April 14, 2011 CAPS 491-11 Spring 2011

In-Class Work: Fractal Growth

1. DLA: Definition Read the hand-out and make a sketch of the model. Get me once you are ready.

2. DLA: Flow Chart Sketch the flow chart for the DLA program.

3. DLA: Circle and Random Walk Copy into your working directory
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/DLA3_sample.cc`
The program includes already the initialization of the lattice and the starting point of your random walker, which is on a circle around the midpoint of the lattice. Include in the program (at indicated place) the random walk being done by the particle. You may use the solution to last class' in-class work
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/rand1c.cc`
Update the lattice for each random walk , so that the random walk is visible as a red line (`lattice[x][y]=2;`). Look at your resulting lattice movie with
`executable | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2`

4. DLA: Distance

4a. We work next on step 2c of the 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 3. the determination of r . (Hint: For a vector (x, y) the length of the vector is $\sqrt{x^2 + y^2}$. In C++ $\sqrt{5}$ is `sqrt(5.0)` and 5^2 is `pow(5.0,2.0)` or `5.0*5.0`. Notice that `sqrt` and `pow` need double or float variables and do not work with `int` variables.) To check your program change the printing such that instead of printing the movie of the lattice it prints for every random walk step two numbers: the step and the distance r. Look at your result with `xgraph`.

4b. 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 $r \geq 2 * R_{\text{max}}$. Notice that R_{max} is initialized to $R_{\text{max}} = 3.0$ in the secion of initialization, the radius is set to `radius = Rmax + 2`, and `idummy = -6`. Run your program and check that your stopping of the random walk works as intended. //(Print no longer `step` but keep the print of `r`.)

Solutions to Random Walk In-Class Work:
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/rand1b.cc` etc.

5. Stick to Cluster (if time)

Next we will work on rule 2d, the sticking of a random walker particle to the cluster, if the random walker is next to a cluster cell (use von Neumann neighbors).
5a. Next add to your program of 4b. 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. To be able to check your program, we want to let the random walker to continue walking instead of stopping because of rule 2c. Therefore, just for the sake of checking your program for rule 2d, comment out the `walkstop=1` of rule2c. Add to the print command that you print not only `r`, but also `x` and `y`.

5b. Now add to your program of 5a 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. To check your program update the lattice such that the random walk is shown as a moving red site (i.e. `lattice[x][y]=0` before random walk step and `lattice[x][y]=2` after walk step). Watch the movie with

```
DLA5b.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

5c. 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.

Upcoming Deadlines:

- **April 15, 10am: Program (in share.dir and read permission)**
- **April 15, 10am: Results First Version** (if not yet)
- **April 19: Results Final Version**
- April 21: Abstract
- April 26: Second Paper 1st Version
- May 3: Second Paper Final Version
- May 3: Program Final Version
- April 28 & May 3: Symposium Talks

In-Class Work: Fractal Growth

5. Stick to Cluster

Next we will work on rule 2d, the sticking of a random walker particle to the cluster, if the random walker is nearest von Neumann neighbor to the cluster.

5a. Either use
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/DLA4b.cc`
or use your working program for 4b from last class. Add to the program 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. To be able to check your program, we want to let the random walker to continue walking instead of stopping because of rule 2c. Therefore, just for the sake of checking your program for rule 2d, comment out the `walkstop=1` of rule2c. Add to the print command that you print not only `r`, but also `x` and `y`.

5b. Now add to your program of 5a 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. To check your program update the lattice such that the random walk is shown as a moving red site (i.e. `lattice[x][y]=0` before random walk step and `lattice[x][y]=2` after walk step). Watch the movie with

```
DLA5b.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

where `-z 0 2` sets the colors to be blue for 0, white for 1, and red for 2.

5c. 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.

