CMPEN 555: Digital Image Processing II

Computer Project # 5:

Fractal Generation Using Iterated Function Systems

Naichao Yin, Wenrui Wang, Zekai Liu

Date:4/24/2020

A. Objectives

1. Learn how to use MATLAB for general image processing.
2. Design an Iterated Function System with affine transformation.
3. Change the parameters of Iterated Function System to get the influence of each parameter.

B. Methods

In the project, our goal is to reconstruct the image using Iterated Function System for the "Fern". An image could be contracted nearly in one point by applying affine transformations. A transformation is a contraction map if:

$$d(f(x), f(y)) \le cd(x, y) \quad . \ (1.1)$$

The c in 1.1 is between the range of 0 and 1. Points get closer to each other as mapping applied. When iterate the transformation many times, the image converges to one point. Based on the theory, it is possible to reconstruct the original image if provided the function f. This method is called Iterated Function System. Provided the collection of contractive transformations and set of probabilities, we can do the reconstruction. The principle is basically to make an affine transformation of x-> Ax + b for a point. It can be described as following.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (1.2)$$

In this project, we could reconstruct "Fern" by applying the matrix listed in L23-11. The probability is also listed. Randomly select a point as the original point and multiply the matrix with

the property predefined. The more times iteration is applied, the more similar result we get with the original image. When the probability changes, the result could be different.

MATLAB

The codes of this project conclude several parts: generate affine transformation matrices, generate different probabilities and generate images with specify size. The all four affine transformation parameters are stored as two multidimensional array A and B. The function randbyp() is used to generate 1 ,2 ,3 and 4 with four different probabilities. Then the result of original IFS is generated after N times iterations. The process of the codes can be shown as following.

