Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises: Modelling and Simulating Social Systems with MATLAB

Project Report

Forest Fire Model and its Self-Organized Criticality

Zhe Sun & Bojun Cheng

Zürich

Dec 2013

Agreement for free-download

We hereby agree to make our source code of this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Zhe Sun Bojun Cheng

Table of Content

1. Abstract

Diffusion is a ubiquitous phenomenon which refers to the process in which a substance or property spreads from one region to another, e.g, transfer of heat, forest fire spreading, etc..
In this project, we will investigate the Forest Fire Model (FFM) and its Self-Organized Criticality (SOC) using MATALB. In 1992, a famous forest fire model was introduced based on the Cellular Automata (CA) by *Drossel* and *Schwabl*[1]. In 1995, *Albano* introduced a more generalized FFM considering the immunity from fire. Then in 2000, a FFM considering finite-size effect was proposed by *Schenk*[9]. Thus, study of SOC started even earlier. SOC was originally introduced by *Bak*, *Tang* and *Wiesenfeld*[2] in 1987 as a general theory to understand fractals and 1/f noise as the nature coupled degrees of freedom. Then, researchers noticed many actual systems including the spreading of fire could show SOC behavior.
We will describe the concept of SOC and FFM in the report. In our forest fire model, we will consider more factors including the size of forest and effect of weather conditions. And we will discuss simulation results using MATLAB code which show the quantitative characteristics of SOC in FFM.

2. Individual Contributions

Zhe Sun and Bojun Cheng contribute equally to programming. Most of the code are a work of collaboration and discussion. Zhe Sun contributes in writing the report and Bojun Cheng contributes in revising the report.

3. Introduction & Motivations

3.1 Brief Review of Self-Organized Criticality

The Self-Organized criticality (SOC) was first coined by *Bak*, *Tang* and *Wiesenfeld*[2] (*BTW*) in 1987. *BTW* pointed out that a dynamic system can evolve into a non-equilibrium steady state without tuning its parameters. It must be such that the system is driven to a state at the boundary between the stable and unstable states[3]. The term 'criticality' or 'critical state' is used to describe such state where two different phases occur simultaneously. A critical state shows long range spatial and temporal fluctuations similar to those in equilibrium critical phenomena[3].
One of the most important features of such states is the robust power-law correlations. The power law can be seen in various systems including earthquake, avalanches and ecosystem. For example, the abundance of species versus the size of the species, the variation of changes in ecosystems versus the frequency, the size of the avalanches versus their frequencies are plotted to show power-law behavior[4].
Another fingerprint of SOC is the 1/f noise or 'flicker noise' which means the power spectrum scales as $1/f^{\alpha}$ ($1<\alpha<2$) at low frequencies[5]. 1/f noise has been observed in the current through transistors, flow of rivers, and even stock exchange price indices.

An intuitive picture of the SOC is given by a sand pile[5]. Suppose we start to build a sand pile by randomly adding sand, a grain at a time. The pile will grow and the slope of pile will increase. Eventually, the slope will reach a critical value such that if more sand is added it will slide off. Alternatively, if we start with a steeper sand pile, it will collapse until it reaches the critical state such that it is stable with respect to further perturbations. In the language of fractal, this critical state is an attractor for the dynamics. The quantity which exhibits 1/f noise is simply the flow of the sand falling off the pile.

3.2 Brief Review of Forest Fire Model

The Forest Fire Model (FFM) proposed by *Drossel* and *Schwabl*[1] is based on Cellular Automata (CA). The forest is in a two-dimensional grid representation, in which tree grow with certain density. Trees can be ignited by lightening with a probability. A burning tree will ignite all its neighboring trees. A burning tree will become an empty site. On each empty site, a tree grows with a certain probability. This FFM will spontaneously evolve into a steady state characterized by the power-law correlation of the frequency-size distribution of fire clusters. In 1995, *Albano* proposed a new FFM, in which tree immunity to fire has been considered. In 1997, *Karafyllidis*[6] developed FFM to show how a forest fire spread in a inhomogeneous forest, in different weather conditions and in different landscape. In 2000, *Schenk*[9] studied finite size effect in FFM. As the system size becomes smaller, the system contains fewer clusters and tends to be homogeneous, with large density fluctuations in time.

3.3 Motivations

It is important to study FFM. It provides a perspective to predict the behavior of a dynamic system which refers as SOC. As mentioned above, SOC can be seen in various systems. SOC behavior of natural or man-made hazards give us abundance of information on risk assessment and hazard protection.

4. Description of the Model

4.1 Cellular Automata & Basic Forest Fore Model

FFM is based on Cellular automata (CA). CA is a model of physical systems where space and time are discrete and interactions are only local. Each cell is restricted to its local neighborhood interaction only. As a result, it is incapable of immediate global response. The state of each cell is updated simultaneously at each time step.

