CSE585/EE555: Digital Image Processing II
Computer Project # 5
Fractal Generation using Iterated Function Systems

Mudit Garg, Mayank Murali, Niranjan Thirusangu

Date: 04/23/2020

A. Objective

The objective of this project is to generate a fractal image using Iterated Function Systems (IFS). We need to understand how the number of iterations and changing probabilities impact the results.

B. Methods

In this project, we implemented the Iterated Function Systems (IFS) to generate a fractal image. At every iteration, we applied Affine Linear Transformation to the result from the last iteration. As per L23-6, affine linear transformation is given by:

$$w(x,y) = w\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}a_{00} & a_{01}\\ a_{10} & a_{11}\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}t_0\\t_1\end{bmatrix}$$

The parameters which were used for this transformation are given in results section.

Algorithm:

The algorithm which we used in this project was taken from section 5.4.4 of the book *The Science of Fractal* Images by Peitgen and Saupe.

The algorithm for $RenderIFS$ code requires the following parameters:

Parameters	Parameter description
L	Image resolution in the x-direction.
M	Image resolution in the y-direction.
P	Probability set (total sum = 1).
num	Number of iterations performed.
$xmin$	Lowest x-value of the image window.
$xmax$	Highest x -value of the image window.
$ymin$	Lowest y-value of the image window.
$ymax$	Highest y -value of the image window.

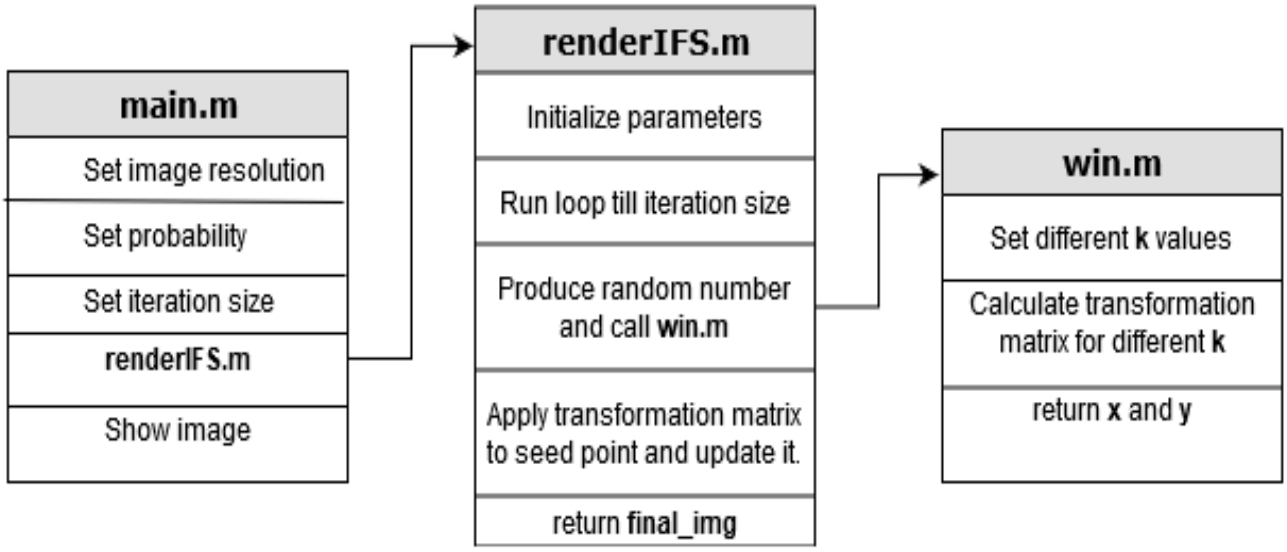
The pseudo code of the algorithm is given below:

1. Specify the desired parameter values for rendering the IFS code, including affine transformation matrices, number of iterations and probability set.
2. Initialize a 1024×1024 zero-value matrix. Here, $L = M = 1024$.

