448 HW3 Report

Mingfu Liang, Student ID:3146919, NetID:MLQ4767

March 2019

1. Problem 1

In this problem we are going to simulate the Susceptible-Infected-Removed (SIR) Model by using Gillespie algorithm. We are going to find out how the infected rate r_{infect}, removing rate r_{remove} and the fraction of vaccinated p_{immune} will influence the exit time t_{exit} when no transitions occur any more in the system and the number N_D of individuals dying in the disease outbreak. In simulation, we are going to track the number of individual in each state which are susceptible, infected or removed as $N_S(t)$,$N_I(t)$ and $N_R(t)$ as the time going and we assume that the rate of infected for an individual is determined by the number of infected neighbors (eight neighbors for each individual) using periodic boundary conditions. In SIR model, each individual can only transit to one state in each iteration, which means that susceptible individual can only transit to be infected, infected individual can only transit to removed state and removed people will stay at the same state and will not change.

When I implement the simulation code in MATLAB, I get some many practical lessons when I try to speed up my code. Here is some of them that I should highlight:

- **Infected Neighbor:** When calculate the infected neighbor of each individual, I use the code as below which takes advantage of matrix operation in MATLAB and avoids the for loop in calculating the transition rates matrix.

Listing 1: Calculating the infected neighbor

```
posit = 1:L; % define the index variables
up_shift = circshift(posit,1); % shift the variables up one unit
down_shift = circshift(posit,-1); % shift the variables down one
    unit
% defind transition rates matrix
transit_rates = r_inject*( 0 + Infected_Neigh(up_shift, posit) ...
+Infected_Neigh(down_shift, posit) ...
+Infected_Neigh(posit, up_shift) ...
+Infected_Neigh(posit, down_shift) ...
+Infected_Neigh(up_shift, down_shift) ...
+Infected_Neigh(down_shift, down_shift) ...
  +Infected_Neigh(up_shift, up_shift) ...
+Infected_Neigh(down_shift, up_shift));
```

- **Gillespie algorithm:** When implement the Gillespie algorithm and to select the transition k to j with the desired probability, what I do is that I first stack the matrix to a vector using (:) so that I can utilize the *cumsum* in MATLAB to avoid the for loop in finding out the smallest j to change that satisfy that $\sum_{m=1,m\neq k}^{j} c_{km} \geqslant U_2 \sum_{m=1,m\neq k}^{N} c_{km}$ and reshape the vector back to matrix when I finish all the calculation. This implementation largely speed up my code as well.

Listing 2: Gillespie algorithm

```
sigma_c_km= sum(transit_rates(:)); %
tau = -log(rand())/sigma_c_km;
normedrates = transit_rates/sigma_c_km;
normedrates_vec = normedrates(:);
i_select=find(cumsum(normedrates_vec)>=rand(), 1);
area_mat_vec = area_mat(:);
if area_mat_vec(i_select) ==2
    area_mat_vec(i_select) =3;
    NR = NR +1;
    NI = NI - 1;
elseif area_mat_vec(i_select) ==1
    area_mat_vec(i_select) = 2;
    NI = NI + 1;
    NS = NS - 1;
end
area_mat = reshape(area_mat_vec,L,L);
```

All these speed up implementation are really helpful for sub-problem (c) since after I avoid all the for-loop, I only need around 1000 seconds to run for $p_{immune} = 0 : 0.02 : 0.4$, $N_{trials} = 100$ and $L = 100$.

1.1 Problem 1(a)

First let's see when starting with an initial condition in which only a single individual in the center of the system($i = 10, j = 10$) is infected ($L = 20$), how the t_{exit} will behave. With $N_{trials} = 1000$, we can see from the Figure(1) that most of the trials, the system exits at a very short time. This makes sense since by Gillespie algorithm we may randomly set the all the infected individuals to be removed at the very beginning of the simulation since the space of the system is not that large ($L = 20$). For the second peak which is concentrating around 15, this seems like a more regular pattern and we finally get that the mean of the t_{exit} is around 10.3601, and it will change slightly each time when you run the code but it should be around 10.

