STAT3004
Project 1
William Jago

1. In the simplified Greenwood model, a susceptible at time t remains susceptible at time $t+1$ with probability α, that is to say they become infected with probability $1-\alpha$

   (a) Let I_i be 1 if individual i is susceptible, 0 otherwise. Denote $\alpha = p(I_1|I_0)$, which is the probability an individual remains susceptible after 1 step. If at time $t = 0$ there are X_0 susceptibles, the number of susceptibles at time $t = 1$ will be binomially distributed with parameters (X_0, α). This has an expectation of:

   $$\mathbb{E}[X_1|X_0] = \alpha X_0$$

   Consider now time t. The probability an individual I remains susceptible after t steps is given by $p(I_t|I_0) = p(I_t|I_{t-1})p(I_{t-1}|I_{t-2})\dots p(I_1|I_0) = \alpha^t$. As there are X_0 initial susceptibles and each one has the same α^t probability of remaining susceptible, taking the binomial expectation as before gives an expectation for X_t of:

   $$\mathbb{E}[X_t|X_0] = \alpha^t X_0$$

   Using a similar method for the number of infected, start by noting that $(1-\alpha)$ is the probability a susceptible becomes infective after 1 step, so

   $$\mathbb{E}[Y_t|X_{t-1}] = (1-\alpha)X_{t-1}$$

   Here however, for an individual to become infected after t steps, they need to remain susceptible for $t-1$. From before,

   $$\mathbb{E}[X_{t-1}|X_0] = \alpha^{t-1}X_0$$

   And applying the expectation of infectives for one step gives the desired result

   $$\Rightarrow \mathbb{E}[Y_t|X_0] = \alpha^{t-1}(1-\alpha)X_0$$

   (b) Given $X_0 = 6$, $\alpha = 0.8$, the following plot shows the expected values for X_t and Y_t over $t = 0, 1, \dots, 9$

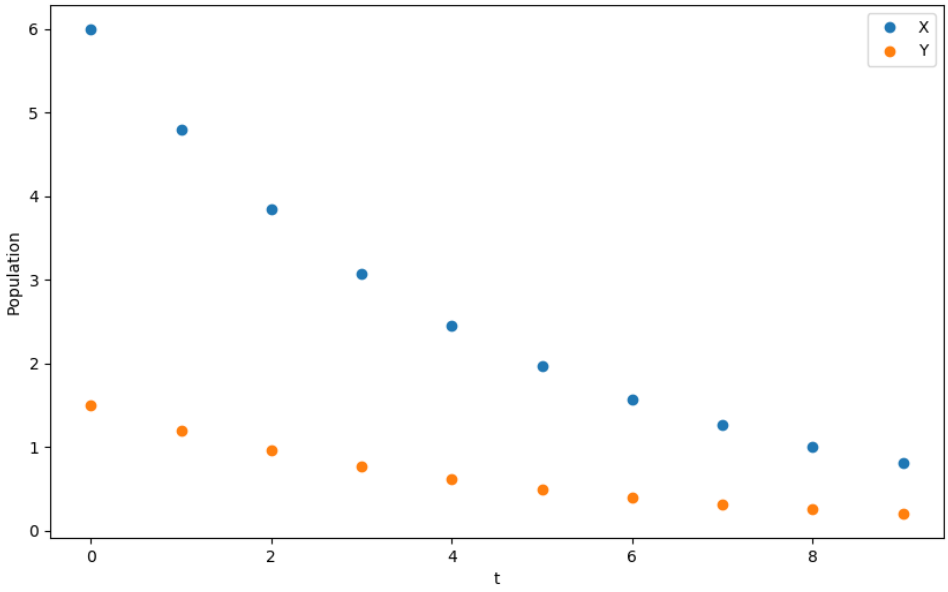

Figure 1: Greenwood expectation

```
import numpy as np
import matplotlib.pyplot as plt

alpha = 0.8
x_0 = 6
t = [i for i in range(0,10)]
x = [alpha ** (t[i]) * x_0 for i in range(len(t))]
y = [alpha ** (t[i] - 1) * (1-alpha) * x_0 for i in range(len(t))]

plt.plot(t,x, 'o', t, y, 'o')
plt.xlabel('t')
plt.ylabel('Population')
plt.legend(['X', 'Y'])
plt.show()
```

2. The Reed-Frost model differs from the Greenwood model in that a susceptible only remains susceptible if they avoid contact with all infectives, which occurs with probability α^{Y_t}, that is to say they become infected with probability $1 - \alpha^{Y_t}$.

