1 *Basic Forest Fire Model*

We define a grid where each cell can have 3 possible states:

- 0: A green tree
- 1: Empty site
- 2: Fire

The basic rules of the dynamics of this system are:

1. If a tree is burning at a given time step (2), we make it an empty site during the next time step.
2. If a site is empty (1), a tree can be grown on it with a probability p. This means that if we found out a site to be empty, we generate a random number between 0 and 1. If the number is less than p, we change the state of the cell to 0, otherwise it stays an empty site.
3. If a site is a green tree (0), we burn this tree in the next step if:
   - Either of its neighbors is a burning tree.
   - We generate a random number between 0 and 1, if it is less than f (lightening probability), we burn this tree.

We usually have a grid in 2 dimensions, say of size N $\times$ N.The neighbor of a cell marked by index (i,j) are: (i-1,j),(i+1,j),(i,j-1) and (i,j+1). We employ the model in periodic boundary condition which is ensure that our system is not influenced by the edges. Just to give an example, 4 neighbors of the site, (1,1) are: (1,2),(2,1),(N,1) and (1,N).

2 *Useful Quantities and Controlling Parameters in the model*

Before, explaining the code let us first list some of the quantities which we want to measure through our simulations. First of all, let us discuss what are the controlling parameters of the system. From the above model, we see that we can control the rate of tree growth and lightening. But from the literature, we know that when the system approaches Self Organized Critical State, only the ratio p/f matters and not the individual values (for an SOC, p/f should be much greater than 1). This can be a good check on whether a state is SOC or not. We can also see how the system behaves for different grid size. An SOC is usually immune to the initial conditions, but its dependence on the initial conditions can also be checked.

Now, a word on the useful quantities associated with the system. Let ρ_e, ρ_t and ρ_f be the fraction of empty sites, green trees and fires respectively. We can see that how these densities vary as the system evolves. In case, the system reaches an equilibrium we expect these densities to achieve a constant value. We do not exactly know what happens if a system reaches a self organized state. Another important property of such a system is that the average number of trees burnt by a fire ($\bar{s}$)

$$\bar{s} \propto \frac{p}{f}$$

A cluster is defined as a set of all the tree which can be burnt by a fire if the only dynamics is the propagation of fire. The full forest grid contains a clusters of various sizes (where by size we mean the number of trees in a cluster). The distribution of number of clusters as a function of cluster size is a power law distribution for SOC state. Hence, it can be used to detect whether a state is am SOC or not. Another useful quantity is the radius of cluster which is the root mean squared distance of a cluster from its center of mass. As a function of cluster, it also exhibits a power law

$$s \propto R(s)^{\mu}$$

Another power law is expected in the distribution of number of clusters as a function of cluster size.

$$n(s) \propto s^{\mu}$$

3 *The code and the various algorithms used*

Let us now discuss the code and various algorithms used.

Basic_fire_model_smaller2.m: Here, we initialize a grid with 50:50 probability of having a green tree or an empty site. In order to employ periodic boundary condition, we work with a grid whose size is 2 more than the expected size in each of the dimensions, so as to take care of the neighbors in opposite edges. We apply the dynamics of the system by the rules defined above. We keep track of the fraction of trees, fires and empty sites. We can also use this code to find $\bar{s}$. In order to do this, we keep track of the number of lightening strokes and the total number of trees burnt after some initial transient period. We repeat the code of different $\frac{f}{p}$ ratio and then can plot the result.

cluster_disrtibution2.m: This is the main function which analyzes the various aspects of the grid. First of all, it marks all the clusters with a unique index which is not a trivial task owing to the periodic boundary conditions. First of all, we give an index to every green tree: by either looking at all its neighbors and finding the minimum index or giving a new index. We also update all the neighbors which are green trees with this index. Because of periodic boundary conditions, it is possible that 1 cluster has more than 1 indices (for

different trees).In order to get rid of this redundancy, we again go through all the cells in the grid and if we find any cell with a neighbor which has a different index, we replace the 2 indices by the smallest of the 2. This step is followed by scanning through the whole grid to replaces all the cells with larger index with the smaller index (*Just to mention, it is the most time consuming step in our algorithm. It will be really great if you can suggest a faster way of uniquely indexing all the clusters*). After obtaining the correct indexing, we use this indexing to count the size of all the clusters. In order to do this, we just count the number of trees marked by a given index. In the process of uniquely indexing all the clusters, it is possible that there are indices which are not at all used in the final indexed grid. As a result, these indices will store 0 as the cluster size. We should get rid of these. The last steps is to calculate the cluster radius which we do by navigating through the full grid and calculating first and second order moments for all the clusters.

neighbor.m: This is used by cluster_disrtibution2.m and it returns the indices of all the neighbors of a cell in the periodic boundary condition.

