

# 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.

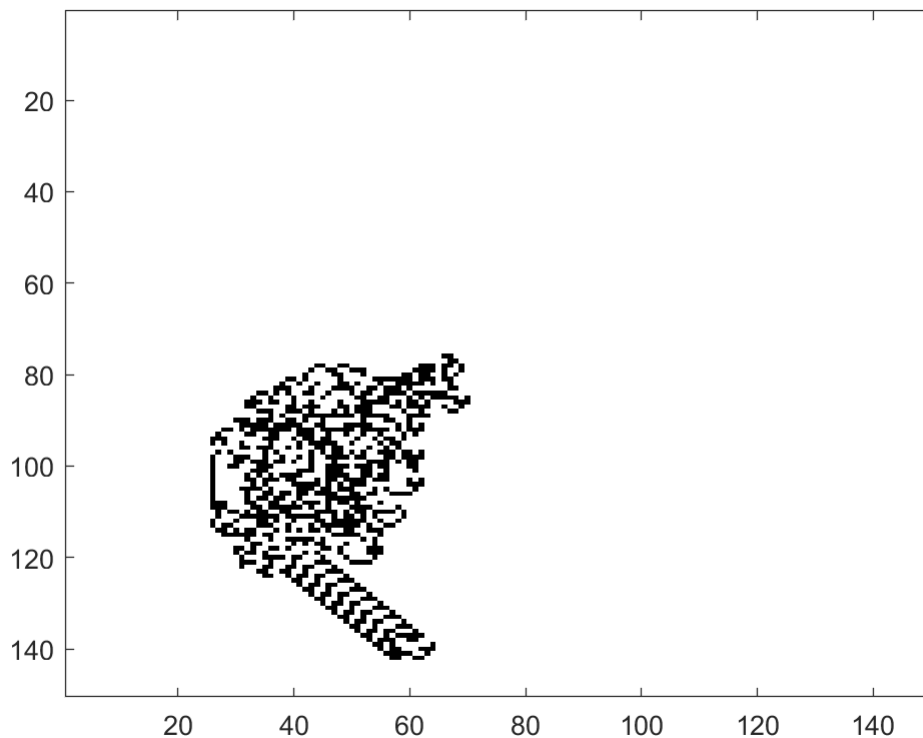


Figure 1: Resulting image from the black and white painter after 11,000 steps

## 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.

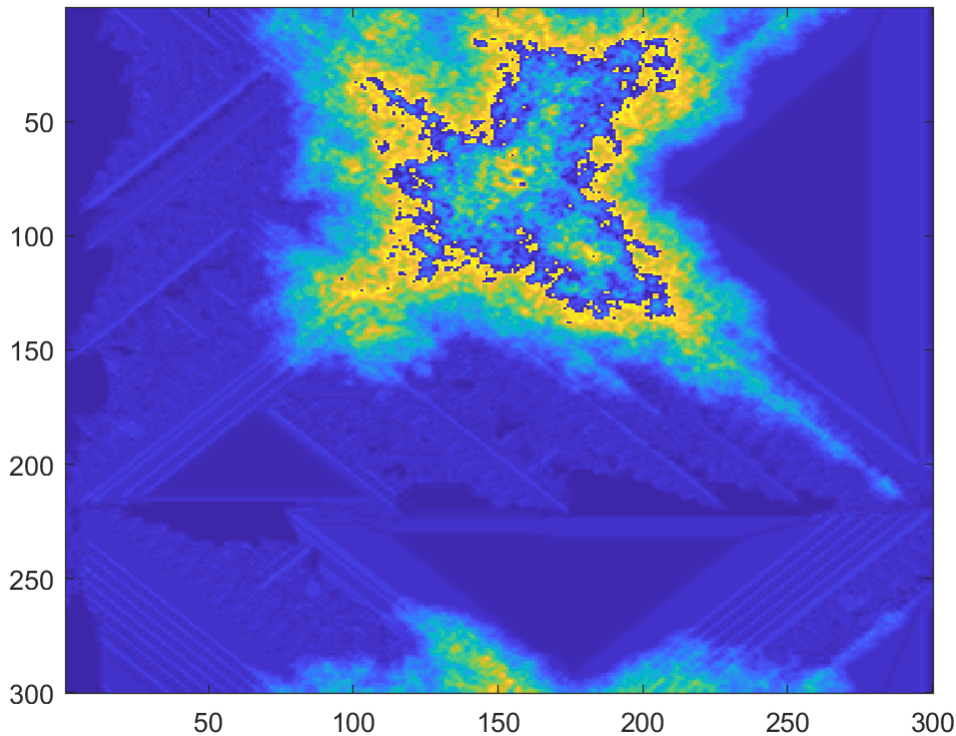


Figure 2: An interesting image

## 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.

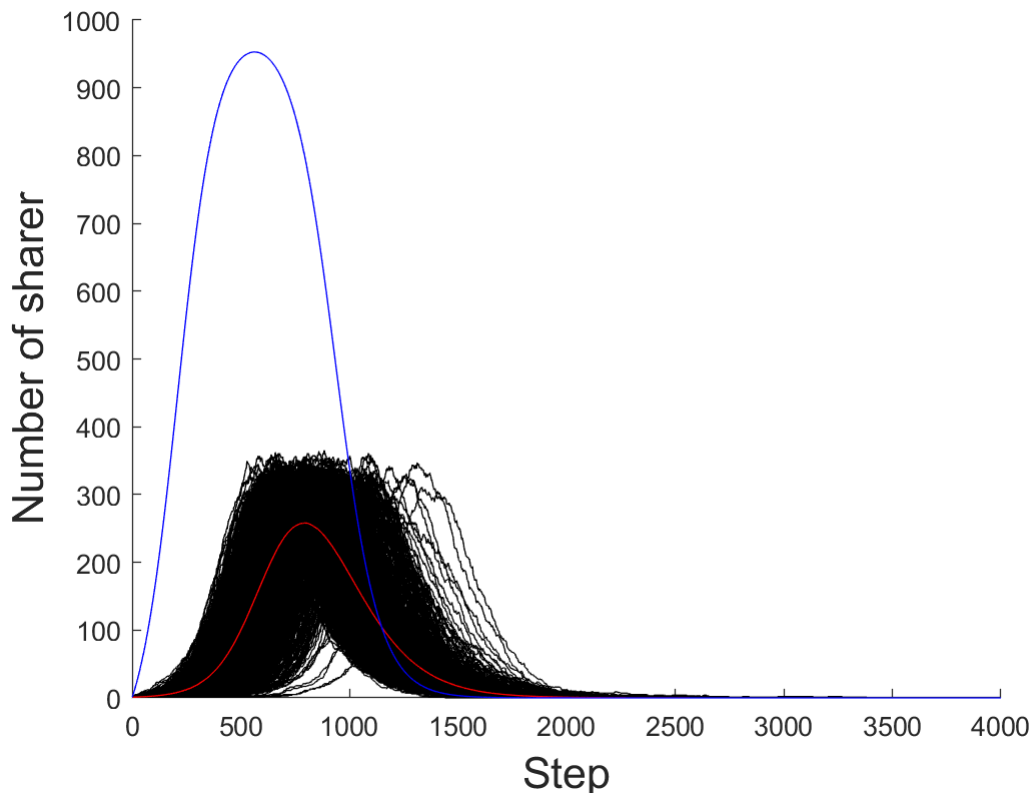
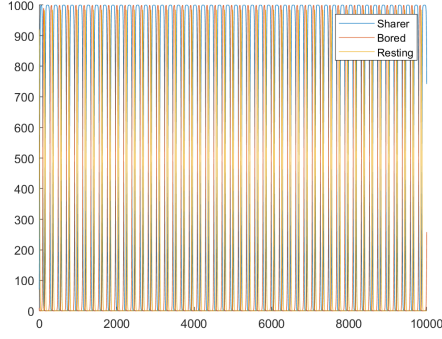


Figure 3: Number of sharers at each time-step

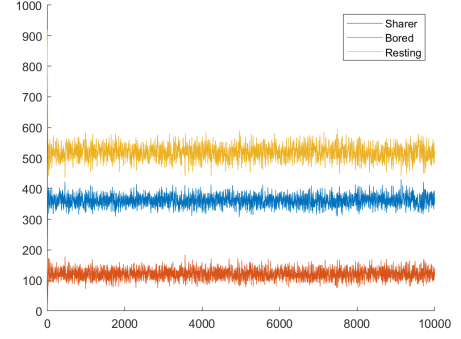
As we can see from figure 3 the solution to the mean field equation have a higher peak value with almost everyone becomeing a sharer. The maximum value for the mean-field equation also occurs sooner then the max value from the simulation, however the mean-field solution and the simulation agrees on the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



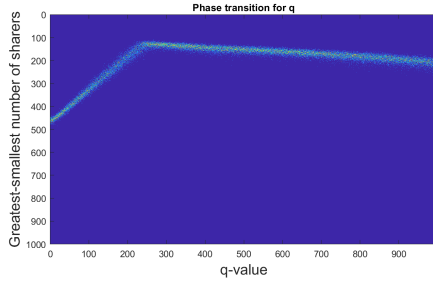
(a) Mean-Field model



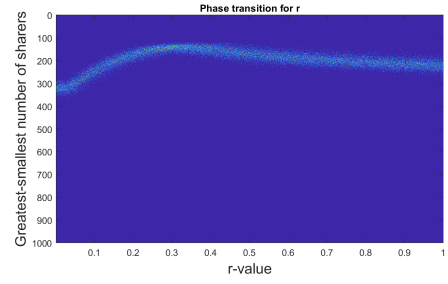
(b) Simulation

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

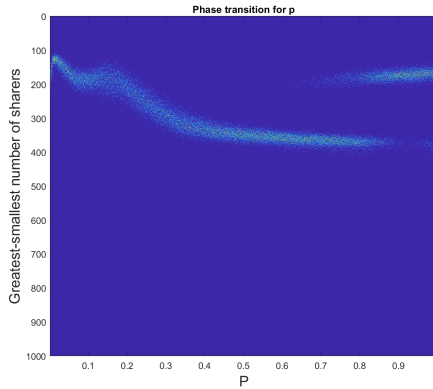
