In-Class Work: DLA (continued)

I will start class with reminding us of the flow chart for the DLA model.

1. Fractal Growth: Random Walk in Two Dimensions

1a. I will guide us through the solution of the random walk in two dimensions. You find the solution in
`~kvollmay/share.dir/inclass2023.dir/classfractal1a.py`
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/inclass2023.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
```

1b. Movie

I will also guide us through making a movie of the random walker on a lattice leaving a trail behind. We first look together at `sample_latticemovie.py` and then at the solution of the random walker on a lattice with a random walker which leaves a trail behind. The soluton is in
`~kvollmay/share.dir/inclass2023.dir/classfractal1b.py`

Next let's make a movie of your random walk. Define a lattice (`lattice`) of size 30x30 and initialize it for all sites equal to zero. Put your initial walker at site $x = 15$ and $y = 15$. We want to make a movie of the random walker where we mark on the lattice the current random walker site with the lattice value 2 and we mark any previously visited site with 1 (This is just for our fun.). To make a movie we first make an image for every random-walk step. (**So please use only `NSTEPS=40` random walk steps!**) To see how to make these pictures see the example
`~kvollmay/share.dir/pythonsamples.dir/sample_latticemovie.py` Once you have all pictures in `frame*` you can run the movie with

```
animate -delay 30 -pause 5 frame*png
```

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 (uniformly distributed) 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 (similar to above) in the header of program

```
import matplotlib.pyplot as plt
```
and for example
```
plt.imshow(lattice,interpolation='nearest')
plt.savefig('frame3.pdf')
```

4. DLA: Random Walk Recycle your program from the previous step and 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: Use your work above from step 1a of the fractal growth section.

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 numpy as np` and then for example $\sqrt{5}$ would be `np.sqrt(5)`). To check your program print for every random walk step `x,y,LATMID,r` and check by hand that you get the right distance from the midpoint of the lattice. Do `NSTEP=200` random walk steps.

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 $r \geq 2\,R_{\max}$. Initialize $R_{\max}$ to $R_{\max} = 3.0$ and `np.random.seed(11)`. Run your program and check that your stopping of the random walk works as intended.