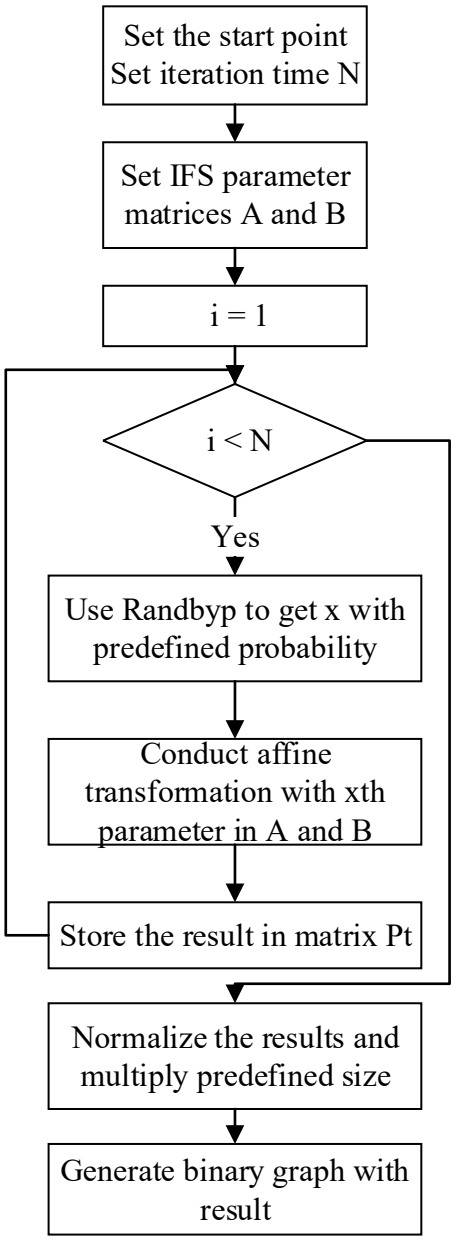

Figure 1. flowchart of codes in project

C. Result

1) Question one

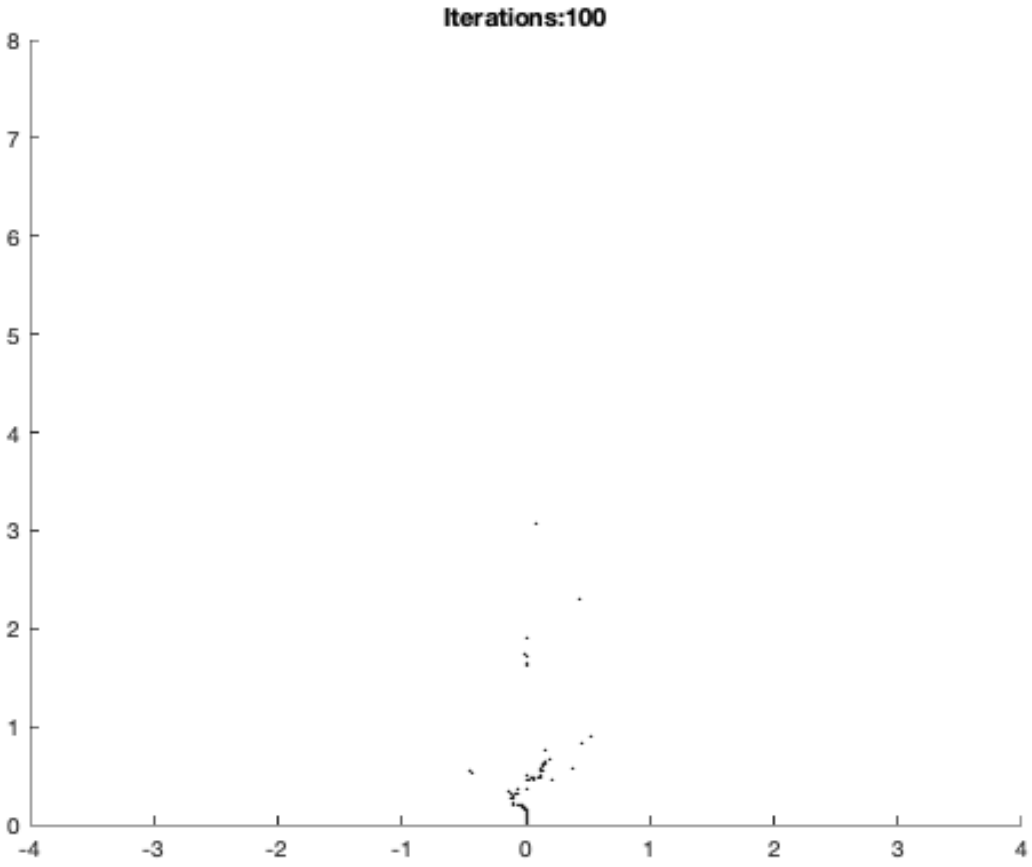

Figure 2.1.1. 100 iterations of Render IFS

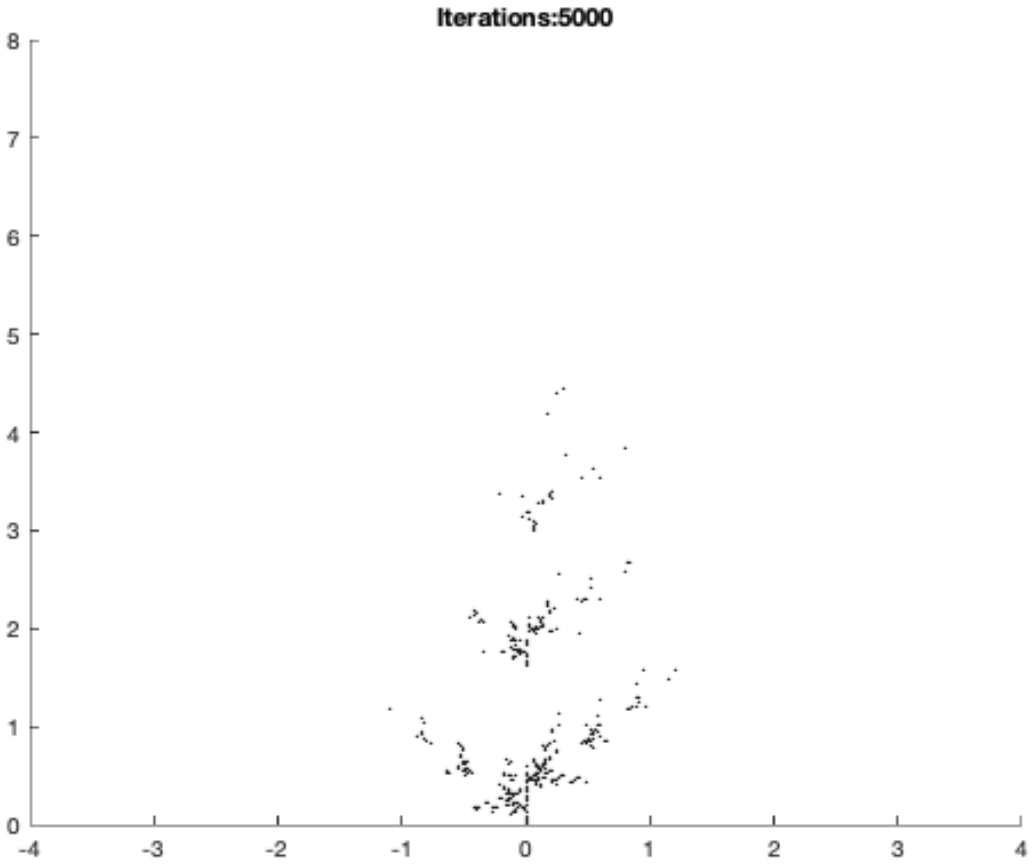

Figure 2.1.2 5,000 iterations of Render IFS

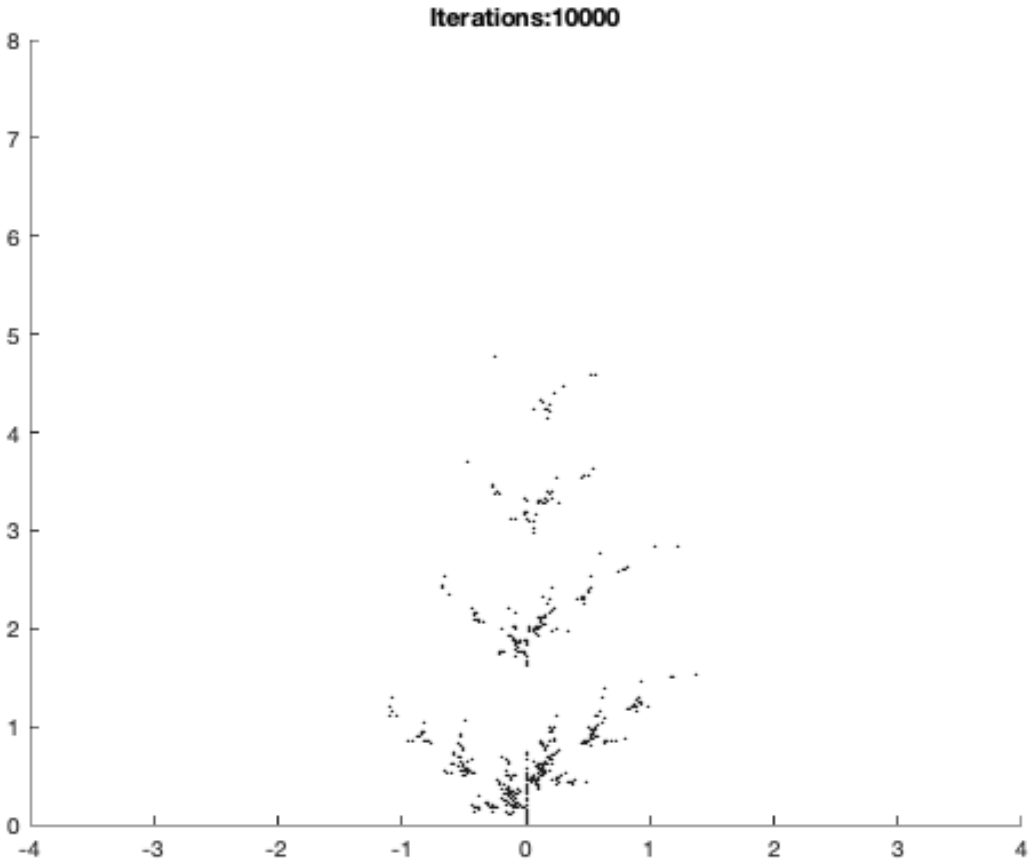

Figure 2.1.3 10,000 iterations of Render IFS

