FFR120, Simulation of Complex Systems
Homework 1

November 13, 2019

Ella Guiladi
930509-0822
guiladi@student.chalmers.se

1 Implementing the model

1. The plot of the random walk of one single agent is visualized in figure (1).

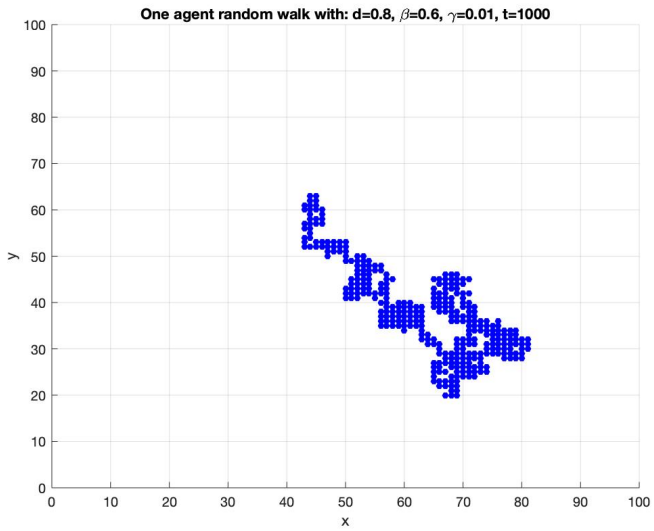


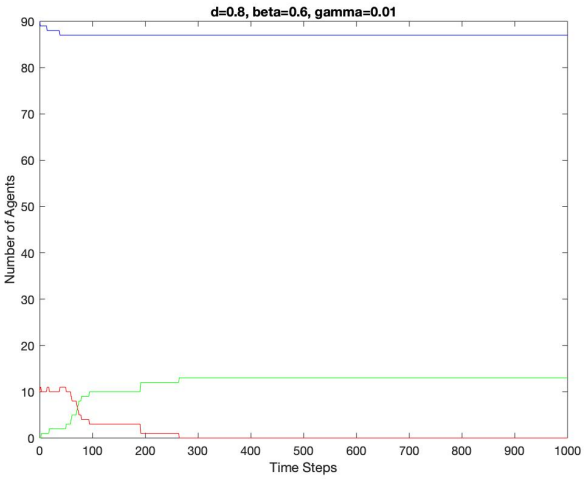
Figure 1: A random walk for one single agent with $d = 0.8$, $\beta = 0.6$ and $\gamma = 0.01$ for 1000 time steps.

2. The parameters was chosen to $\beta = 0.6$, the recovery rate $\gamma = 0.01$ and the diffusion rate was set to 0.8. The step size was chosen to 1000. The small number of agents were chosen to 100 and the number of infected agents were chosen to 10. The number of susceptible agents is given by the difference between the number of agents and the infected agents.

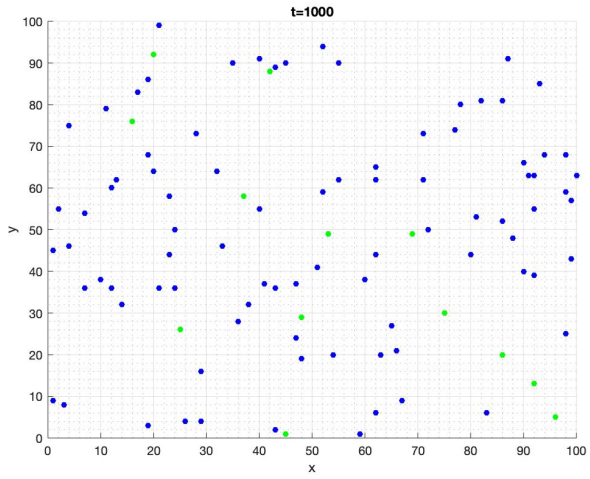
   The result is presented in figure (2) where one can see that the susceptible agents will start to decrease as they first become infected and then overtime recover. This is clearly visible in figure (2b) where the visual number of infected agents is zero and the number of recovered agents has increased. This argument can also be reinforced by figure (2a) where the graphs shows the increase and decrease of the number of agents over time. The number of susceptible agents are decreasing until they stagnate, which is when the number of infected agents reaches their static state. The number of recovered agents increases as the number of infected agents decreases.

