



# 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.

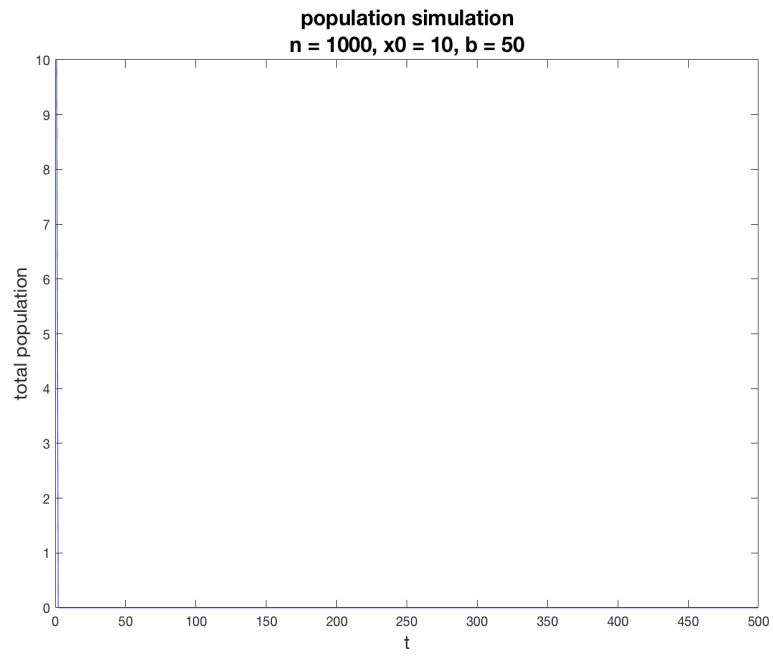


Figure 1: initial population of 10  $b = 50$  not reproducing

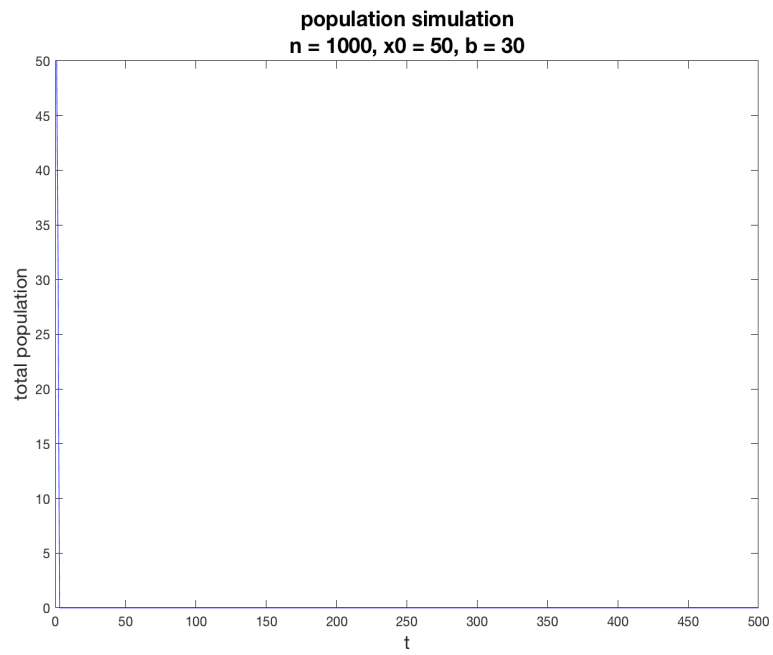


Figure 2: initial population of 50 and  $b = 30$  not reproducing

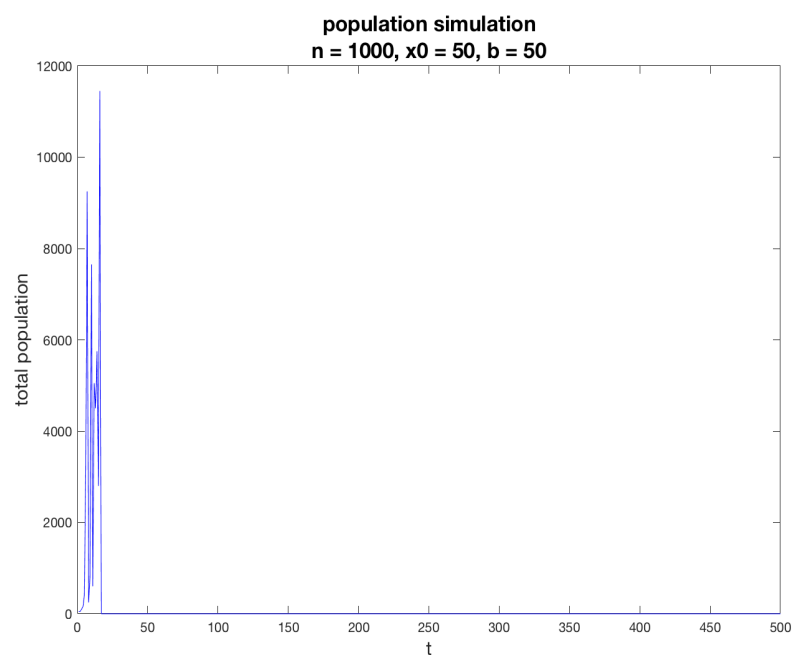


Figure 3: initial population of 50 and  $b = 50$ , in the beginning reproduce but quickly dies

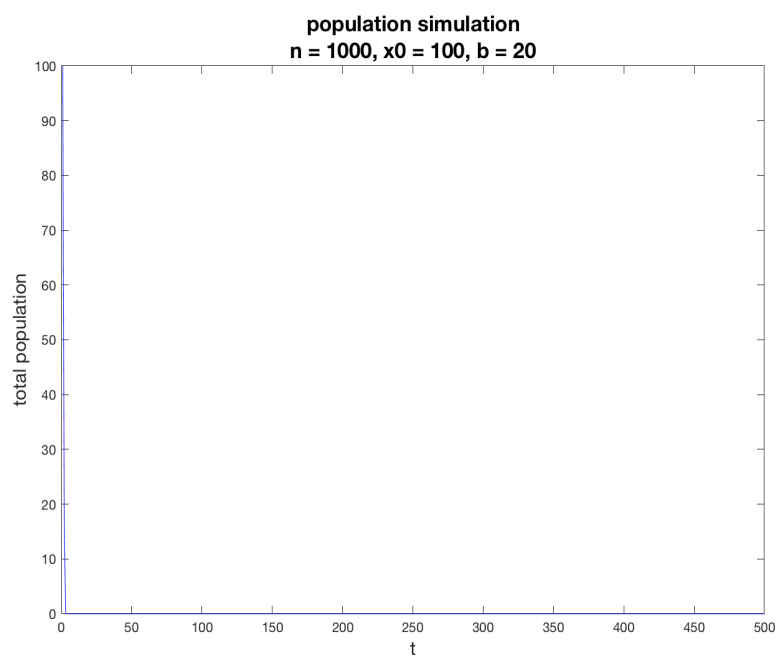


Figure 4: initial population of 100 and  $b = 20$ , it dies

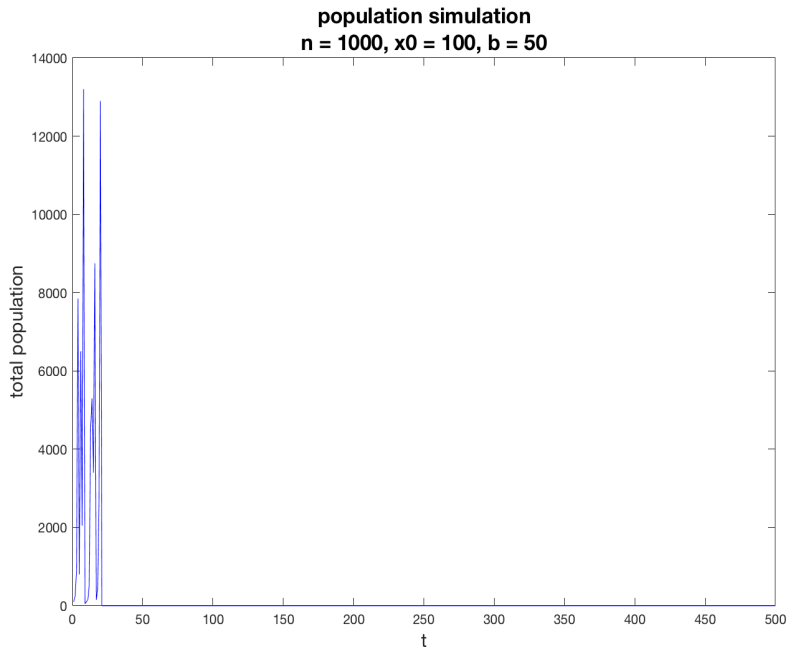


Figure 5: initial population of 100 and  $b = 50$ , it reproduce in the beginning but quickly dies

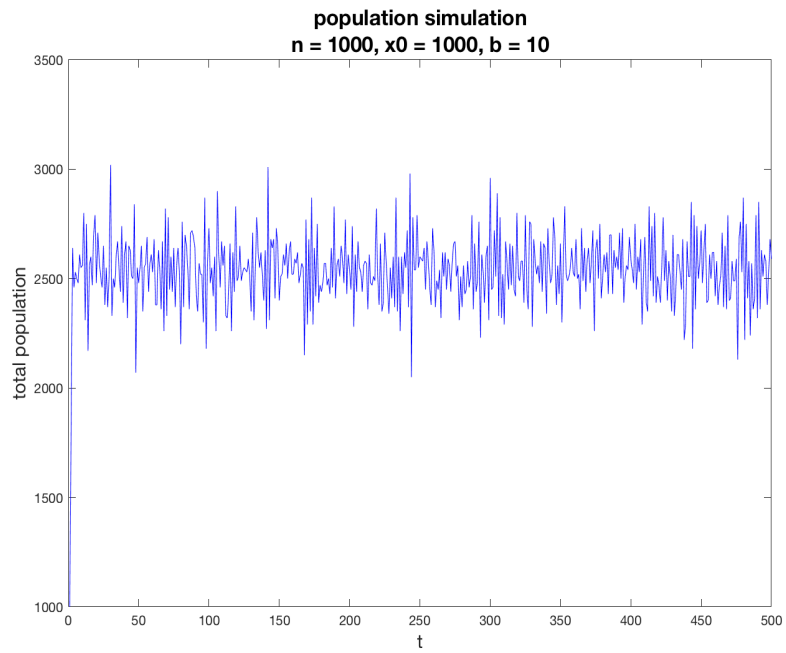


Figure 6: initial population of 1000 and  $b = 10$ , it show that the population oscillates around 2500

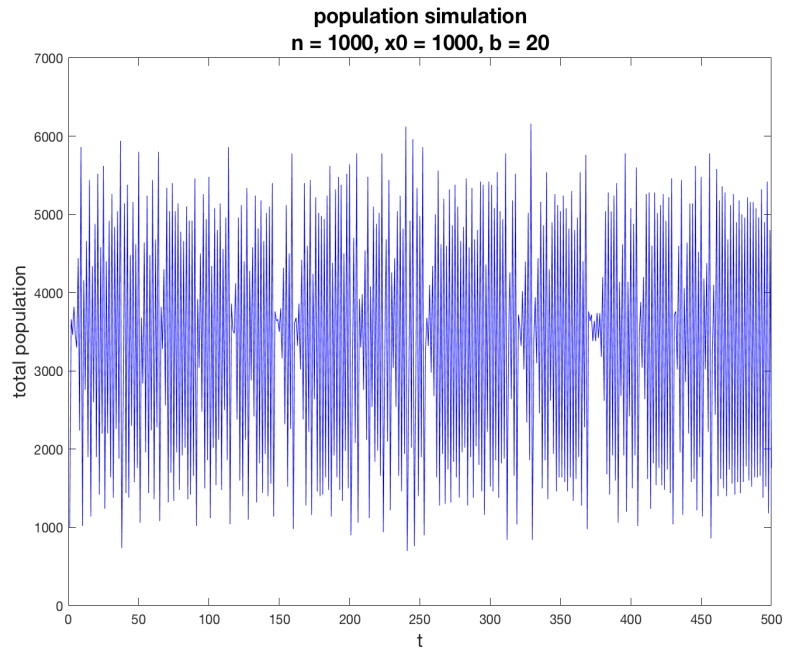


Figure 7: initial population of 1000 and  $b = 20$ , the populations starts to get chaotic

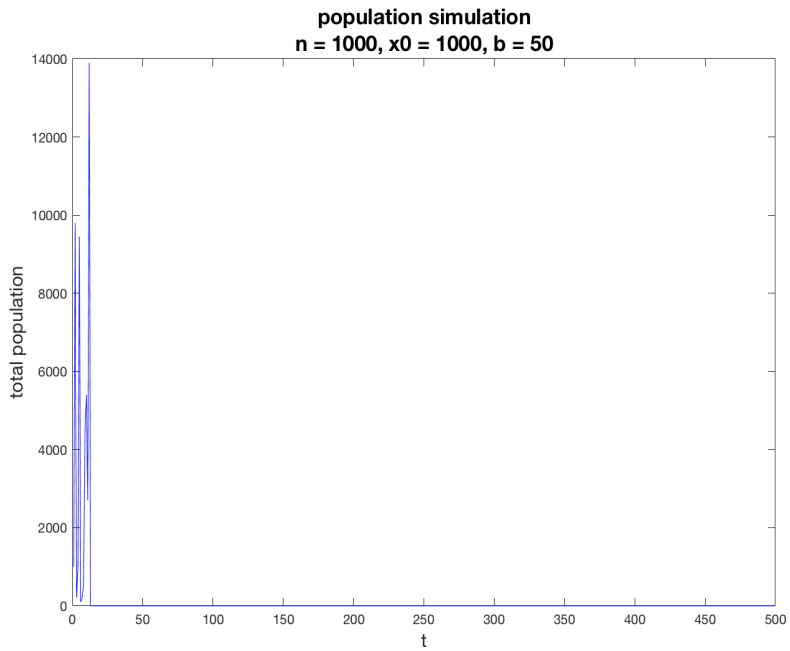


Figure 8: initial population of 1000 and  $b = 500$ , the populations dies very quick

To further study this model, we run it with  $A_0 = 1000$ ,  $n = 1000$ ,  $b = 1, 2, 3, 4, 5, \dots, 48, 49, 50$ . 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.