3. Initialize a starting (seed) point (x, y). We took $(x, y) = (0,0)$.
4. At each iteration:
    a. Produce a random number in the interval $[0, Arand]$.
    b. Determine which transformation matrix w_i is to be use at this iteration.
    c. Apply the chosen transformation matrix w_i to the seed point vector (x, y).
    d. Check if the x and y values are within the max and min range.
    e. Calculate which pixel corresponds to this iteration.
    f. Set pixel (x, y) on the image to be 1.
    g. The updated vector (x, y) will be the seed point for the next iteration.
5. Display the image obtained after the execution of last iteration.

MATLAB code Flow & Execution:

Following is the MATLAB code flow for this project:



In order to execute this project, just run $main.m$ after initializing the parameters. The final image generated by IFS will be shown on the screen.

C. Results

We ran the algorithm mentioned in the Methods section for different probability sets and captured the images after several iterations. For the affine linear transformation, following parameter values were used. They were taken from L23-11.

W	a_{00}	a_{01}	a_{10}	a_{11}	t_0	t_1
1	0	0	0	0.16	0	0
2	0.2	-0.26	0.23	0.22	0	1.6
3	-0.15	0.28	0.26	0.24	0	0.44
4	0.85	0.04	-0.04	0.85	0	1.6

Table 1: Affine Transformation Parameters

The initial seed value for fractal generation was set to $(0,0)$. Also, $xmin = -3$, $xmax = 3$, $ymin = 0$ and $ymax = 9$ were set.

Probability Set 1 [0.2, 0.35, 0.35, 0.1]

We ran the algorithm mentioned in Methods sections with affine transformations given in Table 1 and probability set = [0.2, 0.35, 0.35, 0.1]. Algorithm was executed for 100,000,000 iterations. The image was captured at 100, 1000, 5000, 10,000, 20,000, 100,000, 500,000, 1,000,000, 10,000,000, 30,000,000, 50,000,000 and 100,000,000 iterations. The results obtained are shown below.

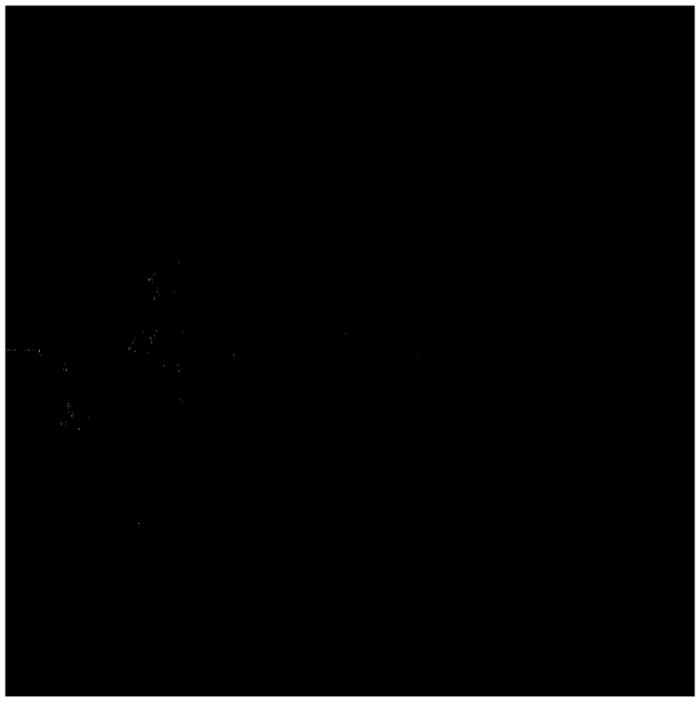


Figure 1: Fern image generated with 100 iteration using probability set 1.

Figure 2: Fern image generated with 1000 iteration using probability set 1.