   (a) The distribution of the Reed-Frost model is given by $p_{(x,y)_{t+1},(x,y)_t} = \mathbb{P}\left((X,Y)_{t+1} = (x,y)_{t+1} | (X,Y)_t = (x,y)_t\right)$:

   $$p_{(x,y)_{t+1},(x,y)_t} = \binom{x_t}{x_{t+1}} \alpha^{y_t x_{t+1}} (1 - a^{y_t})^{y_{t+1}}$$

   Recall that Y_{t+1} is equal to $X_t - X_{t+1}$. The distribution $p_{x_{t+1},(x,y)_t}$ can therefore be written as

   $$p_{x_{t+1},(x,y)_t} = \binom{x_t}{x_{t+1}} \alpha^{y_t x_{t+1}} (1 - a^{y_t})^{x_t - x_{t-1}}$$

   This is simply a binomial distribution with $X_{t+1} \sim \text{Bin}(x_t, \alpha^{y_t})$, with binomial expectation:

   $$\mathbb{E}\left[X_{t+1} | (X_t, Y_t)\right] = \alpha^{y_t} x_t$$

   As $Y_{t+1} = X_t - X_{t+1}$, Y_{t+1} is also binomially distributed with

   $$\begin{aligned} p_{y_{t+1},(x,y)_t} &= \binom{x_t}{x_t - x_{t+1}} \alpha^{y_t x_{t+1}} (1 - a^{y_t})^{x_t - x_{t-1}} \\ &= \binom{x_t}{x_t - x_{t+1}} (1 - a^{y_t})^{x_t - x_{t-1}} (\alpha^{y_t})^{x_t - (x_t - x_{t+1})} \\ &= \binom{x_t}{y_{t+1}} (1 - a^{y_t})^{y_{t-1}} (\alpha^{y_t})^{x_t - y_{t+1}} \end{aligned}$$

   So $Y_{t+1} = X_t - X_{t+1}$ is also binomially distributed but with the complementary probability $1 - \alpha^{y_t}$, so $Y_{t+1} \sim \text{Bin}(x_t, (1 - \alpha^{y_t}))$ which has expectation:

   $$\mathbb{E}\left[Y_{t+1} | (X_t, Y_t)\right] = (1 - \alpha^{y_t}) x_t$$

   (b) The following plot shows the expected values for X_t and Y_t over $t = 0, 1, \ldots, 15$. It can be seen that eventually the points converge where $Y \to 0$ and $X \approx 19$.