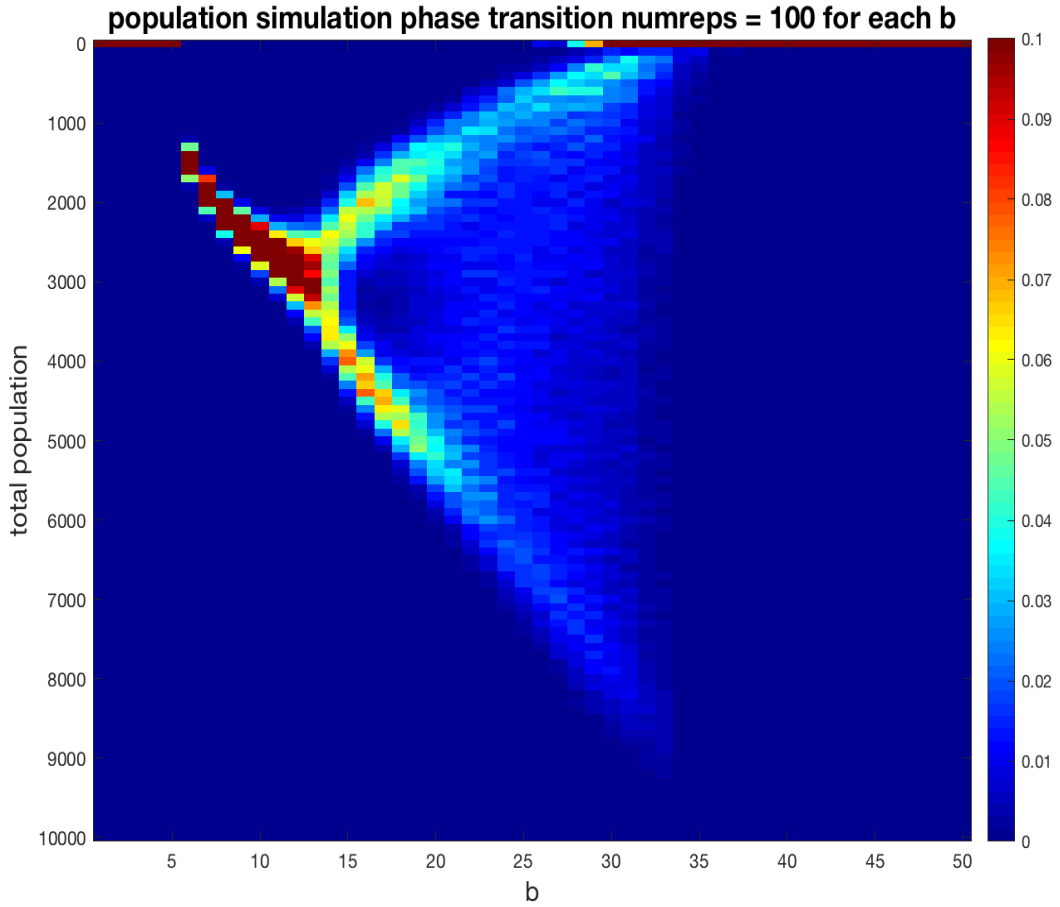


Figure 9: initial population of 1000 and  $b = 500$ , the populations dies very quick

## 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

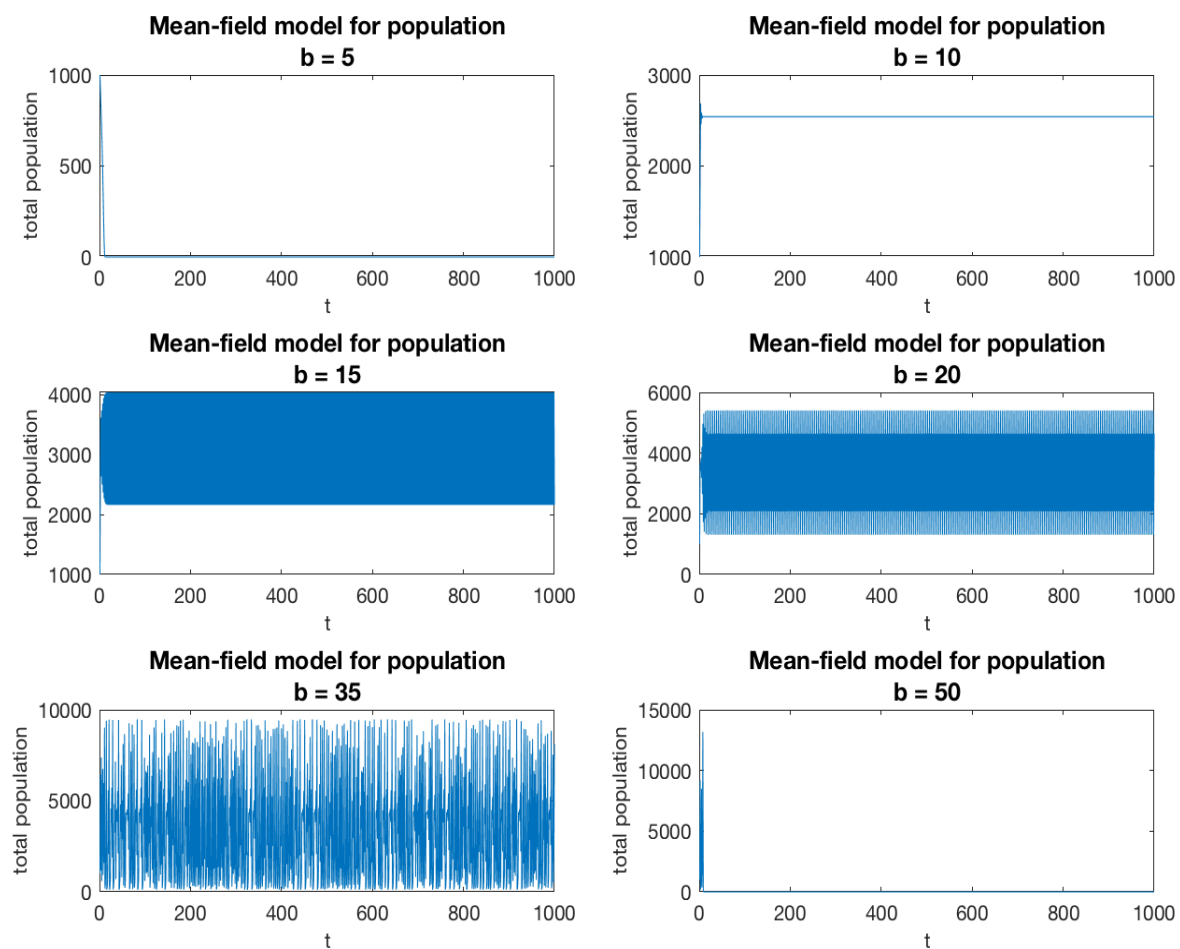


Figure 10: mean field model with selected  $b$  value



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

$$\phi_k = \begin{cases} b, if k = 2 \\ 0, if k \neq 2 \end{cases} \quad (2)$$

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

$$A_{t+1} = n \times P(2atsite, A_t) \times b \quad (3)$$

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

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

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

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

When  $A_t \neq 0$ , we have

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

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

$$1 = \frac{b}{2} \times x \times e^{-x} \quad (7)$$

$$x \times e^{-x} = \frac{b}{2} \quad (8)$$

To find the conditions in terms of  $b$  for the existence of two further non-zero steady states, we need to find that  $f_1(x) = x \times e^{-x}$  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  $b > 5$ .

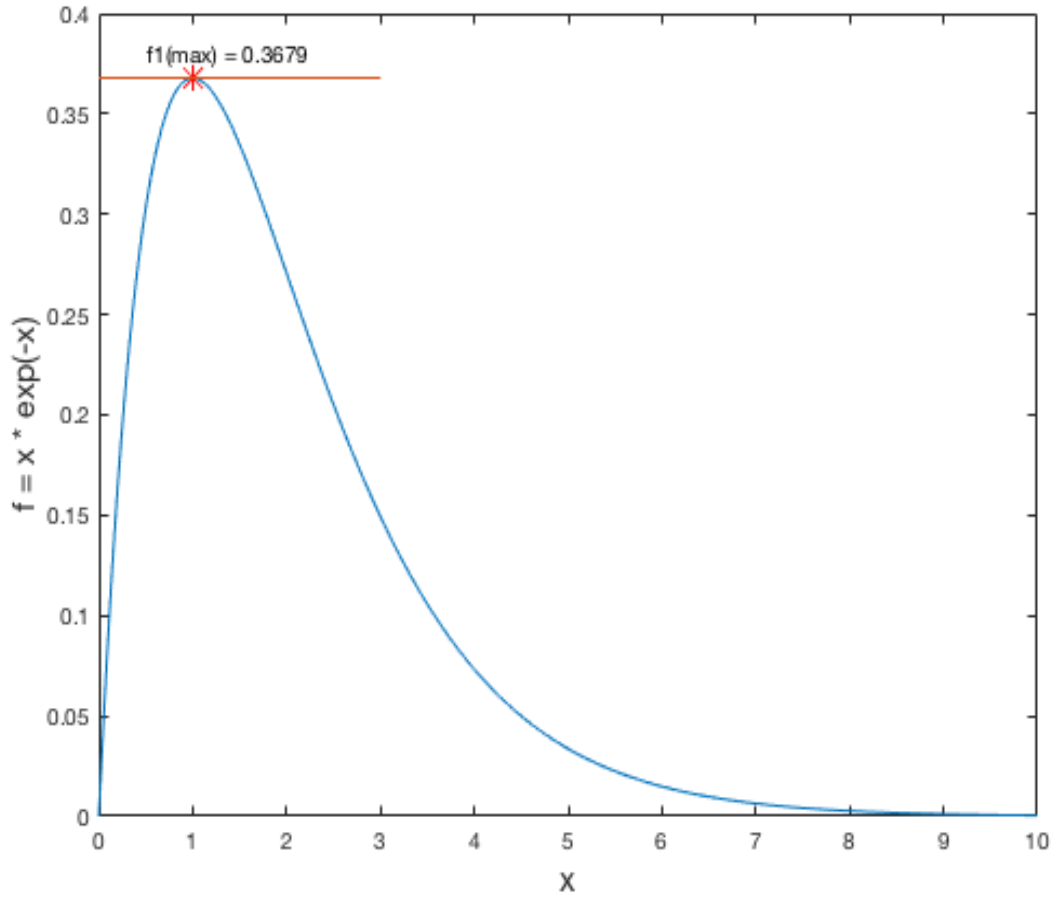


Figure 11: plot of  $f_1(x) = x * \exp(-x)$

### 1.3 Lyapunov exponent

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

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

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

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

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

$$|\Delta a_n| = |\Delta a_0| e^{\lambda n} \quad (13)$$

to compute Lyapunov exponent  $\lambda$  numerically, we use the following:

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

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  $b \geq 31$   $\lambda$  is -infinity that means the system converge very fast, and this explains why the population become extinct very fast for large  $b$ .