For a basic FFM, a forest is represented by a *n* by *n* grid. States of cells are defined as follows:

(i) A empty site has a value of *0*. On images generated by MATLAB code, a empty site is colored grey.

(ii) A tree has a value of *1*. On images generated by MATLAB code, a tree is colored green.

(iii) A burning tree has a value of *2*. On images generated by MATLAB code, a burning tree is colored

red.

Initially, the density of tree in the forest is given by a probability *d* (variable name in MATLAB code : *'probTree')*. The whole evolution process holds for *N* steps (variable name in MATLAB code : *'N'*). In the model, we study a finite-size forest instead of a forest following the periodical boundary condition.

The evolution of states of cells follow rules as follows:

(1) A tree can be ignited by lightening with a probability *f* (variable name in MATLAB code : *'probLightening'*).

(2) A burnt tree will become an empty site.

(3) A tree can grow in an empty site with a probability *p* (variable name in MATLAB code : *'probGrow'*).

(4) A burnt tree will cause fire in its neighboring tree with same probability *g* (variable name in MATLAB code : *'probIgnite'*). 'neighboring' means the eight cells besides the burnt tree (Fig 1).

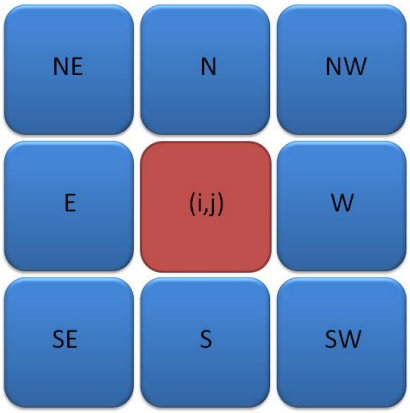

Fig 1 The definition of neighboring cells. The letters in blue cells donate directions.

4.2 Effect of Weather Conditions

Weather conditions is taken into consideration to better simulate the real case. As a preliminary trial, we consider the influence of wind and rain.

Many empirical formulas have been suggested in literatures to model the effect of the wind on the rate of the fire spreading[7]. A commonly used one is:

$$P_w = P_{0w} \exp(-\beta\theta_f)$$

where P_w is the spread rate in the presence of wind, P_{0w} is the spread rate in absence of wind (equals to *g*), θ_f is the flame angle measured from the direction of the fire spread to the direction of wind, β is the fire suppression constant (variable name in MATLAB code: *'beta'*).

In the spirit of CA that all the interactions are local, instead of with a same probability *g* , a burnt tree will cause fire in its neighboring tree with different probabilities based on the formula above The distribution of ignite probability of cell (i, j) can be seen in Fig 2.

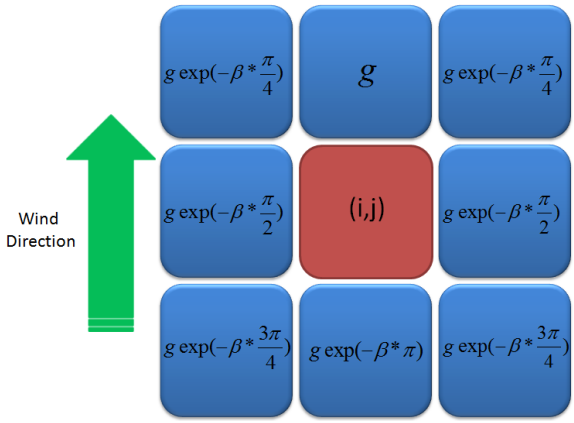

Fig 2 The ignition probability in neighboring cells.

The effect of rain is simple to describe comparing to the effect of wind. The rain can put off a fire with a probability *r* (variable name in MATLAB code: *'probRain'*). So the effect of wind and rain on ignition probability can be described by multiplying a additional factor (1-*r*) .

5. Implementation

In this part, we will first describe the essential part of our FFM to demonstrate the evolution of forest state. Our algorithm makes full use of the advantage of MATLAB: matrix manipulation. In addition, we will describe how to count the size of fire clusters and get the size-frequency distribution. The idea of recursion is used in this algorithm.

5.1 The Evolution of Forest State

In chapter 4, we already define empty site as *0*, tree as *1*, burning tree as *2*. '*grid*' , a (n+2) by (n+2) matrix, is the forest with its boundary. *'n'* is the size of the forest.