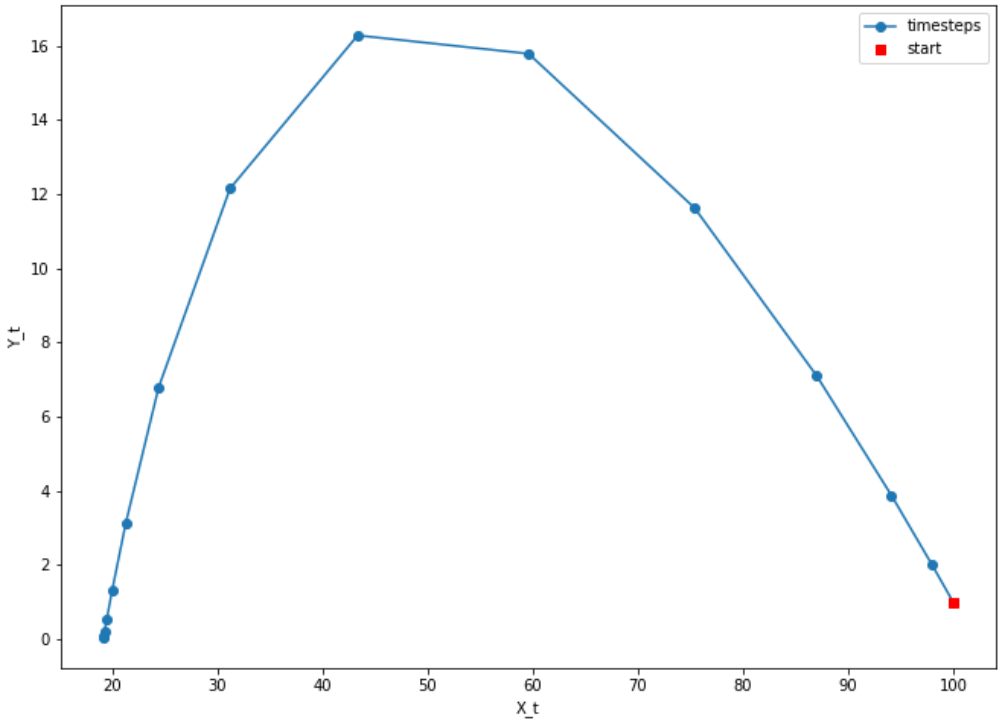

Figure 2: Reed-Frost expectation

```
import numpy as np
import matplotlib.pyplot as plt

alpha = 0.98
x = [100]
y = [1]
t = [0]

for i in range(1,15):
    t.append(i)
    x.append(alpha ** y[i-1] * x[i-1])
    y.append(x[i-1] - x[i])

plt.figure(figsize=(11,8))
plt.plot(x,y, 'o-', [100], [1], 'rs')
plt.xlabel('X_t')
plt.ylabel('Y_t')
plt.legend(['timesteps', 'start'])
plt.savefig("q2bfig")
plt.show()
```

3. (a) The following heatmaps were produced with $\alpha = 0.8$.

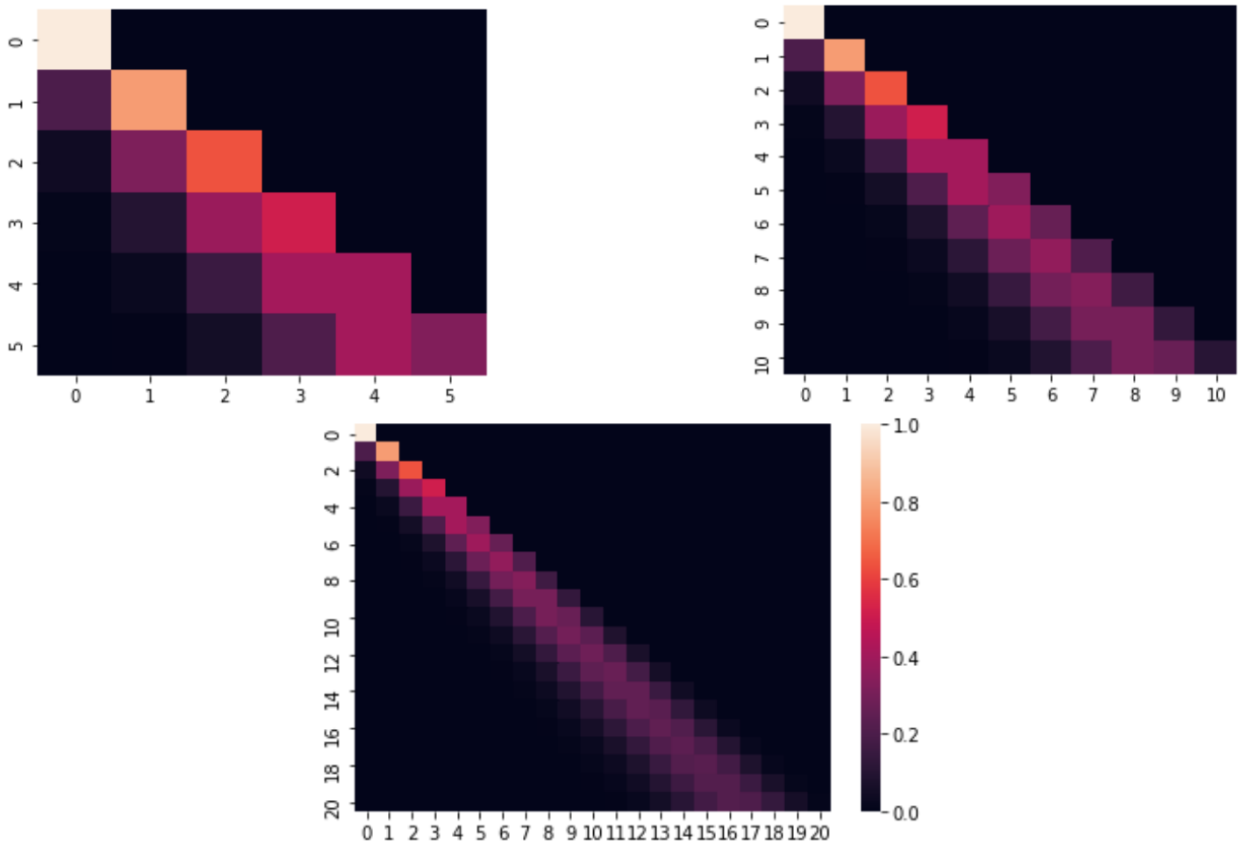

Figure 3: (Clockwise from top left) $x_0 = 5, 10, 20$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb
import seaborn as sns
sns.set(rc={'figure.figsize':(11.7,8.27)})

alpha = 0.8
x_0 = 10
P = []

