Question 1

a)

RTS:

$$1)\ E[X_t|X_0 = x_0] = \alpha^t x_0$$
$$2)\ E[Y_t|X_0 = x_0] = \alpha^{t-1}(1-\alpha)x_0$$

In the Greenwood model. We are given that $E[X_t|X_{t-1}] = \alpha X_t$.

First, lets derive 1).

Since both X_t *and* X_{t-1} are dependent on $X_0 = x_0$, we can condition the expectation $E[X_t|X_{t-1}]$ on $X_0 = x_0$ via the Tower property for the conditional expectation:

$$E[X_t|X_0 = x_0] = E[E[X_t|X_{t-1}]|X_0 = x_0]$$

Since $E[X_t|X_{t-1}] = \alpha X_t$, we have:

$$\begin{aligned} &= E[\alpha X_{t-1}|X_0 = x_0] \\ &= \alpha E[X_{t-1}|X_0 = x_0] \\ &= \alpha^2 E[X_{t-2}|X_0 = x_0] \end{aligned}$$

Continuing in this manner, we see that:

$$E[X_t|X_0 = x_0] = \alpha^t x_0$$

Now for 2), recall that $Y_t = X_{t-1} - X_t$. So, we have via the linearity of the expectation:

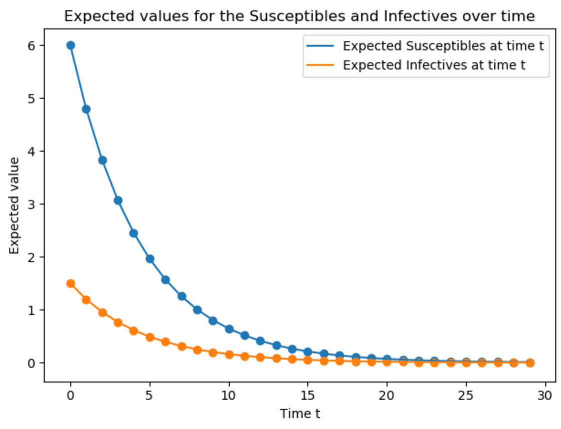
$$\begin{aligned} E[Y_t|X_0 = x_0] &= E[X_{t-1} - X_t|X_0 = x_0] \\ &= E[X_{t-1}|X_0 = x_0] - E[X_t|X_0 = x_0] \end{aligned}$$

Applying the formula from 1) to both terms in this equation yields:

$$\begin{aligned} &= \alpha^{t-1}x_0 - \alpha^t x_0 \\ &= \alpha^{t-1}(1-\alpha)x_0 \end{aligned}$$

b)

A plot of these expected values for $x_0 = 6, \alpha = 0.8$ is below. To generate the expected values of X_t and Y_t at time step t, I compute $E[X_t|X_0 = x_0] = \alpha^t x_0$ and $[Y_t|X_0 = x_0] = \alpha^{t-1}(1-\alpha)x_0$ respectively.



The code to generate this is below:

```
import numpy as np
import matplotlib.pyplot as plt
# Basic Green
x0 = 6
alpha = 0.8
t = 30

def expectation_X(t):
    return x0 * alpha ** t

def expectation_Y(t):
    return x0 * alpha ** (t-1) - x0 * alpha ** t

eX = np.vectorize(expectation_X)
eY = np.vectorize(expectation_Y)
times = np.arange(0, t)
expected_y = eY(times)
expected_x = eX(times)

plt.scatter(times, expected_x)
plt.plot(times, expected_x)
plt.scatter(times, expected_y)
plt.plot(times, expected_y)

plt.title("Expected values for the Susceptibles and Infectives over time")
plt.xlabel("Time t")
plt.ylabel("Expected value")
plt.legend(["Expected Susceptibles at time t", "Expected Infectives at time t"])
plt.show()
```

Question 2

a)

RTS

$$1)\ E[X_{t+1}|(X_t, Y_t) = (x_t, y_t)] = \alpha^{y_t} x_t$$
$$2)\ E[Y_{t+1}|(X_t, Y_t) = (x_t, y_t)] = x_t(1 - \alpha^{y_t})$$

In the Reed-Frost model. Starting with 1):

Firstly, we know that $X_t \sim Bin(X_{t-1}, \alpha^{Y_{t-1}})$. So, the expectation of X_t is given by:

$$E[X_t] = \alpha^{Y_{t-1}} X_{t-1}$$

And therefore,

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

Now let's consider 2):

As before, since we know that $Y_t = X_{t-1} - X_t$, we have:

$$E[Y_{t+1}|(X_t, Y_t) = (x_t, y_t)] = E[X_t - X_{t+1}|(X_t, Y_t) = (x_t, y_t)]$$

Then via the linearity of the expectation:

$$\begin{aligned} &= E[X_t|(X_t, Y_t) = (x_t, y_t)] - E[X_{t+1}|(X_t, Y_t) = (x_t, y_t)] \\ &= x_t - \alpha^{y_t} x_t \\ &= x_t(1 - \alpha^{y_t}) \end{aligned}$$

As required.

b)

Below is my reproduction of Figure 4.2 for $alpha = 0.98$, $x0 = 100$, $y0 = 1$. To create each the following graphs, I first simulated a Reed-Frost epidemic by generating values for $X_t \sim Bin(X_{t-1}, \alpha^{Y_{t-1}})$ and $Y_t = X_{t-1} - X_t$ for t in the range $[0,15]$. Then to compute the expected values of X_t and Y_t at a given time t, I computed $E[X_t|(X_{t-1}, Y_{t-1}) = (x_{t-1}, y_{t-1})] = \alpha^{y_{t-1}} x_{t-1}$ and $E[Y_t|(X_{t-1}, Y_{t-1}) = (x_{t-1}, y_{t-1})] = x_{t-1}(1 - \alpha^{y_{t-1}})$ respectively for all $t \in [1, 15]$.

Because I had to first simulate X_t and Y_t (which are random variables), the graphs of X_t and Y_t can take on a wide range of varying shapes. The graphs in Figure 4.2 are created from deterministic analogues of the Reed-Frost model and will look slightly different to the stochastic Reed-Frost model used here.

Also note that the below graphs are created from the same realization of the Reed-Frost chain.

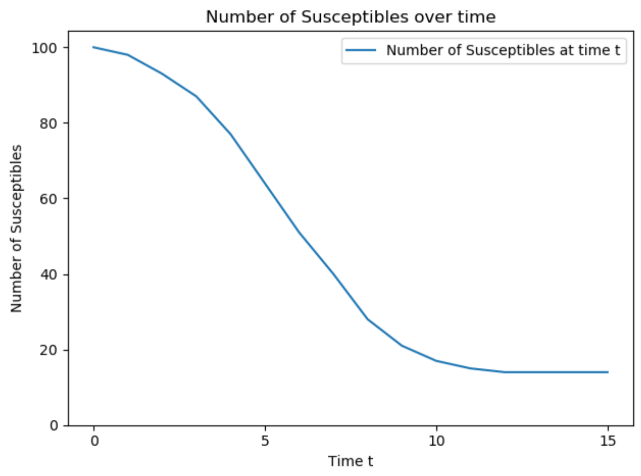


Figure 1: Number of Susceptibles Xt

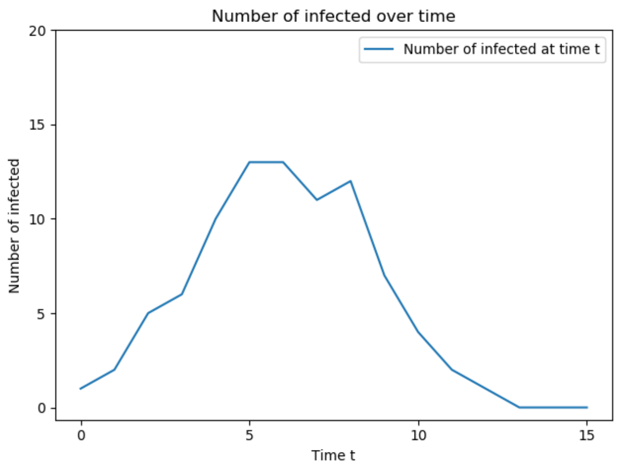


Figure 2: Number of Infectives Yt

Below is a plot of the trajectory of the expected values, jointly on the (X, Y) plane. Note that for each point on the plot represents a pair $(\mathbb{E}X_t, \mathbb{E}Y_t)$, on the X-Y plane.

