Molecular Dynamics Comprehensive Writeup

Katie Ream*

Updated as of May 9, 2025

Abstract

This paper gives a comprehensive overview for crafting, executing, and studying molecular dynamics (MD) based simulations using the LAMMPS software and OVITO visualization tool. An accompanying python notebook is given to export the OVITO data and extract track width data and vacancy density data as a function of ion depth into a sample. The simulations crafted currently involve irradiating various paleodetector samples such as quartz and olivine with an inbound ion, typically gold at increasing energies. The process to execute a successful simulation requires many steps, including downloading and configuring LAMMPS, creating an input script, importing necessary potential and crystal objects, executing the run over a timescale of picoseconds, loading the data into OVITO, ensuring the run executed successfully, and finally exporting the selected data from OVITO for further analysis in the python notebook provided. Each step is broken down further in the body of this writeup and example working LAMMPS scripts are provided. There are also exercises embedded into this to develop a more fundamental understanding of how to craft a successful LAMMPS simulation. The LAMMPS documentation page can be found at the following webpage and is quite comprehensive.

1 Downloading LAMMPS

LAMMPS is an online software designed to run molecular dynamics simulations. It is well documented and is an open source code. We have it downloaded on the local Linux machine in the upstairs Spitz group office, however it can be beneficial to also have it available on your local platform. To download the most current version click the following link. LAMMPS is compatible with macOS, Linux, and Windows based platforms. Once you click the download button tied to the "LAMMPS Stable Release" there will be a tar file to unpack. Run the following command in the terminal in the directory where the tar file is located:

*kmream@umich.edu

```
tar -xzvf file.tar.gz
```

**** this section will be finished later - for now use the local computer/windows computer Emilie has installed LAMMPS on or reach out to Katie Ream kmream03@gmail.com for assistance as needed.

2 Creating a LAMMPS Script

There are many crucial components in order to craft a successful LAMMPS script. You must populate the script with the desired particles and ions, whether that is done manually within the script regime or via importing an external structure, you must add in the desired potentials to govern the physics of the simulation, you must specify the world boundaries of the simulation, you must define how long it will run for, and finally you need to export the data to a tangible document which can be loaded into OVITO. Let's break these sub-components down one by one.

2.1 Defining the World Boundaries

Before you can populate the simulation, the boundaries must be specified along with a couple of other crucial factors.

```
units metal
atom_style atomic
boundary p p p
read_data file_input.lmp
```

The following four lines specify the types of atoms being studied, what the units of the simulation are, the boundary condition type at the edges, and the sample of particles externally imported. For the types of simulations being run for the UMich paleodetector effort I recommend using the read data command. I do not recommend changing units from metal; for atom_style, I recommend using either atomic or charge - charge is the exact same as atomic except it takes into account charges of the particles within the simulation. The boundary conditions have four potential options - p (periodic), f (non-periodic and fixed), s (non-periodic and shrink-wrapped), and m (non-periodic and shrink-wrapped with a minimum value). The boundaries must be explicitly defined in all three directions or an error will be thrown. Based on multiple conversations with different researchers and professors who have used LAMMPS frequently it is recommended to define the axis on which the ion being shot in as m and the axes perpendicular to this motion both as p. For example, if we define a gold ion being shot in at a quartz sample along the x-axis, our command would read as `boundary m p p`.

In the next section I will discuss how to craft a crystal sample using the software CrystalMaker, but for now assume you have a working crystal that is ready to import - call it `file_input.lmp` for now. This is what is fed into the read_data command. At the

top of the `file_input.lmp` file we manually specify the volume of the world - it looks as follows.

```
-100.0 100.0 xlo xhi
-10.0 10.0 ylo yhi
-20.0 20.0 zlo zhi
```

It is recommended to try and manually "shrink-wrap" these specified boundaries as closely as possible around the imported crystal sample about the planes which are specified with a `p` and extend the boundary length about the plane which is specified with an `m`. In this example we once again use `boundary m p p` for consistency.

2.2 Adding Potentials

This section you can spend all day and night on fighting over whether this potential is better to use than that potential. I won't lie, lots of this section involves trial and error to figure out what produces a reasonable result. There are notes left on the script I am leaving about potentials we tried that did and did not work.