A phase transition diagram can be made by varying the parameter which we want to investigate,  $p$ ,  $q$  and  $r$ . In this case the investigated parameter was increased by 0.001 from 0.001 to 1 with 40 repetitions for each value. The parameters which was not investigated was kept constant with the 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  $r$



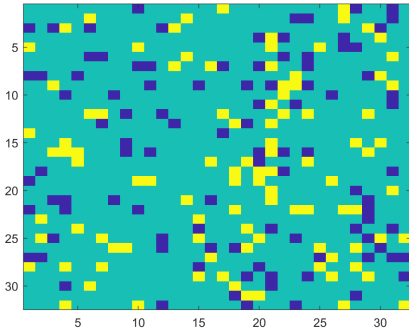
(c) Phase transition for  $p$

Figure 5: Phase transition diagram

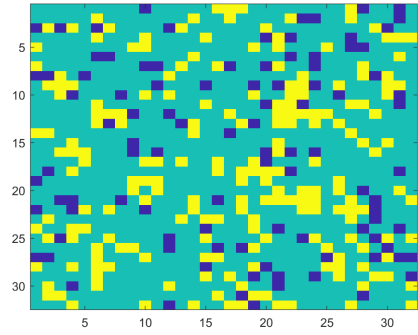
As we can see varying  $q$  and  $r$  gives similar looking phase transition diagram with a difference reaching around 200 and a clear minimum value around 0.25. Varying  $p$  however leads to some fluctuations at first which appears to converge towards 350 however after around 0.9 the value jumps down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

### 2.3 People Interact in a Spatial World

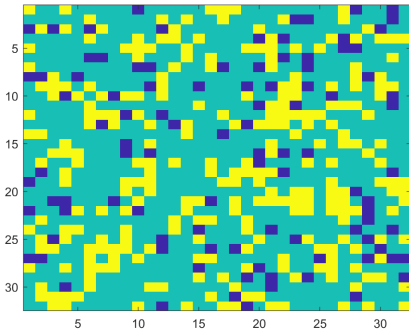
When the interactions can only be done between a person and the four nearest neighbours, up, down, left and right, on a grid the time series shown in figure 6 and 7 can be created.