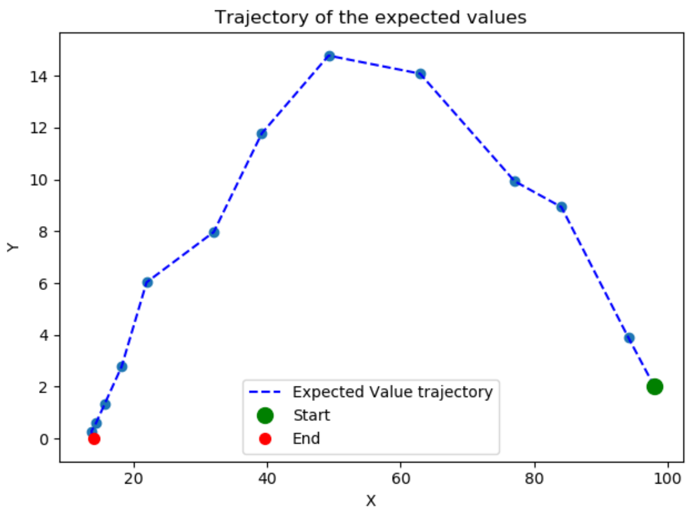


Figure 2: Trajectory of the expected values

The code used to generate these is below:

```
import numpy as np
from numpy.random import binomial as bin
import matplotlib.pyplot as plt

x0 = 100
y0 = 1
alpha = 0.98
t = 16

def expectation_X(xt, yt):
    return alpha ** yt * xt

def expectation_Y(xt, yt):
    return xt - alpha ** yt * xt

times = np.arange(0, t)
x = np.zeros(t)
y = np.zeros(t)
ex = np.zeros(t)
ey = np.zeros(t)
x[0] = x0
y[0] = y0
ex[0] = expectation_X(x[0], y[0])
ey[0] = expectation_Y(x[0], y[0])

for i in range(1, t):
    x[i] = bin(x[i-1], alpha ** y[i-1])
    y[i] = x[i - 1] - x[i]
    ex[i] = x[i] * alpha ** y[i]
    ey[i] = x[i] * (1 - alpha ** y[i])

def plot_xt():
    plt.figure(1)
    plt.plot(times, x)
    plt.title("Number of Susceptibles over time")
    plt.xlabel("Time t")
    plt.ylabel("Number of Susceptibles")
    xv = [0, 5, 10, 15]
    # create an index for each tick position
    xi = list(range(len(times)))
    yv = [0, 20, 40, 60, 80, 100]
    plt.xticks(xv, xv)
    plt.yticks(yv, yv)
    plt.legend(["Number of Susceptibles at time t"])
    plt.show()

def plot_yt():
    plt.figure(2)
    plt.plot(times, y, markevery=1)
    plt.title("Number of infected over time")
    plt.xlabel("Time t")
    plt.ylabel("Number of infected")
    xv = [0, 5, 10, 15]
    # create an index for each tick position
    xi = list(range(len(x)))
    yv = [0, 5, 10, 15, 20]
    yi = list(range(len(yv)))

    plt.xticks(xv, xv)
    plt.yticks(yv, yv)
    plt.legend(["Number of infected at time t"])
    plt.show()

def plot_xy():
    plt.figure(3)
    plt.plot(ex, ey, 'b--')
    plt.scatter(ex[1:-1], ey[1:-1])
    plt.plot(ex[0], ey[0], 'g.', markersize=20)
    plt.plot(ex[-1], ey[-1], 'r.', markersize=14)
    plt.title("Trajectory of the expected values")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.legend(["Expected Value trajectory", "Start", "End"])
    plt.show()


plot_xt()
plot_yt()
plot_xy()
```

Question 3

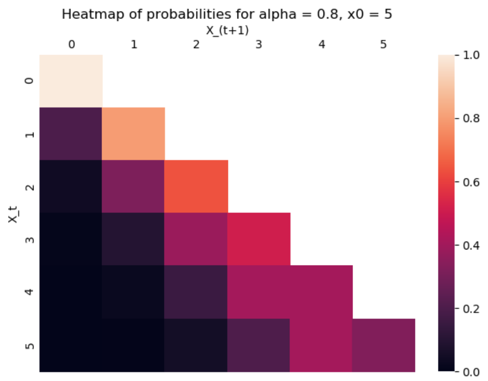
We know that the transition matrix P of the Greenwood model is lower triangular, of order $x_0 + 1$, and has the form:

$$\begin{array}{cc} & X_t \\ X_{t-1} & \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1-\alpha & \alpha & 0 & \dots & 0 \\ (1-\alpha)^2 & 2(1-\alpha)\alpha & \alpha^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (1-\alpha)^{x_0} & x_0(1-\alpha)^{x_0-1} & \binom{x_0}{2}(1-\alpha)^{x_0-2}\alpha^2 & \cdots & \alpha^{x_0} \end{pmatrix} \end{array}$$

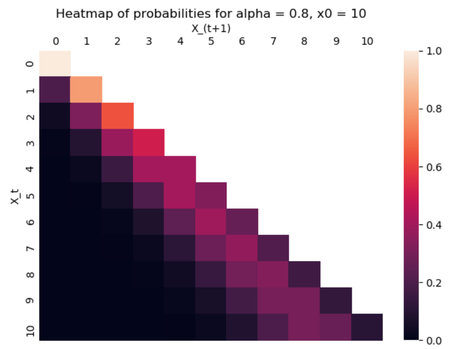
Where the state space is the possible values for $X_t \in [0, x_0]$. Each element P_{ij} represents the probability of going from state i to state j.

In the following heatmaps, I have set $\alpha = 0.8$, and the cells with zero probability are coloured white.

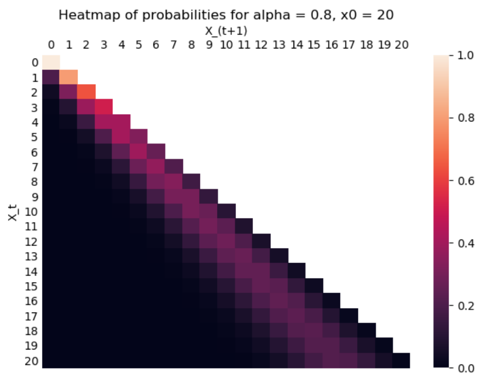
For $x_0 = 5$, the heatmap is:



For $x_0 = 10$:



And lastly for $x_0 = 20$:



The code used to generate these is below:

```
import numpy as np
from numpy.random import binomial as bin
from scipy.stats import binom
import matplotlib.pyplot as plt
import seaborn as sns

x0s = [5, 10, 20]
alpha = 0.8


def init_P(order):
    heatmap = np.zeros((order, order))
    for k in range(order):
        for j in range(k + 1):
            heatmap[k][j] = binom.pmf(j, k, alpha)
    return heatmap


for i, x0 in enumerate(x0s):
    plt.figure(i)
    order = x0 + 1
    mask = np.triu(np.ones((order, order)), k=1)
    heatmap = init_P(order)
    ax = sns.heatmap(heatmap, mask=mask)
    plt.title("Heatmap of probabilities for alpha = {}, x0 = {}".format(alpha, x0))
    plt.xlabel("X_(t+1)")
    plt.ylabel("X_t")
    ax.xaxis.tick_top()  # x axis on top
    ax.xaxis.set_label_position('top')
    ax.tick_params(length=0)
    plt.show()
```

b)

Consider the above heatmaps. In the Greenwood model, we see that there are $x_0 + 1$ communicating classes: $\{0\}, \{1\}, \dots, \{x_0\}$. This is because for any two states $i, j \in [0, x_0]$ such that $i > j$, $p_{ij} > 0$ but

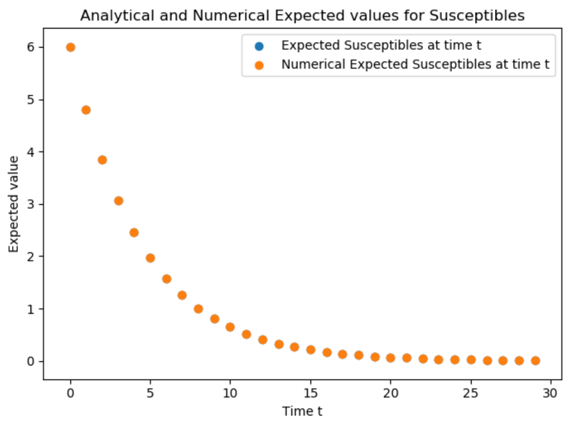
$p_{ji} = 0$ for $i > j$ (i.e. state j is accessible from state i but state i is not accessible from state j). But if $i = j$, then $p_{ii} > 0$. Therefore, each state is its own communicating class.