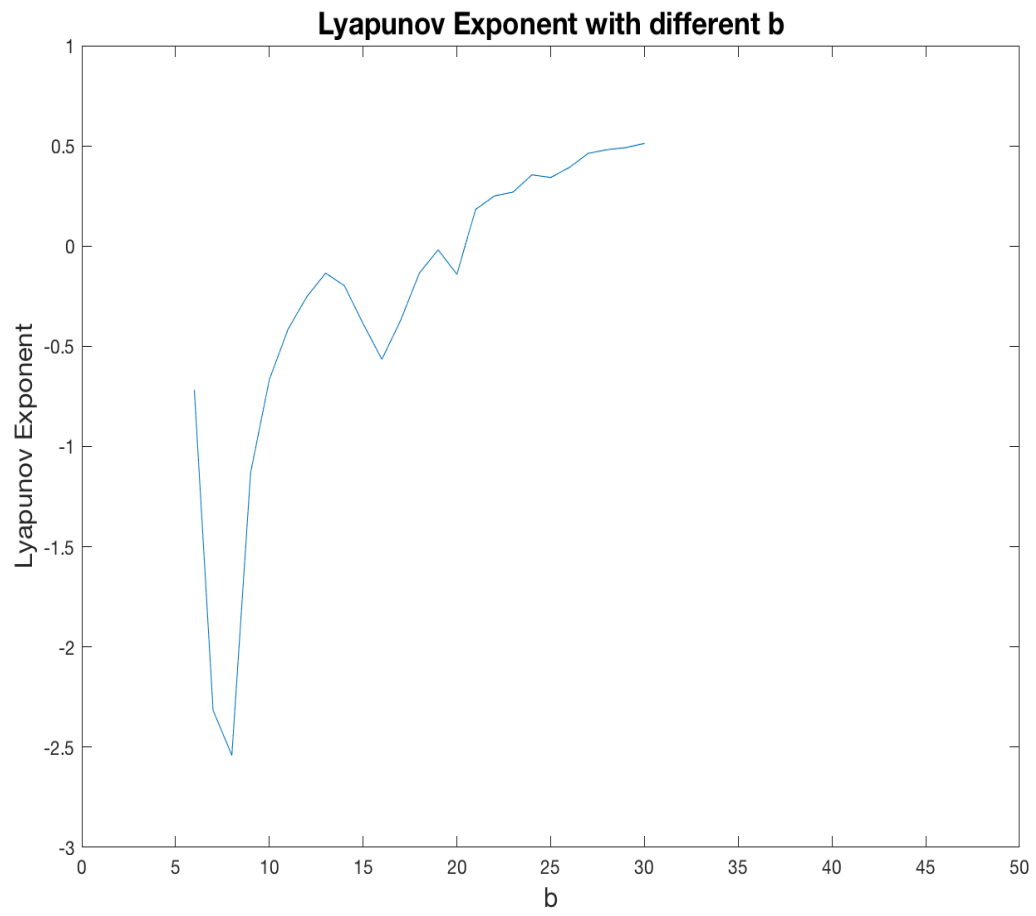


Figure 12: plot of Lyapunov exponent

## 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.

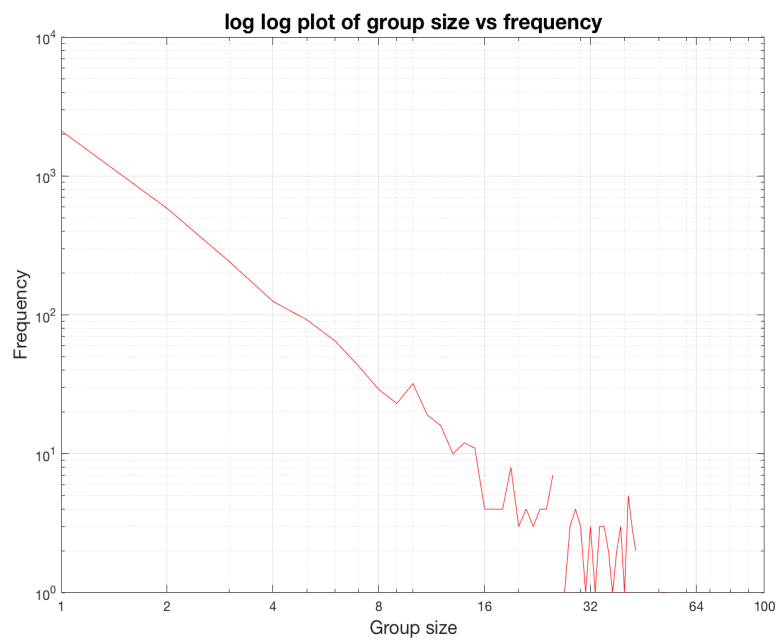


Figure 13: log log plot of group size vs frequency

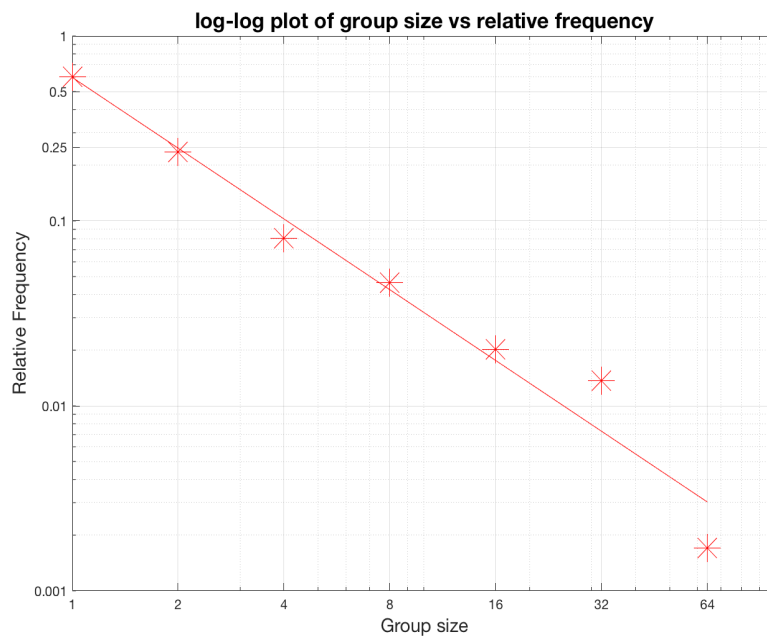


Figure 14: log log plot of group size vs relative frequency

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  $t$ .

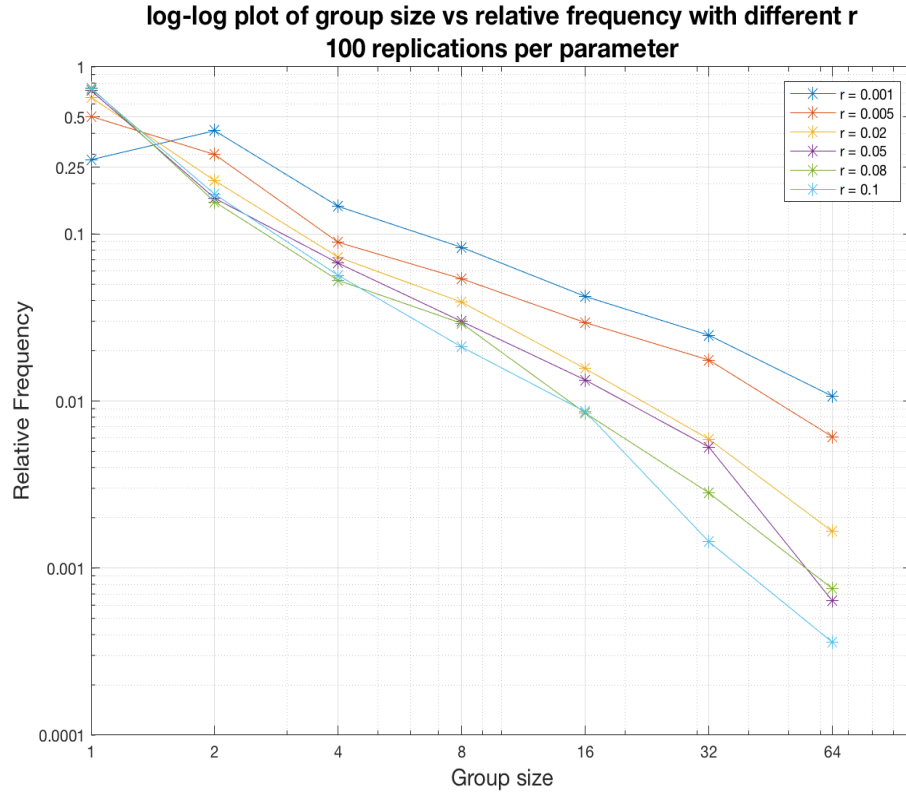


Figure 15: log log plot of group size vs relative frequency with different  $r$

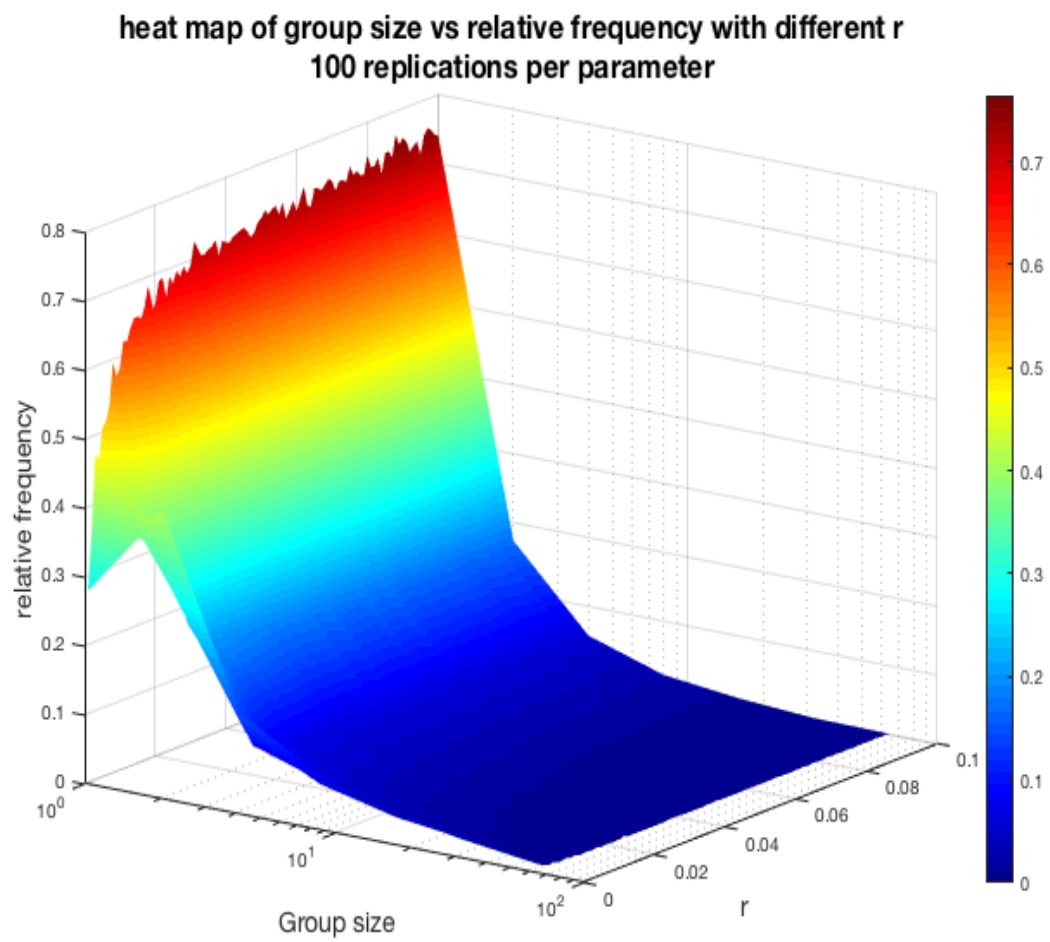


Figure 16: the distribution of groups with different  $r$  showing on a 3d plot



## 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

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

For  $k = 1$ :

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

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

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

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

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

now we find  $c$ :

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

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

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

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.