converter.m: This is used by cluster_disrtibution2.m and it returns infinity if a tree is unindexed, otherwise it returns the index. Note that in our code indexing starts with 3.

changer.m: This is used by cluster_disrtibution2.m and it replaces all the neighbors of a cell with the minimum index among these cells.

enforcer.m: This is used by changer.m and it replaces all the trees with a given index with another smaller index passed as one of its arguments.Finally, it returns the full grid.

radius_size_distribution.m: This is used by cluster_disrtibution2.m and it receives 3 vectors: 1 of them gives the cluster size of all the clusters, the other vector contains the corresponding cluster radius. It calculates the number of clusters and cluster radius as a function of cluster size. *Please note that for a cluster size, if we have more than 1 clusters, we take the average cluster radius of all these clusters to be as the cluster radius corresponding to the given cluster size.*

4 *Results*

Let us describe the various results which we have found in this system.
We have observed **2 different regimes** for high $\frac{p}{f}$ ratios. In one of the regime (Folder: **Equilibrium_situation**)which occurs when p is low, but not very small (greater than roughly 0.01) we observe a state which is like equilibrium. In such a phase, we observe that the fraction of trees, fires and empty sites achieve a

constant value after an initial transient period. The properties of this state are very similar to an equilibrium state, but we suspect that this might be the state of Self Organized Criticality.
In the following figure, we show the time variation of the various fractions (Folder: **Equilibrium_situation\pbf**).

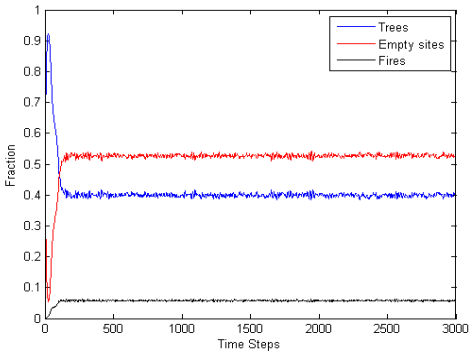

Figure 1: Time variation of the various fractions,$\{N = 256, \frac{p}{f} = 10^4, p = 0.11\}$

One of the defining features of the SOC is its dependence only on the ratio of p and f independent of the values of p and f individually, We tried to test this feature in this regime by calculating the equilibrium fractions for various values of p, but fixed ratio of p and f. An important realization in our simulations is that by changing p, we are able to go from 1 regime to another. The following are the results depicting the variation of fractions as a function of p. (Folder: **Equilibrium_situation\pbf**).

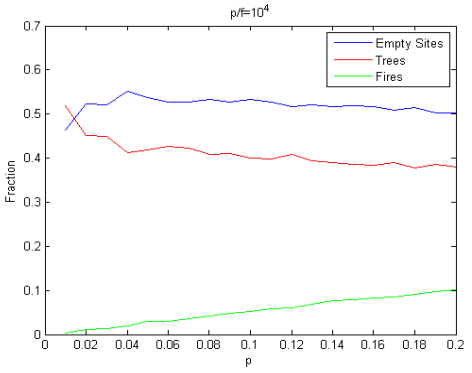

Figure 2: variation of the various fractions for different values of p,$\{N = 256, \frac{p}{f} = 10^4$, Iterations= 1000$\}$

As we can clearly see from the figure, the various fractions change as we change p, but keep $\frac{p}{f}$ constant, which is unexpected from an SOC kind of phase.

We also wanted to check whether our final fractions depends on the initial conditions of the grid. Just to remind, *our initial condition in all the above cases is a grid having trees and empty sites spread randomly with half-half probability.* We changed this probability, We varied the probability of growing a tree from 0.1 to 0.9 in the initial grid and checked for the final fractions. As we can see from the figure below, the fractions are more or less constant(Folder: **Equilibrium_situation\initial_condition_independence**).

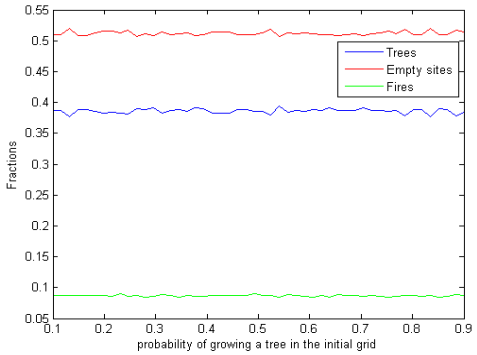

Figure 3: variation of the various fractions for different initial conditions,$\{N = 256, \frac{p}{f} = 10^4, p = 0.17$, Iterations$= 1000\}$

Another feature which we tried to identify how the various fractions change as we change the ratio of p and f, but keep p constant. To our surprise, it came to be nearly constant In the following plot, we show the variation of various fractions as a function of ratio of p and f(Folder: **Equilibrium_situation\density variation as p by f ratio**). .