for i in range(x_0+1):
    row = []
    P.append(row)
    for j in range(x_0+1):
        P[i].append(comb(i,j) * (1-alpha)**(i-j) * alpha**j)

sns_plot = sns.heatmap(P)
```

```
fig = sns_plot.get_figure()
fig.savefig("q3a.png")
plt.show()
```

(b) There are $x_0 + 1$ communicating classes (one for each $X_t = 0, 1, \ldots, x_0$), which are located along the diagonal. All are transient except for $P_{0,0} = 1$ which is recurrent - this is the probability the population remains at 0 given it was previously 0. Intuitively, this means that the population is non-increasing. At each step, the population can either stay the same or get smaller, but not bigger. Each nonzero population size is transient as there is a chance the population becomes smaller, meaning the population will never return to that size. Eventually, if the population reaches 0 it will stay there (this is the recurrent class).

(c) For this question, the expectation of X_t for different t values was calculated using the formula given:

$$\mathbb{E}[X_t] = e_{x_0+1}^T P^t v$$

where the matrix P (which has entries $p_{ij} = \binom{i}{j}(1-\alpha)^{i-j}\alpha^j$) and the vectors e_{x_0+1}, v were generated using python code, and the formula was evaluated using numpy's linear algebra package. The difference between the result found using this method and the results from 1b was found to give the error. This is shown in the table below part (d) as the code was written to calculate both $\mathbb{E}[X_t]$ and $\mathbb{E}[Y_t]$. It can be seen that both methods produced identical results, however the first method is far more efficient and simplistic than the second. The second method works by finding the t step transition matrix, and by multiplying P^t by e_{x_0+1} (e_{x_0+1} is the initial distribution for i at $t = 0$), it returns the bottom row of the matrix which represents $\mathbb{P}(X_t = 0, 1, \ldots, x_0 | X_0 = x_0)$. Multiplying this by the v vector sums the probabilities times each possible number of susceptibles to give the expectation.

(d) To find $\mathbb{E}[Y_t]$, a similar method was used however this time it aimed to find the expectation $\mathbb{E}[X_{t-1} - X_t] = \mathbb{E}[X_{t-1}] - \mathbb{E}[X_t]$. To do this, the formula was modified to become $e_{x_0+1}^T(P^{t-1}(I - P))v$, where I is the identity matrix. The numerical results of both $\mathbb{E}[X_t]$ and $\mathbb{E}[Y_t]$ are shown in the table below.

t	**E**$[X_t]$	**E**$[Y_t]$	**error(X)**	**error(Y)**
0	6.0	1.5	0.0	-0.0
1	4.8	1.2	1e-15	0.0
2	3.84	0.96	1e-15	-0.0
3	3.072	0.768	0.0	0.0
4	2.4576	0.6144	0.0	0.0
5	1.96608	0.49152	-0.0	0.0
6	1.572864	0.393216	-0.0	0.0
7	1.258291	0.314573	-0.0	0.0
8	1.006633	0.251658	-0.0	-0.0
9	0.805306	0.201327	-0.0	0.0

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb

alpha = 0.8
x_0 = 6
P = []

for i in range(x_0 + 1):
  row = []
  P.append(row)
  for j in range(x_0 + 1):
    P[i].append(comb(i,j) * (1-alpha)**(i-j) * alpha**j)

e_x01, v_x, v_y = [], [], []
e_x01.append([])
for i in range(x_0 + 1):
  e_x01[0].append(0)
  v_x.append([i])
  v_y.append([x_0 - i])
e_x01[0][x_0] = 1

e_x01 = np.array(e_x01)
P = np.array(P)
v_x = np.array(v_x)
v_y = np.array(v_y)

for t in range(0, 10):

    exp_x = e_x01.dot(np.linalg.matrix_power(P, t).dot(v_x))[0][0]
    exp_y = (1-alpha) * e_x01.dot(np.linalg.matrix_power(P, t-1).
       dot(v_x))[0][0]
    exp_y = e_x01.dot((np.linalg.matrix_power(P, t-1)-np.linalg.
       matrix_power(P,t)).dot(v_x))[0][0]
    error_x = alpha ** t * x_0 - exp_x
    error_y = (1 - alpha) * alpha ** (t-1) * x_0 - exp_y

    print("\\textbf{",t,"} &",round(exp_x,6),"&",round(exp_y,6),"&"
       ,round(error_x, 15),"&",round(error_y, 14)," \\\ \hline")
```

4. (a) This recursive equation p_j^t represents the probability that the number of susceptibles X_t at time t is equal to j, given that there was at least 1 infective. The base case for $t = 0$ is simple: if $j = X_0$ the result is 1, otherwise it is 0. For $t \geq 1$, it finds the probability recursively by summing the probabilities that $X_t = j$ given $X_{t-1} = i$. Here, i takes values from $j + 1$ (as the previous population was at least one greater than it is now) to $x_0 - (t - 1)$ (as the maximum possible size of the population reduces by 1 each step).

