Fractal Report

Professor: Yimin Xiao
Graduate Assistant: Micheal Delaura
Working Group: Gabriel Raodoccia-Feuerstein
Brantley Simmons
Jihad Nasser
Konstantin Mckenna

July 15, 2016

1 Background

We start with the background of the mathematics of fractals, beginning with the integral property of the majority of fractals currently in existence. Self-similarity. Fractals, such as the Sierpinski Triangle, Sierpinski Carpet, and Cantor set are scaled down by a mapping ratio and subsequently appended together in the shape of the original. This leads to a gap between scaled copies. This process is iterated on every single piece of the fractal until the resulting area or length is, in the traditional two-dimensional sense, zero[1]. Though an infinite level of iterations generates no area, it is possible to imagine that every scaled piece looks exactly like the piece it was generated from, hence self-similarity. We then learned about the Hausdorff dimension. In order to measure fractals, whose dimension is not typically an integer, we look at the fractals formulation. The common definition of dimension refers to the way in which we scale images and the way that their scaling affects the area they represent. In this sense, scaling an image to create smaller copies of itself leads to a logarithmic problem that results in a non-integer Hausdorff dimension[2]. We learned to describe the mappings in a system of trees, like those commonly seen in a family tree. Each mapping breaks off into a family of mappings, so on and so forth until the tree of mappings is so expansive that their fractal representations area is effectively zero. We then move into the idea of randomness. We start by learning about Galton-Watson trees in which each member of a tree has a chance to spawn a number of children between zero and some limiting integer. Every time a vertex produces vertices, there is a weighted probability associated with them and a function that generates said probability[3]. We apply this concept to the generation of random fractals, leading to fractals that also produce no area, but have a different Hausdorff dimension due to the added effect of randomness. We are able to compare the densities of these random fractals based on a given logarithmic function that compares the probability of completing a mapping to the mapping ratio used. We then looked over Falconer's theorem, which allowed us to predict whether or not certain fractals would go extinct based on the number of children produced and the probabilities of certain numbers of children spawning[3]. Learning about these properties of random and deterministic fractals allowed us to begin simulating the very concepts that we learned about. Out of the simulations that we were able to create, the main focus for the group were the simulations about Falconer's Theorem applied to a randomly subdividing box fractal and the damage/repair cycle used to create a self-healing random box fractal[4].

2 Matlab Simulations & Expirements

2.1 Graphing Sierpinski Fractals

As we stated before, we began with the exploration of the fractals such as the One-Third Cantor Set, the Sierpinski Triangle, and the Sierpinski Carpet. We wanted to get a better idea of these fractals, and how exactly they are formed through simulations. We

had trouble simulating the Cantor Set since it isn't just one shape that is altered within itself, but a new group of lines for each generation. Luckily, we were easily able to find the Matlab codes for a nice simulation of the Sierpinski Triangle using Google. The code used the plotting of dots and the iterations of the Sierpinski Triangle to form the fractal as the code is ran to more iterations. The code for the Sierpinski Triangle are as follows[5]:

```
clf
hold on
N=100000;
x=zeros(1,N);y=x;
for a=2:N
c=randi([0 2]);
switch c
    case 0
        x(a)=0.5*x(a-1);
        y(a)=0.5*y(a-1);
    case 1
        x(a)=0.5*x(a-1)+.25;
        y(a)=0.5*y(a-1)+sqrt(3)/4;
    case 2
        x(a)=0.5*x(a-1)+.5;
        y(a)=0.5*y(a-1);
    end
end
plot(x,y,'.')
legend(sprintf('N=%d Iterations',N))
```

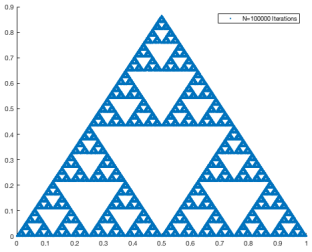

Figure 1: Sierpinski Triangle Simulation

From here, we were able to construct the simulations for the Sierpinski Carpet, but instead of using three cases, we had to use eight:

```
clf
hold on
```

```
N=1000000;
x=zeros(1,N);y=x;
for a=2:N
c=randi([0 7]);
switch c
    case 0
        x(a)=(1/3)*x(a-1);
        y(a)=(1/3)*y(a-1);
    case 1
        x(a)=(1/3)*x(a-1);
        y(a)=(1/3)*y(a-1)+(1/3);
    case 2
        x(a)=(1/3)*x(a-1);
        y(a)=(1/3)*y(a-1)+(2/3);
    case 3
        x(a)=(1/3)*x(a-1)+(1/3);
        y(a)=(1/3)*y(a-1);
    case 4
        x(a)=(1/3)*x(a-1)+(1/3);
        y(a)=(1/3)*y(a-1)+(2/3);
    case 5
        x(a)=(1/3)*x(a-1)+(2/3);
        y(a)=(1/3)*y(a-1);
    case 6
        x(a)=(1/3)*x(a-1)+(2/3);
        y(a)=(1/3)*y(a-1)+(1/3);
    case 7
        x(a)=(1/3)*x(a-1)+(2/3);
        y(a)=(1/3)*y(a-1)+(2/3);
    end
end
plot(x,y,'.')
legend(sprintf('N=%d Iterations',N))
```