6. Finish Program: Loop Over Particles

Now you are ready to finish your DLA program! Put back in rule 2c (stopping of random walk when `r >= 2 Rmax`.) Add to your program the 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 `L=100`, `NPARTMAX=50`. Watch the movie with

```
DLA6.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

When you got your program working, please let me know, so that you can show your movie to the class.

7. DLA Pattern

Next we try to generate a larger cluster to see its fractal structure (and just because it is fun). Print the lattice no longer for every random walk step, but instead print the lattice only once after the while loop over particles. To be able to print only the middle part of the lattice instead of the complete lattice, have a look at the function `printsmalllattice` in
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/DLA4b.cc`
Set `L=500`,`NPARTMAX=3000` and `LPRINT=250`. Have fun with the resulting picture.

8. Fractal Dimension of DLA Cluster (if time)[3]

In the following you will analyze the pattern of the DLA model. You will determine the fractal dimension of one pattern. When you get to this, please get me, because I will give the class an introduction to the fraction dimension.

8a. To avoid having to run the DLA program again and again, let us first prepare one pattern, which you then will analyze in 8b. Use your program of 7. and run it with `L=500`, `NPARTMAX=10000`, `LPRINT=350` and
`DLA7.out` $>$ `bigDLAcluster.data`
to print the final pattern only once at the end of the program into the file "bigDLAcluster.data".

8b. Now write a program which reads in the 350x350 matrix from your file of 8a. To get the fractal dimension d_f we use the following relations.

$$\mathsf{ln}(N) = \mathsf{ln}(c) + d_\mathsf{f} * \mathsf{ln}(b) \tag{2}$$

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.(3) 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_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 350x350 lattice. Loop over the length of your lattice and print out $\mathsf{l}n(N)$ as a function of ln(b). Let's say you do
`DLA8b.out` $>$ `lnNoflnb.data`

8c. Next we fit a line to our data from 8b stored in file `lnNoflnb.data`. For this we use gnuplot. So type in the command line "`gnuplot`". Then type "`plot "lnNoflnb.data"`". Define a function $f(x)$ by typing "`f(x) = a*x + b`". Now fit your data within the xrange [2.0,4.5] to a line by typing "`fit [2.0:4.5] f(x) "lnNoflnb.data" via a,b`". The resulting a is the fractal dimension d_f. You can look at the data and fit with "`plot "lnNoflnb.data",f(x)`"

9. History of DLA Particles (if time)

To understand the shape of the DLA clusters, let us now indicate the history of the particles with their color: Print "5" for the first 500 particles, "6" for the next 500 particles, "7" for the following 500 particles etc.. Look at the final result with DynamicLattice.
Hint: Adjust your check if the random walker is next to a particle of the cluster.
You might recognize the resulting picture (:-)

[3]Ryan: You might want to first work on 9. and possibly also 10.

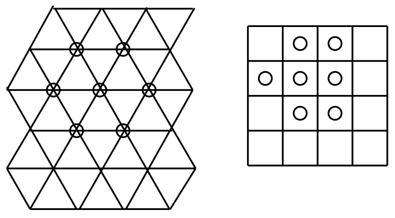

Figure 4

10. Snowflake(IF TIME)

Snwoflakes have a pattern similar to the pattern shown in Fig. 4a. We say that the molecules are on the corners of a triangular lattice. Each molecule is surrounded by 6 neighbors. We have so far always used the square lattice. It turns out that if you shift every second line of the triangular lattice of Fig. 4a , then Fig. 4a looks like a square lattice where the neighbors are right, left, up, down, NE and SE as shown in Fig.4b.

10 a. Use your working program of the inclasswork 8a. (so 500x500 lattice and `NPARTMAX = 10000` and print only once at the end a square lattice of size `LPRINT=350`). Change your program such that the sticking occurs according to the neighborhood of Fig. 4b. Run your program and look at the result with DynamicLattice.

10 b. To shift your square lattice back to the triangular lattice, first print out the resulting cluster

```
DLA10.out > cluster.data
```

then copy

˜ kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/snowflake_shift.cc

into your working directory and compile it. Then run it

```
snowflake_shift.out
```

The program prints x,y coordinates of each clusterparticle into the file "snowflake.data". Look at the resulting snowflake with

```
xgraph -m -nl -lx 180,330 -ly 60,210 snowflake.data
```

Upcoming Deadlines:

- **April 19 (Today): Results Final Version**
- April 21: Title & Abstract (I will make a symposium pamphlet with your abstracts.)
- April 26: Second Paper 1st Version
- May 3: Second Paper Final Version
- May 3: Program Final Version
- April 28 & May 3: Symposium Talks

In-Class Work: Fractal Growth

8. Fractal Dimension of DLA Cluster

In the following you will analyze the pattern of the DLA model. You will determine the fractal dimension of one pattern.

8a. To avoid having to run the DLA program again and again, let us first prepare one pattern, which you then will analyze in 8b. Either use
`~kvollmay/classes.dir/capstone_s2011.dir/fractal.dir/DLA7.cc`
or your finished and working DLA-program of last class. Run the program with `L=500`, `NPARTMAX=10000, LPRINT=350` and
`DLA7.out > bigDLAcluster.data`
to print the final pattern only once at the end of the program into the file "bigDLAcluster.data". You may check (and enjoy) the resulting picture:
`DynamicLattice -nx 350 -ny 350 -matrix < bigDLAcluster.data`

8b. Now write a program which reads in the 350x350 matrix from your file of 8a. To get the fractal dimension d_f we use the following relations.

$$\ln(N) = \ln(c) + d_\mathsf{f} * \ln(b) \tag{3}$$

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.(3) 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_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 350x350 lattice. Loop over the length of your lattice and print out $\ln(N)$ as a function of $\ln(b)$. Let's say you do
`DLA8b.out > lnNoflnb.data`

8c. Next we fit a line to our data from 8b stored in file `lnNoflnb.data`. For this we use gnuplot. So type in the command line "`gnuplot`". Then type "`plot "lnNoflnb.data"`". Define a function $f(x)$ by typing "`f(x) = a*x + b`". Now fit your data within the xrange [2.0,4.5] to a line by typing "`fit [2.0:4.5] f(x) "lnNoflnb.data" via a,b`". The resulting a is the fractal dimension d_f. You can look at the data and fit with "`plot "lnNoflnb.data",f(x)`"

11. Try any variation on the DLA model (e.g. stick a particle only with probability P_{stick}) and "measure" how the fractal dimension d_f depends on the change (e.g. P_{stick}).

Upcoming Deadlines:

- **Today, April 21:** Title & Abstract (I will make a symposium pamphlet with your abstracts and a poster with your titles.)
- April 26: Second Paper 1$^{\text{st}}$ Version
- May 3: Second Paper Final Version
- May 3: Program Final Version
- April 28 & May 3: Symposium Talks