Answers to Project Sheet 1, Modelling Complex Systems

Fredrik Gustafsson

April 2019

1 A Pretty Picture Painter

In the simulations below for the cellular automata painter periodic boundary conditions have been implemented for both simulations.

1.1 Black and White Painter

Running the "painter" for 11,000 steps gives the resulting figure shown below, figure 1. Note that the rotation and position will be random due to the painters initial position and direction being assigned randomly.

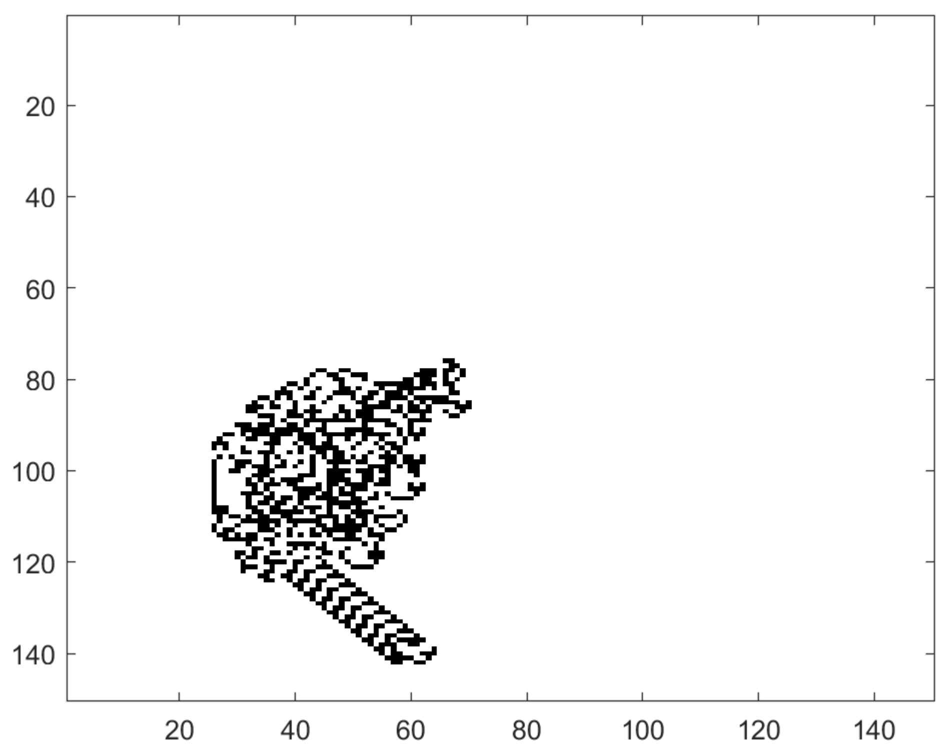


Figure 1: Resulting image from the black and white painter after 11,000 steps

1.2 A Colour Painter

In this part the painter was given more choices of colour and more steps to finish its painting. On result from this is the image, figure 2, shown below. It features a large region with no obvious pattern, probably where the painter started. However what makes it interesting is that it also has structured areas. As an example there's clearly two triangular region each with a small line running out from the two corners furthest from the unstructured part, it also features regions with nice parallel lines. Since both of these more ordered features are less bright then parts of the more "chaotic" center this probably means that it has been visited fewer times, it could also have been visited enough times to make the colour overflow and roll around.

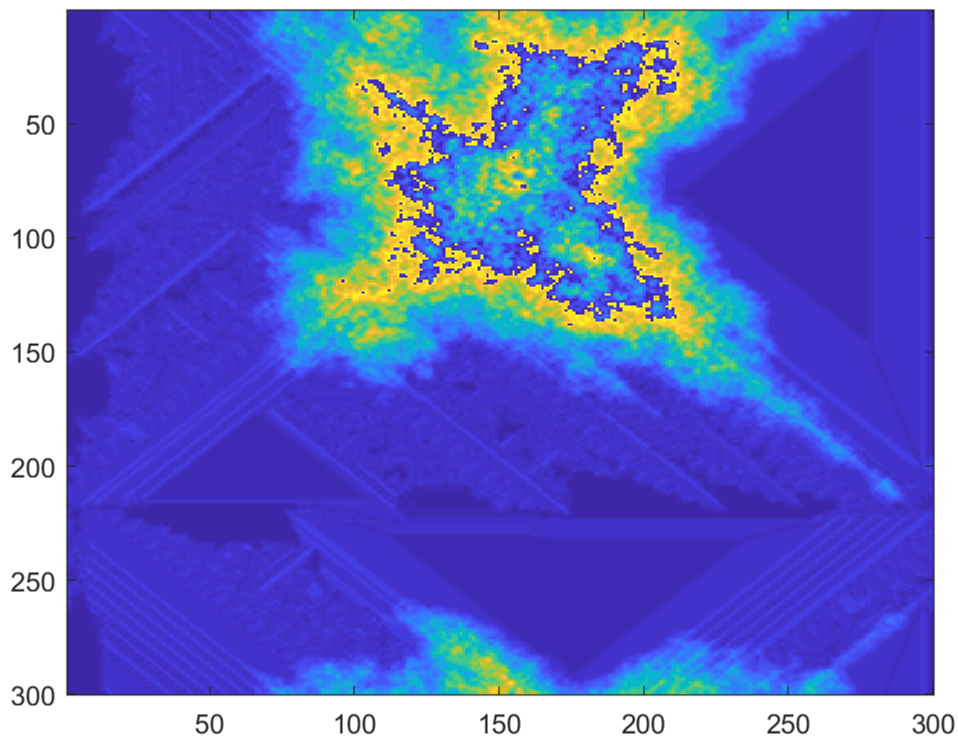


Figure 2: An interesting image

2 Spread of Memes

In the following subsections some assumptions have been made. The first is that the total population stays constant. Secondly we assume that everyone can be a sharer, resting or bored. The third assumption is that the actions are not concurrent. At first every sharer does one action then every resting person and finally every bored one. Finally in the last part we assume that a person only interacts with the four closest persons, on a grid these would be the persons with the coordinates (x+1,y),(x-1,y),(x,y+1) and (x,y-1) with the person having the coordinates (x,y), in the same simulation we also assume periodic boundary conditions.