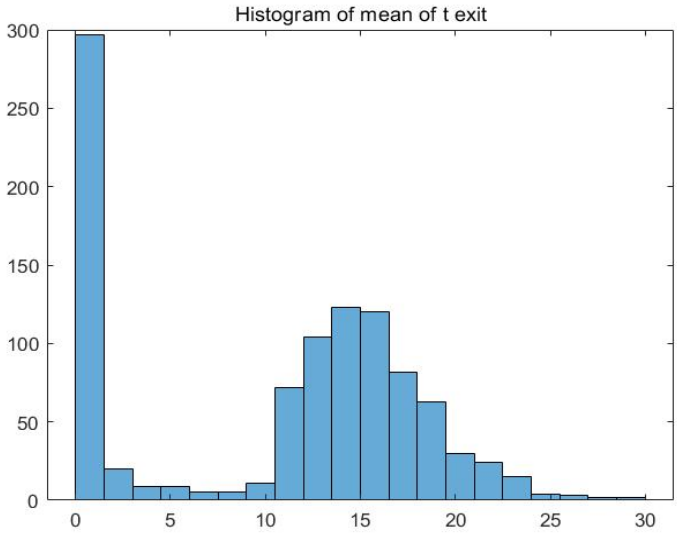

Figure 1: Histogram of t_{exit} where $N_{trial} = 1000$

1.2 Problem 1(b)

Then let's get more feeling about the system by investigating the dependence of its dynamics on r_{remove}. As shown below, Figure (2), (3), (4), (5), (6) and (7) are the representative snapshots when $r_{remove} = 0, 0.3, 0.6, 0.9, 1.2, 1.5$, which are selected after more than 30 repeated run and I choose them as the similar snapshots for each r_{remove}. For Figure (2), the color blue means that all the individuals are infected. This makes sense since based on the SIR model that each individual can only transit to one state each time, which means that susceptible individual can only transit to be infected, infected individual can only transit to removed state and removed people will stay at the same state and will not change, when the r_{remove} is equal to zero and by using the Gillespie algorithm, all the susceptible individual will be infected and will not be removed.

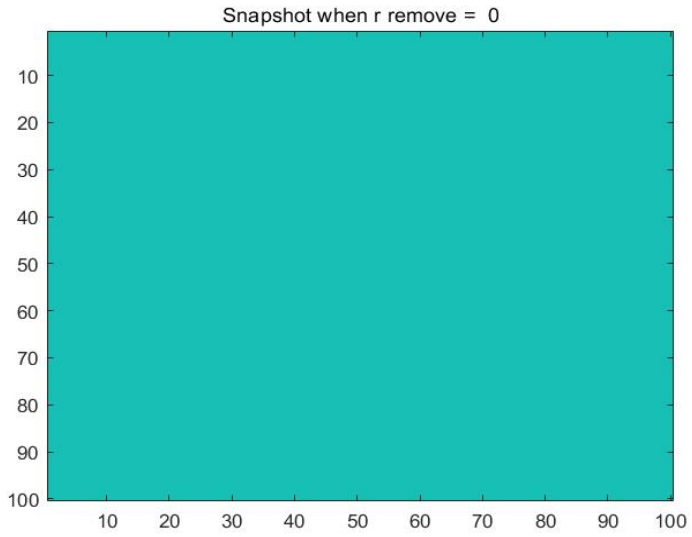

Figure 2: Representative snapshots when $r_{remove} = 0$

For Figure(3), (4), (5), (6),(7), the yellow colors means that the individual is removed and the purple color means that the individual is susceptible. We can see that as the r_{remove} increase from 0.3 to 1.5, the number of removed are decreasing. This also makes sense since when the r_{remove} is increasing, those infected individuals will be removed from the system with higher probability and the transition will stop much quickly, which leads to less injected individuals of the system. That's also the reason why separating the infected individuals from susceptible individuals can effectively avoid the disease outbreaks and transit between people in the real world.

