UPPSALA
UNIVERSITET

Modelling complex systems

Project 2

Population Dynamics
Groups of Friends
Network Epidemics
Flocks and Predators

Peili Guo
Peili.Guo.7645@student.uu.se

May 7, 2018

1 Population Dynamics

In this part, we model the population with a stochastic model. There are n resource sites in the model world, and at time t =0, the population is A_0 and they are assigned randomly to one resource site. At each t step, the population rules are, if there are exactly two individuals on the same site. They reproduce b offsprings and these offsprings are assigned randomly to resources sites. If the number of individuals on a resources sites is any number other than 2, no offspring will be reproduced.

1.1 Matlab model

To begin, we can run this with different parameters and simulate in Matlab and observe the total number of population at different time step. The different parameters are:

1. b: number of offspring if reproduce
2. n: total of number of resource sites
3. A_0: initial population
4. t: the time steps we want to simulate the model

When set the initial population to a small number relative to the resource site, it would be difficult to have 2 individuals at the same site for reproduce, and even if they reproduce a large number of offspring the population dies out very quick. below are some plots showing the total number of population at different time steps with different parameters.

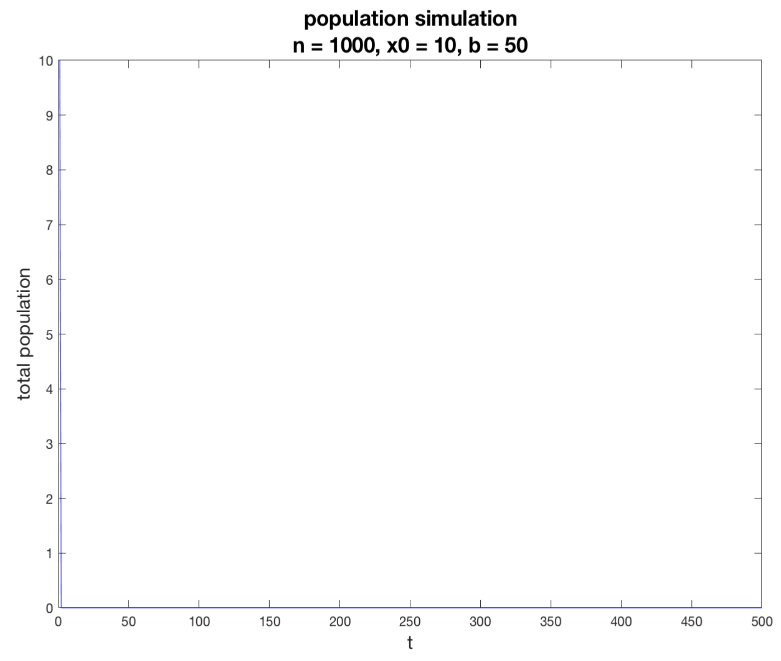

Figure 1: initial population of 10 b = 50 not reproducing

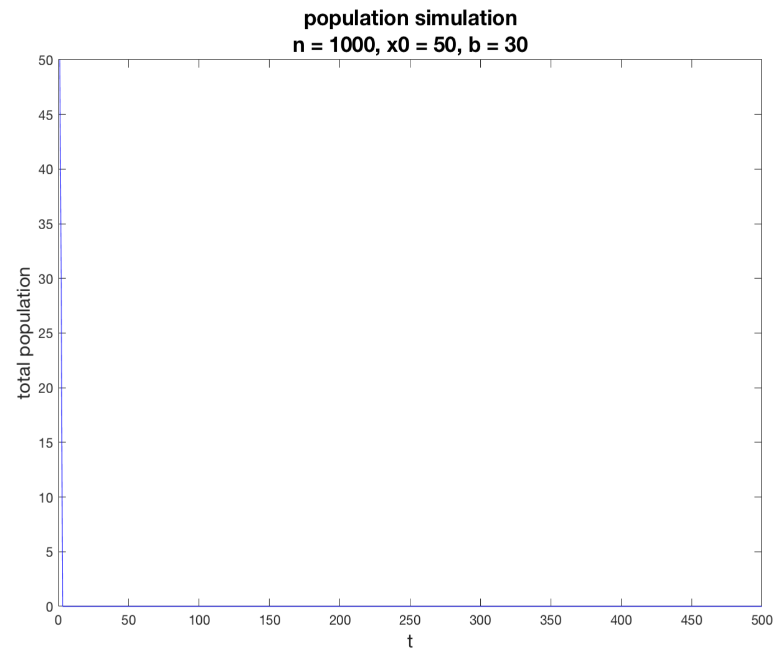

Figure 2: initial population of 50 and b =30 not reproducing

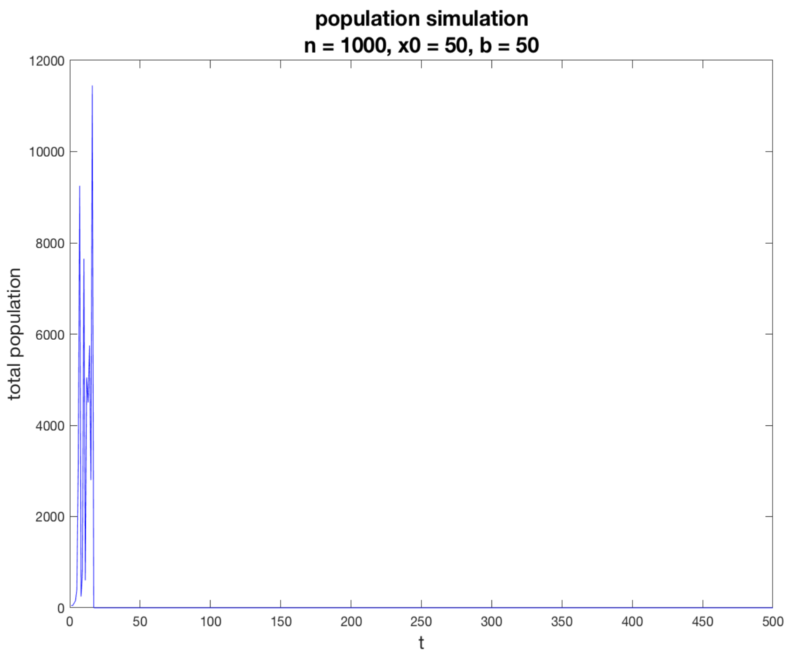

Figure 3: initial population of 50 and b = 50, in the beginning reproduce but quickly dies

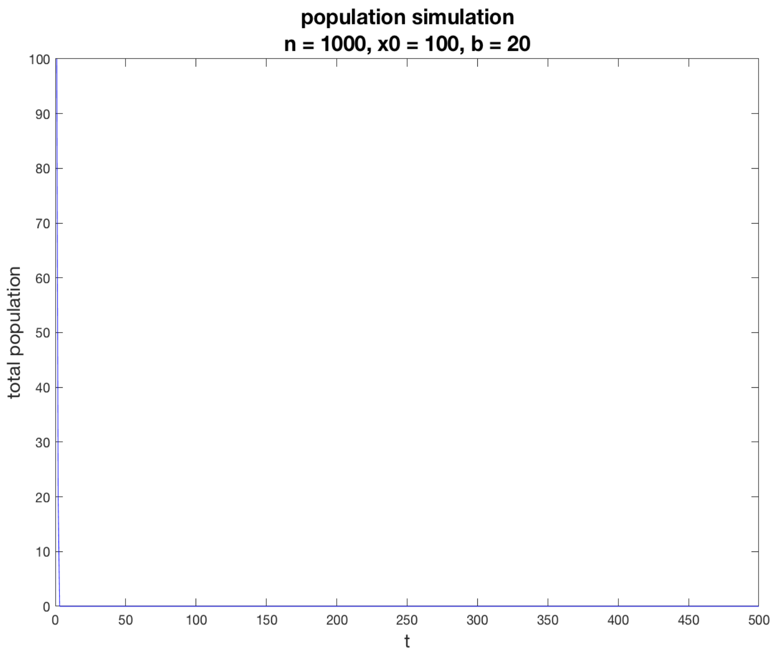

Figure 4: initial population of 100 and b = 20, it dies

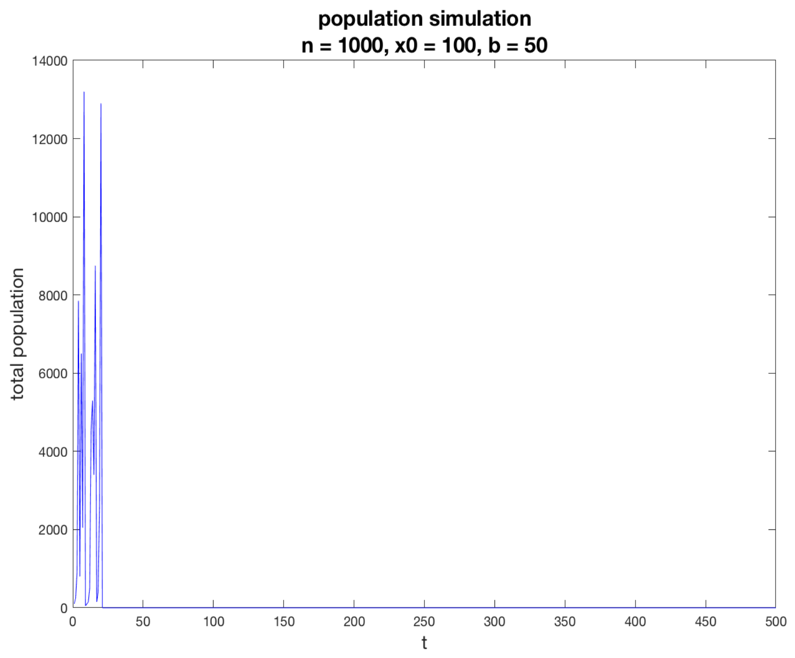

Figure 5: initial population of 100 and b = 50, it reproduce in the beginning but quickly dies

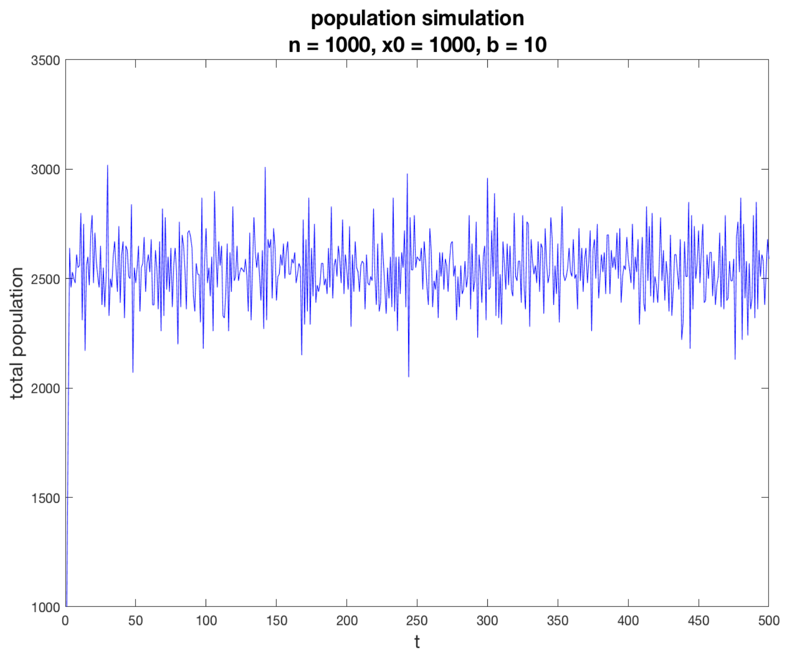

Figure 6: initial population of 1000 and b = 10, it show that the population oscillates around 2500

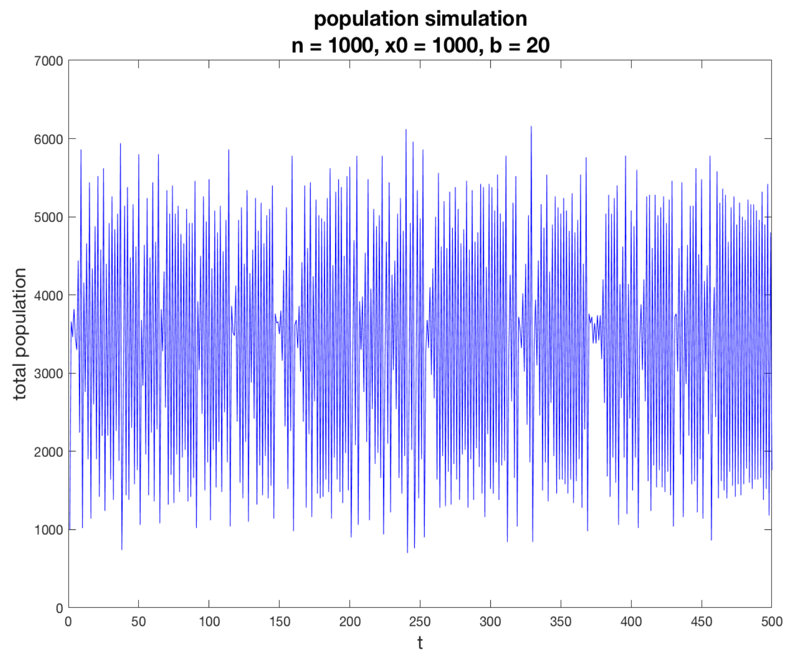

Figure 7: initial population of 1000 and b = 20, the populations starts to get chaotic

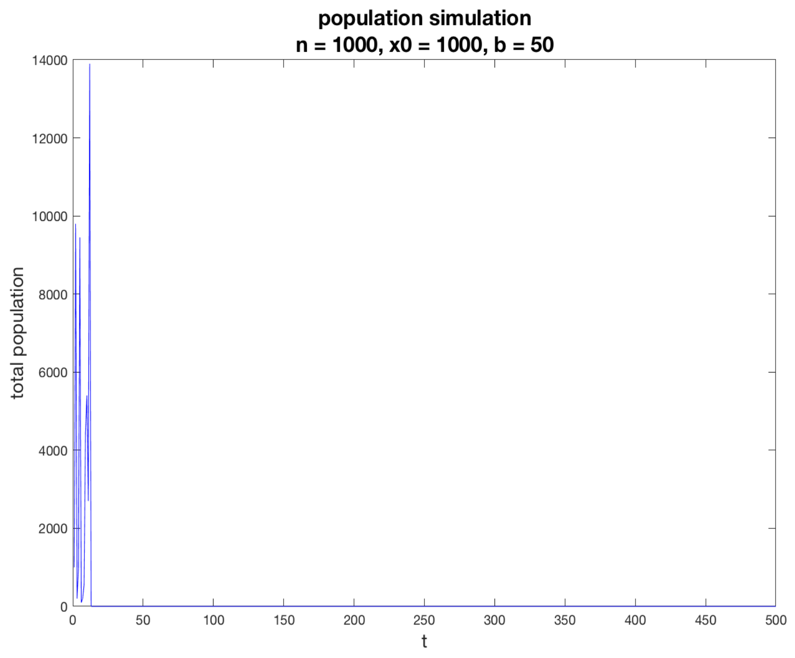

Figure 8: initial population of 1000 and b = 500, the populations dies very quick

To further study this model, we run it with $A_0 = 1000$, n= 1000, b = 1, 2, 3, 4, 5,...,48, 49, 50. and at each b value, run the simulation 100 times and plot a phase transition diagram. and we can see that when b = 5-15, the population increases steady and oscillates around a number. When $15 < b < 35$, the population will get more chaotic and at one time a lot of sites are reproducing, and then the next time steps, they dies because of overcrowding. When $b > 35$ the population will dies very quick due to overcrowding.