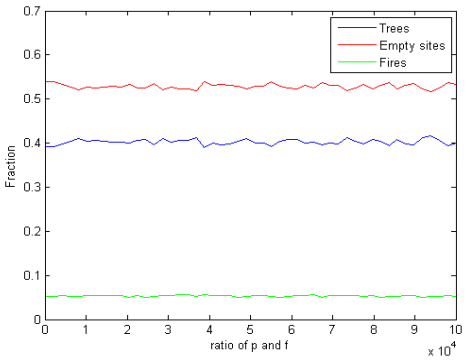

Figure 4: variation of the various fractions for different ratios of p and f,{$N = 256$,$p = 0.1$, Iterations= 1000}

This is unexpected for an SOC state where the $\frac{p}{f}$ is the controlling parameter ;and values of p and f only determine the time scale of the system.

Next we plot the average number of trees burnt by a lightening stroke as a function ratio of p and f. (Folder: **Equilibrium_situation\Avg_no_of_trees_burnt_by_a_light_stroke**).

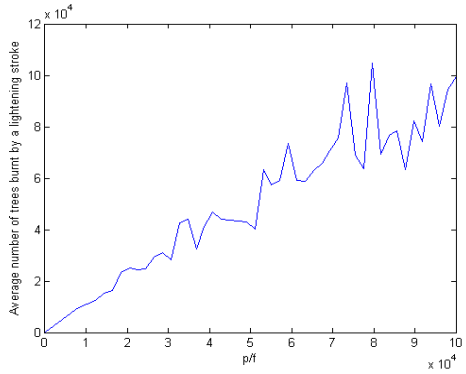

Figure 5: variation of the number of trees burnt per fire stroke for different ratios of p and f,{$N = 256$,$p = 0.1$, Iterations= 1000}

We can see that if we fit a straight line on this plot, it will satisfy the condition set above for SOC. The fluctuations in the curve may be removed by running the simulations for longer time steps and more number of points in the curve. [*We are working on this part*]

We also tried to find the power laws as described above. The following are the plots for cluster number distribution vs cluster size and cluster radius as a function of cluster size.(Folder: **Equilibrium_situation\Cluster_radius_and_cluster_size**).

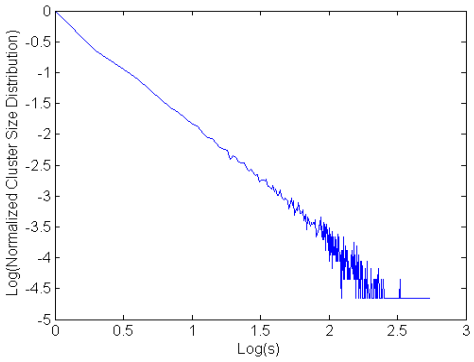

Figure 6: Cluster number distribution vs cluster size,{$N = 1024, \frac{p}{f} = 10^4, p = 0.17$, Iterations= 1000,Number of points in the plot= 246}

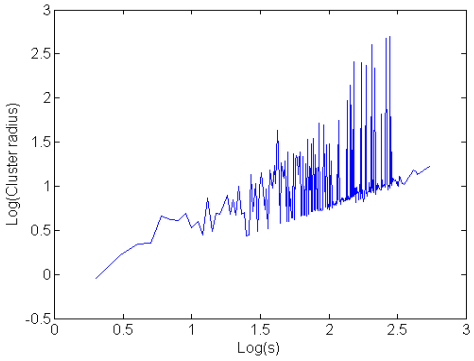

Figure 7: cluster radius as a function of cluster size,{$N = 1024, \frac{p}{f} = 10^4, p = 0.17$, Iterations= 1000,Number of points in the plot= 246}

The plots show a lot of fluctuations which is not a nice feature, but overall the trend is as expected for Self Organized Critical behavior. The fluctuations is most probably because of the finite size effect of the system or lesser number of points taken and if we can improve the algorithm to properly index the clusters.

The **second regime** appears when we reduce the tree growth rate to very small values(Folder: **Oscillating_situation**). Since the $\frac{p}{f}$ is very large,the probability of igniting a tree will be very low. As a result, the forest goes on to become more and more dense, and hence very large clusters are formed in the forest. At some point of time, a single fire can burn the full forest. So, we see a somewhat periodic behavior in the fraction of trees, empty sites and fires. From the literature, we know that the dynamics of the system depends only on the $\frac{p}{f}$ and not on the individual values of p and f. Values of p and f just decide

the time scale of the system. So, it is possible that instead of getting periodic variation, we may get some final equilibrium values if we run the code for very long time. We have tried to find this, but still we have no evidence for decay in the oscillations. In the following 2 plots, we show the fractions of trees, fires and empty sites for 2 different sizes of iterations: one for 10000 iterations and another one for 80000 iterations.(Folder: **Oscillating_situation\pbf**).