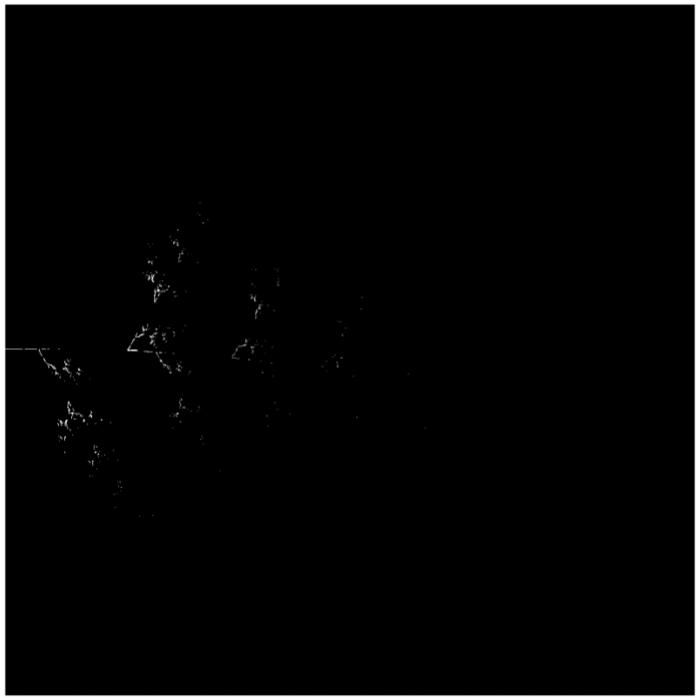


Figure 3: Fern image generated with 5000 iteration using probability set 1.

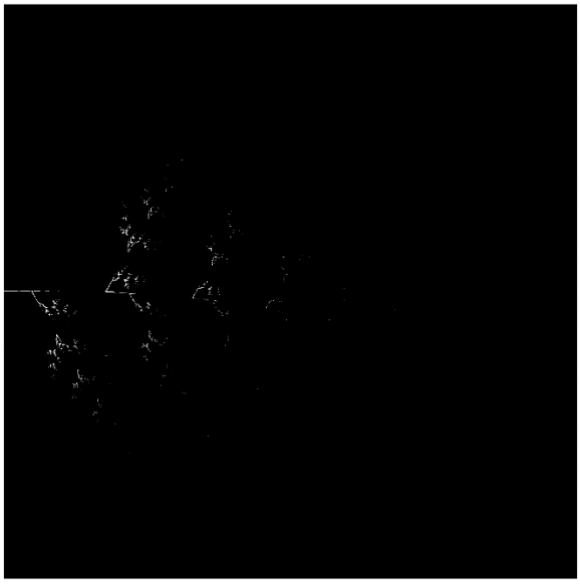


Figure 4: Fern image generated with 10000 iteration using probability set 1.

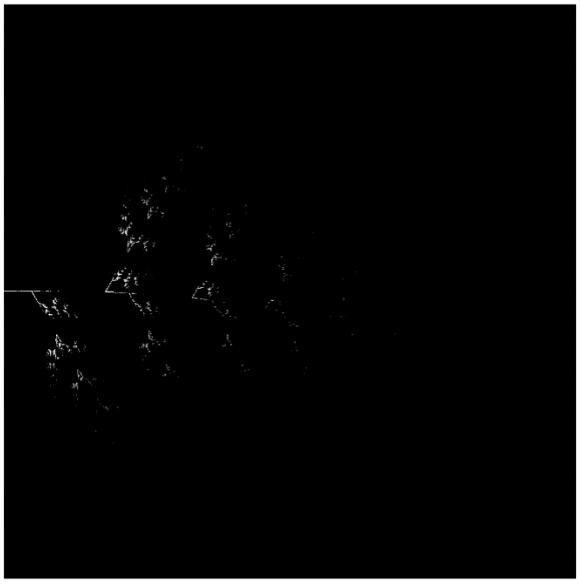


Figure 5: Fern image generated with 20000 iteration using probability set 1.

