Effect Of Obscuring Vision On An Agent Based Model Of A Riot

Angus Blance

March 10, 2023

Contents

1 Abstract

2 Introduction

Instances of large-scale disorder occur frequently around the world, resulting in significant damage to property and, at times, human life. Given the destructive nature of such events, it is crucial to develop models that can accurately capture the dynamics of rioting and identify potential strategies for mitigating the damage. In this report, we present an agent-based model (ABM) that replicates the behaviour of riot populations, and we demonstrate how the use of smoke by police forces can effectively reduce the number of participants in a riot.

What Causes A Riot

riots can be caused by many things such as deprivation, poor police relations, legitimacy of a government or just to many people in one area. Once a riot has started it then gets out of control very fast usually epidemic like(ref 2.). Factor's for this can be knowledge of an ongoing riot through social Medea or friends and Rational choice theory as in if there are shops being looted and police are not doing anything you are more likely to join. riots can then be continued if there is inadequate policing

3 My Model

3.1 What is an Agent Based Model

An agent-based model (ABM) is a computational modelling technique used to simulate the behaviour of individual agents and their interactions within a system. In an ABM, each agent is typically characterized by a set of rules, behaviours, and attributes that influence its decision-making process and its interactions with other agents in the environment. These agents can be anything from people, animals, or even artificial entities.

ABMs are commonly used to simulate complex systems, such as social systems, ecological systems, or financial markets, where the behaviour of individual agents and their interactions can have a significant impact on the overall behaviour of the system. ABMs can also be used to explore the effects of different policies or interventions on the system, by adjusting the rules governing the behaviour of the agents and observing the resulting changes in the system's behaviour.

For this study, I employ an ABM to simulate a riot using Epstein's Model for civil violence, which consists of three agents: quiet agents (Q) who are not participating in the riot, active agents (A) who are rioting, and cop agents (C) who represent a potential intervention strategy. Each agent is placed on an N by N board

attributes for these agents are given:

quiet:
-walk around the board at random
-calculates the amount of police and actives within its vicinity
-Perform a series of calculations, taking into account the number of police and active agents, as well as their own risk level, hardship, and perception of the legitimacy of the government. Based on these factors, they decide whether or not to become an active agent.
-if the ratio of active to police is deemed suitable they become active

Active:
-walk around randomly

police:
-walk around randomly
-when near an active they arrest them and take them off the board

3.2 Mathematical Model

In our N by N board, each agent is assigned a unique identification key to distinguish their agent type, which is represented by the numbers 1, 2, and 3 for quiet, police, and active agents, respectively. Additionally, each agent is given a level of hardship (H) and a level of risk, both drawn from a uniform distribution on the interval (0, 1). We store this information in a data structure, as shown in Figure 1.

Once we have assigned each agent a unique identification key and their corresponding levels of hardship and risk, we place them in their designated

```
crowd =

    1.0000    0.7064    0.6949         0
    1.0000    0.5819    0.5669         0
    1.0000    0.8484    0.2954         0
    1.0000    0.8747    0.2338         0
    1.0000    0.6758    0.5831         0
    1.0000    0.7370    0.4459         0
    1.0000    0.1788    0.6315         0


police =

     2     0     0     0
     2     0     0     0
     2     0     0     0
```

Figure 1: crowd police format

```
{[          2 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
{[1 0.6412 0.7994 0]}    {[1 0.4626 0.3955 0]}    {[          0 0 0 0]}
{[          0 0 0 0]}    {[1 0.9792 0.0672 0]}    {[1 0.1153 0.2751 0]}
```

Figure 2: grid beginning

location on the N by N board. This results in a configuration similar to Figure 2, where each cell in the grid represents an agent with its identification key, hardship level (H), and risk level.

Note that the police do not have a hardship or risk level, this is as they do not need it as they do not perform the calculation to become an active. now i made it so each person (rioter, police person or civilian) would move a random direction each iteration given that there was nobody already in that space.

How Civilians Chose To Riot

civilians also have the ability to become a rioter for this i used the aspects from the Epstein model (ref 1.) where we first find the Grievance.

$$G = H(L - 1) \tag{1}$$

where H is the hardship of each specific civilian and L is the legitimacy of the government that person is in which we set as low (0.1 ish) so that we do create a riot.

now we find the probability (p) to be arrested which each citizen needs to calculate before they decide whether or not they wish to riot:

$$P = 1 - EXP(-k(C/A)_v) \tag{2}$$

note the subscript v notates the vision which each civilian has as shown in fig 3.

C is the number of police in the vision v and A is the number of rioters in vision v. The civilian counts himself when counting A as he wants to see how the riot would be once he is added. K is just a constant which we set such that P = 0,9 when C=A=1, so k = -ln(0.1)
from this we can now define the net risk $N = RPJ$ where J is the jail time one faces once he has been caught by a police officer. For this model it is just set to about 5

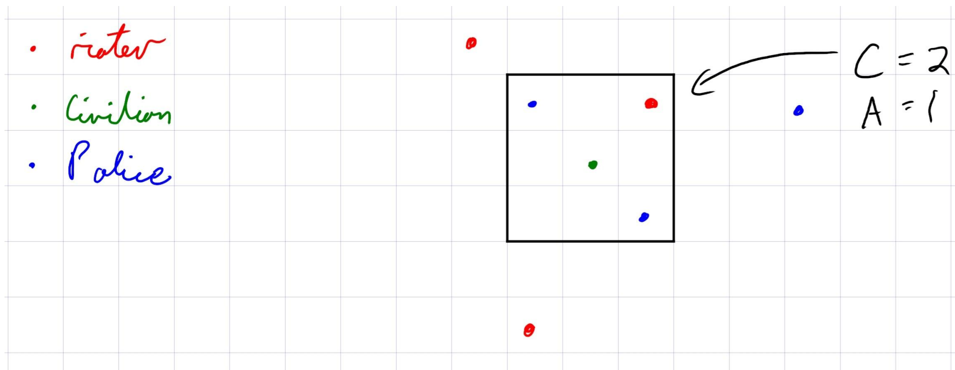

Figure 3: vision example

we now have all we need to create the agent rule:

$$Agent\ rule : if\ G - N > T\ riot;\ otherwise\ stay\ a\ civilian \tag{3}$$

so the difference in G and N is the expected utility of expressing ones private grievance whilst T is the utility of not.