For transience, note that there are two cases we must consider: when the chain $\{X_t\}_{t>0}$ reaches 0, and when it doesn't. If the chain $\{X_t\}_{t>0}$ hits 0, then it will never leave the state 0 because $p_{0,0} = 1$ and hence $P_0(T_0 < \infty) = 1$ (The probability of returning to state 0). Consequently, every state except for 0 would be transient since $P_i(T_i = \infty) \geq p_{ij} > 0\ for\ i > j,\ i \neq 0$. Therefore in this situation, there are x_0 transient states $\{1\}, \{2\}, \dots, \{x_0\}$ with $\{0\}$ being the recurrent state.

The case when the chain $\{X_t\}_{t>0}$ never reaches 0 occurs when $X_{t-1} = X_t = x_t$, in which case the state x_t becomes a recurrent state, because the number of infectives is $Y_t = 0$ and so no more susceptibles can possibly become infected, i.e. X_t will not fall any lower. This means that any state $X_t = x_t \in [0, x_0]$ can become a recurrent state so long as $X_{t-1} = X_t = x_t$.

c)

Below is a plot of the numerically calculated Expected number of Susceptibles $\mathbb{E}X_t$ and the analytically calculated expected values found in $1b$. We can see the value of $\mathbb{E}X_t$ at time t for both methods is exactly the same (since the scatter poins are overlapping each other):



This method works because the expression $e_{x_0+1}^T P^t v$ uses, through matrix multiplication, to apply the age-old formula the formula $\mathbb{E}[X_t|X_0 = x_0] = \sum_x x \cdot P(X_t = x \mid X_0 = x_0)$ to calculate the expected value of X_t given that $X_0 = x_0$. By multiplying the $t - step$ transition matrix, P^t, with the vector $e = [0, \dots, 1]^T$, we are creating a vector in which each element i represents the probability of going from $X_0 = x_o$ to $X_t = i$. I.e.:

$$\begin{aligned} e_{x_0+1}^T P^t &= (0 \quad \dots \quad 0 \quad 1) P^{(t)} \\ &= \begin{pmatrix} p_{x_0 0}^{(t)} & p_{x_0 1}^{(t)} & \dots & p_{x_0 x_0}^{(t)} \end{pmatrix} \end{aligned}$$

Where $p_{x_0,0}^{(t)}$ is the probability of going from state x_0 to state 0 in t steps. Then, by multiplying this by the vector $v = [0, 1, \dots, x_0]^T$, we attain the summation $\mathbb{E}[X_t|X_0 = x_0] = \sum_x x \cdot P(X_t = x \mid X_0 = x_0)$.

Consider the case in which we want to find the expected value of X_t at time $t = 1$, given that we start at x_0. The probability of this using the numerical formula is:

$$\begin{aligned}
e_{x_0+1}^T P^1 v &= \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} (P) \begin{pmatrix} 0 \\ \vdots \\ x_0 \end{pmatrix} \\
&= \begin{pmatrix} p_{x_0 0} & p_{x_0 1} & \dots & p_{x_0 x_0} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ x_0 \end{pmatrix} \\
&= \sum_{x=0}^{x_0} x \cdot P(X_1 = x \mid X_0 = x_0) \\
&= \mathbb{E}[X_1 \mid X_0 = x_0]
\end{aligned}$$

Similarly, with $t = 2$: (Note that $p_{x_0 0}^{(2)}$ denotes the probability of going from x_0 to 0 in 2 steps)

$$\begin{aligned}
e_{x_0+1}^T P^2 v &= \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} \left(P^{(2)}\right) \begin{pmatrix} 0 \\ \vdots \\ x_0 \end{pmatrix} \\
&= \begin{pmatrix} p_{x_0 0}^{(2)} & p_{x_0 1}^{(2)} & \dots & p_{x_0 x_0}^{(2)} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ x_0 \end{pmatrix} \\
&= \sum_{x=0}^{x_0} x \cdot P(X_2 = x \mid X_0 = x_0) \\
&= \mathbb{E}[X_2 \mid X_0 = x_0]
\end{aligned}$$

In this manner, we can see that $e_{x_0+1}^T P^t v = \mathbb{E}[X_t \mid X_0 = x_0]$.

The code used to make the graph of the numerical expectation is below:

```
import numpy as np
from scipy.stats import binom
import matplotlib.pyplot as plt

def init_P(order):
    heatmap = np.zeros((order, order))
    for k in range(order):
        for j in range(k + 1):
            heatmap[k][j] = binom.pmf(j, k, alpha)
    return heatmap


def expectation_X(t):
    return x0 * alpha ** t
```

```
alpha = 0.8
x0 = 6
order = x0 + 1
t = 30
P = init_P(order)

times = np.arange(0, t)
eX = np.vectorize(expectation_X)
expected_x = eX(times)

numerical_ex = np.zeros(t)
p = np.eye(order)
e = np.zeros(order)
e[-1] = 1
v = np.transpose(np.arange(0, order))

for i in range(0, t):
   numerical_ex[i] = np.dot(np.dot(e, p), v)
   p = np.dot(p, P)

plt.scatter(times, expected_x)
plt.scatter(times, numerical_ex)
plt.title("Analytical and Numerical Expected values for Susceptibles")
plt.xlabel("Time t")
plt.ylabel("Expected value")
plt.legend(["Expected Susceptibles at time t", "Numerical Expected Susceptibles at time t"])
plt.show()
```

---

d)

Recall that via the tower property for the conditional expectation, we obtain:

$$\mathbb{E}[X_t|X_0 = x_0] = \mathbb{E}[\mathbb{E}[X_t|X_{t-1}]|X_0 = x_0]$$

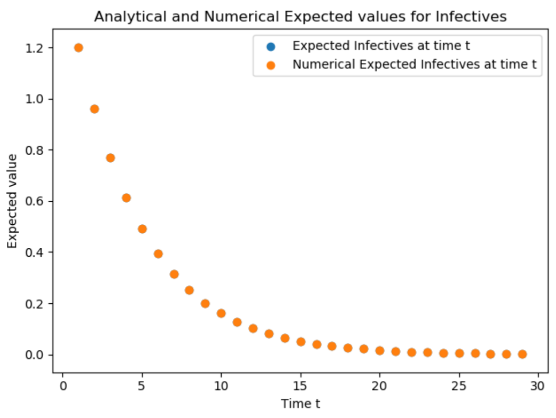
And since $\mathbb{E}[X_t|X_{t-1}] = \alpha X_{t-1}$ as well as the linearity of the expectation:

$$\begin{aligned} &= \mathbb{E}[\alpha X_{t-1}|X_0 = x_0] \\ &= \alpha\mathbb{E}[X_{t-1}|X_0 = x_0] \end{aligned}$$

So, we have:

$$\begin{aligned} \mathbb{E}[Y_t|X_0 = x_0] &= \mathbb{E}[X_{t-1} - X_t|X_0 = x_0] \\ &= \mathbb{E}[X_{t-1}|X_0 = x_0] - \mathbb{E}[X_t|X_0 = x_0] \\ &= \mathbb{E}[X_{t-1}|X_0 = x_0] - \alpha\mathbb{E}[X_{t-1}|X_0 = x_0] \\ &= (1-\alpha)\mathbb{E}[X_{t-1}|X_0 = x_0] \end{aligned}$$