3. The parameters and step size was chosen as before but with 1000 number of agents and 100 number of infected agents. From figure (3), one can see that it is possible to draw the same conclusion as for the case with the small amount of agents. As expected, the recovered agents dominate the population which is visualized in figure (3b). From figure (3a) one can see that the model works in the same way as the model presented in figure 10.1 from the course homework paper. The infected agents decrease, except for an small increase where the infection dominates. The susceptible agents decrease, which is correct since they either become infected or recovered. Finally, the recovered agents increase until reaching a static state where no more infected exists.

   This behaviour is due to the properties of the model, i.e when a susceptible agent and an infected agent land on the same lattice site, they have a certain probability of infecting the agent or recovering. After enough time this will thereby eventually lead to all agents being recovered.



(a) Screen- shot of 100 agents on a lattice after 1000 time steps



(b) Plot of the proportions of susceptible (blue), infected (red) and recovered (green) individuals in each state over time.

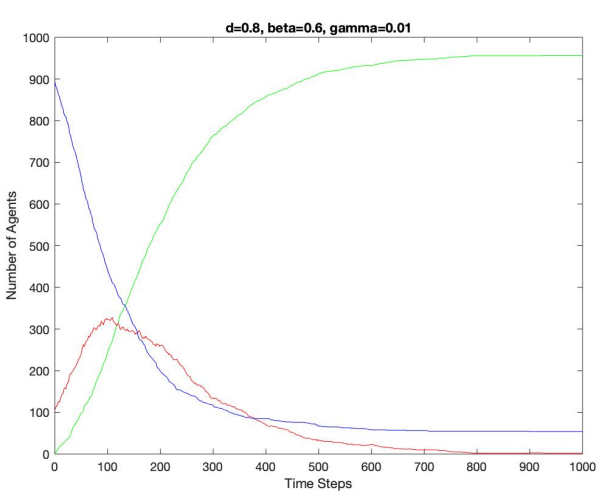
Figure 2: Plots for 100 agents with $d = 0.8$, $\beta = 0.6$ and $\gamma = 0.01$.

2 Presenting the two regimes of the model

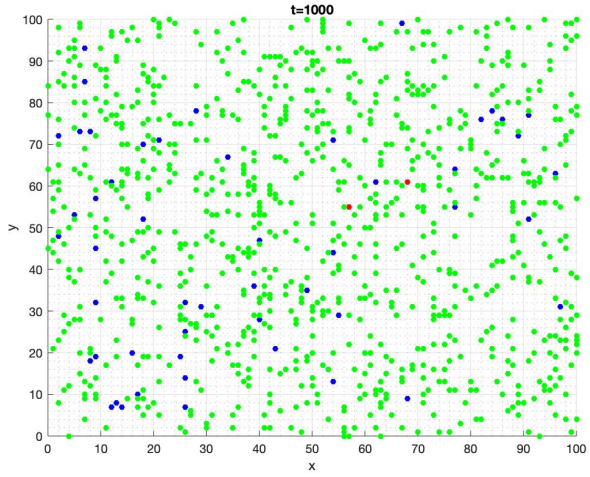
1. The most obvious way to present the two regimes is by adjusting the values for the infection rate and for the recovery rate. The step size was once again chosen to 1000, the number of agents were set to 1000 and the number of infected agents to 100.

   To visualize a population-wide disease spreading, one can set the infection rate $\beta = 0.9$, the recovery rate $\gamma = 0.001$. By looking at figure (4a), one can see that the the graph that presents the infected agents increases strongly and then decreases more slowly as the number of susceptible decreases into a static state. The number of recovered agents has a much slower increase, due to the low recovery rate γ. From figure (4b) one can see a large number of infected agents in comparison to figure (3b), implying a population-wide disease spread.

2. To visualize a limited disease spreading one can set the infection rate $\beta = 0.1$ and the recovery rate $\gamma = 0.1$. As expected, one can see from figure (5) that the limited disease spreading occurs. Since the susceptible agents need to be infected in order to get recovered, and the infection rate is so low, almost no agents will be infected. This leads to the infected agents going in to a static state, which is visualized in figure (5a). From figure (5b), it is also clear that the recovered agents have a small increase before the graph stagnates and the susceptible agents have a small decrease before the graph stagnates, leading to a majority of the susceptible agents.



(a) Screen- shot of 1000 agents on a lattice after 1000 time steps



(b) Plot of the proportions of susceptible (blue), infected (red) and recovered (green) individuals in each state over time.