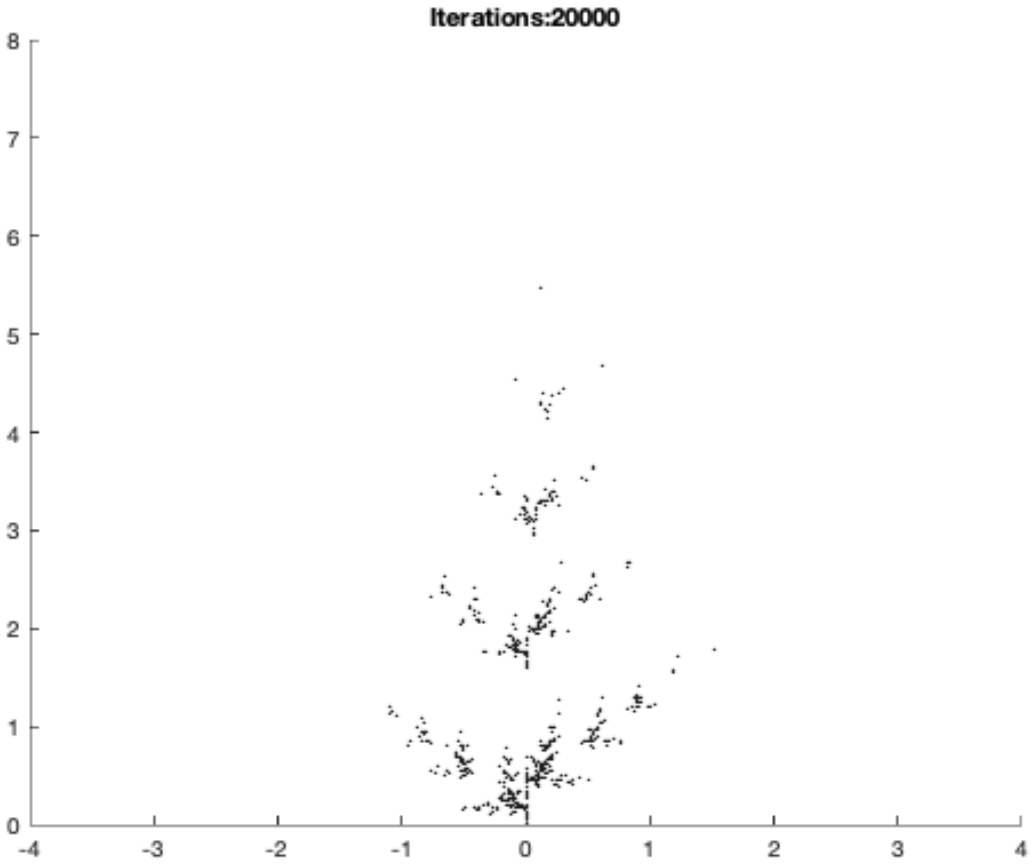

Figure 2.1.4 20,000 iterations of Render IFS

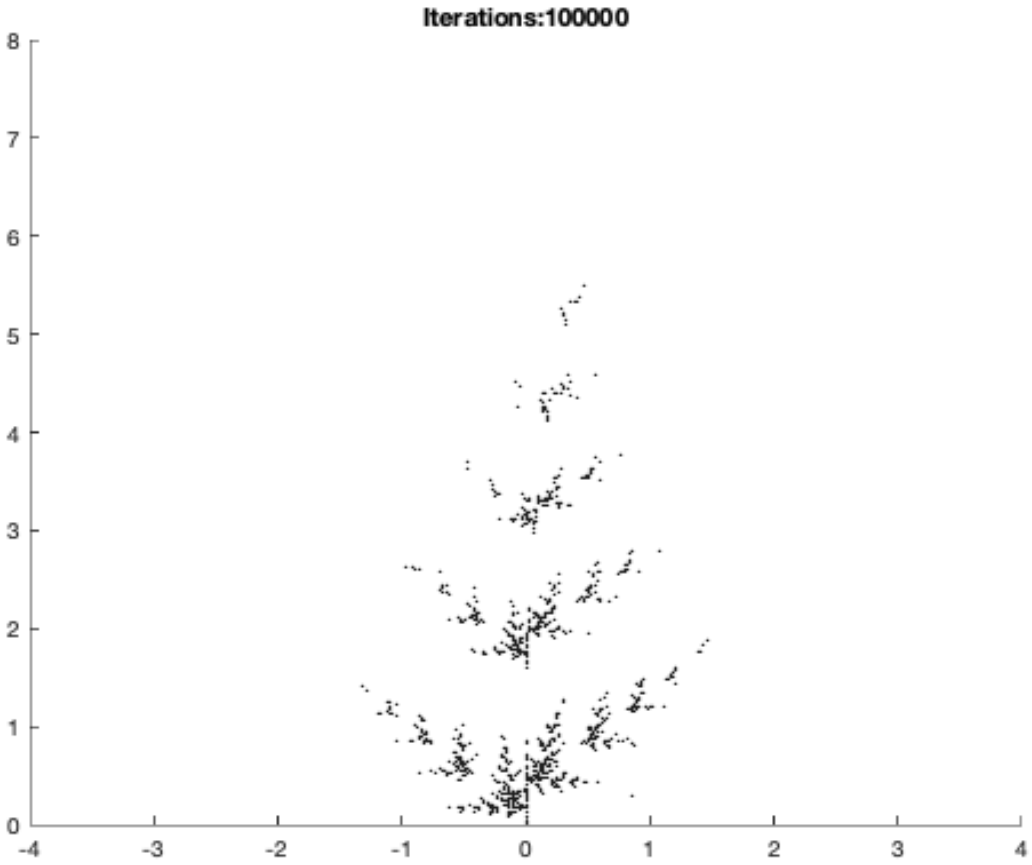

Figure 2.1.5 100,000 iterations of Render IFS

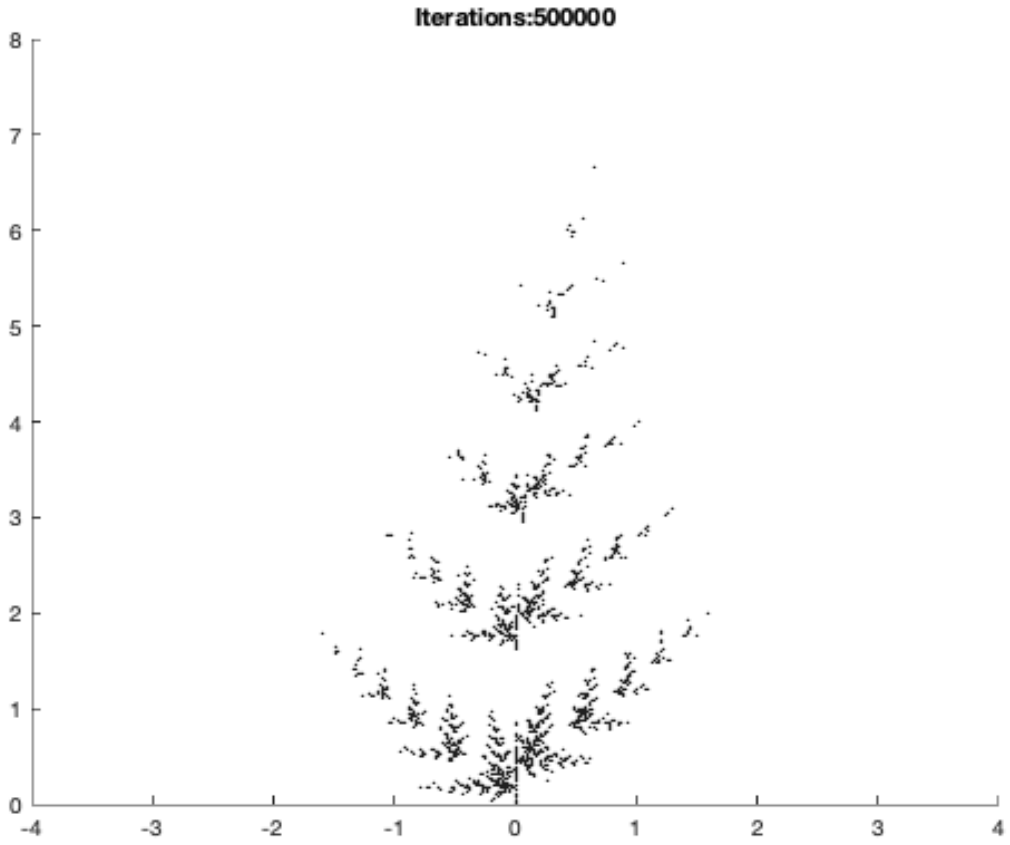

Figure 2.1.6 500,000 iterations of Render IFS

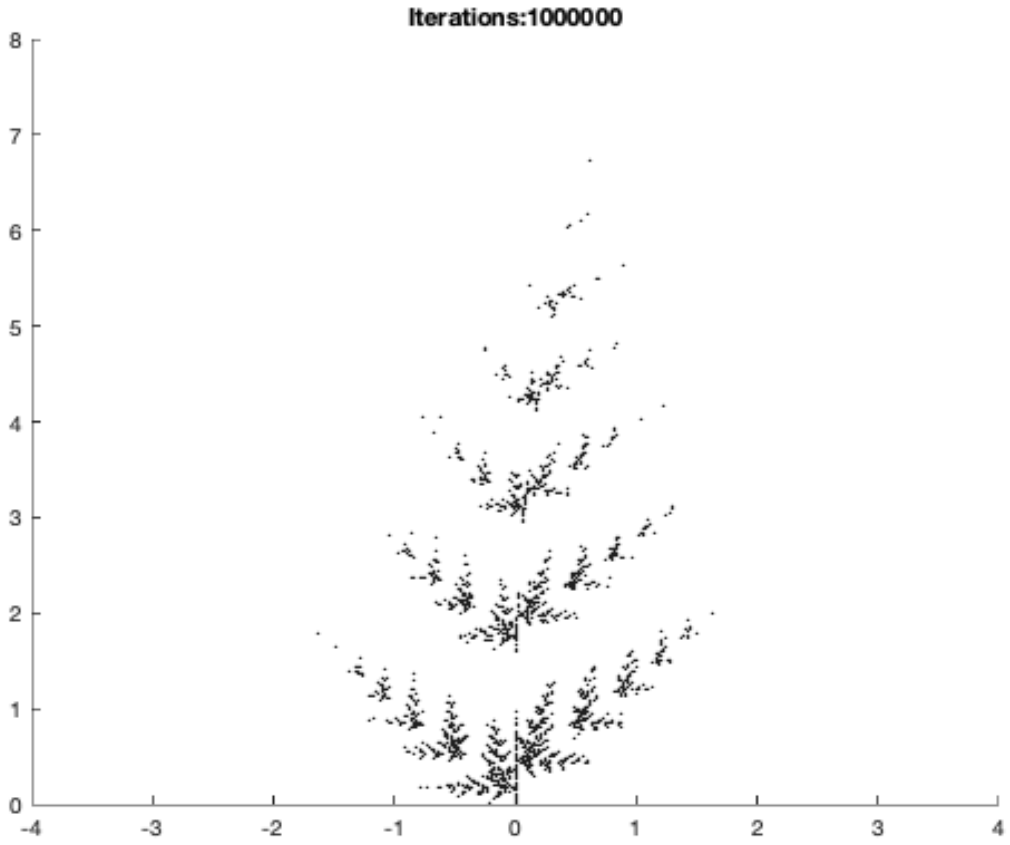

Figure 2.1.7 1,000,000 iterations of Render IFS