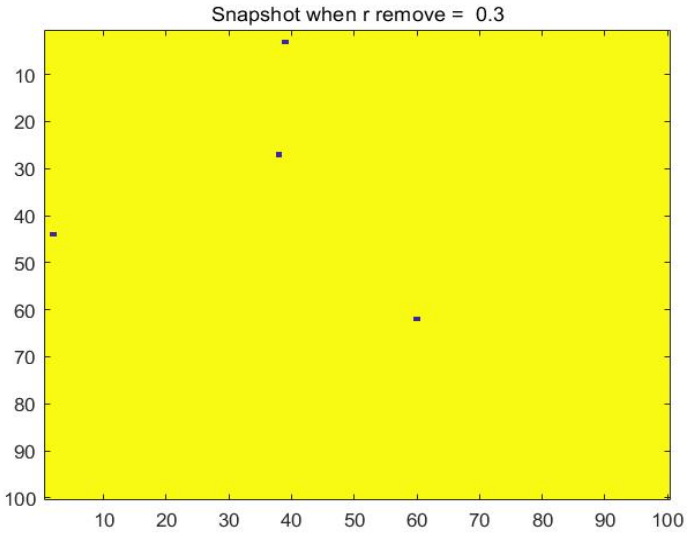

Figure 3: Representative snapshots when $r_{remove} = 0.3$

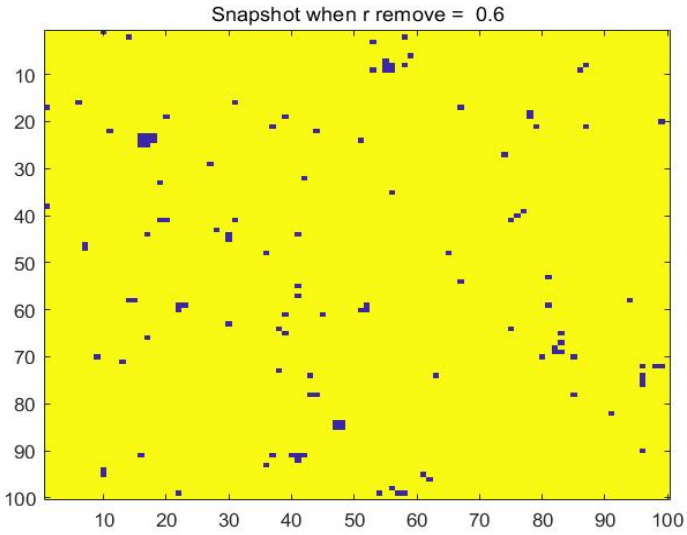

Figure 4: Representative snapshots when $r_{remove} = 0.6$

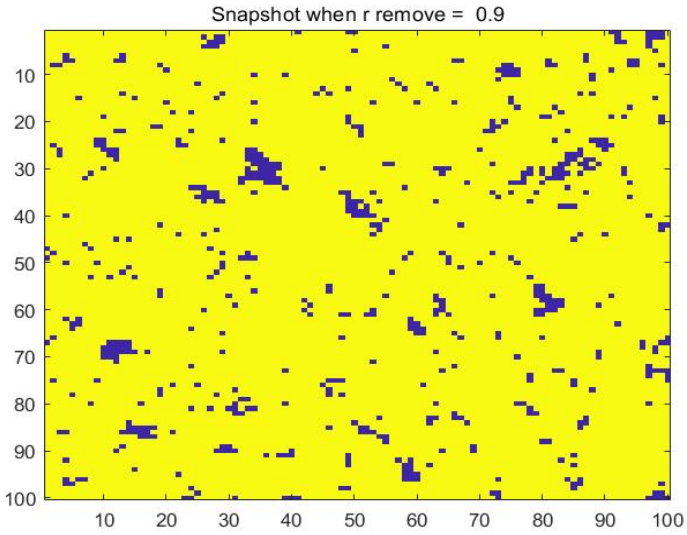

Figure 5: Representative snapshots when $r_{remove} = 0.8$

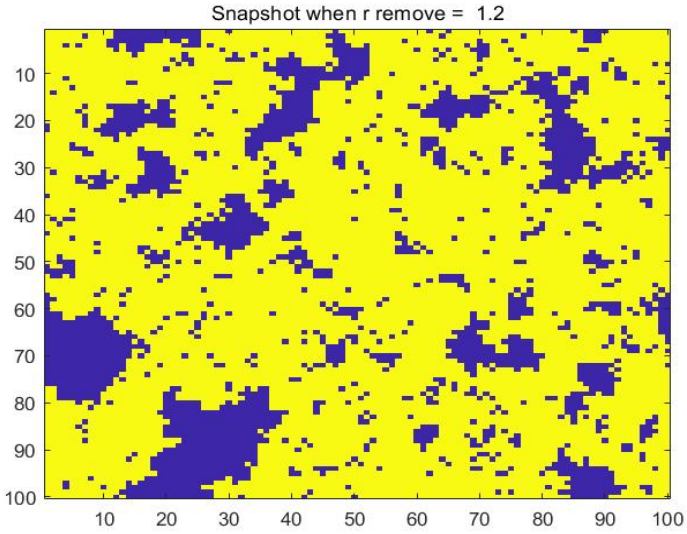

Figure 6: Representative snapshots when $r_{remove} = 1.2$

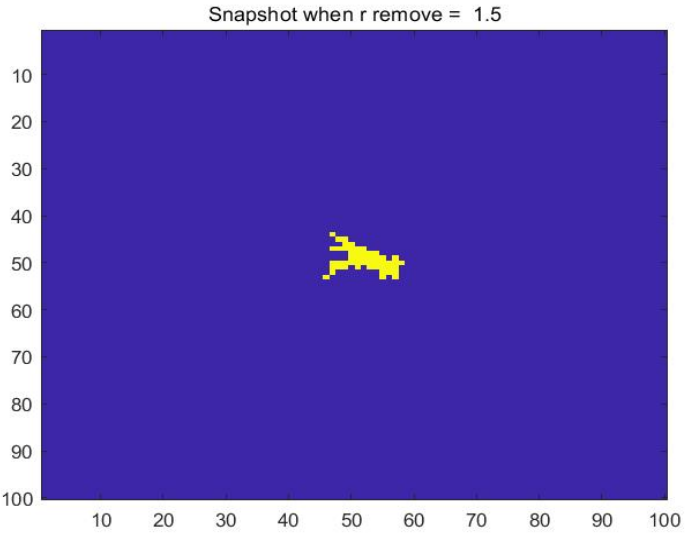

Figure 7: Representative snapshots when $r_{remove} = 1.5$

1.3 Problem 1(c)

Now let's consider a random fraction p_{immune} of individuals has been vaccinated, which makes them immune to the disease and see how these immune individuals will influence the transition of the disease. Here I choose $p_{immune} = 0 : 0.02 : 0.4$, which means that I do experiment for 21 different p_{immune} values and for each experiment I run for 100 realizations. From Figure (8) and (10) we can see that the mean and standard deviation (as error bars) of t_{exit} and the number N_D of individuals dying in the disease