First, we initialize the states of the grid. We save all the positions of the three different cell states in three matrix called '*space*', '*havetree*' and '*burnt*':

```
space=grid ==0;                                    (1)
havetree=grid == 1;                                (2)
burnt=grid ==2;                                    (3)
```

Second, we need to know whether the surrounding of a cell are burning. This is achieved by the *'circshift'* function. *'nfire'* saves all the positions which their north cells are burning, and so on:

```
nfire = circshift(grid, [1 0])  == 2;              (4)
sfire = circshift(grid, [-1 0]) == 2;              (5)
efire = circshift(grid, [0 -1]) == 2;              (6)
wfire = circshift(grid, [0  1]) == 2;              (7)
nefire = circshift(grid, [1 -1])  == 2;            (8)
sefire = circshift(grid, [-1 -1]) == 2;            (9)
```

```
swfire = circshift(grid, [-1 1]) == 2;                    (10)
nwfire = circshift(grid, [1  1]) == 2;                    (11)
```

Next, the following code determines whether a cell can catch fire or not. This is done by generating a random number. If it is smaller than a certain probability, it will be saved in the *'susceptible'* matrix which means it will be burnt in the next step. The suffix-letter such as *'n'* means the north direction because different directions have different probabilities to get ignited. The probability contains three components: the immunity of tree from fire (*'probIgnite'*), the effect of wind ('*exp(x)',* see 4.2), the effect of rain (1-probRain):

```
susceptible_n=rand(n+2)<probIgnite*exp(-0*beta)*(1-probRain);          (12)
susceptible_ne=rand(n+2)<probIgnite*exp(-pi/4*beta)*(1-probRain);      (13)
susceptible_nw=rand(n+2)<probIgnite*exp(-pi/4*beta)*(1-probRain);      (14)
susceptible_e=rand(n+2)<probIgnite*exp(-pi/2*beta)*(1-probRain);       (15)
susceptible_w=rand(n+2)<probIgnite*exp(-pi/2*beta)*(1-probRain);       (16)
susceptible_se=rand(n+2)<probIgnite*exp(-3*pi/4*beta)*(1-probRain);    (17)
susceptible_sw=rand(n+2)<probIgnite*exp(-3*pi/4*beta)*(1-probRain);    (18)
susceptible_s=rand(n+2)< probIgnite*exp(-pi*beta*(1-probRain));        (19)
```

Finally, the *boolean calculations* in the brackets below show the evolution of the grid in the next step. For example, *the boolean calculation* of *(20)* means a cell which will be ignited in the next step must satisfy three conditions: (i) there is a tree; (ii) one of its neighboring cells is burning at least; (iii) its susceptibility is smaller than a certain probability. *(21)* describes the effect of lightening. *(22)* describes that burnt tree will become a empty site. *(23)* describes the growth of trees.

```
grid(havetree & (nfire & susceptible_n | sfire & susceptible_s | wfire &
susceptible_w | efire & susceptible_e | nefire & susceptible_ne | nwfire
& susceptible_nw | sefire & susceptible_se | swfire & susceptible_sw)) =
2;                                                                    (20)
grid(havetree & (rand(n+2) < probLightning))=2;                       (21)
grid= grid-2*burnt;                                                   (22)
grid(space & (rand(n+2)<probGrow))=1;                                 (23)
```

With the above code, we can realize the FFM (see Fig 7(a)-(f)). All the codes is in the file ***'FFM.m'***.

5.2 Fire Clusters Statistics

We assume a *sparse matrix* whose non-zero elements are *'1'*. If many non-zero elements connect with each other, they will form a cluster. Our definition of 'connect' is same as the definition of 'neighboring cells' (see 4.1). In this part, we will discuss how to do statistics on a sparse matrix and get the size-frequency distribution of clusters.

We build a list which saves all the indices of non-zero elements. If we find a *'1'*, the indices will be deleted from the list in case this '1' will be counted in the future. The following function is in the file ***'checklist.m'*** :

```
function marker=checklist(i,j)
global r;
global c;
marker=0;
```

```
m=1;
while marker==0 & m<=length(r)
    if i==r(m) & j==c(m)
        marker=1;
        r(m)=0;
        c(m)=0;
    else marker=0;
    end
    m=m+1;
end
end
```

The global variables *'r'* and *'c'* are the two vectors which save the raw indices and the column indices respectively. If the element (i, j) is *'1'*, it will be deleted from the list and the function will return a *'marker'* which equals to *1*.

The count process is done by a function called *'countsize'* (saved in file ***'countsize.m'***).

```
function countsize(M,i,j)
global r;
global c;
marker=checklist(i,j);
if marker==1
  if M(i,j)==1
      countsize(M,i,j+1);
      countsize(M,i,j-1);
      countsize(M,i+1,j);
      countsize(M,i-1,j);
      countsize(M,i+1,j+1);
      countsize(M,i+1,j-1);
      countsize(M,i-1,j+1);
      countsize(M,i-1,j-1);
  end
end
end
```

'M' is the sparse matrix. If it is the first time for element (i, j) to be counted (*'marker==1'*) and its value is *'1'*, the function will continue to search in its neighboring cells.