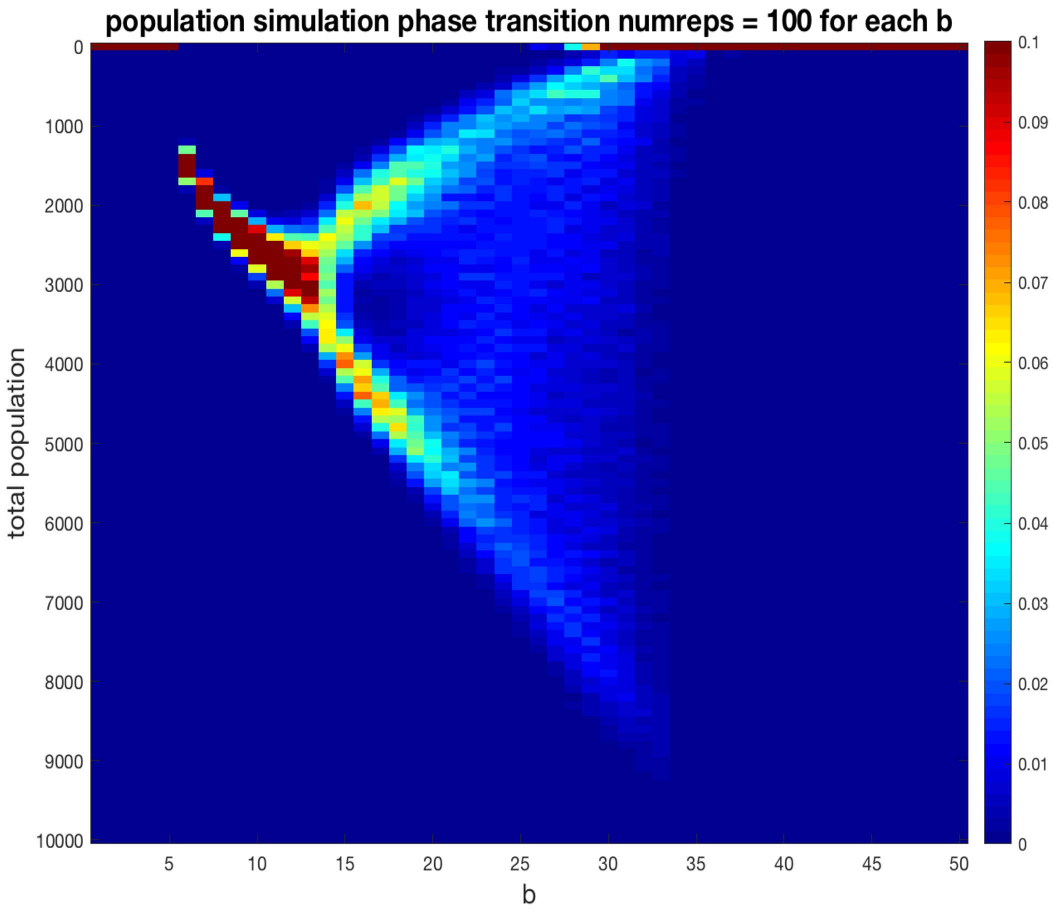

Figure 9: initial population of 1000 and b = 500, the populations dies very quick

1.2 Mean field model

We assume the sites are independent and number of individuals at sites is poisson distributed. I plot selected mean-field model here for b = 5, 10, 15, 20, 35, 50. The model are after the plot

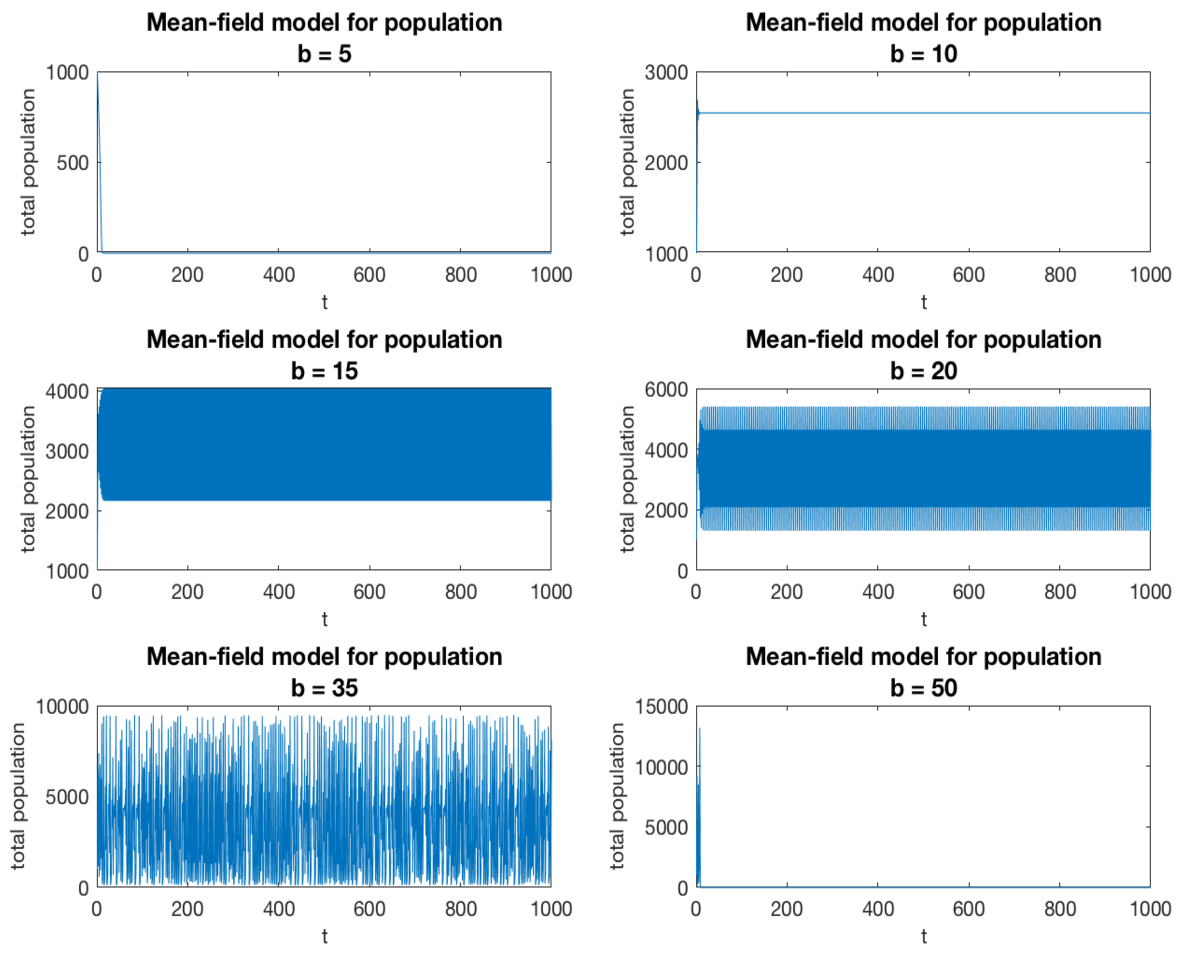

Figure 10: mean field model with selected b value

$$\begin{cases} E(A_{t+1}|A_t) = n \times \sum_{k=0}^{n} P_k \times \phi_k & (1) \\ \phi_k = \begin{cases} b, if k = 2 \\ 0, if k \neq 2 \end{cases} & (2) \end{cases}$$

following the mean field model, and the population only reproduce if there are exactly 2 individuals at same site, we can write the following:

$$A_{t+1} = n \times P(2 at site, A_t) \times b \tag{3}$$

$$A_{t+1} = n \times \frac{(\frac{A_t}{n})^2 \times e^{-\frac{A_t}{n}}}{2} \times b \tag{4}$$

For steady state, we have $A_{t+1} = A_t$.

$$A_t = n \times \frac{(\frac{A_t}{n})^2 \times e^{-\frac{A_t}{n}}}{2} \times b \tag{5}$$

When $A_t = 0$, left hand side always equal to right hand side, they are both 0. When $A_t \neq 0$, we have

$$1 = \frac{b \times A_t}{2n} \times e^{-\frac{A_t}{n}} \tag{6}$$

Let's rearrange equation (6) and let $\frac{A_t}{n} = x$, we have:

$$1 = \frac{b}{2} \times x \times e^{-x} \tag{7}$$

$$x \times e^{-x} = \frac{b}{2} \tag{8}$$

To find the conditions in terms of b for the existence of two further non-zero steady states, we need to find that $f1(x) = x \times e^{-x}$ has intersection with a horizontal line. b need to fulfill the condition that $\frac{2}{b} < 0.3679$, therefore we get: $b > 5.4366$, as b needs to be integer, we get $b > 5$.

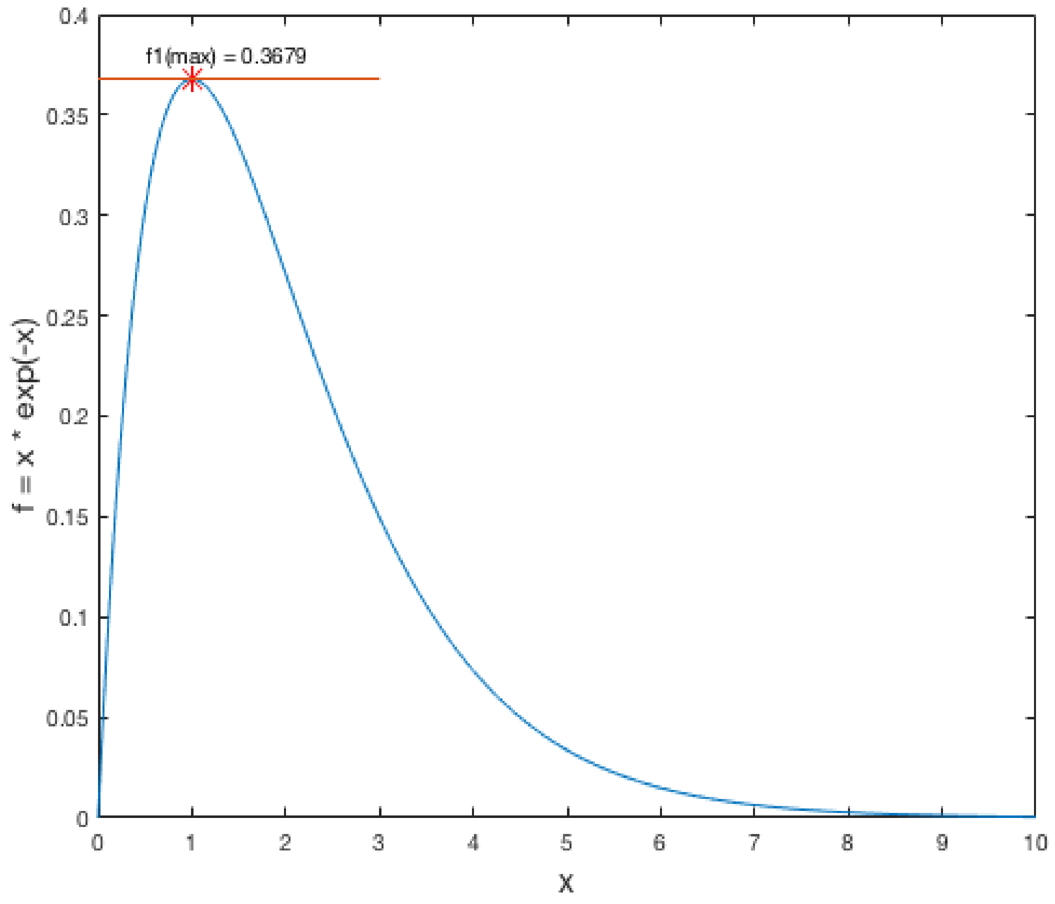

Figure 11: plot of f1(x) = x * exp(-x)

1.3 Lyapunov exponent

From the previous section, we have derived the equation for A_t. that we have

$$f(A_t) = n \times \frac{(\frac{A_t}{n})^2 \times e^{-\frac{A_t}{n}}}{2} \times b \tag{9}$$

$$f(a) = \frac{b}{2n} \times a^2 \times e^{-\frac{a}{n}} \tag{10}$$

$$f'(a) = \frac{b}{2n} \times (2a \times e^{-\frac{a}{n}} + a^2 \times e^{-\frac{a}{n}} \times (-\frac{1}{n})) \tag{11}$$

$$f'(a) = \frac{b}{2n} \times (2a \times e^{-\frac{a}{n}} - \frac{a^2}{n} \times e^{-\frac{a}{n}}) \tag{12}$$

$$|\Delta a_n| = |\Delta a_0| e^{\lambda n} \tag{13}$$

to compute Lyapunov exponent λ numerically, we use the following:

$$\lambda = \frac{1}{n} \sum_{t=0}^{n-1} ln|f'(a_t)| \tag{14}$$

The Lyapunov exponent are plotted below: we can see that when $\lambda > 0$ the diverges, that's when b is between 21-30, the population become chaotic. When $b \leq 5$ and $b \geq 31$ λ is -infinity that means the system converge very fast, and this explains why the population become extinct very fast for large b.

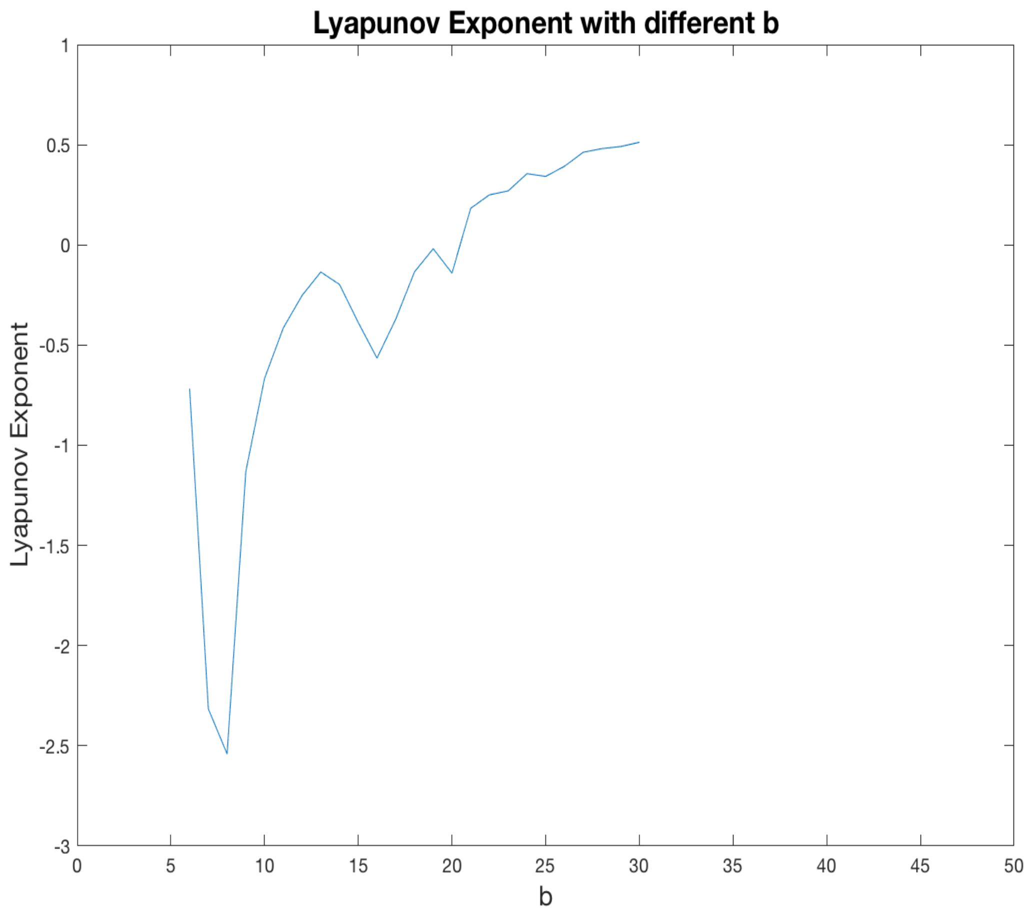

Figure 12: plot of Lyapunov exponent

2 Groups of friends

In this part, we used a model to simulate how students make friends. Initially, there are N students, and they are all alone in a group contains only one student. At each time step, a group is picked:

1. If the picked group contains only one student, this students will join a group with size k with probability of k/N.
2. if the picked group size i ($i > 1$), then this group will split up with probability of ri.