outbreak are decreasing as the p_{immune} is increasing. These phenomena make sense since based on the SIR model that each individual can only transit to one state each time, which means that susceptible individual can only transit to be infected, infected individual can only transit to removed state and removed people will stay at the same state and will not change, and by the Gillespie algorithm, the more people are immune to the disease, then there will be less probability for the disease to transit between individuals, which leads to less time to exit and less people to die. That's also the reason why immunizations are such significant in real life since the more the people have immunizations, the larger effect we will have in preventing the disease to transit and outbreak.

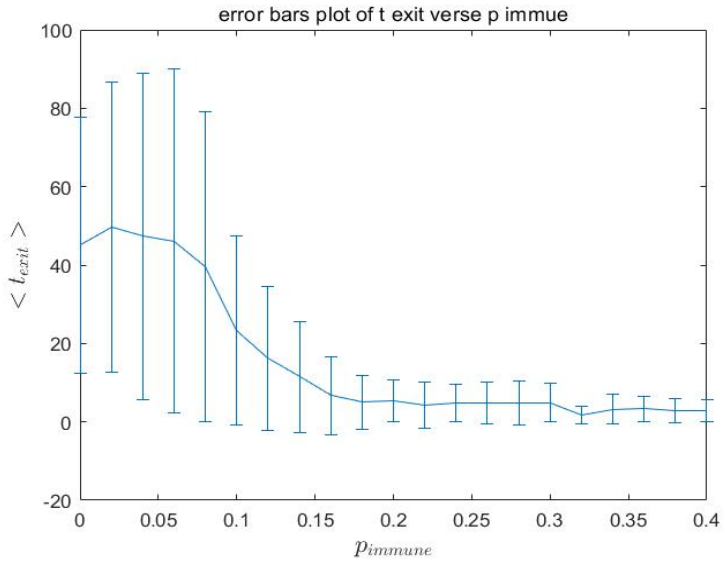

Figure 8: error bars plot of t_{exit} verse p_{immune} where $p_{immune} = 0 : 0.02 : 0.4$

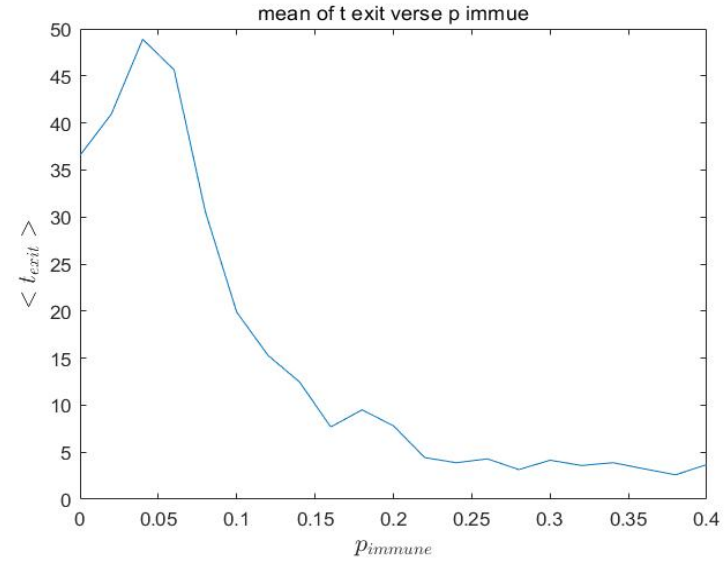

Figure 9: plot of mean of t_{exit} verse p_{immune} where $p_{immune} = 0 : 0.02 : 0.4$