With the two functions above, it is available for us to do the statistics. The code is following (saved in file ***'sizedistribution_calculater.m'***):

```
global r;
global c;
set(0,'RecursionLimit',2000);
r=zeros(500^2,1);
c=zeros(500^2,1);
[r,c,v]=find(M);
SIZE=zeros(length(M));
Length_max=length(r);
```

```
for i=1:length(M)
    for j=1:length(M)
        countsize(M,i,j);
        SIZE(i,j)=Length_max-nnz(r);
        Length_max=nnz(r);
    end
end
[r1,c1,v1]=find(SIZE);
```

By using the function *'find'*, it is convenient to build the lists *'r'* and *'c'*. The *'for'* loops start to travel all the elements in *'M'*. For the element (i, j), the *'for'* loops will travel all the *'1'* connecting to it. They will become *0* in the list. By calculating the number of *0*, we can know how many *'1'* connect to (i, j). The size of the cluster (i, j) is saved in the matrix *'SIZE'*. Finally, the vector *'v1'* collects all the sizes of clusters.

6. Simulation Results & Discussion

6.1 The Regime of Self-Organized Criticality

The controlling of the rate between *p* ('*probGrow*') and *f* ('*probLightening*') which gives the average number of trees planted between two lightening strikes allows us to drive the system into SOC state. SOC state will appear when $p>>f$[8].

We keep $p/f=10^3$, *g* ('*probIgnite*')=1, *d* ('*probTree*')=0.5 and change the growth probability of tree *p.* Fig 3(a)-(c) show the change of cell states with $p=0.05$, $p=10^{-2}$ and $p=10^{-3}$ respectively. All the data is taken from a forest with size $n=500$ without considering wind and rain.

In Fig 3(a), we cannot observe the oscillation when *p* is large. In Fig 3(b), we see a fast damped oscillation which means it is not a long-time correlation. However, In Fig 3(c), we don't observe a clear damping in this time scale and the curves tend to show periodic fluctuation which is the result of the competition between the fire and the growth of trees. These two features indicate that the system is in SOC state.

The red line in Fig 3(a) shows how the number of fire changes in the forest with time. This indicates the large fires tend to happen within a relatively short time. The behavior of the red curve is an important feature of SOC. We will use it in the following analysis.

To understand the relation between SOC and the growth probability of tree, we fix the value $p/f=10^3$, $g=1$ and scan the number of fire (red curves in Fig 3(a)-(c))with different *p* from 10^{-3} to 0.6. The result is shown in Fig 3(d). To make it clear, we take the logarithm of the number of fire which can be seen in the color bar. All the data is taken from a forest with size $n=300$ without considering wind and rain.

When *p* is small, we can see the alternate conversion between blue color and yellow color which means the fluctuation. The yellow regions tend to become broader with the increasing step. This indicates the decay of the peaks in the red lines in Fig 3(a)-(c).

In Fig 3(e), we keep the $p=10^{-3}$, tune *f* from 10^{-5} to 10^{-7} ($p/f=10^2$~10^4) and sweep the number of fire. We can see if $p/f>10^3$ or $f<10^{-6}$ the peaks shows a long temporal fluctuation and the system is in SOC

state as suggested in the reference.