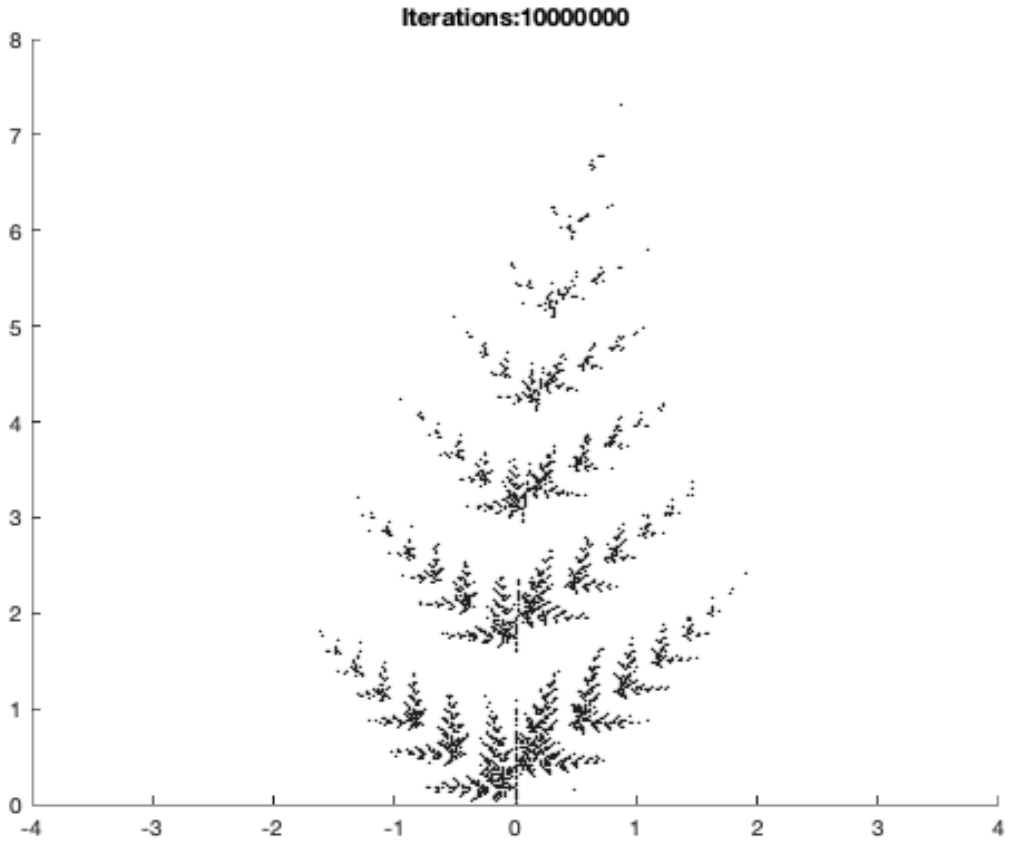

Figure 2.1.8 10,000,000 iterations of Render IFS

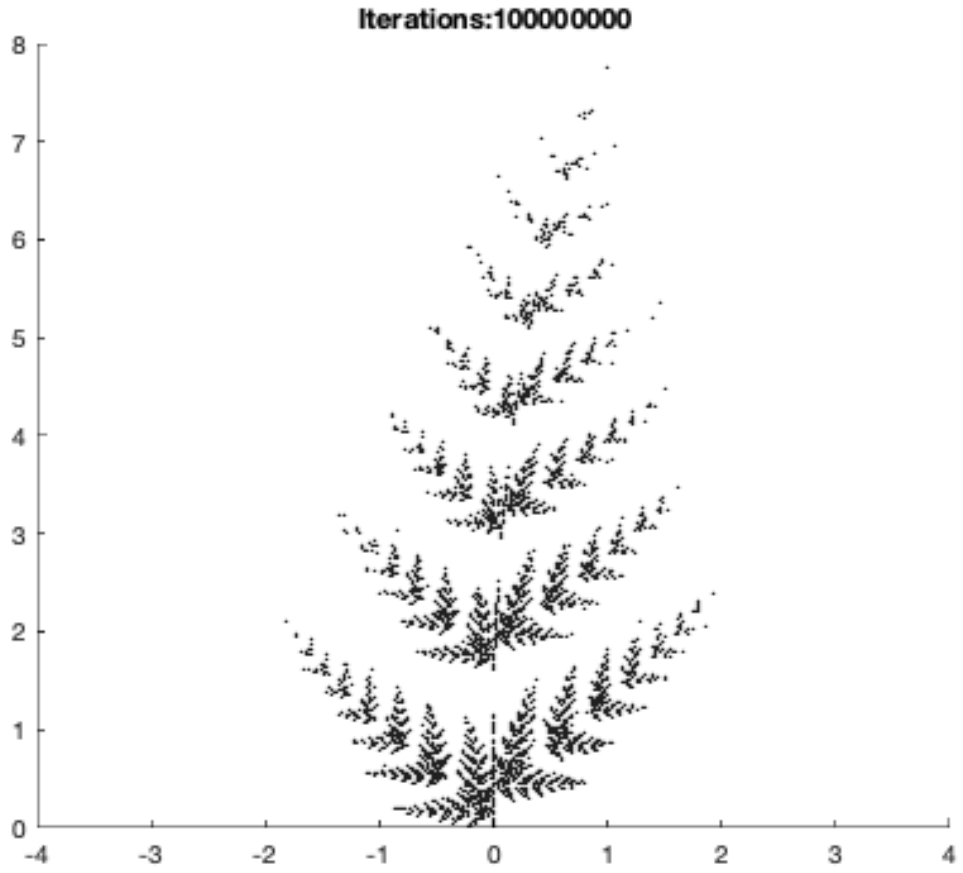

Figure 2.1.9 100,000,000 iterations of Render IFS

According to all the above Figures, we can conclude that the generation of points is not a bottom-up accumulation, but a certain probability of falling on the position of the result set of an affine transformation each time. As the number of points (iterations) increases, the contour becomes more obvious and refined.

Although the starting point is randomly chosen, the result is the same shape and every sub-leaf on the fern is the same shape as its leaf, so it doesn't matter where the position of the starting point. Because this transformation is a linear transformation of coordinates, it does not change the original linearity and parallelism, it just flips and scales. Therefore, under certain actions of the

same transformation matrix, the action of two points in a different position is the same, so the resulting shape is also the same.