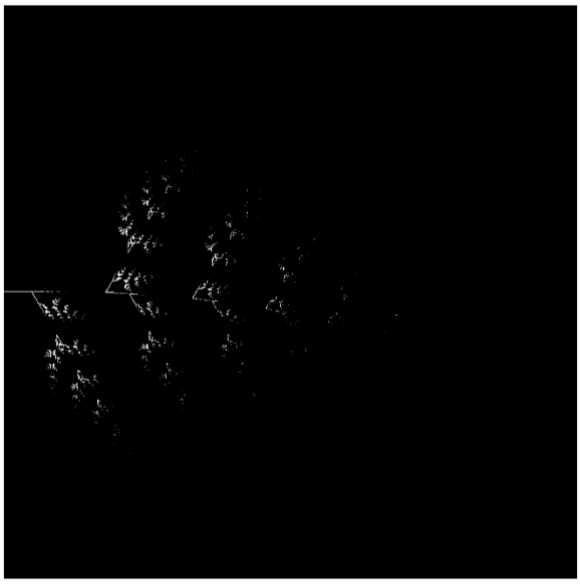


Figure 6: Fern image generated with 100000 iteration using probability set 1.

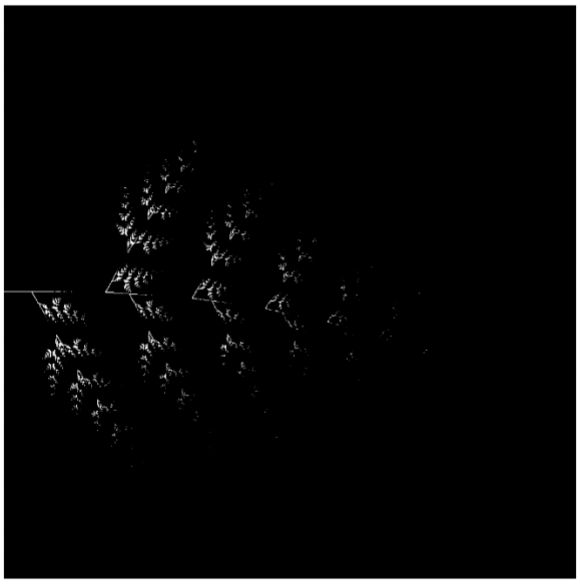


Figure 7: Fern image generated with 500000 iteration using probability set 1.

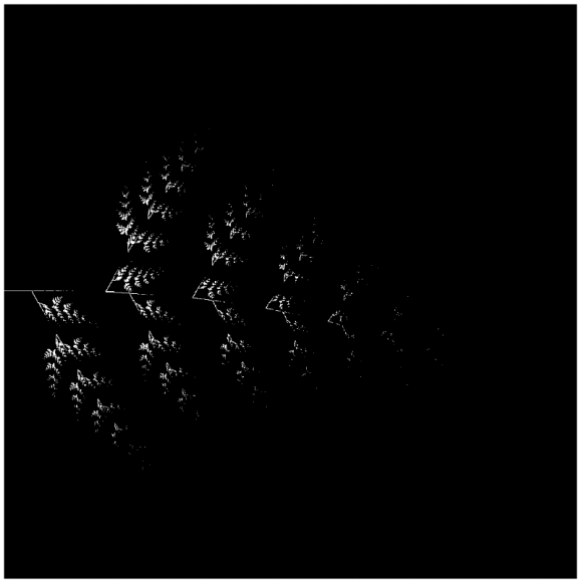


Figure 8: Fern image generated with 1000000 iteration using probability set 1.

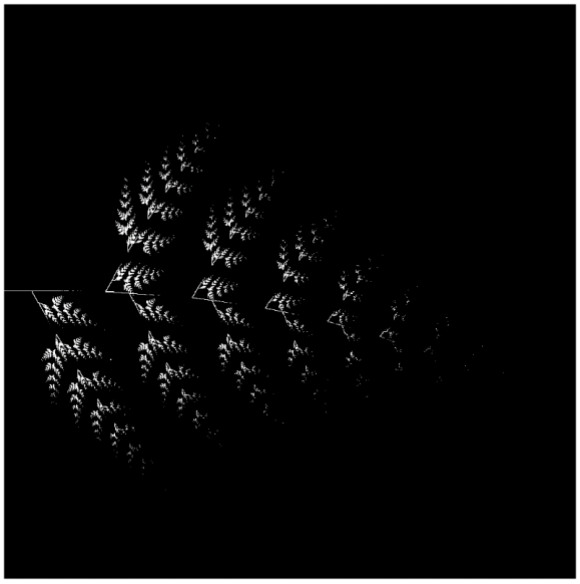


Figure 9: Fern image generated with 10,000,000 iteration using probability set 1.

