

SSIE 641 Homework 03

Grant Aguinaldo

November 4, 2021

1 Introduction

In this short report, we summarize our results from the simulations that was done on the modified adaptive network on the PyCX repository.¹

As part of this work, we decided to include functionality so that the infection probability, `recoveryProb`, is not constant, but rather decreases with time. We reason that in an actual setting, the infection probability will not be constant, but will likely decrease with time since the susceptible population decreases due to natural immunity and vaccinations.

To effect the time-dependent change on the infection probability, we introduced a differential equation, and simulated the results over 100 timesteps, using the Euler forward method (see Line 56 of Appendix B). Once we integrated the differential equation into the overall model, we conducted nine (9) simulations by varying the constants `vax_const` and `recoveryProb`. The results from these simulations are provided in Figure 1, and the supporting code is provided in the Appendix.

As shown in Figure 1, the frequency of infections are highest when the `recoveryProb` are at it's lowest and the `vax_const` is at its highest (see Figure 1, Case 1). Conversely, the frequency of infections are at its lowest when the `recoveryProb` is at it's highest and the `vax_const` is at its lowest (see Figure 1, Case 9). These data suggests that the rate at which immunity probability decreases, along with the probability recovery are a good indicator of how high can the curve reach, and quickly can a population recover from a given pandemic.

¹<https://github.com/hsayama/PyCX/blob/master/net-SIS-large-graph-adaptive.py>

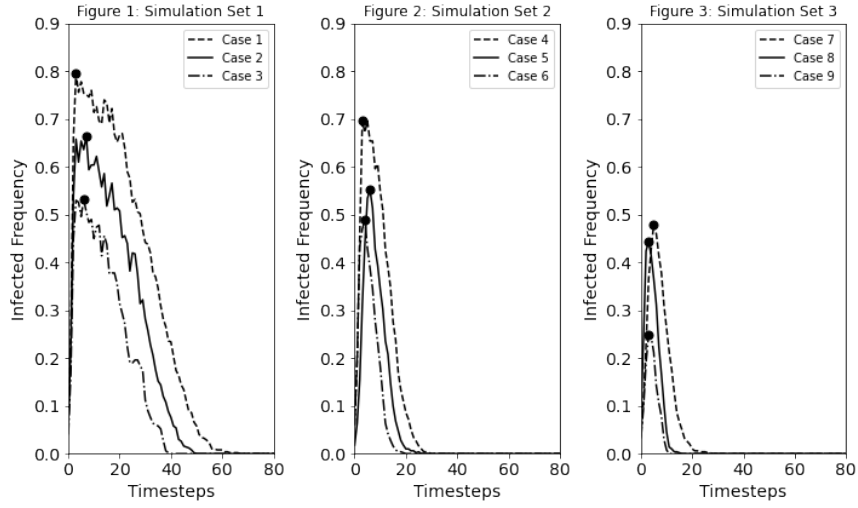


Figure 1: Summary of Results from the nine simulation runs. Case 1: $\text{recoveryProb} = 0.3$, $\text{vax_const} = 0.95$, Case 2: $\text{recoveryProb} = 0.5$, $\text{vax_const} = 0.95$, Case 3: $\text{recoveryProb} = 0.8$, $\text{vax_const} = 0.95$, Case 4: $\text{recoveryProb} = 0.3$, $\text{vax_const} = 0.85$, Case 5: $\text{recoveryProb} = 0.5$, $\text{vax_const} = 0.85$, Case 6: $\text{recoveryProb} = 0.8$, $\text{vax_const} = 0.85$, Case 7: $\text{recoveryProb} = 0.3$, $\text{vax_const} = 0.75$, Case 8: $\text{recoveryProb} = 0.5$, $\text{vax_const} = 0.75$, Case 9: $\text{recoveryProb} = 0.8$, $\text{vax_const} = 0.75$. The circles on the Figure illustrate the peak of the infected frequency curve.