2.1 People Stay Bored

For the case in which a person who becomes bored stays bored the result from running the model for 4000 steps, q = 0.01, p = 0.001, $B_0 = S_0 = 1$ and doing 1000 simulations gives figure 3 below. In the image the black lines are from the simulation the red line are the mean value of the simulation and the blue line represents the solution from the mean-field equation.

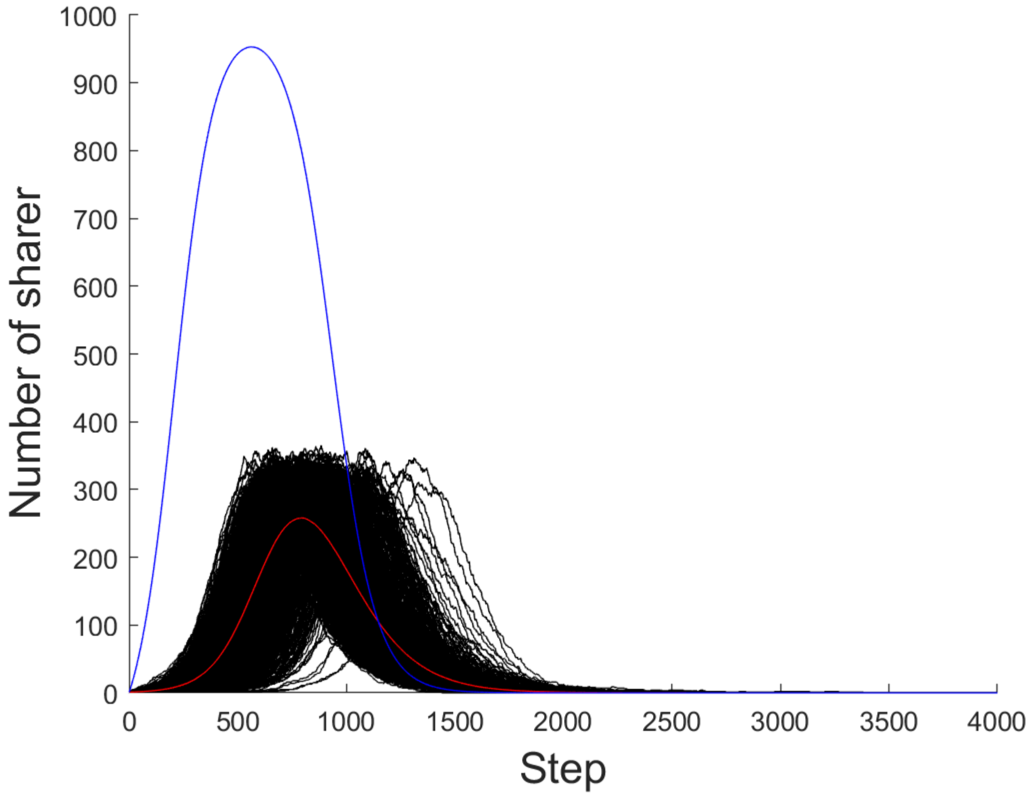
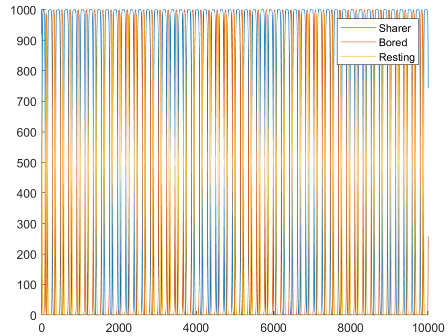


Figure 3: Number of sharers at each time-step

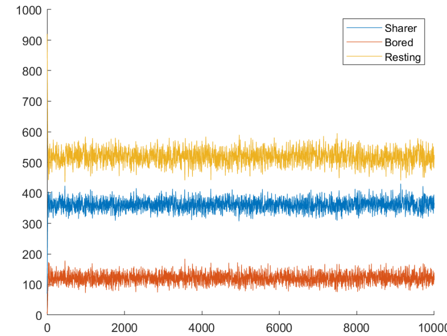
As we can see from figure 3 the solution to the mean field equation have a higher peak value with almost everyone becomeing a sharer. The maximum value for the mean-field equation also occurs sooner then the max value from the simulation, however the mean-field solution and the simulation agrees on the general shape.

2.2 Bored People Recovers

In the case when bored people can recover and become resting again some different dynamics of the system can be observed in the mean-field model. Everyone can end up being sharers, everyone ending up bored, sinusoidal behaviour with decreasing amplitude and which approaches constant values. Finally one can end up with neverending cycles in which the population changes between sharers, bored and resting. One set of parameters which achieves this in the mean field model is q = 0.2, p = 0.001, r = 0.6, $B_0 = 10$ and $S_0 = 70$ which is shown in figure 4 below, however as the same image shows the simulation does not exhibit the same dynamics.



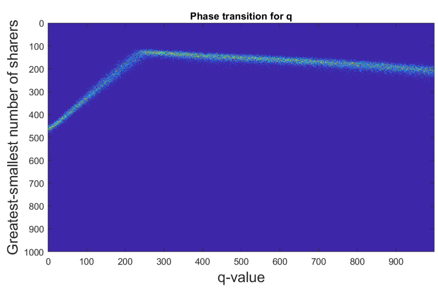
(a) Mean-Field model



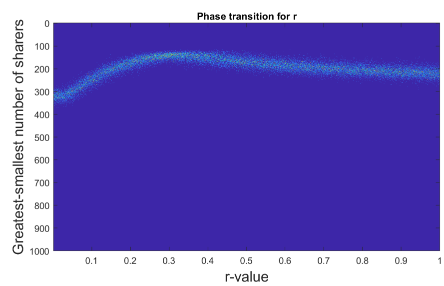
(b) Simulation

Figure 4: Simulation with q = 0.2, p = 0.001, r = 0.6, $B_0 = 10$ and $S_0 = 70$

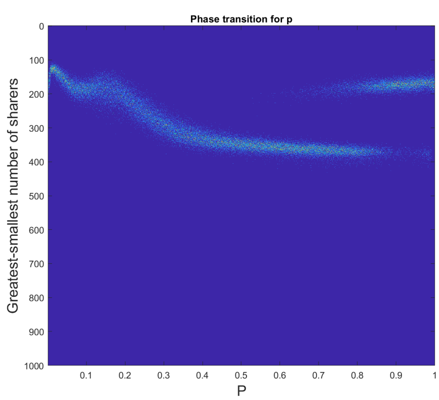
A phase transition diagram can be made by varying the parameter which we want to investigate, p, q and r. In this case the investigated parameter was increased by 0.001 from 0.001 to 1 with 40 repetitions for each value. The parameters which was not investigated was kept constant with the values $S_0 = 70$, $B_0 = 1$, total population of 1000, steps = 10000, q = 0.2, p = 0.001 and r = 0.6. Figure 5 shows the result when using the difference between maximum and minimum number of sharers. Since we're not interested in the case in which the number of sharers increase rapidly in one step and then goes to zero in the next, giving a difference of 1000, the first 10 and last 20 steps was discarded.



