**Nanocluster Molecular Dynamics (NCMD) Project**

Overview

Molecular Dynamics is a common computational technique used to understand how materials and chemical systems evolve over time using classical physics assumptions. In this project, you will explore how nanocluster systems evolve over short periods of time.

Project Instructions

(You may need to hop to the “explanation of files” section if you are confused about the directions!)

We are interested in seeing how nanoparticle systems evolve over time at different temperatures. There are some modifications we will make to the code, as described below:

1. Start by opening “MD\_Create.ipynb”.
   1. Change cell dimension to 250 in the first line of the second cell (first cell under “Create Species to add to scene”
   2. We will not create gold atoms or pairs for this simulation. Delete “Au\_tet”, “Au\_pair”, and “Au” from the second cell.
   3. Delete Au\_tet.set\_cell(cell) and repeat for Au\_pair and Au.
   4. In the first cell under “Create Scene and set simulation cell”:
      1. Delete atoms3 and atoms. Change the name of atoms2 to atoms.
      2. In atoms0, atoms1, and atoms2, half the third number (we are telling the scene maker to put the clusters closer together by doing this).
   5. See Explanation of files to visualize produced cell.
2. Next, open “MD\_Simulate.ipynb”
   1. Change the name of the file that you are reading in if you named it something different in the previous step.
   2. Similar to the example described in the “Explanation of files” section, we will run 4 simulations at different temperatures. We will do this by reading in the file, setting the calculator, and **the only thing we will change is the temperature and length in the second to last cell.** The length is length in iterations, so we will be running a simulation that is time\_step\*length long, in femptoseconds (10e-15 seconds).
   3. Set the simulation length to at least 10,000.
   4. Pick any 3-4 temperatures to run the simulation at. Remember, this is in Kelvin. In the example, we show that we start to melt our particles around 473K and the particles are completely melted at 1000K, so it may be in your interest to explore different temperature ranges.
   5. **The simulations will take a long time.** Be prepared to leave your computer alone for a bit while they are running. After the simulation, you should check the trajectory file in the final cell. Then you should reset the notebook and repeat for a different temperature. **Do not just change the temperature and click run, you will just add on to the previous simulation and your results will not make sense. Additionally, you need to change the section “fn = ‘test\_simulation.traj’ every time, otherwise it will write over your data!**
3. Next, open “MD\_Analyze.ipynb”
   1. The only thing you should have to change are the file names in the first cell. You may want to change the name of the data you are reading in as well to describe what temperature you are looking at, but you may be able to just keep track of this on your own.

Questions to answer:

1. In the beginning of the program, we discussed how atoms bond. Do you think the potential we are using here makes sense? You may need to revisit some readings on the Lennard Jones potential and what it is best used for.
2. What do you observe as a function of temperature in the MD\_analyze.ipynb file? How does this align with what we discussed about kinetics?
   1. What do the radial distances tell you about the population?
   2. What do the cluster sizes tell you about the population?
3. Do you think that this is an accurate way to understand how nanoparticle systems evolve? Why or why not.
4. See if you can figure out how long we have simulated the system for. Were you surprised by the length?
5. Bonus- can you figure out how to read in frames at different steps during a single simulation and compare them? We may want to track the evolution of a single system over time to determine if there is constant particle growth, or “burst-like” particle growth. What do you observe when doing this?

Explanation of files

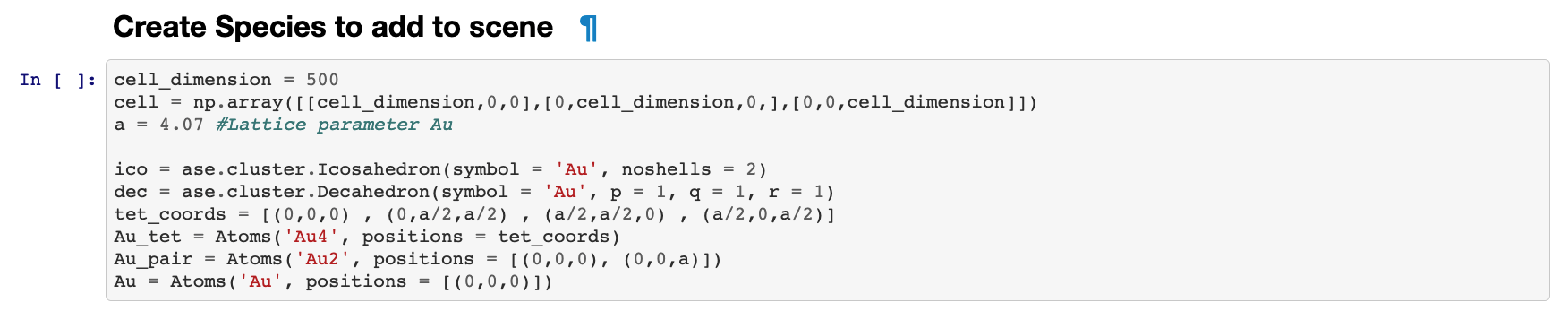
All of the files can be accessed on the github repository, “NCMD”

<https://github.com/abruefach/NCMD>

The readme.md file describes the simulation details and program dependencies. You will need to download Atomic Simulation Environment, ASAP, Numpy, Scipy, and Matplotlib as described on their websites.