A Appendix A

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [2]: #recoveryProb = 0.3
#vax_const = 0.95
# Case 1
data = [33, 126, 330, 398, 378, 389, 377, 376, 373, 380, 357, 363, 348,
        345, 370, 367, 346, 361, 334, 326, 333, 335, 320, 295, 288, 263,
        266, 253, 251, 234, 221, 219, 210, 189, 185, 173, 155, 141, 133,
        118, 117, 102, 87, 82, 72, 67, 55, 44, 43, 33, 29, 20, 17, 16, 14,
        11, 5, 5, 4, 4, 4, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
In [3]: #recoveryProb = 0.5
#vax_const = 0.95
# Case 2
data8 = [30, 118, 269, 329, 305, 327, 318, 332, 297, 302, 302, 311, 295,
        279, 293, 259, 270, 283, 255, 257, 254, 226, 228, 220, 191, 210,
        209, 188, 157, 161, 141, 129, 114, 103, 88, 76, 72, 60, 54, 44,
        38, 29, 27, 15, 13, 8, 7, 6, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
In [4]: #recoveryProb = 0.8
#vax_const = 0.95
# Case 3
data9 = [20, 85, 225, 265, 263, 248, 266, 253, 242, 245, 225, 237,
        239, 207, 228, 219, 188, 189, 189, 178, 157, 147, 136, 112,
        95, 95, 98, 98, 92, 87, 56, 45, 38, 32, 30, 29, 26, 14, 3,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
In [5]: #recoveryProb = 0.3
#vax_const = 0.85
# Case 4
data2 = [30, 96, 228, 348, 336, 348, 327, 327, 298, 303, 274, 255, 213,
        195, 174, 138, 119, 85, 67, 55, 43, 36, 25, 17, 13, 8, 5, 2, 2,
```

11/4/21, 8:21 AM

ssie-641-hw-03-simulation

[illegible]

In [6]:

```
#recoveryProb = 0.5
#vax_const = 0.85
# Case 5
data6 = [9, 32, 77, 155, 219, 267, 276, 257, 214, 197, 172, 151, 118, 104,
74, 50, 34, 24, 14, 9, 5, 5, 3, 3, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [7]:

```
#recoveryProb = 0.8
#vax_const = 0.85
# Case 6
data7 = [30, 104, 216, 235, 244, 215, 194, 173, 139, 116, 91, 61, 32, 21,
14, 8, 4, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [8]:

```
#recoveryProb = 0.3
#vax_const = 0.75
# Case 7
data3 = [18, 55, 113, 182, 214, 240, 230, 207, 189, 155, 131, 107, 88,
        64, 43, 33, 25, 16, 13, 10, 5, 3, 3, 2, 2, 2, 1, 1, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [9]:

```
#recoveryProb = 0.5  
#vax_const = 0.75  
# Case 8  
data4 = [42, 120, 212, 222, 207, 183, 156, 112, 83, 51, 23, 7, 5, 2, 2,  
          1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]
```

In [10]:

```
#recoveryProb = 0.8
#vax_const = 0.75
# Case 9
data5 = [16, 48, 95, 124, 122, 106, 71, 53, 39, 15, 4, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0]
```