So, to numerically compute the expectation of Y_t, we just numerically compute $\mathbb{E}[X_{t-1}|X_0 = x_0]$ with $e_{x_0+1}^T P^{t-1} v$ and return $Y_t = (1-\alpha)\mathbb{E}[X_{t-1}|X_0 = x_0]$. A graph of the analytical and numerical expected values for Y_t given $x_0 = 6$ and with $\alpha = 0.8$ is below as well as the code used to make it. Clearly, both the numerical and analytical values agree on the expected values (as the two sets of points are overlapping each other on the scatterplot):



```
import numpy as np
from scipy.stats import binom
import matplotlib.pyplot as plt

def init_P(order):
    heatmap = np.zeros((order, order))
    for k in range(order):
        for j in range(k + 1):
            heatmap[k][j] = binom.pmf(j, k, alpha)
    return heatmap

def expectation_X(t):
    return x0 * alpha ** t

def expectation_Y(t):
    return x0 * alpha ** (t-1) - x0 * alpha ** t

alpha = 0.8
x0 = 6
order = x0 + 1
t = 30
P = init_P(order)

times = np.arange(0, t)
eY = np.vectorize(expectation_Y)
expected_y = eY(times)
eX = np.vectorize(expectation_X)
expected_x = eX(times)

numerical_ex = np.zeros(t)
numerical_ey = np.zeros(t)
p = np.eye(order)
e = np.zeros(order)
e[-1] = 1
v = np.transpose(np.arange(0, order))

numerical_ex[0] = np.dot(np.dot(e, p), v)
p = np.dot(p, P)
for i in range(1, t):
    numerical_ex[i] = np.dot(np.dot(e, p), v)
    numerical_ey[i] = (1-alpha) * numerical_ex[i-1]
    p = np.dot(p, P)
```

```
plt.scatter(times[1:], expected_y[1:])
plt.scatter(times[1:], numerical_ey[1:])
plt.title("Analytical and Numerical Expected values for Infectives")
plt.xlabel("Time t")
plt.ylabel("Expected value")
plt.legend(["Expected Infectives at time t", "Numerical Expected Infectives at time t"])
plt.show()
```

Question 4

For this question, I have used $\alpha = 0.8$ and $x_0 = 6$.

a)

First consider the one-step transition matrix for the number of susceptibles X_t (in the Greenwood model):

$$\begin{array}{cc} & X_t \\ X_{t-1} & \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1-\alpha & \alpha & 0 & \dots & 0 \\ (1-\alpha)^2 & 2(1-\alpha)\alpha & \alpha^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (1-\alpha)^{x_0} & x_0(1-\alpha)^{x_0-1} & \binom{x_0}{2}(1-\alpha)^{x_0-2}\alpha^2 & \cdots & \alpha^{x_0} \end{pmatrix} \end{array}$$

Note that the number of susceptibles X_t is always non-increasing and if for some time step t, $X_t = X_{t-1}$ then the number of infectives must be 0, i.e. $Y_t = 0$. So, for there to be at least one infected person at time t, then we must have it that $X_t < X_{t-1} < \cdots < X_0$. This implies that if $X_t = j$ and $Y_t > 0$, then $X_{t-1} \in [j+1,\ x_0 - (t-1)]$. The upper limit for X_{t-1} is $x_0 - (t-1)$ because we are starting from $X_0 = x_0$ susceptibles, and the number of susceptibles X must decrease by at least 1 person each time step to maintain the inequality $X_t < X_{t-1}$. The lower limit for X_{t-1} is $j+1$ because X_{t-1} has to be at minimum 1 larger than $X_t = j$.

Now consider Equation 4.1.6, which describes that probability of having j susceptibles and at least 1 infected person at time step t:

$$p_j^t \equiv \mathrm{P}(X_t = j, Y_t > 0) = \sum_{i=j+1}^{x_0-(t-1)} p_i^{t-1} p_{ij}$$

Where $p_j^t = P(X_t = j, Y_t > 0)$ and $p_{ij} = P(X_{t+1} = j \mid X_t = i, Y_t > 0)$. The equation can also be expressed as:

$$\mathrm{P}(X_t = j, Y_t > 0) = \sum_{i=j+1}^{x_0-(t-1)} P(X_{t-1} = i, Y_{t-1} > 0)\, P(X_t = j \mid X_{t-1} = i)$$

This summation is just an application of the law of total probability, in which we find the probability of having $X_t = j$ with $Y_t > 0$ by summing the probabilities of reaching $X_t = j$ and $Y_t > 0$ from all possible, previous states X_{t-1}. As explained previously, to have at least 1 infected person $(Y_t > 0)$, and have $X_t = j$, then X_{t-1} must take values in the range $[j+1, x_0 - (t-1)]$ and so we must sum from $i = j+1$ to $i = x_0 - (t-1)$ in equation 4.1.6.

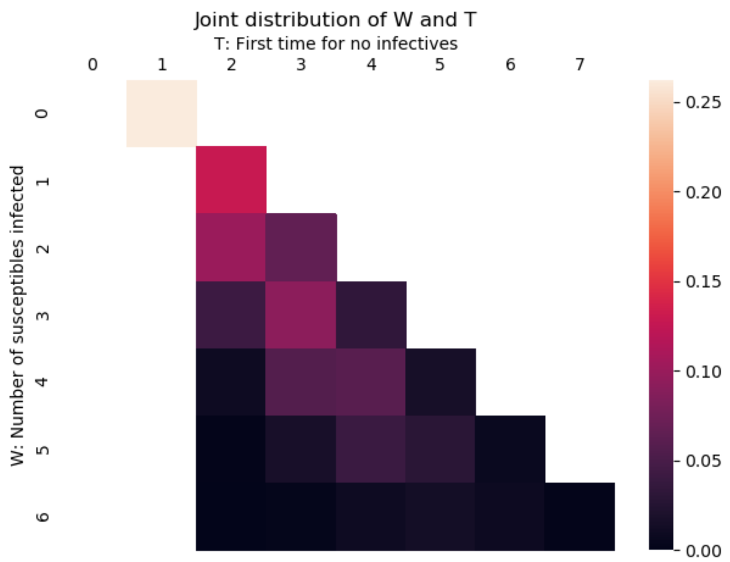
Inside the summation, we have the recursive term $p_i^{t-1} = P(X_{t-1} = i, Y_{t-1} > 0)$, the probability of having i susceptibles at time $t-1$, with at least 1 infected person.

We also have $p_{ij} = P(X_t = j \mid X_{t-1} = i)$, the probability of dropping to $X_t = j$ susceptibles given that $X_{t-1} = i$. Because $X_t = j < i = X_{t-1}$, we know that $Y_t = X_{t-1} - X_t > 0$, so this probability can simply be drawn from the one-step transition matrix (i.e. p_{ij}).

b)

To calculate $P(W > 4)$, I first calculated the joint distribution of W and T using the recursive relationship $\Gamma(W = k, T = n | x_0) = p_{x_0-k}^{n-1}\alpha^{x_0-k}$Then I found $P(W > 4)$ with $P(W > 4) = 1 - \sum_{k=0}^{4}\sum_{n=1}^{7}\Gamma(k, n|x_0)$. Note that here $T \in \{1, 2, .., 7\}$ but in the heatmap below, the graphing software automatically inserted a value for $T = 0$. The max value for T is 7 because it takes at most 7 time steps for both X_t and Y_t to reach 0 for $x_0 = 6$. I.e. the longest path to 0 for X_t and Y_t would be $X_0 = 6, X_1 = 5, X_2 = 4, \ldots, X_5 = 1, X_6 = 0\ (Y_6 = 1), X_7 = 0\ (Y_7 = 0)$

With $x0 = 6$ and $\alpha = 0.8$, the probability was found to be: $\mathbb{P}(W > 4) = 0.130298\ (6.d.p)$. A heatmap of the joint distribution is below.



