Fractal Growth

1. Random Walk in Two Dimensions

1a. Please copy the following solution python program into your working directory:

```
cp ~kvollmay/share.dir/inclass.dir/classfractal1a.py .
```

This is 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 it prints $t, x(t), y(t)$. I will explain this program at the beginning of today's class. 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

Copy next the python program which makes a movie of a two dimensional random walker.

```
cp ~kvollmay/share.dir/inclass.dir/classfractal1b.py .
```

I will explain this program at the beginning of today's class. The random walker is initially starting at $x = 50$ and $y = 50$. The current random walker site is indicated with a lattice value 2 and any previously visited site has the lattice value 1. (This is just for our fun.). To make a movie we first make an image for every random-walk step. **(This is why we use only `NSTEPS=50` random walk steps!)** Once you have all pictures in `frame*` you can run the movie with
`animate -delay 10 -pause 5 frame*`

2. Fractal Growth Introduction

I will give you an introduction to **Cellular Automata**, **fractal growth**, and the **D**iffusion**L**imited**A**ggregation model.

In case you would like to read a bit more about fractal growth see Chapter 13 of the Gould & Tobochnik book. §13.3 introduces several fractal growth models, the DLA model is described on page 529 (page 23 of Ch 13).

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 `imshow` as above and use for example `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/inclass.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 have a look at the random-walk program from step 1 above `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.