   (b) Python code was written for this question to define the recursive function (`p_jt(j,t)`) and the joint distribution $\Gamma(k, n|x_0)$ (`P_WT(k,n,i)`). The probability $P(W \leq 4)$ was found by summing $\Gamma(k, n|x_0)$ over $k = 0, 1, \ldots, 5$ for $n = 1, 2, \ldots, x_0 + 1$. Note than convergence will be reached after at most $n = x_0 + 1$, as it can be assumed that the population decreases by at least 1 at each step if $Y_t > 0$. This value was then subtracted from 1 to give the result $\mathbb{P}(W > 4) = 0.130298$.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb

alpha = 0.8
x_0 = 6
P = []

for i in range(x_0 + 1):
    row = []
    P.append(row)
    for j in range(x_0 + 1):
        P[i].append(comb(i,j) * (1-alpha)**(i-j)*alpha**j)

def p_jt(j,t):
    result = 0
    if t == 0:
        if j == x_0:
            return 1
        else:
            return 0
    for i in range(j+1, x_0-(t-1)+1):
        result += P[i][j] * p_jt(i,t-1)
    return result

def P_WT(k,n,i):
    return alpha**(i-k) * p_jt(i-k,n-1)

p = 0
for W in range(0,5):
```

```
    for n in range(1,x_0 + 1):
        a = P_WT(W, n, x_0)
        p += a

print(1-p)
```

(c) Using 10^6 Monte Carlo simulations, the epidemic was simulated and the proportion of simulations where $W > 4$ was 0.1302243, which is the same as the numerical result to 3 significant figures.

```
import numpy as np
import random

alpha = 0.8
x_0 = 6
W, T = [], []
n = 10**6

for simulation in range(n):
    x_t = x_0
    y_t = 1
    t = 0
    while (x_t * y_t) > 0:
        x_next = 0
        for individual in range(x_t):
            p = random.random()
            if (p < alpha):
                x_next += 1
        y_t = x_t - x_next
        x_t = x_next
        t += 1
    W.append(x_0 - x_t)
    T.append(t)

count = 0
for i in range(len(W)):
    if W[i] > 4:
        count+=1

print(count/n)
```

(d) Defining the PGF as in the book and setting $x_0 = 6$, $\alpha = 0.8$ gives

$$\Psi_W(\phi) = A'(I - \bar{P}(\phi))^{-1}QE$$

With the vectors A, QE and matrix $\bar{P}(\phi)$ as defined in EM4. This gives the probability generating function for the distribution of W. Currently, the vector QE is equal to

$$QE = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{pmatrix}'$$

The PGF in its current form gives $\mathbb{P}(W = 6) + \mathbb{P}(W = 5) + \ldots + \mathbb{P}(W = 0) = 1$. By augmenting the QE vector to become

$$QE = \begin{pmatrix} 1 & \alpha & 0 & 0 & 0 & 0 & 0 \end{pmatrix}'$$

The terms $\mathbb{P}(W = 4)$, $\mathbb{P}(W = 3)$, $\ldots \mathbb{P}(W = 0)$ will be removed, leaving only $\mathbb{P}(W = 6) + \mathbb{P}(W = 5) = \mathbb{P}(W > 4)$. This calculation was carried out on Python using the following code, and the result was 0.130298, which is identical to the result from Q4(b).

```
import numpy as np
from scipy.special import comb

def cdf_W(phi, Pbar, x_0, alpha, W):
    A = [[0 for i in range(x_0)]]
    A[0].append(1)
    A = np.array(A)

    for i in range(x_0 + 1):
        for j in range(x_0 + 1):
            if i > j:
                Pbar[i][j] = Pbar[i][j] * phi**(i-j)
    I = np.identity(x_0 + 1)

