MODELLING COMPLEX SYSTEMS

FRACTALS

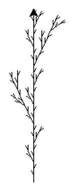
October 11, 2018

David W. Landay, Maxfield Green
University of Vermont
Comp. Systems and Data Sci.

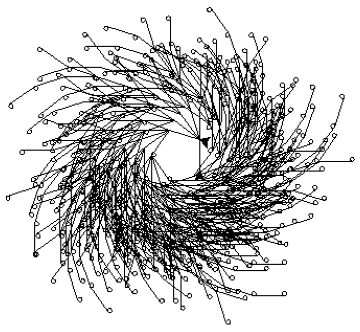
0.1 L-SYSTEMS

1) Fix/recreate the L-Systems Code from the blackboard. Use modular design provided by Prof. Eppstein. Focus on: lack of modularity, hardcoding,misleading variable names, inefficient codes, poor documentation. Demonstrate that the program works properly by reproducing the Weed-1 fractal of the text Fig 6.5.

The L-Systems program was rewritten in Python as it was a language we are more familiar with. We designed a modular program that has methods or functions that call one another to expand and draw the L- system. A user can input an axiom and a set of rules that will be applied to move the system forward. In this way the program does not contain any hard coded values such as the original program. Additionally, intuitive variable names are used throughout to maximize readability. The weed from the textbook was reproduced and is shown below. The program written to produce this particular L-System is attached to the assignment.



2) Modifying the improved code, implement integer multipliers of + and - that flake describes in table 6.2. Model something with the real world.
I choose to model a birds nest. I implemented the ability for Carl, our penguin to rotate N number of times in either direction before stepping forward when directed. This made creating the torus shape of a nest much easier. I also added very small circles in the hopes that they would add some dimensional and variance to the image. The final image is shown below. A further question that we have is whether one could infer an L-System Rule from the structure of the fractal. There must be classes of fractals that can be grown by groups of similar rules. Are there characteristics of the very cool looking fractals that appear in the L-System rules? It is wild that some of the patterns that are grown were ever discovered.



(a)

(b)

Figure 1: Birds Nest and L-System Fractal Comparison

The image follows the rules:

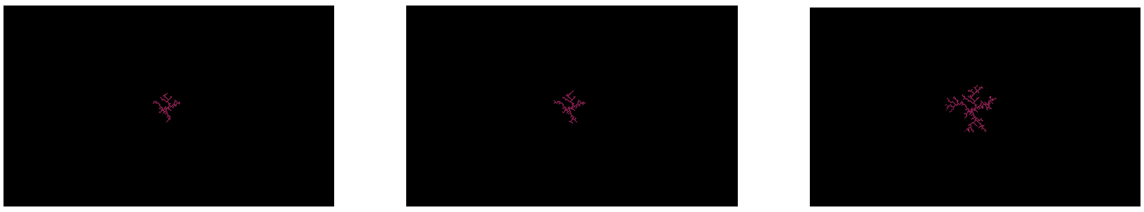
Axiom: "F"

Rules: "F","[","F","-","F","C","]","+","[","F","+","F","C","+","F","C","]"

The potential integers in front of the +/- indicates the number of n degree rotations that will take place. To determine this rule, a small version of the nest and the accompanying L-System rules were drawn and than refined over time. Originally, the integer multiplication was used but after trial and error the figure more closely resembled the birds nest with consistent angle changes. However, to prove the concept, an example of the implementation in the code is included that can be run if desired. To improve the system, one would implement a minor change that would allow the system to take in more than one axiom and more than one set of rules.

0.2 Diffusion Limited Aggregation

Diffusion Limited Aggregation, or DLA, is a random process that can be used to model particulate diffusion through a medium. Imagine an attraction process by which particles move randomly about a space until they collide with a fixed particle. If we assume that the collisions are inelastic, then the randomly moving one will become fixed in a space adjacent to the particle it collided with. The DLA process can be seen in culture growth in biology, and many other diffusion processes in chemistry and physics, where branching structures emerge as collisions occur over time. The figure below demonstrates this growth process.


(a) growth at stage a **(b)** growth at stage b **(c)** growth at stage c

From the stages of growth seen above, we can see a fractal pattern emerge as more and more collisions occur. This is due, in part, to the fact that the probability of a collision is higher when a randomly moving particle is in the neighborhood of a fixed one. Hence, we see a branching effect as the system develops over time. The easiest way to model this system and see the reaction first hand, is to initialize a grid of some dimension (width x height) and introduce n particles at randomly selected cartesian coordinates (i, j), for $i, j \in \mathbb{Z}$. In addition, introduce a fixed particle to the center of the grid, $(\frac{W}{2}, \frac{H}{2})$, where W and H denote the width and height of the space respectively. With equal probability, choose to add $+1$, or -1 to one of the cardinal directions to create a random walk for each particle. If the absolute distance between a walking particle and fixed particle is 1, then the walking particle stops moving. In practice (code), we can create an object called `Particle` that takes in a position as a mutable representation of a coordinate, and a boolean value to indicate whether or not it should be moving. A `move()` function will then choose the direction in which the particle should move at the next time step. If we keep all of the moving particles in one data structure, and the fixed particles in another, then the quantity of fixed particles should increase as moving particles flow to the other data structure. Hence, the process can be allowed to stop once the number of moving particles becomes 0, or after some predetermined number of `steps`.