(a) Phase transition for q



(b) Phase transition for r



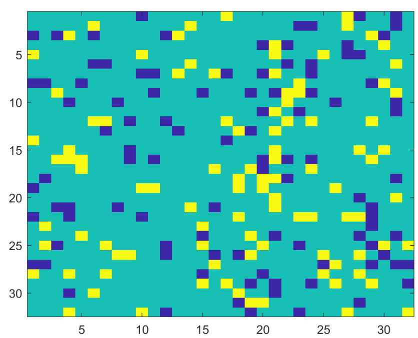
(c) Phase transition for p

Figure 5: Phase transition diagram

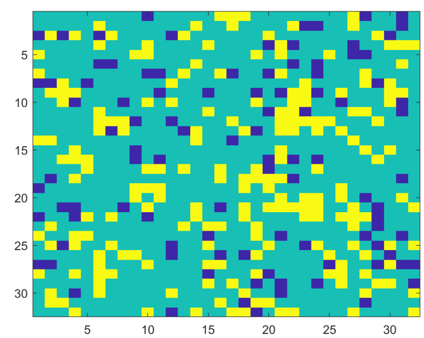
As we can see varying q and r gives similar looking phase transition diagram with a difference reaching around 200 and a clear minimum value around 0.25. Varying p however leads to some fluctuations at first which appears to converge towards 350 however after around 0.9 the value jumps down to around 150 instead.

2.3 People Interact in a Spatial World

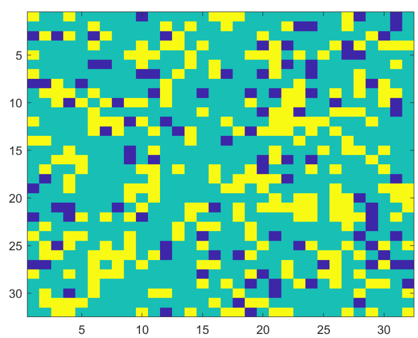
When the interactions can only be done between a person and the four nearest neighbours, up, down, left and right, on a grid the time serie shown in figure 6 and 7 can be created.



