In [1]:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import math
import sympy as sym #used to manipulate matrices in Q4 and Q5

#These are used to define N choose R. They are not in this version of python I'm
using.
import operator as op
from functools import reduce
```

Question 1 (a)

Conditioning on X_t, each member of X_{t+1} follows a Bernouli trial of $\mathrm{Ber}(\alpha)$ of not getting infected. So X_{t+1} follows a binomial distribution of $X_{t+1} \sim \mathbf{Bion}(X_t, \alpha)$. Thus the expectation of X_t is then $\mathbb{E}[X_{t+1}|X_t] = \alpha X_t$.

$$\mathbb{E}[X_t] = \mathbb{E}[\mathbb{E}[X_t|X_{t-1}]] = \mathbb{E}[\alpha X_{t-1}] = \alpha \mathbb{E}[X_{t-1}].$$

Recurse on this formula and we arrive at

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

So

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

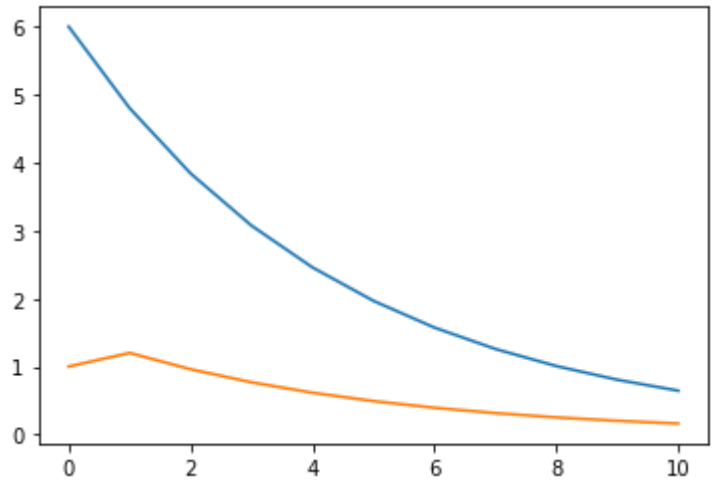
For Y_t we have $X_t + Y_t = X_{t-1}$. So the expected value for Y is

$$\mathbb{E}[Y_t] = \mathbb{E}[X_{t-1} - X_t] = \mathbb{E}[X_{t-1}] - \mathbb{E}[X_t] = \alpha^{t-1}(1-\alpha)x_0$$

Question 1(b)

In [2]:

```
x0 = 6
alpha = 0.8
t = 10
x_1=[x0] + t*[0]
y_1=[1] + t*[0]
for i in range(1, t+1):
    x_1[i] = alpha**i*x0
    y_1[i] = alpha**(i-1)*(1-alpha)*x0
figx = plt.figure(1)