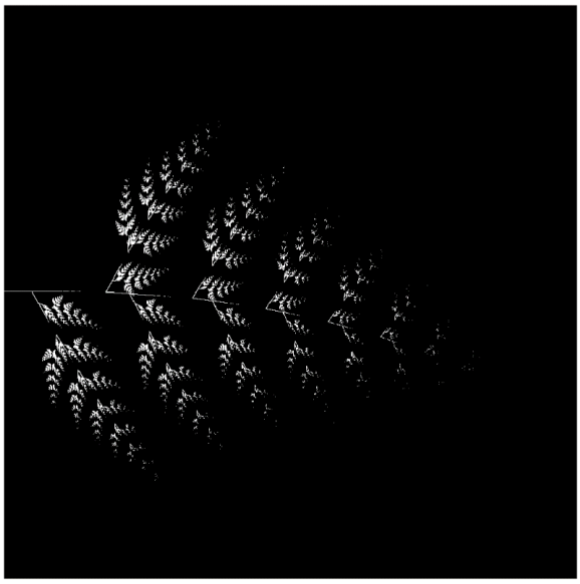


Figure 10: Fern image generated with 30000000 iteration using probability set 1.

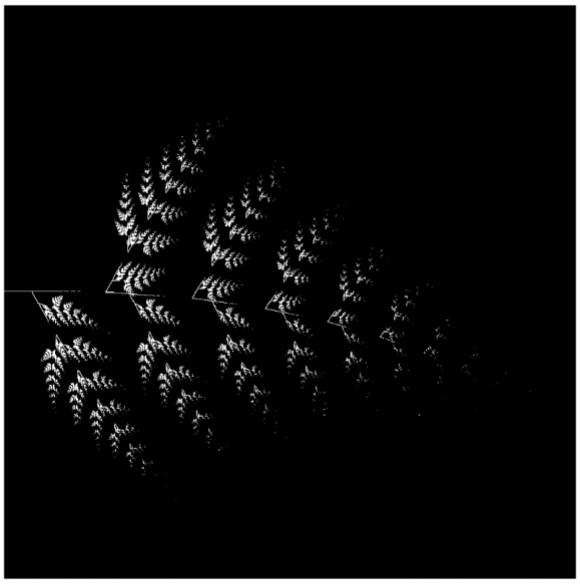


Figure 11: Fern image generated with 50000000 iteration using probability set 1.

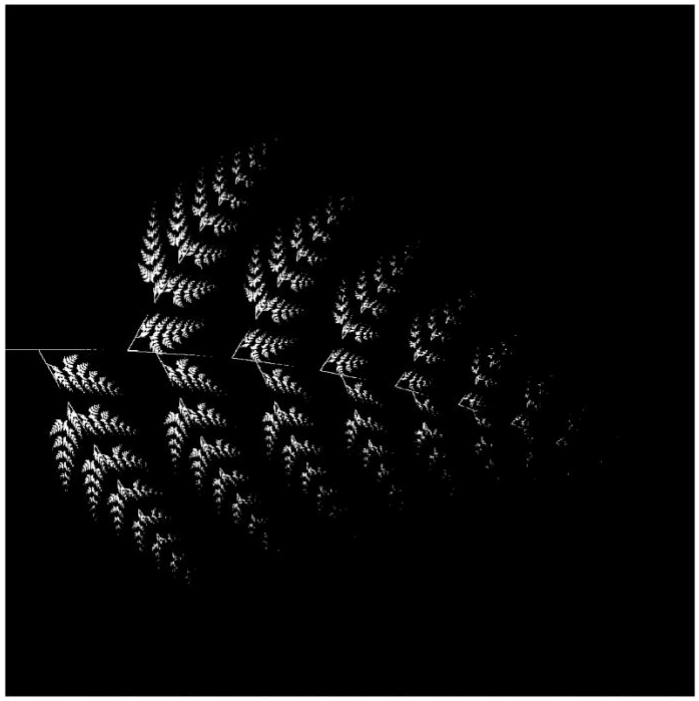


Figure 12: Fern image generated with 100,000,000 iteration using probability set 1.