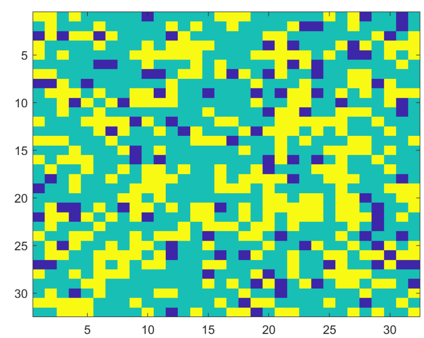
(a) Start



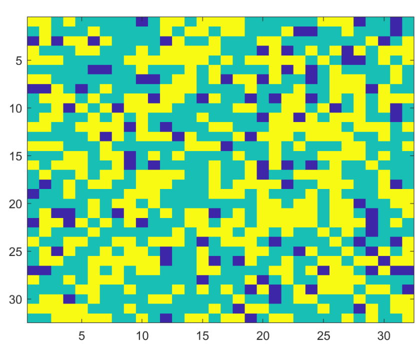
(b) After 100 steps



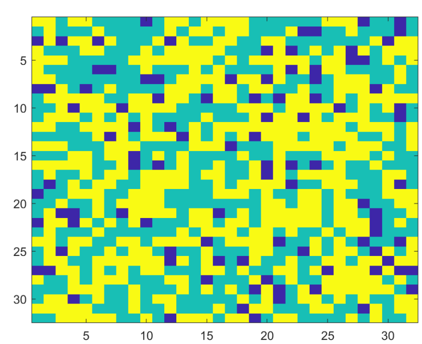
(c) After 200 steps



(d) After 300 steps

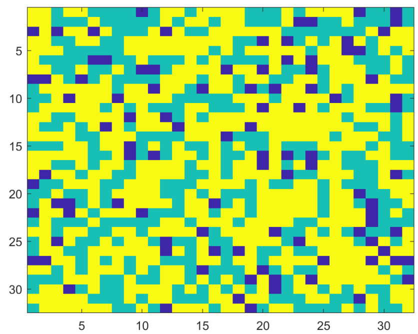


(e) After 400 steps

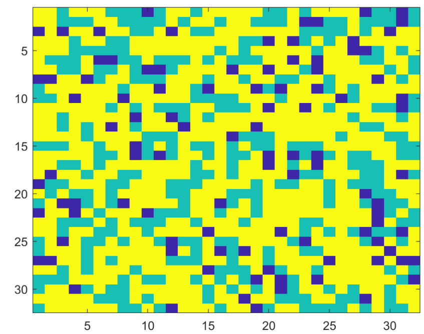


(f) After 500 steps

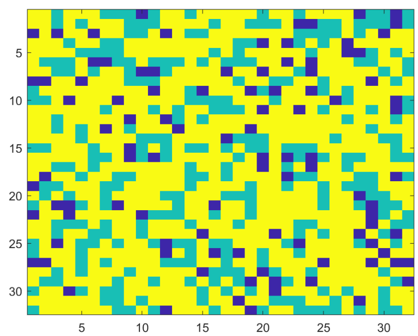
Figure 6: Time serie



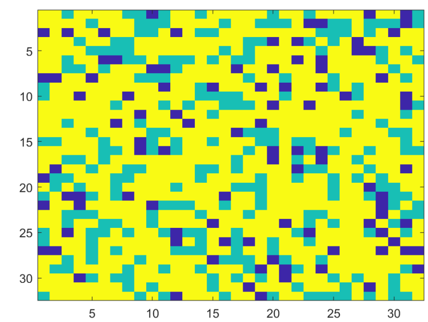
(a) After 600 steps



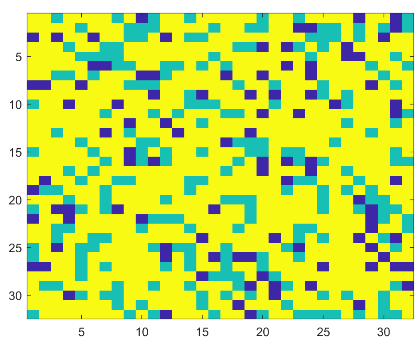
(b) After 700 steps



(c) After 800 steps



(d) After 900 steps



(e) After 999 steps

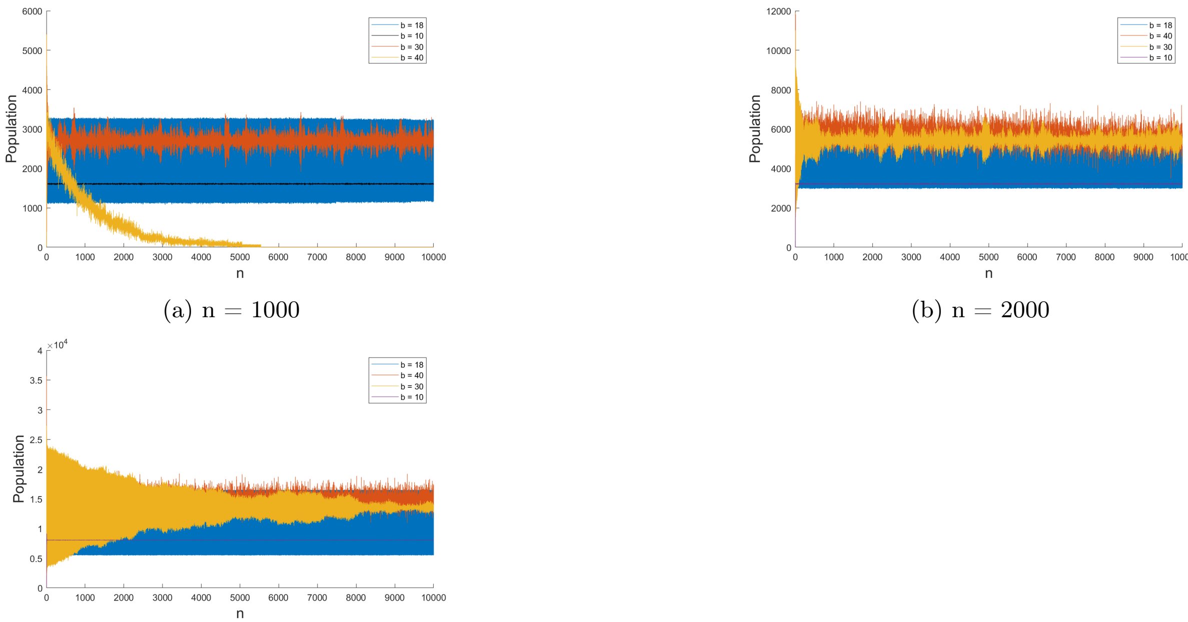
Figure 7: Time serie continuation

The serie was generated with the parameters q = 0.01, p = 0.001, r = 0.04, $B_0 = 100$, $S_0 = 100$ and total population 1024 for 1000 steps. In the figures yellow represents sharers, blue bored and turquoise resting. As we can see there is some clear regions of sharer, bored and resting persons. As we can see when the agents interacting have a spatial position and can only interact with their neighbors we get more sharer in the end than what might be expected. The initial spread of the bored persons in relation to the sharing ones matters quite a bit. If the bored and sharing agents where position in a checkerboard pattern, in a region, every sharer would interact with a bored one and hence would themselves turn bored. In the case presented in figures 6 and 7 we can see that the bored persons are fairly sparsely located at the start thereby decreasing the chance that a sharer will interact with them. Another thing to note however is that we only see snapshots every 100 step hence bored people can change to resting or sharer between images.

3 Population Dynamics

3.1 Population can Move Everywhere

In the case when every one can move everywhere the figure 8 shown below can be made. In it three plots are shown for n = 1000, 2000 and 5000 which is the number of resource spots. In each of these plots curves for b=[10 18 30 40] and p = 0.5 with an initial population of 10 are shown.



(a) n = 1000

(b) n = 2000

(c) n = 5000

Figure 8: Change of population

As one can see from figure **??** the population becomes larger for larger n. This is not too surprising since the chance that individuals will die of from overcrowding decreases. For b values of 10, 18 and 40 the population have a similar behaviour for the different values of n, larger n leads to larger population as mentioned. However for b= 30 the behaviour of the population varies a lot more. In the case when n = 1000 the population simply dies out after around 5500 generations. for n = 2000 the population stabilizes fairly quickly after an initial spike. When n = 5000 the population once again stabilizes and although it takes longer the end result is less noisy.

A phase transition can be made by varying b from 1 to 50 with steps of 1 doing 500000 repetition for each value an initial population of 10 and 50 steps for each repetition, saving the final number of individuals. The result is shown below in figure 9

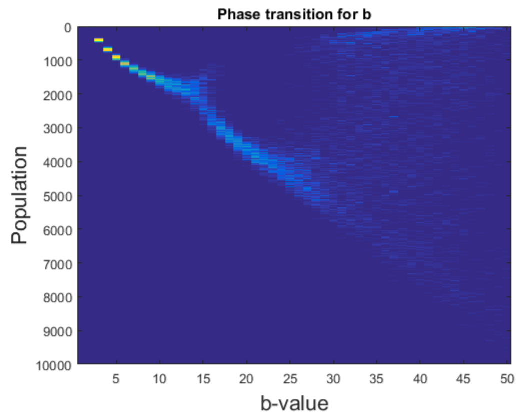


Figure 9: Phase transition

From figure 9 one can see that as b increases so does the population. However once b reaches 13 it becomes increasingly harder to predict what value the population will have in the end. The phase transition diagram stops following a clear line and instead appears more random, indicating that the system is showcasing properties of chaos.