Police Attributes

The police in the model are pretty simple, they just move around and scan the squares around them every turn, if there is a rioter within those squares he takes them of the board turning it into [0,0,0], an empty space (ie arrested).
along with this every turn each police officer has an opportunity to throw smoke if a rioter is within its specified range for throwing smoke.

Smoke

As smoke does not move like the agents in this model I created a second grid the same size as the grid that our agents are on.
each square has 2 values for it firstly a 0 or 1 representing whether there is no smoke (0) or smoke (1) and another value from 0 to how many iterations the smoke lasts for.
The smoke will be a square n $\times$ n area.
What the smoke actually affects is the probability for a civilian to start rioting as it reduces the number of rioters that the civilian can see within its range v.

I have also made it so they still count police officers even if they are inside of the smoke, this is done as i assume police officers don't generally throw smoke at each-other.

Small Detail

In the Epstein model the rioters and the civilians can move to anywhere within there vision whilst the police officers can only move within the small distance which they can arrest rioters within. I made it so everyone can only move 1 square each tun as i do not believe there is that much of a difference in speed between the general public and police.

4 Computational Approach and Implementation

The code is quite complex and hard to describe through words alone so in this section I wish to use diagrams in order to visually explain my code step by step from the very beginning.

First of all we have our N × N board say N=10 shown in figure 4

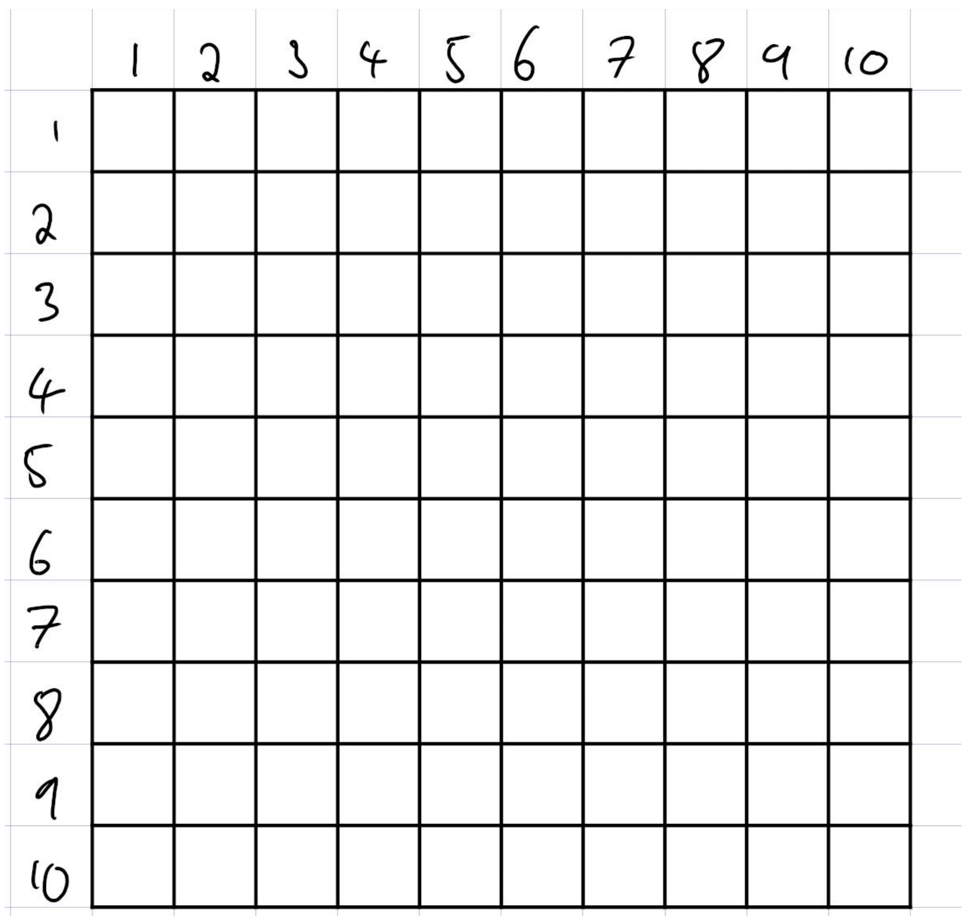

Figure 4: French riot

Now we introduce our agents with attributes as described in our "What is an Agent Based Model" section, Let our Actives be denoted by the colour red, our quiets to be the colour green and our Cops to be the colour Blue. Say we loop through our hole N $\times$ N grid and place 20 quiets and 10 cops at random we will end up with something like Figure 5