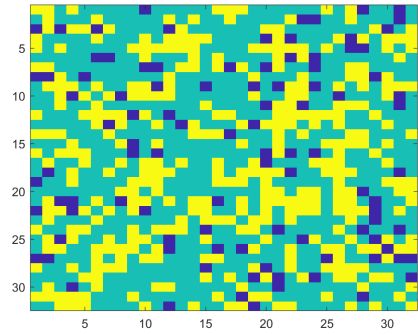
(a) Start



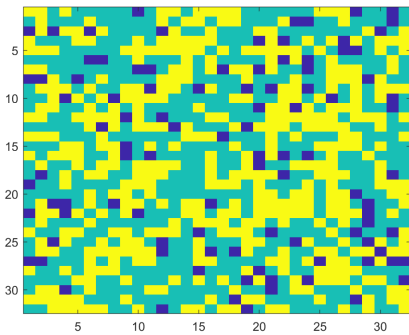
(b) After 100 steps



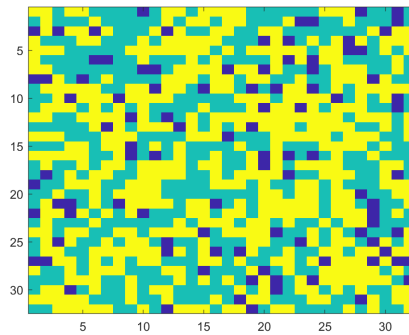
(c) After 200 steps



(d) After 300 steps

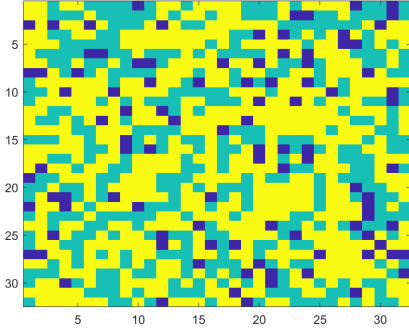


(e) After 400 steps

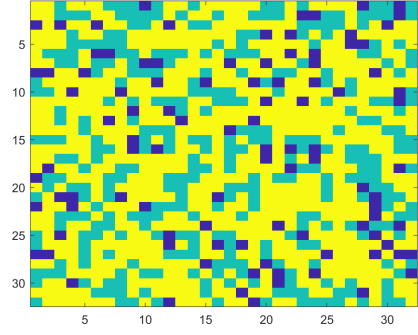


(f) After 500 steps

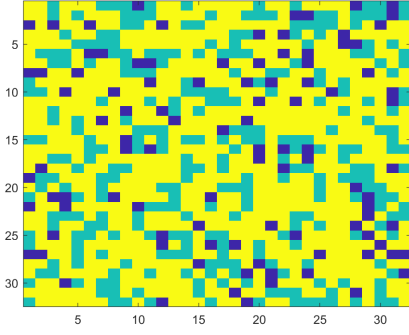
Figure 6: Time serie



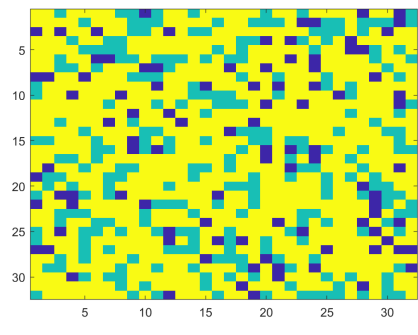
(a) After 600 steps



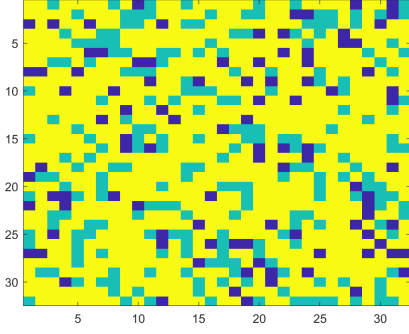
(b) After 700 steps



(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.