The documentation for adding in different potentials look like the following. We note that for all the different types of particles included they need to be specified by a type given by an integer starting with 1. More of this will be touched on in the next section when we import a crystal sample. Take for this example hydrogen is 1, helium is 2, and lithium is 3.

```
pair_style hybrid/overlay potential1 (parameters) potential2(parameters) etc...
pair_coeff 1 1 potential1 parameters
pair_coeff 1 2 potential1 parameters
pair_coeff 1 3 potential1 parameters
pair_coeff 2 2 potential1 parameters
pair_coeff 2 3 potential1 parameters
pair_coeff 3 3 potential1 parameters
```

The types of parameters required depend based on the type of potential added. Here is an example set of potentials which I currently use in my script, where type 1 is silicon, type 2 is oxygen, and type 3 is gold.

```
pair_style hybrid/overlay zbl 0.1 1.5 tersoff lj/cut 10.0
pair_coeff * * tersoff 2007_SiO.tersoff Si O NULL
pair_coeff 1 1 zbl 14.0 14.0
pair_coeff 1 2 zbl 14.0 8.0
pair_coeff 2 2 zbl 8.0 8.0
pair_coeff 1 3 zbl 14.0 79.0
pair_coeff 2 3 zbl 8.0 79.0
pair_coeff 3 3 zbl 79.0 79.0
```

```
pair_coeff 3 1 lj/cut 0.01 3.4 10.0
pair_coeff 3 2 lj/cut 0.01 3.4 10.0
pair_coeff 3 3 lj/cut 0.01 3.4 10.0
```

Here I have imported an interatomic potential file downloaded from a potential repository specifically crafted and tested for crystalline quartz and have added in addition ZBL and leonard jones potential. We see at the top of the line pair_style that any potentials I wish to add as a pair_coeff must be added to the pair_style line with their respective parameters and all delimited by a space-bar.

2.3 Configuring Particle Dynamic Evolutions

Now that the particles have been imported and the potentials define we need to impart velocity on the inbound ion and set it to run for a specific period of time. The following is how this is done - note beginning a line in LAMMPS with a # symbol comments out the following text in that line. In this particular example the inbound particle has an initial energy of 1 keV.

```
# create the inbound particle - ensure the particle's position resides
# within the previously set world boundaries
group incoming type 3
variable incoming_x_pos equal 0.0
variable incoming_y_pos equal 8.0
variable incoming_z_pos equal 0.0
create_atoms 3 single ${incoming_x_pos} ${incoming_y_pos} ${incoming_z_pos}

# give the inbound particle an initial energy kick
run 0 # this tells the simulation to begin evolving at timestep 0
group incoming type 3
variable eV equal 1000
variable eVtoJoules equal 1.6e-19
variable KEnorm equal ${eV}*1000*${eVtoJoules}
variable atom_mass equal 196.96655
variable velocity equal sqrt(2*${KEnorm}/${atom_mass})
variable velocityAps equal ${velocity}*1e12*0.01
velocity incoming set 0.0 -${velocityAps} 0.0  units box

# allow the inbound ion to start moving
timestep 0.001
run 100
```

We note that the unit of the timestep given by the line `timestep 0.001` is dependent what the command `units` is defined as at the top of the input LAMMPS script; if it remains as `units metal` then `timestep 1.0` corresponds with 1 picosecond, however it

is strongly recommended to start at something much smaller like 1 femtosecond or even 0.1 femtoseconds. Oftentimes errors arise in the calculations done by LAMMPS because the system evolves too quickly from timestep to timestep, so reducing this value by an order of magnitude and allowing it to run for more timesteps typically resolves this issue.

2.4 Writing to an Output File

Finally the simulation you've crafted is running smoothly without any errors and you're ready to output the data! Here is a line to write at the bottom of the script that writes all the particle information at each timestep into a .dump file. Note these files can get quite large for larger simulations - we currently have a 5 TB hard-drive that you can export this data to to free up storage on the local Linux machine.

```
# write the data to .dump file after every 10 timesteps
dump 1 all custom 10 /desired/path/output_file.dump id type x y z vx vy vz fx fy fz
dump_modify 1 sort id

# display certain information in the terminal after every 10 timesteps
thermo 10
thermo_style custom step time temp press pe ke etotal fmax
```

The first time you choose to output something to a dump using the command dump you write the number 1 afterwards. If following later on in the script you wish to output specific information to a separate output file you would write instead `dump 2 etc...`. In the line `dump 1 all custom 10 etc...`, all specifies all particles in the simulation are saved and the data is saved to the output after every timesteps which have passed. You may choose to omit or add additional information to the dump beyond what I chose to output here specifically. The thermo_style outputs those characteristics of the simulation as well after every 10 timesteps - this doesn't modify the simulation in any manner by adding or omitting information to display in the terminal. Now the data is ready to load into OVITO for further visualization and analysis. This process is described in section 4.

3 Crafting a Crystal Sample

There are many ways to go about creating a crystal sample which we will shoot our inbound particle with - we explain 2 potential options below. Depending on the size of the crystal needed and how specific of a structure desired will modify whether you use CrystalMaker or the connected python script, and both ways are described in this section as follows.