3.2 Limited Movement for the Population

When simulating the resource sites as being located on a ring and introducing a new parameter d which controls how far a person can move one can plot the mean population for combinations of b and d. From such a plot one can find the combinations at which the population dies out. By also plotting the `mean(population-mean(population))` in which the population is the population at the end of each repetition one can see how stable the end result is. For these two plots the parameters was set to n = 1000, reps = 5, p= 0.5, steps = 100, initial population = 20 b = [1:1:50] and d = [1:1:n].

(a) Mean value of population at end

(b) `mean(population-mean(population))`

Figure 10: Population dynamics measurements

As we can see from figure 10a as b increases a small increase in d is necessary not not have the population dying out. The population also dies out when b is 45 or more and d is larger than 350. for d values larger than 100 and b values between 15 and 28 a valley of low population is formed. The popoulation doesn't die out but is smaller in the end when compared to larger and smaller b values, larger than 28 smaller than 15. As figure 10b shows a smaller end population corresponds to less variance between the different runs. However at the same time the largest variance is still on a scale of 10^{-13}.

4 Appendix

4.1 Cellular automata

4.1.1 randomPainter.m

```
clear all
close all
N = 150;
grid = zeros(N,N);
steps = 11000;
right = [2 3 4 1];
left = [4 1 2 3];
x = randi(N,1);
y = randi(N,1);
dir = randi(4,1);
for i = 0:steps
    if(grid(x,y) == 0)
        grid(x,y) = 1;
        dir = right(dir);
    else
        grid(x,y) = 0;
        dir = left(dir);
    end
    if(dir == 1)
        if( x == 1)
            x = N;
        else
            x = x-1;
        end
    elseif(dir == 2)
        if(y == N)
            y = 1;
        else
            y = y+1;
        end
    elseif(dir == 3)
        if(x == N)
            x = 1;
        else
            x = x+1;
        end
    else
        if(y == 1)
            y = N;
        else
            y = y-1;
        end
    end
end
colormap('gray')
imagesc(imcomplement(grid))
```

4.1.2 paintPainter.m

```
c = 300;
N = 300;
steps = 1100000000;
```

```
right = [2 3 4 1];
left = [4 1 2 3];

for l = 1:20
    grid = zeros(N,N);
    x = randi(N,1);
    y = randi(N,1);
    dir = randi(4,1);


    for i = 1:c
        rules(i) = round(rand());
    end
    %if all rules are 1 or 0 change one
    if sum(rules) == c
        rules(randi(c,1)) = 0;
    elseif sum(rules) == 0
        rules(randi(c,1)) = 1;
    end

    for i = 1:steps
        k = mod(grid(x,y),c);
        grid(x,y) = mod(k+1,c);
        if(rules(k+1)==1)
            dir = right(dir);
        else
            dir = left(dir);
        end
        if(dir == 1)
            if( x == 1)
                x = N;
            else
                x = x-1;
            end
        elseif(dir == 2)
            if(y == N)
                y = 1;
            else
                y = y+1;
            end
        elseif(dir == 3)
            if(x == N)
                x = 1;
            else
                x = x+1;
            end
        else
            if(y == 1)
                y = N;
            else
             y = y-1;
            end
        end
    end
    figure()
    imagesc(grid)
end
```