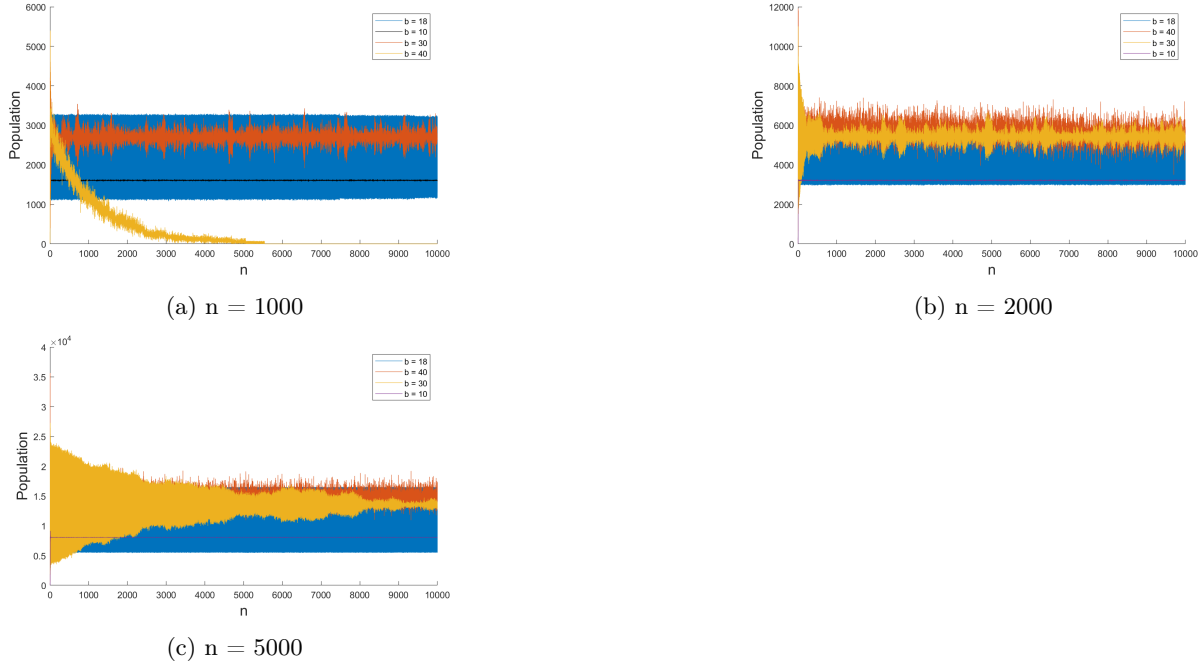


Figure 8: Change of population

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

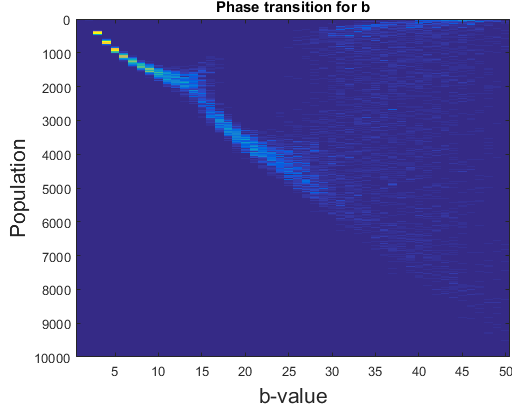
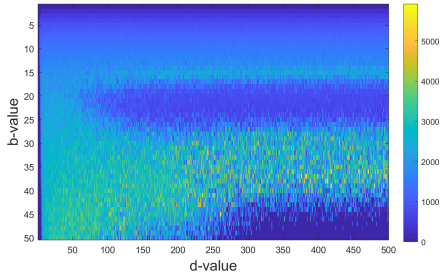


Figure 9: Phase transition

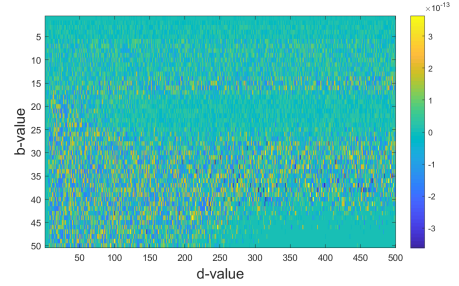
From figure 9 one can see that as  $b$  increases so does the population. However once  $b$  reaches 13 it becomes increasingly harder to predict what value the population will have in the end. The phase transition diagram stops following a clear line and instead appears more random, indicating that 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  $\text{mean}(\text{population} - \text{mean}(\text{population}))$  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  $n = 1000$ ,  $\text{reps} = 5$ ,  $p = 0.5$ ,  $\text{steps} = 100$ ,  $\text{initial population} = 20$   $b = [1:1:50]$  and  $d = [1:1:n]$ .



(a) Mean value of population at end



(b)  $\text{mean}(\text{population} - \text{mean}(\text{population}))$

Figure 10: Population dynamics measurements

As we can see from figure 10a as  $b$  increases a small increase in  $d$  is necessary not to 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 population 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

```
1 clear all
2 close all
3 N = 150;
4 grid = zeros(N,N);
5 steps = 11000;
6 right = [2 3 4 1];
7 left = [4 1 2 3];
8 x = randi(N,1);
9 y = randi(N,1);
10 dir = randi(4,1);
11 for i = 0:steps
12     if(grid(x,y) == 0)
13         grid(x,y) = 1;
14         dir = right(dir);
15     else
16         grid(x,y) = 0;
17         dir = left(dir);
18     end
19     if(dir == 1)
20         if( x == 1)
21             x = N;
22         else
23             x = x-1;
24         end
25     elseif(dir == 2)
26         if(y == N)
27             y = 1;
28         else
29             y = y+1;
30         end
31     elseif(dir == 3)
32         if(x == N)
33             x = 1;
34         else
35             x = x+1;
36         end
37     else
38         if(y == 1)
39             y = N;
40         else
41             y = y-1;
42         end
43     end
44 end
45 colormap('gray')
46 imagesc(imcomplement(grid))
```

#### 4.1.2 paintPainter.m

```
1 c = 300;
2 N = 300;
3 steps = 1100000000;
```

```

4 right = [2 3 4 1];
5 left = [4 1 2 3];
6
7 for l = 1:20
8     grid = zeros(N,N);
9     x = randi(N,1);
10    y = randi(N,1);
11    dir = randi(4,1);
12
13
14    for i = 1:c
15        rules(i) = round(rand());
16    end
17    %if all rules are 1 or 0 change one
18    if sum(rules) == c
19        rules(randi(c,1)) = 0;
20    elseif sum(rules) == 0
21        rules(randi(c,1)) = 1;
22    end
23
24    for i = 1:steps
25        k = mod(grid(x,y),c);
26        grid(x,y) = mod(k+1,c);
27        if(rules(k+1)==1)
28            dir = right(dir);
29        else
30            dir = left(dir);
31        end
32        if(dir == 1)
33            if( x == 1)
34                x = N;
35            else
36                x = x-1;
37            end
38        elseif(dir == 2)
39            if(y == N)
40                y = 1;
41            else
42                y = y+1;
43            end
44        elseif(dir == 3)
45            if(x == N)
46                x = 1;
47            else
48                x = x+1;
49            end
50        else
51            if(y == 1)
52                y = N;
53            else
54                y = y-1;
55            end
56        end
57    end
58    figure()
59    imagesc(grid)
60 end