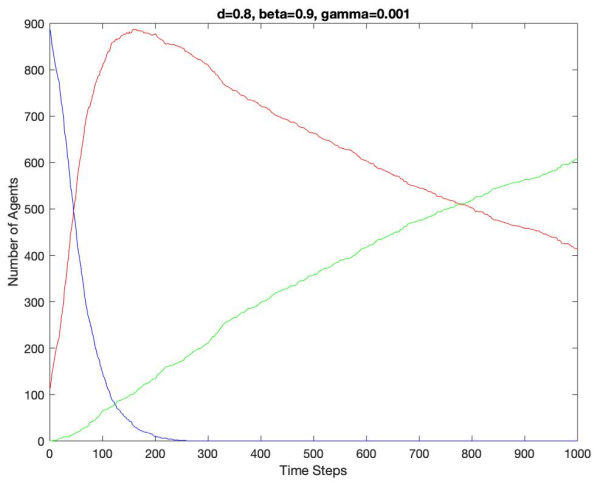
Figure 3: Plots for 1000 agents with $d = 0.8$, $\beta = 0.6$ and $\gamma = 0.01$.

3 The dependence of the epidemic threshold

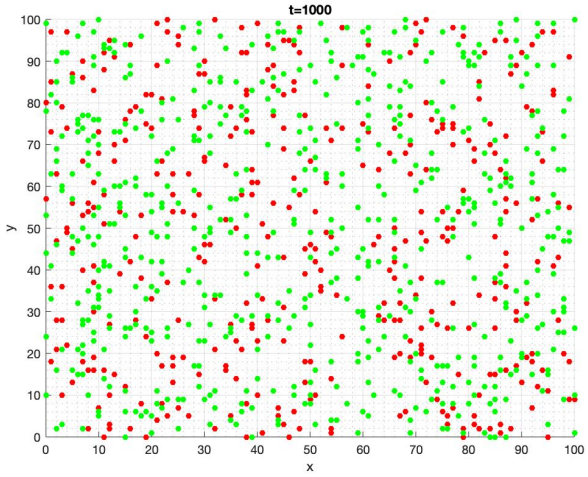
The plot presented in figure (6) shows the proportion of recovered agents for different β/γ with fixed infection rate β. The point where the increase of the graphs occur can be seen as the epidemic threshold. One can see that the threshold lies on different values of the ratio of the infection rate and recovery rate, depending on the value of β. This means that the epidemic threshold not only depends on the ratio but also on the parameters themselves.

When the ratio of β/γ is small, i.e β and γ has similar values, the number of recovered agents will also be small for both values of β. When the ratio increases, i.e γ decreases, one can see that the number of recovered agents start to increase. The increase is steeper for $\beta = 0.1$ than for $\beta = 0.9$, which is due to the lower probability of getting infected and the higher recovery rate.

In figure (6) β has the values 0.1 and 0.9 and the values of γ are 20 uniformly distributed number in the range $0.0007 \cdot 1.398^{1:20}$. The number of time steps are chosen to 1000 as well as the number of agents and the number of infected agents are chosen to 10. The number of runs are chosen to 2 in figure (6a) and 4 in figure (6b).