2.1 simulations in Matlab

First, we simulate the above model with 100 students, r = 0.01 and 100 replications. First, we plot the group size vs frequence on a log log scale. Second, we get the relative frequency with group size histogram taken on a log scale that is we take groups size [1,2,4,8,16,32,64,128] and plot the group size vs relative frequency on a log log plot. in figure 14, we can observe that the group distribution following the power law.

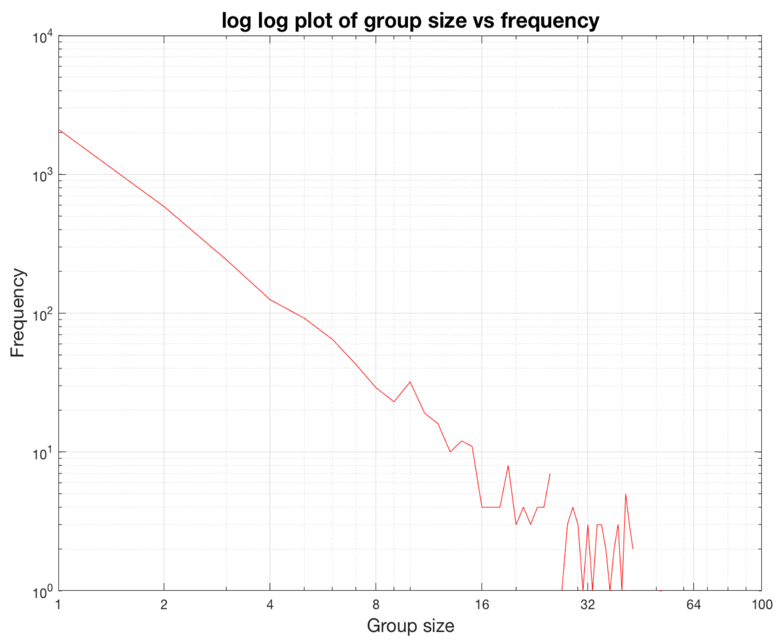

Figure 13: log log plot of group size vs frequency

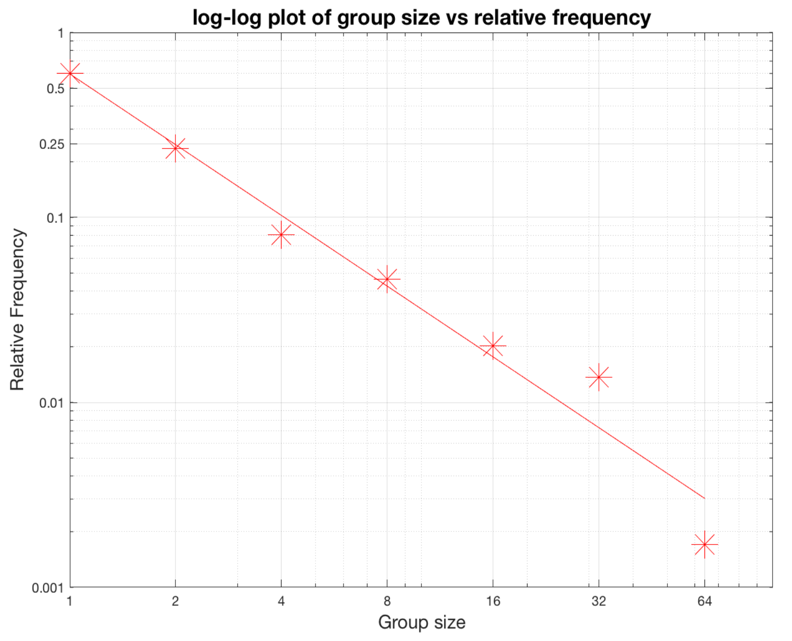

Figure 14: log log plot of group size vs relative frequency

To investigate the how the group size distribution changes with r, we run the model with r from 0.001:0.001:0.1 with 100 replications. In 15, I choose some r values to plot shows how the group size changes, and in 16, the whole group distribution is shown on a 3d plot, it might be a little difficult to from the colourful map, but we can observe the z-axis value that the relative frequency of the group.

When r is small at 0.001 we have groups of size 1 and 2 around $\frac{1}{4}$ of the population and some large groups. when we increase the r, then the number of groups of 1 become the dominant group in the distribution and large group appear less and less. With a bigger r, the larger group will split up if it was chosen at time t. For example, setting r = 0.1, it means that groups of size 10 + will slip up at the probability of 1 if chosen at time t.

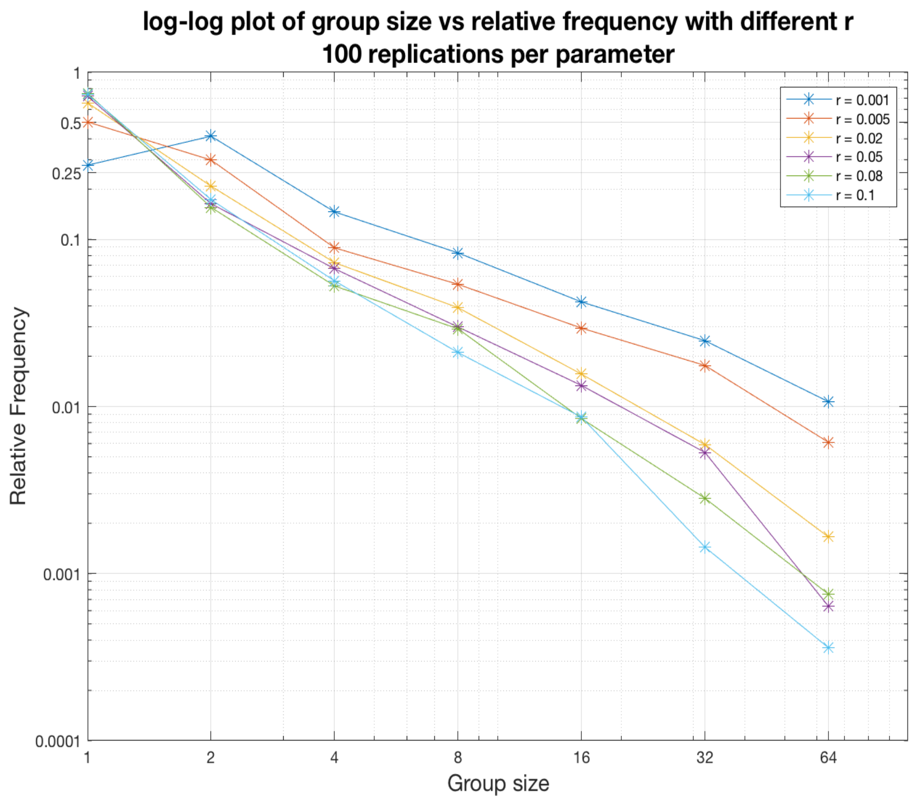

Figure 15: log log plot of group size vs relative frequency with different r

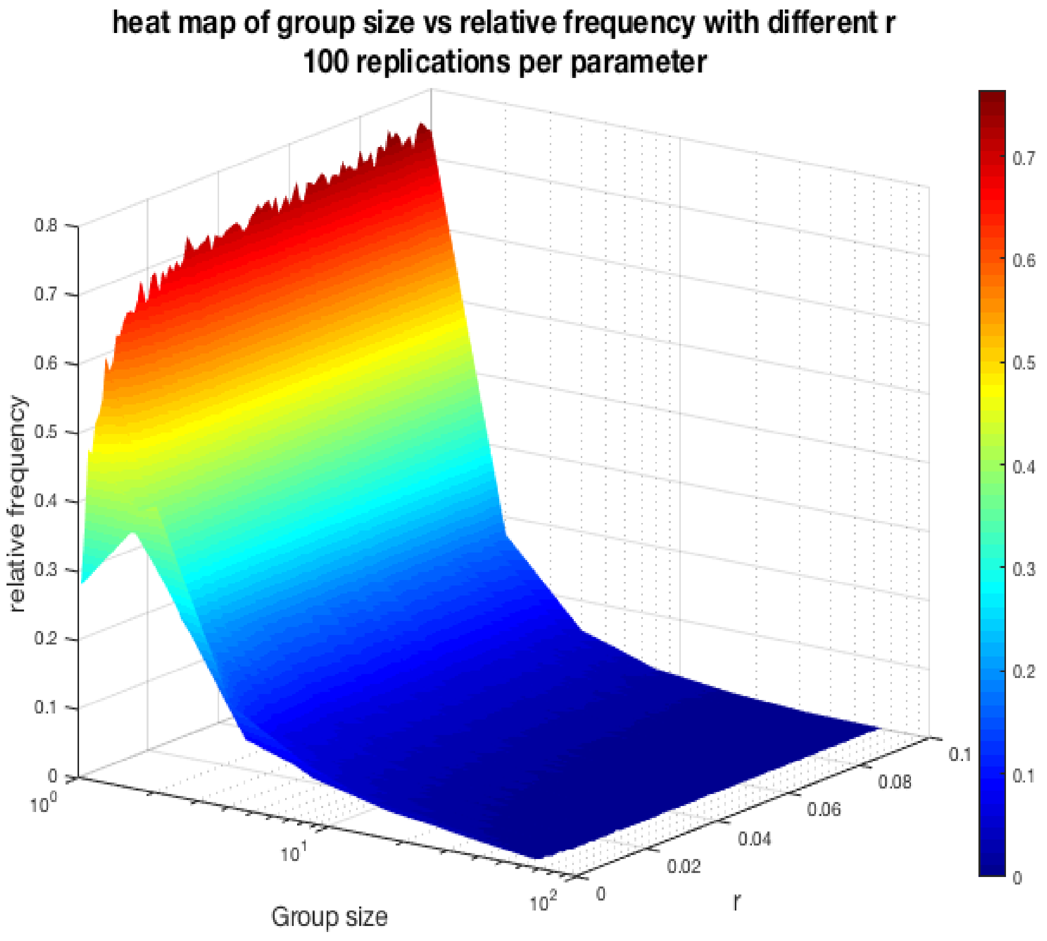

Figure 16: the distribution of groups with different r showing on a 3d plot

2.2 Master Equation

For $k > 1$: we have $\pi(k, t+1) = \pi(k, t)$ + (k-1)group get a new one - (k) group get a new one become (k+1)group - (k) group split. then we get

$$\pi(k,t+1) = \pi(k,t)+\pi(1,t)\times\frac{k-1}{N}\times\pi(k-1,t)-\pi(1,t)\times\frac{k}{N}\times\pi(k,t)-\pi(k,t)\times(kr) \tag{15}$$

For k = 1:

$$\pi(1,t+1) = \pi(1,t) - \pi(1,t)\times\frac{1}{N} + \sum_{j=2}^{n}\pi(j,n)\times j^2\times r \tag{16}$$

$$\pi(1,t+1) = \pi(1,t) + \sum_{j=2}^{n}\pi(j,n)\times j^2\times r \tag{17}$$

when $k > 1$, at steady state, we have $\pi(k, t+1) = \pi(k, t)$.

$$\pi(1)\times\frac{k-1}{N}\times\pi(k-1) = \pi(k)\times(kr+\pi(1)\times\frac{k}{N}) \tag{18}$$

$$\pi(k) = \frac{k-1}{Nkr+\pi(1)k}\times\pi(1)\times\pi(k-1) \tag{19}$$

now we find c:

$$c = \frac{\pi(1)}{Nr+\pi(1)} \tag{20}$$

$$\pi(k) = \frac{c^k\times\pi(1)}{2k} \tag{21}$$

$$K = \frac{\pi(1)}{c} \tag{22}$$

from the model, we find $\pi = 0.6143$. and we get $c = \frac{0.6143}{100*0.01+0.6143} = 0.3805$. and $K = 1.6145$. Overall, it fits, but there are more groups with size 2-10 appear in the model than from master equation.

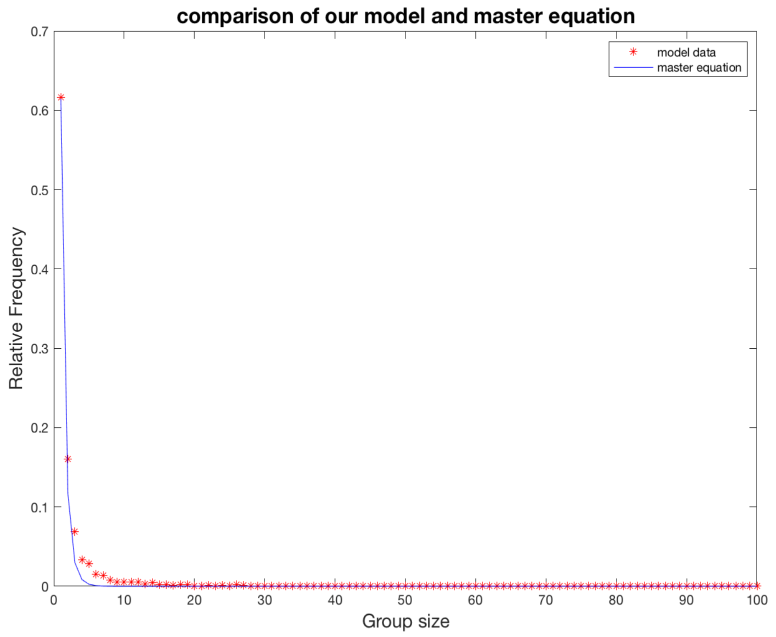

Figure 17: comparion of the distribution from simulation to master equation

3 Network Epidemics

3.1 random undirected social network

An undirected networking was made with 5000 students and a link density of 0.0016. we use matrix A to represent the network, as the network is undirected, we have A as a symmetric matrix, and with a link density of 0.0016, we generate A and plot the histogram of the degree distribution. The average degree of this network is around 8 and the degree follows a binomial distribution.