4.2 Spread of Memes

4.2.1 mainMeme.m

```
%% 1.2.1
clear all

N = 1000;
steps = 4000;
q = 0.01;
p = 0.001;
B0 = 1;
S0 = 1;
R0 = N-B0-S0;
runs = 1000;
plotS = zeros(runs,steps);
plotB = zeros(runs,steps);
plotR = zeros(runs,steps);
for i = 1:runs
    [S B R] = memeSpread1(R0,S0,B0,N,p,q,steps);
    plotS(i,:) = S;
    plotB(i,:) = B;
    plotR(i,:) = R;
end
figure
hold on
plot(1:1:steps,plotS,'k')
pr = plot(1:1:steps, mean(plotS),'r')
set(pr,'LineWidth',2)

%Mean field equations
Smf(1) = S0;
Bmf(1) = B0;
Rmf(1) = R0;

for i = 1:steps-1
    Smf(i+1) = Smf(i)+p*Rmf(i)-q*Smf(i)*Bmf(i)/N+q*Smf(i)*Rmf(i)/N;
    Bmf(i+1) = Bmf(i)+q*Smf(i)*Bmf(i)/N;
    Rmf(i+1) = Rmf(i)-p*Rmf(i)-q*Smf(i)*Rmf(i)/N;
end
figure
hold on
plot(1:1:steps,plotS,'k')
plot(1:1:steps, mean(plotS),'r')
plot(1:1:steps, Smf, 'b')
ylabel('Number of sharer','FontSize',16)
xlabel('Step','FontSize',16)
%legend('','Mean value of simulation', 'Mean-Field equation')
%xlabe
%set(pr,'LineWidth',2)

%% 1.2.2.1
clear all
close all
N = 1000;
steps = 10000;
%Cycles B0 = 10, S0 = 100, q= 0.02, p = 0, r = 0.01
q = 0.2; % sort of neverending cycles
p = 0.001; % sort of neverending cycles
r = 0.6; % sort of neverending cycles
B0 = 10;
```

```
S0 = 70;
R0 = N-B0-S0;
runs = 1;
plotS = zeros(runs,steps);
plotB = zeros(runs,steps);
plotR = zeros(runs,steps);
for i = 1:runs
    [S B R] = memeSpread2(R0,S0,B0,N,p,q,r,steps);
    plotS(i,:) = S;
    plotB(i,:) = B;
    plotR(i,:) = R;
end
figure
hold on
plot(1:1:steps,plotS)
plot(1:1:steps,plotB)
plot(1:1:steps,plotR)
legend('Sharer','Bored','Resting');



%Mean field model
Smf(1) = S0;
Bmf(1) = B0;
Rmf(1) = R0;
for i = 1:steps-1
    Smf(i+1) = Smf(i)+p*Rmf(i)-(q*Smf(i)*Bmf(i)/N)+(q*Smf(i)*Rmf(i)/N);
    Bmf(i+1) = Bmf(i)+q*Smf(i)*Bmf(i)/N-(r*Bmf(i)*Rmf(i)/N);
    Rmf(i+1) = Rmf(i)-(p*Rmf(i))-(q*Smf(i)*Rmf(i)/N)+(r*Bmf(i)*Rmf(i)/N);
end
figure
hold on
plot(1:1:steps,Smf)
plot(1:1:steps,Bmf)
plot(1:1:steps,Rmf)
hold off
legend('Sharer','Bored','Resting');
%% 1.2.2.2
clear all
N = 1000;
steps = 10000;
q = 0.2; %0.01 sort of neverending cycles
p = 0.001; %0.001 sort of neverending cycles
r = 0.6; %0.09 sort of neverending cycles
B0 = 1;
S0 = 70;
R0 = N-B0-S0;
Smf(1) = S0;
Bmf(1) = B0;
Rmf(1) = R0;
p =[0.001:0.001:1];
reps = 40;
hrange=[0:1:N];
histSp=zeros(length(r),length(hrange));
for k = 1:length(p)
    for j = 1:reps

        [S B R] = memeSpread2(R0,S0,B0,N,p(k),q,r,steps);
        [Sm, L] = bounds(S(10:end-20));
        finalSp(k,j)= L - Sm;
    end
     histSp(k,:)=hist(finalSp(k,:),hrange);
```

```
end
figure
imagesc(p,hrange,histSp'/reps,[0 0.1])
title('Phase transition for r')
q = 0.2; %0.01 sort of neverending cycles
p = 0.001; %0.001 sort of neverending cycles
r = 0.6; %0.09 sort of neverending cycles
B0 = 1;
S0 = 70;
R0 = N-B0-S0;
Smf(1) = S0;
Bmf(1) = B0;
Rmf(1) = R0;
r =[0.001:0.001:1];
reps = 40;
hrange=[0:1:N];
histSr=zeros(length(r),length(hrange));
for k = 1:length(r)
    for j = 1:reps

        [S B R] = memeSpread2(R0,S0,B0,N,p,q,r(k),steps);
        [Sm, L] = bounds(S(10:end-20));
        finalSr(k,j)= L - Sm;
    end
     histSr(k,:)=hist(finalSr(k,:),hrange);
end
figure
imagesc(r,hrange,histSr'/reps,[0 0.1])
title('Phase transition for r')
q = 0.2; %0.01 sort of neverending cycles
p = 0.001; %0.001 sort of neverending cycles
r = 0.6; %0.09 sort of neverending cycles
B0 = 1;
S0 = 70;
R0 = N-B0-S0;
Smf(1) = S0;
Bmf(1) = B0;
Rmf(1) = R0;
q =[0.001:0.001:1];
reps = 40;
hrange=[0:1:N];
histSq=zeros(length(r),length(hrange));
for k = 1:length(q)
    for j = 1:reps

        [S B R] = memeSpread2(R0,S0,B0,N,p,q(k),r,steps);
        [Sm, L] = bounds(S(10:end-20));
        finalSq(k,j)= L - Sm;
    end
     histSq(k,:)=hist(finalSq(k,:),hrange);
end
figure
imagesc(q,hrange,histSq'/reps,[0 0.1])
title('Phase transition for q')
r = 0.01;
%% 1.2.3
clear all
close all
N = 32^2;
steps = 1000;
q = 0.01;
p = 0.001;
```

```
r = 0.04;
B0 = 100;
S0 = 100;
R0 = N-B0-S0;
runs = 1;
plotS = zeros(runs,steps);
plotB = zeros(runs,steps);
plotR = zeros(runs,steps);
for i = 1:runs
    [S B R] = memeSpreadGrid(R0,S0,B0,N,p,q,r,steps);
    plotS(i,:) = S;
    plotB(i,:) = B;
    plotR(i,:) = R;
end
```

4.2.2 memeSpread1.m

```
function [S B R] = memeSpread1(R0,S0,B0,N,p,q,steps)
R = zeros(steps,1);
S = zeros(steps,1);
B = zeros(steps,1);
R(1) = R0;
S(1) = S0;
B(1) = B0;

for i = 1:steps-1
    R(i+1) = R(i);
    S(i+1) = S(i);
    B(i+1) = B(i);
    for j = 1:S(i)
        if q > rand()
            if rand() < R(i+1)/N
                R(i+1) = R(i+1)-1;
                S(i+1) = S(i+1)+1;
            else
                S(i+1) = S(i+1)-1;
                B(i+1) = B(i+1)+1;
            end
        end
    end
    if rand()< p
        if S(i+1) ~= 0
            S(i+1) = S(i+1)+1;
            R(i+1) = R(i+1)-1;
        end
    end
end
```

4.2.3 memespread2.m

```
function [S B R] = memeSpread2(R0,S0,B0,N,p,q,r,steps)
R = zeros(steps,1);
S = zeros(steps,1);
B = zeros(steps,1);
R(1) = R0;
S(1) = S0;
B(1) = B0;
```

```

for i = 1:steps-1
    R(i+1) = R(i);
    S(i+1) = S(i);
    B(i+1) = B(i);
    for j = 1:S(i)
        if q > rand()
            if rand() < R(i+1)/N
                R(i+1) = R(i+1)-1;
                S(i+1) = S(i+1)+1;
            else
                S(i+1) = S(i+1)-1;
                B(i+1) = B(i+1)+1;
            end
        end
    end
    for j = i:R(i)
        if rand()< p
            S(i+1) = S(i+1)+1;
            R(i+1) = R(i+1)-1;
        end
    end
    for j = 1:B(i)
        if rand()< r
            if rand() < R(i+1)/N
                R(i+1) = R(i+1)+1;
                B(i+1) = B(i+1)-1;
            end
        end
    end

end
```