3.1 Crafting Crystals with CrystalMaker

Navigate to the following link to install the software, which is quite easy to do. A key is needed to access the software - reach out to Kai Sun (kaisun@umich.edu) for this

key. Once this key has been obtained open the CrystalMaker library and type into the search bar for the particular crystal desired. Let's start with pure gold, Au. Open the gold option upon searching or navigating to the Ore sub-folder on the bottom left of the CrystalViewer platform (there is both a CrystalMaker and a CrystalViewer, yes) and you should see a unit cell of gold displayed with some information about the crystal structure. We need to make a larger sample than simply one unit cell to give our inbound ion enough room to propagate and leave a residual track. To increase the size of the crystal, navigate to the range button at the top of the page situated between centre and direction. Continue to keep adding cells by clicking the add cells button or hitting the drop down arrow by cell count button for the x, y, and z directions and click add cells. Keep doing this until the cell has reached a significant size - for starting simulations getting the crystal on the order of 3 x 3 x 3 nanometers should suffice. Note that CrystalMaker is sometimes a glitchy software and as the crystal gets larger and larger the program may lag, so do not click the add cells button 1000 times because it will overload - instead click the add cells button after the cells have been added each time individually. The process of creating a large enough cell should take between 1 minute - 15 minutes depending on how large the sample needs to be.

3.2 Crafting Crystals with Python

For crystals that are on the order of tens of nm x tens of nm x tens of nm, unfortunately CrystalMaker is not equipped to handle the amount of data and atoms at this scale. In this case, pivot to using the following python script. You will need to input the symbols associated with the unit-cell structure for that crystal, the space-group number (this can be looked up with the associated python package imported), the parameters of the cell, and defining the coordinates for a basis cell. This information can all be found online, and the associated documentation for the main python package involved in crafting these molecules can be found here. This script will output to an .lmp file which writes to the same format as below; if you use this approach the following step needs to be slightly modified (instead of outputting the letter abbreviation of that atom-type, it outputs the Z number associated).

Now the crystal has reached a sufficient size and it needs to be exported. Navigate to File → Export Data → XYZ → Visible Structures. Save the sample with the desired name and location and click save. This export is a type .XYZ file - rename this instead as a .txt file and override any warnings associated with doing so. The text file should look like the following.

```
908

Au 20.393000 -8.157200 -8.157200
Au 20.393000 -6.117900 -6.117900
Au 20.393000 -8.157200 -4.078600
Au 20.393000 -6.117900 -2.039300
```

```
.....
Au 4.078600 0.000000 4.078600
Au 0.000000 4.078600 4.078600
Au 4.078600 4.078600 4.078600
```

This tells us there are 908 atoms total, the element the atom is, and the position of the particle. This needs to be converted using the following python script to change the `Au` to 1. All atoms in LAMMPS need to have an associated "type" with it, starting at 1 and increasing by integer value. These type numbers correspond directly to the numbers in the potentials subsection. To easily convert these numbers, use the following python script.

At this point your output .lmp file should have the following format:

```
908

1 1 20.393000 -8.157200 -8.157200
1 2 20.393000 -6.117900 -6.117900
1 3 20.393000 -8.157200 -4.078600
1 4 20.393000 -6.117900 -2.039300
.....
1 906 4.078600 0.000000 4.078600
1 907 0.000000 4.078600 4.078600
1 908 4.078600 4.078600 4.078600
```

3.3 Turning the Crystal Rectangular (Optional, if Needed)

As we can see, the particle positions have not been affected and all particles now have a corresponding type. It is important to ensure that the shape of the sample is rectangular or cubic such that the incident ions hit the plane perpendicularly. If the sample exported from CrystalMaker is already perpendicular then you can continue on; if not, use the following python script to crop the sample such that it is rectangular and proceed as follows. This script also allows you to add charges to the atoms - at least a 0 charge is necessary in order for LAMMPS to properly read in the script. Modify these charges to be whatever value as needed.

3.4 Importing the Final Crystal into a Tangible Format for LAMMPS

There are still a few more steps which need to be taken such that the input crystal doesn't give an error. This opening looks as follows.

```
# Base file created to be used for any crystal imported by Katie Ream, @kmream.

100 atoms #adjust this number to account for all atoms present
1 atom type
```

```
 -100.000000000000 100.000000000000 xlo xhi
 -100.000000000000 100.000000000000 ylo yhi
 -100.000000000000 100.000000000000 zlo zhi

Masses

 1 196.00000 # Au

Atoms # charge

#begin pasting your output from the change_letter_to_type.ipynb python notebook
```

Replace the number of atoms with the correct number at the top of your .lmp output and paste the rest of the modified particles starting at the line with the comment dictating the correct place to do so. One crucial portion of LAMMPS is ensuring the world boundaries at the top of this file are large enough to account for the sample you have loaded in. Additionally it is important for our efforts, as described previously in section 2, to shrink-wrap the boundaries perpendicular to the motion of the inbound ion as close as possible, but not going into the crystal body itself - if this happens there will be lots of errors arising.

