In-Class Work: Fractal Growth

Intro to Fractal Growth

Let's gather info you had found for HW7 (task 2a,b):

- Cellular Automata
- Fractal Growth
- Diffusion-Limited Aggregation (Gould & Tobochnik Ch13, pages 529 and problem a on page 532)

1. Fractal Growth: Random Walk in Two Dimensions

1a. For the fractal growth DLA model we will need a random walk in two dimensions. Write a python-program for a random walker on a two dimensional lattice (all four directions being equally likely), starting at $x = 0$ and $y = 0$ and (print and) look at $x(t)$ and $y(t)$. You may use the solution program `~kvollmay/share.dir/inclass2019.dir/classrndwalk2a.py`.
For looking at $x(t)$ and $y(t)$ in the same figure, you can use the command (assuming that your program is called `classfractal1a.py`)

```
./classfractal1a.py > j; xmgrace -block j -bxy 1:2 -bxy 1:3
```

3. Start Random Walker on Circle

Initialize the lattice with all lattice sites being zero and only in the middle of the lattice is a seed with value one. Use a lattice size `LATSIZE=100`. We want next to implement that a new random walker starts randomly somewhere on a circle with midpoint in the middle of the lattice and with radius `RMAX+2`. For the DLA-program we will use `RMAX=2` but for now use `RMAX=20`. To check that you draw your random walker indeed equally likely on a circle, add to your program that you put 50 initial walkers on their starting point (not yet with random walk steps) and mark for each of these 50 starting points the lattice with the value 2. Make only one image of your lattice after these 50 markings on the lattice. To get a pdf-file of your frame use in header of program

```
import matplotlib.pyplot as plt
```

and for example

```
plt.imshow(lattice,interpolation='nearest')
plt.savefig('frame3.pdf')
```

Think about how you would program this task. When you have planned your strategy, you may go straight to not writing the program but instead you may copy the solution program `~kvollmay/share.dir/inclass2019.dir/classfractal3.py`
into your working directory. Read the program, to convince yourself what it does and run the program. Look at the resulting `frame3.pdf`.

4. DLA: Random Walk Copy `classfractal3.py` into another file, e.g. into `classfractal4.py`. Modify `classfractal4.py`. Take out of this program now the loop

over 50 starting points, instead start the random walker at **only one** point on the circle and use RMAX=2. Add to your program a loop over NSTEP=50 random walk steps. To check your program, assign to each site, which is visited by the random walker, the value 2. Print the lattice after this random walk and look at it. Hint: You may look at your 1dim- random walk program, or you may have a look at the random-walk program
~kvollmay/share.dir/inclass2019.dir/classfractal1a.py

5. DLA: Distance

5a. We work next on step IV of the DLA rules, for which you need to keep track of the distance r of your walker from the midpoint of the lattice. Add to your program of 4. the determination of r . (Hint: For $\sqrt{}$ in python use in the header, as already in the sample file, `import scipy as sp` and then for example $\sqrt{5}$ would be `sp.sqrt(5)`). To check your program print for every random walk step `x,y,LMID,r` and check by hand that you get the right distance from the midpoint of the lattice.

5b. Now replace the loop of 100 random steps with a while loop `while walkstop == 0:`. Set `walkstop = 0` before this while loop and set `walkstop = 1`, as soon as $r \geq 2 * R_{\mathsf{max}}$. Initialize R_{max} to $R_{\mathsf{max}} = 3.0$ and `sp.random.seed(11)`. Run your program and check that your stopping of the random walk works as intended.

6. Stick to Cluster

Next we will work on rule V, the sticking of a random walker particle to the cluster, if the random walker is next to a cluster cell. We use "von Neumann neighbors", which means a neighbor cell up,down,left or right.

6a. Next add to your program of 5b. that whenever the random walker is next (left, right, top, bottom) to a particle of the cluster then the random walk stops (`walkstop` update). Use the flow chart to decide where to add the necessary lines. If you have kept the `print(x,y,LATMID,r)` from 5a (and the same seed) then you can check that your program is working right.

6b. Now add to your program of 6a that you also have an integer variable `npart` which is initialized to be `npart=1` and gets increased by one whenever a particle sticks to the cluster. Also update `lattice` whenever a particle sticks. Whenever a particle sticks to the cluster, you also need to check if `RMAX` has grown and if so, then you need to update `RMAX`. Add this to your program.

7. Finish Program: Loop Over Particles

Now you are ready to finish your DLA program! Add to your program a while loop over particles. Condition for this while loop are both that the wanted number of cluster particles `NPARTMAX` has not yet been reached and that the radius for the start of the random walk fits into the lattice. Use the flow chart of class to decide where to add this `while`-loop. Use the constants `LATSIZE=100, NPARTMAX=50`. Comment out the printing of `(x,y,LATMID,r)`, but print the resulting lattice at the end (so after the particle-loop). In case you would like not to print the complete lattice, you may use the following commands

```
plt.imshow(lattice[int(LATMID-RMAX-2):int(LATMID+RMAX+2),\
  int(LATMID-RMAX-2):int(LATMID+RMAX+2)],interpolation='nearest')
plt.savefig('frame7.pdf')
```