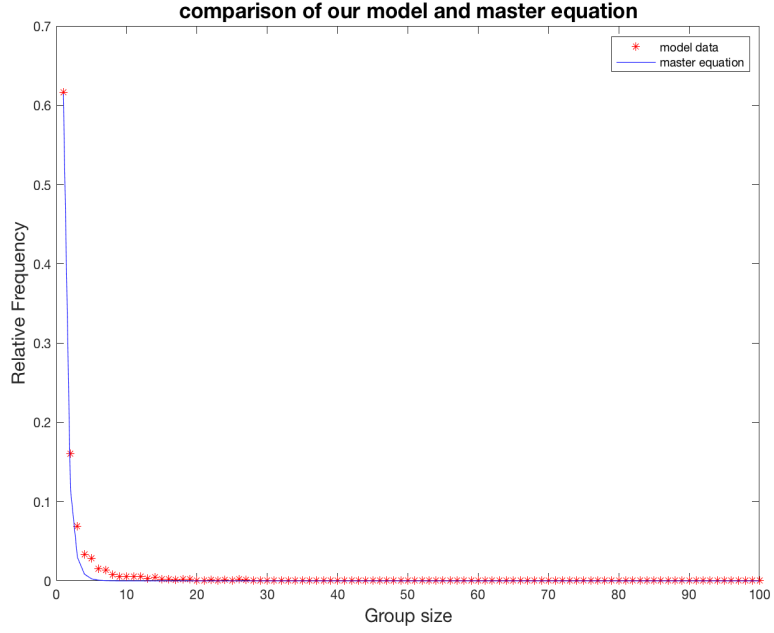


Figure 17: comparison of the distribution from simulation to master equation

## 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.

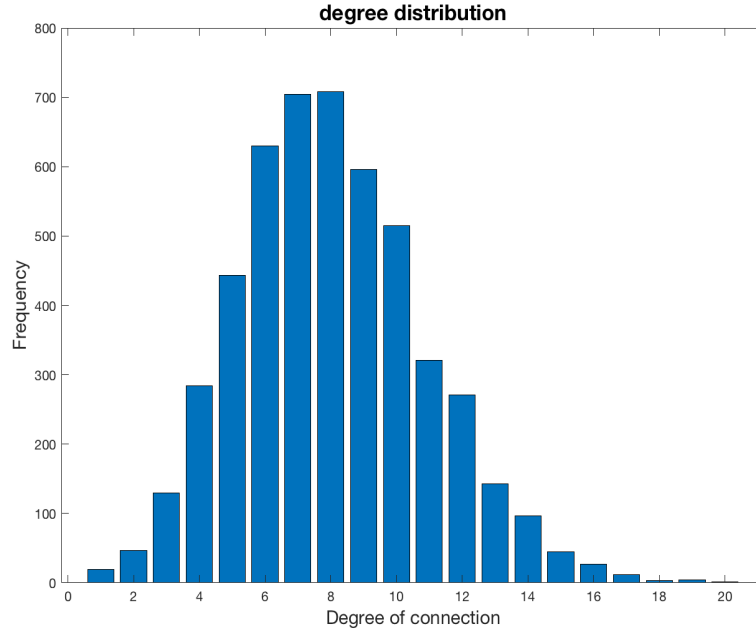


Figure 18: Frequency of the connection degree

### 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  $r/p \leq 5$ .

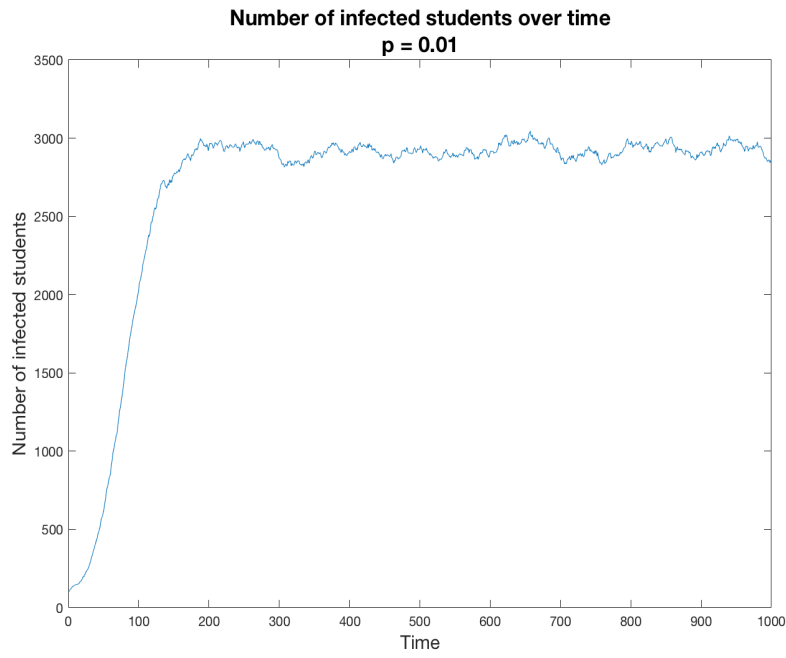


Figure 19: Number of infected individuals against time with  $p = 0.01$

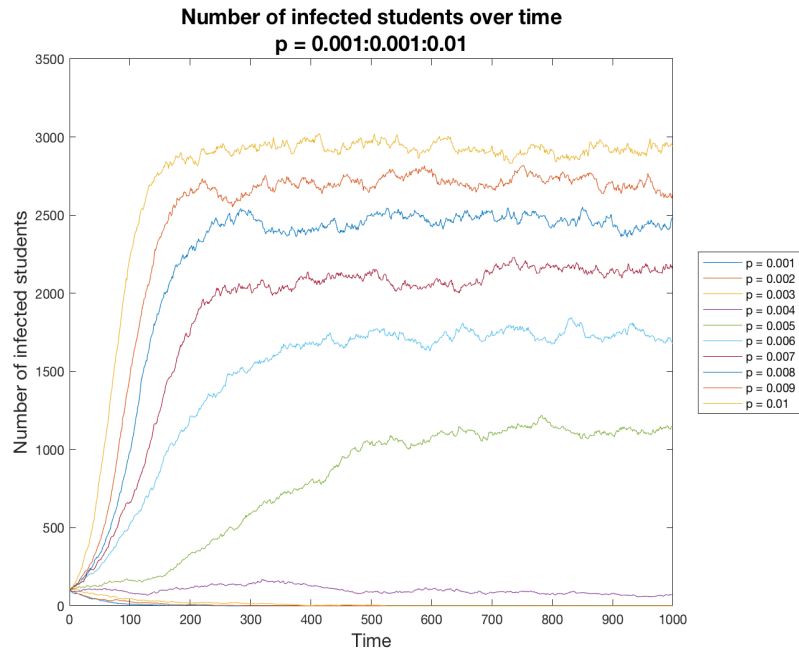


Figure 20: Number of infected individuals against time with different  $p$

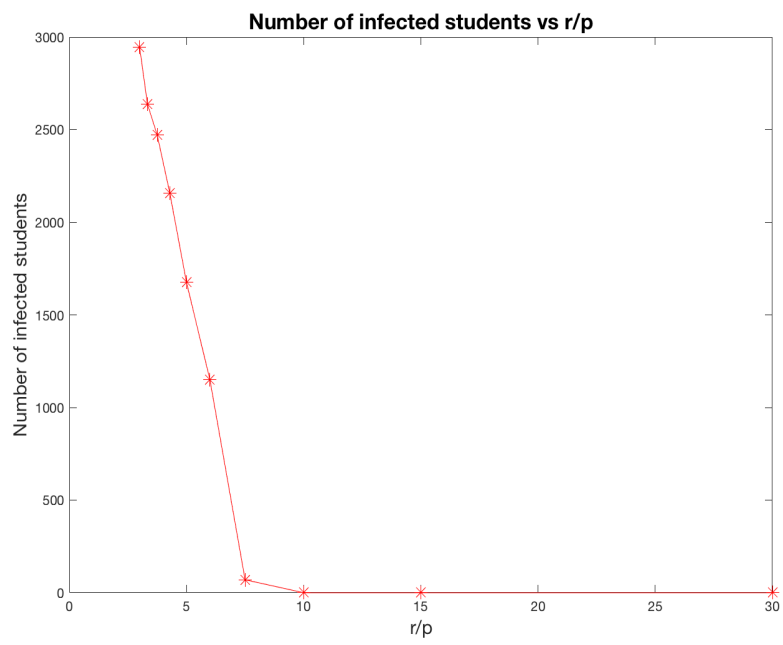


Figure 21: Number of infected individuals vs.  $r/p$

### 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.

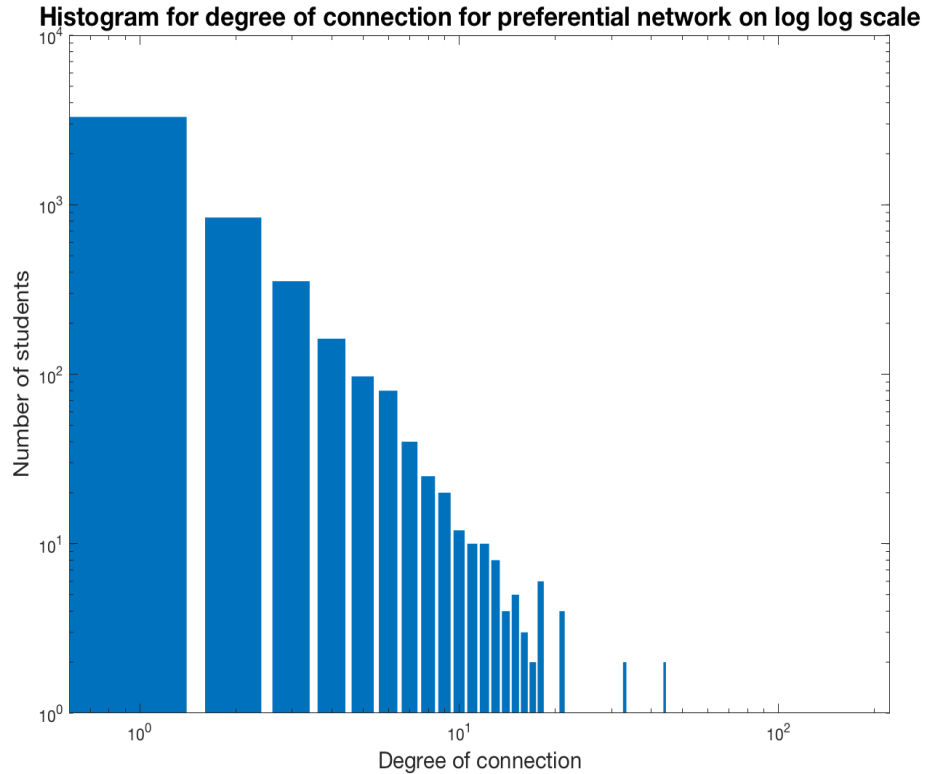


Figure 22: Histogram of the degree distribution

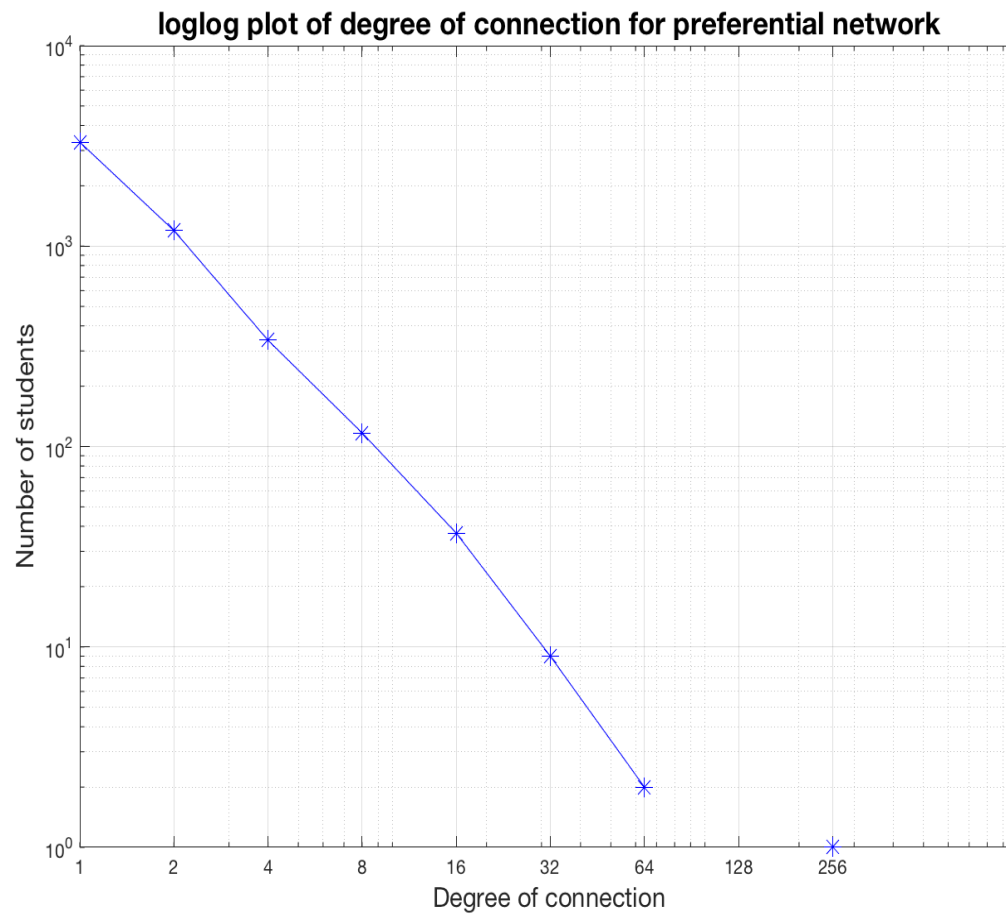


Figure 23: loglog plot of degree of connection



### 3.4 Master Equation

Below, I show my work for master equation.

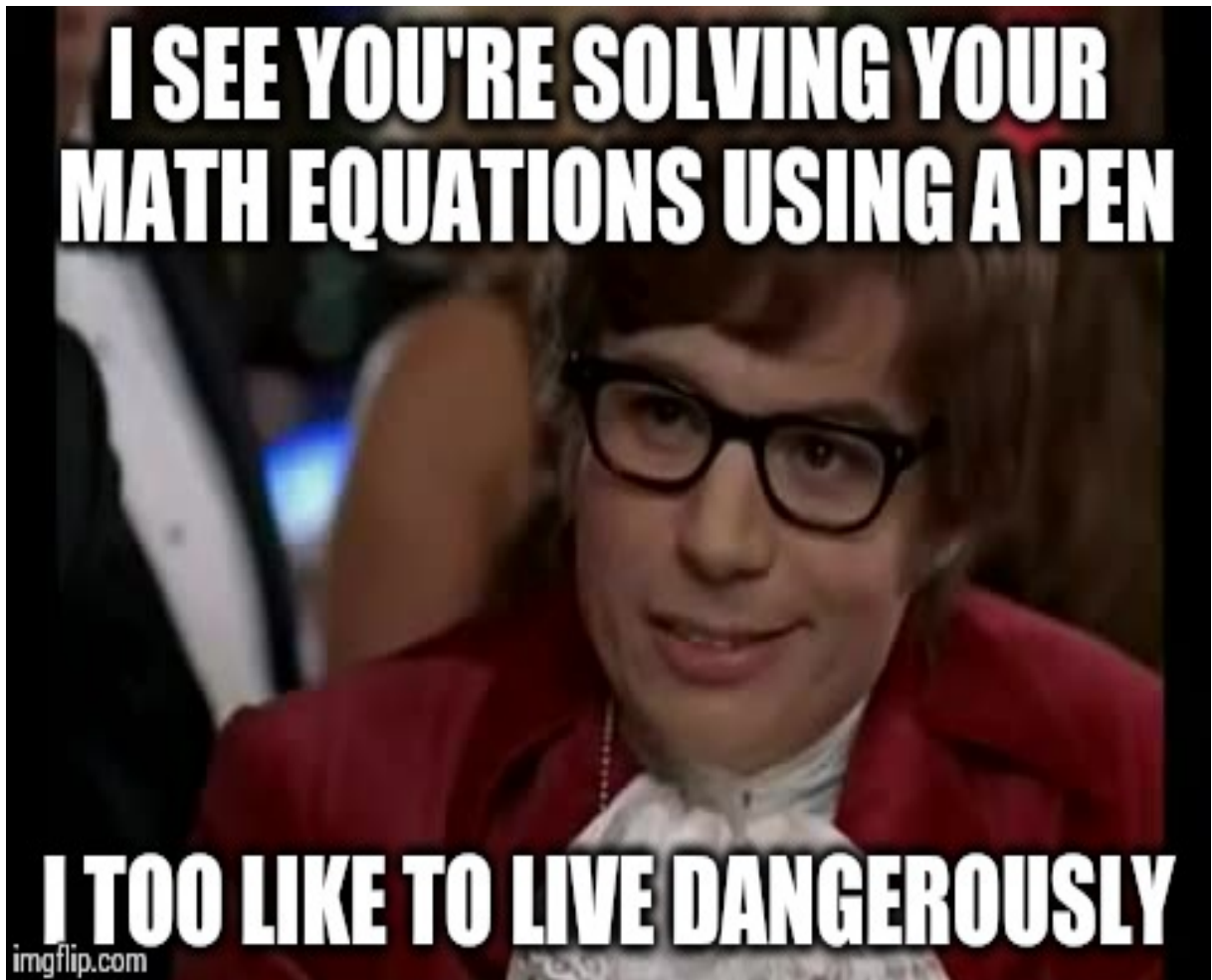


Figure 24: master equation for the preferential network

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:

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

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

to find steady state, we set  $n$  and  $t$  to  $\infty$ . we get:

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

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

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

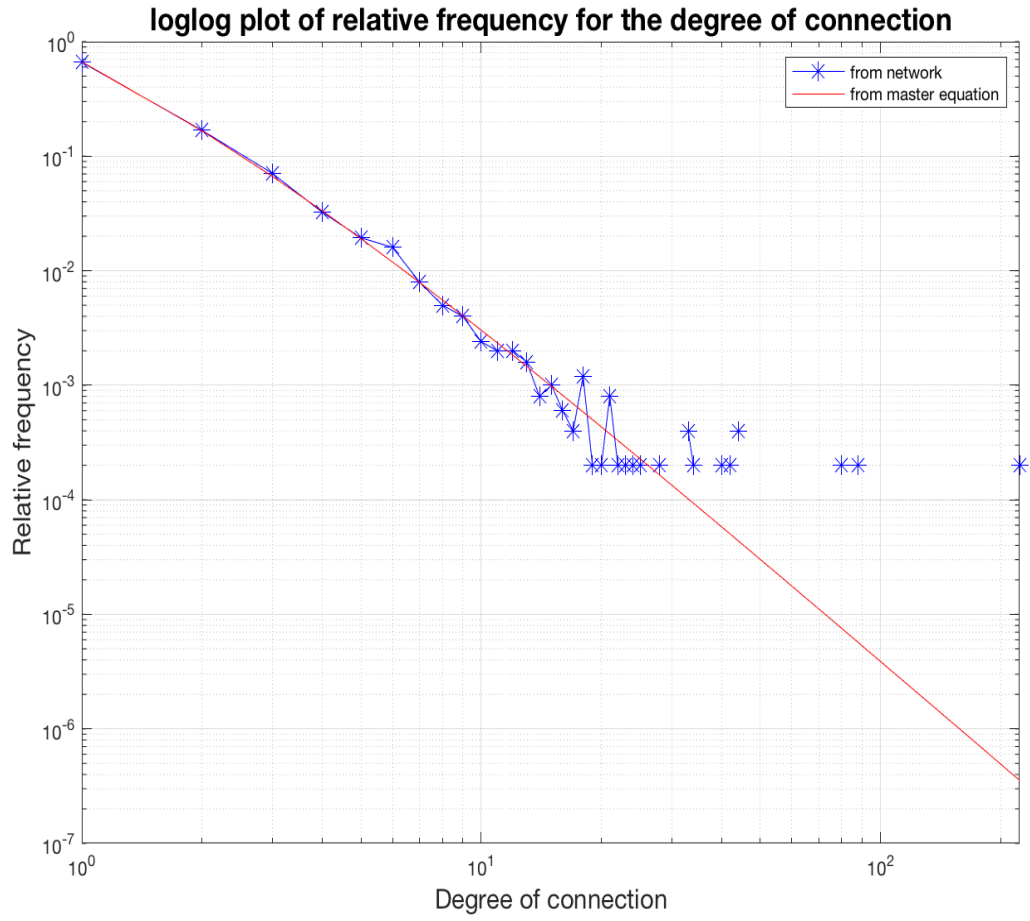
when  $k = 1$ :

$$p_1 = 1 - \frac{1}{2} \times p_1 \quad (28)$$

$$p_1 = \frac{2}{3} \quad (29)$$

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

Then we got our result  $c = 4$ .