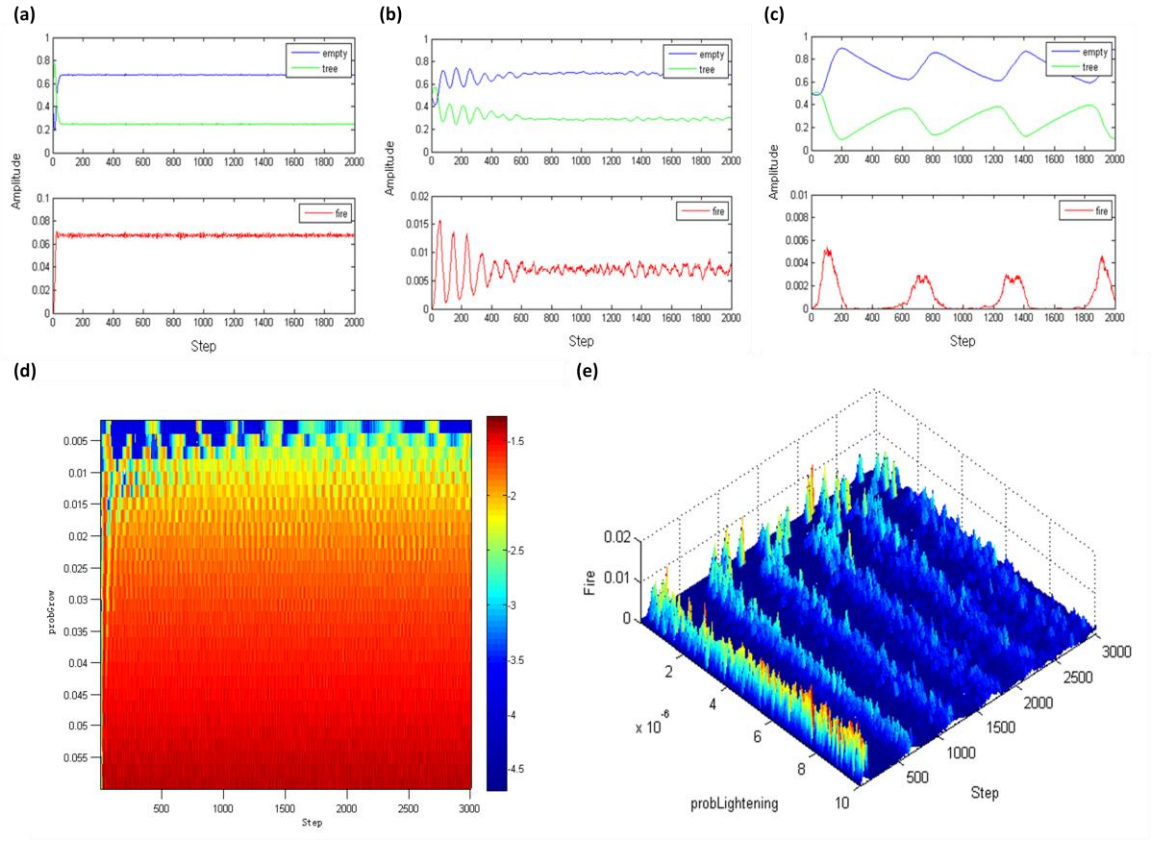

Fig 3 (a)-(c):Evolution of cell states with different p; (d): Scan of the number of fire with varied p; (e) Scan of the number of fire with varied f.

6.2 Effect of the Forest Size

The effect of the forest size on SOC state is also interesting for us. Fig 4(a)-(c) show the change of cell states with n=100, n=200 and n=300 respectively. In this simulation, we keep p=10^{-3}, f=10^{-6}, g=1, d=0.5 and don't consider wind and rain. The peaks in red curves show a broadening with the increasing size. That is due to the finite-size effect. The number of fire is easier to saturated with a smaller size.

In the Fig 4(d), the three green arrows point out the cases in Fig 4(a)-(c). When the size of forest is larger than 300, the lines become more or less straight which indicates SOC states become stable and the time of large fire doesn't vary so much with different size. While if the size of forest is small, the fluctuation caused by the finite-size effect become remarkable[9]. The broadening effect can also be seen from Fig 4(d).