It is observed that the with the execution of each iteration, we get a better version of fractal. Thus, if we keep on iterating again and again, better quality of fractal is obtained.

Probability Set 2 [0, 0.35, 0.35, 0.3]

RenderIFS Algorithm was executed for 100,000,000 iterations with probability set 2 and Affine transformation parameters which are specified in Table 1. In this probability set, p_1 was set to 0 and the probability of other 3 regions was adjusted so that the overall sum of probabilities is 1. The aim was to check which region was affected by p_1.

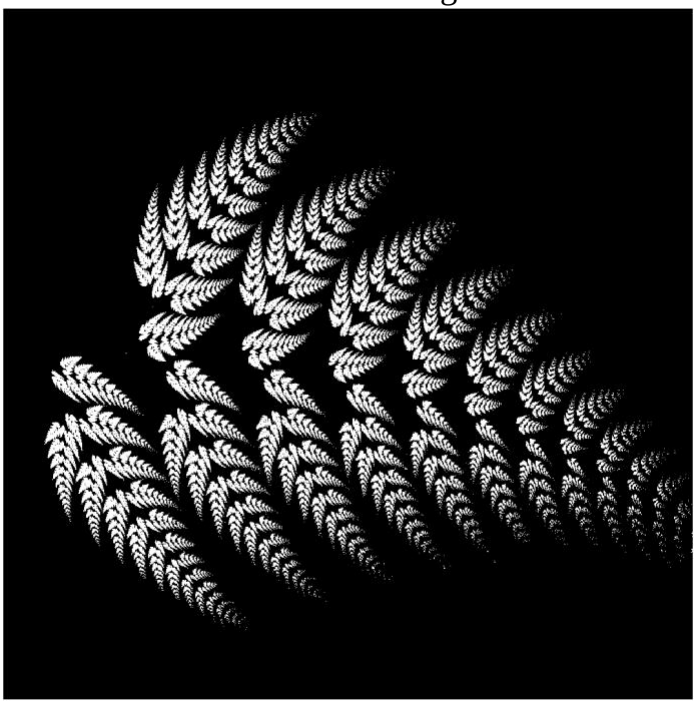


Figure 13: Fern image generated with 100,000,000 iteration using probability set 2.

The image generated has a fractal pattern, but middle stem is not visible. This shows that with $p_1 = 0$, one of the components of fractal is not generated.

Probability Set 3 [0.4, 0, 0.35, 0.25]

RenderIFS Algorithm was executed for 100,000,000 iterations with probability set 2 and Affine transformation parameters which are specified in Table 1. In this probability set, p_2 was set to 0 and the probability of other 3 regions was adjusted so that the overall sum of probabilities is 1.

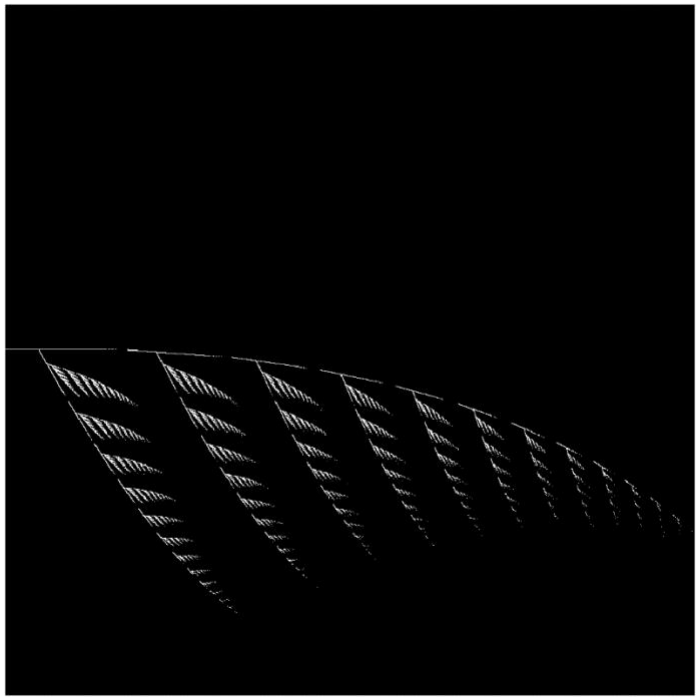


Figure 14: Fern image generated with 100,000,000 iteration using probability set 3.