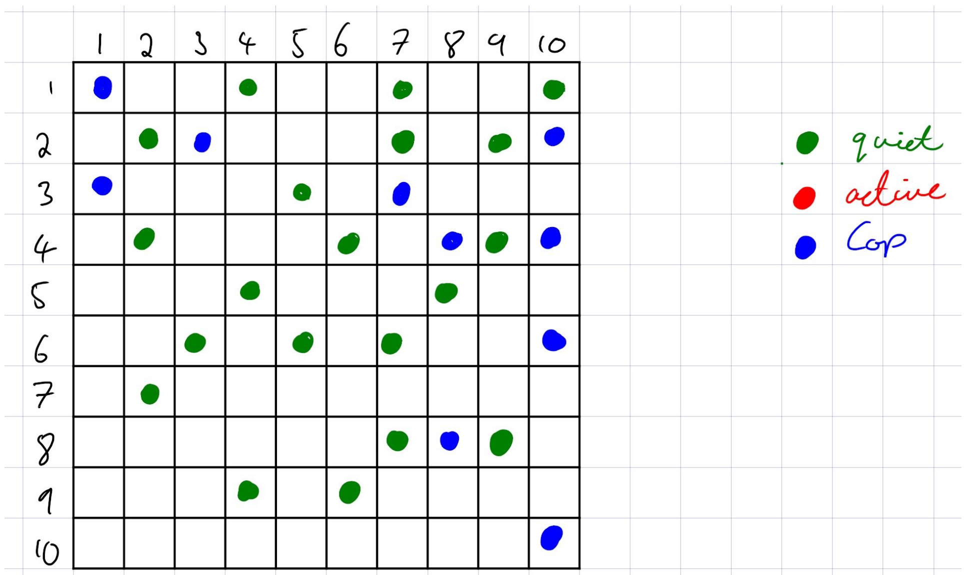

Figure 5: French riot

Great! That's our hole grid set up. But also note that each individual quiet has 2 attributes to them (risk level and Hardship) which is easier to see on a 10 $\times$ 10 example run on our code in Matlab as shown in figure 6

```
10×10 cell array

Columns 1 through 7

  {[1 0.8147 0.6557 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          2 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.5469 0.3922 0]}    {[          0 0 0 0]}    {[1 0.9157 0.3171 0]}    {[          0 0 0 0]}    {[          2 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.1419 0.8235 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.9058 0.0357 0]}    {[          0 0 0 0]}
  {[          2 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.9595 0.0344 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.4218 0.6948 0]}
  {[          0 0 0 0]}    {[1 0.9575 0.6555 0]}    {[1 0.9572 0.2769 0]}    {[          0 0 0 0]}    {[          2 0 0 0]}    {[1 0.9134 0.9340 0]}    {[1 0.1576 0.7060 0]}
  {[          0 0 0 0]}    {[1 0.8003 0.0971 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.0975 0.7577 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.9706 0.0318 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.2785 0.7431 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          2 0 0 0]}    {[1 0.1270 0.8491 0]}    {[1 0.4854 0.0462 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}

Columns 8 through 10

  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          2 0 0 0]}    {[          2 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          2 0 0 0]}
  {[          0 0 0 0]}    {[1 0.6324 0.6787 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          2 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          2 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[1 0.7922 0.9502 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[1 0.9649 0.1712 0]}
  {[          0 0 0 0]}    {[          0 0 0 0]}    {[          0 0 0 0]}
```

Figure 6: French riot

Notice how we store the risk level and Hardship level by making our board a cell array storing multiple values for each cell. Also see how all of the cops have no hardships or risk, This is because they do not have the attribute to become an active meaning giving them such values will be redundant.

4.1 Moving

Now the next thing to do is make each person move, I decided everybody will only move one space (Cell) at a time how they move is shown in figure 7 where the arrows on the diagram on the left show the direction in which each agent will move and the diagram on the left shows the grid after each person has moved

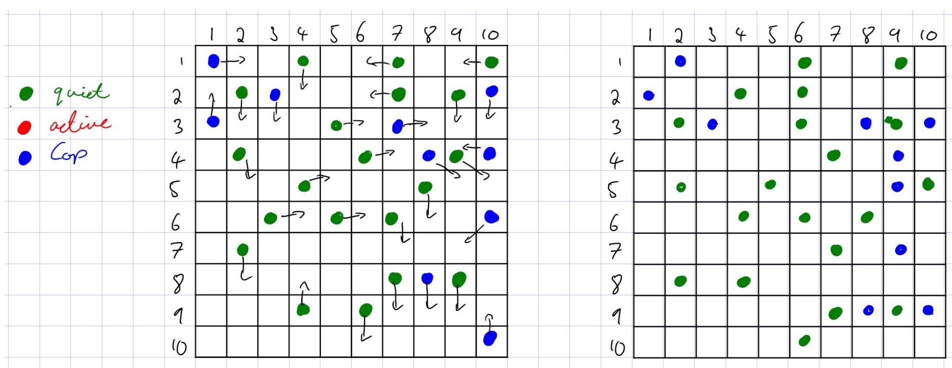

Figure 7: French riot

see how the Cop in column 10 row 4 (10,4) moves to space (9,4) which is already occupied by a quiet? This is allowed to happen due to how I am looping through the board, I do this by going through (column,row) where row = 1,2,...,10 and column = 1,2,...,10 from the top left of the board to the bottem right meaning that the quiet in space (9,4) has already moved to (10,5) before the cop makes his move into the space the quiet was just in.