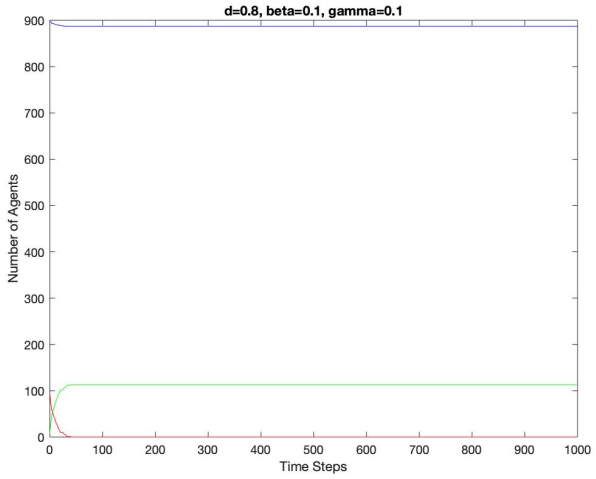


(a) Screen- shot of 1000 agents on a lattice after 1000 time steps

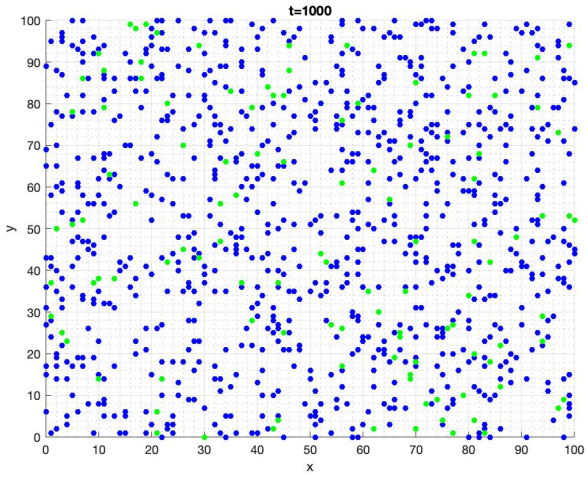


(b) Plot of the proportions of susceptible (blue), infected (red) and recovered (green) individuals in each state over time.

Figure 4: Plots of a population-wide disease spreading for 1000 agents with $d = 0.8$, $\beta = 0.9$ and $\gamma = 0.001$.

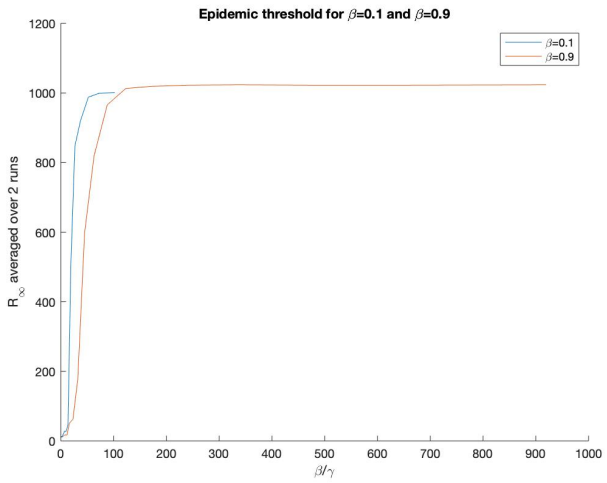


(a) Screen- shot of 1000 agents on a lattice after 1000 time steps

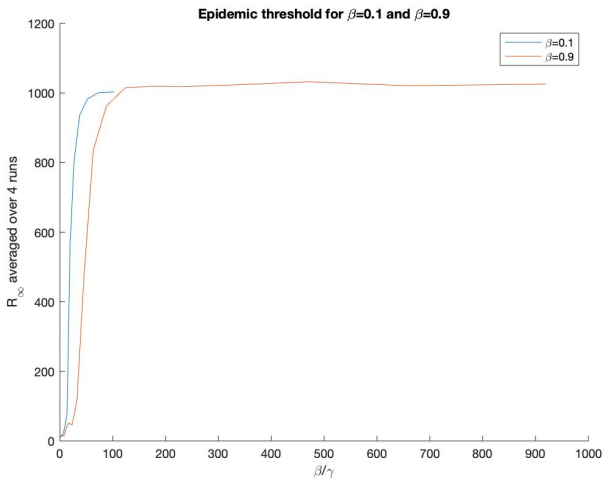


(b) Plot of the proportions of susceptible (blue), infected (red) and recovered (green) individuals in each state over time.

Figure 5: Plots of a limited disease spreading for 1000 agents with $d = 0.8$, $\beta = 0.1$ and $\gamma = 0.1$.



(a) Recovered agents over 2 runs against β/γ



(b) Recovered agents over 4 runs against β/γ

Figure 6: Plots comparing two data sets and showing respective thresholds for 1000 time steps for 2 respective 4 runs.

4 Important features of phase diagram

In figure (7) β are set to 9 uniformly distributed values between $0.1 - 0.9$ and values of γ are 20 uniformly distributed number in the range $0.0007 \cdot 1.398^{1:20}$. The number of time steps are chosen to 1000 as well as the number of agents and the number of infected agents are chosen to 10. The number of runs are chosen to 2 and the number of susceptible agents are chosen as before.

In the ODE- version of the SIR model, only the ratio β/γ matters for the behaviour in the model. It is clear from the 3D phase diagram that in this case, the model also depends on varying parameter values, since the epidemic threshold varies for different values of β.