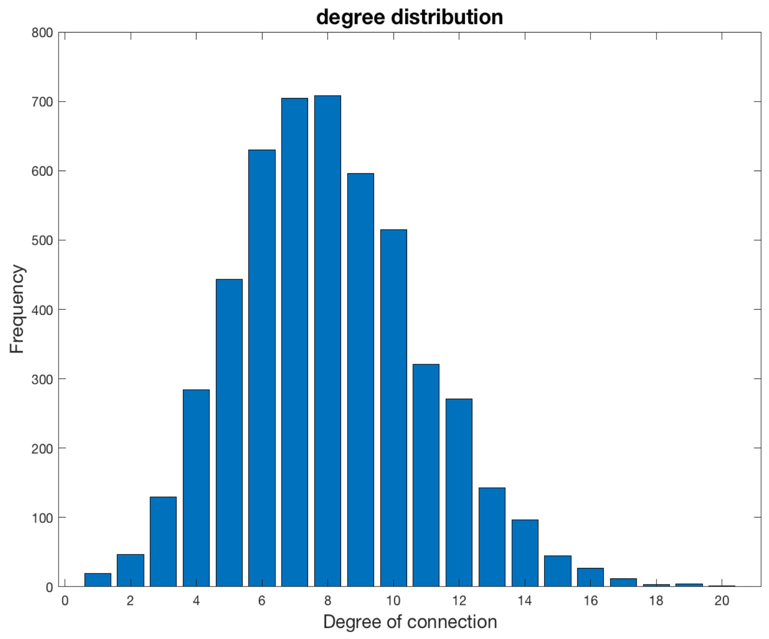

Figure 18: Frequency of the connection degree

3.2 Infection within the random network

A infection on above random network was modelled. In the beginning, there are 100 random infected students. The infection will spread according to:

1. An uninfected student will get infected depends on the infected student he is connected with. $P_{infected}(n) = 1 - e^{-pn}$.
2. An infected student will recover with the probability of r = 0.03.
3. A recovered student can get infected again.

To study the model, we set p = 0.01 and plot the number of infected students over time and it showed that the infection spreads very quick. At around 200 time, the total number of infected students are around 3000, and it stays stable around that number, which is more than half of the population. To further investigate, we run the model with p = 0.001:0.001:0.01, and the infection does not spread for $p \leq 0.003$, the infection will stay around the initial number when p = 0.004, and the infection will spread when $p \geq 0.005$, and the larger the p, the faster the spread of the infection and more infected students.

Finally, in Figure 21 we can see than the infection will die when $r/p \geq 10$ and spread very fast when $r/p \leq 5$.

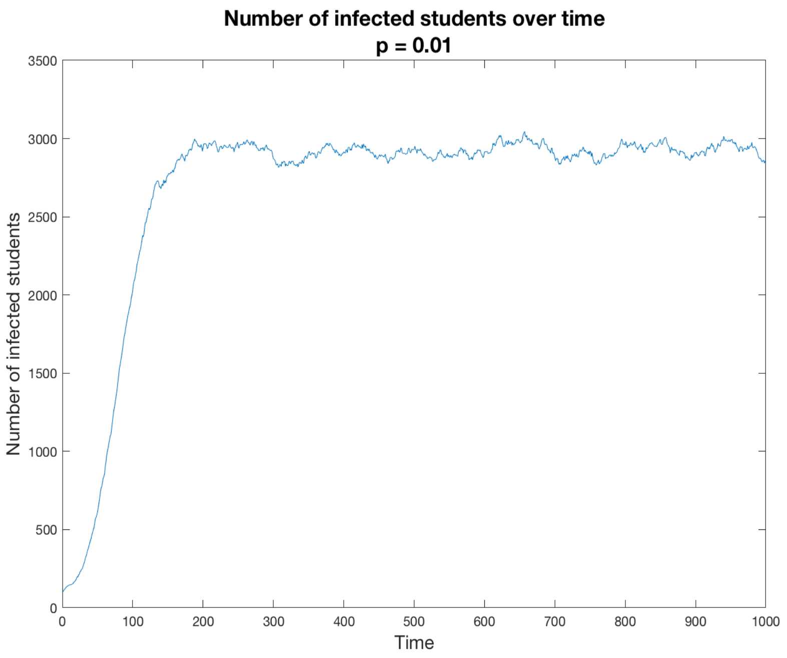

Figure 19: Number of infected individuals against time with p = 0.01

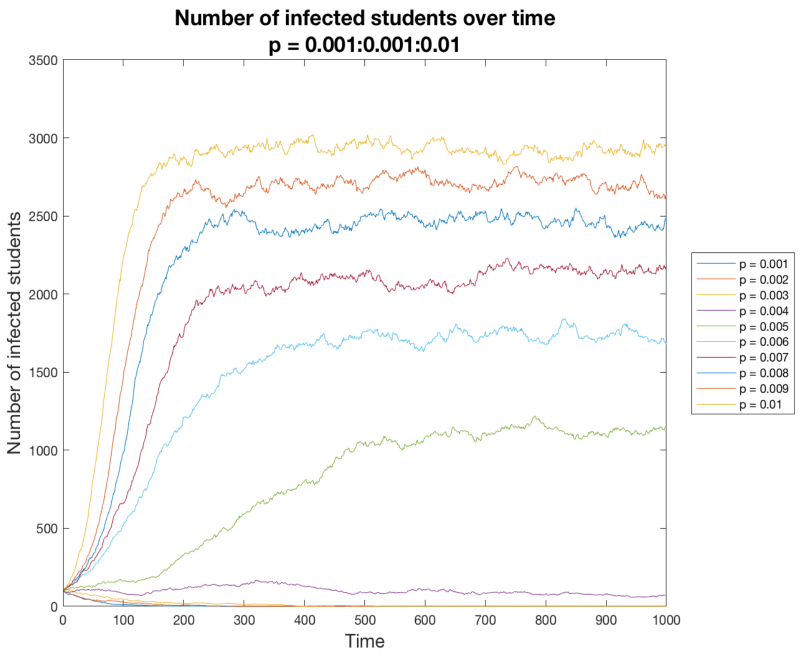

Figure 20: Number of infected individuals against time with different p

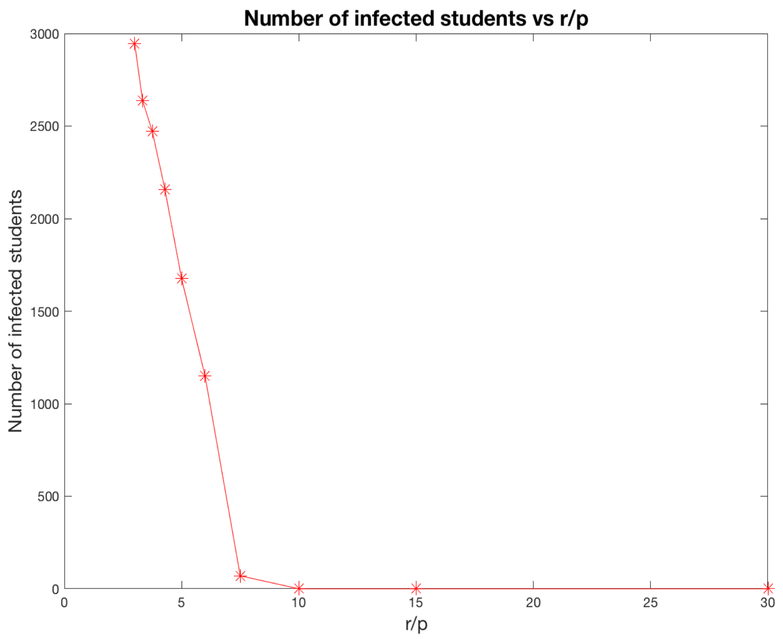

Figure 21: Number of infected individuals vs. r/p

3.3 Infection within the random network

In this part, we create a network with preferential attachment, starting at t_0, there are 2 students that are linked. Each time step, one new student is added to the network and he will chose to connect with a students at $p = \frac{k_i}{2(n-2)}$.

First, we can see that in Figure 22, the connection degree distribution follows the power law distribution. The average degree of the network is around 2. But to have a better idea, we take histogram at interval of 2^i. And in Figure 23, it eliminates the random noise, and shows a strong power law distribution for the preferential network.

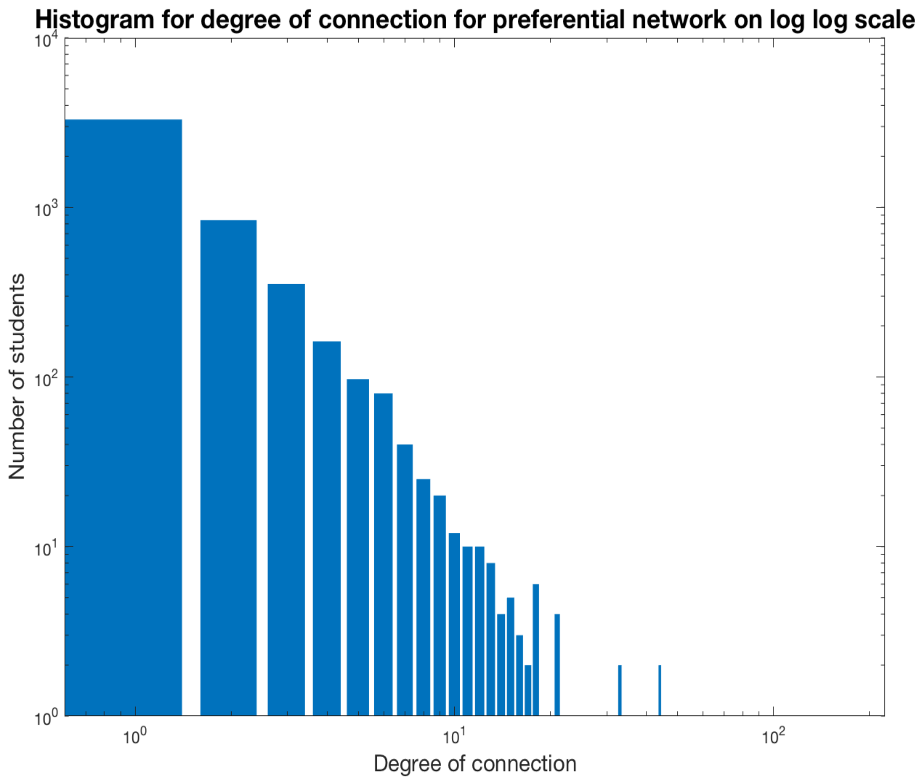

Figure 22: Histogram of the degree distribution

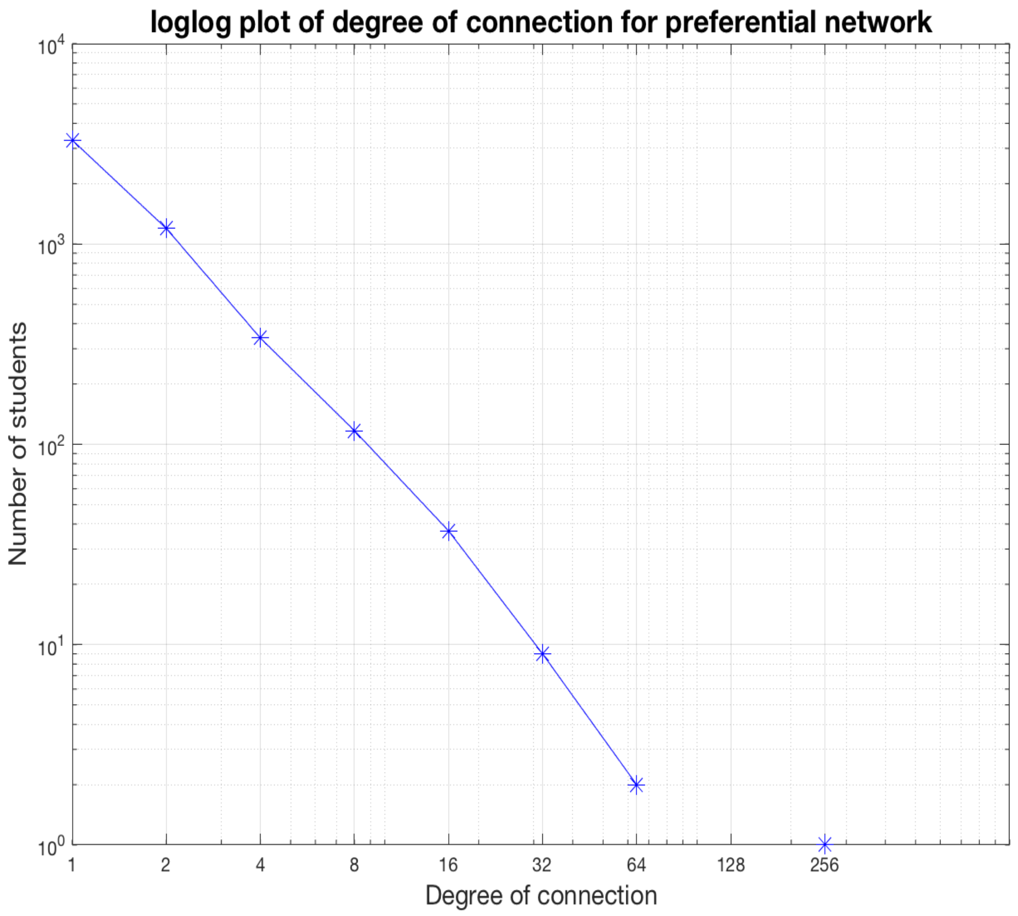

Figure 23: loglog plot of degree of connection

3.4 Master Equation

Below, I show my work for master equation.

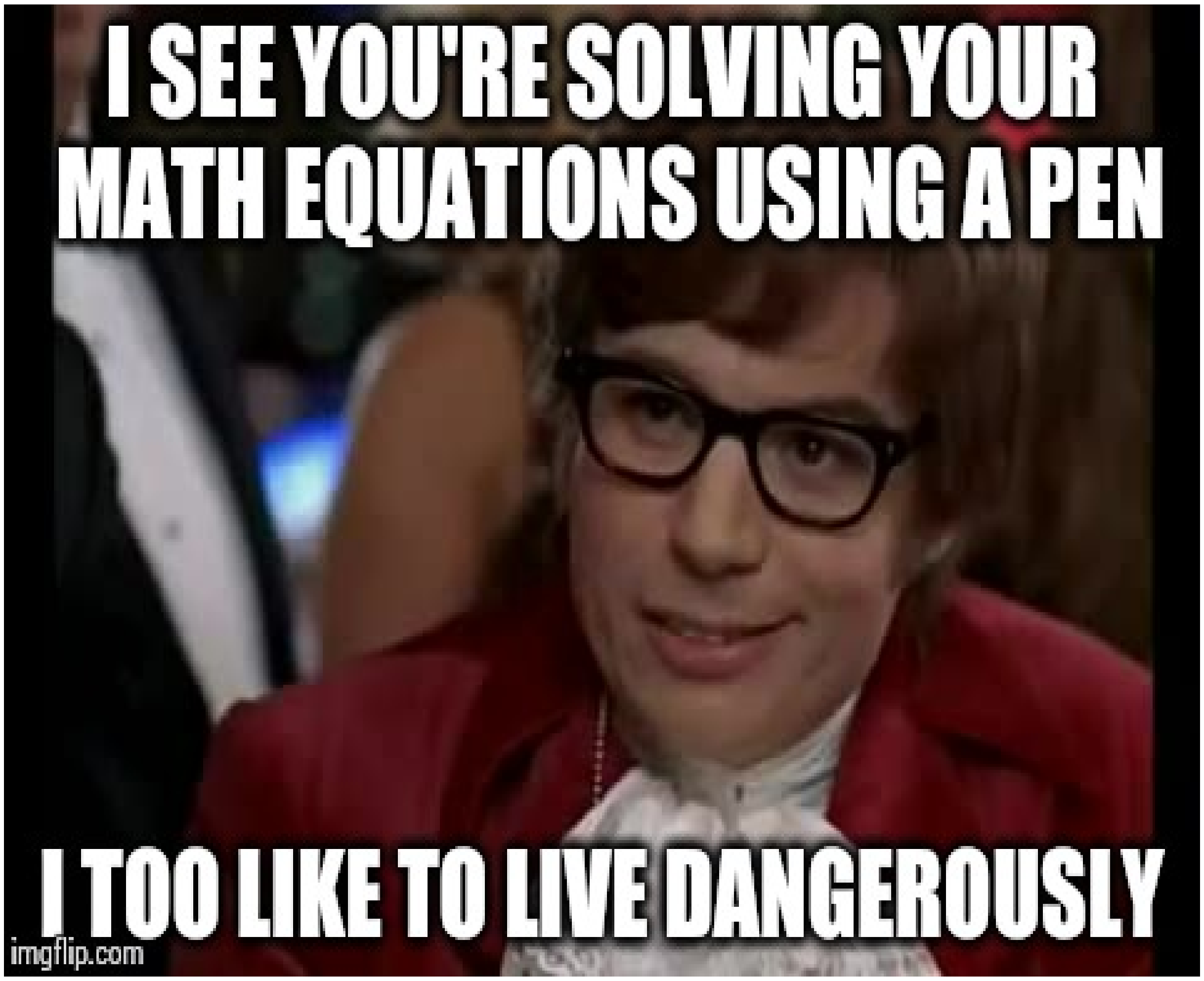
I SEE YOU'RE SOLVING YOUR MATH EQUATIONS USING A PEN
I TOO LIKE TO LIVE DANGEROUSLY
imgflip.com

Figure 24: master equation for the preferential network

For the preferential network, we have the number of students n and time t, and at each time the probability of student with k links will get an extra link is $p_{k,n} = \frac{k}{2(n-2)}$.