There are three “.ipynb” files present. “ipynb” stands for “iPython Notebook”, which is Jupyer’s former name. These files should all be opened through Jupyter notebook or Jupyter Lab.

1. **MD\_Create.ipynb**
   1. This file contains the functions to create the simulation scene. Here, we use ASE to design several different clusters that will be present in the simulation cell. This is done in the “Create Species to add to scene” section.

In this section, you can change the size of the structures as described:

***Icosahedron*** size can be increased by changing “noshells = \_\_” to a number greater than 2.

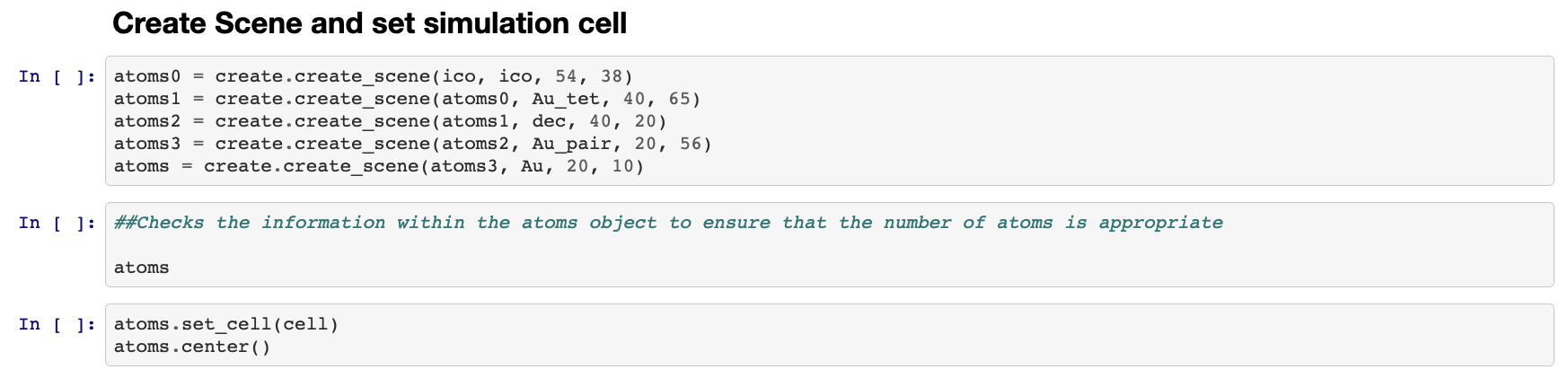
***Decahedron*** size can be changed by increasing the “p = \_\_” or “q = \_\_” to a value greater than 1.

***Au\_tet*** contains 4 Au atoms coordinated in a tetrahedron shape. Do not change this.

***Au\_pair*** contains 2 Au atoms coordinated in a pair. Do not change this.

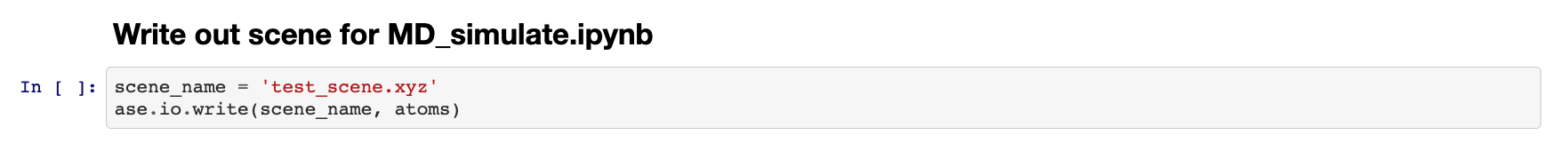
***Au*** contains 1 Au atom. Do not change this.

* 1. The next important step is the “Create Scene and set simulation cell”. Here, we allocate a certain number of ico, Au, dec, Au\_pair, and Au species to the cell at minimum distances from each other.



In the first line, the atoms0 object will consist of 38 icosahedral structures that are at least 54 Angstroms (5.4 nm) apart from each other. Then, the atoms1 object will add 65 Au\_tet objects to the 38 icosahedral structures. It will make sure everything is at least 40 Angstroms apart. This trend is continued until the final “atoms” object is made. This can be viewed by replacing the next cell (replace “atoms”) with **ase.visualize.view(atoms, viewer = ‘x3d’)**. If the viewer = ‘x3d’ does not work, delete this part. You should be able to see the structures. We will modify this section in the project, because the simulation cell will be a bit too large for this project.

