t_data, BC_FIT(t_data, *popt), color = 'k', label = 'Fit for [B]',
369          linestyle = '—')
370 print(popt)
371
372 popt, pcov = curve_fit(CC_FIT, t_data, CC1(t_data), maxfev=2000)
373
374 plt.plot(t_data, CC_FIT(t_data, *popt), color = 'm', label = 'Fit for [C]',
375          linestyle = '—')
376 print(popt)

```

```

377
378
379 #Actual Data
380 plt.scatter(t_data, A_c1, color = 'c', label = 'Data [A]')
381 plt.scatter(t_data, B_c1, color = 'k', label = 'Data [B]')
382 plt.scatter(t_data, C_c1, color = 'm', label = 'Data [C]')
383
384 my_fig.set_xlabel('Time (s)', fontsize = '13')
385 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
386
387 plt.legend(loc='best', fontsize = '13')
388 plt.savefig('PCHEML2CONSECUTIVE1FITS', dpi=300)
389
390 #=====Consecutive 2=====#
391
392 t_values = np.linspace(0, 57)
393 t_data = np.linspace(0, 57, 58)
394
395 popt, pcov = curve_fit(AC_FIT2, t_data, AC2(t_data), maxfev=4000)
396
397 fig = plt.figure(8, figsize=(15, 10))
398 my_fig = fig.add_subplot(111)
399 my_fig.set_title('Parallel Reactions Fit Functions', fontsize = '15')
400 #Computational Values
401 plt.plot(t_data, AC_FIT2(t_data, *popt), color = 'c', label = 'Fit for [A]',
402         linestyle = '—')
403 print(popt)
404 popt, pcov = curve_fit(BC_FIT2, t_data, BC2(t_data), maxfev=4000)
405
406 plt.plot(t_data, BC_FIT2(t_data, *popt), color = 'k', label = 'Fit for [B]',
407         linestyle = '—')
408 print(popt)
409
410 popt, pcov = curve_fit(CC_FIT2, t_data, CC2(t_data), maxfev=2000)
411
412 plt.plot(t_data, CC_FIT2(t_data, *popt), color = 'm', label = 'Fit for [C]',
413         linestyle = '—')
414 print(popt)
415
416
417 #Actual Data
418 plt.scatter(t_data, A_c2, color = 'c', label = 'Data [A]')
419 plt.scatter(t_data, B_c2, color = 'k', label = 'Data [B]')
420 plt.scatter(t_data, C_c2, color = 'm', label = 'Data [C]')
421
422 my_fig.set_xlabel('Time (s)', fontsize = '13')
423 my_fig.set_ylabel('Concentration (M/# of Dice)', fontsize = '13')
424
425 plt.legend(loc='best', fontsize = '13')
426 plt.savefig('PCHEML2CONSECUTIVE2FITS', dpi=300)

```