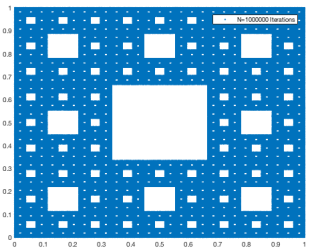

Figure 2: Sierpinski Carpet Simulation

Simulating these fractals gave us a good picture of how fractals are created, and just how we could create them moving forward. Using this idea of graphing our fractals allowed us to zoom into each fractals and truly see the multiple levels of the figures. After being able to create these figures, we switched gears as we started to look at Galton-Watson Networks, Falconer's Theorem and trees, to try and represent random fractals and trees and see the extinction and survival of these fractals through many generations[3].

2.2 Trees and Random Box Fractals with Falconer's Theorem

When we began looking at trees, we used used them to help represent our previous non-random fractals, such as the cantor set. This tree would have two children come off of the parent for every parent on every generation of the tree. That representation is exactly the same for the Sierpinski Triangle and Sierpinski Carpet. However, as we moved into Galton-Watson Networks, which use randomness, we decided to try and simulate these random trees to show exactly how Falconer's Theorem works as time goes on.

Falconer's Theorem:

$$\text{Given} \sum_{i=1}^{N} A_i \neq 1, \text{we define} \, \gamma := E\left\{ \sum_{i=1}^{N} A_i \right\}$$

1. If $\gamma \leq 1$, flow is almost surely not possible.
2. If $\gamma > 1$, flow is almost surely possible. (non-extinction).[3]

However, the trees ended up becoming so large that the our graphs were incomprehensible at the larger generations. As a replacement, we represented Falconers Theorem using a random box fractal. The code is shown below[6]:

```
A=binornd(1,p,n,n);
while length(A)<L;
A=kron(A, ones(n,n));
for i=1:length(A);
```

```
for j=1:length(A);
if A(i,j)==1;
A(i,j)=binornd(1,p);
end
end
end
end

[r,c] = size(A);
imagesc((1:c)+0.5,(1:r)+0.5,A);
colormap(gray);
axis equal
set(gca,'XTick',1:(c+1),'YTick',1:(r+1),...
        'XLim',[1 c+1],'YLim',[1 r+1],...
        'GridLineStyle','-','XGrid','on','YGrid','on');
```

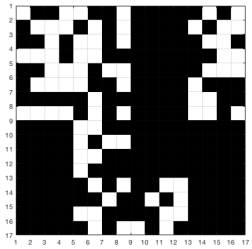

Figure 3: Random Box Fractal

Letting p equal some probability for survival, n equal to some size of the original box, and L to some length for the box to end simulations once that length is exceeded. This allowed us to see how long and how many of the boxes that we created survived using specific values of p, n, and L, while the second part of the code is what actually plots the results onto a graph. For this specific picture, we let $p = .5$, $n = 4$, and $L = 14$.

Once we were able to understand this simulation, we looked at specific values of n and p to see whether or not we could show that the result of Falconer's Theorem can easily be seen over large numbers of generations. We went through various values of n and p and checked if each box fractal would survive to one-hundred generations or not, and did so for each pair of n and p one-thousand times each. Our results are as follow: The boxes ✓symbols in them were combinations which we did not have the computing power to simulate, but through observation of the other simulations and Falconer's Theorem we assume they would not go extinct in almost all cases. These simulations gave us great evidence of the success of Falconer's Theorem.

p \ n^2	**4**	**16**	**64**
$\frac{1}{64}$	$\frac{0}{1000}(2)$	$\frac{0}{1000}(6)$	$\frac{26}{1000}$
$\frac{1}{16}$	$\frac{0}{1000}(4)$	$\frac{20}{1000}$	✓
$\frac{1}{4}$	$\frac{20}{1000}$	✓	✓

Table 1: $\frac{\text{survival to 100 generations}}{\text{trials}}$(highest generation of survival)

2.3 Damage and Repair

From here, we looked at a paper which focused on the damage and repair of the Cantor Set, and how the randomness of the repair would change the Hausdoff Dimension of that fractal in relation to the original Cantor Set[7]. As we stated before, simulating the Cantor Set was very difficult, so we took this damage and repair idea to our random box fractal. The only issue here is that for the Cantor Set, there was repair done specific to each step, as described in the figure below:

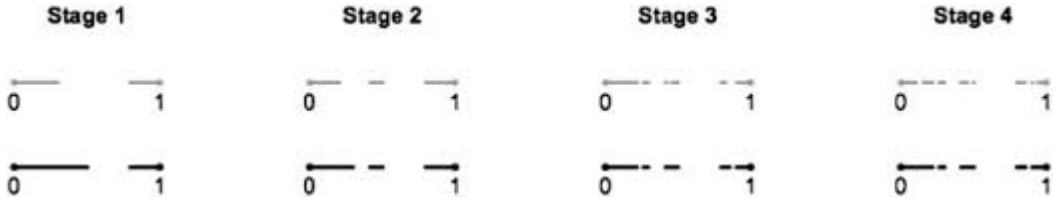

Figure 4: Cantor Set with Damage and Repair[8]