In [11]:

```
plt.figure(figsize=(10, 6))

plt.subplot(1, 3, 3)
plt.plot([each/500 for each in data3], color='black', label='Case 7', linestyle='dashed')
plt.plot([each/500 for each in data4], color='black', label='Case 8', linestyle='solid')
plt.plot([each/500 for each in data5], color='black', label='Case 9', linestyle='dashdot')
plt.plot(np.argmax([each/500 for each in data3]),
np.max([each/500 for each in data3]), marker="o",
markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data4]),
np.max([each/500 for each in data4]), marker="o",
markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data5]), np.max([each/500 for each in data5]), marker="o",
markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.legend()
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylim(0, 0.9)
plt.xlim(0, 80)
plt.xlabel('Timesteps', fontsize=14)
plt.ylabel('Infected Frequency', fontsize=14)
plt.title('Figure 3: Simulation Set 3')

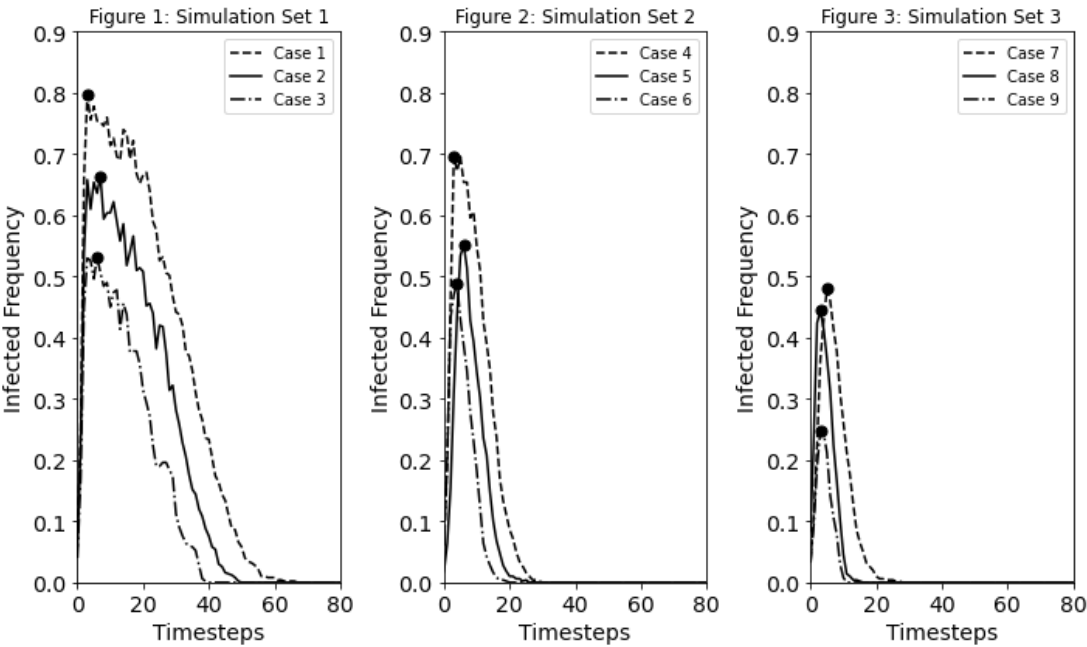
plt.subplot(1, 3, 2)
plt.plot([each/500 for each in data2], color='black', label='Case 4', linestyle='dashed')
plt.plot([each/500 for each in data6], color='black', label='Case 5', linestyle='solid')
plt.plot([each/500 for each in data7], color='black', label='Case 6', linestyle='dashdot')
plt.plot(np.argmax([each/500 for each in data2]), np.max([each/500 for each in data2]),
marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data6]), np.max([each/500 for each in data6]),
```

```
        marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data7]), np.max([each/500 for each in data7]),
        marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.legend()
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylim(0, 0.9)
plt.xlim(0, 80)
plt.xlabel('Timesteps', fontsize=14)
plt.ylabel('Infected Frequency', fontsize=14)
plt.title('Figure 2: Simulation Set 2')

plt.subplot(1, 3, 1)
plt.plot([each/500 for each in data], color='black', label='Case 1', linestyle='dashed')
plt.plot([each/500 for each in data8], color='black', label='Case 2', linestyle='solid')
plt.plot([each/500 for each in data9], color='black', label='Case 3', linestyle='dashdot')
plt.plot(np.argmax([each/500 for each in data9]), np.max([each/500 for each in data9]),
        marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data8]), np.max([each/500 for each in data8]),
        marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")
plt.plot(np.argmax([each/500 for each in data]), np.max([each/500 for each in data]),
        marker="o", markersize=7, markeredgecolor="black", markerfacecolor="black")

plt.legend()
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.ylim(0, 0.9)
plt.xlim(0, 80)
plt.xlabel('Timesteps', fontsize=14)
plt.ylabel('Infected Frequency', fontsize=14)
plt.title('Figure 1: Simulation Set 1')

plt.tight_layout()
plt.savefig('ssie-641-hw-03-simulation-figure.png')
plt.show()
```