Expected number of student with k links are:

$$E = \frac{k}{2(n-2)} \times p_{k,n} \times n \tag{23}$$

$$(n+1) \times p_{k,n+1} = n \times p_{k,n} + \frac{k-1}{2(n-2)} \times p_{k-1,n} \times n - \frac{k}{2(n-2)} \times p_{k,n} \times n \tag{24}$$

to find steady state, we set n and t to ∞. we get:

$$(n+1) \times p_k = n \times p_k + \frac{k-1}{2} \times p_{k-1} - \frac{k}{2} \times p_{k,n} \tag{25}$$

$$p_k = \frac{1}{2} \times ((k-1) \times p_{k-1} - k \times p_k) \tag{26}$$

$$p_k = \frac{(k-1)!}{(k+2)(k+1)(k)...4} \times p_1 \tag{27}$$

when k = 1:

$$p_1 = 1 - \frac{1}{2} \times p_1 \tag{28}$$

$$p_1 = \frac{2}{3} \tag{29}$$

$$p_k = \frac{(k-1)!}{(k+2)(k+1)(k)...4} \times \frac{2}{3} = \frac{4}{(k+2)(k+1)(k)} \tag{30}$$

Then we got our result c = 4.

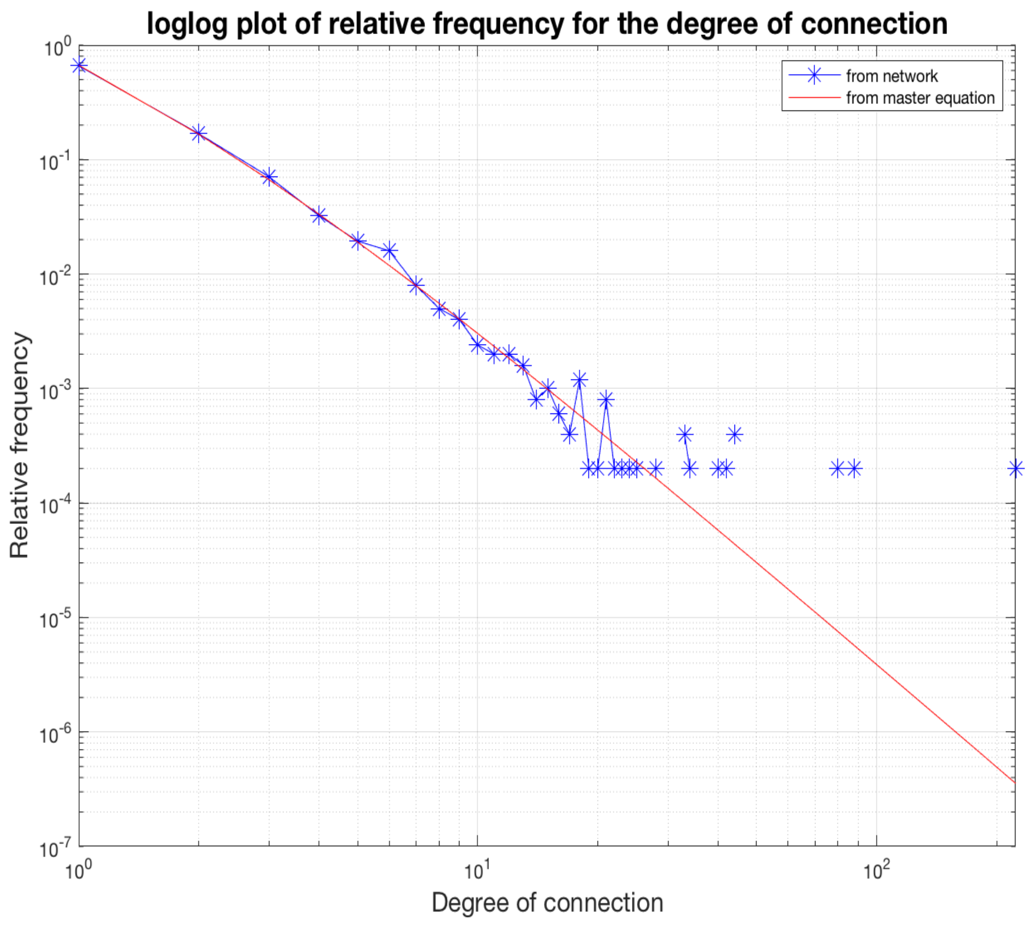

Figure 25: distribution of the connection from the network and master equation

The master equation aligns with the results from the network.

3.5 infection with the preferential attachment network

Here, the infection in 3.2 are simulated in the preferential network created in 3.3. We can see in Figure 26 that the number of infection will die when $p \leq 0.005$, the infected number will stay around the initial number or decrease a little when p is between 0.006 to 0.008, the infection will spread when $p \geq 0.009$.

The total number of infected students are less with same p in the preferential network compared to the random network as the average degree of network is lower at the preferential network. The infection are more difficult to spread in the preferential network than the random network. If a student with a lot of connections are infected then the infection will likely spread. But in a network with high r/p, the infected ones may recover sooner than spreading it to other students.

Using the spread of internet memes within the preferential network. the constant p here can be seen as how funny the memes is. If the memes is very funny, then the student will be very likely to hear from another students, thus become infected with the memes and spread it, if a key student with lots of connections become infected with the meme, the memes will take off and become viral. However we have a r = 0.03 recovery rate, can be seen as lost interest. While some students are sharing memes, some other students lost interests on the memes. When r/p ¿ 6, the memes are not likely to get spread out. When r/p are smaller than 4, the number of students that are interested in the memes are likely to remain the same as the initial number. If the r/p are decreased slightly under 4, the memes is likely to take off spreading very rapidly. However, this depends on the preferential network A. With the same rule in 3.3, sometimes the network can have key students with a large number of connections (100+), sometimes, the network does not have these big key student. It is suggested to run the model with different network A generated using the same methods in 3.3 to further investigate the preferential attachment network.

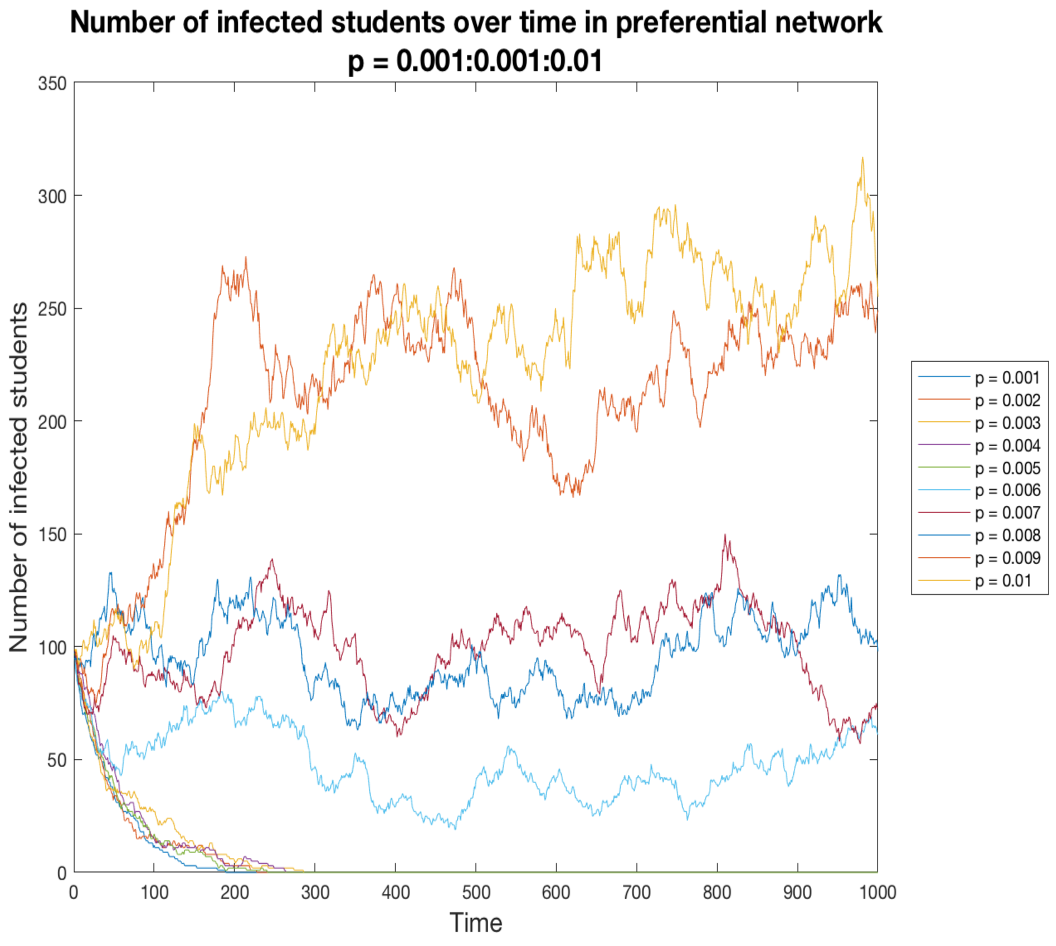

Figure 26: Number of infected individuals against time with different p

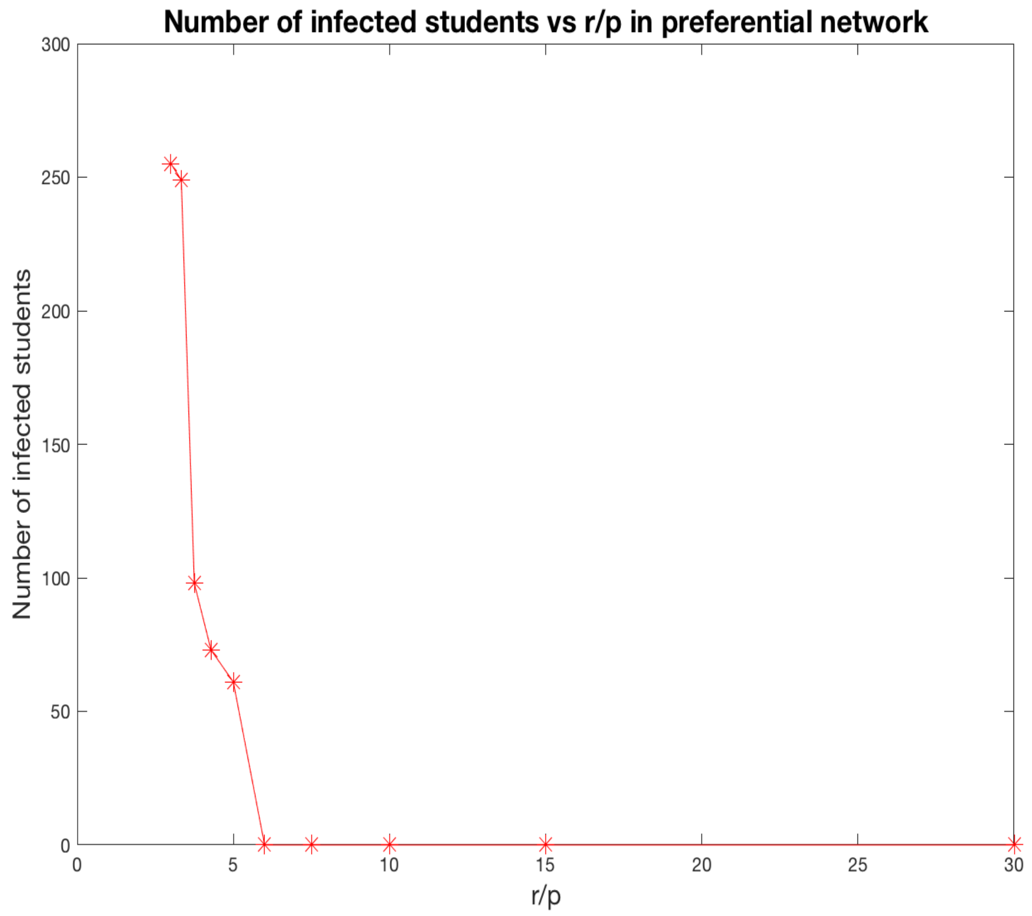

Figure 27: Number of infected individuals against time with different p

4 Flocks and Predators

4.1 Alignment

Run the Vicsek alignment model with 40 particles, domain size L = 10, angular noise e = 0.5, for 200 time steps, at each time, the direction of a particle is the average direction of its 4 nearest neighbours. The alignment varies from individual runs of the model, but generally, the particles all move towards the same direction very quick.