plt.plot(range(0, 11), x_1)
plt.plot(range(0, 11), y_1)
plt.show()
```



Question 2 (a)

In the Reed and Frost model, a person in X_t is not infected only if avoidance with all Y_{t-1} people are avoided. Then clearly X_t follows a binomial distribution conditioned on (X_{t-1}, Y_{t-1}) and $X_t \sim \mathbf{Bion}(X_t, \alpha^{Y_{t+1}})$ and $Y_t = X_{t-1} - X_t$. The expection of X_t is $\alpha^{Y_{t-1}} X_{t-1}$ because X_t is a binomial distribution, and $Y_t = X_{t-1}(1 - \alpha^{Y_{t-1}})$ because $Y_t = X_{t-1} - X_t$. Then $\mathbb{E}[(X_t, Y_t)] = \mathbb{E}[\mathbb{E}[(X_t, Y_t)|(X_{t-1}, y_{t-1})]] = \mathbb{E}[\mathbb{E}[(\alpha^{Y_{t-1}} X_t, X_{t-1}(1 - \alpha^{Y_{t-1}})|(X_{t-1}, y_{t-1})]]$ $=\ldots= (\alpha^{y_{t-1}} x_t, x_{t-1}(1 - \alpha^{y_{t-1}}))$

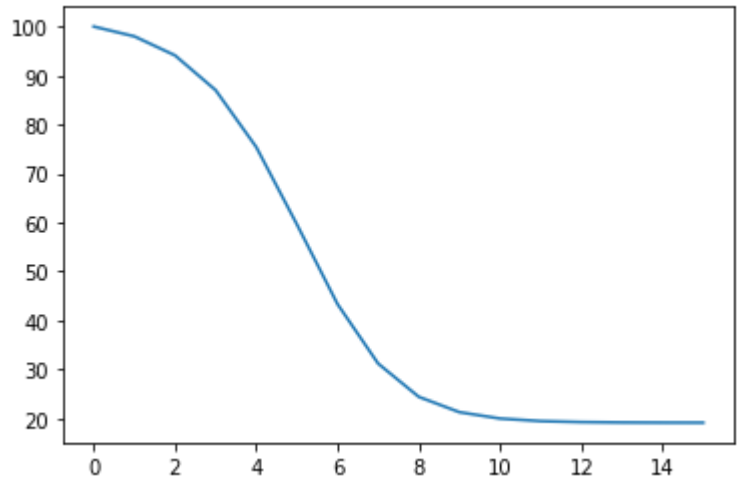
Question 2 (b)

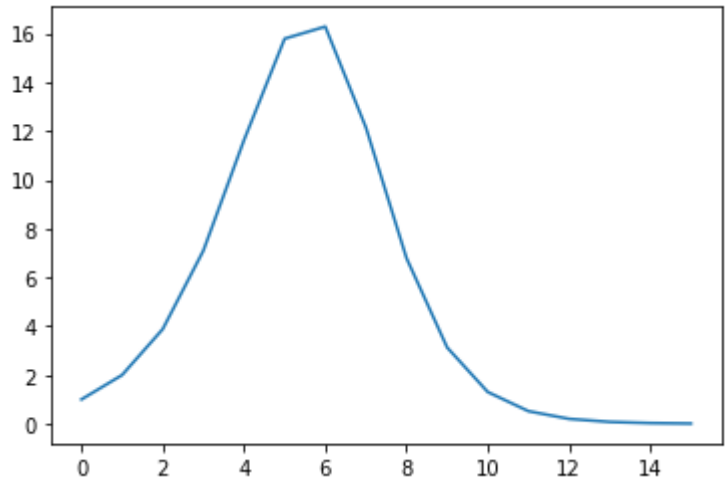
In [3]:

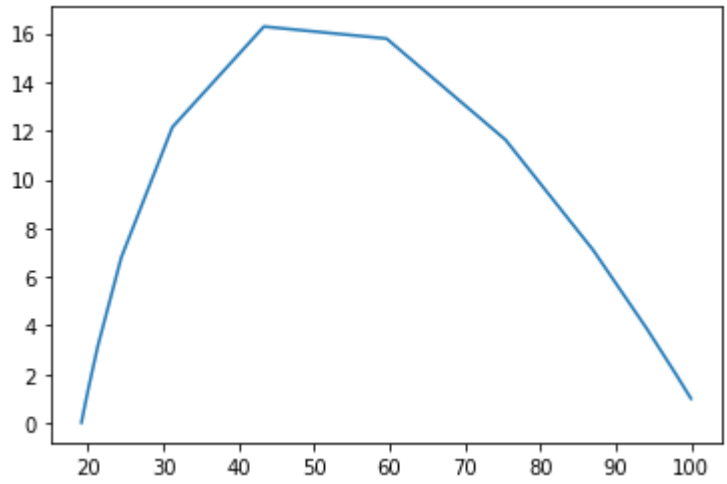
```
t = 15
x0 = 100
alpha = 0.98
x=[x0] + (t)*[0]
y=[1] + (t)*[0]
#y=(time+1)*[0]
for i in range(0, t):
    x[i+1] = x[i]*alpha**(y[i])
    y[i+1] = x[i]*(1-alpha**(y[i]))
plt.figure(1)
plt.plot(range(0, t+1), x)
plt.figure(2)
plt.plot(range(0,t+1), y)
plt.figure(3)
plt.plot(x, y)
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x7f4218b9e358>]
```







Question 3 (a)

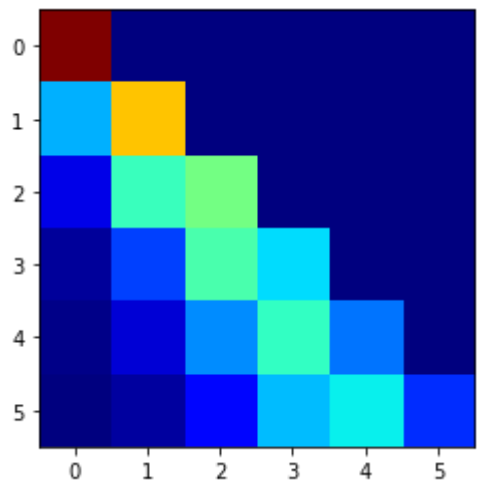
In [4]:

```
#defining N choose R in an efficient manner. Had to dig around some documentatio
n to arrive at this.
def nCr(n, r):
    if r<=n:
        r = min(r, n-r)
        numer = reduce(op.mul, range(n, n-r, -1), 1)
        denom = reduce(op.mul, range(1, r+1), 1)
        return numer / denom
    else:
        return 0
#defining matrix exponetiation, such that we can input the index as a variable
def Matexpt(matrix, exponent):
    if exponent == 0:
        n = np.shape(matrix)[1] #the dimension of matrix
        ID = [[1 if i == j  else 0 for i in range(1, n+1)] for j in range(1, n+1
)]
        return ID
    elif exponent == 1:
        return matrix
    else:
        result = matrix
        for i in range(exponent-1):
            result = result@matrix
        return result
```

In [5]:

```
#Figure 1
alpha = 0.7
x0 = 5

#defining the matrix in the paper, n choose r then  \alpha^j and (1-\alpha)^i-j
P = np.array([[ round(nCr(i, j)*((1-alpha)**(i-j))*(alpha**j), 4) for j in range
(x0+1)] for i in range(x0+1)])
fig1 = plt.figure(1)
fig1, axis = plt.subplots()
image = plt.imshow(P, cmap='jet')
axis.set_xticks([i for i in range(0, x0+1)])
axis.set_yticks([i for i in range(0, x0+1)])
plt.show(fig1)
```

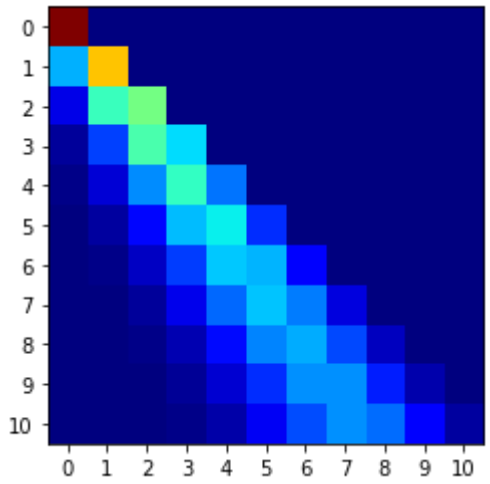
```
<Figure size 432x288 with 0 Axes>
```



In [6]:

```
#Figure 2
x0 = 10
#defining the matrix in the paper, n choose r then  \alpha^j and (1-\alpha)^i-j
P = np.array([[ round(nCr(i, j)*((1-alpha)**(i-j))*(alpha**j), 4) for j in range
(x0+1)] for i in range(x0+1)])
fig2 = plt.figure(2)
fig2, axis = plt.subplots()
image = plt.imshow(P, cmap='jet')
axis.set_xticks([i for i in range(0, x0+1)])
axis.set_yticks([i for i in range(0, x0+1)])
plt.show(fig2)
```

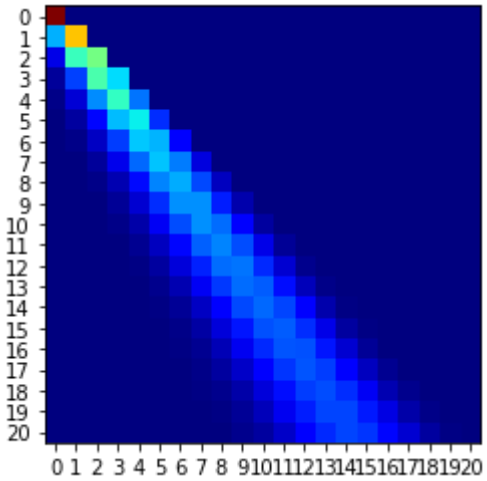
```
<Figure size 432x288 with 0 Axes>
```



In [7]:

```
#Figure 3
x0 = 20
#defining the matrix in the paper, n choose r then  \alpha^j and (1-\alpha)^i-j
P = np.array([[ round(nCr(i, j)*((1-alpha)**(i-j))*(alpha**j), 4) for j in range
(x0+1)] for i in range(x0+1)])
fig3 = plt.figure(3)
fig3, axis = plt.subplots()
image = plt.imshow(P, cmap='jet')
axis.set_xticks([i for i in range(0, x0+1)])
axis.set_yticks([i for i in range(0, x0+1)])
plt.show(fig3)
```

```
<Figure size 432x288 with 0 Axes>
```



Question 3 (b)

Every element in the state space S is in its own singleton communicating class. Proof: Let $i, j \in S$ be two states. By the transition matrix (an lower triangular real matrix) then if $i \to j$ then $j \nrightarrow i$, because the transition matrix is lower triangular. On the other hand $i \to i$ with probability α^i, so every state is in a singleton communicating class. Now for all $i \neq 0 \in S$, $p_{ii}^n = (\alpha^i)^n \to 0$ as $n \to \infty$, since $\alpha < 1$. So for all $i \neq 0$, i is a transient state. For $i = 1$, $p_{ii}^n = 1^n = 1$ for 0 is a recurrent state.

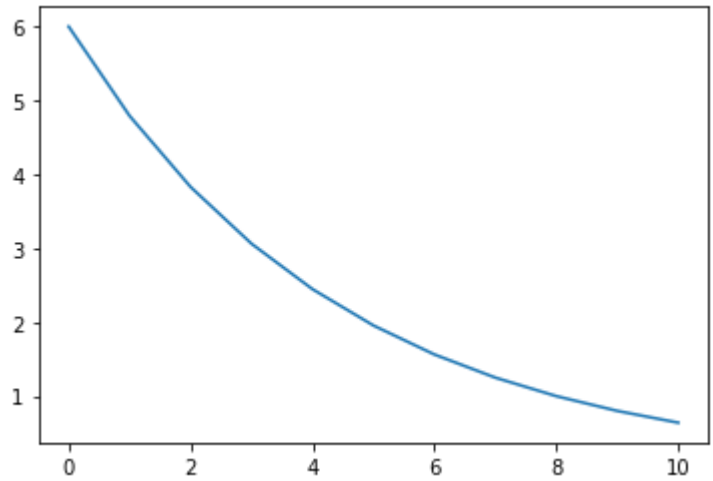
Question 3 (c)

In [8]:

```
#Question 3 (c)

#values for Question 1
x0 = 6
alpha = 0.8
t = 10

e = np.zeros(x0)
e = np.append(e, 1)
P = np.array([[ round(nCr(i, j)*((1-alpha)**(i-j))*(alpha**j), 2) for j in range
(x0+1)] for i in range(x0+1)])
v = [i for i in range(x0+1)]

vecA = 11*[0]
for time in range(0, t+1): #here t represents the time
    vecA[time] = e@Matexpt(P, time)@v

plt.plot(vecA)
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x7f4218a94f60>]
```



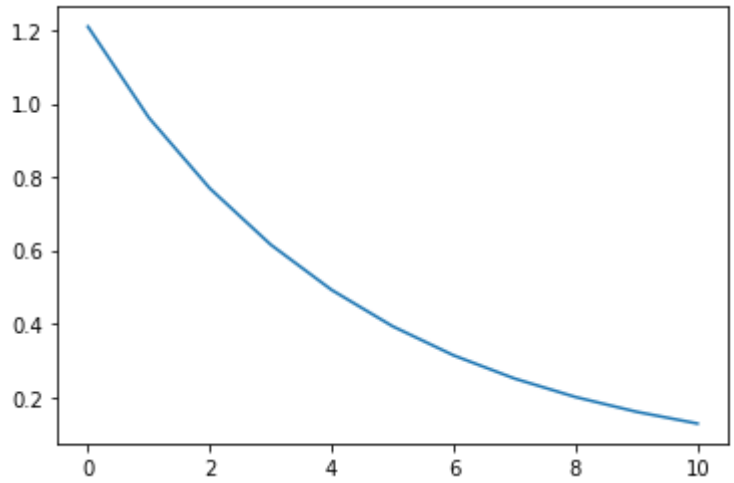
Question 3(c) The P^t part is the matrix of probability of where our population is likely to end up after t steps. Since we started with probability 1 that our population is $X_0 = x_0$ then we are interested in where our population ended up after t step, i.e. the bottommost row in P. So e captures our initial distribution, and eP^t capture the probability i. Finally, the v vector multiplies probability at state i with i to give the expected number of people at state i. Taking the sum of all possible state (which is a dot product between the row vector eP^t and v thus give our expected outcome.

In [9]:

```
#Question 3(d)
Q = np.array([[ round(nCr(i, j)*((alpha)**(i-j))*((1-alpha)**j), 2) for j in ran
ge(x0+1)] for i in range(x0+1)])
vecB = 11*[0]
for time in range(0, t+1): #here t represents the time
    vecB[time] = e@Matexpt(P, time)@Q@v

plt.plot(vecB)
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7f42189f88d0>]
```



Question 3(d)

To calculate Y_{t+1}, we need to calculate the chain $X_0, \dots X_t$, and then selected from a binomial distribution $Y_{t+1} \sim \mathbf{Bion}(X_t, 1-\alpha)$, because the chain Y_{t+1} depends not on the Y_t's, but on the X_t's. The transition matrix for the last one can be computed directly from the binomial distribution, by having $X_t = i$. Then the matrix is the matrix Q given above, where Q takes j infected individual from i healthy people at X_t.
Finally we take eP^tQv for the same reason as before, e specify that we begin at x_0, and v is the weight used to calculated expected number of person at each stage.

Questio 4 (a)

p_j^t roughly describes the probabiliy that there are j healthy individuals at time t, i.e. $(X_t = j)$. Since the population is decreasing by at least one at each time step (assumption in the Greenwood model), then for all $p_j^t = 0$ for $j > x_0 - t$. Then by law of total of total probability conditioning on X_{t-1}
$\mathbb{P}(X_t = j) = \sum_{i=0}^{x_0} \mathbb{P}(X_t = j | X_{t-1} = i)\mathbb{P}(X_{t-1} = i) = \sum_{i=j+1}^{x_0-(t-1)} p_{ij} p_i^{t-1}$

In [10]:

```
#Question 4 (b)
def transition(i,j, alpha):
    '''probability of j sick given i was sick a time interval just before'''
    return nCr(i, j)*(1-alpha)**(i-j)*alpha**j

def healthyattimet(j, t, x0, p):
    '''give the probability of j healthy at time t'''
    tprime = t
    pt  = np.zeros(x0+1)
    pt[x0]  = 1
    ptplus1= np.zeros(x0+1)
    while t >= 0:
        for j1 in range(x0+1):#we fill ptplus1, i.e. we calculate p_j^t for each
j
            value = 0
            for k1 in range(j1+1, x0-(tprime - t )+1):
                value = value + pt[k1]*transition(k1,j1,p)
            ptplus1[j1] = value
        for i1 in range(x0+1): #put ptplus1 into pt and reduce the counter
            pt[i1] = ptplus1[i1]
        t = t-1
    return pt[j]



def gamma(k, n, x0, p):
    '''Tells us the prob of k sick at time n, given x0 and p=alpha'''
    return healthyattimet(x0-k, n-1, x0, p)*p**(x0-k)
```

In [11]:

```
x0= 6
alpha = 0.8
sum = 0
for t in range(0,6+1):
    for w in range(5, 6+1):
        sum = sum + gamma(w, t, x0, alpha)
sum
```

Out[11]:

```
0.13029835750312502
```

Question 4(b)

$\mathbb{P}(W > 4) = 1 - \mathbb{P}(W <= 4)$, but (W, T) are bi-increasing, so to compute $\mathbb{P}(W <= 3)$, we only need to consider the case when $W, T \leq 3$.

In [12]:

```
#Question 4 (c)

x0 = 6
n = 100000
W = np.zeros(n)
T = np.zeros(n)


alpha = 0.8

for i in range(n):
    xvec = [x0]
    while True:
        val = 0
        for j in range(xvec[-1]):
            if np.random.rand(1)< alpha:
                val = val + 1
        xvec.append(val)
        if xvec[-1] == xvec[-2]:
            W[i] = x0 - xvec[-2]
            T[i] = len(xvec) - 2
            break
(W>4).sum()/n
```

Out[12]:

```
0.13085
```

Question 4 (d)

In [13]:

```
x0 = 6
alpha = 0.8

phi = sym.symbols('phi')
P = sym.Matrix(list([[nCr(i, j)*((1-alpha)**(i-j))*(phi**(i-j))*(alpha**j) for j
in range(x0+1)] for i in range(x0+1)]))
def qmatrixcreatefunction(i,j):
    if i == j:
        return alpha**i
    else:
        return 0