In []:


B Appendix B

11/4/21, 8:22 AM

ssie/ssie-641-hw-03-simulation.py at master · grantaguinaldo/ssie

 [grantaguinaldo](#) / [ssie](#) Private

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

 master ▾

...

[ssie](#) / [ssie641](#) / [homework-03](#) / [ssie-641-hw-03-simulation.py](#) / [Code](#) Jump to ▾



grantaguinaldo added homework 3

 History

 1 contributor

93 lines (71 sloc) | 2.51 KB

...

```
1 import pycxsimulator
2 from pylab import *
3 import networkx as nx
4 import numpy as np
5
6 populationSize = 500
7 linkProbability = 0.01
8 initialInfectedRatio = 0.01
9 #infectionProb = 0.2 #made this dynamic, via a differential equation.
10 recoveryProb = 0.8
11 linkCuttingProb = 0.1
12 vax_const = 0.95
13
14 susceptible = 0
15 infected = 1
16
17 infected_count_list = []
```

<https://github.com/grantaguinaldo/ssie/blob/master/ssie641/homework-03/ssie-641-hw-03-simulation.py>

1/4

```
18
19 def initialize():
20     global time, network, positions, nextNetwork, x_i, results, infected_count_list
21     x_i = 1
22     results = [x_i]
23
24     time = 0
25
26     network = nx.erdos_renyi_graph(populationSize, linkProbability) #can make a BA network.
27     #network = nx.barabasi_albert_graph(n=500, m=200, seed=42)
28
29     positions = nx.random_layout(network)
30
31     for i in network.nodes:
32         if random() < initialInfectedRatio:
33             network.nodes[i]['state'] = infected
34         else:
35             network.nodes[i]['state'] = susceptible
36
37     nextNetwork = network.copy()
38
39 def observe():
40     global x_i, results, infected_count_list
41     results.append(x_i)
42
43     cla()
44     nx.draw(network,
45             pos = positions,
46             node_color = [network.nodes[i]['state'] for i in network.nodes],
47             cmap = cm.Wistia,
48             vmin = 0,
49             vmax = 1)
50     axis('image')
51     title('t = ' + str(time))
52
53
54 def update():
```

```
55     global time, network, nextNetwork, x_i, results, infected_count_list
56     x_i = vax_const * results[-1]
57
58
59
60     vax_rate = 1-x_i #growing vax rate.
61
62     #print(x_i)
63
64     time += 1
65     infected_count = 0
66
67     for i in network.nodes:
68         if network.nodes[i]['state'] == susceptible:
69             nextNetwork.nodes[i]['state'] = susceptible
70             for j in network.neighbors(i):
71                 if network.nodes[j]['state'] == infected:
72                     if random() < x_i: #infectionProb
73                         nextNetwork.nodes[i]['state'] = infected
74                         infected_count += 1
75                         break
76                 else: # adaptive link cutting behavior
77                     if random() < linkCuttingProb:
78                         if nextNetwork.has_edge(i, j):
79                             nextNetwork.remove_edge(i, j)
80             else:
81                 if random() < recoveryProb:
82                     nextNetwork.nodes[i]['state'] = susceptible
83             else:
84                 nextNetwork.nodes[i]['state'] = infected
85                 infected_count += 1
86
87     del network
88     network = nextNetwork.copy()
89     infected_count_list.append(infected_count)
90
91     print(infected_count_list)
```

11/4/21, 8:22 AM

ssie/ssie-641-hw-03-simulation.py at master · grantaguinaldo/ssie

92

93 pycxsimulator.GUI().start(func=[initialize, observe, update])