```

## 4.2 Spread of Memes

### 4.2.1 mainMeme.m

```
1 %% 1.2.1
2 clear all
3
4 N = 1000;
5 steps = 4000;
6 q = 0.01;
7 p = 0.001;
8 B0 = 1;
9 S0 = 1;
10 R0 = N-B0-S0;
11 runs = 1000;
12 plotS = zeros(runs,steps);
13 plotB = zeros(runs,steps);
14 plotR = zeros(runs,steps);
15 for i = 1:runs
16     [S B R] = memeSpread1(R0,S0,B0,N,p,q,steps);
17     plotS(i,:) = S;
18     plotB(i,:) = B;
19     plotR(i,:) = R;
20 end
21 figure
22 hold on
23 plot(1:1:steps,plotS,'k')
24 pr = plot(1:1:steps, mean(plotS), 'r')
25 set(pr,'LineWidth',2)
26
27 %Mean field equations
28 Smf(1) = S0;
29 Bmf(1) = B0;
30 Rmf(1) = R0;
31
32 for i = 1:steps-1
33     Smf(i+1) = Smf(i)+p*Rmf(i)-q*Smf(i)*Bmf(i)/N+q*Smf(i)*Rmf(i)/N;
34     Bmf(i+1) = Bmf(i)+q*Smf(i)*Bmf(i)/N;
35     Rmf(i+1) = Rmf(i)-p*Rmf(i)-q*Smf(i)*Rmf(i)/N;
36 end
37 figure
38 hold on
39 plot(1:1:steps,plotS,'k')
40 plot(1:1:steps, mean(plotS), 'r')
41 plot(1:1:steps, Smf, 'b')
42 ylabel('Number of sharer','FontSize',16)
43 xlabel('Step','FontSize',16)
44 %legend('', 'Mean value of simulation', 'Mean-Field equation')
45 %xlabe
46 %set(pr,'LineWidth',2)
47
48 %% 1.2.2.1
49 clear all
50 close all
51 N = 1000;
52 steps = 10000;
53 %Cycles B0 = 10, S0 = 100, q= 0.02, p = 0, r = 0.01
54 q = 0.2; % sort of neverending cycles
55 p = 0.001; % sort of neverending cycles
56 r = 0.6; % sort of neverending cycles
57 B0 = 10;
```

```

58 S0 = 70;
59 R0 = N-B0-S0;
60 runs = 1;
61 plotS = zeros(runs,steps);
62 plotB = zeros(runs,steps);
63 plotR = zeros(runs,steps);
64 for i = 1:runs
65     [S B R] = memeSpread2(R0,S0,B0,N,p,q,r,steps);
66     plotS(i,:) = S;
67     plotB(i,:) = B;
68     plotR(i,:) = R;
69 end
70 figure
71 hold on
72 plot(1:1:steps,plotS)
73 plot(1:1:steps,plotB)
74 plot(1:1:steps,plotR)
75 legend('Sharer','Bored','Resting');
76
77
78
79 %Mean field model
80 Smf(1) = S0;
81 Bmf(1) = B0;
82 Rmf(1) = R0;
83 for i = 1:steps-1
84     Smf(i+1) = Smf(i)+p*Rmf(i)-(q*Smf(i)*Bmf(i)/N)+(q*Smf(i)*Rmf(i)/N);
85     Bmf(i+1) = Bmf(i)+q*Smf(i)*Bmf(i)/N-(r*Bmf(i)*Rmf(i)/N);
86     Rmf(i+1) = Rmf(i)-(p*Rmf(i))-(q*Smf(i)*Rmf(i)/N)+(r*Bmf(i)*Rmf(i)/N);
87 end
88 figure
89 hold on
90 plot(1:1:steps,Smf)
91 plot(1:1:steps,Bmf)
92 plot(1:1:steps,Rmf)
93 hold off
94 legend('Sharer','Bored','Resting');
95 %% 1.2.2.2
96 clear all
97 N = 1000;
98 steps = 10000;
99 q = 0.2; %0.01 sort of neverending cycles
100 p = 0.001; %0.001 sort of neverending cycles
101 r = 0.6; %0.09 sort of neverending cycles
102 B0 = 1;
103 S0 = 70;
104 R0 = N-B0-S0;
105 Smf(1) = S0;
106 Bmf(1) = B0;
107 Rmf(1) = R0;
108 p =[0.001:0.001:1];
109 reps = 40;
110 hrange=[0:1:N];
111 histSp=zeros(length(r),length(hrange));
112 for k = 1:length(p)
113     for j = 1:reps
114
115         [S B R] = memeSpread2(R0,S0,B0,N,p(k),q,r,steps);
116         [Sm, L] = bounds(S(10:end-20));
117         finalSp(k,j)= L - Sm;
118     end
119     histSp(k,:)=hist(finalSp(k,:),hrange);

```

```

120 end
121 figure
122 imagesc(p,hrange,histSp'/reps,[0 0.1])
123 title('Phase transition for r')
124 q = 0.2; %0.01 sort of neverending cycles
125 p = 0.001; %0.001 sort of neverending cycles
126 r = 0.6; %0.09 sort of neverending cycles
127 B0 = 1;
128 S0 = 70;
129 R0 = N-B0-S0;
130 Smf(1) = S0;
131 Bmf(1) = B0;
132 Rmf(1) = R0;
133 r = [0.001:0.001:1];
134 reps = 40;
135 hrange=[0:1:N];
136 histSr=zeros(length(r),length(hrange));
137 for k = 1:length(r)
138     for j = 1:reps
139
140         [S B R] = memeSpread2(R0,S0,B0,N,p,q,r(k),steps);
141         [Sm, L] = bounds(S(10:end-20));
142         finalSr(k,j)= L - Sm;
143     end
144     histSr(k,:)=hist(finalSr(k,:),hrange);
145 end
146 figure
147 imagesc(r,hrange,histSr'/reps,[0 0.1])
148 title('Phase transition for r')
149 q = 0.2; %0.01 sort of neverending cycles
150 p = 0.001; %0.001 sort of neverending cycles
151 r = 0.6; %0.09 sort of neverending cycles
152 B0 = 1;
153 S0 = 70;
154 R0 = N-B0-S0;
155 Smf(1) = S0;
156 Bmf(1) = B0;
157 Rmf(1) = R0;
158 q = [0.001:0.001:1];
159 reps = 40;
160 hrange=[0:1:N];
161 histSq=zeros(length(r),length(hrange));
162 for k = 1:length(q)
163     for j = 1:reps
164
165         [S B R] = memeSpread2(R0,S0,B0,N,p,q(k),r,steps);
166         [Sm, L] = bounds(S(10:end-20));
167         finalSq(k,j)= L - Sm;
168     end
169     histSq(k,:)=hist(finalSq(k,:),hrange);
170 end
171 figure
172 imagesc(q,hrange,histSq'/reps,[0 0.1])
173 title('Phase transition for q')
174 r = 0.01;
175 %% 1.2.3
176 clear all
177 close all
178 N = 32^2;
179 steps = 1000;
180 q = 0.01;
181 p = 0.001;