### 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  $r/p \geq 6$ , 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.

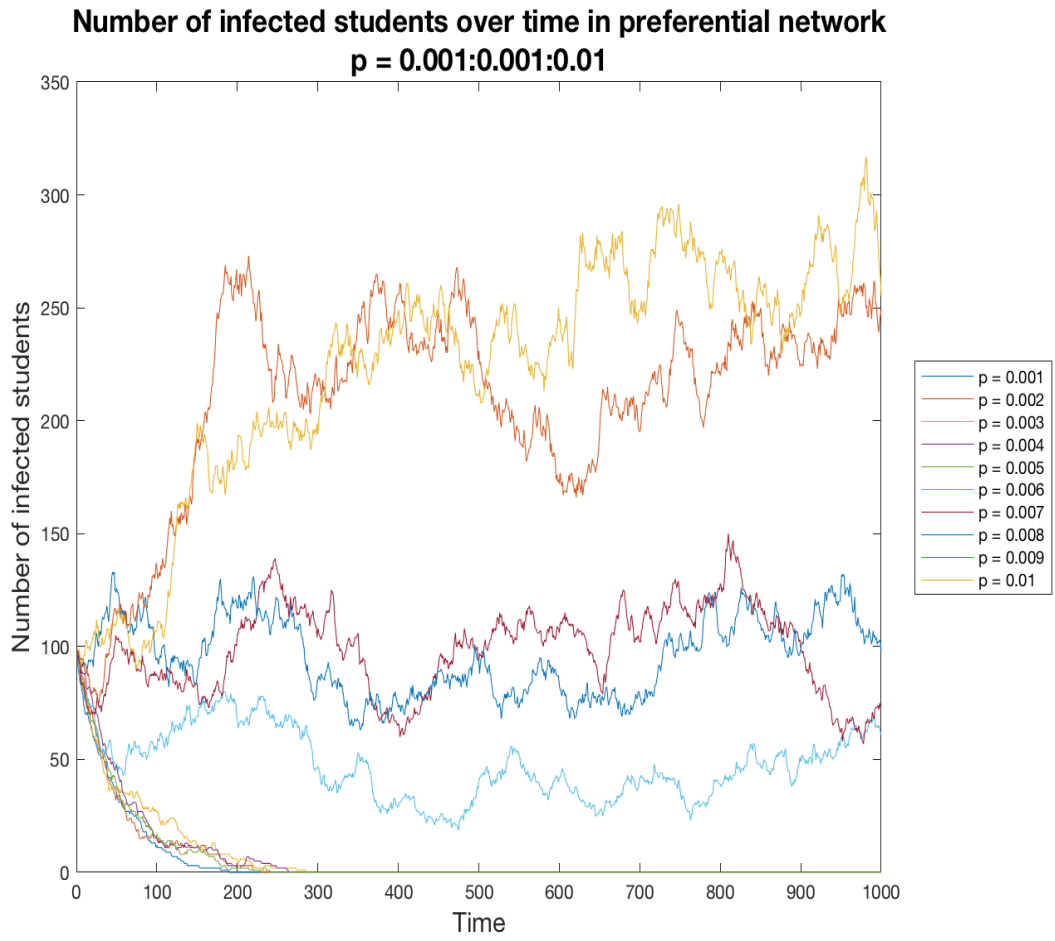


Figure 26: Number of infected individuals against time with different  $p$

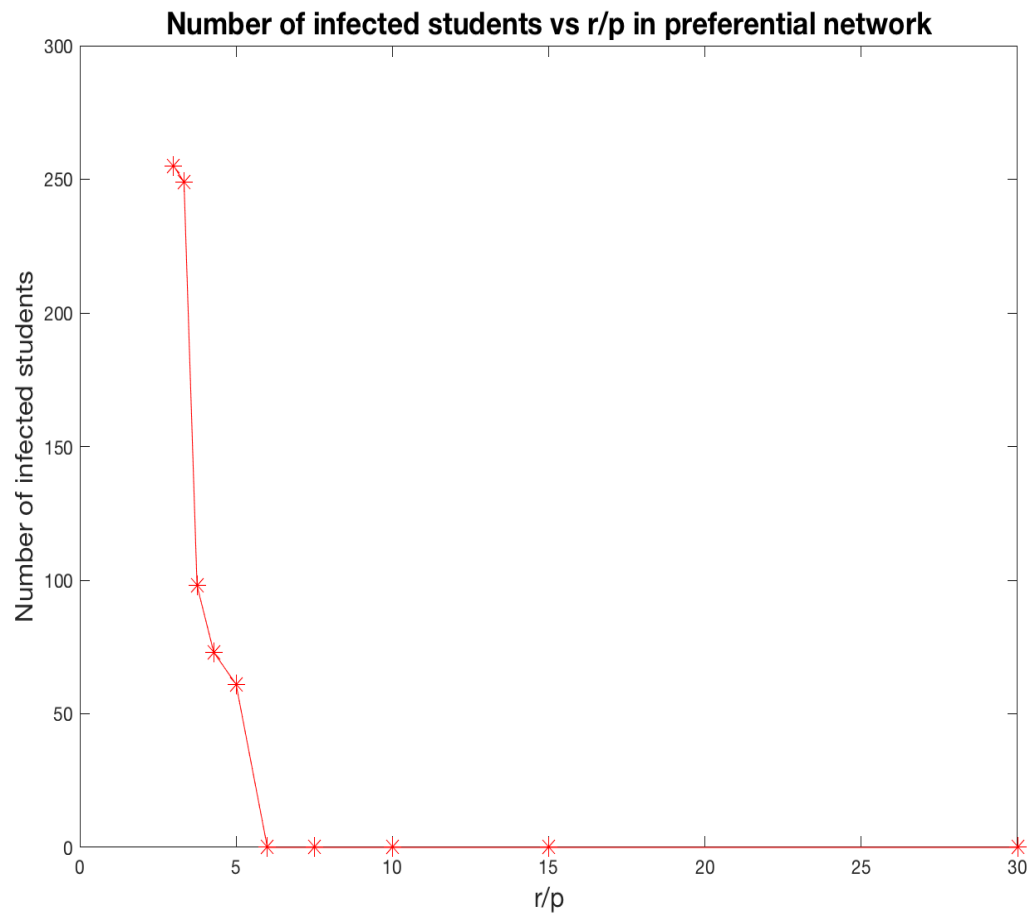


Figure 27: Number of infected individuals against time with different  $p$

## 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.

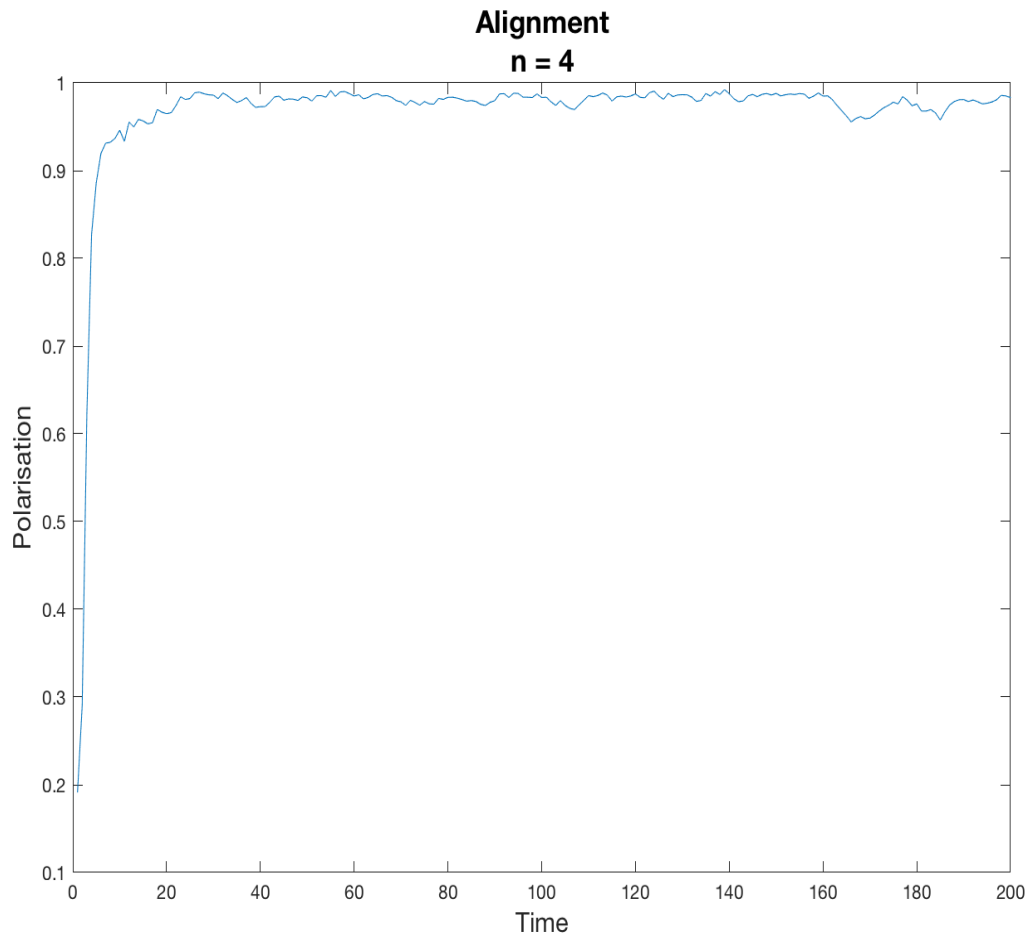


Figure 28: Alignment (4 nearest neighbour)

## 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  $e$ .

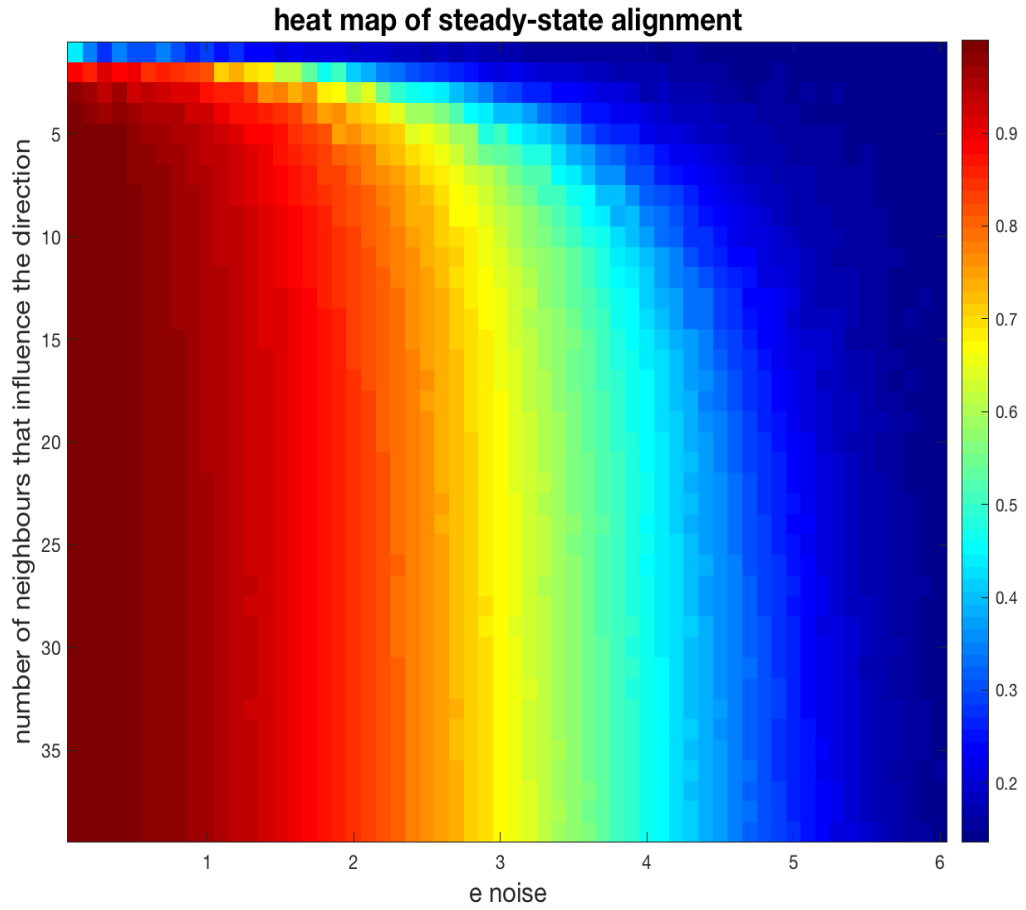


Figure 29: Heat map of steady state alignment



### 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( $k_1$ ), and the second half of the group will move with direction based on small number of neighbours( $k_2$ ). The predator is attracted to the 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  $k_1$  and  $k_2$  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.