4.2 Quiets and Actives The Descension to Riot

Just as before we first need to loop though the hole board, first starting form the top left and ending up on the bottom right. whenever we land on a quiet we take its risk and Hardship as well as searching in a M $\times$ M grid around said quiet to collect the number of actives and Cops within its vision, its vision being the M$\times$M area. One example of this is shown in Figure (fig num)

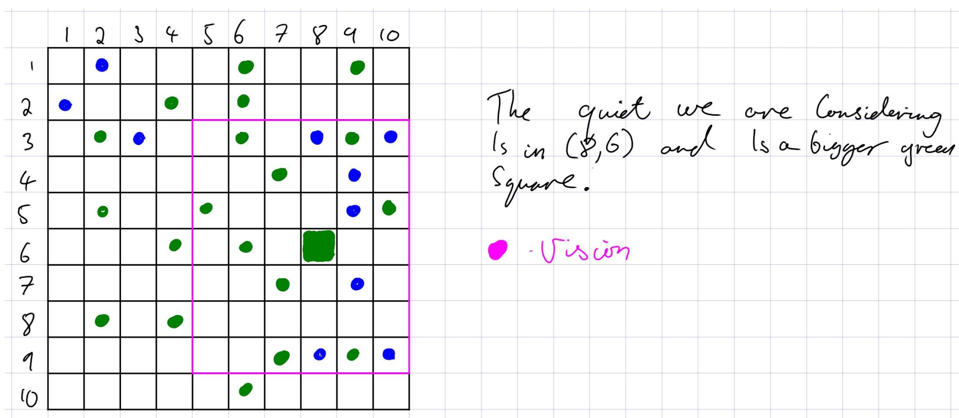

Figure 8: French riot

Before we would have even gotten to this particular active we would have consider all other active in (column, row) for column = 1,2,...,10 and row = 1,2,...,6. In this particular case we assume that non of these quites transitioned into an active.
Now looking at what data we have for our particular case in figure (fig num) we have the number of police = 7 and the number of quiets = 9 not including the quiet we are currently considering. This is as when a quiet is considering to become and active it is thinking, if I am an active what is the probability of me getting arrested so it considers itself an active meaning we have actives = 1 in the area.

For This particular quiet to become active we need equation 3 to be satisfied. Lets explore this more mathematically looking at each variable (3) individually. First lets look at N

$$N = RPJ \tag{4}$$

Where R is the level of risk aversion they will take given to each individual quiet, higher level of risk aversion the less likely they are to take risks. J is the jail time and P is the probability to get arrested given by

$$P = 1 - EXP(-k(C/A)_v) \tag{5}$$

We have C and A from figure (fig num) and k is a constant set such that p = 0.9 when C=1 and A = 1 so k = ln(0.1) subbing these values in we get P = 0.9999999 for our specific case, which makes sence as there are no actives within the vicinity of our quiet and there are many police meaning the probability of arrest should be high. Also note as C -¿ ∞ P -¿ 1 and as A -¿ ∞ P -¿ 0.
say in our particular case J = 5 which is a smallish jail sentence which should mean a rioter is more likely to riot then our N = R$\times$4.9999995 and using this and substiduting G from (1) we obtain

$$G - N = H(L - 1) - R \times 4.9999995 \tag{6}$$

Say we want our quiet to become active we want (6) to be Large such that it is ¿ T meaning we want H(L-1) the total grievance of this person to be large and R $\times$ 4.9999995 to be small. First for H(L-1) to be large we need want the hardship of our specific individual to be large and the legitimacy of our government to be low, Note that just having a high level of hardship does not mean our quiet will become active, we need both a high hardship AS WELL as a low perceived legitimacy of the government. next for R $\times$ 4.9999995 to be low we need our individuals level of risk aversion to be low. Notice that for our agent to become active that shorter jail centances as well as a smaller probability of arrest make this more likely to happen

4.3 Police Arresting

police move just the same way as the quiets and actives, However they cannot become active. The main characteristics Cops have is the ability to arrest this is done by them just happening to walking about and searching similarly to quiets "scanning" the area around them, except there "vision" us much smaller than that of the quiets and actives. IF an active is within a cops vision it then arrests them and either takes them of the board if the jail time is very high or puts them into "jail" a matrix storing all actives that have been caught by police. What an arrest would look like in my model

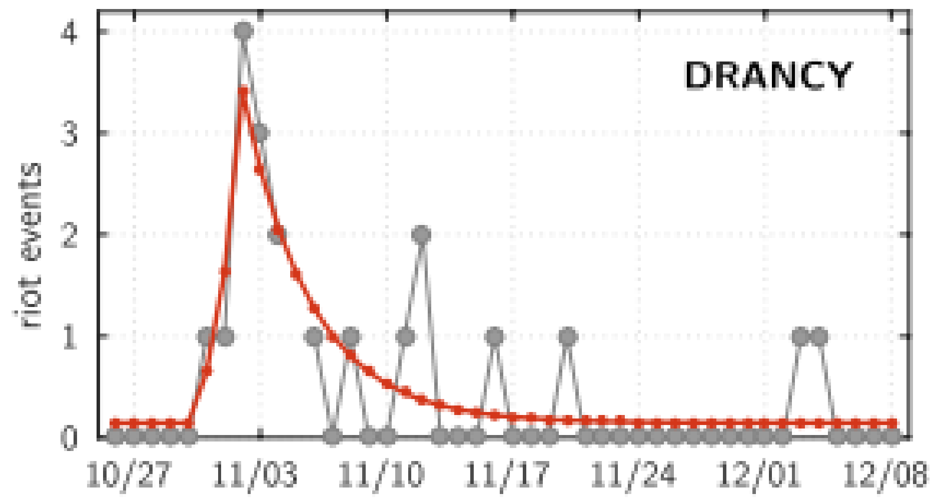

Figure 9: French riot

can be shown in figure 8 where on the left a cop is next to an active and in the next the active has been put into jail. The jail sentence J is set to 5 meaning it takes 5 turns for our agents to be set loos from our jail, not how the jail inmate's number next to him is the time he has spent in jail which goes down by 1 after the turn has taken place

5 My Results

What is expected of this model is for when i implement the smoke into the riot that the number of agents that are rioters will significantly decrease. to show this i will be looking at the total number of rioters within the grid at iteration n. These results should be similar to fig.4 from a suburb in Paris during the 2005 French riots. (ref 2.). Except for this model we are only looking from the peak of the riot until the eventual stop of the riot. This is as the probability for everyone to riot is very high at the beginning of the program due to there being many more civilians to police people.