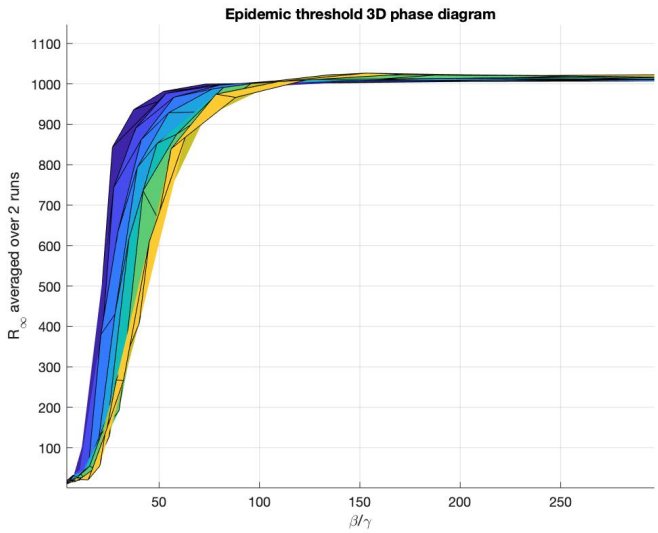


Figure 7: 3D phase diagram showing the epidemic thresholds.

4.1 Functions

4.1.1 Initialize Positions

```
function positions=InitializePosition(numberOfAgents,gridSize)

for j=1:numberOfAgents
    x = randperm(gridSize,1);
    y = randperm(gridSize,1);
    positions(j,:)= [x,y];
end
end
```

4.1.2 Update Positions

```
function updatePositions=UpdatePositions(numberOfAgents,positions,diffusionRate,gridSize)

updatePositions=positions;

for j=1:numberOfAgents
    r=rand;
    randomisedPositions=randperm(2,1);

    if r < diffusionRate
        updatePositions(j,randomisedPositions)=positions(j,randomisedPositions)+sign(randn);

        if updatePositions(j,randomisedPositions) > gridSize
            updatePositions(j,randomisedPositions) = gridSize;

        else if updatePositions(j,randomisedPositions) < 0
                updatePositions(j,randomisedPositions) = 0;
            end
        end

    else
        updatePositions(j,randomisedPositions) = positions(j,randomisedPositions);
    end
end
end
```

4.1.3 Check infection

```
function [susceptibleAgents, infectedAgents]=CheckInfection(infectedAgents,susceptibleAgents↩
    ,infectionRate)


numberOfSusceptibleAgents=size(susceptibleAgents,1);
numberOfInfectedAgent=size(infectedAgents,1);
tempVector=[];


for j=1:numberOfSusceptibleAgents

    for k=1:numberOfInfectedAgent

        if susceptibleAgents(j,:) == infectedAgents(k,:)
            r=rand;
            if r < infectionRate
                infectedAgents(size(infectedAgents,1)+1,:)=susceptibleAgents(j,:);
                tempVector=[tempVector j];

            end

        end

    end

end
susceptibleAgents(tempVector,:)=[];
end
```

4.1.4 Recovery infection

```
function [recoveredAgents, infectedAgents]=RecoveryInfection(infectedAgents,recoveredAgents,↩
    recoveryRate)

counter=size(recoveredAgents,1)+1;
numberOfInfectedAgents=size(infectedAgents,1);
tempVector=[];


for j=1:numberOfInfectedAgents
    r=rand;

    if r < recoveryRate
        recoveredAgents(counter,:) = infectedAgents(j,:);
        tempVector=[tempVector j];
        counter=counter+1;

    end
end
infectedAgents(tempVector,:)=[];
end
```

4.2 Exercise files

4.2.1 Exercise 1.1

```
clear all
clc
clf

numberOfAgents=1;
gridSize=100;
diffusionRate=0.8;
infectionRate=0.6;
numberOfTimeSteps= 1000;

positions=InitializePosition(numberOfAgents,gridSize);
x=positions(:,1);
y=positions(:,2);

scatter(x,y,25,'bo','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])

for i=1:numberOfTimeSteps

updatedPositions=UpdatePositions(numberOfAgents,positions,diffusionRate,gridSize);
positions=updatedPositions;
x=positions(:,1);
y=positions(:,2);

scatter(x,y,25,'bo','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])
drawnow
title('One agent random walk with: d=0.8, \beta=0.6, \gamma=0.01, t=1000')
xlabel('x')
ylabel('y')
hold on
end
```