    QE = [[alpha**i] for i in range(x_0 - W)]
    for i in range(x_0 - W, x_0 + 1):
        QE.append([0])
    QE = np.array(QE)

    result = A.dot(np.linalg.matrix_power((I-Pbar),-1)).dot(QE)
    return result[0][0]

alpha = 0.8
x_0 = 6
P = []
W = 4

for i in range(x_0 + 1):
    row = []
```

```
    P.append(row)
    for j in range(x_0 + 1):
        if (i == j):
            P[i].append(0)
        else:
            P[i].append(comb(i,j)*(1-alpha)**(i-j)*alpha**j)
P = np.array(P)

print(cdf_W(1, P, x_0, alpha, W))
```

5. (a) The state space of the Reed-Frost model is the combination of all possible values for $(X,Y)_t$ and $(X,Y)_{t+1}$, subject to $Y_{t+1} + X_{t+1} = X_t$. The probability $\mathbb{P}((X,Y)_{t+1} = (l,j)|(X,Y)_t = (k,i))$ is given by the matrix entry P_{mn} where $m = i \times (x_0+1) + k, n = j \times (x_0+1) + l$.

   (b) The communicating classes are the recurrent classes where $i = j = 0, k = l$ (where there are no more infectives and the population stays at its current level). These are the diagonals of the submatrix P_{00}. Similarly to the Greenwood model, there is a communicating class for each population size from 0 to x_0, except each state communicates only with itself given that there are no infectives.

   (c) The following plot shows the 1 step transition probabilities $p_{(x,y)_t,(x,y)_{t+1}}$. The code used to generate it is given below.

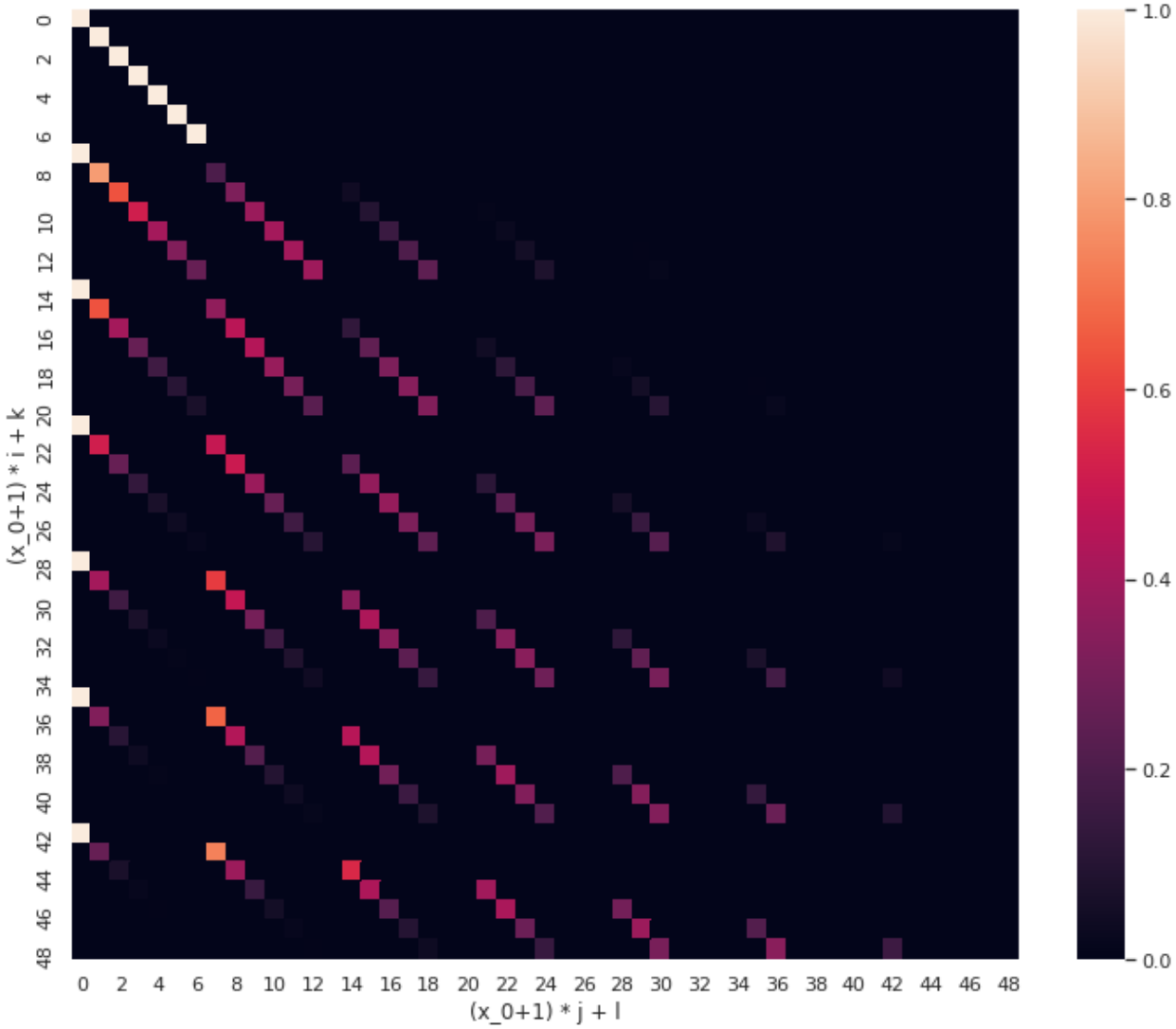

Figure 4: Reed-Frost 1 step heatmap

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb
import seaborn as sns
sns.set(rc={'figure.figsize':(12,9.9)})

alpha = 0.8
x_0 = 6

P = []
for m in range((x_0+1)**2):
    P.append([])
    i = m // (x_0+1)
    k = m % (x_0+1)
    for n in range((x_0+1)**2):
        j = n // (x_0+1)
        l = n % (x_0+1)
        if j+l==k:
            P[m].append(comb(k,l)*(1-alpha**i)**j*alpha**(i*l))
        else:
            P[m].append(0)

sns_plot = sns.heatmap(P)
fig = sns_plot.get_figure()
fig.savefig("q5c.png")
plt.show()
```