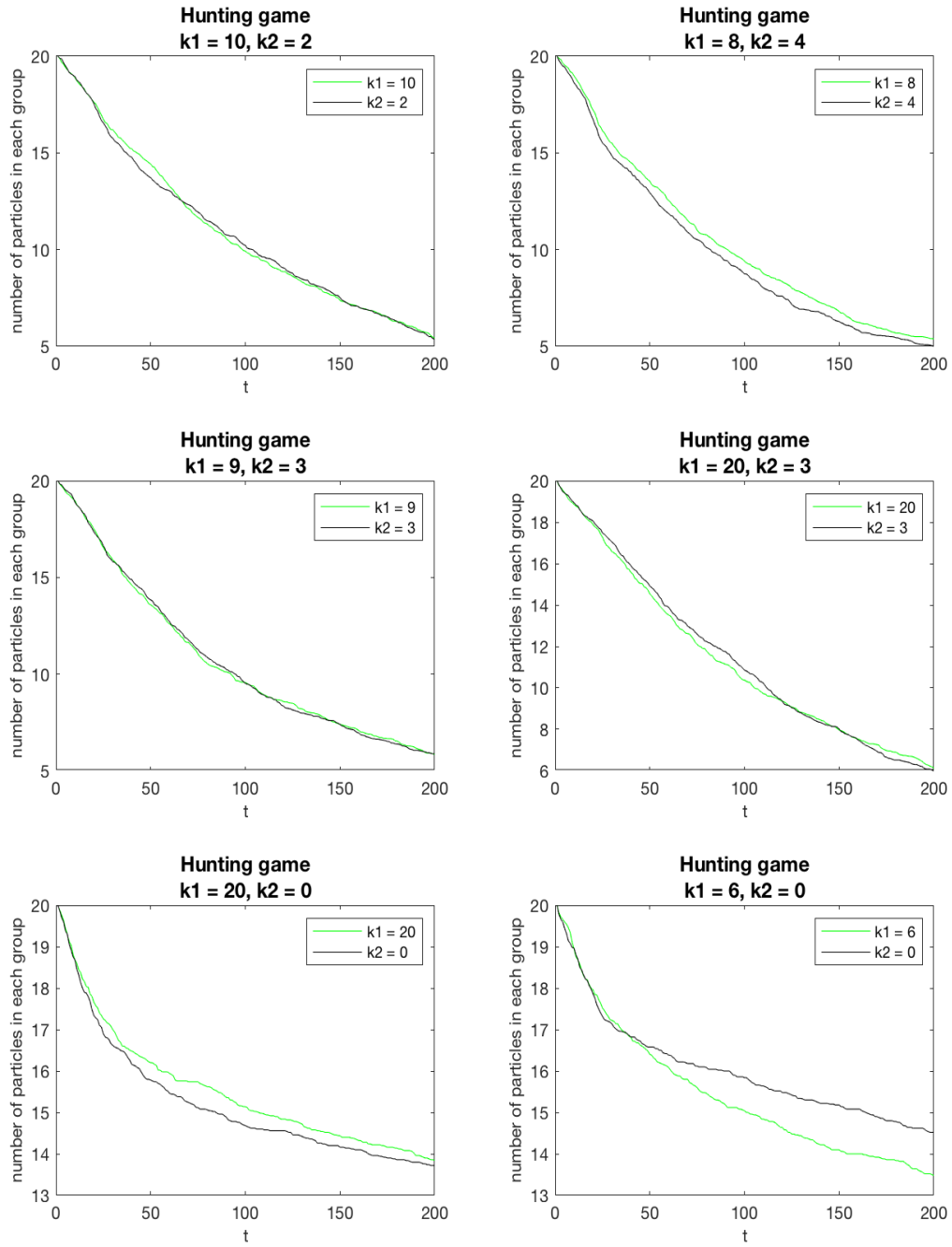


Figure 30: number of survivors with different neighbours influence the direction

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/00as2\\_Q7ef8](https://youtu.be/00as2_Q7ef8)>

## 5 Appendix

### 5.1 Population Dynamics

- sim population

```
1 function y = sim_population(n,p,t,b)
2 %% this function simlate the population with specific n,
   population, time, offspring
3 % output the vector
4 % n- number of resource site
5 % p- population at t0
6 % t- simulate the time step t
7 % b- number of offspring per 2 individual
8 % generate nxt matrix, where each row is the resource
   site, and each column
9 % is each time
10 y = zeros(n,t);
11 y(:,1) = get_initial(n,p);
12
13 for t = 2:1:t
14     y(:,t) = next_generation(y(:,t-1),n,b);
15 end
16 y = sum(y);
17
18 end
```

- get initial

```
1 function y = get_initial(n,p)
2 %% this function gives the generation 0's location
3 % input: n- number of resources
4 %         p- the total number of population at t0.
5
6     y = zeros(n,1);
7     for i = 1:1:p
8         loc = randi(n);
9         y(loc) = y(loc) + 1;
10    end
11 end
```

- next generation

```

1  function y = next_generation(x,n,b)
2  %% this function gets the next_generation column vector
3  %x is the current population vector at each location
4  %n is the number of locations
5  %b is the number of offspring per 2 individual
6
7  y = zeros(n,1); % initialize our next generation vector
8
9      for i = 1:1:n
10         if x(i) == 2 % only care about if y(i) is
                exactly 2
11             for j = 1:1:b
12                 loc = randi(n); %change the random number
                                generator
13                 y(loc) = y(loc) + 1;
14             end
15         end
16     end
17
18 end
19
20 %{
21 tips from teacher
22 loc=ceil(rand(b,1)*n)
23 y(loc)=y(loc)+1
24 %}
```

- mean field model

```

1  clear all;
2
3  n = 1000;
4  a0 = n;
5
6  bval = [5, 10, 15, 20, 35, 50];
7
8
9  for b = bval
10     A(b,1) = a0;
11
```

```

12     for i = 2:1:1000
13
14         A(b,i) = b*n*poisspdf(2,A(b,i-1)/n);
15     end
16 end
17
18 %p = [5, 10, 15, 20, 35, 50];
19 figure2 = figure('position', [0, 0, 700, 500]);
20 subplot(3,2,1)
21 plot(A(5,:))
22 xlabel('t','FontSize',10)
23 ylabel('total population','FontSize',10)
24 title({'Mean-field model for population'; 'b = 5'}, '
    FontSize', 12)
25
26 subplot(3,2,2)
27 plot(A(10,:))
28 xlabel('t','FontSize',10)
29 ylabel('total population','FontSize',10)
30 title({'Mean-field model for population'; 'b = 10'}, '
    FontSize', 12)
31
32 subplot(3,2,3)
33 plot(A(15,:))
34 xlabel('t','FontSize',10)
35 ylabel('total population','FontSize',10)
36 title({'Mean-field model for population'; 'b = 15'}, '
    FontSize', 12)
37
38 subplot(3,2,4)
39 plot(A(20,:))
40 xlabel('t','FontSize',10)
41 ylabel('total population','FontSize',10)
42 title({'Mean-field model for population'; 'b = 20'}, '
    FontSize', 12)
43
44 subplot(3,2,5)
45 plot(A(35,:))
46 xlabel('t','FontSize',10)
47 ylabel('total population','FontSize',10)
48 title({'Mean-field model for population'; 'b = 35'}, '
    FontSize', 12)

```

```

49
50 subplot(3,2,6)
51 plot(A(50,:))
52 xlabel('t','FontSize',10)
53 ylabel('total population','FontSize',10)
54 title({'Mean-field model for population'; 'b = 50'}, '
      FontSize', 12)
55
56
57 saveas(figure2, 'meanfield_model.png');

```

- phase transition

```

1 clear all
2
3 %enter b
4 Bvals = 1:1:50;
5
6 n = 1000; % number of resource sites
7 x0 = 100; %initial population number
8 t = 200;%simulate to t = 200
9 count = 0;
10 reps = 100;
11 hrange = 0:100:10000;
12
13 for b = Bvals
14     yall = [];
15     count = count + 1
16
17     for i = 1:1:reps
18         y = sim_population(n,x0,t,b);
19         y_end = y(end);
20         y_store = y((end/2+1):end);
21         yall = [yall, y_store];
22         %take the last half of the simulation
23     end
24
25     % take histogram
26     histu(count,:) = hist(yall, hrange);
27 end
28

```

```

29
30 figure2 = figure('position', [0, 0, 700, 500]);
31 imagesc(Bvals,hrange,histu'/(reps*t/2), [0 0.1])
32 %normalize the display
33 colormap jet
34 colorbar
35 xlabel('b','FontSize',14)
36 ylabel('total population','FontSize',14)
37 title('population simulation phase transition','FontSize',16)
38 title(sprintf('population simulation phase transition
numreps = %s for each b', string(reps)), 'FontSize',16)
39
40
41 saveas(figure2, 'q1_smallp.png');

```

- dfda

```

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

```

- lyapunov

```

1 clear all;
2 %need to compute average?
3 n = 1000; % number of resource sites
4 x0 = 1000; %initial population number
5 t = 500;%simulate to t = 500
6
7 % if lamda < 0 ,it converge
8 % if lamda > 0 ,it diverge
9
10
11 for b = 1:1:50
12     b
13     x = get_initial(n,x0);
14

```



```

15     lamda(1) = abs(dfda(n,b,x0));
16
17     for i = 2:1:t
18
19         x = next_generation(x,n,b);
20         a = sum(x);
21         lamda(i) = log(abs(dfda(n,b,a)));
22     end
23
24     lya_exp(b) = (1/t)* sum(lamda);
25 end
26
27
28
29 figure2 = figure('position',[0, 0, 700, 500]);
30 plot([1:1:50],lya_exp)
31 xlim([0 50])
32 xlabel('b','FontSize',14)
33 ylabel('Lyapunov Exponent','FontSize',14)
34 title({'Lyapunov Exponent with different b'}, 'FontSize'
35       , 16)
35 %saveas (figure2, 'lyapunov_exp.png');

```

## 5.2 Groups of friends

- join group

```

1 function G = join_group(G,pick)
2 %% pick a person/group and join them all to another join
   a group
3 % p to join depends on groupsize = k/N
4
5 target = ceil(rand()*length(G));
6 P = cumsum(G);
7
8 k = find(P>=target,1);%find the index of the group to
   join
9
10 if k ~= pick %not join the existing group
11

```

```

12     G(k) = G(k) + 1;%add one number
13     G = G([1:pick-1 pick+1:end]);
14 else
15     G = G;
16 end

```

- split group

```

1 function G = split_group(G,pick)
2 %% this function to get a group split up
3 k = G(pick);
4 G = G([1:pick-1 pick+1:end]);
5 add = ones(1,k);
6 G = [G add];
7 end

```

- sim friends

```

1 clear all;
2 close all;
3
4 N = 100; % initial number of individuals
5
6 G = ones(1,N);
7
8 r = 0.01; %p for split
9 ft = 1000;%final time step
10
11 result = [];
12 i = 0;
13 hrange = [];
14 hrange(1) = 1;
15 h = 0;
16
17 while h < 100
18     i = i + 1;
19     h = 2^i;
20     hrange = [hrange h]; %for our histogram
21 end

```

```

22  ngroup = 0;
23
24  for j = 1:1:100
25      j;
26
27      for t = 1:1:ft
28
29          pick = ceil(rand()*length(G));
30
31          if G(pick) == 1
32              G = join_group(G,pick);
33          elseif rand() < r*G(pick)
34              G = split_group(G,pick);
35          end
36          ngroup = ngroup + length(G);
37
38          %storeh(t,:)=hist(G,[1:50])';
39          %imagesc(storeh,[0 20])
40
41
42      end
43      result= [result G];%take the result when it is
44          stable
45  end
46  histu = hist(result,hrange);
47  nhistu = histu/sum(histu); %frequency distribution
48
49  if nhistu(end) == 0
50      nhistu = nhistu(1:end-1);
51      hrange = hrange(1:end-1);
52  end
53
54  figure2 = figure('position',[0, 0, 700, 500]);
55  loglog(hrange,nhistu,'r*','markersize',20)
56  hold on
57  grid on
58  p = polyfit(log(hrange), log(nhistu), 1);
59  y_hat = exp(p(1) * log(hrange) + p(2));
60  loglog(hrange, y_hat, 'r')
61  xlabel('Group size','FontSize',14)
62  set(gca, 'XTick', hrange)

```

```

63 set(gca, 'YTick', [0.0001 0.001 0.01 0.1 0.25 0.5 1])
64 ylabel('Relative Frequency','FontSize',14)
65 title({'log-log plot of group size vs relative frequency',
        'FontSize', 16)
66 saveas('figure2','loglog1.png');
67
68 hrange2 = 1:1:100;
69 histu2 = hist(result,hrange2);
70 nhistu2 = histu2/sum(histu2);
71
72 figure3 = figure('position', [0, 0, 700, 500]);
73 loglog(hrange2,histu2,'r','markersize',15)
74 hold on
75 grid on
76 %p = polyfit(log(hrange), log(nhistu), 1);
77 %y_hat = exp(p(1) * log(hrange) + p(2));
78 %loglog(hrange, y_hat,'r')
79 xlabel('Group size','FontSize',14)
80 set(gca, 'XTick', [hr])
81 %set(gca, 'YTick', [0.0001 0.001 0.01 0.1 0.25 0.5 1])
82 ylabel('Frequency','FontSize',14)
83 title({'log log plot of group size vs frequency'}, '
        FontSize', 16)
84 saveas('figure3','loglog2.png');

```

- sim friends with replication

```

1 clear all;
2
3 N = 100; % initial number of individuals
4
5 G = ones(1,N);
6
7 r = 0.01;
8 ft = 1000;%final time step
9 hrange = 1:1:100; %for histogram
10 result = [];
11 gn = [];
12 reps = 100;
13
14 for j = 1:1:100

```

```

15         j;
16
17         for t = 1:1:ft
18
19             pick = ceil(rand()*length(G));
20
21             if G(pick) == 1
22                 G = join_group(G,pick);
23             elseif rand() < r*G(pick)
24                 G = split_group(G,pick);
25             end
26
27             %storeh(t,:)=hist(G,[1:50])';
28             %imagesc(storeh,[0 20])
29
30
31         end
32         result= [result G];
33         gn = [gn length(G)];
34     end
35
36     histu = hist(result,hrange);
37     nhistu = histu/sum(histu);
38
39
40     %store the final group distribution
41
42
43
44     histu = hist(result,hrange);
45     %loglog(histu);
46
47     K = 1.6145;
48     c = 0.3805;
49     y_predict = [];
50
51     for ii = hrange
52         y_predict(ii) = (K * c^ii) /ii;
53     end
54
55     save('findpi2.mat')
56

```

```

57 figure2 = figure('position', [0, 0, 700, 500]);
58 plot(hrange, nhistu, 'r*')
59 hold on
60
61 plot(y_predict, 'b')
62 xlabel('Group size', 'FontSize', 14)
63
64
65 ylabel('Relative Frequency', 'FontSize', 14)
66 legend('model data', 'master equation');
67 title({'comparison of our model and master equation'}, '
        FontSize', 16)
68 saveas(figure2, 'mel.png');

```

- sim friends with different r

```

1 clear all;
2
3 N = 100; % initial number of individuals
4
5 G = ones(1, N);
6
7 ft = 1000; % final time step
8 %hrange = 1:1:N; %for histogram
9 result = [];
10 reps = 100; %number of reps with each parameter
11 diffrr = 0.001:0.001:0.1;
12 count = 0;
13 total = zeros(length(diffrr), length(diffrr)*reps);
14 hrange = [];
15 hrange(1) = 1;
16 h = 0;
17 ii = 0;
18 while h < 100
19     ii = ii + 1;
20     h = 2^ii;
21     hrange = [hrange h]; %for our histogram
22 end
23
24
25 for r = diffrr

```

```

26     count = count + 1
27     result = [];
28     for re = 1:1:reps
29         G = final_groups(G,ft,r);
30         %this is the final group size distribution
31         result = [result G];
32     end
33     histu(count,:) = hist(result,hrange);
34     total(count,1:length(result)) = result;
35     %take the final step histogram
36 end
37
38 histu_mean = histu/reps;
39 %take the nomalized?
40
41 loglog(histu_mean');
42
43 save('sim_friends_try3')