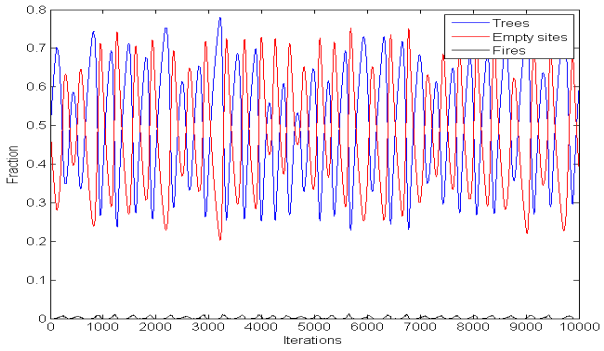

Figure 8: Fraction of trees, fires and empty sites,$\{N = 1024, \frac{p}{f} = 10^4, p = 0.005$, Iterations$= 10000\}$

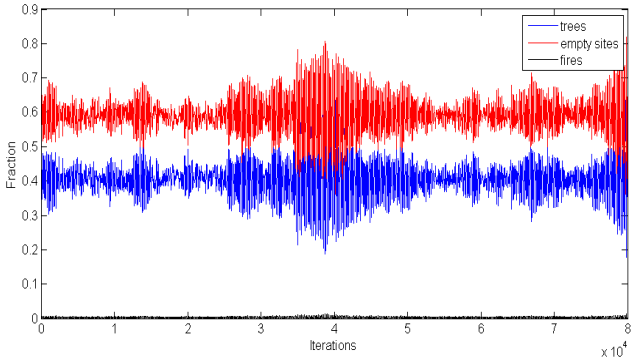

Figure 9: Fraction of trees, fires and empty sites for larger number of iterations,$\{N = 256, \frac{p}{f} = 10^3, p = 0.005$, Iterations$= 80000\}$

One more calculation which we did is the number of trees burnt per lightening for different $\frac{p}{f}$ ratios(Folder: **Oscillating_situation\Avg_no_of_trees_burnt_by_a_light_stroke**). As we can see that there is a lot of noise in the data, but also does not follow the overall trend as expected ($\bar{s} \propto \frac{p}{f}$).

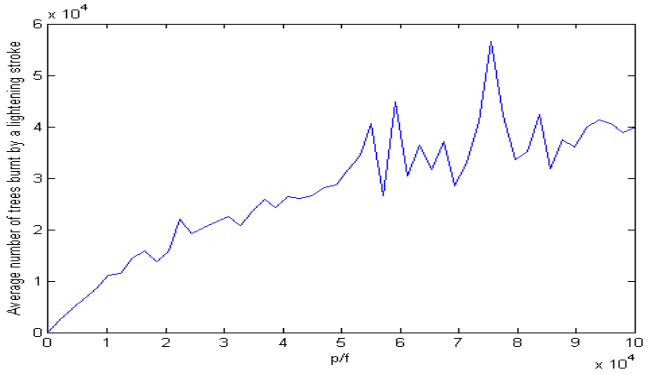

Figure 10: variation of the number of trees burnt per fire stroke for different ratios of p and f,$\{N = 256, p = 0.005, Iterations = 10000\}$

We don't see any point in calculating power laws for this system since the various fractions are changing with time and not constant.

Please have a look at the videos we made for each of the 2 regimes.

5 *Questions and Doubts*

We would like to have your suggestions on whether we are going in the right way or are we looking in the wrong parameter regimes? It would be really helpful if you can suggest us some of the things which we can do with the current model. In the reference 1, as mentioned below is the one which we have used to understand about the different parameters and the model [1]. There are certain other things which we can calculate with the current model like correlation length but right now, we don't have any clue regarding how to implement this. It would be great if you can suggest some way to improve the efficiency of indexing the clusters uniquely.

Another problem which we face is: In order to see SOC, we need to verify that the time needed to burn the biggest cluster in the forest should be much less than the time needed to grow trees (i.e. p^{-1}). This condition is difficult to verify owing to the way we model our system. One way which we have used is to implement a slightly different version of this model where we set the time needed to burn a cluster in 0 time [2]. We will let you know the details and results regarding this model shortly.

References

[1] Siegfried Clar, Barbara Drossel, and Franz Schwabl. Forest fires and other examples of self-organized criticality. *Journal of Physics: Condensed Matter*, 8(37):6803, 1996.

[2] Gunnar Pruessner and Henrik Jeldtoft Jensen. Efficient algorithm for the forest fire model. *Phys. Rev. E*, 70:066707, Dec 2004.