In-Class Work: Fractal Growth

We are implementing the Diffusion Limited Aggregation (DLA) model by Witten and Sander [T.A. Witten, L.M. Sander, Phys. Rev. Lett **47**, 1400 (1981)]

4. DLA: Random Walk If you had finished in last class to include the 2dim random walk into the DLA program, you may continue with your working program. Otherwise, please start today's class with the solution program
`~kvollmay/share.dir/inclass2021.dir/classfractal4.py`
We will look at this program together at the beginning of class.

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. Copy `classfractal4.py` (or your already working program from last class) and add to the program the determination of r . Before you change the program, add this task to the flow chart, to know excactly where in your program you need to determine r. (Hint: For $\sqrt{}$ in python use in the header, as already in the sample file, `import numpy as sp` 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.

5b. Now replace the loop of 50 random steps with a while loop `while walkstop == 0:`. (Note == instead of = .) 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 `sp.random.seed(11)`. Run your program and check that your stopping of the random walk works as intended.

6. Stick to Cluster

For this step it is necessary that you comment out or delete the line we used in `classfractal4.py` for testing purposes only (it was in `classfractal4.py` the line 40), that is delete
`lattice[x,y]=2`
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 stopping of a too far out random walker 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')
```

8. Finished DLA program (if time)

So, now you have programmed the DLA model [T. A. Witten Jr, L. M. Sander, Phys. Rev. Lett. 47, 1400 (1981)]! Next you will learn how the DLA-cluster can be analyzed, namely you will measure the so called fractal dimension. You may compare your program with the following program:
`~kvollmay/share.dir/inclass2021.dir/classfractal8.py`

9. Fractal Dimension of DLA Cluster (if time)

9a. Please tell me when you get here. I will give you an intro to a defintion for the fractal dimension.

9b. Now lets get ready to analyze the pattern of the DLA model. You will determine the fractal dimension of one pattern using the method of checking squares of length `b`, as just described in class.

To avoid having to run the DLA program again and again, let us first prepare one pattern, which you then will analyze in 9c. Run the program
`~kvollmay/share.dir/inclass2021.dir/classfractal8.py`
This program makes the file `bigDLAcluster.dat` (and a nice pdf-file `frame8.pdf` just for fun). (Or if you have your own finished DLA program, have a look at the last few lines of `classfractal8.py` to see how to write the file `bigDLAcluster.dat`.) Ensure that you run the program for `LATSIZE=500` and for `NPARTMAX=3000`. This will take a while, but we have to do this only once, because for the analysis we use `bigDLAcluster.dat`.

9c. Now you need a program which reads in the 224x224 matrix from your file `bigDLAcluster.dat`. You may use for this task
`~kvollmay/share.dir/inclass2021.dir/classfractal9start.py`
To get the fractal dimension $d_{\rm f}$ we use the following relation.

$$\ln(N) = \ln(c) + d_{\rm f} * \ln(b) \qquad (1)$$

where N is the number of occupied sites, c is some constant and b is the length of your square for which you count the number of occupied sites. You see that Eq.(1) defines d_f and it tells us that if we plot $\ln(N)$ as a function of $\ln(b)$ then we should get a line with slope $d_{\rm f}$. So our task is to get N and b. Add to your program that you count the number of occupied sites N

for a lattice of lenght b, where you center your lattice of lenght b around the midpoint of your 224 x 224 lattice. Loop over the length of your lattice and print out $\ln(N)$ as a function of $\ln(b)$. Let's say you do
`classfractal9c.py > lnNoflnb.dat`
Hint: $\ln(N)$ is in python `np.log(N)`

9d. Next we fit a line to our data from 9c stored in file `lnNoflnb.dat`. For this we use gnuplot. So type in the command line "`gnuplot`". Then type "`plot "lnNoflnb.dat"`". Define a function $f(x)$ by typing "`f(x) = a*x + b`". Now fit your data within the xrange [2.0,4.2] to a line by typing "`fit [2.0:4.2] f(x) "lnNoflnb.dat" via a,b`". The resulting `a` is the fractal dimension d_{f}. You can look at the data and fit with "`plot "lnNoflnb.dat",f(x)`" Compare your fractal dimension with the expected value of 1.71