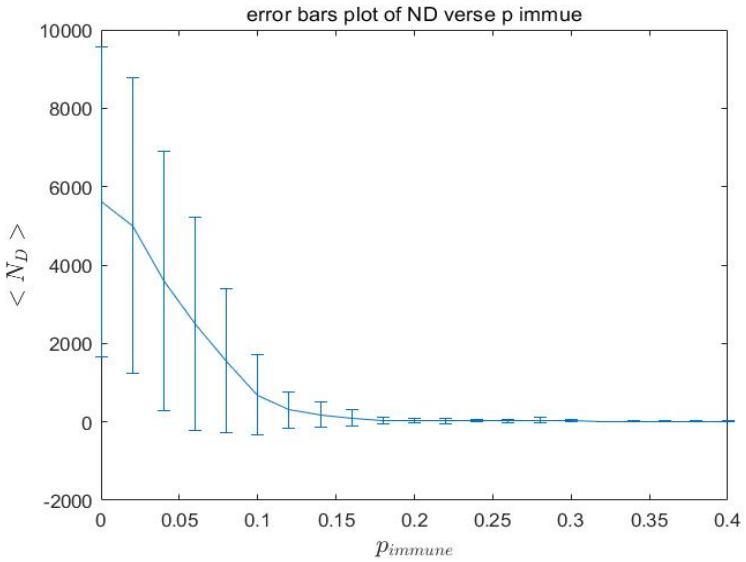

Figure 10: error bars plot of N_D verse p_{immune} where $p_{immune} = 0 : 0.02 : 0.4$

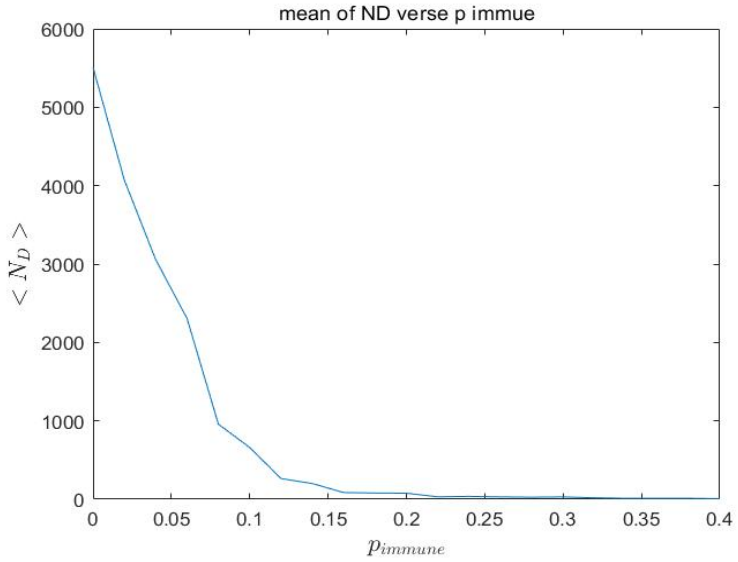

Figure 11: plot of mean of N_D verse p_{immune} where $p_{immune} = 0 : 0.02 : 0.4$