Once i run my program i get graphs like fig. 5 where the blue line is a riot without any smoke and the red line is a riot with. We see that the 2 lines look as if they are perpendicular. This is because of the agent rule, the agent rule is the comparison of a linear equation on the LHS and a constant on the RHS so smoke decreases the number of people in the riot but does

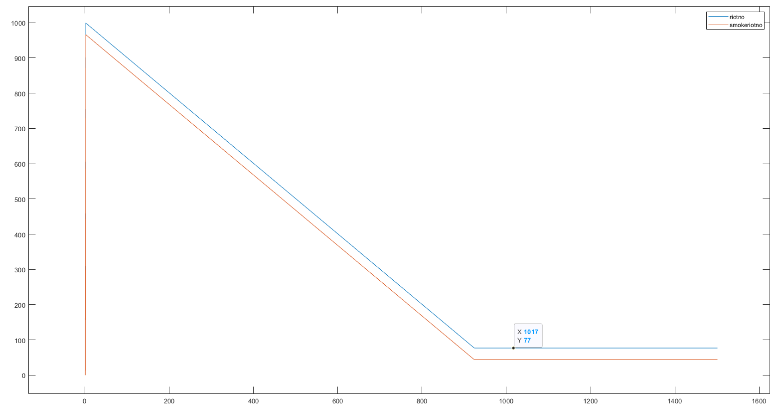

Figure 10: Riot With Smoke vs Without

not decrease the rate at the population of the riot tends to 0. If you want to increase the rate at which the number of rioting members tends to 0 you can do this by increasing the number of police people

Notice how there are still rioters at the end of the graph in fig 5? This is due to our rioters having a range greater than that of the distance of which the police can arrest them from, so the rioters become rioters once there are so few police in there vision but once there are too many within there vision they become a civilian again evading the law.

6 Evaluation

What went well

The model discussed is able to create a general riot and has key features in what creates a riot. From this Smoke was implemented which the police could throw which was shown to decrease the number of people rioting compared to people rioting when there is no smoke. showing the effect of obscuring vision in a riot

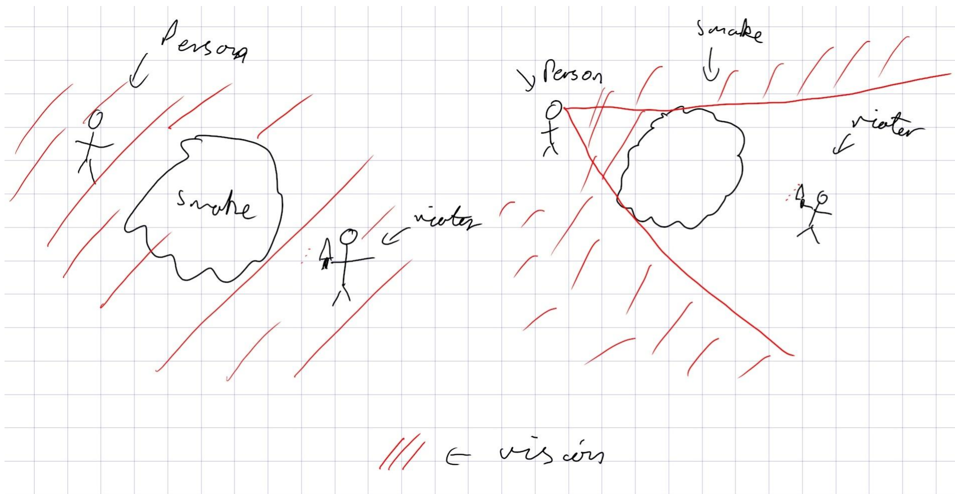

Figure 11: civilian vision

To Improve

I could make more realistic smoke, as in this current model civilians can see the side of the smoke that they are facing as well as the area on the other side of this smoke as shown in fig. 6 showing how my model works on the left and a more realistic vision on the right.

I have also made a pretty simple riot program where there are civilians rioters and police people. when rioters are near civilians they have a chance to make them rioters. When a police person is near a rioter they take them of the board. This has an epidemic like effect as the number of rioters multiplies depending on how many rioters there are giving us fig. 7.
I can then improve this model to show how smoke affects it by giving the rioters vision to seek out civilians to turn into rioters and then see how the obstruction of there vision via smoke affects how quickly the riot can be stopped. This works as an effective model as we know riots behave in an epidemic fashion as seen in the article (ref 2.)