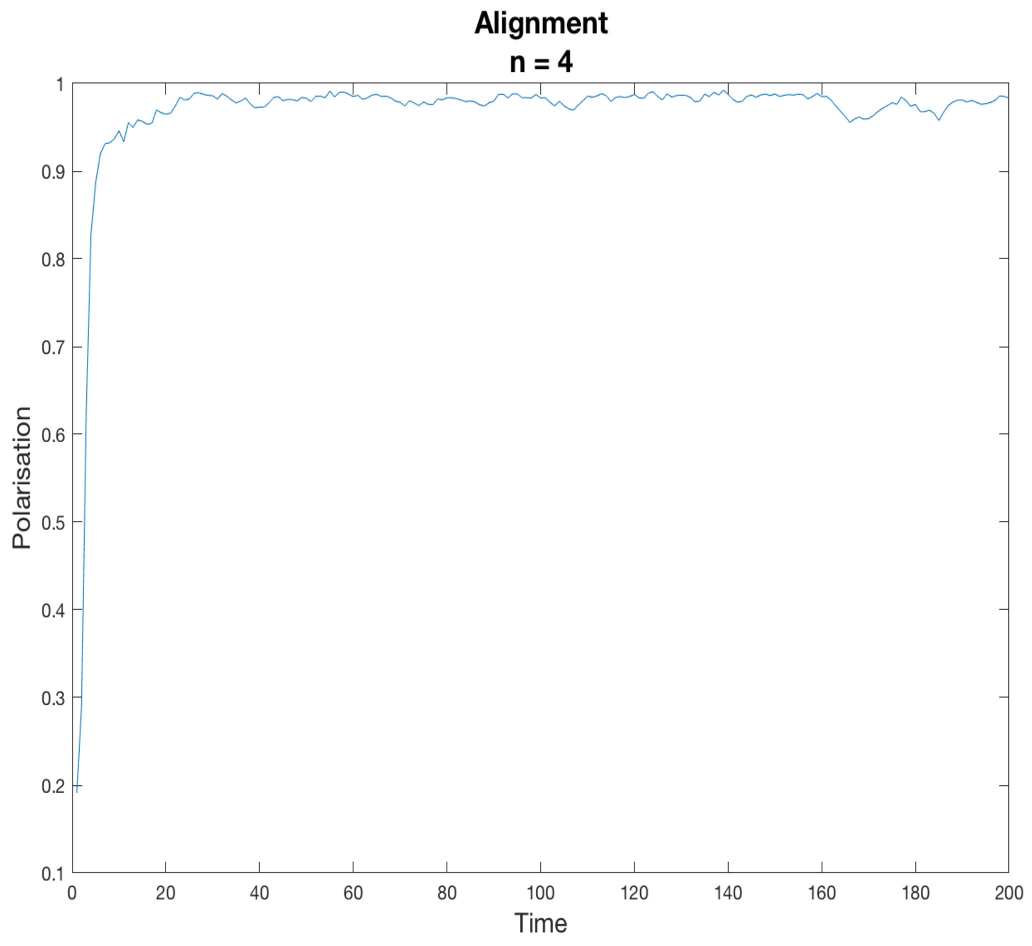

Figure 28: Alignment (4 nearest neighbour)

4.2 Add a small force

We add a small force that pulls the particles toward the center of mass. with coefficient 0.1. and we run the model with 40 particles, domain size L = 10, angular noise e = 0.1:0.1:6, nearest neighbour (n) = 1: 1:39 for 200 time steps with 10 replication. Then take average of the last half of the alignment. we get the heat map showing the steady-state alignment change with n and e.

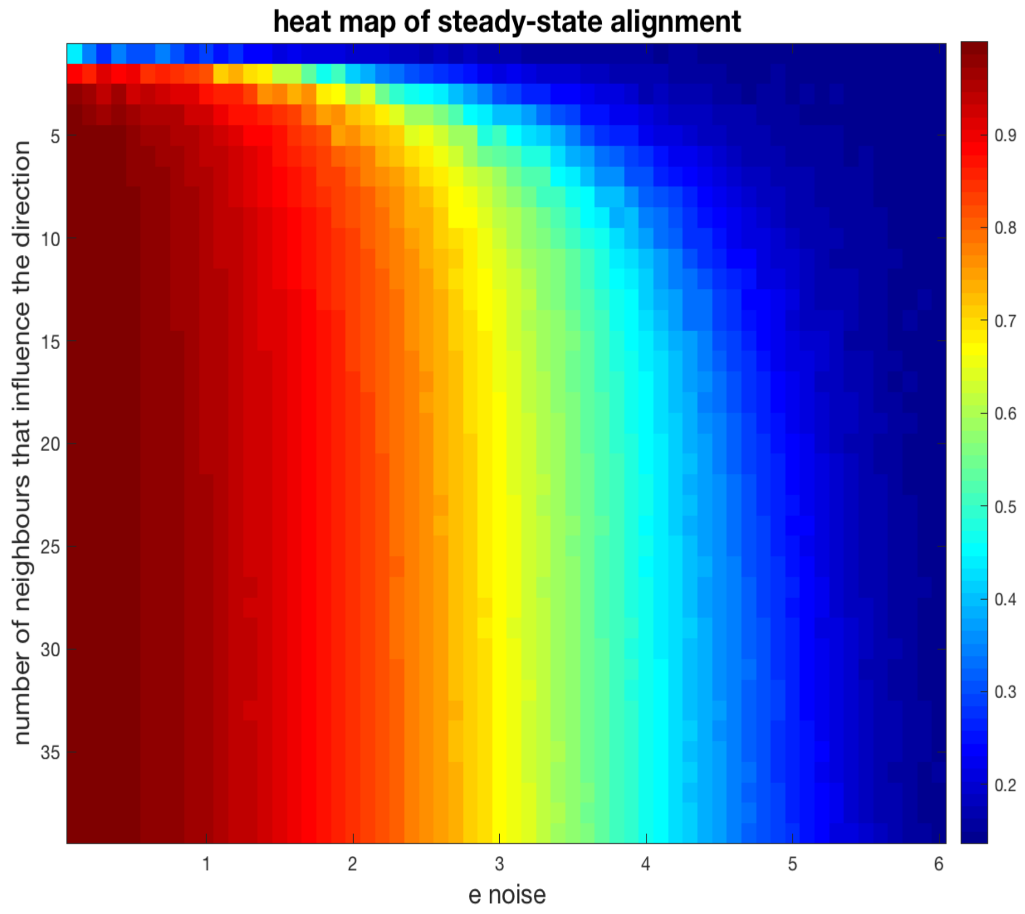

Figure 29: Heat map of steady state alignment

4.3 Predator

In this part, we will assign the particles to different settings. The 40 particles are divided into two groups, the first half of the group will move with direction based on large number of neighbours(k1), and the second half of the group will move with direction based on small number of neighbours(k2). The predator is attracted tot he centre of mass by his very accurate sensor and he will move to the centre of mass of the particles at a speed of 1.2 times the speed of the particles and it has a eating range of 0.5. Now we simulate this with different k1 and k2 and show the remaining number of particles at different times.

Unfortunately, there is no obvious clue that whether with a large n or small n will increase the survival rate. It depends on the initial location of particles, in the simulation, the particles and predator are all set to random location, and they all move at around the same speed. We only change the direction at each time step. Sometimes, the group with large n will be eaten faster, and sometimes the other way, depends on the initial location. There are more to do to simulate it to get a more realistic model, for example changing the speed of the particles, their reaction to predator, and predator's reaction to the particles, different initial locations, etc. In all, for my hunting model, there seems to be too much randomness in it, and the predator will move to the centre of the mass. There is no obvious strategy that can increase the survival. So it seems to be good just with a random walk strategy.

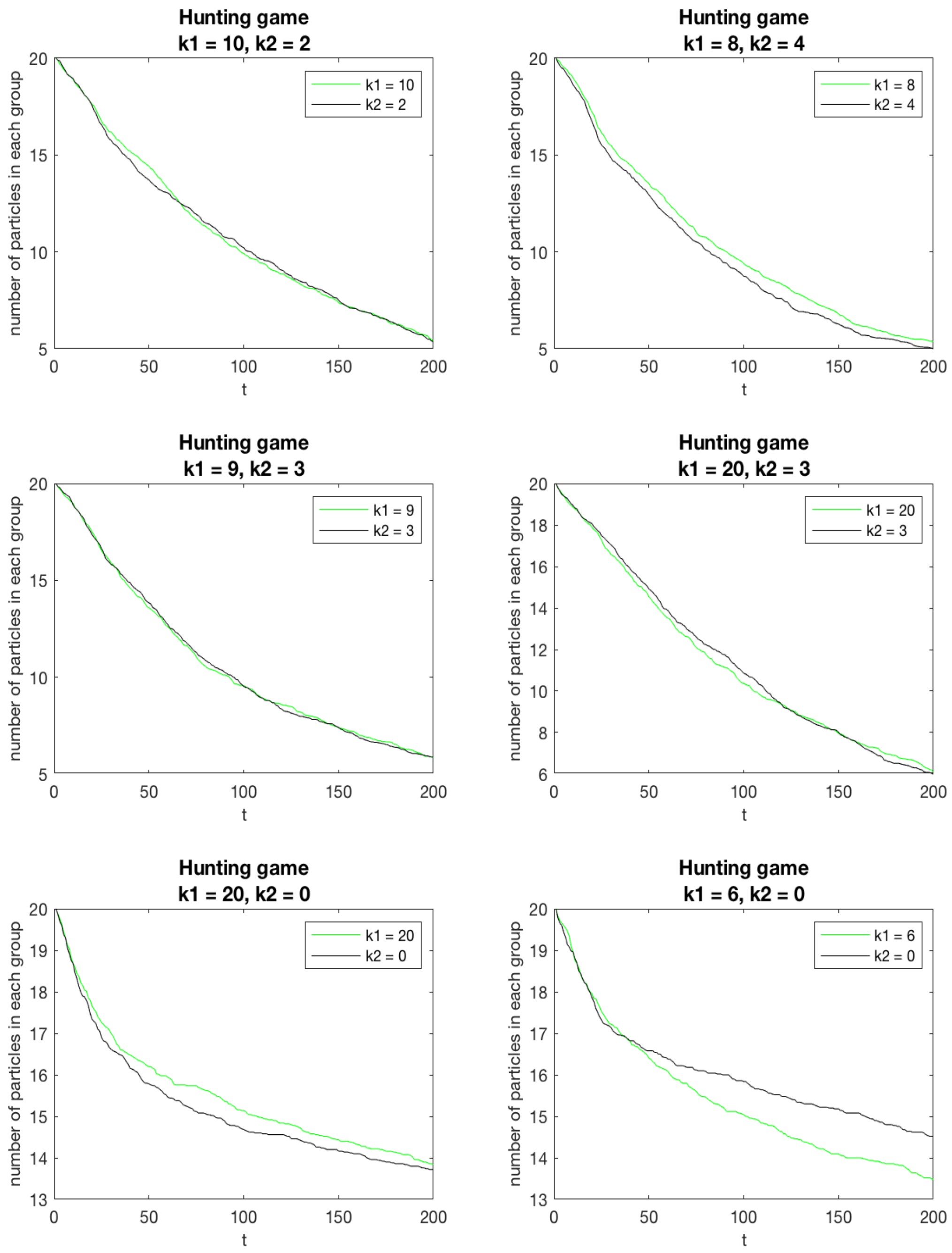

Figure 30: number of survivors with different neighbours influence the direction

Here are the videos you can watch for fun:

1. Hunting1 `<https://youtu.be/H3jBpG4da4s`
2. Hunting2 `<https://youtu.be/i7qd-9QbBCE>`
3. Hunting3 `<https://youtu.be/1Fe-Ej9SFXk>`
4. Hunting4 `<https://youtu.be/E3WqrBnrxUQ>`
5. Hunting5 `<https://youtu.be/0Oas2_Q7ef8>`

5 Appendix

5.1 Population Dynamics

- sim population

```
function y = sim_population(n,p,t,b)
%% this function simlate the population with specific n,
    population, time, offspring
% output the vector
% n- number of resource site
% p- population at t0
% t- simulate the time step t
% b- number of offspring per 2 individual
% generate nxt matrix, where each row is the resource
   site, and each colomn
% is each time
y = zeros(n,t);
y(:,1) = get_initial(n,p);

for t = 2:1:t
    y(:,t) = next_generation(y(:,t-1),n,b);
end
y = sum(y);

end
```

- get initial

```
function y = get_initial(n,p)
%% this function gives the generation 0's location
% input: n- number of resources
%        p- the total number of population at t0.

    y = zeros(n,1);
    for i = 1:1:p
        loc = randi(n);
        y(loc) = y(loc) + 1;
    end
end
```

- next generation

```
function y = next_generation(x,n,b)
%% this function gets the next_generation column vector
%x is the current population vector at each location
%n is the number of locations
%b is the number of offspring per 2 individual

y = zeros(n,1); % initialize our next generation vector

    for i = 1:1:n
        if x(i) == 2 % only care about if y(i) is
           exactly 2
            for j = 1:1:b
                loc = randi(n);%change the random number
                    generator
                y(loc) = y(loc) + 1;
            end
        end
    end

end

%{
tips from teacher
loc=ceil(rand(b,1)*n)
y(loc)=y(loc)+1
%}
```

- mean field model

```
clear all;

n = 1000;
a0 = n;

bval = [5, 10, 15, 20, 35, 50];


for b = bval
    A(b,1) = a0;

    for i = 2:1:1000

        A(b,i) = b*n*poisspdf(2,A(b,i-1)/n);
    end
end

%p = [5, 10, 15, 20, 35, 50];
figure2 = figure('position', [0, 0, 700, 500]);
subplot(3,2,1)
plot(A(5,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 5'}, '
   FontSize', 12)

subplot(3,2,2)
plot(A(10,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 10'}, '
   FontSize', 12)

subplot(3,2,3)
plot(A(15,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 15'}, '
   FontSize', 12)

subplot(3,2,4)
plot(A(20,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 20'}, '
   FontSize', 12)

subplot(3,2,5)
plot(A(35,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 35'}, '
   FontSize', 12)
```

```

subplot(3,2,6)
plot(A(50,:))
xlabel('t','FontSize',10)
ylabel('total population','FontSize',10)
title({'Mean-field model for population';'b = 50'}, '
   FontSize', 12)


saveas(figure2,'meanfield_model.png');
```

- phase transition

```
clear all

%enter b
Bvals = 1:1:50;

n = 1000; % number of resource sites
x0 = 100; %initial population number
t = 200;%simulate to t = 200
count = 0;
reps = 100;
hrange = 0:100:10000;

for b = Bvals
    yall = [];
    count = count + 1

    for i = 1:1:reps
        y = sim_population(n,x0,t,b);
        y_end = y(end);
        y_store = y((end/2+1):end);
        yall = [yall,y_store];
       %take the last half of the simulation
    end

    % take histogram
    histu(count,:) = hist(yall,hrange);
end

```

```

figure2 = figure('position', [0, 0, 700, 500]);
imagesc(Bvals,hrange,histu'/(reps*t/2), [0 0.1])
%normalize the display
colormap jet
colorbar
xlabel('b','FontSize',14)
ylabel('total population','FontSize',14)
title('population simulation phase transition', '
    FontSize', 16)
title(sprintf('population simulation phase transition
    numreps = %s for each b', string(reps)),'FontSize'
    ,16)


saveas(figure2,'q1_smallp.png');
```

- dfda

```
function y = dfda(n,b,a)
y = (b/(2*n)) * (2*a*exp(-a/n) - a*a*exp(-a/n)/n);
end
```

- lyapunov

```
clear all;
%need to compute average?
n = 1000; % number of resource sites
x0 = 1000; %initial population number
t = 500;%simulate to t = 500

% if lamda < 0 ,it converge
% if lamda > 0 ,it diverge


for b = 1:1:50
    b
    x = get_initial(n,x0);

    lamda(1) = abs(dfda(n,b,x0));

    for i = 2:1:t

        x = next_generation(x,n,b);
        a = sum(x);
        lamda(i) = log(abs(dfda(n,b,a)));
    end

    lya_exp(b) = (1/t)* sum(lamda);
end



figure2 = figure('position', [0, 0, 700, 500]);
plot([1:1:50],lya_exp)
xlim([0 50])
xlabel('b','FontSize',14)
ylabel('Lyapunov Exponent','FontSize',14)
title({'Lyapunov Exponent with different b'}, 'FontSize'
    , 16)
%saveas(figure2,'lyapunov_exp.png');
```