4.2.2 Excerice 1.2, 1,3 and excercise 2

```
clear all
clc
clf

recoveryRate=0.01; %change depending on task
diffusionRate=0.8;
infectionRate=0.6; %change depending on task
numberOfAgents=1000; %change depending on task
gridSize=100;
numberOfInfected=100; %change depending on task
numberOfSusceptibles=numberOfAgents-numberOfInfected;
recoveredAgents=zeros(0,2);
numberOfSteps=1000;


infectedAgents=InitializePosition(numberOfInfected,gridSize);
x1=infectedAgents(:,1);
y1=infectedAgents(:,2);

scatter(x1,y1,25,'r','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])
hold on

susceptibleAgents=InitializePosition(numberOfSusceptibles,gridSize);
x2=susceptibleAgents(:,1);
y2=susceptibleAgents(:,2);

scatter(x2,y2,25,'b','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])

for iSteps=1:numberOfSteps
hold off

[susceptibleAgents, infectedAgents]=CheckInfection(infectedAgents,susceptibleAgents,↩
    infectionRate);
[recoveredAgents, infectedAgents]=RecoveryInfection(infectedAgents,recoveredAgents,↩
    recoveryRate);

%susceptibles
numberOfSusceptibles=size(susceptibleAgents,1);
updatedSusceptibleAgents=UpdatePositions(numberOfSusceptibles,susceptibleAgents,↩
    diffusionRate,gridSize);
susceptibleAgents=updatedSusceptibleAgents;

x3=susceptibleAgents(:,1);
y3=susceptibleAgents(:,2);
scatter(x3,y3,25,'b','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])
hold on

%infected
numberOfInfected=size(infectedAgents,1);
updatedInfectedAgents=UpdatePositions(numberOfInfected,infectedAgents,diffusionRate,gridSize↩
    );
infectedAgents=updatedInfectedAgents;

x4=infectedAgents(:,1);
y4=infectedAgents(:,2);
scatter(x4,y4,25,'r','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])
hold on

%recovered
numberOfRecovered=size(recoveredAgents,1);
```

```
updatedRecoveredAgents=UpdatePositions(numberOfRecovered,recoveredAgents,diffusionRate,↩
    gridSize);
recoveredAgents=updatedRecoveredAgents;

x5=recoveredAgents(:,1);
y5=recoveredAgents(:,2);
scatter(x5,y5,25,'g','filled')
grid on
grid minor
axis([0 gridSize 0 gridSize])

outputAgents(iSteps,:)=[numberOfRecovered, numberOfInfected, numberOfSusceptibles];

title('t=1000')
xlabel('x')
ylabel('y')


drawnow
end

figure(2);
x =linspace(0,numberOfSteps,numberOfSteps);
y6 = outputAgents(:,1);
y7 = outputAgents(:,2);
y8 = outputAgents(:,3);
plot(x,y6,'g',x,y7,'r',x,y8,'b')
title('d=0.8, beta=0.6, gamma=0.01')
xlabel('Time Steps')
ylabel('Number of Agents')
```