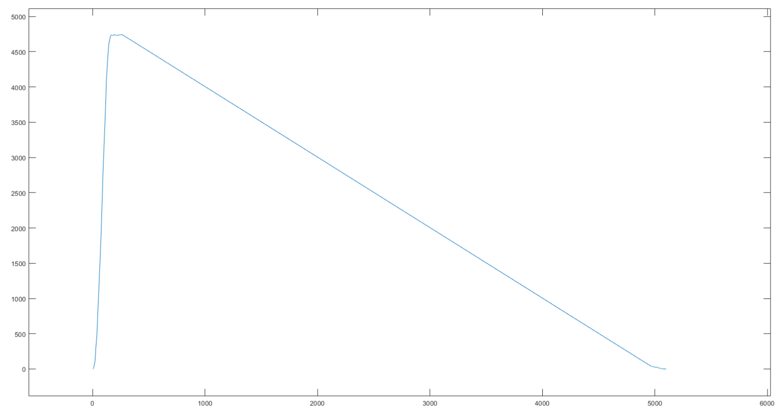

Figure 12: epedemic like program

7 Bibliography

References

1. Epstein, Joshua M. Modeling civil violence: An agent-based computational approach. 2002

2. Bonnasse-Gahot, Laurent and Berestycki, Henri and Depuiset, Marie-Aude and Gordon, Mirta B and Roché, Sebastian and Rodriguez, Nancy and Nadal, Jean-Pierre. Epidemiological modelling of the 2005 French riots: a spreading wave and the role of contagion. 2018

8 Appendices (Matlab Code)

here is my code for my riot simulation

```
clc
%%creating paramaters
crowd_n = 7000;
police_n = 3000;
grid_size = 100;
vision_l = 2;
smoke_size = 1;
%number of iterations
no = 1500;

%legitimacy of goverment
L = 0.1;

% T is a value which if G - N > T a active begins to riot (xpected utility
% of not expressing there private grievance)
T = 0.3;


% [crowd,police] = create_crowd_police(crowd_n, police_n);
% grid  = rand_initiallize_grid(crowd_n, crowd, police_n , police , grid_size);
% grid = agents_smoke(grid, grid_size ,L, vision_l ,T);
% sgrid = smoke_grid_init(grid_size);
%
% sgrid = smoke(grid, grid_size ,smoke_size ,sgrid)
%
%
%  smokeriotno = run_till_no_smoke(grid_size ,L, vision_l ,T, crowd_n , police_n ,no, sgrid , smoke_size);
%  riotno = run_till_no_nosmoke(grid_size ,L, vision_l ,T ,crowd_n , police_n , no);
%  plot(1:length(riotno),riotno ,1:length(smokeriotno), smokeriotno)
%
```

```
%
% legend('riotno','smokeriotno')
riotno = run_till_no_nosmoke_smoke(grid_size,L,vision_l,T,crowd_n, police_n, no,sgrid,smoke_size);
plot(1:length(riotno),riotno)


%%my functions
%{

run through a riot until all rioters have been arested
%}
function riotno = run_till_no_smoke(grid_size,L,vision_l,T,crowd_n, police_n, no,sgrid,smoke_size)
    %create the grid
    [crowd,police] = create_crowd_police(crowd_n,police_n);
    grid  = rand_initiallize_grid(crowd_n, crowd,police_n, police, grid_size);
    %create the smoke grid
    sgrid = smoke_grid_init(grid_size);
    %make agents rioters
    grid = agents_smoke(grid,grid_size,L,vision_l,T,sgrid);
    riotno = zeros(100,1);
    agentno = zeros(100,1);

    count = 1;
    riotno(count,1)=0;

    for i = 1:no

        count = count+1;


        [sgrid,grid] = policemen_smoke(grid, grid_size,smoke_size,sgrid);

        grid = agents_smoke(grid,grid_size,L,vision_l,T,sgrid);
```

```
        riotno(count,1)=count_keys(grid,3);
        agentno(count-1,1) = count_keys(grid,1);



    end

end



function riotno = run_till_no_nosmoke_smoke(grid_size,
   L,vision_l,T,crowd_n, police_n, no,sgrid,smoke_size
   )
    %create the grid
    [crowd,police] = create_crowd_police(crowd_n,
        police_n);
    grid  = rand_initiallize_grid(crowd_n, crowd,
        police_n, police, grid_size);
    %create the smoke grid
    sgrid = smoke_grid_init(grid_size);
    %make agents rioters
    grid = agents_smoke(grid,grid_size,L,vision_l,T,
        sgrid);
    riotno = zeros(no,1);
    agentno = zeros(100,1);

    count = 1;
    riotno(count,1)=0;

    for i = 1:no/2

        count = count+1;


        grid = policemen(grid, grid_size);
        grid = agents_nosmoke(grid,grid_size,L,
            vision_l,T);

```

```
            riotno(count,1)=count_keys(grid,3);
            agentno(count-1,1) = count_keys(grid,1);

        end
        for i = 1:no/2

            count = count+1;


            [sgrid,grid] = policemen_smoke(grid, grid_size
                ,smoke_size,sgrid);

            grid = agents_smoke(grid,grid_size,L,vision_l,
                T,sgrid);

            riotno(count,1)=count_keys(grid,3);
            agentno(count-1,1) = count_keys(grid,1);



        end

end



function riotno = run_till_no_nosmoke(grid_size,L,
    vision_l,T,crowd_n, police_n, no)
    [crowd,police] = create_crowd_police(crowd_n,
        police_n);
    grid = rand_initiallize_grid(crowd_n, crowd,
        police_n, police, grid_size);

    grid = agents_nosmoke(grid,grid_size,L,vision_l,T)
        ;
    riotno = zeros(100,1);
    agentno = zeros(100,1);

    count = 1;
    riotno(count,1)=0;
```

```
    for i = 1:no
        count = count+1;


        grid = policemen(grid, grid_size);
        grid = agents_nosmoke(grid,grid_size,L,
            vision_l,T);

        riotno(count,1)=count_keys(grid,3);
        agentno(count-1,1) = count_keys(grid,1);



    end

end

%%
function grid = agents_smoke(grid,grid_size,L,vision_l
    ,T,sgrid)
    grid = move_rand(1, grid_size, grid);
    grid = move_rand(3,grid_size, grid);
    grid = agent_to_riot_smoke(grid,grid_size,L,
        vision_l,T,sgrid);
end

function grid = agents_nosmoke(grid,grid_size,L,
    vision_l,T)
    grid = move_rand(1, grid_size, grid);
    grid = move_rand(3,grid_size, grid);
    grid = agent_to_riot_nosmoke(grid,grid_size,L,
        vision_l,T);
end


function [sgrid, grid] = policemen_smoke(grid,
    grid_size,smoke_size,sgrid)
    grid = move_rand(2, grid_size, grid);
    sgrid = smoke(grid,grid_size,smoke_size,sgrid);
    grid = riot_to_arrest(grid_size, grid);
```

```
end



function grid = policemen(grid, grid_size)
    grid = move_rand(2, grid_size, grid);
    grid = riot_to_arrest(grid_size, grid);

end











%{
grievance():
    -Hardship G (physical or economic deprivation)
       drawn from U(0,1) ]
    uniform distrobution on 0,1

    -Legitamacy L percived legitamacy of a regime.
       This will be arbritary
    number from 0,1
%}
function G  = grievance(H,L)
    G = H*(1-L);
end

%{
arrest_probability():
    -constant k set such that P = 0.9 when C=A=1

    -active rioters A

