# Biomath Final Project (Investigation Through Programming)

Kevin Roberts

April 19th, 2023

## 1 Modeling Diffusion Limited Aggregation with Python

For my final project, I decided to write a python program that models diffusion limited aggregation (DLA). Recall that DLA is the process by which an initial group of particles (red dots in my case) with a neighboring boundary (purple dots) grows as other wandering particles hit its boundary. Observe the following image of an initial aggregate with a wandering (black dot) particle:

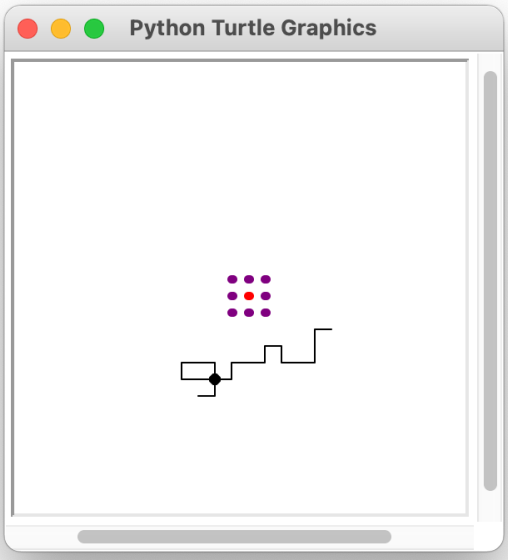



Figure 1: Initial Aggregate

I've written a python program that models DLA almost exactly (with a killing boundary addition) and in the next few paragraphs I'll explain how it works. I won't go too much into programming

details, instead I'll explain what I believe are the important attributes.

Inputs
To start, in the code Appendix, observe lines 278, 279, and 280. In line 278 a user can pick the environment size to be a value to be 1, 2, 3, 4, 5, 6, 7 or 8. Picking 1 will generate an environment with a size of 100x100 pixels, 2 will generate an environment that is 200x200 pixels, and so on. In line 279, a user can pick how large they wish the aggregate to grow. For example, I have picked 20 in my example and so the aggregate will grow until there are 21 red dots (21, not 20 because there's an initial red dot). In line 280, a user can choose how many purple dots surround a red dot. This value can only be 4 or 8. Observe in the figure above how I've chosen there to be 8 surrounding purple dots.

Run time
After the inputs are set, I use Python Turtle animation to draw the initial aggregate and then run a while loop where particles (black dots) are randomly placed into the environment. There are a few checks I make to ensure that the position of the randomly placed particle is "good". For example, I've made it so the particle cannot spawn inside the the imaginary black square boundary. This is to ensure the particle isn't randomly placed inside of a hole in the aggregate. Observe the following image that describes where a particle can be placed:

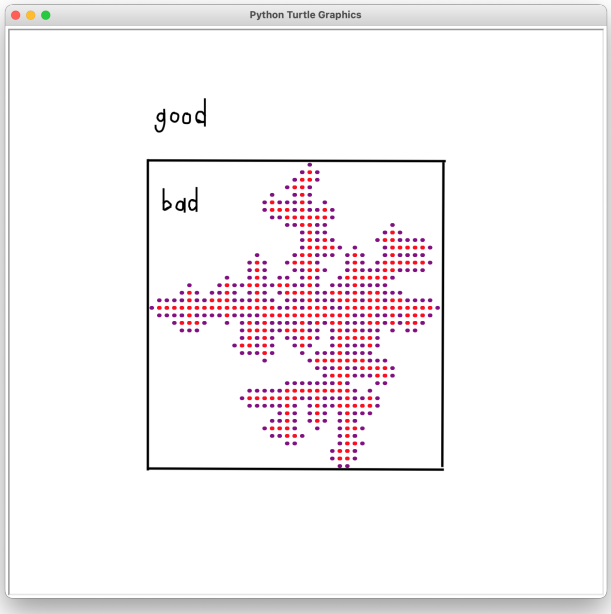


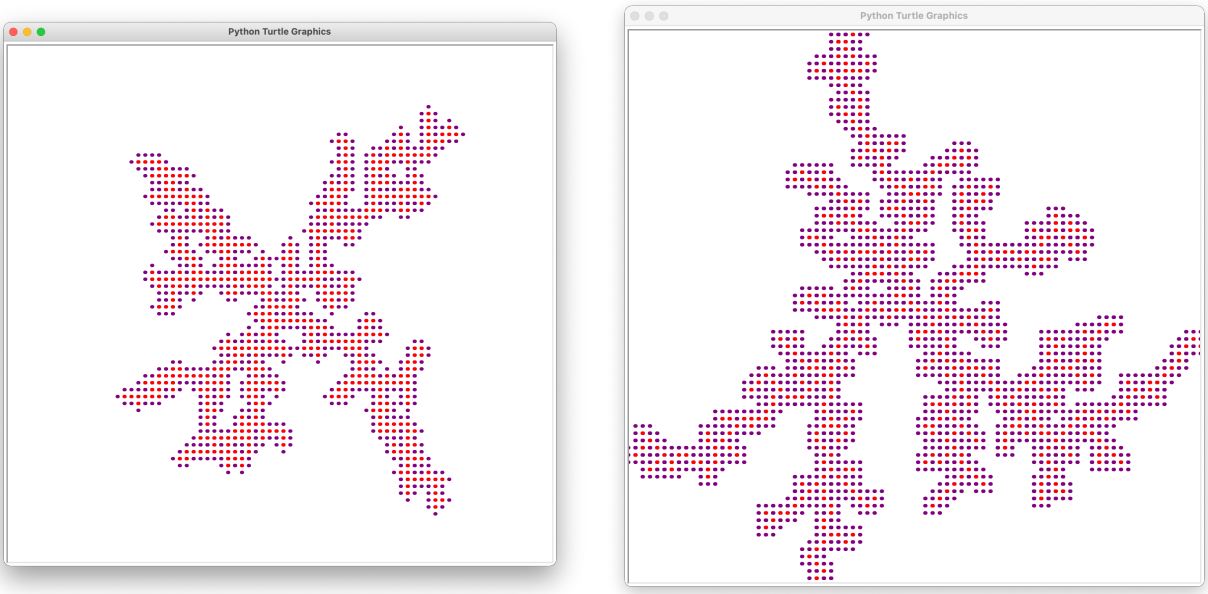

Figure 2: Describes where "good" and "bad" spots are to spawn a random particle.

After the checks are made, the particle spawns and wanders around. Every time it moves, it decides

randomly (with equal probability) to face up, down, left, or right. Then the particle moves forward. It continues this behavior until it either hits the boundary of the environment (if this happens, then I kill the particle and spawn a new one) or hits the boundary (a purple dot) of the aggregate. If the particle hits a purple dot, then I update the entire aggregate. When the black dot hits the purple dot the purple dot turns into a red one and it's surrounding dots turn purple (or stays red if one of it's neighbors is already red). After the aggregate grows, another particle spawns and the process is repeated.

Data collection

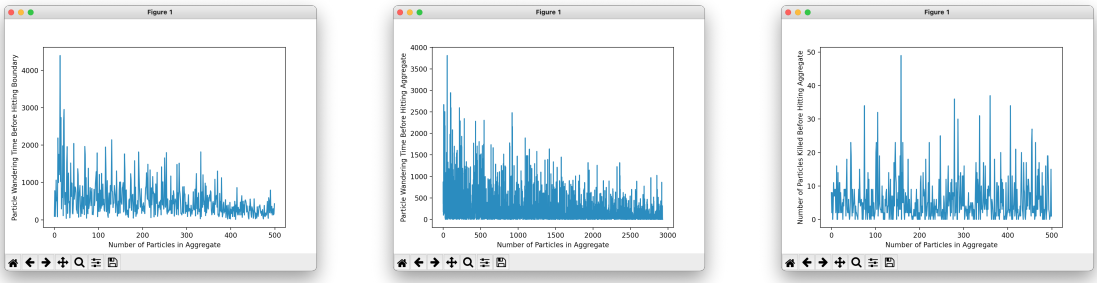
From our discussions in class, we assume that DLA models closely snowflake-like images and perhaps crystals as well. Running the DLA program I wrote until the aggregate grows to 501 red dots, with an environment boundary of 800x800 pixels, I obtained the following two images which look similar to what we expect:




(a) (b)

Figure 3: Aggregate growing to 501 red dots with (left) 4 purple boundary dots to each red dot and (right) 8 purple boundary dots to each red dot.

From Figure 3, I have found that the aggregate grows to be much bigger and faster with 8 purple boundary dots instead of just 4. Additionally, I've collected data for the Figure 1 (a) which tells us more about the behavior of the aggregate:




(a) (b) (c)

Figure 4: Aggregate data from Figure 3 (a) image.

From Figure 4, we see that the wandering of the particle on average decreases as the aggregate gets bigger. This can be observed in (a) and (b) from Figure 4. Figure 4 (c) did not make sense to me at first. It would seem that as the aggregate grows, there wouldn't be as many particles to kill because we assume the particle has a higher chance of hitting the bigger aggregate. However, as the aggregate grows, the places the particle can spawn creeps closer and closer the to environment boundary. This causes more and more particle killings which makes sense why particle killings do NOT go down as the aggregate size increases. This is solely due to the killing boundary attribute.

Future Simulations

I've found that with my DLA program I can model DLA and get results very similar to what we expect to see from class notes. Future programs that I plan to write will model IDLA (Internal Diffusion Limited Aggregation) where a particle spawns in the center of the aggregate and increases the aggregate size once it hits the aggregate boundary. This simulation wouldn't be difficult at all to add, given what I've already written in Python. Additionally, I'd like to add a "sniffing" effect to each black particle. With the "sniffing" effect, the particle will move in a biased direction towards the aggregate instead of moving in a simple symmetric random walk. Finally, I'd like to add multiple particles in the environment that move at the same time, which can be used to model FPT (First Passage Times) and slowest first passage times.

## 2 Code Appendix

Code for this project can be found here:
https://github.com/Kevin-Jay-Roberts21/DLA-IDLA-Simulations