5.2 Groups of friends

- join group

```
function G = join_group(G,pick)
%% pick a person/group and join them all to another join
    a group
% p to join depends on groupsize = k/N

target = ceil(rand()*length(G));
P = cumsum(G);

k = find(P>=target,1);%find the index of the group to
    join

if k ~= pick %not join the existing group

```

```
        G(k) = G(k) + 1;%add one number
        G = G([1:pick-1 pick+1:end]);
else
        G = G;
end
```

- split group

```
function G = split_group(G,pick)
%% this function to get a group split up
k = G(pick);
G = G([1:pick-1 pick+1:end]);
add = ones(1,k);
G = [G add];
end
```

- sim friends

```
clear all;
close all;

N = 100; % initial number of individuals

G = ones(1,N);

r = 0.01; %p for split
ft = 1000;%final time step

result = [];
i = 0;
hrange = [];
hrange(1) = 1;
h = 0;

while h < 100
        i = i + 1;
        h = 2^i;
        hrange = [hrange h]; %for our histogram
end
```

```
ngroup = 0;

for j = 1:1:100
    j;

    for t = 1:1:ft

        pick = ceil(rand()*length(G));

        if G(pick) == 1
            G = join_group(G,pick);
        elseif rand() < r*G(pick)
            G = split_group(G,pick);
        end
        ngroup = ngroup + length(G);

        %storeh(t,:)=hist(G,[1:50])';
        %imagesc(storeh, [0 20])


    end
    result= [result G];%take the result when it is
        stable
end
histu = hist(result,hrange);
nhistu = histu/sum(histu); %frequency distribution

if nhistu(end) == 0
    nhistu = nhistu(1:end-1);
    hrange = hrange(1:end-1);
end


figure2 = figure('position', [0, 0, 700, 500]);
loglog(hrange,nhistu,'r*','markersize',20)
hold on
grid on
p = polyfit(log(hrange), log(nhistu), 1);
y_hat = exp(p(1) * log(hrange) + p(2));
loglog(hrange, y_hat,'r')
xlabel('Group size','FontSize',14)
set(gca, 'XTick', hrange)
set(gca, 'YTick', [0.0001 0.001 0.01 0.1 0.25 0.5 1])
ylabel('Relative Frequency','FontSize',14)
title({'log-log plot of group size vs relative frequency
    '}, 'FontSize', 16)
saveas(figure2,'loglog1.png');

hrange2 = 1:1:100;
histu2 = hist(result,hrange2);
nhistu2 = histu2/sum(histu2);

figure3 = figure('position', [0, 0, 700, 500]);
loglog(hrange2,histu2,'r','markersize',15)
hold on
grid on
%p = polyfit(log(hrange), log(nhistu), 1);
%y_hat = exp(p(1) * log(hrange) + p(2));
%loglog(hrange, y_hat,'r')
xlabel('Group size','FontSize',14)
set(gca, 'XTick', [hr])
%set(gca, 'YTick', [0.0001 0.001 0.01 0.1 0.25 0.5 1])
ylabel('Frequency','FontSize',14)
title({'log log plot of group size vs frequency'}, '
    FontSize', 16)
saveas(figure3,'loglog2.png');
```

- sim friends with replication

```
clear all;

N = 100; % initial number of individuals

G = ones(1,N);

r = 0.01;
ft = 1000;%final time step
hrange = 1:1:100; %for histogram
result = [];
gn = [];
reps = 100;

for j = 1:1:100
```

```
        j;

        for t = 1:1:ft

            pick = ceil(rand()*length(G));

            if G(pick) == 1
                G = join_group(G,pick);
            elseif rand() < r*G(pick)
                G = split_group(G,pick);
            end

        %storeh(t,:)=hist(G,[1:50])';
        %imagesc(storeh, [0 20])


        end
        result= [result G];
        gn = [gn length(G)];
end

histu = hist(result,hrange);
nhistu = histu/sum(histu);


    %store the final group distribution



histu = hist(result,hrange);
%loglog(histu);

K = 1.6145;
c = 0.3805;
y_predict =[];

for ii = hrange
    y_predict(ii) = (K * c^ii) /ii;
end

save('findpi2.mat')

```

```
figure2 = figure('position', [0, 0, 700, 500]);
plot(hrange,nhistu,'r*')
hold on

plot(y_predict,'b')
xlabel('Group size','FontSize',14)


ylabel('Relative Frequency','FontSize',14)
legend('model data','master equation');
title({'comparison of our model and master equation'}, '
   FontSize', 16)
saveas(figure2,'me1.png');
```

- sim friends with different r

```
clear all;

N = 100; % initial number of individuals

G = ones(1,N);

ft = 1000;%final time step
%hrange = 1:1:N; %for histogram
result = [];
reps = 100; %number of reps with each parameter
diffr = 0.001:0.001:0.1;
count = 0;
total=zeros(length(diffr),length(diffr)*reps);
hrange = [];
hrange(1) = 1;
h = 0;
ii = 0;
while h < 100
     ii = ii + 1;
     h = 2^ii;
     hrange = [hrange h]; %for our histogram
end


for r = diffr
```

```
    count = count + 1
    result = [];
    for re = 1:1:reps
        G = final_groups(G,ft,r);
        %this is the final group size distribution
         result = [result G];
    end
    histu(count,:) = hist(result,hrange);
    total(count,1:length(result)) = result;
   %take the final step histogram
end

histu_mean = histu/reps;
%take the nomalized?

loglog(histu_mean');

save('sim_friends_try3')
```

5.3 Network Epidemics

- random network

```
clear all

n = 5000;
A = zeros(n,n);
p = 0.0016;

for i = 1:1:n
    for j = i+1:1:n
        if rand < p
            A(i,j) = 1;
            A(j,i) = 1;
        end
    end
end

dir = sum(A);

```

```
range = 1:1:max(dir);
histu = hist(dir,range);
nhistu = histu/sum(histu);

figure2 = figure('position', [0, 0, 700, 500]);
bar(nhistu)

xlabel('Degree of connection','FontSize',14)
ylabel('Frequency','FontSize',14)
title({'degree distribution'}, 'FontSize', 16)
%saveas(figure2,'randomf2.png');




k = (1/n)*sum(dir);

%save('try1')
```

- infection transit

```
function y = infection_transit(A,b,n,p,r)
% A connection matrix
% b state vector (1 infected, 0 good)
% n number of population
% p constant related to infect
% r recover rate

y = zeros(n,1); %initial the next state
neighbours = zeros(n,1);
for i = 1:n
    if b(i) == 0
        neighbours(i) = A(i,:) * b;
        pi(i) = 1 - exp(-p*neighbours(i));
        if rand < pi(i)
            y(i) = 1;
        else
            y(i) = 0;
        end

    else
```

```
            if rand < r
                y(i) = 0;
            else
                y(i) = 1;
            end
        end
    end
end
```

- probability vector

```
function P = probability_vector(A,i)
    links = sum(A);
    links = links(1:i-1);

    for j = 1:1:i-1
        P(j) = links(j)/(2*(i-2));
    end
end
```

- sim infection

```
clear all

n = 5000; %number of people
A = zeros(n,n); %initial network
p = 0.0016; %connection probability

for i = 1:1:n
    for j = i+1:1:n
        if rand < p
            A(i,j) = 1;
            A(j,i) = 1;% create network one by one
        end
    end
end

infected_start = randperm(n,100); % the infected ones
infected_vector = zeros(n,1);
```

```
for i = 1:1:length(infected_start)
    infected_vector(infected_start(i)) = 1;
end
y0 = sum(infected_vector);
b0 = infected_vector;
ps = 0.001:0.001:0.01;
r = 0.03;
j = 0;

for pe = ps
    j = j+1
    infected_start = randperm(n,100); % the infected
       ones
    infected_vector = zeros(n,1);
    for i = 1:1:length(infected_start)
        infected_vector(infected_start(i)) = 1;
    end

    y0 = sum(infected_vector);
    b0 = infected_vector;

    for t = 1:1:1000
        b1 = infection_transit(A,b0,n,pe,r);
        y(j,t) = sum(b1);
        b0 = b1;
    end

end


save('resultp10run')

plot(y)
```

- preferential network

```
function A = p_network(n)
% generate the preferential network
A = zeros(n,n);
A(1,2) = 1;
A(2,1) = 1;
```

```
%link the first two students.

for i = 3:1:n
    links = sum(A);
    cum_links = cumsum(links);
    target = rand;
    p = probability_vector(A,i);
    tp = cumsum(p);
    k = find(tp>= target , 1);
    %found our link student
    A(k,i) = 1;
    A(i,k) = 1;
    %write our connect matrix
end
end
```

- preferential network

```
clear all
close all
n = 5000;
%{
s = randperm(n,2);

A = zeros(n,n);
A(s(1),s(2)) = 1;
A(s(2),s(1)) = 1;
%created the first link
%}
A = zeros(n,n);
A(1,2) = 1;
A(2,1) = 1;
%link the first two students.

for i = 3:1:n
    links = sum(A);
    cum_links = cumsum(links);
    target = rand;
    p = probability_vector(A,i);
    tp = cumsum(p);
    k = find(tp>= target , 1);
    %found our link student
    A(k,i) = 1;
    A(i,k) = 1;
    %write our connect matrix
end

degree_nw = sum(A);
max_degree = max(degree_nw);
hrange = 1:1:max_degree;

histu = hist(degree_nw,hrange);
bar(histu);
loglog(histu);

%average degreee of network

ave_degree = (1/(n))*sum(sum(A));
```

- preferential infection

```
clear all

n = 5000;
A = zeros(n,n);
A(1,2) = 1;
A(2,1) = 1;
%link the first two students.
for i = 3:1:n
    links = sum(A);
    cum_links = cumsum(links);
    target = rand;
    p = probability_vector(A,i);
    tp = cumsum(p);
    k = find(tp>= target , 1); %found our link student
    A(k,i) = 1;
    A(i,k) = 1; %write our connect matrix
end


ps = 0.001:0.001:0.01;
r = 0.03;
```

```
j = 0;

for pe = ps
    j = j+1
    infected_start = randperm(n,100); % the infected
       ones
    infected_vector = zeros(n,1);%initial a state vector
    for i = 1:1:length(infected_start)
        infected_vector(infected_start(i)) = 1;
        %write the infected student one by one
    end

    y0 = sum(infected_vector);
    b0 = infected_vector;

    for t = 1:1:1000
        b1 = infection_transit(A,b0,n,pe,r);
        y(j,t) = sum(b1);
        b0 = b1;
    end

end
plot(y)

%{
degree_nw = sum(A);
max_degree = max(degree_nw);
hrange = 1:1:max_degree;

histu = hist(degree_nw,hrange);
bar(histu);
loglog(histu);

%average degreee of network

ave_degree = (1/(n*n))*sum(sum(A));
%}
```

- full simulation for preferential network

```
clear all
```

```

% to simulate infection with preferential network
sim_t = 1; %want to get different network matrix A
n = 5000; %population
final_t = 5; % simulate for 1000 time steps

ps = 0.001:0.001:0.01;
y = zeros(length(ps), final_t);

for st = 1:1:sim_t
    st
   A = p_network(n);
    r = 0.03;
    j = 0;

    for pe = ps
        j = j+1
        infected_start = randperm(n,100); % the infected
            ones
        infected_vector = zeros(n,1);%initial a state
           vector
        for i = 1:1:length(infected_start)
            infected_vector(infected_start(i)) = 1;
           %write the infected student one by one
        end


        y0 = sum(infected_vector);
        b0 = infected_vector;

        for t = 1:1:final_t
            b1 = infection_transit(A,b0,n,pe,r);
            y(j,t) = y(j,t)+sum(b1);
            b0 = b1;
        end
    end
end


final_y = y/sim_t;
plot(final_y')
save('result_sim10_pn')
```

5.4 Flocks and Predators

- distance

```
function D = distance(x,y,t,N)
%compute the distance matrix at time t
D = zeros(N,N);
xx = x(:,t);
yy = y(:,t);
%extract the curent location

for i = 1:1:N
    for j = i+1:1:N
        D(i,j) = sqrt((xx(i)-xx(j))^2 + (yy(i) - yy(j))
            ^2);
        D(j,i) = D(i,j);


    end
end

end
```

- distance to predator

```
function D = distance_predator(x,y,t,N,px,py)
%compute the distance matrix at time t
D = [];
xx = x(:,t);
yy = y(:,t);
px = px(t);
py = py(t);
%extract the curent locations

for i = 1:1:N


        D(i) = sqrt((xx(i)-px)^2 + (yy(i) - py)^2);
        %if D(i,j) == NaN
         %   D(i,j) = 10000;
        %end
```

```


end

end
```

- alignment

```
clear
close all

%Set up movie
fig=figure;
makemovie=0;
%movien = avifile('Vicsekmovie','FPS',3,'compression','
    none')

J=200;
%Number of timestep t0 be used

UJ=1;
%Rate at which film is updated


t=1/J; %Size of one time step

N=40;
%Number of particles

e=0.5;
%e is eta the noise parameter, whose maximum value is 2*
    pi

r=1;
%The radius of influence of a particle

L=10;
%L is the size of the domain on which the particles can
    move

v=0.5; %v is the speed at which the particles move
```

```

% x(i,j) gives the x coordinate of the ith particle at time j
x=zeros(N,J+1);
x(:,1)=L*rand(N,1); %define initial x coordiantes of all particles

% y(i,j) gives the y coordinate of the ith particle at time j
y=zeros(N,J+1);
y(:,1)=L*rand(N,1); %define initial y coordiantes of all particles

% T(i,j) gives the angle with the x axis of the direction of motion of the ith
% particle at time j
T=zeros(N,J+1);
op = [];
T(:,1)=2*pi*rand(N,1); %define initial direction of all particles
k = 4;
ssin = sum(sin(T(:,1)));
scos = sum(cos(T(:,1)));
op(1) = (1/N) * sqrt(ssin^2 + scos^2);
%For all time steps
for j=1:1:J %iterate in time
    %For each particle
    D = distance(x,y,j,N);
    [B,I] = sort(D);
    near_n = I(2:k+1,:); %found our nearest 4 neighbours for each one

    for i=1:1:N

            %get the current distance vector
            neighbour = near_n(:,i);
            tn = []; % the neighbour direction
            for m = 1:1:length(neighbour)
                tn(m) = T(neighbour(m),j);
            end

            ss = sum(sin(tn));
            sc = sum(cos(tn));
            S = atan2(ss,sc);

            T(i,j+1)=S+e*(rand-0.5); %adds noise to the measured angle

            x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates the particles' x-coordinates
            y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates the particles' y coordinates

            % Jumps from the right of the box to the left or vice versa
            x(i,j+1)=mod(x(i,j+1),L);

            %Jumps from the top of the box to the bottom or vice versa
            y(i,j+1)=mod(y(i,j+1),L);

            %Plot particles

            if makemovie
                if abs(x(i,j)-x(i,j+1))<v & abs(y(i,j)-y(i,j+1))<v
                        plot([x(i,j), x(i,j+1)] ,[y(i,j),y(i,j+1)],'k-','markersize',4) %plots the first half of the particles in black
                    axis([0 L 0 L]);
                    hold on
                    plot(x(i,j+1) ,y(i,j+1),'k.','markersize',10)
                    xlabel('X position')
                    ylabel('Y position')

                end
            end

    end

    ssin = sum(sin(T(:,j+1)));
    scos = sum(cos(T(:,j+1)));
    op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);

    if makemovie
        hold off
        M(j)=getframe; %makes a movie fram from the plot

        %movien = addframe(movien,M(j)); %adds this
            movie fram to the movie
    end

end


%movien = close(movien); %finishes the movie
figure2 = figure('position', [0, 0, 700, 500]);
plot([1:1:J+1], op)
xlim([0 200]);
xlabel('Time','FontSize',14)
ylabel('Polarisation','FontSize',14)
title({'Alignment';'n = 4'}, 'FontSize', 16)
saveas(figure2,'palign.png');
```