The code used to generate this is on the following page.

```
import numpy as np
from scipy.stats import binom
import matplotlib.pyplot as plt
import seaborn as sns

np.set_printoptions(precision=2)
x0 = 6
# Largest num of time steps to have Y_t = 0 for x0=6 is 7. But add 1 for indexing and stuff.
# For x0=6 its 0,1,2,...,7
# ex: X0=6, X1=5 x2=4 x3=3 x4=2 x5=1 x6=0 (Y6=1) X7=0 (Y7=0).
max_time = x0 + 2
order = x0 + 1
alpha = 0.8

# One step transition matrix
one_step = np.zeros((order, order))
for k in range(order):
    for j in range(k + 1):
        one_step[k][j] = binom.pmf(j, k, alpha)

# Matrix of probabilities of having Xt=j and Yt>0 at time step t.
probs = np.zeros((max_time, order))
probs[0][x0] = 1  # start with x0=6 at time 0. so P(X0=6)=1

for t in range(1, max_time):
    for j in range(0, x0 + 1):
        for i in range(j + 1, x0 - t + 2):
            probs[t][j] += probs[t - 1, i] * one_step[i][j]

# Joint distribution of W and T
joint_WT = np.zeros((order, max_time))
for k in range(0, x0 + 1):
    for n in range(1, max_time):
        joint_WT[k][n] = probs[n-1][x0-k] * alpha ** (x0 - k)

# Calculating the probability that W > 4.
p_w_gt_4 = 0
for w in (5, 6):
    for t in range(1, max_time):
        p_w_gt_4 += joint_WT[w][t]
print("P(W>4) =", p_w_gt_4)

# Optional heatmap
ax = sns.heatmap(joint_WT, mask=joint_WT == 0)
plt.title("Joint distribution of W and T".format(alpha, x0))
plt.xlabel("T: First time for no infectives")
plt.ylabel("W: Number of susceptibles infected")
ax.xaxis.tick_top()  # x axis on top
ax.xaxis.set_label_position('top')
ax.tick_params(length=0)
plt.show()
```

c)

In the Monte Carlo simulation, I simulated 10^6 trajectories of X_t and Y_t and with these got 10^6 values for T and W. With these I calculated the joint distribution of T and W, where

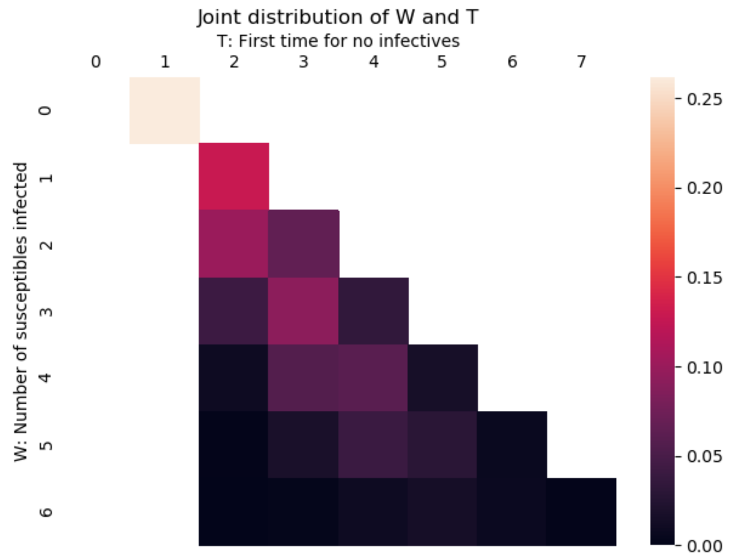
$$P(W = w, T = t) = \frac{Number\ of\ occurrences\ where\ W = w\ and\ T = T}{10^6}$$

Then to calculate that probability that $P(W > 4)$, I simply summed over all the cells in this matrix for which $W > 4$. This probability was found to be:

$$P(W > 4) = 1 - \sum_{k=0}^{4}\sum_{n=1}^{7}\Gamma(k, n|x_0\ )\ 0.130453\ (6.d.p)$$

Which is similar to the result found in part b). A heatmap of the joint distribution of W and T calculated via Monte Carlo simulations is below, with $x_0 = 6$ and $\alpha = 0.8$.

And a similar heatmap:



The code used to generate this is below:

```
import numpy as np
from numpy.random import binomial as bin
from scipy.stats import binom
import matplotlib.pyplot as plt
import seaborn as sns

np.set_printoptions(precision=2)
```

```
x0 = 6
max_time = x0 + 2
order = x0 + 1
alpha = 0.8
N = 10 ** 6

# Joint distribution of W and T
joint_WT = np.zeros((order, max_time))

for _ in range(N):
    Xt = x0
    k = 0
    n = 0

    while True:
        n += 1
        k = x0 - Xt
        Xnext = bin(Xt, alpha)
        if Xnext == Xt:
            break
        Xt = Xnext

    joint_WT[k][n] += 1

joint_WT = joint_WT / N


# Calculating the probability.
p_w_gt_4 = 0
for w in (5, 6):
    for t in range(1, max_time):
        p_w_gt_4 += joint_WT[w][t]
print("P(W>4) =", p_w_gt_4)


# Optional heatmap
ax = sns.heatmap(joint_WT, mask=joint_WT == 0)
plt.title("Joint distribution of W and T".format(alpha, x0))
plt.xlabel("T: First time for no infectives")
plt.ylabel("W: Number of susceptibles infected")
ax.xaxis.tick_top()  # x axis on top
ax.xaxis.set_label_position('top')
ax.tick_params(length=0)
plt.show()
```

d)

To calculate $P(W > 4)$ with the PGF method, I made use of the formula on page 111 of EM-4 that described the marginal pgf of W:

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

Where $\bar{P}(\phi)$ is:

$$\begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1-\alpha & 0 & 0 & \dots & 0 & 0 \\ (1-\alpha)^2 & 2(1-\alpha)\alpha & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & \vdots \\ (1-\alpha)^{x_0}\phi^{x_0} & x_0(1-\alpha)^{x_0-1}\phi^{x_0-1} & \binom{x_0}{2}(1-\alpha)^{x_0-2}\alpha^2\phi^{x_0-2} & \cdots & \binom{x_0}{1}(1-\alpha)\alpha^{x_0-1}\phi & 0 \end{pmatrix}$$

As per equation 4.1.11 (pg 110). The other terms are Q= $(1, \alpha, \dots, \alpha^{x_0})$, $A' = (0,0,\dots,0,1)$, and $E = (1,1,\dots,1)$ (pg 109).

Once I had the pgf of W, I could use its property that $P(W = k) = \frac{\Psi^k(\phi)}{k!}$, where Ψ^k refers to the k^{th} derivative of $\Psi(\phi)$. Therefore, with $x0 = 6$ and $\alpha = 0.8$, the probability that W s greater than 4 is,

$$\begin{aligned} P(W > 4) &= P(W = 5) + P(W = 6) \\ &= \frac{\Psi^5(\phi)}{5!} + \frac{\Psi^6(\phi)}{6!} \\ &= 0.130298\ (6\ d.p) \end{aligned}$$

Note that in this question, I have used MATLAB to calculate the probabilities. This is because MATLAB has a nice inbuilt symbolic library that can compute derivatives for you. The MATLAB code to compute this is below:

```
x0 = 6;
order = x0 + 1;
alpha = 0.8;
syms phi;
I = eye(order);
P = sym(zeros(order, order));
A = zeros(1,order);
A(end) = 1;
Q = zeros(1,order);


for k = 0:x0
   Q(k+1) = alpha ^ k;
end
Q = Q.';
```

```
for k = 0:order-1
   for j = 0:k-1
      P(k+1,j+1) = binopdf(j, k, alpha) * phi ^(k-j);
   end
end
sum = sym(zeros(order, order));

for k = 0:order-1
   for j = 0:k-1
      P(k+1,j+1) = P(k+1,j+1);
   end
end
marginal_w = A*inv(I-P)* Q
prob_gt_4 = 0;
for p=1:6
   marginal_w = diff(marginal_w);
   subs(marginal_w, 0) / factorial(p)
   if p > 4
      prob_gt_4 = prob_gt_4 + subs(marginal_w, 0) / factorial(p);
   end
end
prob_gt_4 % Probability that W is greater than 4.
```

Question 5

a)

The transition probability matrix in $(4.2.2)$ is:

$$P = \begin{pmatrix} P_{00} & 0 & \cdots & 0 \\ P_{10} & P_{11} & \cdots & P_{1x_0} \\ \vdots & \vdots & \ddots & \vdots \\ P_{x_0 0} & P_{x_0 1} & \cdots & P_{x_0 x_0} \end{pmatrix}$$

Where each submatrix P_{ij} is another transition probability matrix, representing the probabilities of going from $(X_t, Y_t = i)$ to $(X_{t+1}, Y_{t+1} = j)$.

Therefore, each element in P (assume all submatrices are combined into a large 2d matrix) describes the probability of transitioning from one pair (X_t, Y_t) to (X_{t+1}, Y_{t+1}). In particular, the probability of going from (X_t, Y_t) to (X_{t+1}, Y_{t+1}) is the element at position $(Y_t \cdot x_0 + X_t, Y_{t+1} \cdot x_0 + X_{t+1})$ in the matrix P.