Q = sym.Matrix(x0 + 1, x0 + 1, qmatrixcreatefunction)
A = sym.Matrix([[0, 0, 0, 0, 0, 0, 1]])
E = sym.Matrix([[1,1,1,1,1,1,1]])
I = sym.eye(x0+1)
Pbar = P - Q
pgf4w = A*(I - Pbar).inv()*Q*E.T
one = sym.Matrix([[1]])

pgf4wdash = sym.diff(pgf4w).subs(phi, 0)
pgf4wddash = sym.diff(sym.diff(pgf4w)).subs(phi, 0)*sym.Matrix([[1/2]])
pgf4wdddash = sym.diff(sym.diff(sym.diff(pgf4w))).subs(phi, 0)*sym.Matrix([[1/6
]])
pgf4wddddash = sym.diff(sym.diff(sym.diff(sym.diff(pgf4w)))).subs(phi, 0)*sym.Ma
trix([[1/24]])
prob = one - (pgf4w.subs(phi,0) + pgf4wdash + pgf4wddash + pgf4wdddash + pgf4wdd
ddash)
```

In [14]:

```
prob
```

Out[14]:

```
Matrix([[0.130298357503125]])
```

Question 5(a)

We set $x_{-1} = x_0 + y_0$. In this model, we must keep track of Y_t as well as X_t, since X_{t+1} is dependent on Y_t, the number of people currently infected. Now $X_{t-1} = X_t + Y_t$ and (for some t, X_t can take any vale from 1 to x_{-1} . But of course, the initial state is determined, so there is only one possible state where $x + y = x_{-1}$, namely (x_0, y_0). So the possible state space is of the form

$$S = \{(x, y) | x + y \leq x_{-1} - 1\} \cup \{(x_0, y_0)\}$$

With the specification $x_0 = 6$ and $y_0 = 1$ (we're not given this value, so I assumed $y_0 = 1$ because this is also the value they used) we have

$$S = \{(x, y) | x + y \leq 6\} \cup \{(6, 1)\}$$

Question 5 (b)

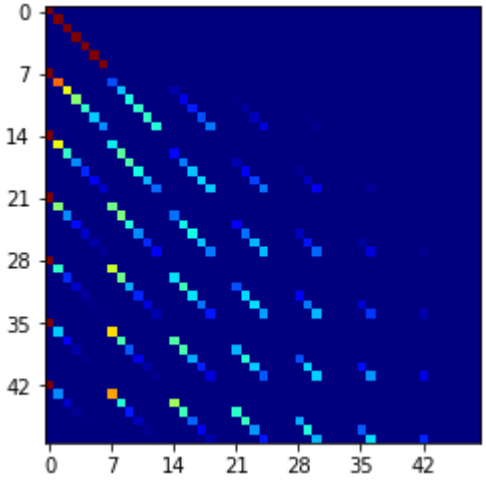
Each state is in its own communicating class and only states with $Y_t = 0$ is recurrent.

Proof: Either $Y_t = 0$ or $Y_t > 0$. In the first case we the chain stop because $X_{t+1} \sim \mathbf{Bion}(X_t, \alpha^{Y_t} = 1) = X_t$, so $(x, 0)$ is a absorbing state (i.e. a singleton recurrent communicating class). In the second case $Y_t > 0$, in which case X_{t+1} is strictly monotonically decreasing or is one of the absorbing states $(x, 0)$, so it cannot communicate with X_t. $X_t \neq X_{t+1}$ implies they are in distinct communicating class. Furthermore, there can be no communication between (x, y_1) and (x, y_2) since if $(X_t, Y_t) = (x, y_1)$, then $X_{t+1} = x$ then $Y_{t+1} = 0$ which is an absorbing state. So every state is in its singleton communicating class, and only states with $Y_t = 0$ is recurrent.

In [15]:

```
#Question 5 (c)

alpha = 0.8
x0 = 6
#defining the submatrix
def p(i,j , x0, alpha):
    M = np.zeros((x0+1, x0+1))
    for i1 in range(x0+1):
        for j1 in range(x0+1):
            if j1 >=j:
                M[j1][j1-j] = (nCr(j1, j)*(alpha)**(i*(j1 - j)))*(1 - alpha**i)*
*j
    return M