Of course, this process can take infinite time if we allow the wallks to be truly random. Some techniques we used to speed the process up were to change the allowable sticking distance a moving particle could be from a fixed one, to be greater than 1, and to change the topology of the grid itself. For example, by changing the "sticking" distance, we reduce the hypothetical time it will take for all particles to become "stuck". By adding periodic boundaries at the edges of the grid, we prevent particles from walking a certain distance away from the fixed particle, thus improving the odds of a collision (think asteroids). consequently, we are mapping the grid onto the surface of a torus when we do this.

Another method we attmpted (not shown here) was to initialize a circle of fixed points about the center. The idea here is that the growth process will now occur at points along the circle. Since the circle covers a greater area than a single fixed particle, then we expect to see the branching effect occur more often. A cool consequence of this technique is that branches begin to form in a radial pattern.

0.2.1 Box Counting Dimension

The concept of a fractal dimension was developed as a means of quantifying the measure of a fractal. As an example of measure, a line has dimension 1 because it only requires knowledge about its span along its only axis (dimension), so we say that the measure of a length (span) is linear. Similarly, an area has dimension 2 because describing it effectively requires measuring two lines; one in the "x" direction, and one in the "y". A cartesian coordinate system, for example, can be described by real number lines in two distinct dimensions. Finally, volume is 3-Dimensional as we need only add another number line with which to describe any point within it's space.

For an object of siz N, we can estimate a total measure of that object using some measuring device a, by $a \times N$ (think a ruler to measure an area of paper).We can re-write the relationship:

$$N = \left(\frac{1}{a}\right)^D$$

so that the dimension D of the object is in terms of the device used to measure the total measure of the object, and the number of objects as follows:

$$D = \frac{\ln(N)}{\ln(\frac{1}{a})}$$

If we conisder N boxes of size $a \times a$ covering the span of our grid, we can count how many boxes contain elements of our DLA system and compare that to the total number of boxes covering the space to understand the overall dimension of the procedure.

Below a the number of boxes is plotted against the size of boxes in log-log space. The slope of the line of best fit is an approximation of the fractal dimension of the object in the image. The Box Counting Dimension algorithm used multidimensional histograms to determine when boxes covered the fractal object. As we expected, the fractal dimension is slightly less than 2. However, we would expect that given additional data, the dimension would be closer to 1.7, accepted as a typical dimension for a DLA structure in 2D space. It is unclear why there is a slight disparity in this calculation, potentially a larger set of points are needed. However scale should inherently be invariant in this calculation, and the dimension of the DLA system should be fractal.

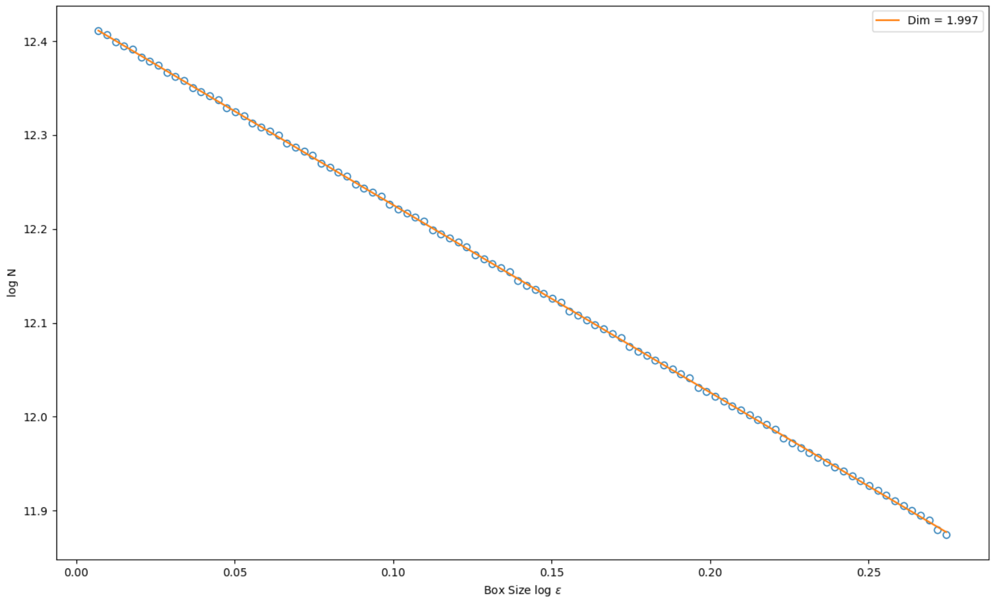


Figure 3: fractal dimension of the DLA model with periodic bounding, image dimensions: 500x500 pixel grid