4.2.4 memeSpreadGrid.m

```
function [S B R] = memeSpreadGrid(R0,S0,B0,N,p,q,r,steps)

%%Modeling spread of memes on grid with periodic boundary
% 0-Resting, 1-Spreading, -1-Boring
dim = sqrt(N);
population = zeros(dim,dim);


for i = 1:S0
    x = randi(dim,1);
    y = randi(dim,1);
    if population(x,y) ~= 0
        while population(x,y) ~= 0
            x = randi(dim,1);
            y = randi(dim,1);
        end
    end
    population(x,y) = 1;
    s_pos(:,i) =[x;y];
end
for i = 1:B0
    x = randi(dim,1);
    y = randi(dim,1);
    if population(x,y) ~= 0
```

```
        while population(x,y) ~= 0
            x = randi(dim,1);
            y = randi(dim,1);
        end
    end
    population(x,y) = -1;
    b_pos(:,i) = [x;y];
end
dir = [-1 1];
R = zeros(1,steps);
B = zeros(1,steps);
S = zeros(1,steps);
R(1) = R0;
S(1) = S0;
B(1) = B0;

for i = 1:steps-1
    k = 1;
    R(i+1) = R(i);
    S(i+1) = S(i);
    B(i+1) = B(i);
    for j = 1:S(i)
        if q > rand() %Will sharer interact
            direction = round((dir(round(rand)+1)));
            if rand()<0.5 %x direction
                if s_pos(1,k) == 1 & direction == -1
                    if population(dim, s_pos(2,k)) == 0
                        population(dim, s_pos(2,k)) = 1;
                        S(i+1) = S(i+1)+1;
                        R(i+1) = R(i+1)-1;
                        s_pos = [s_pos, [dim;s_pos(2,k)]];
                        k = k+1;
                    elseif population(dim, s_pos(2,k)) == -1
                        population(1, s_pos(2,k)) = -1;
                        S(i+1) = S(i+1)-1;
                        B(i+1) = B(i+1)+1;
                        b_pos = [b_pos,[1;s_pos(2,k)]];

                        s_pos(:,k) = [];
                    else
                        k = k+1;
                    end
                elseif s_pos(1,k) == dim & direction == 1
                    if population(1, s_pos(2,k)) == 0
                        population(1, s_pos(2,k)) = 1;
                        S(i+1) = S(i+1)+1;
                        R(i+1) = R(i+1)-1;
                        s_pos = [s_pos, [1;s_pos(2,k)]];
                        k = k+1;
                    elseif population(1, s_pos(2,k)) == -1
                        population(dim, s_pos(2,k)) = -1;
                        S(i+1) = S(i+1)-1;
                        B(i+1) = B(i+1)+1;
                        b_pos = [b_pos,[dim;s_pos(2,k)]];
                        s_pos(:,k) = [];

                    else
                        k = k+1;
                    end
                else
                    if population(s_pos(1,k)+direction, s_pos(2,k)) == 0
                        population(s_pos(1,k)+direction, s_pos(2,k)) = 1;
```

```
                    S(i+1) = S(i+1)+1;
                    R(i+1) = R(i+1)-1;
                    s_pos = [s_pos, [s_pos(1,k)+direction;s_pos(2,k)]];
                    k = k+1;
                elseif population(s_pos(1,k)+direction, s_pos(2,k)) == -1
                    population(s_pos(1,k), s_pos(2,k)) = -1;
                    S(i+1) = S(i+1)-1;
                    B(i+1) = B(i+1)+1;
                    b_pos = [b_pos, [s_pos(1,k); s_pos(2,k)]];

                    s_pos(:,k) = [];
                else
                    k = k+1;
                end
            end
        else %y-position
            if s_pos(2,k) == 1 & direction == -1
                if population(s_pos(1,k),dim) == 0
                    population(s_pos(1,k),dim) = 1;
                    S(i+1) = S(i+1)+1;
                    R(i+1) = R(i+1)-1;
                    s_pos = [s_pos, [s_pos(1,k);dim]];
                    k = k+1;
                elseif population(s_pos(1,k),dim) == -1
                    population(s_pos(1,k),1) = -1;
                    S(i+1) = S(i+1)-1;
                    B(i+1) = B(i+1)+1;
                    b_pos = [b_pos,[s_pos(1,k);1]];

                    s_pos(:,k) = [];
                else
                    k = k+1;
                end
            elseif s_pos(2,k) == dim & direction == 1
                if population(s_pos(1,k),1) == 0
                    population(s_pos(1,k),1) = 1;
                    S(i+1) = S(i+1)+1;
                    R(i+1) = R(i+1)-1;
                    s_pos = [s_pos, [s_pos(1,k);1]];
                    k = k+1;
                elseif population(s_pos(1,k),1) == -1
                    population(s_pos(1,k),dim) = -1;
                    S(i+1) = S(i+1)-1;
                    B(i+1) = B(i+1)+1;
                    b_pos = [b_pos,[s_pos(1,k);dim]];

                    s_pos(:,k) = [];
                else
                    k = k+1;
                end
            else
                if population(s_pos(1,k), s_pos(2,k)+direction) == 0
                    population(s_pos(1,k), s_pos(2,k)+direction) = 1;
                    S(i+1) = S(i+1)+1;
                    R(i+1) = R(i+1)-1;
                    s_pos = [s_pos, [s_pos(1,k);s_pos(2,k)+direction]];
                    k = k + 1;
                elseif population(s_pos(1,k), s_pos(2,k)+direction) == -1
                    population(s_pos(1,k), s_pos(2,k)) = -1;
                    S(i+1) = S(i+1)-1;
                    B(i+1) = B(i+1)+1;
                    b_pos = [b_pos,[s_pos(1,k);s_pos(2,k)]];
```

```

                        s_pos(:,k) = [];
                    else
                        k = k+1;
                    end
                end
            end
        end
    end
    for j = 1:R(i)
        if rand()< p
            S(i+1) = S(i+1)+1;
            R(i+1) = R(i+1)-1;
            while population(x,y) ~= 0
                x = randi(dim,1);
                y = randi(dim,1);
            end
            population(x,y) = 1;
            s_pos = [s_pos,[x;y]];
        end
    end
    k = 1;
    for j = 1:B(i)
        if rand()< r %Boring interacts
            direction = round((dir(round(rand)+1)));
            if rand()<0.5 %x direction
                if b_pos(1,k) == 1 & direction == -1
                    if population(dim, b_pos(2,k)) == 0
                        population(1, b_pos(2,k)) = 0;
                        R(i+1) = R(i+1)+1;
                        B(i+1) = B(i+1)-1;
                        b_pos(:,k) = [];
                    else
                        k = k+1;
                    end
                elseif b_pos(1,k) == dim & direction == 1
                    if population(1, b_pos(2,k)) == 0
                        population(dim, b_pos(2,k)) = 0;
                        R(i+1) = R(i+1)+1;
                        B(i+1) = B(i+1)-1;
                        b_pos(:,k) = [];
                    else
                        k = k+1;
                    end
                else
                    if population(b_pos(1,k)+direction, b_pos(2,k)) == 0
                        population(b_pos(1,k)+direction, b_pos(2,k)) = 0;
                        R(i+1) = R(i+1)+1;
                        B(i+1) = B(i+1)-1;
                        b_pos(:,k) = [];
                    else
                        k = k+1;
                    end
                end
            else %y-position
                if b_pos(2,k) == 1 & direction == -1
                    if population(b_pos(1,k),dim) == 0
                        population(b_pos(1,k),dim) = 0;
                        R(i+1) = R(i+1)+1;
                        B(i+1) = B(i+1)-1;
                        b_pos(:,k) = [];
                    else
```

```
                    k = k+1;
                end
            elseif b_pos(2,k) == dim & direction == 1
                if population(b_pos(1,k),1) == 0
                    population(b_pos(1,k),1) = 0;
                    R(i+1) = R(i+1)+1;
                    B(i+1) = B(i+1)-1;
                    b_pos(:,k) = [];
                else
                    k = k+1;
                end
            else
                if population(b_pos(1,k), b_pos(2,k)+direction) == 0
                    population(b_pos(1,k), b_pos(2,k)+direction) = 0;
                    R(i+1) = R(i+1)+1;
                    B(i+1) = B(i+1)-1;
                    b_pos(:,k) = [];
                else
                    k = k+1;
                end
            end
        end
    end
end
if i == 1
    figure()
    imagesc(population)
    title('Start')
elseif i == steps/10
        figure()
    imagesc(population)
    title('steps/10')
elseif i == 2*steps/10
        figure()
    imagesc(population)
    title('2*steps/10')
elseif i == 3*steps/10
        figure()
    imagesc(population)
    title('3*steps/10')
elseif i == 4*steps/10
        figure()
    imagesc(population)
    title('4*steps/10')
elseif i == 5*steps/10
        figure()
    imagesc(population)
    title('5*steps/10')
elseif i == 6*steps/10
        figure()
    imagesc(population)
    title('6*steps/10')
elseif i == 7*steps/10
        figure()
    imagesc(population)
    title('7*steps/10')
elseif i == 8*steps/10
        figure()
    imagesc(population)
    title('8*steps/10')
elseif i == 9*steps/10
        figure()
```

```
        imagesc(population)
        title('9*steps/10')
    elseif i == steps-1
        figure
        imagesc(population)
        title('Population at end')
    end

end
end
```