Going from Stage 1 to Stage 2 and so on includes damage and repair in each step. For the random box fractal, to keep the binomial distribution we used in the coding working, we weren't able to include repair at each step like this version of the Cantor Set. Instead, we decided that the best representation of the damage and repair would be to create an original damage box fractal using certain n and p values, and then form another box fractal using the same n and p values which would be a repair fractal. To make it a repair fractal, we really just added the two fractals together, which would keep any surviving parts of the original damage fractal the same no matter what happened in the repair fractal, while changing any dead portion of the original fractal to back to a surviving portion if that section of the repair fractal had survived. It can easily be seen by looking at the graphs of each fractal and then the combination of both:

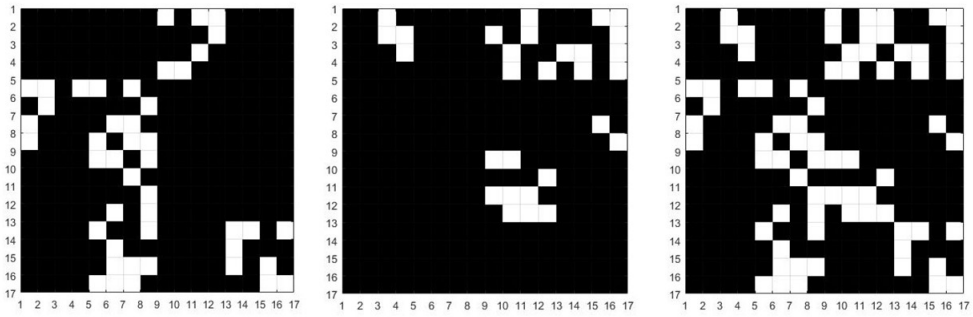

Figure 5: Damage and Repair Box Fractals Progression

The codes for these fractals are exactly the same as the when we first generated the random box fractals, except we named the second box fractal B instead of A in the codes, then created this final code to bring the two together to create the combination fractal called C:

```
C=A+B;
for i=1:length(C);
for j=1:length(C);
if C(i,j)>1;
C(i,j)=1;
end
end
end
```

This made it very easy to see just how much repair was done on the original fractal. We were hoping that this would lead to some cool results such that we would be able to tweak Falconer's Theorem for the survival of this combination fractal, but as long as all we do is add them together, then it is clear that if either the repair or damage fractal survives, then the combination will clearly survive as well. This was one of many questions that we weren't able to address completely and could be good question to strive to work on moving forward.

3 Conclusion and Future Questions

Along with all of the research we have done, there were many questions and topics that we had which we weren't able to look as deeply as we had hoped to. During our time

looking at Galton-Watson Networks we were looking for a way to connect our random trees to genealogy and a relation to the inheritance of land, but weren't able to find enough of a connection to truly get started on the topic. We also briefly talked about Brownian Motion and how it, as a random movement through some timespace, connects to the concepts we covered with fractals and Hausdorff Dimensions. We wanted to apply the concept of random stuttering cantor-like sets to Brownian motion and other fractals to see if there was any overlap or insight that could be applied to healing lesions and other processes that involve random damage and repair to sets. Overall, we learned a lot across many different topics, which has lead to many different areas of study to look further into.

4 Acknowledgements

We would like to thank the National Security Agency and the National Science Foundation for funding this program (project sponsored by the National Security Agency under Grant Number H98230-16-1-0031; project sponsored by the National Science Foundation under Grant Number DMS-1559776). We would also like to thank Michigan State University and Lyman Briggs College for hosting the REU. Finally, we would like to thank Dr.Yimin Xiao and Micheal Delaura for their assistance.

5 References

1. Peres, Yuval, Kroly Simon, and Boris Solomyak. "Self-similar sets of zero Hausdorff measure and positive packing measure." Israel Journal of Mathematics 117.1 (2000): 353-379.

2. Falconer, Kenneth J. The geometry of fractal sets. Vol. 85. Cambridge university press, 1986.

3. Falconer, Kenneth J. "Random fractals." Mathematical Proceedings of the CambridgePhilosophical Society. Vol. 100. No. 03. Cambridge University Press, 1986.

4. Pestana, Dinis D., Sandra M. Aleixo, and J. Leonel Rocha. "Hausdorff dimension of the random middle third Cantor set." Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on. IEEE, 2009.

5. Silva, Paulo. "Plotting Sierpinski's Triangle." Plotting Sierpinski's Triangle. N.p., 28 Feb. 2011. Web. 14 July 2016.

6. Gnovice. "How to Convert 2D Binary Matrix to Black & White Plot?" Matlab. N.p., 19 July 2010. Web. 14 July 2016.

7. Pestana, Dinis D., Sandra M. Aleixo, and J. Leonel Rocha. "Hausdorff dimension of the random middle third Cantor set." Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on. IEEE, 2009.

8. Ian. Family Growth. Digital image. The Galton Watson Process Part I — Core Computations. N.p., June-July 2011. Web. 20 June 2016.