(d) Python code was written to generate $n = 10^6$ simulations of the epidemic with $X_0 = 6$, $Y_0 = 1$, and the proportion of epidemics where $W > 4$ was 0.2566889. In comparison to the Greenwood simulation, which had $\mathbb{P}(W > 4) = 0.1302243$, this is close to double. The reason for this increased proportion of large epidemics is that there are more sources of transmission as a susceptible can catch the disease from each infected with a probability of $1 - \alpha$, whereas in the Greenwood model α is the probability of catching the disease from any source and is independent of the number of infected. Because of this, the Reed-Frost model also differs in that the number of infected at each period, including at $t = 0$, has a significant effect on the course of the epidemic whereas in the Greenwood model there is no distinction between having 1 or 100 infected, as long as it is > 0.

```
import numpy as np
import random

alpha = 0.8
x_0 = 6
y_0 = 1
```

```
W, T = [], []
n = 10**7

for simulation in range(n):
    x_t = x_0
    y_t = y_0
    t = 0
    while (x_t * y_t) > 0:
        x_next = 0
        for susceptible in range(x_t):
            contact = False
            for infected in range(y_t):
                p = random.random()
                if (p > alpha):
                    contact = True
                    break
            if contact is False:
                x_next += 1
        y_t = x_t - x_next
        x_t = x_next
        t += 1
    W.append(x_0 - x_t)
    T.append(t)

count = 0
for i in range(len(W)):
    if W[i] > 4:
        count+=1

print(count/n)
```

6. This scenario is of a town during the COVID-19 epidemic, which has a single motel to house new arrivals. It is assumed each batch of arrivals contains x_0 outsiders of which each has an $\eta = 0.05$ probability that they are infective, and the town only allows in one batch at a time. With the motel, each new batch is held for a certain number of days, testing every day with results arriving the following day until the results show no infectives present. This prevents all infectives from ever being allowed into the town. Additional assumptions are that there is no immunity among susceptibles, and that test results are always on time and accurate.

   Without the motel, the expected infection rate per person is $\eta = 0.05$, so the expected number of infections per day (as one batch is allowed in each day) is $x_0\eta$. With the motel, all x_0 people will be placed into the motel and the virus will be allowed to spread its course among the motel residents, following a Reed-Frost model with probability of contact $p = 0.1$ and probability of infection $\beta = 0.05$. This gives a probability of no infection from a single infective $\alpha = 1 - p\beta = 0.995$. When test results arrive the following day, infectives will be removed from the motel which is in line with the Reed-Frost model. The total size of the epidemic inside the motel W is given by summing over the number of new infectives for each day, and the duration of the stay in the motel is given by $T + 1$, which is the number of days taken before $Y_T = 0$. At the end of the motel epidemic, any remaining susceptibles are allowed to leave before a new batch is processed. The rate of infectives coming out of the motel per person who entered is given by $\frac{W}{x_0}$, and the rate of infectives coming out of the motel per day is given by $\frac{W}{T}$.