2. Part 1 MATLAB code

2.1 Problem 1(a)

```
%%%% Homework 3(a) %%%%%%%%
%%%% author: Mingfu Liang %%%%%%%%
%%%% date: 03/05/2019 %%%%%%%

function [t_exit]=MingfuLiang_HW3(n_trial,L)
tic
```

```
7
8  %L = 20;
9  %n_trial = 1000;
10 %L_2 = L*L;
11
12 %%%%%%%%%%% initialize the parameter %%%%%%%%%%%%%%%%%%%%%%%%%%
13
14 N_trials = n_trial;
15 r_inject = 0.5;
16 r_remove = 1.1;
17 N_S = zeros(1000,400); % define the vector to store the number of
       susceptible individual for each iteration
18 N_I = zeros(1000,400); % define the vector to store the number of infected
       individual for each iteration
19 N_R = zeros(1000,400); % define the vector to store the number of removed
       individual for each iteration
20 t_exit_mat = zeros(1000,1); % define the vector to store the exit time for
       each iteration
21 posit = 1: L; % define the index variables
22 up_shift = circshift(posit,1); % shift the variables up one unit
23 down_shift = circshift(posit,-1); % shift the variables down one unit
24
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26
27 for i =1: N_trials
28
29     % Starting with an initial condition in which only a single individual
          in the center of the
30     % system (i = 10; j = 10) is infected (L = 20)
31     % here denote 1 is susceptible, 2 is infected, 3 is removed
32
33     area_mat = ones(L, L); % define the matrix as the system
34     area_mat(10,10) = 2; % define the (10,10) in the system to be
          susceptiable
35     t_counter =1;
36     t_exit = 0;
37
38     % initialize the NS, NI, NR
39     NS = 399;
40     NI = 1;
```

```
41      NR = 0;
42
43      %%%%%%%%%%%% uncomment the code below so that you can watch a moive : )
44
45      %figure;
46      %h11=imagesc(area_mat);
47
48      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49
50          while NI~=0
51                          % define the infected neighbor matrix
52                          Infected_Neigh=area_mat==2;
53
54                          % define the transition rate matrix
55                          transit_rates = r_inject*( 0 + Infected_Neigh(
                                up_shift, posit) ...
56                           +Infected_Neigh(down_shift, posit) ...
57                           +Infected_Neigh(posit, up_shift) ...
58                           +Infected_Neigh(posit, down_shift)  ...
59                           +Infected_Neigh(up_shift, down_shift) ...
60                           +Infected_Neigh(down_shift, down_shift) ...
61                           +Infected_Neigh(up_shift, up_shift)  ...
62                           +Infected_Neigh(down_shift, up_shift));
63                          transit_rates(area_mat==2) =r_remove;
64                          transit_rates(area_mat==3) =0;
65
66              % define the sigma c_km, which is the sum of c_km for m =1, m~=j
67              sigma_c_km= sum(transit_rates(:));
68
69              % define the tau
70              tau = -log(rand())/sigma_c_km;
71
72              %define the normalized transition rate matrix
73              normedrates = transit_rates/sigma_c_km;
74              normedrates_vec = normedrates(:);
75
76              % find the smallest j satisfies the (sigma ckm from 1 to j) <=
77              % U2 * (sigma ckm from m =1 and m~=k)
78              i_select=find(cumsum(normedrates_vec)>=rand(), 1);
79
```

```
80              area_mat_vec = area_mat(:);% stack the matrix to vector
81
82              % update the NS, NI, NR
83              if area_mat_vec(i_select) ==2
84                  area_mat_vec(i_select) =3;
85                  NR = NR +1;
86                  NI = NI - 1;
87              elseif area_mat_vec(i_select) ==1
88                  area_mat_vec(i_select) = 2;
89                  NI = NI + 1;
90                  NS = NS - 1;
91              end
92
93              area_mat = reshape(area_mat_vec,L,L);% reshape vector into matrix
                    back
94
95          % update t_exit and store the stats at this iteration
96          t_exit = t_exit  + tau;
97          N_S(i,  t_counter) = NS;
98          N_I(i,  t_counter) = NI;
99          N_R(i,  t_counter) = NR;
100         t_counter = t_counter +1;
101         end
102
103     t_exit_mat(i,1)=t_exit;
104
105     %%%%%%%%%%%% uncomment the code here so that you can watch a moive : )
106
107     %set(h11,'CData',area_mat)
108     %pause(0.01);
109
110     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111
112 end
113 figure;
114 histogram(t_exit_mat,20);
115 title1 = 'Histogram of mean of t exit';
116 title(title1);
117 mean_t_exit = mean(t_exit_mat);
118 text = ['mean of t_exit is ', num2str(mean_t_exit),' \n'];
```

```
fprintf(text);
t_exit = t_exit_mat;
toc
end
```

2.2 Problem 1(b)

```
%%%% Homework 3(b) %%%%%%%%
%%%% author: Mingfu Liang %%%%%%%
%%%% date: 03/05/2019 %%%%%%
%%%%%%%%%%% initialize the parameter %%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
L =100;
L_2 = L*L;
N_trials = 1;
r_inject = 0.5;
N_S = zeros(6,1); % define the vector to store the number of susceptible
    individual for each iteration
N_I = zeros(6,1); % define the vector to store the number of infected
    individual for each iteration
N_R = zeros(6,1); % define the vector to store the number of removed
    individual for each iteration
t_exit_mat = zeros(6,1); % define the vector to store the exit time for each
     iteration
k=1;
posit = 1: L; % define the index variables
up_shift = circshift(posit,1); % shift the variables up one unit
down_shift = circshift(posit,-1); % shift the variables down one unit
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for r_remove = 0:0.3:1.5
    % Starting with an initial condition in which only a single individual
       in the center of the
    % system (i = 10; j = 10) is infected (L = 20)
    % here denote 1 is susceptible, 2 is infected, 3 is removed
     area_mat = ones(L, L); % define the matrix as the system
     area_mat(50,47:53) = 2; % define the (50,47:53) in the system to be
```

```
        susceptiable
    t_counter =1;
    t_exit = 0;

    %%%%%%%%%%% uncomment the code below so that you can watch a moive : )

    %figure;
    %h11=imagesc(area_mat);
    %title1 = ['snapshot of r remove = ',num2str(r_remove)];
    %title(title1)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % initialize the NS, NI, NR
    NS = L_2-7;
    NI = 7;
    NR = 0;

        while NI~=0
            if NI == L_2
                break
            end
            % define the infected neighbor matrix
            Infected_Neigh=area_mat==2;
            % define the transition rate matrix
            transit_rates = r_inject*( 0 + Infected_Neigh(up_shift, posit)
                ...
                            +Infected_Neigh(down_shift, posit) ...
                            +Infected_Neigh(posit, up_shift) ...
                            +Infected_Neigh(posit, down_shift)  ...
                            +Infected_Neigh(up_shift, down_shift) ...
                            +Infected_Neigh(down_shift, down_shift) ...
                            +Infected_Neigh(up_shift, up_shift)  ...
                            +Infected_Neigh(down_shift, up_shift));
             transit_rates(area_mat==2) =r_remove;
             transit_rates(area_mat==3) =0;

            % define the sigma c_km, which is the sum of c_km for m =1, m~=j
            sigma_c_km= sum(transit_rates(:));