4.2.5 Population Dynamics

4.2.6 mainPopDyn.m

```
%% 1.3.1.1
clear all
close all
p = 0.5;
b = [10 18];
n = [10 100 1000 5000];
steps = 100;
init_pop = 10;
n = 1000;
for i = 1:length(b)
    [state population] = popDyn11(n,b(i),p,init_pop,steps);
    pop1(i,:) = population
end

figure
plot(1:1:steps, population)
%% 1.3.1.2
clear all
clc
b = [1:1:50];
n = 1000;
p = 0.5
steps = 50;
init_pop = 10;
reps = 5000;
max_pop = 10000
hrange=[0:1:max_pop];
histSq=zeros(length(b),length(hrange));
for i = 1:length(b)
    for j = 1:reps
        [state population] = popDyn11(n,b(i),p,init_pop,steps);
        finalSq(i,j)=population(end);
    end
     histSq(i,:)=hist(finalSq(i,:),hrange);
end
figure
imagesc(b,hrange,histSq'/reps,[0 0.01])
title('Phase transition for b')




%% 1.3.3
```

```
clear all
close all
n = 1000;
b = [1:1:50];
d = [1:1:n];
reps = 5;
p = 0.5;
steps = 100;
init_pop = 20;
for i = 1:length(b)
    for j = 1:length(d);
        for k = 1:reps
            [state population] = popDyn12(n,b(i),p,d(j),init_pop,steps);
            pop(k) = population(end);
        end
        plotPop(i,j) = mean(pop);
        stab(i,j) = mean(pop-mean(pop));
    end
end
```

4.2.7 memespread2.m

```
function [state ret_pop] = popDyn11(n,b,p,init_pop,steps)
state = zeros(steps,n);
pop = init_pop;
ret_pop = zeros(1,steps);
mean_val = b*p;
standard_dev = sqrt(b*p*(1-p));
ent = 0;
    for i = 1:steps
        temp_state = zeros(1,n);
        if i == 1
            for j = 1:pop
                place = round(rand*(n-1))+1;
                temp_state(place) = temp_state(place) + 1;
            end
        else
            for j = 1:pop
                place = round(rand*(n-1))+1;
                temp_state(place) = temp_state(place) + 1;
            end
            for j = 1:n
                if temp_state(j) == 1
                    temp_state(j) = round(standard_dev*randn()+mean_val);
                else
                    temp_state(j) = 0;
                end
            end
        end
        pop = sum(temp_state);
        ret_pop(1,i) = pop;
        state(i,:) = temp_state;
    end


end
```

4.2.8 memeSpreadGrid.m

```
function [state, ret_pop] = popDyn12(n,b,p,d,init_pop,steps)
state = zeros(steps,n);
test_state = state;
pop = init_pop;
mean_val = b*p;
standard_dev = sqrt(b*p*(1-p));
dir = [-1 1];
for i = 1:steps
    temp_state = zeros(1,n);
    if i == 1
        for j = 1:pop
            place = round(rand*(n-1))+1;
            temp_state(place) = temp_state(place) + 1;
        end
    else
        for j = 1:n
            if state(i-1,j) ~= 0
                number = state(i-1,j);
                for k = 0:number
                    place = j+round((dir(round(rand)+1))*rand*d);
                    if place < 1
                        place = n + place;
                    elseif place > n
                        place = place - n;
                    end
                    temp_state(place) = temp_state(place)+1;
                end
            end
        end
        test_state(i,:) = temp_state;
        for j = 1:n
            if temp_state(j) == 1
                temp_state(j) = round(standard_dev*randn()+mean_val);
            else
                temp_state(j) = 0;
            end
        end
    end
    pop = sum(temp_state);
    ret_pop(1,i) = pop;
    state(i,:) = temp_state;
end
end
```