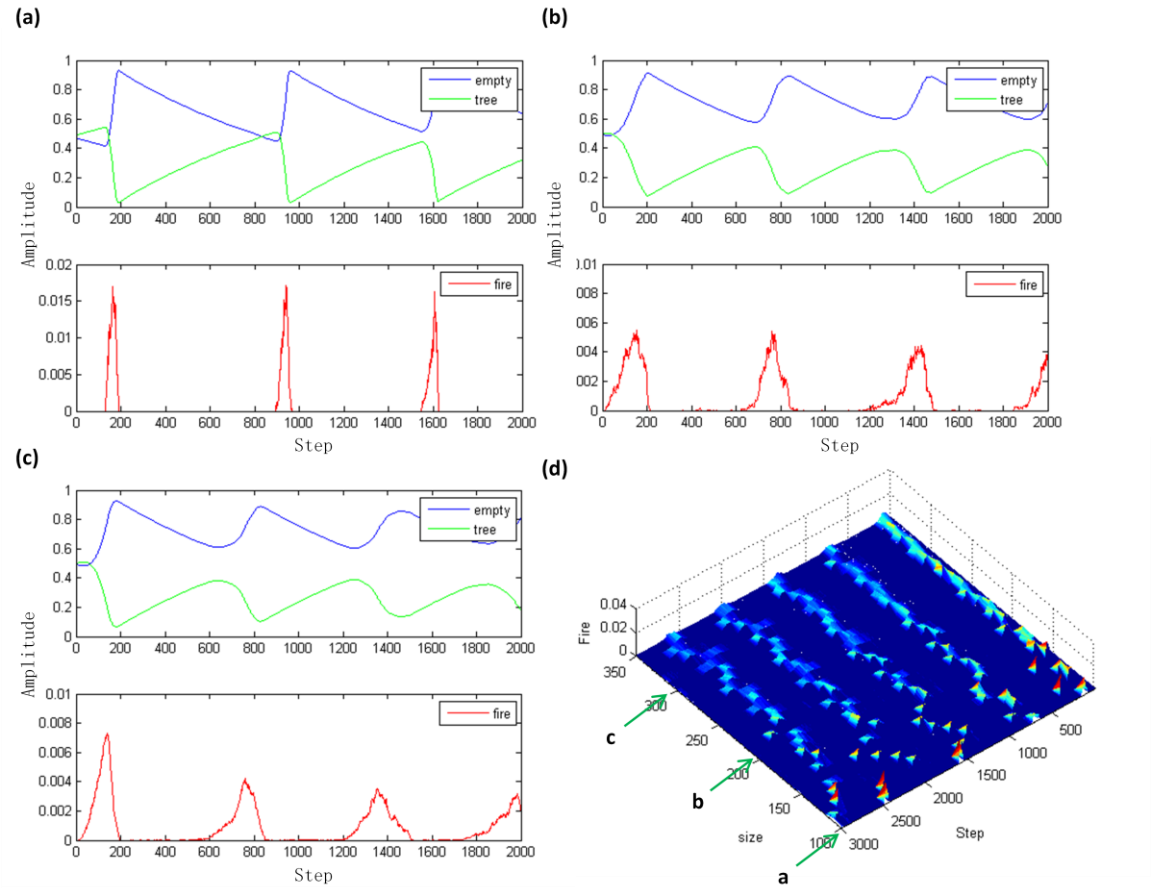

Fig 4 (a)-(c): Evolution of cell states with different n; (d): Scan of the number of fire with varied n.

6.3 Scan Other Parameters

To understand how other parameters work in FFM, we would like to do more scans. In the following scans, we keep p=10^{-3}, f=10^{-6}, d=0.5 and n=300. So the system is in SOC state.

Fig 5(a) shows the scan for the ignition probability without considering the weather effect. A common sense that higher ignition probability induces more frequent occurrence of fire and more serious fire can be proved by this simulation.

Fig 5(b) shows the effect of wind. In our model, the role of wind are in two aspects: one is to break the isotropy in the system; the other is to suppress the fire (exponential decay). The first aspect is not presented in this simulation. But the other role can be observed.

Fig 5(c) shows the effect of rain. As we discussed in 4.2, the effect of rain is simply suppression of the fire. So the trend of the peaks should be same as the trend of peaks in Fig 5(a). The result in Fig 5(c) agrees with what we would expect from code in 5.1.

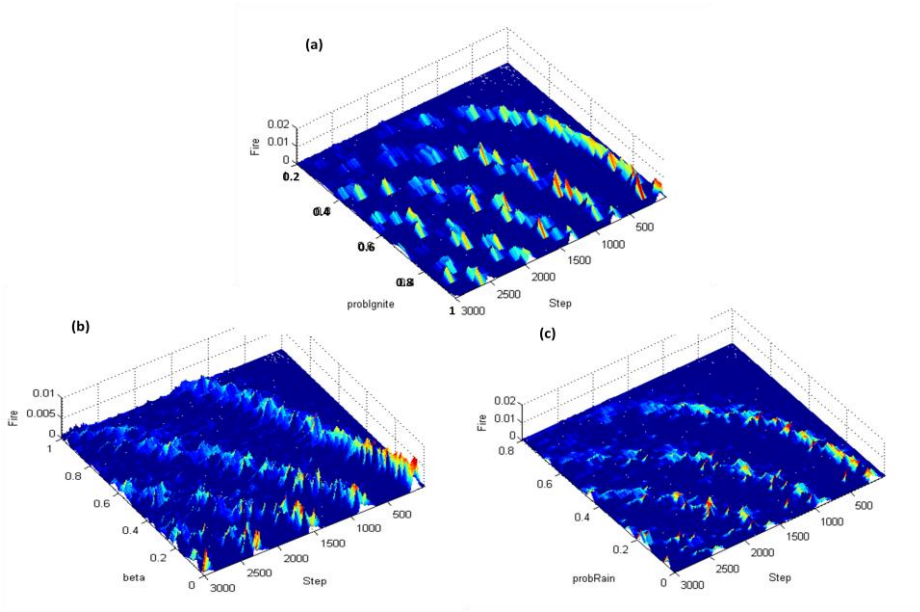

Fig 5 (a): Scan of the number of fire with varied g; (b): Scan of the number of fire with varied beta; (c): Scan of the number of fire with varied r.

6.4 Flicker Noise

In analogy with the description in 3.1 about sand pile, we can analysis the power spectrum of the number of fire versus step (time). This can be done by the Fast Fourier Transform (FFT).