* 1. Finally, we must write out our scene once we have created a scene we think will be interesting to study. Here, we do this by naming the scene and typing “ase.io.write(\_\_\_\_)”

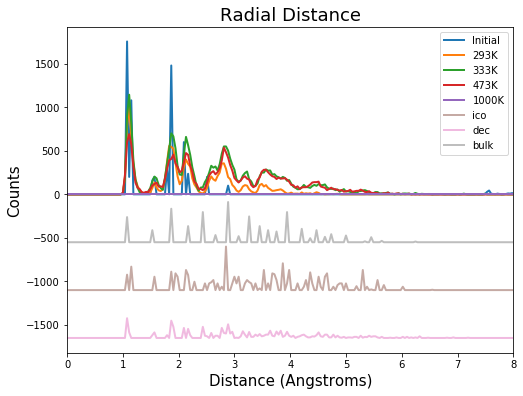


1. **MD\_Simulate.ipynb**
   1. Here, we simply import the scene, set a calculator to determine the forces between the atoms, and let the code run. You may recognize the Lennard Jones potential that we set for gold from the beginning of the program - this potential describes the interaction between atoms as a function of distance. This allows us to understand how the forces that the atoms face will impact how the atoms move over time.
   2. We will change some parameters in the **simulate.simulate\_asap function**. The most interesting thing to change here to see results faster will be the temperature. We will get into this later.
   3. In the last box, we read in the trajectory file, which is a sequence of images that make up a video of the simulation. We only read a select few frames of the complete video because it will slow our computers down if we read a lot in at once.

\*\*\*\* IMPORTANT NOTE \*\*\*\*

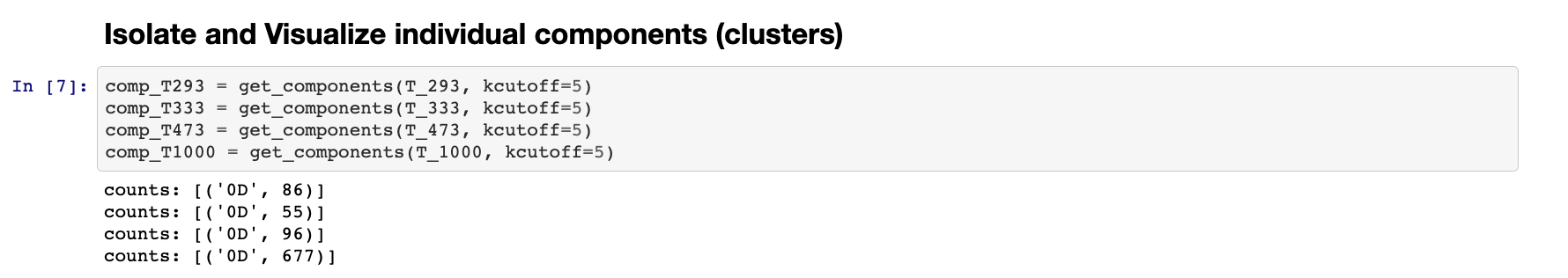
If you plan to run multiple simulations, you must read in the original scene again. If you do not, you will just run a different simulation on the final arrangement of the last simulation you ran. You might not want to do this if you are comparing how temperature impacts the coalescence of particles, where you will just be adding on to particles that have already coalesced!

1. **MD\_Analyze.ipynb**
   1. You will use this code to analyze how the structure of the system is changing over time. In this program, we reviewed how atoms tend to pack in specific structures. We can visualize how the distances between the atoms and between the particles change over time in this notebook. First, we will read in the data that we have simulated.
   2. The first section is “Get Radial Distances”. This can tell us 1. How the distance between atoms changes depending on the dataset and 2. How far apart the clusters generally are. Let’s take a look at the demo output (see below)

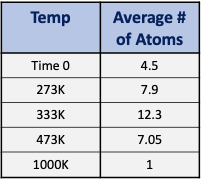
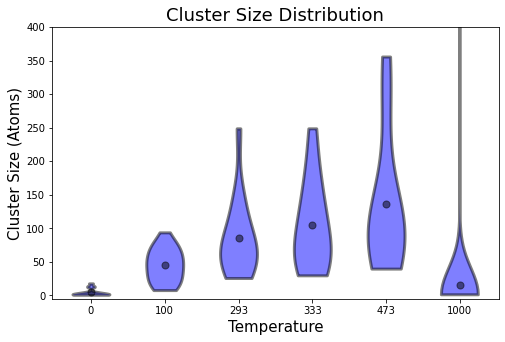


Here, we normalize the distances (R) by the equilibrium distance (R0) to help us understand how far apart atoms are. The purple, green, and yellow lines show us how some of the initial structures that we put in look like, while the red, blue, and black lines will show us how our atoms are arranged after heading to different temperatures. We can see that as the temperature increases, the peaks between 3 and 4 “R/R0” increase as well. Keep in mind, this is 3-4x the equilibrium distance, indicating that the particle sizes may be getting larger. We look further into this in the next section!

* 1. In the next section, we plot the cluster size distribution using violin plots. The first step is to isolate the atoms, which we do in the section where we “get\_components”, as shown below. The “kcutoff” describes how far apart atoms in the same cluster should be.



* 1. Next, we get the cluster sizes from each of the components. Then we put this information into violin plots. These plots take in the cluster sizes at each temperature and estimate the distribution using Kernel Density Estimation. Below are the sample plots.



As we see, it does seem that the cluster size generally increases from 100K to 473K. At T = 1000K, we have completely melted all of our particles!