4.2.3 Exercise 3

```
clear
clc
clf

diffusionRate=0.8;
gridSize=100;
numberOfRuns=2;

vectorOfGamma=0.0007*1.398.^(1:20);
vectorOfBeta=[0.1 0.9];


for i=1:length(vectorOfBeta)
    outputAgents3=0;
    infectionRate=vectorOfBeta(i);


    for iRuns=1:numberOfRuns

        for j=1:length(vectorOfGamma)
            recoveryRate=vectorOfGamma(j);

            numberOfAgents=1000;
            numberOfInfected=10;
            numberOfSusceptibles=numberOfAgents-numberOfInfected;
            infectedAgents=InitializePosition(numberOfInfected,gridSize);
            susceptibleAgents=InitializePosition(numberOfSusceptibles,gridSize);
            recoveredAgents=zeros(0,2);
            k=0;


            while numberOfInfected > 0
                k=k+1;


                [susceptibleAgents, infectedAgents]=CheckInfection(infectedAgents,↩
                    susceptibleAgents,infectionRate);
                [recoveredAgents, infectedAgents]=RecoveryInfection(infectedAgents,↩
                    recoveredAgents,recoveryRate);


                %susceptibles
                numberOfSusceptibles=size(susceptibleAgents,1);
                updatedSusceptibleAgents=UpdatePositions(numberOfSusceptibles,↩
                    susceptibleAgents,diffusionRate,gridSize);
                susceptibleAgents=updatedSusceptibleAgents;

                %infected
                numberOfInfected=size(infectedAgents,1);
                updatedInfectedAgents=UpdatePositions(numberOfInfected,infectedAgents,↩
                    diffusionRate,gridSize);
                infectedAgents=updatedInfectedAgents;

                %recovered
                numberOfRecovered=size(recoveredAgents,1);
                updatedRecoveredAgents=UpdatePositions(numberOfRecovered,recoveredAgents,↩
                    diffusionRate,gridSize);
                recoveredAgents=updatedRecoveredAgents;

                outputAgents1(k,:)=[numberOfRecovered, numberOfInfected, ↩
                    numberOfSusceptibles];

                if k == 10000
                    outputAgents1(k,:)=[1000, 0, 0];
                    break
                end

            end

            outputAgents2(j)=outputAgents1(k,1);
            ratio(j)=infectionRate/recoveryRate;
```

```
        end
        outputAgents3=outputAgents2+outputAgents3;

    end
    averageRecovery(i,:)=outputAgents3/numberOfRuns;

    hold on
    plot(ratio, averageRecovery(i,:))
    drawnow
end

title('Epidemic threshold for \beta=0.1 and \beta=0.9')
legend('\beta=0.1','\beta=0.9')
xlabel('\beta/\gamma')
ylabel('R_\infty averaged over 4 runs')
```

4.2.4 Exercise 4

```
clear
clc
clf

diffusionRate=0.8;
gridSize=100;
numberOfRuns=2;

vectorOfGamma=0.0007*1.398.^(1:20);
vectorOfBeta=linspace(0.1,0.9,9);


for i=1:length(vectorOfBeta)
    outputAgents3=0;
    infectionRate=vectorOfBeta(i);


    for iRuns=1:numberOfRuns

        for j=1:length(vectorOfGamma)
            recoveryRate=vectorOfGamma(j);

            numberOfAgents=1000;
            numberOfInfected=10;
            numberOfSusceptibles=numberOfAgents-numberOfInfected;
            infectedAgents=InitializePosition(numberOfInfected,gridSize);
            susceptibleAgents=InitializePosition(numberOfSusceptibles,gridSize);
            recoveredAgents=zeros(0,2);
            k=0;


            while numberOfInfected > 0
                k=k+1;


                [susceptibleAgents, infectedAgents]=CheckInfection(infectedAgents,↩
                    susceptibleAgents,infectionRate);
                [recoveredAgents, infectedAgents]=RecoveryInfection(infectedAgents,↩
                    recoveredAgents,recoveryRate);


                %susceptibles
                numberOfSusceptibles=size(susceptibleAgents,1);
                updatedSusceptibleAgents=UpdatePositions(numberOfSusceptibles,↩
                    susceptibleAgents,diffusionRate,gridSize);
                susceptibleAgents=updatedSusceptibleAgents;

                %infected
                numberOfInfected=size(infectedAgents,1);
                updatedInfectedAgents=UpdatePositions(numberOfInfected,infectedAgents,↩
                    diffusionRate,gridSize);
                infectedAgents=updatedInfectedAgents;

                %recovered
                numberOfRecovered=size(recoveredAgents,1);
                updatedRecoveredAgents=UpdatePositions(numberOfRecovered,recoveredAgents,↩
                    diffusionRate,gridSize);
                recoveredAgents=updatedRecoveredAgents;

                outputAgents1(k,:)=[numberOfRecovered, numberOfInfected, ↩
                    numberOfSusceptibles];

                if k == 10000
                    outputAgents1(k,:)=[1000, 0, 0];
                    break
                end

            end

            outputAgents2(j)=outputAgents1(k,1);
            ratio(j,i)=infectionRate/recoveryRate;
```

```
        end
        outputAgents3=outputAgents2+outputAgents3;

    end
    averageRecovery(:,i)=outputAgents3/numberOfRuns;

end
finalInfectionRate=vectorOfBeta.*ones(length(vectorOfGamma),length(vectorOfBeta));

surf(ratio, averageRecovery, finalInfectionRate)


title('Epidemic threshold 3D phase diagram')
xlabel('\beta/\gamma')
ylabel('R_\infty averaged over 2 runs')
zlabel('\beta')
```