```

### 5.3 Network Epidemics

- random network

```

1 clear all
2
3 n = 5000;
4 A = zeros(n,n);
5 p = 0.0016;
6
7 for i = 1:1:n
8     for j = i+1:1:n
9         if rand < p
10             A(i,j) = 1;
11             A(j,i) = 1;
12         end
13     end
14 end
15
16 dir = sum(A);
17

```

```

18 range = 1:1:max(dir);
19 histu = hist(dir,range);
20 nhistu = histu/sum(histu);
21
22 figure2 = figure('position',[0, 0, 700, 500]);
23 bar(nhistu)
24
25 xlabel('Degree of connection','FontSize',14)
26 ylabel('Frequency','FontSize',14)
27 title({'degree distribution'}, 'FontSize', 16)
28 %saveas(figure2,'randomf2.png');
29
30
31
32
33 k = (1/n)*sum(dir);
34
35 %save('try1')

```

- infection transit

```

1 function y = infection_transit(A,b,n,p,r)
2 % A connection matrix
3 % b state vector (1 infected, 0 good)
4 % n number of population
5 % p constant related to infect
6 % r recover rate
7
8 y = zeros(n,1); %initial the next state
9 neighbours = zeros(n,1);
10 for i = 1:n
11     if b(i) == 0
12         neighbours(i) = A(i,:) * b;
13         pi(i) = 1 - exp(-p*neighbours(i));
14         if rand < pi(i)
15             y(i) = 1;
16         else
17             y(i) = 0;
18         end
19     else
20

```



```

21         if rand < r
22             y(i) = 0;
23         else
24             y(i) = 1;
25         end
26     end
27 end
28 end

```

- probability vector

```

1 function P = probability_vector(A,i)
2     links = sum(A);
3     links = links(1:i-1);
4
5     for j = 1:1:i-1
6         P(j) = links(j)/(2*(i-2));
7     end
8 end

```

- sim infection

```

1 clear all
2
3 n = 5000; %number of people
4 A = zeros(n,n); %initial network
5 p = 0.0016; %connection probability
6
7 for i = 1:1:n
8     for j = i+1:1:n
9         if rand < p
10             A(i,j) = 1;
11             A(j,i) = 1;% create network one by one
12         end
13     end
14 end
15
16 infected_start = randperm(n,100); % the infected ones
17 infected_vector = zeros(n,1);

```

```

18 for i = 1:1:length(infected_start)
19     infected_vector(infected_start(i)) = 1;
20 end
21 y0 = sum(infected_vector);
22 b0 = infected_vector;
23 ps = 0.001:0.001:0.01;
24 r = 0.03;
25 j = 0;
26
27 for pe = ps
28     j = j+1
29     infected_start = randperm(n,100); % the infected
30     ones
31     infected_vector = zeros(n,1);
32     for i = 1:1:length(infected_start)
33         infected_vector(infected_start(i)) = 1;
34     end
35     y0 = sum(infected_vector);
36     b0 = infected_vector;
37
38     for t = 1:1:1000
39         b1 = infection_transit(A,b0,n,pe,r);
40         y(j,t) = sum(b1);
41         b0 = b1;
42     end
43
44 end
45
46
47 save('resultp10run')
48
49 plot(y)

```

- preferential network

```

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

```

```

6 %link the first two students.
7
8 for i = 3:1:n
9     links = sum(A);
10    cum_links = cumsum(links);
11    target = rand;
12    p = probability_vector(A,i);
13    tp = cumsum(p);
14    k = find(tp>= target , 1);
15    %found our link student
16    A(k,i) = 1;
17    A(i,k) = 1;
18    %write our connect matrix
19 end
20 end

```

- preferential network

```

1 clear all
2 close all
3 n = 5000;
4 %{
5 s = randperm(n,2);
6
7 A = zeros(n,n);
8 A(s(1),s(2)) = 1;
9 A(s(2),s(1)) = 1;
10 %created the first link
11 %}
12 A = zeros(n,n);
13 A(1,2) = 1;
14 A(2,1) = 1;
15 %link the first two students.
16
17 for i = 3:1:n
18     links = sum(A);
19     cum_links = cumsum(links);
20     target = rand;
21     p = probability_vector(A,i);
22     tp = cumsum(p);
23     k = find(tp>= target , 1);

```

```

24     %found our link student
25     A(k,i) = 1;
26     A(i,k) = 1;
27     %write our connect matrix
28 end
29
30 degree_nw = sum(A);
31 max_degree = max(degree_nw);
32 hrange = 1:1:max_degree;
33
34 histu = hist(degree_nw,hrange);
35 bar(histu);
36 loglog(histu);
37
38 %average degree of network
39
40 ave_degree = (1/(n))*sum(sum(A));

```

- preferential infection

```

1  clear all
2
3  n = 5000;
4  A = zeros(n,n);
5  A(1,2) = 1;
6  A(2,1) = 1;
7  %link the first two students.
8  for i = 3:1:n
9      links = sum(A);
10     cum_links = cumsum(links);
11     target = rand;
12     p = probability_vector(A,i);
13     tp = cumsum(p);
14     k = find(tp>= target, 1); %found our link student
15     A(k,i) = 1;
16     A(i,k) = 1; %write our connect matrix
17 end
18
19
20 ps = 0.001:0.001:0.01;
21 r = 0.03;

```

```

22 j = 0;
23
24 for pe = ps
25     j = j+1
26     infected_start = randperm(n,100); % the infected
        ones
27     infected_vector = zeros(n,1); %initial a state vector
28     for i = 1:1:length(infected_start)
29         infected_vector(infected_start(i)) = 1;
30         %write the infected student one by one
31     end
32
33     y0 = sum(infected_vector);
34     b0 = infected_vector;
35
36     for t = 1:1:1000
37         b1 = infection_transit(A,b0,n,pe,r);
38         y(j,t) = sum(b1);
39         b0 = b1;
40     end
41
42 end
43 plot(y)
44
45 %{
46 degree_nw = sum(A);
47 max_degree = max(degree_nw);
48 hrange = 1:1:max_degree;
49
50 histu = hist(degree_nw,hrange);
51 bar(histu);
52 loglog(histu);
53
54 %average degree of network
55
56 ave_degree = (1/(n*n))*sum(sum(A));
57 %}

```

- full simulation for preferential network

```

1 clear all

```

```

2
3 % to simulate infection with preferential network
4 sim_t = 1; %want to get different network matrix A
5 n = 5000; %population
6 final_t = 5; % simulate for 1000 time steps
7
8 ps = 0.001:0.001:0.01;
9 y = zeros(length(ps), final_t);
10
11 for st = 1:1:sim_t
12     st
13     A = p_network(n);
14     r = 0.03;
15     j = 0;
16
17     for pe = ps
18         j = j+1
19         infected_start = randperm(n,100); % the infected
20             ones
21             infected_vector = zeros(n,1);%initial a state
22             vector
23             for i = 1:1:length(infected_start)
24                 infected_vector(infected_start(i)) = 1;
25                 %write the infected student one by one
26             end
27
28             y0 = sum(infected_vector);
29             b0 = infected_vector;
30
31             for t = 1:1:final_t
32                 b1 = infection_transit(A,b0,n,pe,r);
33                 y(j,t) = y(j,t)+sum(b1);
34                 b0 = b1;
35             end
36         end
37     end
38
39 final_y = y/sim_t;
40 plot(final_y ')
41 save('result_sim10_pn')

```

## 5.4 Flocks and Predators

- distance

```
1 function D = distance(x,y,t,N)
2 %compute the distance matrix at time t
3 D = zeros(N,N);
4 xx = x(:,t);
5 yy = y(:,t);
6 %extract the curent location
7
8 for i = 1:1:N
9     for j = i+1:1:N
10         D(i,j) = sqrt((xx(i)-xx(j))^2 + (yy(i) - yy(j))^2);
11         D(j,i) = D(i,j);
12
13
14     end
15 end
16
17 end
```

- distance to predator

```
1 function D = distance_predator(x,y,t,N,px,py)
2 %compute the distance matrix at time t
3 D = [];
4 xx = x(:,t);
5 yy = y(:,t);
6 px = px(t);
7 py = py(t);
8 %extract the curent locations
9
10 for i = 1:1:N
11
12
13     D(i) = sqrt((xx(i)-px)^2 + (yy(i) - py)^2);
14     %if D(i,j) == NaN
15     %    D(i,j) = 10000;
16     %end
```

```

17
18
19 end
20
21 end

```

- alignment

```

1 clear
2 close all
3
4 %Set up movie
5 fig=figure;
6 makemovie=0;
7 %movien = avifile('Vicsekmovie','FPS',3,'compression','
    none')
8
9 J=200;
10 %Number of timestep t0 be used
11
12 UJ=1;
13 %Rate at which film is updated
14
15
16 t=1/J; %Size of one time step
17
18 N=40;
19 %Number of particles
20
21 e=0.5;
22 %e is eta the noise parameter, whose maximum value is 2*
    pi
23
24 r=1;
25 %The radius of influence of a particle
26
27 L=10;
28 %L is the size of the domain on which the particles can
    move
29
30 v=0.5; %v is the speed at which the particles move

```



```

31
32 % x(i,j) gives the x coordinate of the ith particle at
    time j
33 x=zeros(N,J+1);
34 x(:,1)=L*rand(N,1); %define initial x coordiantes of all
    particles
35
36 % y(i,j) gives the y coordinate of the ith particle at
    time j
37 y=zeros(N,J+1);
38 y(:,1)=L*rand(N,1); %define initial y coordiantes of all
    particles
39
40 % T(i,j) gives the angle with the x axis of the
    direction of motion of the ith
41 % particle at time j
42 T=zeros(N,J+1);
43 op = [];
44 T(:,1)=2*pi*rand(N,1); %define initial direction of all
    particles
45 k = 4;
46 ssin = sum(sin(T(:,1)));
47 scos = sum(cos(T(:,1)));
48 op(1) = (1/N) * sqrt(ssin^2 + scos^2);
49 %For all time steps
50 for j=1:1:J %iterate in time
51     %For each particle
52     D = distance(x,y,j,N);
53     [B,I] = sort(D);
54     near_n = I(2:k+1,:); %found our nearest 4 neighbours
        for each one
55
56     for i=1:1:N
57
58         %get the current distance vector
59         neighbour = near_n(:,i);
60         tn = []; % the neighbour direction
61         for m = 1:1:length(neighbour)
62             tn(m) = T(neighbour(m),j);
63         end
64
65         ss = sum(sin(tn));

```

```

66         sc = sum(cos(tn));
67         S = atan2(ss, sc);
68
69         T(i, j+1)=S+e*(rand-0.5); %adds noise to the
           measured angle
70
71         x(i, j+1)=x(i, j)+v*cos(T(i, j+1)); %updates
           the particles' x-coordinates
72         y(i, j+1)=y(i, j)+v*sin(T(i, j+1)); %updates
           the particles' y coordinates
73
74         % Jumps from the right of the box to the
           left or vice versa
75         x(i, j+1)=mod(x(i, j+1), L);
76
77         %Jumps from the top of the box to the bottom
           or vice versa
78         y(i, j+1)=mod(y(i, j+1), L);
79
80         %Plot particles
81
82         if makemovie
83             if abs(x(i, j)-x(i, j+1))<v & abs(y(i, j)-y
           (i, j+1))<v
84                 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
85                 axis([0 L 0 L]);
86                 hold on
87                 plot(x(i, j+1), y(i, j+1), 'k.', '
           markersize', 10)
88                 xlabel('X position')
89                 ylabel('Y position')
90
91             end
92         end
93
94     end
95
96     ssin = sum(sin(T(:, j+1)));
97     scos = sum(cos(T(:, j+1)));

```

```

98         op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);
99
100     if makemovie
101         hold off
102         M(j)=getframe; %makes a movie fram from the plot
103
104         %movien = addframe(movien,M(j)); %adds this
            movie fram to the movie
105     end
106
107 end
108
109
110 %movien = close(movien); %finishes the movie
111 figure2 = figure('position',[0, 0, 700, 500]);
112 plot([1:1:J+1], op)
113 xlim([0 200]);
114 xlabel('Time','FontSize',14)
115 ylabel('Polarisation','FontSize',14)
116 title({'Alignment';'n = 4'}, 'FontSize', 16)
117 saveas(figure2,'palign.png');

```

- with small force

```

1 clear
2 close all
3
4 %Set up movie
5 fig=figure;
6 makemovie=1;
7 %movien = avifile('Vicsekmovie','FPS',3,'compression','
    none')
8
9 J=200;
10 %Number of timestep t0 be used
11
12 UJ=1;
13 %Rate at which film is updated
14
15
16 t=1/J; %Size of one time step

```

```

17
18 N=40;
19 %Number of particles
20 cforce = 0.1;
21 %the force that draw particles together
22 e=0.5;
23 %e is eta the noise parameter, whose maximum value is 2*
    pi
24
25 r=1;
26
27 %The radius of influence of a particle
28
29 L=10;
30 %L is the size of the domain on which the particles can
    move
31
32 v=0.5; %v is the speed at which the particles move
33
34 % x(i,j) gives the x coordinate of the ith particle at
    time j
35 x=zeros(N,J+1);
36 x(:,1)=L*rand(N,1); %define initial x coordiantes of all
    particles
37
38 % y(i,j) gives the y coordinate of the ith particle at
    time j
39 y=zeros(N,J+1);
40 y(:,1)=L*rand(N,1); %define initial y coordiantes of all
    particles
41
42 % T(i,j) gives the angle with the x axis of the
    direction of motion of the ith
43 % particle at time j
44 T=zeros(N,J+1);
45 T(:,1)=2*pi*rand(N,1); %define initial direction of all
    particles
46 k = 4;
47 %For all time steps
48 for j=1:1:J %iterate in time
49     %For each particle
50     D = distance(x,y,j,N);

```

```

51 [B,I] = sort(D);
52 near_n = I(2:k+1,:); %found our nearest 4 neighbours
    for each one
53 xc = mean(x(:,j)); %center of mass x
54 yc = mean(y(:,j)); %center of mass y
55
56 for i=1:1:N
57
58     %get the current distance vector
59     neighbour = near_n(:,i);
60     tn = []; % the neighbour direction
61     for m = 1:1:length(neighbour)
62         tn(m) = T(neighbour(m),j);
63     end
64
65     fcenter = atan2(yc-y(i,j),xc-x(i,j));
66     %location of center
67     ss = sum(sin(tn)) + sin(fcenter)*cforce;
68     sc = sum(cos(tn)) + cos(fcenter)*cforce;
69
70     S = atan2(ss,sc);
71
72
73
74     T(i,j+1)=S+e*(rand-0.5); %adds noise to the
        measured angle
75
76     x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
        the particles' x-coordinates
77     y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
        the particles' y coordinates
78
79     % Jumps from the right of the box to the
        left or vice versa
80     x(i,j+1)=mod(x(i,j+1),L);
81
82     %Jumps from the top of the box to the bottom
        or vice versa
83     y(i,j+1)=mod(y(i,j+1),L);
84
85     %Plot particles
86