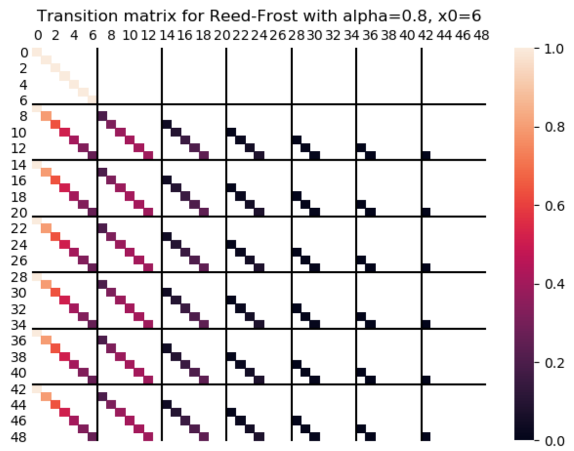
Therefore, the state space must be: $\{(x, y) \mid x, y \in [0, x_0]\}$. However, note that a state (X_{t+1}, Y_{t+1}) can only be reached by the state (X_t, Y_t) if and only if $X_t = X_{t+1} + Y_{t+1}$.

b)

The only communicating classes in this Markov chain are $\{(0,0)\}, \{(1,0)\} \dots, \{(x_0, 0)\}, i.e. (x, 0)\ \forall x \in [0, x_0]$. This is because once the number of infectives Y_t is 0, the infection in the population ceases and no more susceptibles can be infected. Thus, probability of going from $(x, 0)$ at time t to $(x, 0)$ at time t+1 is 1, i.e. $P((x, 0)_t, (x, 0)_{t+1}) = 1\ \forall x \in [0, x_0]$. The are no other communicating classes in this model.

c)

Below is a heatmap of the transition probability matrix for the Reed-Frost model with $x0 = 6$ and $\alpha = 0.8$. The cells with 0 probabilities are in white, and black gridlines split the heatmap into the $(x_0 + 1)^2$ submatrices P_{00}, P_{01}, …. etc.



The code used to generate this is below:

```
import numpy as np
from scipy.stats import binom
import matplotlib.pyplot as plt
import seaborn as sns
from numpy.linalg import matrix_power

x0 = 2
alpha = 0.8
y0 = 1

def init_P():
    p = np.zeros(((x0 + 1), (x0 + 1), (x0 + 1), (x0 + 1)))
    p[0][0] = np.eye(x0+1)
    for i in range(1, x0+1):
        for j in range(x0+1):
            for xt in range(x0+1):
                for xtnext in range(x0+1):
                    if xtnext == (xt-j):
                        p[i][j][xt][xtnext] = binom.pmf(xtnext, xt, alpha ** i)
                        break
    return p

def p_to_2d(p):
    p2d = np.zeros(((x0 + 1) ** 2, (x0 + 1) ** 2))
    annots = np.zeros(((x0 + 1) ** 2, (x0 + 1) ** 2), dtype=object)

    for i in range((x0 + 1) ** 2):
        for j in range((x0 + 1) ** 2):
            p2d[i][j] = p[int(i/(x0+1))][int(j/(x0+1))][i % (x0 + 1)][j % (x0 + 1)]
            annots[i][j] = f"{int(i % (x0 + 1))},{int(i/(x0+1))}-{j % (x0 + 1)},{int(j/(x0+1))}"
    return p2d, annots
p = init_P()

# Optional heatmap
p_2d, annots = p_to_2d(p)
print(p_2d)
plt.figure(1)
# ax = sns.heatmap(p_2d, annot=annots.tolist(), annot_kws={"size": 10}, fmt='', mask=p_2d == 0)
ax = sns.heatmap(p_2d, mask=p_2d == 0)
plt.title(f"Transition matrix for Reed-Frost with alpha={alpha}, x0={x0}")
# plt.xlabel("T: First time for no infectives")
# plt.ylabel("W: Number of susceptibles infected")
print(*ax.get_xlim())
ax.hlines([(x0+1)*k for k in range(1,x0 + 1)], *ax.get_xlim())
ax.vlines([(x0+1)*k for k in range(1,x0 + 1)], *ax.get_ylim())
ax.xaxis.tick_top()  # x axis on top
ax.xaxis.set_label_position('top')
ax.tick_params(length=0)
plt.show()
```

d)

Using Monte Carlo simulation, it was estimated that the probability $P(W > 4) \approx 0.256263$. This estimation is approximately 96.4% larger than that found using the Greenwood model. This is because in the Reed-Frost model, the probability of no-infection decreases as the number of infectives increases – i.e. the more infectives there are, the greater the probability of a susceptible getting infected. In the Greenwood model, the probability of no infection is constant, and therefore people are more likely to be infected in the Reed-Frost model than in the Greenwood.

Since W is the total number of susceptibles infected, then you are more likely to see a greater number of infected susceptibles, and therefore, a larger value of $P(W > 4)$. A heatmap of the joint distribution of W and T was also estimated and is below.

The code for the Monte Carlo simulation is below:

```
import numpy as np
from numpy.random import binomial as bin
import matplotlib.pyplot as plt
import seaborn as sns

np.set_printoptions(precision=2)
x0 = 6
y0 = 1
max_time = x0 + 2
order = x0 + 1
alpha = 0.8
N = 10 ** 6
```

```
# Joint distribution of W and T
joint_WT = np.zeros((order, max_time))

for _ in range(N):
    Xt = x0
    Yt = y0
    k = 0
    n = 0

    while True:
        n += 1
        k = x0 - Xt
        Xnext = bin(Xt, alpha ** Yt)
        Yt = Xt-Xnext
        if Xnext == Xt:
            break
        Xt = Xnext

    joint_WT[k][n] += 1

joint_WT = joint_WT / N

# Calculating the probability.
p_w_gt_4 = 0
for w in (5, 6):
    for t in range(1, max_time):
        p_w_gt_4 += joint_WT[w][t]
print("P(W>4) =", p_w_gt_4)

# Optional heatmap
ax = sns.heatmap(joint_WT, mask=joint_WT == 0)
plt.title("Joint distribution of W and T".format(alpha, x0))
plt.xlabel("T: First time for no infectives")
plt.ylabel("W: Number of susceptibles infected")
ax.xaxis.tick_top()  # x axis on top
ax.xaxis.set_label_position('top')
ax.tick_params(length=0)
plt.show()
```

Question 6

Task

In this task, we are to investigate the expected daily rate of infection of COVID-19 amongst visitors quarantined in a motel. A Reed-Frost model will be employed to model this infection for population sizes ranging from 1 to 10 and it will be used in analytical and Monte Carlo methods to calculate the long-term expected rate of infections per day coming out of the motel. This motel conducts daily COVID-19 tests and those who test positive are immediately moved to another facility. The results of each test are delayed by one day, meaning that infections may persist in the population despite removing the infected.

We are given that the probability of new arrivals being infected by COVID-19 is $\eta = 0.05$, the probability of contact between an infective and susceptible is $p = 0.1$, and that the probability of this contact resulting in an infection is $\beta = 0.05$. This yields a probability of no infection due to a single infective being $\alpha = 1 - p\beta = 0.995$. With this information, we note that the number of infected individuals y_0 out of the x_{-1} new arrivals, is distributed binomially: $y_0 \sim Bin(x_{-1}, \eta = 0.05)$. Note that I have altered the terminology used in the task – the task says we are "to accept x_0 new individuals" per batch. Let us re-denote x_0 by x_{-1}. Then, in the following time step, we obtain the initial values x_0 and y_0 used to initiate the Reed-Frost model.

Several key assumptions were used to model this task.

Assumptions of the Reed Frost Model

1. A new batch of x_{-1} infectives is brought in on the same day the previous batch of people are deemed free of infection. i.e. If $X_t = x \ and \ Y_t = 0$ at time t $\forall x \in [0, x_{-1}]$, then a new batch of x_{-1} are brought in on that same day (time t), such that $X_{t+1} = x_{-1} - y_0$ and $X_{t+1} = y_0, where \ y_0 \sim Bin(x_{-1}, \eta)$.
2. Assume that COVID-19 spreads only by some form of contact with an infected individual, with probability $\beta = 0.05$.
3. Assume that the population of susceptibles and infectives are will mixed, homogenous, and have no immunity to the disease.
4. There are no births or deaths in the motel during the infection's outbreak.