```

```

182 r = 0.04;
183 B0 = 100;
184 S0 = 100;
185 R0 = N-B0-S0;
186 runs = 1;
187 plotS = zeros(runs,steps);
188 plotB = zeros(runs,steps);
189 plotR = zeros(runs,steps);
190 for i = 1:runs
191     [S B R] = memeSpreadGrid(R0,S0,B0,N,p,q,r,steps);
192     plotS(i,:) = S;
193     plotB(i,:) = B;
194     plotR(i,:) = R;
195 end

```

#### 4.2.2 memeSpread1.m

```

1 function [S B R] = memeSpread1(R0,S0,B0,N,p,q,steps)
2 R = zeros(steps,1);
3 S = zeros(steps,1);
4 B = zeros(steps,1);
5 R(1) = R0;
6 S(1) = S0;
7 B(1) = B0;
8
9 for i = 1:steps-1
10     R(i+1) = R(i);
11     S(i+1) = S(i);
12     B(i+1) = B(i);
13     for j = 1:S(i)
14         if q > rand()
15             if rand() < R(i+1)/N
16                 R(i+1) = R(i+1)-1;
17                 S(i+1) = S(i+1)+1;
18             else
19                 S(i+1) = S(i+1)-1;
20                 B(i+1) = B(i+1)+1;
21             end
22         end
23     end
24     if rand() < p
25         if S(i+1) ~= 0
26             S(i+1) = S(i+1)+1;
27             R(i+1) = R(i+1)-1;
28         end
29     end
30 end

```

#### 4.2.3 memespread2.m

```

1 function [S B R] = memeSpread2(R0,S0,B0,N,p,q,r,steps)
2 R = zeros(steps,1);
3 S = zeros(steps,1);
4 B = zeros(steps,1);
5 R(1) = R0;
6 S(1) = S0;
7 B(1) = B0;

```

```

8
9  for i = 1:steps-1
10     R(i+1) = R(i);
11     S(i+1) = S(i);
12     B(i+1) = B(i);
13     for j = 1:S(i)
14         if q > rand()
15             if rand() < R(i+1)/N
16                 R(i+1) = R(i+1)-1;
17                 S(i+1) = S(i+1)+1;
18             else
19                 S(i+1) = S(i+1)-1;
20                 B(i+1) = B(i+1)+1;
21             end
22         end
23     end
24     for j = i:R(i)
25         if rand() < p
26             S(i+1) = S(i+1)+1;
27             R(i+1) = R(i+1)-1;
28         end
29     end
30     for j = 1:B(i)
31         if rand() < r
32             if rand() < R(i+1)/N
33                 R(i+1) = R(i+1)+1;
34                 B(i+1) = B(i+1)-1;
35             end
36         end
37     end
38
39 end

```

#### 4.2.4 memeSpreadGrid.m

```

1  function [S B R] = memeSpreadGrid(R0,S0,B0,N,p,q,r,steps)
2
3  %%Modeling spread of memes on grid with periodic boundary
4  % 0-Resting, 1-Spreading, -1-Boring
5  dim = sqrt(N);
6  population = zeros(dim,dim);
7
8
9  for i = 1:S0
10     x = randi(dim,1);
11     y = randi(dim,1);
12     if population(x,y) ~= 0
13         while population(x,y) ~= 0
14             x = randi(dim,1);
15             y = randi(dim,1);
16         end
17     end
18     population(x,y) = 1;
19     s_pos(:,i) = [x;y];
20 end
21 for i = 1:B0
22     x = randi(dim,1);
23     y = randi(dim,1);
24     if population(x,y) ~= 0

```

```

25         while population(x,y) ~= 0
26             x = randi(dim,1);
27             y = randi(dim,1);
28         end
29     end
30     population(x,y) = -1;
31     b_pos(:,i) = [x;y];
32 end
33 dir = [-1 1];
34 R = zeros(1,steps);
35 B = zeros(1,steps);
36 S = zeros(1,steps);
37 R(1) = R0;
38 S(1) = S0;
39 B(1) = B0;
40
41 for i = 1:steps-1
42     k = 1;
43     R(i+1) = R(i);
44     S(i+1) = S(i);
45     B(i+1) = B(i);
46     for j = 1:S(i)
47         if q > rand() %Will sharer interact
48             direction = round((dir(round(rand)+1)));
49             if rand() < 0.5 %x direction
50                 if s_pos(1,k) == 1 & direction == -1
51                     if population(dim, s_pos(2,k)) == 0
52                         population(dim, s_pos(2,k)) = 1;
53                         S(i+1) = S(i+1)+1;
54                         R(i+1) = R(i+1)-1;
55                         s_pos = [s_pos, [dim;s_pos(2,k)]];
56                         k = k+1;
57                     elseif population(dim, s_pos(2,k)) == -1
58                         population(1, s_pos(2,k)) = -1;
59                         S(i+1) = S(i+1)-1;
60                         B(i+1) = B(i+1)+1;
61                         b_pos = [b_pos, [1;s_pos(2,k)]];
62
63                         s_pos(:,k) = [];
64                     else
65                         k = k+1;
66                     end
67                 elseif s_pos(1,k) == dim & direction == 1
68                     if population(1, s_pos(2,k)) == 0
69                         population(1, s_pos(2,k)) = 1;
70                         S(i+1) = S(i+1)+1;
71                         R(i+1) = R(i+1)-1;
72                         s_pos = [s_pos, [1;s_pos(2,k)]];
73                         k = k+1;
74                     elseif population(1, s_pos(2,k)) == -1
75                         population(dim, s_pos(2,k)) = -1;
76                         S(i+1) = S(i+1)-1;
77                         B(i+1) = B(i+1)+1;
78                         b_pos = [b_pos, [dim;s_pos(2,k)]];
79                         s_pos(:,k) = [];
80
81                     else
82                         k = k+1;
83                     end
84                 else
85                     if population(s_pos(1,k)+direction, s_pos(2,k)) == 0
86                         population(s_pos(1,k)+direction, s_pos(2,k)) = 1;