```

```

87         if makemovie
88             if abs(x(i,j)-x(i,j+1))<v & abs(y(i,j)-y
               (i,j+1))<v
89                 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
90                 axis([0 L 0 L]);
91                 hold on
92                 plot(x(i,j+1) ,y(i,j+1), 'k.', '
                   markersize',10)
93                 xlabel('X position')
94                 ylabel('Y position')
95
96             end
97         end
98
99     end
100     if makemovie
101         hold off
102         M(j)=getframe; %makes a movie fram from the plot
103
104         %movien = addframe(movien,M(j)); %adds this
               movie fram to the movie
105     end
106
107 end
108
109
110 %movien = close(movien); %finishes the movie

```

- polarization

```

1 function [x,y,T,op] = polarization(J,N,e,L,movie,k,cf)
2 %% this function simulate the particle movement
3 % J(200) total time steps 200
4 % N(40) Number of particles 40
5 % e(0.5) is eta the noise parameter, whose maximum value
   is 2*pi, 0.5
6 % L(10) is the size of the domain on which the particles
   can move

```

```

7  % movie(1 or 0) to make movie or not
8  % k(4) is the influence neighbours
9  % cf is the force effect towards the center of mass
10
11
12  %Set up movie
13  %fig=figure;
14  makemovie = movie;
15  %movien = avifile('Vicsekmovie','FPS',3,'compression','
      none')
16  UJ=1;
17  %Rate at which film is updated
18
19  t=1/J; %Size of one time step
20
21
22  r=1;%The radius of influence of a particle
23
24  v=0.5; %v is the speed at which the particles move
25
26  % x(i,j) gives the x coordinate of the ith particle at
      time j
27  x=zeros(N,J+1);
28  x(:,1)=L*rand(N,1); %define initial x coordiantes of all
      particles
29
30  % y(i,j) gives the y coordinate of the ith particle at
      time j
31  y=zeros(N,J+1);
32  y(:,1)=L*rand(N,1); %define initial y coordiantes of all
      particles
33
34  % T(i,j) gives the angle with the x axis of the
      direction of motion of the ith
35  % particle at time j
36  T=zeros(N,J+1);
37  op = [];
38  T(:,1)=2*pi*rand(N,1); %define initial direction of all
      particles
39
40  ssin = sum(sin(T(:,1)));
41  scos = sum(cos(T(:,1)));

```

```

42 op(1) = (1/N) * sqrt(ssin^2 + scos^2);
43 %For all time steps
44 for j=1:1:J %iterate in time
45     %For each particle
46     D = distance(x,y,j,N);
47     [B,I] = sort(D);
48     near_n = I(2:k+1,:); %found our nearest 4 neighbours
49     for each one
50         xc = mean(x(:,j)); %center of mass x
51         yc = mean(y(:,j)); %center of mass y
52
53     for i=1:1:N
54
55         %get the current distance vector
56         neighbour = near_n(:,i);
57         tn = []; % the neighbour direction
58         for m = 1:1:length(neighbour)
59             tn(m) = T(neighbour(m),j);
60         end
61
62         fcenter = atan2(yc-y(i,j),xc-x(i,j));
63         %location of center
64         ss = sum(sin(tn)) + sin(fcenter)*cf;
65         sc = sum(cos(tn)) + cos(fcenter)*cf;
66
67         S = atan2(ss,sc);
68
69         T(i,j+1)=S+e*(rand-0.5); %adds noise to the
70             measured angle
71
72         x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
73             the particles' x-coordinates
74         y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
75             the particles' y coordinates
76
77         % Jumps from the right of the box to the
78             left or vice versa
79         x(i,j+1)=mod(x(i,j+1),L);
80
81         %Jumps from the top of the box to the bottom
82             or vice versa

```



```

78         y(i,j+1)=mod(y(i,j+1),L);
79
80         %Plot particles
81
82         if makemovie
83             if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-
84                 y(i,j+1))<v
85                 plot([x(i,j), x(i,j+1)], [y(i,j)
86                     ,y(i,j+1)], 'k-', 'markersize',
87                     4) %plots the first half of
88                         the particles in black
89                 axis([0 L 0 L]);
90                 hold on
91                 plot(x(i,j+1), y(i,j+1), 'k.', '
92                     markersize',10)
93                 xlabel('X position')
94                 ylabel('Y position')
95
96             end
97         end
98
99         ssin = sum(sin(T(:,j+1)));
100        scos = sum(cos(T(:,j+1)));
101        op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);
102
103        if makemovie
104            hold off
105            M(j)=getframe; %makes a movie fram from the plot
106
107            %movien = addframe(movien,M(j)); %adds this
108                movie fram to the movie
109        end
110
111    end
112
113    %plot([1:1:J+1], op)
114
115    %movien = close(movien); %finishes the movie
116    end

```

- steady state alignment

```

1  clear
2  close all
3
4  N = 40;
5  J = 200;
6  L = 10;
7  movie = 0;
8  cf = 0.1;
9  erange = 0.1:0.1:6;
10 reps = 10;
11
12
13 for k = 1:1:N-1
14     k
15     yy = 0;
16     for e = erange
17         yy = yy+1;
18         steady_a(k,yy) = 0;
19         for r = 1:1:reps
20
21             [x,y,T,op] = polarization(J,N,e,L,movie,k,cf
22             );
23             sizeop = length(op);
24             ssalign = mean(op(round(sizeop/2):end));
25             steady_a(k,yy) = steady_a(k,yy) + ssalign;
26         end
27     end
28     steady_a(k,yy) = steady_a(k,yy)/reps;
29
30
31 save('ssatry2.mat')
```

- hunting function

```

1
2 function [x,y,T,op,s1,s2] = test43_hunt(J,N,e,L,movie,k1
3     ,k2,cf)
4 %% this function simulate the particle movement
5 % J(200) total time steps 200
```

```

5 % N(40) Number of particles 40
6 % e(0.5) is eta the noise parameter, whose maximum value
   is 2*pi, 0.5
7 % L(10) is the size of the domain on which the particles
   can move
8 % movie(1 or 0) to make movie or not
9 % k1 is the influence neighbours for the first group
10 % k2 is the influence neighbours for the second group
11 % cf is the force effect towards the center of mass
12 %J=200;N=40;e=0.5;L=10;movie=1;k1=10;k2=3;cf=0;
13 %Set up movie
14
15 figure('position', [0, 0, 700, 600])
16 makemovie = movie;
17 if makemovie == 1
18 myv = VideoWriter('hunting5.avi');
19 myv.FrameRate = 10;
20 open(myv);
21 set(gca, 'nextplot', 'replacechildren');
22 end
23
24
25 %movien = avifile('Vicsekmovie', 'FPS', 3, 'compression', '
   none')
26 UJ=1;
27 %Rate at which film is updated
28
29 t=1/J; %Size of one time step
30
31 r=0.5;%The radius of eating
32 fast = 1.2;
33
34 v=0.5; %v is the speed at which the particles move
35 vp = v * fast;
36
37 % x(i,j) gives the x coordinate of the ith particle at
   time j
38 x=zeros(N,J+1);
39 x(:,1)=L*rand(N,1); %define initial x coordiantes of all
   particles
40 px = [];
41 px(1) = L*rand();%predator x

```

```

42
43 % y(i,j) gives the y coordinate of the ith particle at
    time j
44 y=zeros(N,J+1);
45 y(:,1)=L*rand(N,1); %define initial y coordiantes of all
    particles
46 py = [];
47 py(1) = L*rand(); %predator y
48
49 % T(i,j) gives the angle with the x axis of the
    direction of motion of the ith
50 % particle at time j
51 T=zeros(N,J+1);
52 op = [];
53 T(:,1)=2*pi*rand(N,1); %define initial direction of all
    particles
54 dirp = [];
55 dirp(1) = 2*pi*rand();
56
57 ssin = sum(sin(T(:,1)));
58 scos = sum(cos(T(:,1)));
59 op(1) = (1/N) * sqrt(ssin^2 + scos^2);
60 %For all time steps
61 for j=1:1:J %iterate in time
62     %For each particle
63     D = distance(x,y,j,N);
64     [B,I] = sort(D);
65     big_n = I(2:k1+1,:); %found our nearest 10
        neighbours for each one
66     small_n = I(2:k2+1,:); %found our nearest 3
        neighbours for each one
67     xc = nanmean(x(:,j)); %center of mass x
68     yc = nanmean(y(:,j)); %center of mass y
69
70     dirp(j+1) = atan2(yc - py(j), xc - px(j));
71
72     px(j+1) = px(j) + vp * cos(dirp(j+1));
73     py(j+1) = py(j) + vp * sin(dirp(j+1));
74
75     for i=1:1:round(N/2)
76         %the group that depends on larger number of
            neighbours

```

```

77
78 %get the current distance vector
79 neighbour = big_n(:,i);
80 tn = []; % the neighbour direction
81 for m = 1:1:length(neighbour)
82     tn(m) = T(neighbour(m),j);
83 end
84
85
86 fcenter = 0;
87 %location of center
88 ss = nansum(sin(tn)) + sin(fcenter)*cf;
89 sc = nansum(cos(tn)) + cos(fcenter)*cf;
90
91 S = atan2(ss,sc);
92
93 T(i,j+1)=S+e*(rand-0.5); %adds noise to the
    measured angle
94
95 x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
    the particles' x-coordinates
96 y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
    the particles' y coordinates
97
98 % Jumps from the right of the box to the
    left or vice versa
99 x(i,j+1)=mod(x(i,j+1),L);
100
101 %Jumps from the top of the box to the bottom
    or vice versa
102 y(i,j+1)=mod(y(i,j+1),L);
103 end
104
105
106
107 for i=round(N/2)+1:1:N
108     %the group that depends on small number of
        neighbours
109
110     %get the current distance vector
111     neighbour = small_n(:,i);
112     tn = []; % the neighbour direction

```

```

113         for m = 1:1:length(neighbour)
114             tn(m) = T(neighbour(m),j);
115         end
116
117
118
119         fcenter = atan2(yc-y(i,j),xc-x(i,j));
120         fcenter = 0;
121         %location of center
122         ss = nansum(sin(tn)) + sin(fcenter)*cf;
123         sc = nansum(cos(tn)) + cos(fcenter)*cf;
124
125         S = atan2(ss,sc);
126
127         T(i,j+1)=S+e*(rand-0.5); %adds noise to the
            measured angle
128
129         x(i,j+1)=x(i,j)+v*cos(T(i,j+1)); %updates
            the particles' x-coordinates
130         y(i,j+1)=y(i,j)+v*sin(T(i,j+1)); %updates
            the particles' y coordinates
131
132         % Jumps from the right of the box to the
            left or vice versa
133         x(i,j+1)=mod(x(i,j+1),L);
134
135         %Jumps from the top of the box to the bottom
            or vice versa
136         y(i,j+1)=mod(y(i,j+1),L);
137     end
138
139     %Plot particles
140     Dp = distance_predator(x,y,j,N,px,py);
141
142     for ii = 1:1:N
143
144         if Dp(ii) < r
145             x(ii,j+1) = NaN;
146             y(ii,j+1) = NaN;
147         end
148
149     end

```

```

150
151
152     if makemovie
153         for i= 1:1:round(N/2)
154             if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-y(i,
155                 j+1))<v
156                 plot([x(i,j), x(i,j+1)], [y(i,j)
157                     ,y(i,j+1)], 'g-', 'markersize',
158                     4) %plots the first half of
159                     the particles in black
160                 axis([0 L 0 L]);
161                 hold on
162                 %plot([px(j) px(j+1)], [py(j) py(j+1)
163                     ], 'r-', 'markersize', 4) %plot the
164                     predator
165                 plot(x(i,j+1), y(i,j+1), 'g.', '
166                     markersize', 10)
167                 %plot(px(j+1), py(j+1), 'rd', '
168                     markersize', 10)
169                 xlim([0 L])
170                 ylim([0 L])
171                 xlabel('X position')
172                 ylabel('Y position')
173                 title({'hunting game'; 'Green:
174                     movement depends on 10 neighbours
175                     '; 'Black: movement depends on 3
176                     neighbours'; 'Red: Predator'}, '
177                     FontSize', 16);
178
179             end
180         end
181
182     for i = round(N/2)+1:1:N
183         if abs(x(i,j)-x(i,j+1))<v && abs(y(i,j)-y(i,
184             j+1))<v
185             plot([x(i,j), x(i,j+1)], [y(i,j)
186                 ,y(i,j+1)], 'k-', 'markersize',
187                 4) %plots the first half of
188                 the particles in black
189             axis([0 L 0 L]);
190
191             plot([px(j) px(j+1)], [py(j) py(j+1)

```

```

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

183    end
184    end
185
186
187
188    end
189
190    %{
191    ssin = sum(sin(T(:, j+1)));
192    scos = sum(cos(T(:, j+1)));
193    op(j+1) = (1/N) * sqrt(ssin^2 + scos^2);
194    %}
195    if makemovie
196        frame = getframe(gcf);
197        writeVideo(myv, frame);
198
199        hold off
200        %pause(0.1)
201
202
203    M(j)=getframe; %makes a movie fram from the plot
204
205    %movien = addframe(movien, M(j)); %adds this
                                movie fram to the movie
206    end
207
208

```



```

209 end
210
211 if makemovie ==1
212 close(myv); %finishes the movie
213 end
214
215 for t = 1:1:J
216
217
218     s1(t) = N/2 - sum(isnan(x(1:round(N/2),t)));
219     s2(t) = N/2 - sum(isnan(x(round(N/2)+1:end,t)));
220
221 end
222
223 end

```

- add predator

```

1 clear
2 close all
3
4 J=200;N=40;e=0.5;L=10;movie=0;
5 k1=6;k2=0;cf=0;
6
7 for i = 1:1:50
8     [x,y,T,op,s1,s2] = test43_hunt(J,N,e,L,movie,k1,k2,cf);
9     ss1(i,:) = s1;
10    ss2(i,:) = s2;
11 end
12
13 save('run436.mat')

```