```

```
68          % define the random number and tau
69          U1 = rand();
70          tau = -log(U1)/sigma_c_km;
71          U2 = rand();
72          sigma_ckm = 0;
73
74          %define the normalized transition rate matrix
75          normedrates = transit_rates./sigma_c_km;
76          normedrates_vec = normedrates(:);
77
78          % find the smallest j satisfies the (sigma ckm from 1 to j) <=
79          % U2 * (sigma ckm from m =1 and m~=k)
80          i_select=min(find(cumsum(normedrates_vec)>=U2));
81
82          area_mat_vec = area_mat(:); % stack the matrix to vector
83
84          % update the NS, NI, NR
85          if area_mat_vec(i_select) ==2
86              area_mat_vec(i_select) =3;
87              NR = NR +1;
88              NI = NI - 1;
89          elseif area_mat_vec(i_select) ==1
90              area_mat_vec(i_select) = 2;
91              NI = NI + 1;
92              NS = NS - 1;
93          end
94          area_mat = reshape(area_mat_vec,L,L); % reshape vector into
                matrix back
95
96      % update t_exit and store the stats at this iteration
97      t_exit = t_exit  + tau;
98      N_S(k,  t_counter) = NS;
99      N_I(k,  t_counter) = NI;
100     N_R(k,  t_counter) = NR;
101     t_counter = t_counter +1;
102
103     %%%%%%%%%%%% uncomment the code here so that you can watch a moive
            : )
104
105     %set(h11,'CData',area_mat)
```

```
        %keyboard
        %pause(0.01);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        end

t_exit_mat(k,1)=t_exit;
k = k+1;
figure;
imagesc(area_mat)
mytitle1=['Snapshot when r remove =  ',num2str(r_remove)];
title(mytitle1);
end
toc
```

