1 August

1.1 First week - 12.aug

This week I had the first meeting with my supervisor and co-supervisor. We talked about the outlines of my thesis. I am suppose to start with SiO_2 in a glass state, and make spherical, ellipsoidal and cylindrical pores. These pores will be filled with H_2O. I will do measurements on the water inside the pores, and see how the measurements depend on the curvature of the pore walls. I am not sure what kind of measurements I should do.
Camilla and Anders Hafreager helped me and Filip to get startet, by showing us how we should use the python-wrapper for the fortran code.

This weeks product: Used base_code to generate a β-cristobalite crystal, warm it up,save states and create xyz-files. The xyz-files are best visualized in VMD.

1.2 Second week - 19.aug

I created the C++ file cut_out_sphere.cpp. Generally, this program takes the mts0 directory where a state is saved and loads the atoms. Next, it removes the atoms within the wanted pore and saves the new state in a directory defined by the user. cut_out_sphere.cpp is not a good name for the program, because the user can choose to cut out cylindrical and ellipsoidal pores as well as the spherical pores. The name should be changed.

This weeks product: The C++ file cut_out_sphere.cpp.

1.3 Third week - 26.aug

Anders Hafreager gave Filip and I the task of making a program that makes a histogram of the number of voxels that contains at least one particle, given different voxel sizes. We first start with the smallest desired voxel size, v_{min}, and create a grid that divides our system into voxels of size v_{min}. This grid is represented by a, for now, empty three-dimensional matrix, M. The next step is to fill this matrix. The idea was to let each element of the matrix be the number of particles contained by the voxel with the same coordinates as the element. This is easily found by iterating all the particles, and for each particle find the coordinates of the voxel it belongs to and increase the corresponding element in the matrix with one. Say the particles position is given by $\vec{r} = [x, y, z]$, then the corresponding voxel has the coordinates $v\vec{o}x = [\lfloor \frac{x}{v_{min}} \rfloor, \lfloor \frac{y}{v_{min}} \rfloor, \lfloor \frac{z}{v_{min}} \rfloor]$. The first element of the histogram is made by counting the number of

occupied voxels. Now we want to increase the voxel size. For a given voxel size, we use the original matrix M to make a new similar matrix M_{new}. M_{new} is made by merging as many neigboring elements in M as we want.

This weeks product: Filip and I created the C++ file voxel_counter.cpp. I have read a little bit differential geoimetry of curves and surfaces.

2

September

2.1 Fourth week - 2.sept

This week we were handed another task from Anders Hafreager (we have sort of become his staff now, it seems). The task was to translate a existing program written in Fortran to C++, and write the code in a parallal manner. The program is called distance_to_atom.cpp.
The idea behind the code:
We divide the system into a given number of voxels, and represent the voxels by a three-dimensional matrix, **M** which will store the distances from the center of the voxels to the nearest atoms. Each process iterates all the atoms, and each atom "let" the neighboring voxels know it exists, and if the distance from the atom to voxel is smaller than the distance the voxel already had stored, the new distance is stored in the matrix **M**. By neighboring voxels I mean all the voxels within a distance, max_distance, from the atom. This distance is given by the user. This means that voxels far away from any atoms may not register any atoms. In those cases the corresponding elements in **M** is just what the matrix was initialized with. We initialize the matrix, **M**, with a distance equal to half the system size, L_2. Since we assume periodic boundaries we know that half the system size is the largest distance we will find (maybe an idea to initialize with max_distance instead?)
As of wednesday, each process have to load all the atoms. This is a lot of data, and completely unnecessary.

On monday I had a meeting with my supervisor, where I asked him to help me set some deadlines. The next thing he wanted me to do, was to passivate my system. This should be done within 2-3 weeks.

This weeks product:
Filip and I created the parallel program distance_to_atom.cpp. The master log was created.

2.2 Fifth week - 9.sept

2.3 Sixth week - 16.sept