- with small force

```
clear
close all

%Set up movie
fig=figure;
makemovie=1;
%movien = avifile('Vicsekmovie','FPS',3,'compression','
    none')

J=200;
%Number of timestep t0 be used

UJ=1;
%Rate at which film is updated


t=1/J; %Size of one time step
```

```

N=40;
%Number of particles
cforce = 0.1;
%the force that draw particles together
e=0.5;
%e is eta the noise parameter, whose maximum value is 2*
    pi

r=1;

%The radius of influence of a particle

L=10;
%L is the size of the domain on which the particles can
    move

v=0.5; %v is the speed at which the particles move

% x(i,j) gives the x coordinate of the ith particle at
    time j
x=zeros(N,J+1);
x(:,1)=L*rand(N,1); %define initial x coordiantes of all
     particles

% y(i,j) gives the y coordinate of the ith particle at
    time j
y=zeros(N,J+1);
y(:,1)=L*rand(N,1); %define initial y coordiantes of all
     particles

% T(i,j) gives the angle with the x axis of the
    direction of motion of the ith
% particle at time j
T=zeros(N,J+1);
T(:,1)=2*pi*rand(N,1); %define initial direction of all
    particles
k = 4;
%For all time steps
for j=1:1:J %iterate in time
    %For each particle
    D = distance(x,y,j,N);
    [B,I] = sort(D);
    near_n = I(2:k+1,:); %found our nearest 4 neighbours
        for each one
    xc = mean(x(:,j)); %center of mass x
    yc = mean(y(:,j)); %center of mass y

    for i=1:1:N

            %get the current distance vector
            neighbour = near_n(:,i);
            tn = []; % the neighbour direction
            for m = 1:1:length(neighbour)
                tn(m) = T(neighbour(m),j);
            end

            fcenter = atan2(yc-y(i,j),xc-x(i,j));
            %location of center
            ss = sum(sin(tn)) + sin(fcenter)*cforce;
            sc = sum(cos(tn)) + cos(fcenter)*cforce;

            S = atan2(ss,sc);



            T(i,j+1)=S+e*(rand-0.5); %adds noise to the
               measured angle

            x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
               the particles' x-coordinates
            y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
               the particles' y coordinates

            % Jumps from the right of the box to the
               left or vice versa
            x(i,j+1)=mod(x(i,j+1),L);

            %Jumps from the top of the box to the bottom
                or vice versa
            y(i,j+1)=mod(y(i,j+1),L);

            %Plot particles

```

```
                if makemovie
                    if abs(x(i,j)-x(i,j+1))<v & abs(y(i,j)-y(i,j+1))<v
                        plot([x(i,j), x(i,j+1)] ,[y(i,j),y(i,j+1)],'k-','markersize',4) %plots the first half of the particles in black
                    axis([0 L 0 L]);
                    hold on
                    plot(x(i,j+1) ,y(i,j+1),'k.','markersize',10)
                    xlabel('X position')
                    ylabel('Y position')

                    end
                end

    end
    if makemovie
        hold off
        M(j)=getframe; %makes a movie fram from the plot

        %movien = addframe(movien,M(j)); %adds this movie fram to the movie
    end

end


%movien = close(movien); %finishes the movie
```

- polarization

```
function [x,y,T,op] = polarization(J,N,e,L,movie,k,cf)
%% this function simulate the particle movement
% J(200) total time steps 200
% N(40) Number of particles 40
% e(0.5) is eta the noise parameter, whose maximum value is 2*pi, 0.5
% L(10) is the size of the domain on which the particles can move
% movie(1 or 0) to make movie or not
% k(4) is the influence neighbours
% cf is the force effect towards the center of mass


%Set up movie
%fig=figure;
makemovie = movie;
%movien = avifile('Vicsekmovie','FPS',3,'compression','
   none')
UJ=1;
%Rate at which film is updated

t=1/J; %Size of one time step


r=1;%The radius of influence of a particle

v=0.5; %v is the speed at which the particles move

% x(i,j) gives the x coordinate of the ith particle at
   time j
x=zeros(N,J+1);
x(:,1)=L*rand(N,1); %define initial x coordiantes of all
    particles

% y(i,j) gives the y coordinate of the ith particle at
   time j
y=zeros(N,J+1);
y(:,1)=L*rand(N,1); %define initial y coordiantes of all
    particles

% T(i,j) gives the angle with the x axis of the
   direction of motion of the ith
% particle at time j
T=zeros(N,J+1);
op = [];
T(:,1)=2*pi*rand(N,1); %define initial direction of all
   particles

ssin = sum(sin(T(:,1)));
scos = sum(cos(T(:,1)));
```

```
op(1) = (1/N) * sqrt(ssin^2 + scos^2);
%For all time steps
for j=1:1:J %iterate in time
    %For each particle
    D = distance(x,y,j,N);
     [B,I] = sort(D);
     near_n = I(2:k+1,:); %found our nearest 4 neighbours for each one
     xc = mean(x(:,j)); %center of mass x
     yc = mean(y(:,j)); %center of mass y

     for i=1:1:N

            %get the current distance vector
            neighbour = near_n(:,i);
            tn = []; % the neighbour direction
            for m = 1:1:length(neighbour)
                tn(m) = T(neighbour(m),j);
            end


            fcenter = atan2(yc-y(i,j),xc-x(i,j));
            %location of center
            ss = sum(sin(tn)) + sin(fcenter)*cf;
            sc = sum(cos(tn)) + cos(fcenter)*cf;

            S = atan2(ss,sc);

            T(i,j+1)=S+e*(rand-0.5); %adds noise to the measured angle

            x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates the particles' x-coordinates
            y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates the particles' y coordinates

            % Jumps from the right of the box to the left or vice versa
            x(i,j+1)=mod(x(i,j+1),L);

            %Jumps from the top of the box to the bottom or vice versa
            y(i,j+1)=mod(y(i,j+1),L);

            %Plot particles

            if makemovie
                if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-y(i,j+1))<v
                        plot([x(i,j), x(i,j+1)] ,[y(i,j),y(i,j+1)],'k-','markersize',4) %plots the first half of the particles in black
                    axis([0 L 0 L]);
                    hold on
                    plot(x(i,j+1) ,y(i,j+1),'k.','markersize',10)
                    xlabel('X position')
                    ylabel('Y position')

                end
            end

    end

    ssin = sum(sin(T(:,j+1)));
    scos = sum(cos(T(:,j+1)));
    op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);

    if makemovie
        hold off
        M(j)=getframe; %makes a movie fram from the plot

        %movien = addframe(movien,M(j)); %adds this movie fram to the movie
    end

end

%plot([1:1:J+1], op)

%movien = close(movien); %finishes the movie
end
```

- steady state alignment

```
clear
close all

N = 40;
J = 200;
L = 10;
movie = 0;
cf = 0.1;
erange = 0.1:0.1:6;
reps = 10;


for k = 1:1:N-1
    k
    yy = 0;
    for e = erange
        yy = yy+1;
        steady_a(k,yy) = 0;
        for r = 1:1:reps

            [x,y,T,op] = polarization(J,N,e,L,movie,k,cf
               );
            sizeop = length(op);
            ssalign = mean(op(round(sizeop/2):end));
            steady_a(k,yy) = steady_a(k,yy) + ssalign;
        end
        steady_a(k,yy) = steady_a(k,yy)/reps;
    end
end


save('ssatry2.mat')
```

- hunting function

```

function [x,y,T,op,s1,s2] = test43_hunt(J,N,e,L,movie,k1
   ,k2,cf)
%% this function simulate the particle movement
% J(200) total time steps 200
% N(40) Number of particles 40
% e(0.5) is eta the noise parameter, whose maximum value
    is 2*pi, 0.5
% L(10) is the size of the domain on which the particles
    can move
% movie(1 or 0) to make movie or not
% k1 is the influence neighbours for the first group
% k2 is the influence neighbours for the second group
% cf is the force effect towards the center of mass
%J=200;N=40;e=0.5;L=10;movie=1;k1=10;k2=3;cf=0;
%Set up movie

figure('position', [0, 0, 700, 600])
makemovie = movie;
if makemovie == 1
myv = VideoWriter('hunting5.avi');
myv.FrameRate = 10;
open(myv);
set(gca,'nextplot','replacechildren');
end


%movien = avifile('Vicsekmovie','FPS',3,'compression','
    none')
UJ=1;
%Rate at which film is updated

t=1/J; %Size of one time step

r=0.5;%The radius of eating
fast = 1.2;

v=0.5; %v is the speed at which the particles move
vp = v * fast;

% x(i,j) gives the x coordinate of the ith particle at
    time j
x=zeros(N,J+1);
x(:,1)=L*rand(N,1); %define initial x coordiantes of all
    particles
px = [];
px(1) = L*rand();%predator x
```

```

% y(i,j) gives the y coordinate of the ith particle at
   time j
y=zeros(N,J+1);
y(:,1)=L*rand(N,1); %define initial y coordiantes of all
    particles
py = [];
py(1) = L*rand(); %predator y

% T(i,j) gives the angle with the x axis of the
   direction of motion of the ith
% particle at time j
T=zeros(N,J+1);
op = [];
T(:,1)=2*pi*rand(N,1); %define initial direction of all
   particles
dirp = [];
dirp(1) = 2*pi*rand();

ssin = sum(sin(T(:,1)));
scos = sum(cos(T(:,1)));
op(1) = (1/N) * sqrt(ssin^2 + scos^2);
%For all time steps
for j=1:1:J %iterate in time
    %For each particle
    D = distance(x,y,j,N);
    [B,I] = sort(D);
    big_n = I(2:k1+1,:); %found our nearest 10
       neighbours for each one
    small_n = I(2:k2+1,:); %found our nearest 3
       neighbours for each one
    xc = nanmean(x(:,j)); %center of mass x
    yc = nanmean(y(:,j)); %center of mass y

    dirp(j+1) = atan2(yc - py(j), xc - px(j));

    px(j+1) = px(j) + vp * cos(dirp(j+1));
    py(j+1) = py(j) + vp * sin(dirp(j+1));

    for i=1:1:round(N/2)
        %the group that depends on larger number of
           neighbours
```

```

            %get the current distance vector
            neighbour = big_n(:,i);
            tn = []; % the neighbour direction
            for m = 1:1:length(neighbour)
                tn(m) = T(neighbour(m),j);
            end


            fcenter = 0;
            %location of center
            ss = nansum(sin(tn)) + sin(fcenter)*cf;
            sc = nansum(cos(tn)) + cos(fcenter)*cf;

            S = atan2(ss,sc);

            T(i,j+1)=S+e*(rand-0.5); %adds noise to the
               measured angle

            x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
               the particles' x-coordinates
            y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
               the particles' y coordinates

            % Jumps from the right of the box to the
               left or vice versa
            x(i,j+1)=mod(x(i,j+1),L);

            %Jumps from the top of the box to the bottom
                or vice versa
            y(i,j+1)=mod(y(i,j+1),L);
    end



    for i=round(N/2)+1:1:N
        %the group that depends on small number of
           neighbours

            %get the current distance vector
            neighbour = small_n(:,i);
            tn = []; % the neighbour direction
            for m = 1:1:length(neighbour)
                tn(m) = T(neighbour(m),j);
            end



            fcenter = atan2(yc-y(i,j),xc-x(i,j));
            fcenter = 0;
            %location of center
            ss = nansum(sin(tn)) + sin(fcenter)*cf;
            sc = nansum(cos(tn)) + cos(fcenter)*cf;

            S = atan2(ss,sc);

            T(i,j+1)=S+e*(rand-0.5); %adds noise to the
               measured angle

            x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
               the particles' x-coordinates
            y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
               the particles' y coordinates

            % Jumps from the right of the box to the
               left or vice versa
            x(i,j+1)=mod(x(i,j+1),L);

            %Jumps from the top of the box to the bottom
                or vice versa
            y(i,j+1)=mod(y(i,j+1),L);
    end

    %Plot particles
    Dp = distance_predator(x,y,j,N,px,py);

    for ii = 1:1:N

            if Dp(ii) < r
                x(ii,j+1) = NaN;
                y(ii,j+1) = NaN;
            end

    end


    if makemovie
        for i= 1:1:round(N/2)
            if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-y(i,j+1))<v
                        plot([x(i,j), x(i,j+1)] ,[y(i,j),y(i,j+1)],'g-','markersize',4) %plots the first half of the particles in black
                    axis([0 L 0 L]);
                    hold on
                    %plot([px(j) px(j+1)],[py(j) py(j+1)],'r-','markersize',4) %plot the predator
                    plot(x(i,j+1),y(i,j+1),'g.','markersize',10)
                    %plot(px(j+1),py(j+1),'rd','markersize',10)
                    xlim([0 L])
                    ylim([0 L])
                    xlabel('X position')
                    ylabel('Y position')
                    title({'hunting game';'Green: movement depends on 10 neighbours';'Black: movement depends on 3 neighbours';'Red:Predator'},'FontSize',16);

            end
        end

        for i = round(N/2)+1:1:N
            if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-y(i,j+1))<v
                        plot([x(i,j), x(i,j+1)] ,[y(i,j),y(i,j+1)],'k-','markersize',4) %plots the first half of the particles in black
                    axis([0 L 0 L]);

                    plot([px(j) px(j+1)],[py(j) py(j+1)
                        ],'r-','markersize',4) %plot the
                        predator
                    plot(x(i,j+1),y(i,j+1),'k.','
                        markersize',10)
                    plot(px(j+1),py(j+1),'rd','
                        markersize',10)
                    xlim([0 L])
                    ylim([0 L])
                    xlabel('X position')
                    ylabel('Y position')
                    title({'hunting game';'Green:
                        movement depends on 10 neighbours
                        ';'Black: movement depends on 3
                        neighbours';'Red:Predator'},'
                        FontSize',16);
                end
            end



    end

    %{
    ssin = sum(sin(T(:,j+1)));
    scos = sum(cos(T(:,j+1)));
    op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);
    %}
    if makemovie
        frame = getframe(gcf);
        writeVideo(myv,frame);

        hold off
        %pause(0.1)


        M(j)=getframe; %makes a movie fram from the plot

        %movien = addframe(movien,M(j)); %adds this
            movie fram to the movie
    end


```

```
end

if makemovie ==1
close(myv); %finishes the movie
end

for t = 1:1:J


    s1(t) = N/2 - sum(isnan(x(1:round(N/2),t)));
    s2(t) = N/2 - sum(isnan(x(round(N/2)+1:end,t)));

end

end
```

- add predator

```
clear
close all

J=200;N=40;e=0.5;L=10;movie=0;
k1=6;k2=0;cf=0;

for i = 1:1:50
[x,y,T,op,s1,s2] = test43_hunt(J,N,e,L,movie,k1,k2,cf);
ss1(i,:) = s1;
ss2(i,:) = s2;
end

save('run436.mat')
```