```

```
    -cops in the area C
C/A changes for every rioter depending on how many C's and A's are within
the vision of each crowd member within a set vision
%}
function P = arrest_probability(k,C,A)
    P = 1 - exp(-(k*C/A));
end


%{
create_crowd_police()
-crowd_n:   number of people in crowd
-police_n:  number of police

-crowd:     our crowd with h r vals and key 1
-police:    our police with h r vals and key 2

creates the crowd aswell as the police such that:
-in the crowd each member has a number 1 in row1 to represent that they
are non rioters aswell as a hardship (H) row2 and a level of
risk they are willing to take (R) in row3. form:
[0,h1,r1; 0,h2,r2; ...; 0,hn,rn]

-in the police we have our key as 2 instead of 1 and we have H=R=0 as all
the police do is arrest and have no R H
%}
function [crowd,police] = create_crowd_police(crowd_n, police_n)
    %take our H and R randomly from the uniform distrobution beween 0-1
    H = unifrnd(0,1,1,crowd_n)';
    R = unifrnd(0,1,1,crowd_n)';
    %make a list of 1's from each person in the crowd
    crowdkey = ones(1,crowd_n)';
    %put crowd in the form
    crowd = [crowdkey,H, R, zeros(1,crowd_n)'];
```

```
    %same but police
    police = [2*ones(1,police_n)',zeros(1,police_n)',
        zeros(1,police_n)',zeros(1,police_n)'];
end

%{
rand_initiallize_grid()
-crowd_n:   number of people in crowd
-crowd:     from create_crowd_police()
-ploice_n:  number of police
-police:    from create_crowd_police
-grid_size: how big we want our grid

-grid:      our initialized grid
%}

function grid  = rand_initiallize_grid(crowd_n, crowd,
    police_n, police, grid_size)

    %first start our grid out as being a grid of zeros
        for our given grid
    %size
    for i = 1:grid_size
        for j = 1:grid_size
            grid{i,j} = [0,0,0,0];
        end
    end



%     now we put in our crowd, distrobuted randomly
%     throughout the grid
    while crowd_n ~= 0

        i = randi(grid_size,1);
        j = randi(grid_size,1);
        if grid{i,j}(1) == 0
            grid{i,j}=crowd(crowd_n,1:4);
            crowd_n = crowd_n - 1;
        end
```

```
    end


    %now we put the desired number of plolice randomly
        into our grid
    while police_n ~=0
        i = randi(grid_size,1);
        j = randi(grid_size,1);
        if grid{i,j}(1) == 0
            grid{i,j}=police(police_n,1:4);
            police_n = police_n - 1;
        end
    end
end

%{
grid:        the grid
grid_size:   size of the grid
vision_l:    length of the vision

v:           vision at point i j

takes in the grid and a point of the grid, it then
    returns a subsection
ofthe grid around this point
%}
function v = vision(i,j,grid, vision_l,grid_size)

    w = i - vision_l;
    e = i + vision_l;
    s = j + vision_l;
    n = j - vision_l;
    if w < 1
        w = 1;
    end

    if e > grid_size
        e = grid_size;
    end

```

```
    if s >grid_size
        s = grid_size;
    end

    if n < 1
        n = 1;
    end
    dns = s-n+1;
    dew = e-w+1;
    v = grid(w:e,n:s);

    %v = reshape({grid(w:e,n:s)},[dew,dns]);

end



%{


takes in the grid and checks the number of rioters around each agent with
the number of police and uses equations to evaluate if the agent becomes a
rioter aswell
%}


function grid = agent_to_riot_nosmoke(grid,grid_size,L,vision_l,T)
    %loop through each space in the grid
    for i = 1:grid_size
    for j = 1:grid_size
        %if we land on an agent
        if grid{i,j}(1)==1
            %find its grievance number
            G  = grievance(grid{i,j}(2),L);

            %find its risk probablility
            R  = grid{i,j}(3);
```

```

            %find the vision at that point
            v = vision(i,j, grid, vision_l, grid_size);

            %count number of police in vision
            C=count_keys(v,2);

            %count number of actives(rioters) in vision
            A=count_keys(v,3)+1;
            A
            %with our C and A we can now find the probability for our guy
            %at i j to be arrested
            P = arrest_probability(-log(0.1),C,A);

            %our agents net risk
            N = R*P;

            %expected utility of publicly expressing ones private grievanc
            gmn = G-N;

            if gmn >T
                grid{i,j}(1) = 3;
            else
                grid{i,j}(1) = 1;
            end

        end

    end
    end
    %grid;
end