2.3 Problem 1(c)

```
%%%%% Homework 3(c) %%%%%%%%%
%%%%% author: Mingfu Liang %%%%%%%%
%%%%% date: 03/05/2019 %%%%%%%


tic
p_immune_count = 1;
range = 0.4/0.02 +1;
mean_t_exit = zeros(range,1); % define the vector to store the mean of
    t_exit for each p_immune
std_t_exit = zeros(range,1); % define the vector to store the standard
    derivation for each p_immune
mean_ND = zeros(range,1); % define the vector to store the mean of N_{D} for
     each p_immune
std_ND = zeros(range,1); % define the vector to store the standard
    derivation of N_{D} for each p_immune

for p_immune = 0:0.02:0.4

text = ['p_immune = ', num2str(p_immune), ' \n'];
fprintf(text);

%%%%%%%%%%%% initialize the parameter %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
20
21 L =100;
22 L_2 = L*L;
23 N_trials = 100;
24 r_inject = 0.5;
25 r_remove = 1.1;
26 N_S = zeros(N_trials,1);% define the vector to store the number of
      susceptible individual for each iteration
27 N_I = zeros(N_trials,1); % define the vector to store the number of infected
       individual for each iteration
28 N_R = zeros(N_trials,1); % define the vector to store the number of removed
      individual for each iteration
29 t_exit_mat = zeros(N_trials,1); % define the vector to store the exit time
      for each iteration
30 N_D_mat =zeros(N_trials,1); % define the vector to store the number of dying
       people for each iteration
31 posit = 1: L; % define the index variables
32 up_shift = circshift(posit,1); % shift the variables up one unit
33 down_shift = circshift(posit,-1); % shift the variables down one unit
34 %%%% make p_immune* L^2 number of people to immune
35 [i_ind,j_ind]=ind2sub([L, L],randperm(L^2)); % generate a random permutation
       of all index
36
37 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38
39 for i =1: N_trials
40
41     % Starting with an initial condition in which only a single individual
          in the center of the
42     % system (i = 10; j = 10) is infected (L = 20)
43     % Here denote 1 is susceptible, 2 is infected, 3 is removed
44
45     immune_num = L_2 * p_immune; % calculate how many individuals will be
          immune
46     area_mat = ones(L, L);  % define the matrix as the system
47
48     % set individuals to be immune randomly
49     for select_ind = 1:immune_num
50                 i_select = i_ind(select_ind);
51                 j_select = j_ind(select_ind);
```

```
52                 area_mat(i_select, j_select) =3;
53         end
54
55         area_mat(50,50) = 2;
56         t_counter =1;
57         t_exit = 0;
58         NS = L_2 - immune_num;
59         NI = 1;
60         NR = immune_num;
61
62         %%%%%%%%%%%%% uncomment the code below so that you can watch a moive : )
63
64         %transit_rates = zeros(L, L);
65         %figure;
66         %h11=imagesc(area_mat);
67
68         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69
70             while NI~=0
71
72                 % define the infected neighbor matrix
73                 Infected_Neigh=area_mat==2;
74
75                 % define the transition rate matrix
76                 transit_rates = r_inject*( 0 + Infected_Neigh(up_shift, posit)
                       ...
77                                 +Infected_Neigh(down_shift, posit) ...
78                                 +Infected_Neigh(posit, up_shift) ...
79                                 +Infected_Neigh(posit, down_shift)  ...
80                                 +Infected_Neigh(up_shift, down_shift) ...
81                                 +Infected_Neigh(down_shift, down_shift) ...
82                                 +Infected_Neigh(up_shift, up_shift)  ...
83                                 +Infected_Neigh(down_shift, up_shift));
84                 transit_rates(area_mat==2) =r_remove;
85                 transit_rates(area_mat==3) =0;
86
87                 % define the random number and tau
88                 sigma_c_km= sum(transit_rates(:));
89                 U1 = rand();
90                 tau = -log(U1)/sigma_c_km;
```

```
91              U2 = rand();
92              sigma_ckm = 0;
93
94              %define the normalized transition rate matrix
95              normedrates = transit_rates./sigma_c_km;
96              normedrates_vec = normedrates(:);
97
98              % find the smallest j satisfies the (sigma ckm from 1 to j) <=
99              % U2 * (sigma ckm from m =1 and m~=k)
100             i_select=find(cumsum(normedrates_vec)>=U2, 1 );
101
102             area_mat_vec = area_mat(:);% stack the matrix to vector
103
104             % update the NS, NI, NR
105             if area_mat_vec(i_select) ==2
106                 area_mat_vec(i_select) =3;
107                 NR = NR +1;
108                 NI = NI - 1;
109             elseif area_mat_vec(i_select) ==1
110                 area_mat_vec(i_select) = 2;
111                 NI = NI + 1;
112                 NS = NS - 1;
113             end
114             area_mat = reshape(area_mat_vec,L,L);% reshape vector into
                    matrix back
115
116         % update t_exit and store the stats at this iteration
117         t_exit = t_exit  + tau;
118         N_S(i,  t_counter) = NS;
119         N_I(i,  t_counter) = NI;
120         N_R(i,  t_counter) = NR;
121         t_counter = t_counter +1;
122         end
123
124     t_exit_mat(i,1)=t_exit;
125     N_D = N_R(i,t_counter-1) - immune_num; % calculate the number of dying
            individual
126     N_D_mat(i,1) = N_D;
127
128     %%%%%%%%%%%% uncomment the code here so that you can watch a moive : )
```

```
    %set(h11,'CData',area_mat)
    %pause(0.01);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

mean_t_exit(p_immune_count,1)=mean(t_exit_mat);
std_t_exit(p_immune_count,1)=std(t_exit_mat);
mean_ND(p_immune_count,1)=mean(N_D_mat);
std_ND(p_immune_count,1)=std(N_D_mat);
p_immune_count = p_immune_count + 1;
end
toc

t_range = 0:0.02:0.4;
figure;
plot(t_range,mean_t_exit);
title2=['mean of t exit verse p immue'];
title(title2)
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
ylabel('$<t_{exit}>$','Interpreter','latex','FontSize',13)

figure;
plot(t_range,std_t_exit);
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
ylabel('$std(t_{exit})$','Interpreter','latex','FontSize',13)
title3=['standard deviation of t exit verse p immue'];
title(title3)

figure;
plot(t_range,mean_ND);
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
ylabel('$<N_{D}>$','Interpreter','latex','FontSize',13)
title4=['mean of ND verse p immue'];
title(title4)

figure;
plot(t_range,std_ND);
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
```

```
ylabel('$std(N_{D})$','Interpreter','latex','FontSize',13)
title5=['standard deviation of ND verse p immue'];
title(title5)

figure;
errorbar(t_range, mean_t_exit,std_t_exit);
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
ylabel('$<t_{exit}>$','Interpreter','latex','FontSize',13)
title6=['error bars plot of t exit verse p immue'];
title(title6)
figure;
errorbar(t_range, mean_ND,std_ND);
xlabel('$p_{immune}$','Interpreter','latex','FontSize',13)
ylabel('$<N_{D}>$','Interpreter','latex','FontSize',13)
title7=['error bars plot of ND verse p immue'];
title(title7)
```