As seen from Figure 14, pattern above the stem is not generated. However, fractal in other 3 regions were generated with respect to the probability set.

Probability Set 4 [0.2, 0.7, 0, 0.1]

RenderIFS Algorithm was executed for 100,000,000 iterations with probability set 2 and Affine transformation parameters which are specified in Table 1. In this probability set, p_3 was set to 0 and the probability of other 3 regions was adjusted so that the overall sum of probabilities is 1. The aim was to check which region was affected by p_1.

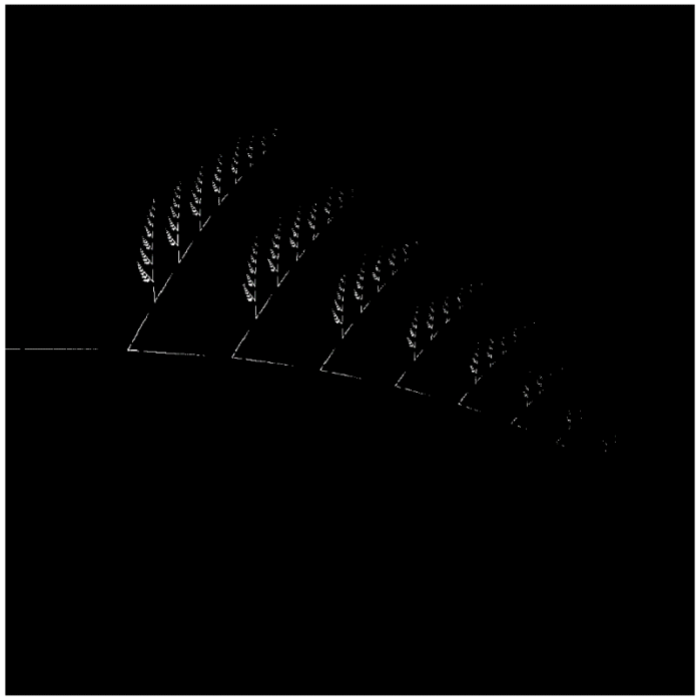


Figure 15: Fern image generated with 100,000,000 iteration using probability set 4.

From the figure 3, it is clear that region below the stem was not generated. This is because p_3 was 0. However, the other regions are also light. This is because probabilities are not equally distributed.

Probability Set 5 [0.3, 0.35, 0.35, 0]

RenderIFS Algorithm was executed for 100,000,000 iterations with probability set 2 and Affine transformation parameters which are specified in Table 1. In this probability set, p_4 was set to 0 and the probability of other 3 regions was adjusted so that the overall sum of probabilities is 1. The aim was to check which region was affected by p_1.