```

```
%{

turns agents into rioters taken into acount the smoke
%}
function grid = agent_to_riot_smoke(grid,grid_size,L,
    vision_l,T,sgrid)
    %loop through each space in the grid
     for i = 1:grid_size
     for j = 1:grid_size
        %if we land on an agent
         if grid{i,j}(1)==1
             %find its grievance number
             G  = grievance(grid{i,j}(2),L);

             %find its risk probablility
             R  = grid{i,j}(3);

             %find the vision at that point
             v = vision(i,j, grid, vision_l, grid_size)
                 ;

             %count number of police in vision
             C=count_keys(v,2);

             %count number of actives(rioters) in
                 vision excluding those
             %with smoke
             A=count_keys_smoke(v,3,sgrid)+1;
             A
             %with our C and A we can now find the
                 probability for our guy
             %at i j to be arrested
             P = arrest_probability(-log(0.1),C,A);

             %our agents net risk
             N = R*P;

             %expected utility of publicly expressing
                 ones private grievanc
```

```
            gmn = G-N;

            if gmn >T
                grid{i,j}(1) = 3;
            else
                grid{i,j}(1) = 1;
            end

        end

    end
    end
    %grid;
end







%{
grid:   the matrix you want to count the keys in
key:    the key you want to count

n:      the count

counts the number of a given key in a given matrix
%}
function n = count_keys(grid,key)
    n=0;
    [r,c] = size(grid);
    for x = 1:r
    for y = 1:c
        if grid{x,y}(1)==key
            n=n+1;
        end

    end
    end
```

```
end

%{

count keys in matrix excluding the ones with smoke on them
%}


function n = count_keys_smoke(grid,key,sgrid)
    n=0;
    [r,c] = size(grid);
    for x = 1:r
    for y = 1:c
        if grid{x,y}(1)==key && sgrid{x,y}(1) == 0
            n=n+1;
        end

    end
    end
end

%{
grid_size:  size of grid
grid:       current grid

grid:       updated grid

looks at every police person and arrestsa rioter if it is near by
%}

function grid = riot_to_arrest(grid_size, grid)
    %for every (i,j)th place on the grid
    for i = 1:grid_size
        for j = 1:grid_size
            %if the (i,j)th place is a policeman
            if grid{i,j}(1) == 2
                %check every square 3*3 around the police man
```

```
                for n = -1:1
                    for m = -1:1
                        %assighn new (i,j)th position we are considering
                        positioni = i + n;
                        positionj = j + m;
                        %check point is on grid and it is a rioter
                        if positioni<=grid_size && positionj <= grid_size && positioni > 0 && positionj > 0 && grid{positioni, positionj}(1) == 3
                            %arrest that rioter
                            grid{positioni,positionj} = [0,0,0];
                            return

                        end
                    end
                end
            end
        end
    end
end

%{
crowd_trype:    the type of crowd you want to move
grid_size:      the size of the grid
grid:           the current grid

grid:           the grid after all the agents of a certain type have moved
%}

function grid = move_rand(crowd_type, grid_size, grid)
    v = 0;
    %for every (i,j)th place on the grid
    for i = 1:grid_size
        for j = 1:grid_size
```

```
                %while we reach a place with the correct crowd type, and we
                %have made a certain number of attempts v
                while grid{i,j}(1) == crowd_type && v < 9
                    %create a random number -1 to 1 for n-s plane and w-e plane
                    %and take the absolute value to get rid of negatives
                    directionns = i + randi([-1,1],1);
                    directionwe = j + randi([-1,1],1);
                    v = v + 1;
                    %check if the new position is inside of the grid and not =0
                    if directionns<=grid_size && directionwe <= grid_size && directionns > 0 && directionwe > 0
                        %if that space is empty (=0)
                        if grid{directionns,directionwe}(1) == 0
                            %move that person to that location
                            grid{directionns,directionwe} = grid{i,j};
                            %remove that person from where he was previously
                            grid{i,j} =  [0,0,0];
                        end
                    end
                end
            end
        end

end



function sgrid = smoke(grid,grid_size,smoke_size,sgrid)
    %go through each square on grid
    for i = 1:grid_size
```

```
        for j = 1:grid_size


            if grid{i,j}(1) == 2
                %throw smoke at fools
                sgrid = throwsmoke(i,j,grid,smoke_size,
                    grid_size,sgrid);
            end
        end
    end
end




function sgrid = throwsmoke(i,j,grid,smoke_size,
    grid_size,sgrid)
    L = length(smoke_size);
    %search around the police officer for rioters
    for n = -L:L
        for m = -L:L
            %assighn new (i,j)th position we are
                considering
            positioni = i + n;
            positionj = j + m;


            %check point is on grid and it is a rioter
            if positioni<=grid_size && positionj <=
                grid_size && positioni > 0 && positionj
                 > 0 && grid{positioni,positionj}(1) ==
                 3
                %create an square area of length
                    vision_l around our rioter
                w = positioni - smoke_size;
                e = positioni + smoke_size;
                s = positionj + smoke_size;
                north = positionj - smoke_size;
                %make each point inside of the grid
                if w < 1
```

```
                    w = 1;
                end

                if e > grid_size
                    e = grid_size;
                end

                if s >grid_size
                    s = grid_size;
                end

                if north < 1
                    north = 1;
                end

                for k = w:e
                    for o = north:s


                        sgrid{k,o}(1) = 1;
                    end
                end


            end
        end
    end
end




function sgrid = smoke_grid_init(grid_size)
    sgrid = {};
    for i = 1:grid_size
        for j = 1:grid_size
            sgrid{i,j} = [0,0];
        end
    end
end
```