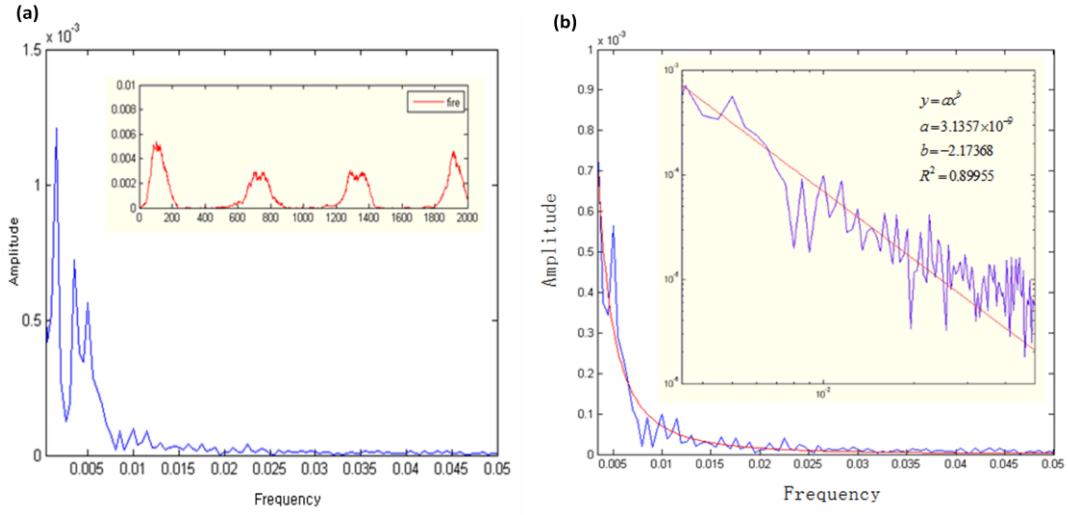

Fig 6 (a): Power spectrum of the number of fire; (b): Fitting of the power spectrum

In Fig 6(a), a high peak can be seen in the power spectrum. This peak corresponds to the frequency of the occurrence of large fire in the time domain (the insert picture in Fig 6(a)). We assume other peaks are induced by system noise. By using the formula $y=ax^b$, we can fit the noise power spectrum. The fitting results are shown in Fig 6(b). The insert picture is the fitting in *log-log* coordinates. The fitting is done by *OriginPro 8.5*. The coefficient of determination, denoted R^2, indicates how well data points fit a curve. In our case, R^2 is about 0.9 which means the fitting is quite good. All the

parameters in this analysis are same with the data of Fig 3(c). The power spectrum scales as f^{-2} at low frequencies.

6.5 Size-Frequency Distribution

Power law in size-frequency distribution is regarded as a fingerprint of SOC state. The values of parameters in the following simulation are n=500, $p=10^{-3}$, $f=10^{-6}$, g=1, d=0.5, β=0, r=0, *N=1000*. To get the size-frequency distribution in a periodic dynamic system is quite tricky.