```



```

87         S(i+1) = S(i+1)+1;
88         R(i+1) = R(i+1)-1;
89         s_pos = [s_pos, [s_pos(1,k)+direction;s_pos(2,k)]];
90         k = k+1;
91     elseif population(s_pos(1,k)+direction, s_pos(2,k)) == -1
92         population(s_pos(1,k), s_pos(2,k)) = -1;
93         S(i+1) = S(i+1)-1;
94         B(i+1) = B(i+1)+1;
95         b_pos = [b_pos, [s_pos(1,k); s_pos(2,k)]];
96
97         s_pos(:,k) = [];
98     else
99         k = k+1;
100     end
101 end
102 else %y-position
103     if s_pos(2,k) == 1 & direction == -1
104         if population(s_pos(1,k),dim) == 0
105             population(s_pos(1,k),dim) = 1;
106             S(i+1) = S(i+1)+1;
107             R(i+1) = R(i+1)-1;
108             s_pos = [s_pos, [s_pos(1,k);dim]];
109             k = k+1;
110         elseif population(s_pos(1,k),dim) == -1
111             population(s_pos(1,k),1) = -1;
112             S(i+1) = S(i+1)-1;
113             B(i+1) = B(i+1)+1;
114             b_pos = [b_pos, [s_pos(1,k);1]];
115
116             s_pos(:,k) = [];
117         else
118             k = k+1;
119         end
120     elseif s_pos(2,k) == dim & direction == 1
121         if population(s_pos(1,k),1) == 0
122             population(s_pos(1,k),1) = 1;
123             S(i+1) = S(i+1)+1;
124             R(i+1) = R(i+1)-1;
125             s_pos = [s_pos, [s_pos(1,k);1]];
126             k = k+1;
127         elseif population(s_pos(1,k),1) == -1
128             population(s_pos(1,k),dim) = -1;
129             S(i+1) = S(i+1)-1;
130             B(i+1) = B(i+1)+1;
131             b_pos = [b_pos, [s_pos(1,k);dim]];
132
133             s_pos(:,k) = [];
134         else
135             k = k+1;
136         end
137     else
138         if population(s_pos(1,k), s_pos(2,k)+direction) == 0
139             population(s_pos(1,k), s_pos(2,k)+direction) = 1;
140             S(i+1) = S(i+1)+1;
141             R(i+1) = R(i+1)-1;
142             s_pos = [s_pos, [s_pos(1,k);s_pos(2,k)+direction]];
143             k = k + 1;
144         elseif population(s_pos(1,k), s_pos(2,k)+direction) == -1
145             population(s_pos(1,k), s_pos(2,k)) = -1;
146             S(i+1) = S(i+1)-1;
147             B(i+1) = B(i+1)+1;
148             b_pos = [b_pos, [s_pos(1,k);s_pos(2,k)]];

```

```

149
150         s_pos(:,k) = [];
151     else
152         k = k+1;
153     end
154 end
155
156 end
157
158 for j = 1:R(i)
159     if rand() < p
160         S(i+1) = S(i+1)+1;
161         R(i+1) = R(i+1)-1;
162         while population(x,y) ~= 0
163             x = randi(dim,1);
164             y = randi(dim,1);
165         end
166         population(x,y) = 1;
167         s_pos = [s_pos, [x;y]];
168     end
169 end
170 k = 1;
171 for j = 1:B(i)
172     if rand() < r %Boring interacts
173         direction = round((dir(round(rand)+1)));
174         if rand() < 0.5 %x direction
175             if b_pos(1,k) == 1 & direction == -1
176                 if population(dim, b_pos(2,k)) == 0
177                     population(1, b_pos(2,k)) = 0;
178                     R(i+1) = R(i+1)+1;
179                     B(i+1) = B(i+1)-1;
180                     b_pos(:,k) = [];
181                 else
182                     k = k+1;
183                 end
184             elseif b_pos(1,k) == dim & direction == 1
185                 if population(1, b_pos(2,k)) == 0
186                     population(dim, b_pos(2,k)) = 0;
187                     R(i+1) = R(i+1)+1;
188                     B(i+1) = B(i+1)-1;
189                     b_pos(:,k) = [];
190                 else
191                     k = k+1;
192                 end
193             else
194                 if population(b_pos(1,k)+direction, b_pos(2,k)) == 0
195                     population(b_pos(1,k)+direction, b_pos(2,k)) = 0;
196                     R(i+1) = R(i+1)+1;
197                     B(i+1) = B(i+1)-1;
198                     b_pos(:,k) = [];
199                 else
200                     k = k+1;
201                 end
202             end
203         else %y-position
204             if b_pos(2,k) == 1 & direction == -1
205                 if population(b_pos(1,k),dim) == 0
206                     population(b_pos(1,k),dim) = 0;
207                     R(i+1) = R(i+1)+1;
208                     B(i+1) = B(i+1)-1;
209                     b_pos(:,k) = [];
210                 else