The Model

Typically, in a Reed-Frost model, the infection ceases as soon as the number of infected Y_t drops to 0 since the epidemic Markov Chain enters some recurrent state $(X_t, Y_t) = (x, 0)$ $\forall x \in [0, x_0]$ and remains in this state for all time steps $t + n, n \in \mathbb{N}$. In this task however, as soon as the infection ends in the motel, a new batch of x_{-1} arrivals are brought into the motel on the same day, tested, and the infection begins anew on the following day. This necessitated a modified reed-frost transition matrix that is described in Appendix A, along with an example for $x_{-1} = 3$. The main changes were that:

If $X_t + Y_t > x_{-1}$ for some state (X_t, Y_t), then the probability of transitioning to or from it was set to 0 (as accessing this state in a simulation is impossible), and instead it would always return to itself $\mathrm{P}\big((X_t, Y_t), (X_t, Y_t)\big) = 1$.

And, if a state $(X_t, Y_t) = (x, 0), x \in [0, x_{-1}]$, then a new set of arrivals are brought at time t, such that the number of infectives in the next time step is given by $y_0 \sim Bin(x_{-1}, \eta = 0.05)$, and the number of susceptibles is given by $x_0 = x_{-1} - y_0$. I.e. $P\big((x, 0), (x_0, y_0)\big) = \binom{x_{-1}}{y_0} 0.05^{y_0}(1 - 0.05)^{x_{-1} - y_0}$ if and only if $x_0 + y_0 = x_{-1}$.

Results

Daily expected infectives and Daily infection rate estimation was conducted through two different methods: Monte Carlo Simulations, and via an analytical approach.

Analytical Approach

This method involved calculating the stationary distribution π of the transition matrix described in the previous section, and then using this to calculate the expected number of infectives per day, $\mathbb{E}Y_t$. The formula to calculate this is below:

$$\mathbb{E}Y_t = \sum_{(X_t, Y_t)} \sum_{(X_{t+1}, Y_{t+1})} \pi(X_t, Y_t) \cdot P_{(X_t, Y_t), (X_{t+1}, Y_{t+1})} \cdot Y_{t+1}$$

Where $\pi(X_t, Y_t)$ is the long-term probability (from the stationary distribution) of being in state (X_t, Y_t), $P_{(X_t, Y_t), (X_{t+1}, Y_{t+1})}$ is the probability of transitioning from $(X_t, Y_t)\ to\ (X_{t+1}, Y_{t+1})$, and Y_{t+1} is the number of infectives at time $t + 1$. I was unable to calculate the daily *rate* of infection however, as I couldn't calculate the expected length of the infection - instead, this value was calculated via the Monte Carlo simulation. A graph of the expected number of infectives $\mathbb{E}Y_t$ versus the initial population size $x_{-1} \in [1,10]$ for the analytical and Monte Carlo methods is seen in $Figure\ 3$. Note that all values here are the same as those calculated by the Monte Carlo simulation, and so the values are overlapping each other.

Monte Carlo simulations

This method involved simulating $N = 10$ million Reed-Frost-modelled epidemics, in which each epidemic ended as soon as $Y_t = 0$.

The expected number of infectives per day was then estimated with

$$\mathbb{E}Y_t \approx \frac{\sum_{j=1}^{N} \sum_{i}^{k_j} Y_{j,i}}{\sum_{j=1}^{N} k_j} = \frac{total\ number\ of\ infectives\ across\ N\ simulations}{sum\ of\ all\ epidemic\ lengths.}$$

Where k_j is the length of epidemic j (including the last day when $Y_{k_j} = 0$), $Y_{j,i}$ is the number of infectives in epidemic simulation j at time i, and N is the total number of simulations.

A graph of the expected number of infectives $\mathbb{E}Y_t$ versus the initial population size $x_{-1} \in [1,10]$ for the analytical and Monte Carlo methods is seen to the right. There is an almost linear relationship between the initial population size and expected number of infectives

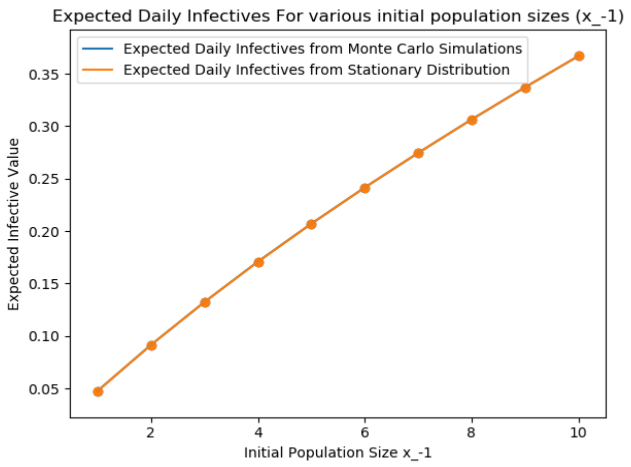


Figure 3: Expected daily infectives

Additionally, the long-term rate of infection was calculated with:

$$rate = \frac{\frac{\sum_{j=1}^{N}\sum_{i}^{k_j} Y_{j,i}}{N}}{x_{-1}} = \frac{average\ number\ of\ infectives\ per\ simulation}{total\ number\ of\ arrivals\ per\ simulation}$$

A graph of this rate versus the initial population size x_{-1} is in $Figure\ 4$ below.

Observe that although the rate of infection starts out at 0.05, it increases linearly with the initial population size such that the long-term expected rate of infection for an initial population size of $x_{-1} = 10$ people is 0.0525, 50% larger than the rate of infection in new arrivals. Thus, we can confirm that the rate of infections per person with this motel is larger than that of the new arrivals without this motel.

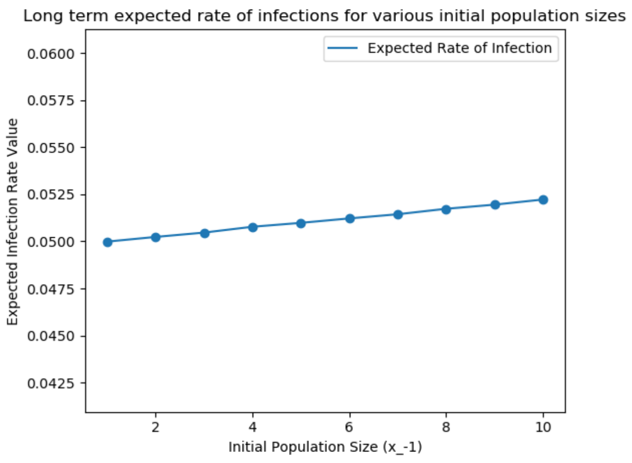


Figure 4: Expected rate of infection

There were several unrealistic aspects in this analysis, with the foremost, the assumption that the populations are homogeneous, well mixed and have no innate immunity. People have varying degrees of immunity, and many are more susceptible to infection than others. Further, the arrivals in this motel would not all be placed in one room/be free to mingle – they would be socially isolated in quarantine. Additionally, new arrivals wouldn't be brought in on the same day the previous susceptibles are moved out. There needs to be at least a day for cleaning & disinfecting the rooms as well as induction for new arrivals.

The source code for to generate the two above graphs is below:

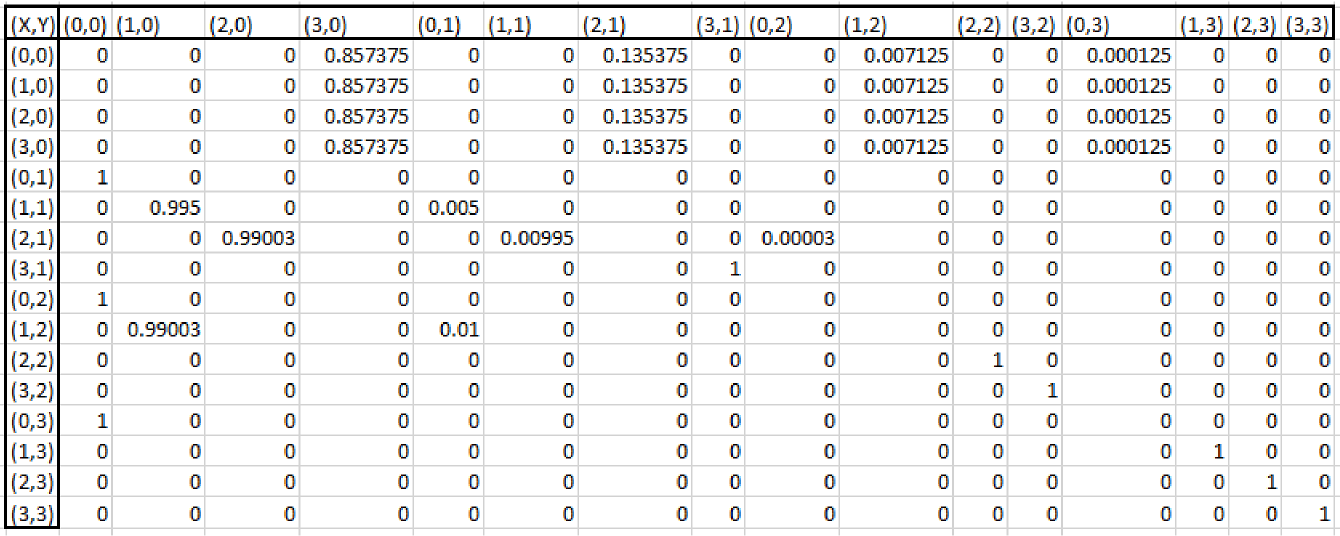
```
import numpy as np
from numpy.random import binomial as bin
from scipy.stats import binom
import matplotlib.pyplot as plt
from numpy.linalg import matrix_power

np.set_printoptions(precision=5, threshold=np.inf, linewidth=np.inf, suppress=True)
eta = 0.05
RUNS = 150
alpha = 0.995


def init_P(x__1):
    order_ = x__1 + 1
    p = np.zeros((order_, order_, order_, order_))
    for yt in range(order_):
        for ytnext in range(order_):
            for xt in range(order_):
                for xtnext in range(order_):
                    if yt == 0:
                        if xtnext + ytnext == x__1:
                            p[yt][ytnext][xt][xtnext] = binom.pmf(ytnext, x__1, eta)
                    elif xt == xtnext + ytnext and xt + yt <= x__1:
                        p[yt][ytnext][xt][xtnext] = binom.pmf(xtnext, xt, alpha ** yt)
                    elif xt == xtnext and yt == ytnext and xt + yt > x__1:
                        p[yt][ytnext][xt][xtnext] = 1
    return p


def p_to_2d(p, order_):
    p2d = np.zeros((order_ ** 2, order_ ** 2))
    for i in range(order_ ** 2):
        for j in range(order_ ** 2):
            p2d[i][j] = p[int(i/(order_))][int(j/(order_))][i % order_][j % order_]
    return p2d


# Gets stationary distribution by raising transition matrix to the power 500.
def get_stationary(trans_mat, pow=500):
    return matrix_power(trans_mat, pow)[0]

def run_monte_carlo(x__1, N=10**6):
    k = 0
    yt_sum = 0
    for _ in range(N):
        k += 1
        Yt = bin(x__1, eta)  # y0
        Xt = x__1 - Yt  # x0
        yt_sum += Yt
        while Yt > 0:
            k += 1
            Xnext = bin(Xt, alpha ** Yt)
            Yt = Xt - Xnext
            yt_sum += Yt
            Xt = Xnext
    mc_EY = yt_sum / k
    rate = mc_EY * k / N / x__1
    print("Avg Infectives over", N, "simulations:", mc_EY)
    print("AVG infection rate:", rate)
    return mc_EY, rate
"""
order_ is matrix order -> x_1 + 1
x__1 is just the number of people per batch (in the spec denote x0, in my work denoted x-1)
     i made it a param
"""
def find_expected_daily_infectives(x__1):
    p = init_P(x__1)

    order_ = x__1 + 1
    p_2d = p_to_2d(p, order_)
    print(p_2d)
    s = get_stationary(p_2d)
    # print(s)
    EY = 0
    for yt in range(order_):
        for ytnext in range(order_):
            for xt in range(order_):
                for xtnext in range(order_):
                    EY += ytnext * s[order_ * yt  + xt] * p[yt][ytnext][xt][xtnext]

    # print("Via Stationary dist: EY:", EY)
    return EY


def graph_monte_carlo_rate(mc_rates=None, separate_pic=True):
    if separate_pic:
        plt.figure(1)
    x0_vals = [i for i in range(1, 11)]  # x0=1 ... x0=10
    if not mc_rates:
        mc_rates = [run_monte_carlo(x0, N=10 ** 7)[1] for x0 in x0_vals]  # Expected Y values for all x0s. via monte carlo

    plt.scatter(x0_vals, mc_rates)
    plt.plot(x0_vals, mc_rates)
    plt.title("Long term expected rate of infections for various initial population sizes")
    plt.xlabel("Initial Population Size (x_-1)")
    plt.ylabel("Expected Infection Rate Value")
    plt.legend(["Expected Rate of Infection"])
    plt.show()

def graph_reed_frost_expected_y(separate_pic=True):
    if separate_pic:
        plt.figure(2)

def graph_mc_and_stationary_EXPECTED_Y(separate_pic=True):
    if separate_pic:
        plt.figure(3)
    x0_vals = [i for i in range(1,11)] # x0=1 ... x0=10
    st_ey = [find_expected_daily_infectives(x0) for x0 in x0_vals] # Expected Y values for all x0s. via stationary dist
    mc_out = [run_monte_carlo(x0, N=10**7) for x0 in x0_vals] # Expected Y values for all x0s. via monte carlo
    mc_ey = [v[0] for v in mc_out]
    mc_rates = [v[1] for v in mc_out]
    graph_monte_carlo_rate(mc_rates)

    plt.scatter(x0_vals, mc_ey)
    plt.plot(x0_vals, mc_ey)
    plt.scatter(x0_vals, st_ey)
    plt.plot(x0_vals, st_ey)

    plt.title("Expected Daily Infectives For various initial population sizes (x_-1)")
    plt.xlabel("Initial Population Size x_-1")
    plt.ylabel("Expected Infective Value")

    plt.legend(["Expected Daily Infectives from Monte Carlo Simulations",
                "Expected Daily Infectives from Stationary Distribution"])
    plt.show()

# Makes 2 graphs, one containing the expected Y values from both the MC sim. and Stationary dist.
# The other is of the infection rate
# Note that it uses 10**7 MC runs for each x0 value so takes a while
graph_mc_and_stationary_EXPECTED_Y()
```

Appendix A

The transition matrix will have the following form.

	$(0,0)$	$(1,0)$	…	$(x_{-1},0)$	…	$(x_{-1}-Y_{t+1},Y_{t+1})$	…	(x_{-1},x_{-1})
$(0,0)$	0	0	…	$P(y_0=0)$	…	$P(y_0=Y_{t+1})$	…	0
$(1,0)$	0	0	…	$P(y_0=0)$	…	$P(y_0=Y_{t+1})$	…	0
⋮	⋮	⋮	…	⋮	…	⋮	…	0
$(x_{-1},0)$	0	0	…	$P(y_0=0)$	…	$P(y_0=Y_{t+1})$	…	0
$(0,1)$	1	0	…	0	…	0	…	0
$(1,1)$	0	0.995	…	0	…	0	…	0
⋮	⋮	⋮	…	⋮	…	⋮	…	0
(x_{-1},x_{-1})	0	0	…	0	…	0	…	1

If $x_0 = 3$, then the matrix would be:



(X,Y)	(0,0)	(1,0)	(2,0)	(3,0)	(0,1)	(1,1)	(2,1)	(3,1)	(0,2)	(1,2)	(2,2)	(3,2)	(0,3)	(1,3)	(2,3)	(3,3)
(0,0)	0	0	0	0.857375	0	0	0.135375	0	0	0.007125	0	0	0.000125	0	0	0
(1,0)	0	0	0	0.857375	0	0	0.135375	0	0	0.007125	0	0	0.000125	0	0	0
(2,0)	0	0	0	0.857375	0	0	0.135375	0	0	0.007125	0	0	0.000125	0	0	0
(3,0)	0	0	0	0.857375	0	0	0.135375	0	0	0.007125	0	0	0.000125	0	0	0
(0,1)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1,1)	0	0.995	0	0	0.005	0	0	0	0	0	0	0	0	0	0	0
(2,1)	0	0	0.99003	0	0	0.00995	0	0	0.00003	0	0	0	0	0	0	0
(3,1)	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
(0,2)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1,2)	0	0.99003	0	0	0.01	0	0	0	0	0	0	0	0	0	0	0
(2,2)	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
(3,2)	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
(0,3)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1,3)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
(2,3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
(3,3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1