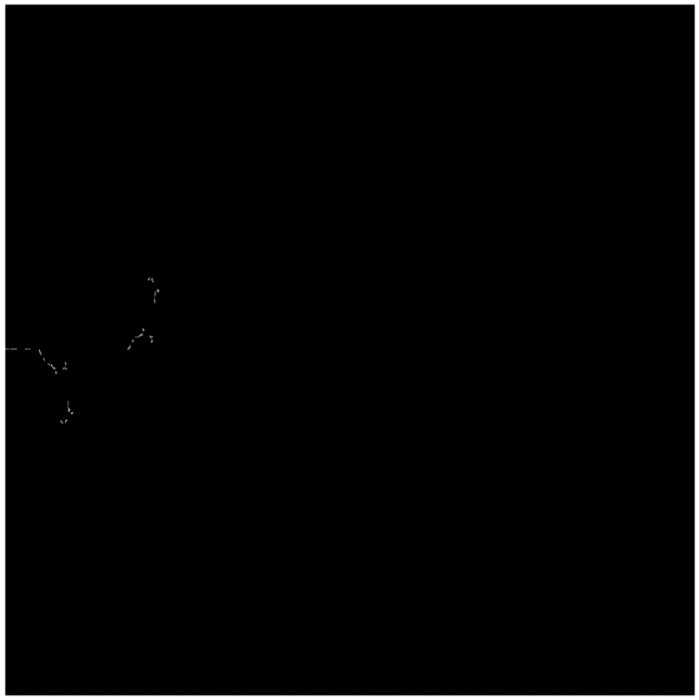


Figure 16: Fern image generated with 100000000 iteration using probability set 5.

The fractal generated with this probability set seems to be the worst one. Only few patterns are observable. It also shows that fractal generation is also dependent on widow values for which the algorithm is executed.

Probability Set 6 [0.840, 0.075, 0.075, 0.010]

RenderIFS Algorithm with the probability set mentioned above.

Figure 17: Fern image generated with 100000000 iteration using probability set 6.

The result shows that region with low probabilities were generated less. Since the major part of this fern fractal lies in region 2 and region 3 and the probabilities of these regions were less, very poor fractal was obtained.

Probability Set 7 [0.25, 0.25, 0.25, 0.25]

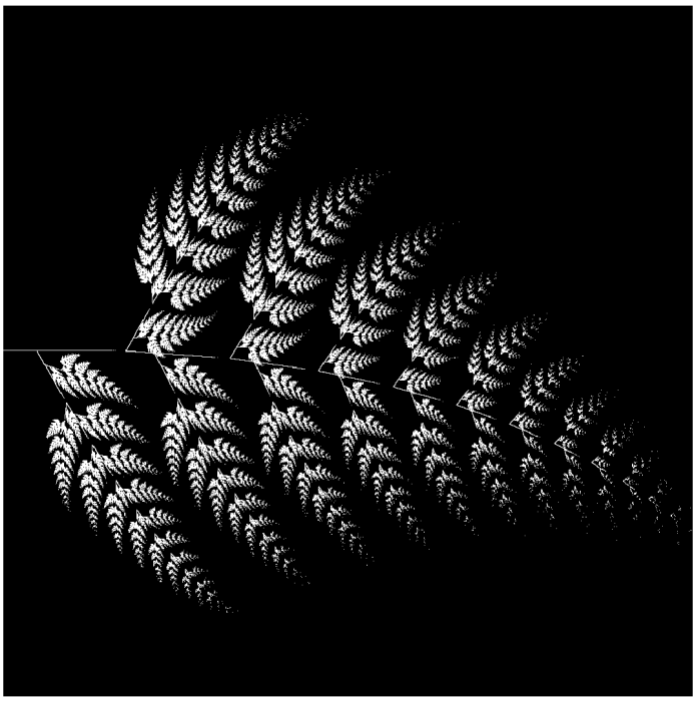


Figure 18: Fern image generated with 100000000 iteration using probability set 7.

All the regions were equally distributed. Hence, the fractal (shown in Fig 18) which was obtained is very dense. All the regions are equally visible.

Comments

From the above results, it is observed that fractal generation is highly dependent on the set of probabilities used. However, overall shape of fractal remains same. This is because the transformation matrices are the same in all the iterations. The probabilities determine the direction of spread at every iteration. Equal probabilities (0.25) would provide equal 12 density spread in all directions at each iteration. However, as the number of iterations increase, better quality of fractal was obtained.

D. Conclusion

IFS is a very simple algorithm from implementation point of view. However, as the number of iterations increase, it becomes computationally expensive and thus, takes lot of time to complete. The shape is determined by affine transformation matrices. However, the intensity of fractal is highly dependent on probability set and number of iterations. Also, in one case, we found that window to be used for fractal generation also plays a pivotal role. Thus, parameter tuning is crucial for an optimized IFS execution.