```

```

211         k = k+1;
212     end
213     elseif b_pos(2,k) == dim & direction == 1
214         if population(b_pos(1,k),1) == 0
215             population(b_pos(1,k),1) = 0;
216             R(i+1) = R(i+1)+1;
217             B(i+1) = B(i+1)-1;
218             b_pos(:,k) = [];
219         else
220             k = k+1;
221         end
222     else
223         if population(b_pos(1,k), b_pos(2,k)+direction) == 0
224             population(b_pos(1,k), b_pos(2,k)+direction) = 0;
225             R(i+1) = R(i+1)+1;
226             B(i+1) = B(i+1)-1;
227             b_pos(:,k) = [];
228         else
229             k = k+1;
230         end
231     end
232 end
233 end
234 end
235 if i == 1
236     figure()
237     imagesc(population)
238     title('Start')
239 elseif i == steps/10
240     figure()
241     imagesc(population)
242     title('steps/10')
243 elseif i == 2*steps/10
244     figure()
245     imagesc(population)
246     title('2*steps/10')
247 elseif i == 3*steps/10
248     figure()
249     imagesc(population)
250     title('3*steps/10')
251 elseif i == 4*steps/10
252     figure()
253     imagesc(population)
254     title('4*steps/10')
255 elseif i == 5*steps/10
256     figure()
257     imagesc(population)
258     title('5*steps/10')
259 elseif i == 6*steps/10
260     figure()
261     imagesc(population)
262     title('6*steps/10')
263 elseif i == 7*steps/10
264     figure()
265     imagesc(population)
266     title('7*steps/10')
267 elseif i == 8*steps/10
268     figure()
269     imagesc(population)
270     title('8*steps/10')
271 elseif i == 9*steps/10
272     figure()

```

```

273     imagesc(population)
274     title('9*steps/10')
275     elseif i == steps-1
276         figure
277         imagesc(population)
278         title('Population at end')
279     end
280
281 end
282 end

```

## 4.2.5 Population Dynamics

### 4.2.6 mainPopDyn.m

```

1  %% 1.3.1.1
2  clear all
3  close all
4  p = 0.5;
5  b = [10 18];
6  n = [10 100 1000 5000];
7  steps = 100;
8  init_pop = 10;
9  n = 1000;
10 for i = 1:length(b)
11     [state population] = popDyn11(n,b(i),p,init_pop,steps);
12     pop1(i,:) = population
13 end
14
15 figure
16 plot(1:1:steps, population)
17 %% 1.3.1.2
18 clear all
19 clc
20 b = [1:1:50];
21 n = 1000;
22 p = 0.5
23 steps = 50;
24 init_pop = 10;
25 reps = 5000;
26 max_pop = 10000
27 hrangle=[0:1:max_pop];
28 histSq=zeros(length(b),length(hrangle));
29 for i = 1:length(b)
30     for j = 1:reps
31         [state population] = popDyn11(n,b(i),p,init_pop,steps);
32         finalSq(i,j)=population(end);
33     end
34     histSq(i,:)=hist(finalSq(i,:),hrangle);
35 end
36 figure
37 imagesc(b,hrangle,histSq'/reps,[0 0.01])
38 title('Phase transition for b')
39
40
41
42
43
44 %% 1.3.3

```

```

45 clear all
46 close all
47 n = 1000;
48 b = [1:1:50];
49 d = [1:1:n];
50 reps = 5;
51 p = 0.5;
52 steps = 100;
53 init_pop = 20;
54 for i = 1:length(b)
55     for j = 1:length(d);
56         for k = 1:reps
57             [state population] = popDyn12(n,b(i),p,d(j),init_pop,steps);
58             pop(k) = population(end);
59         end
60         plotPop(i,j) = mean(pop);
61         stab(i,j) = mean(pop-mean(pop));
62     end
63 end

```

#### 4.2.7 memespread2.m

```

1 function [state ret_pop] = popDyn11(n,b,p,init_pop,steps)
2 state = zeros(steps,n);
3 pop = init_pop;
4 ret_pop = zeros(1,steps);
5 mean_val = b*p;
6 standard_dev = sqrt(b*p*(1-p));
7 ent = 0;
8 for i = 1:steps
9     temp_state = zeros(1,n);
10    if i == 1
11        for j = 1:pop
12            place = round(rand*(n-1))+1;
13            temp_state(place) = temp_state(place) + 1;
14        end
15    else
16        for j = 1:pop
17            place = round(rand*(n-1))+1;
18            temp_state(place) = temp_state(place) + 1;
19        end
20        for j = 1:n
21            if temp_state(j) == 1
22                temp_state(j) = round(standard_dev*randn()+mean_val);
23            else
24                temp_state(j) = 0;
25            end
26        end
27    end
28    pop = sum(temp_state);
29    ret_pop(1,i) = pop;
30    state(i,:) = temp_state;
31 end
32
33
34 end

```

#### 4.2.8 memeSpreadGrid.m

```
1 function [state, ret_pop] = popDyn12(n,b,p,d,init_pop,steps)
2 state = zeros(steps,n);
3 test_state = state;
4 pop = init_pop;
5 mean_val = b*p;
6 standard_dev = sqrt(b*p*(1-p));
7 dir = [-1 1];
8 for i = 1:steps
9     temp_state = zeros(1,n);
10    if i == 1
11        for j = 1:pop
12            place = round(rand*(n-1))+1;
13            temp_state(place) = temp_state(place) + 1;
14        end
15    else
16        for j = 1:n
17            if state(i-1,j) ~= 0
18                number = state(i-1,j);
19                for k = 0:number
20                    place = j+round((dir(round(rand)+1))*rand*d);
21                    if place < 1
22                        place = n + place;
23                    elseif place > n
24                        place = place - n;
25                    end
26                    temp_state(place) = temp_state(place)+1;
27                end
28            end
29        end
30        test_state(i,:) = temp_state;
31        for j = 1:n
32            if temp_state(j) == 1
33                temp_state(j) = round(standard_dev*randn()+mean_val);
34            else
35                temp_state(j) = 0;
36            end
37        end
38    end
39    pop = sum(temp_state);
40    ret_pop(1,i) = pop;
41    state(i,:) = temp_state;
42 end
43 end
```