   For $x_0 = 1, 2, \ldots, 10$, a Monte Carlo method ($n = 10^6$, each simulation being one batch) was used to find the expected infection rate per person and per day. Additionally, a numerical method was used to also find the expected infection rate per person. The MC simulations each modelled a Reed-Frost epidemic within the motel using a similar program as in Q5(d). The numerical matrix method used the formula

$$\mathbb{E}[X_{\text{end}}|x_0] = e^T_{(x_0+1)^2} P^{x_0+1} v$$

   where $e^T_{(x_0+1)^2}$ is the initial distribution: for m from 0 to $(x_0 + 1)^2 - 1$, setting $i = \lfloor m(x_0 + 1)^{-1} \rfloor$, $k = m \bmod (x_0 + 1)$, if $i + k = x_0$ the m-th entry is $\binom{x_0}{i}\eta^i(1 - \eta)^k$. The P matrix is the matrix as defined in Q5(c) taken to the $(x_0 + 1)$-th power - it is taken to this power as by this time (or earlier) it will have converged to the stationary distribution. Finally, v is the $(x_0+1)^2$ dimensional vector $[0, 1, \ldots, x_0, 0, 1, \ldots, x_0, \ldots]$. This formula finds the expectation for the number of susceptibles remaining, so the infection rate per person is simply

$$\frac{x_0 - \mathbb{E}[X_{\text{end}}|x_0]}{x_0}$$

   The results are shown in the table and Figure 5, and compared against results without any motel. It can be seen that as x_0 increases, the infection rate per person coming out of the motel increases and is larger than the base rate of 0.05. This is because

	Infections per person			Infections per day	
x0	**MC w/motel**	**Numerical w/motel**	**w/o motel**	**MC w/motel**	**w/o motel**
1	0.049725	0.050000	0.05	0.049725	0.05
2	0.050154	0.050238	0.05	0.091611	0.1
3	0.050372	0.050477	0.05	0.132159	0.15
4	0.050770	0.050719	0.05	0.170868	0.2
5	0.050759	0.050963	0.05	0.206286	0.25
6	0.051128	0.051210	0.05	0.241247	0.3
7	0.051409	0.051458	0.05	0.274435	0.35
8	0.051775	0.051709	0.05	0.306708	0.4
9	0.051885	0.051963	0.05	0.336803	0.45
10	0.052179	0.052219	0.05	0.366764	0.5

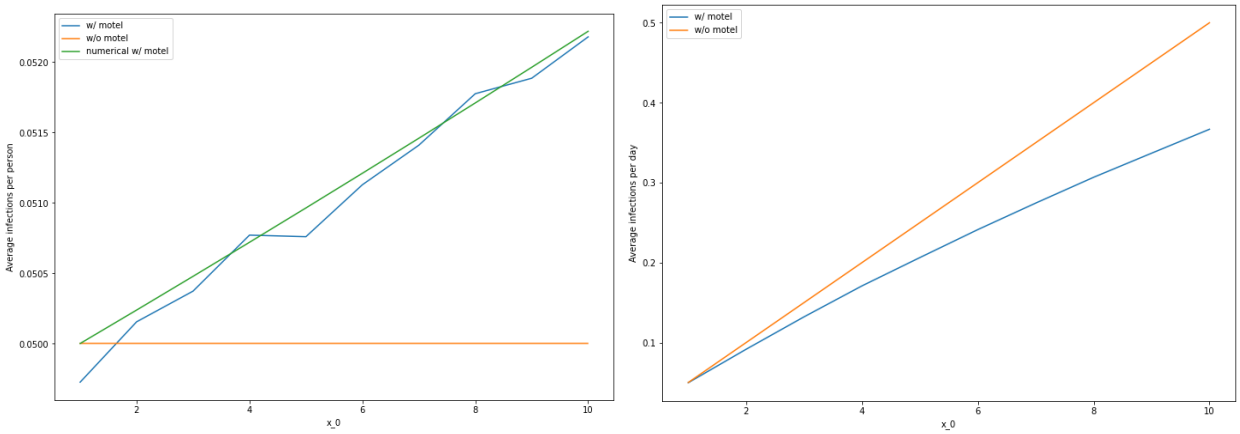

Figure 5: Infections per person (left) and per day (right)

quarantining new arrivals in the motel increases the chance of them spreading the disease amongst themselves. However, due to the increased processing time inside the motel, the extra days means that less batches are being processed in the same time frame so the infection rate per day is less than it would be without the motel. The first rate appears to be increasing linearly with x_0 but the second rate has a slightly decreasing gradient, however due to the small domain this may not be accurate for larger x_0.

Clearly, this town has adopted a policy which protects itself from any new infectives, but significantly increases the risk outsiders have of catching the disease. There are also several aspects of the model which could be considered unrealistic: the lack of immunity, the fixed values of x_0, η, p and β and the seemingly infinite external healthcare resources for sick people leaving the motel. One could also predict that the number of new arrivals to the town might decrease when they find out about the policy of imprisoning new arrivals to spread the disease amongst themselves. More practical results could be obtained by using a more complex model than the simplistic Reed-Frost, and by analysing cases where the parameters change over time.