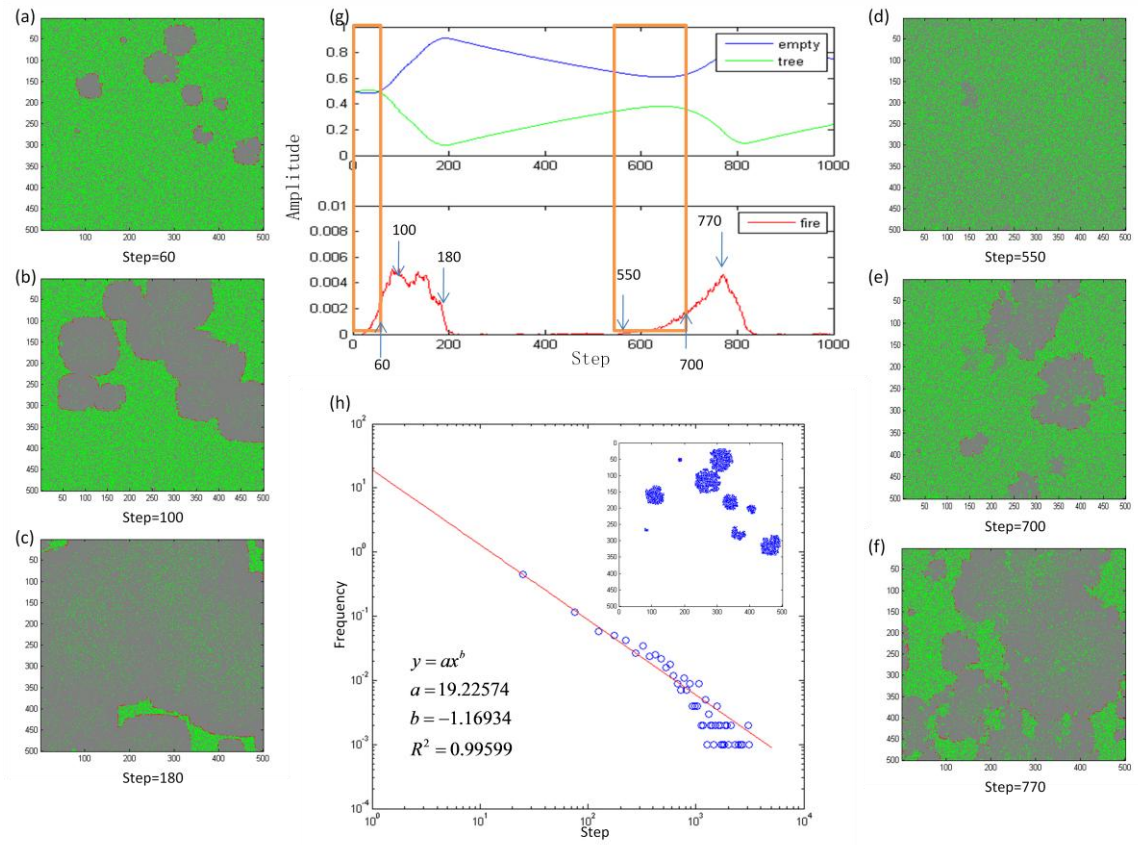

Fig 7 (a)-(f): forest states in different steps; (g): The definition of 'window'; (h): Size-frequency distribution of fire clusters and its fitting curve.

Fig 7(a)-(f) are the forest states in step 60, 100, 180, 550, 700 and 770 respectively. When the forest states are outside the orange 'windows' in Fig 7(g), large number of the fire clusters start to merge together and the fire become global. In this case, the counting of clusters is meaningless.

We choose the range in the window. The left edge is at the left tail of a peak (step 1) and the right edge is at the left half height position (step 60). We notice that inside the window the density of tree doesn't vary so much (see Fig 7(g) upper image).

We use the state on the left edge as initial state and the state on the right edge as final state. In this way, the distribution of burnt site (insert picture in Fig 7(h)) in the time window can be calculate by file ***'sizedistribution_plot.m'***. Then with the code in 5.2, we can acquire the size-frequency distribution of fire clusters.

One thing need to point out is that to get sufficient numbers of data, in principle we need to run

'FFM.m' for many steps. But our computer cannot provide enough space for such numerous data. To get rid of it, we can use one of properties of SOC state. A stable SOC state shows a periodic fluctuation. We only need the first peak of fire. That is why we don't need to run the ***'FFM.m'*** for many steps (*N=100 is enough*). However we need to run it for many times.
Fig 7(h) is the size-frequency distribution and its power-law fitting. The fitting is done by *OriginPro 8.5*. The good value of R^2 indicates a good fitting.

7. Open Questions & Future Scope

In the project, we met some questions and didn't get answers. The most important one is in 6.4. Three high peaks can be observed in the power spectrum. Other peaks are basically one order of magnitude lower than these three peaks. When we changed other parameters and analyzed the power spectrum, we noticed that the other two peaks are still there. So regarding them as pure system noise is a suspicious claim. We expected the fluctuation is due to the finite-size effect. By varying the size of forest, these peaks will shift. More quantitative analysis need to be done to give a deterministic result.
What's more, the arithmetic capability of our computers also limits us to study larger forest within longer time scale.
Last but not least, the explanation of power-law and flicker noise are also interesting for us. Why so many dynamic system show those properties? To answer this question, further study is required in the future.

8. Summary & Outlook

In this project, a FFM based on CA has been built using MATLAB code. Basic characterization of this model has been done by scanning different parameters. We also present the two fingerprints of the SOC state: flicker noise in power spectrum and power-law correlation of size-frequency distribution.
To make the model similar to real case, more factors including landscape, species of trees, etc. need to be considered. With a well-designed model, risk assessment and hazard protection will become available for people. FFM is a promising key to understand the behavior of SOC and its properties. With further study, researchers may find the physical background behind the SOC and develop a innovative way to describe dynamic systems.

References

[1] B.Drossel et al., Self-organized critical forest-fire model, Physics Review Letters 69 11 (1992)
[2] Per Bak et al., Self-organized criticality: A explanation of 1/f noise, Physical Review Letters 59 4 (1987)
[3] D.Dhar et al., Theoretical studies of self-organized criticality, Physica A 369 29-70 (2006)
[4] W.Song et al., Self-organized criticality of forest fire in China, Ecological Modelling 145 61-68 (2001)

[5] Per Bak et al., Self-organized criticality, Physical Review A 38 1 (1988)
[6] I.Karafyllidis et al., A model for predicting forest fire spreading using cellular automata, Ecological Modelling 99 87-97 (1997)
[7] A.Alexandridis et al., A cellular automata model for forest fire spread prediction, Applied Mathematics and Computation 204 191-201 (2008)
[8] P.Grassberger et al., On a self-organized critical forest-fire model, J. Phys A 26 2081-2089 (1993)
[9] K.Schenk et al., Finite-size effects in the self-organized critical forest-fire model, arXiv: cond-mat/9904356v2