2) Question two

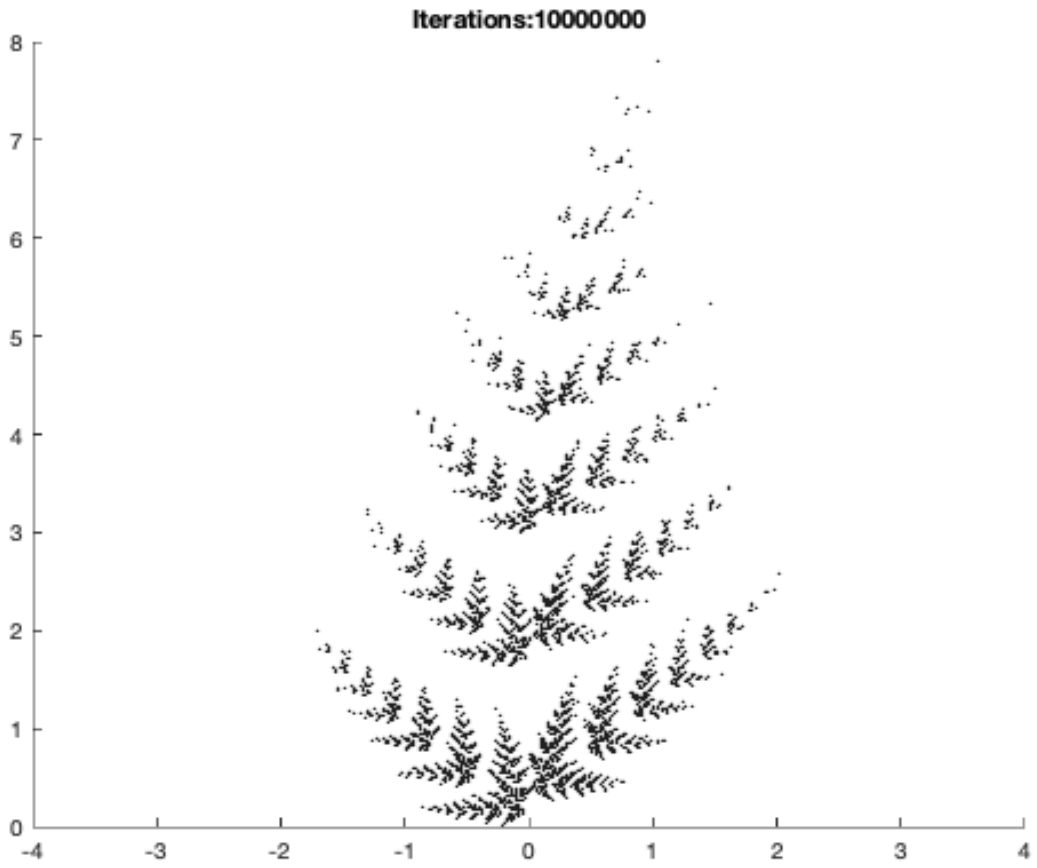

Figure 2.2.1 The iteration result after zero out p1

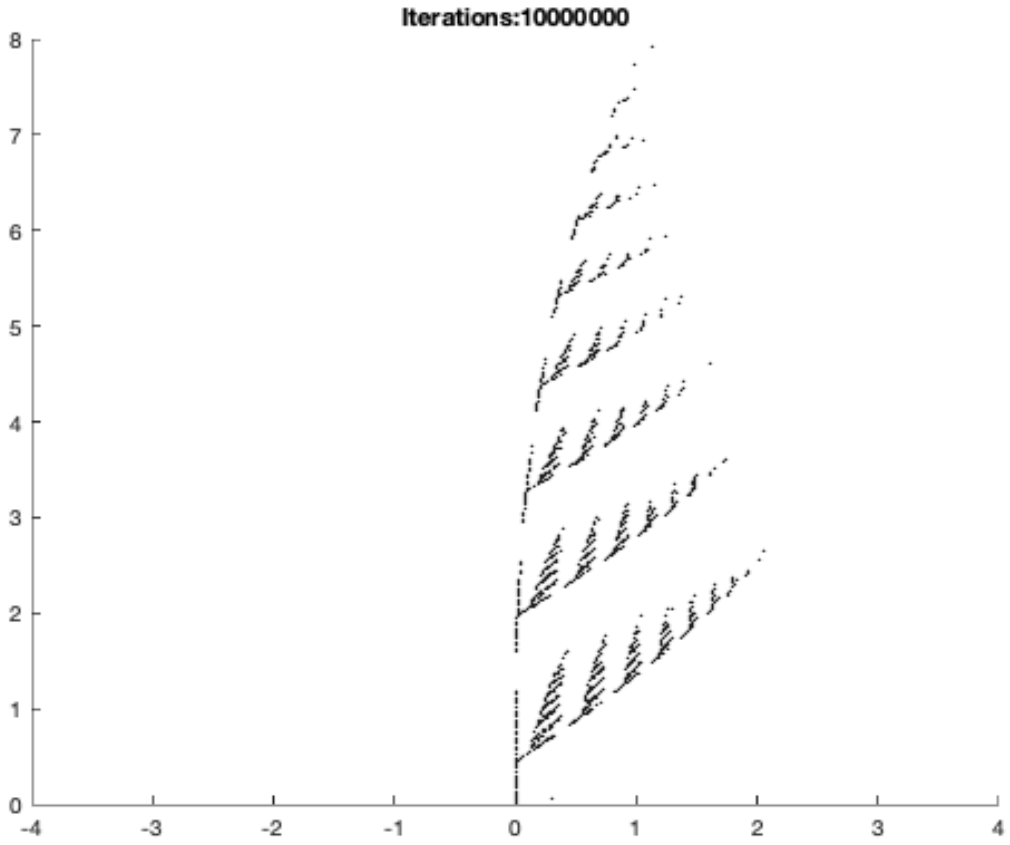

Figure 2.2.2 The iteration result after zero out p2

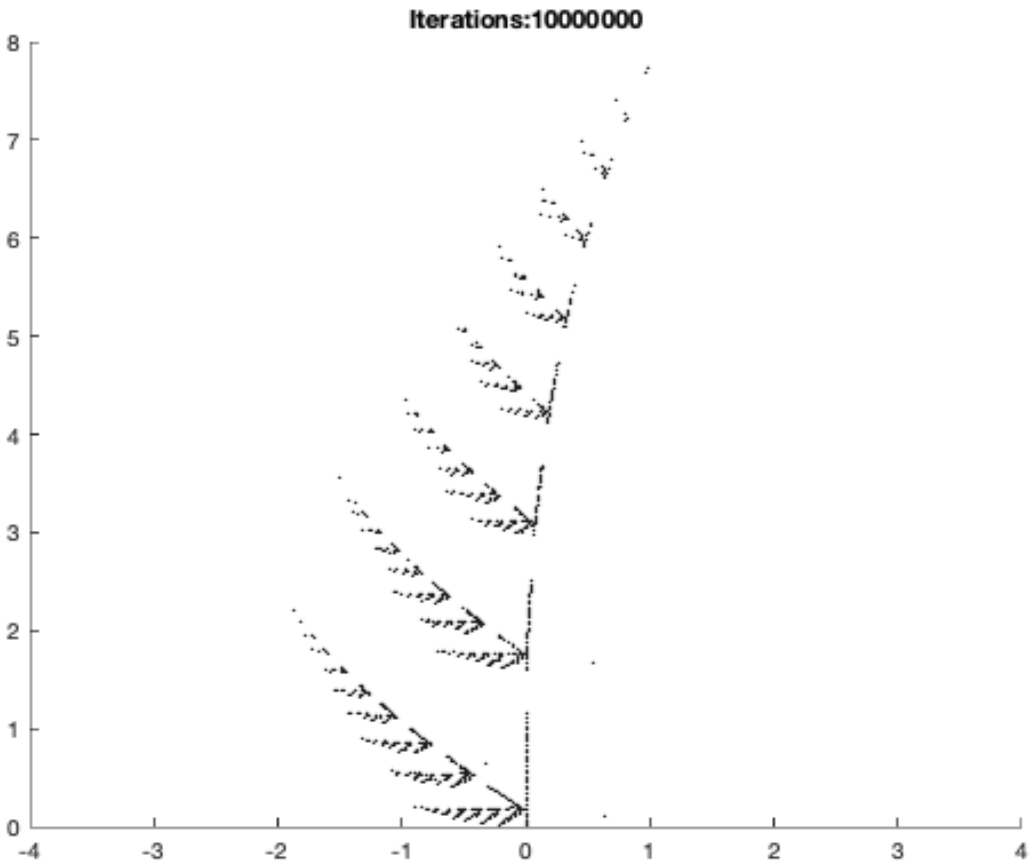

Figure 2.2.3 The iteration result after zero out p3

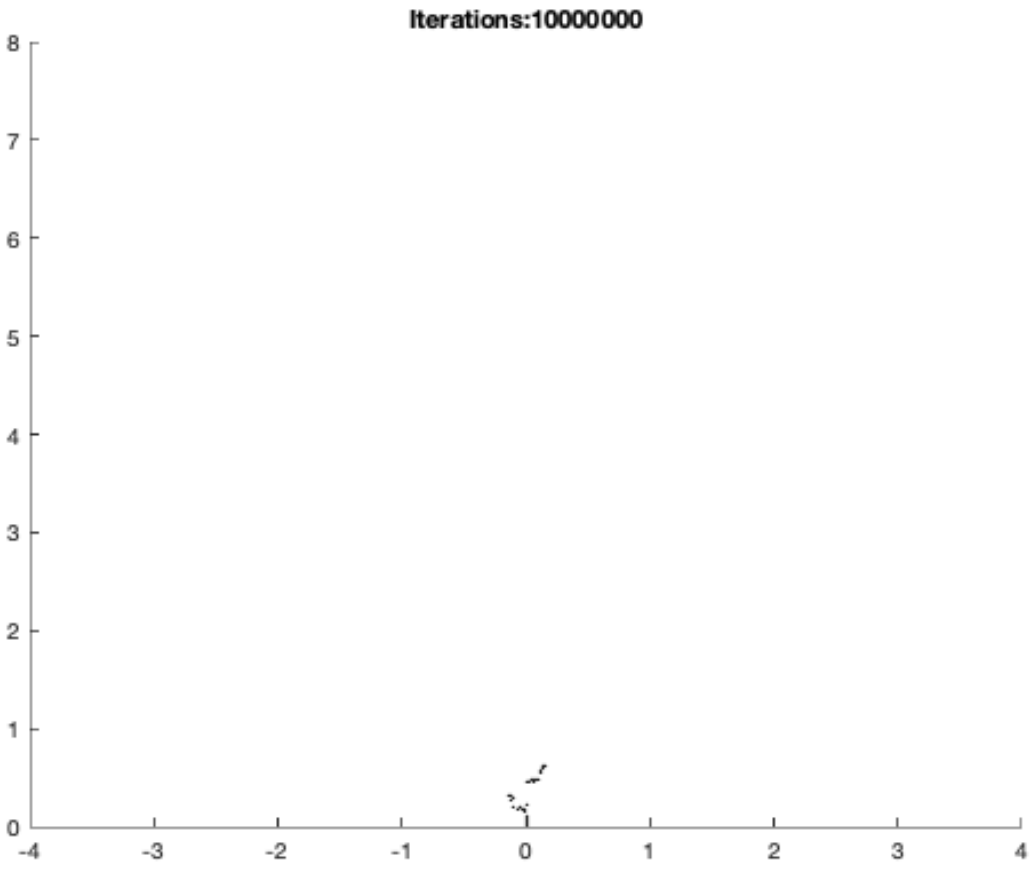

Figure 2.2.4 The iteration result after zero out p4

The structure of results is controlled by the affine maps {w1, w2, w3, w4} in the code. The combination of these affine maps fixes the geometry of the underlying model and will, in turn, determine the geometry of associated images. Therefore, if we artificially zero out the affine transform w by removing its probability, the corresponding affine mapping space is also lost. For example, the left side of the leaf disappears by setting p3 to 0 (Figure 2.2.2), and the right side of the leaf disappears by setting p4 to 0 (Figure 2.2.3).

D. Conclusion

Iterated Function System(IFS) is a method of constructing fractals, and the results are usually self-similar, that is, the part is a small copy of the whole. IFS is defined by a set of affine transformations, which usually include rotation, scaling, translations, skew, and so on. Each transformation function must be contracted, which means that the distance between two points must be reduced. These methods do not change the self-similar object shape, according to this self-similarity, we can start from a point or a simple geometric figure, according to a certain mapping, repeatedly iterate until generating a complex image. The final result of IFS is independent of the initial position of points but only depends on the affine maps of the iteration.

The RenderIFS algorithm in our project can be widely used in plant simulation and other irregular contour images. But there are still some shortcomings, such as the plumpness and the sharpness of the reconstructed image.