#creating the P matrix
for i in range (x0 + 1):
    for j in range(x0 + 1):
        if j == 0:
            A =  p(i,j,x0,alpha)
        else:
            A = np.append(A, p(i,j,x0,alpha), axis=1)
    if i == 0:
        P = A
    else:
        P =  np.append(P, A, axis=0)

fig1 = plt.figure(1)
fig1, axis = plt.subplots()
image = plt.imshow(P, cmap='jet')
axis.set_xticks([i for i in range(0, (x0+1)**2, x0 + 1)])
axis.set_yticks([i for i in range(0, (x0+1)**2, x0 + 1)])
plt.show(fig1)
```

```
<Figure size 432x288 with 0 Axes>
```



In [16]:

```
#Question 5(c)

x0 = 6
n = 100000
W = np.zeros(n)
T = np.zeros(n)


alpha = 0.8

for i in range(n):
    xvec = [x0]
    yvec = [1]
    while True:
        val = 0
        for j in range(xvec[-1]):
            if np.random.rand(1)< alpha**(yvec[-1]):
                val = val + 1
        xvec.append(val)
        yvec.append(xvec[-2] - xvec[-1] )
        if xvec[-1] == xvec[-2]:
            W[i] = x0 - xvec[-2]
            T[i] = len(xvec) - 2
            break
(W>4).sum()/n
```

Out[16]:

```
0.25736
```

The results are higher than Question 4 (c). Of course this is true becase the more people we have infected, the less likely a suseptable person is going to stay healthy. The greenwood model basically assumes $Y_t = 1$ when calculating whether people are going to get infected in α^{Y_t}, where as this mode, the higher Y_t the less likely you're going to stay healthy. As W is how many people are infected, we expect $\mathbb{P}(W > 4)$ to be higher than Question 4 (c).

Question 6

See attached LaTeX document.

STAT3004 Project 1 - Epidemic Modelling (Q6)

Abstract

In light of the novel COVID-19, we numerically model the behaviour of the epidemic in a small isolated unit, which acts as a gateway into a territory. We carried out numerical and computational analysis of our model based on the works of Reed and Frost (1925)[1] and conclude the rate of infections are marginally higher.

1 Senario and Assumptions

In this scenario, we model COVID-19 epidemic in a closed isolation unit with a capacity of x_0 people, each with a probability $\eta = 0.05$ of arriving pre-infected, for a total of y_0 pre-infected. A susceptible person has a probability p to come in contact with one of y_0 and a probability β being infected by that person. We assume that a daily rapid testing with total accuracy and seneitivity is conducted with one day turnaround. Those test positive are removed, and the rest remain in the unit until all return negative, in which we replace then with new x_0 people. Mathematically, this is $X_t = X_{t+1} + Y_{t+1}$ and (X_t, Y_t) is a bivariate Markov chain. Once $Y_t = 0$, new x_0 people takes in their place and we repeat. We wish to compare the average infection rate in this isolation unit in per person per day.

2 Model and Computational Analysis

We assume $p = 0.01$ and $\beta = 0.05$, so probability of on person not infected is $\alpha = 0.9995$. Then $X_0 \sim \mathsf{Bion}(x_0, \eta)$, $X_t \sim \mathsf{Bion}(X_{t-1}, \alpha^{Y_t})$, for $t \geq 1$ and $Y_t = X_{t-1} - X_t$ for all t. We assume $\eta = 0.05$.

We calculate the new rate of infection $\bar{\eta}$ via PGF $\rho(w, t)$ of the joint distribution (W, T), where W is the number of infected at the stopping time T when $Y_t = 0$. We modify the procedure found in Daley and Gani (1999). Since X_t is binomially distributed, we construct the submatrix P_{ij} of rank (x_0+1) with entries $\binom{k}{l}(1-\alpha^i)^j\alpha^{il}$ along the jth subdiagonal for some fixed i, j, and zero otherwise. So P_{ij} has rows and column indexed by $X_t = k$ and $X_{t+1} = l$ respectively, and P_{ij} represents the transition matrix from $Y_t = i$ to $Y_{t+1} = j$. Let $P = (P_{ij})$ be the block matrix containing P_{ij} as submatrices, so P is a transition matrix of rank $(x_0 + 1)^2$. Thus the PGF has the form

$$\rho(w, t) = A(I - t(P(w) - Q(w))^{-1}tQE$$

where w, t are variables, I is the identity matrix of appropriate size, Q is the matrix with entries from P if and only if $Y_{t+1} = 0$, E is a $(x_0 + 1)^2$ column vector with entries 1 and A is a row vector consists of entry $\binom{x_0}{k}(\eta)^k(1 - \eta)^{x_0-k}$ at position $k(x_0 + 1) + (x_0 + 1 - k)$ and zero elsewhere. In essence, we modified the values of A in Daley and Gani (1999) to reflect the initial distribution of population with respect to $X_0 \sim \mathsf{Bion}(x_0, \eta)$. So we set $t = 1$ and calculate w computationally.

We also verify the results from PGF with a Monte-Carlo simulation $n = 10^6$ samples for each $x_0 = 1, 2, \ldots, 10$, by dividing the number of infected over number of person. That is $\hat{\eta} = W/nx_0$. The numerical results can be found below.

[1] Daley DJ. and Gani J. Epidemic Modelling: An Introduction. Cambridge University Press 1999

Capacity size x_0	1	2	3	4	5
MC Simulation	0.05020	0.05010	0.05059	0.05078	0.05104
PGF Method	0.05000	0.05024	0.05058	0.05072	0.05096
Capacity size x_0	6	7	8	9	10
MC Simulation	0.05141	0.05159	0.05166	0.05185	0.05224
PGF Method	0.05121	0.05146	0.05171	0.05196	0.05222

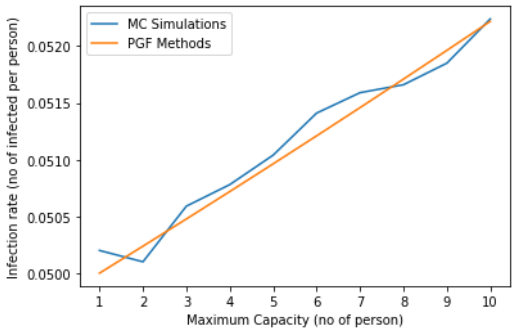


Figure 1: The result comparing the rate of infection of our model with the PGF (Orange) and the Monte Carlo *blue* method. Both shows a linear increasing for small value of x_0 above the overall rate of infection $\eta = 0.05$.

3 Discussions

From the PGF, we see a linear increase with a gradient 2.2×10^{-4} rate per person. As $x_0 \to \infty$, we expect $\hat{\eta} \to \eta$, since larger value of x_0 behaves like a population size. So $\hat{\eta}$ attains a maximum before returning to $\eta = 0.05$, and $\hat{\eta}$ exhibit a local increase behaviour for small x_0.

We assumes the population had no immunity to COVID-19. This is largely untrue, for Sheridan (2020)[2] suggests young adults have some innate immunity to COVID-19. Then α would depend on risk factors, so $\hat{\eta}$ differ in a typical household of 2 of each kids, adults, and grandparents

COVID-19 practice, has an incubation of 14 days, meaning a test with one day turnaround and perfect accuracy/sensitivity are unrealistic. During of which essential trips like groceries may further contribute to the infections from outside. Though new advancement in proteonomic/genetic testing seems promising and may approach our model of the testing procedure (Kolifarhood et.al, 2020)[3].

Despite the marginal increase in the longterm infection rate, Shoukat, Wells et.al (2020)[4] demonstrates that self-isolation lessens the burden of ICU beds, resulting in proper healthcare for more. It is possible that trading off a marginal increase for a proper health care will overall reduces the likelihood of death. For this reason we conclude that despite the marginal increase, it is advisable to isolate at home.

4 Conclusions

We present a brief exploratory analysis of the COVID-19 spread in small household. We see a marginal increase in η, conclude that further modelling is required and advice self-isolation.

[2]Shoukat, Wells et.al.Projecting demand for critical care beds during COVID-19 outbreaks in Canada. CMAJ 2020, doi: 10.1503/cmaj.200457

[3]Kolifarhood, Goodarz et al. Epidemiological and Clinical Aspects of COVID-19; a Narrative Review. Archives of Academic Emergency Medicine 2020, vol. 8,1 e41. 1 Apr.

[4]Sheridan C. Coronavirus and the Race to Distribute Reliable Diagnostics. Nat. Biotechnol. 2020, doi: 10.1038/d41587-020-00002-2.