3.5 Executing the LAMMPS Command in Terminal

At this point you now know how to define a base LAMMPS script with an imported crystal structure. LAMMPS is convenient additionally because it can be parallelized across multiple cores and threads, meaning the runtime is faster. This is especially helpful for simulations with large crystals and long simulation lifetimes. To initialize a simulation, navigate to the directory where your LAMMPS script and components which need to be imported are located and run the following command. Recall that the LAMMPS script itself is a .txt file, but the imported crystal is a .lmp file and the simulation writes out to a .dump file.

```
mpiexec --np (number of cores) lmp -in name_of_files.txt
```

You will know the simulation initialized running successfully if the following is printed out in the terminal:

```
Per MPI rank memory allocation (min/avg/max) = 22.09 | 36.76 | 94.83 Mbytes
    Step    Time    Temp    Press   PotEng  KinEng   TotEng   Fmax
    0       0       100     1000    -1000   100      1000     1000
    10      0.001   110     1100    -900    200      2000     2000
    .....
```

and the following message is printed when the simulation is complete containing summary statistics about the simulation overall.

```
Total # of neighbors = 20000
Ave neighs/atom = 0.2542
Neighbor list builds = 100
Dangerous builds = 0
Total wall time: 1:00:00
```

3.6 Exercises for Configuring LAMMPS Simulations

Congrats! You have just completed your first successful LAMMPS run. Here are some exercises to complete to get comfortable with the above steps.

- Create and export a crystal of pure gold from CrystalMaker, load it into the various python notebooks, and verify you can add and/or modify the charges associated with the atoms. Verify the gold atoms are of the correct type and that the IDs are assigned properly.
- Create and export a crystal of olivine from CrystalMaker, load it into the various python notebooks, and verify you can add and/or modify the charges associated with the atoms. Verify the four types of atoms are of the correct respective types (keep track of these) and that the IDs are assigned properly. *Bonus: modify the turn_crystal_rectangular.ipynb notebook such that the different types of atoms are color-coded and confirm this result matches what was exported from CrystalMaker.*
- Create a LAMMPS script using the gold crafted from the first exercise using ONLY coulomb potential and shooting a gold ion on the gold crystal. Let it run for 1000 timesteps at 0.1 femtosecond increments *(remember - what are the units associated with* `units metal`*? Look it up on the LAMMPS documentation page if needed).* For a singular potential you should use `pair_style potential1 (parameters)` and omit the `hybrid/overlay` portion since we are not overlaying or combining potentials. Note - if the output in the terminal shows the simulation is approaching infinity, the timestep is most likely too large. Try taking it down an order of magnitude and running again.
- Create a LAMMPS script using the gold crafted from the first exercise using Leonard-Jones and coulomb potential and shooting a gold ion on the gold crystal. Let it run for 1000 timesteps at 0.1 femtosecond increments. Refer back to subsection 2.2 to recall how to combine multiple potentials if needed.

4 Visualizing the Simulations Using OVITO

4.1 Loading Output Files into OVITO

OVITO is an open source visualization tool which is designed very well to study molecular dynamics evolutions, and is compatible with macOS, windows, and Linux based systems. OVITO is already downloaded on the local Linux machine, but to download

it on your local computer click the following link to install it. To launch OVITO on the Linux computer, navigate to your respective directory, where there is a folder on desktop called OVITO and enter the following in the terminal to open the platform (if opening on local mac, just double click the OVITO icon).

```
./bin/ovito
```

Drag the .dump file into the empty OVITO world. Once it is finished uploading the simulation at timestep 0 should be displayed, and depending on how long the simulation ran and how often the information was dumped out indicates how many steps are visualized in OVITO. First click the play button and watch the completed simulation and ensure it behaves generally correct. This is a great way to ensure that the gold ion is heading in the correct direction and actually makes proper contact with the paleodetector sample.

4.2 Applying Modifiers to the Simulation

There are many potential modifiers we can add to our simulations and many ways in which these can be added. In the future you may find you need to analyze the output data in a different manner, but for the following I will go over the modifiers I use on a daily basis. **Note with these modifiers that the order in which they are applied matters - swapping the ordering can result in quite weird results!**

The first modifier I like to add is **Compute Property** in order to output the timestep information when I export the data. To do this, click the arrow by *Add modification* and scroll down until you see *Compute Property* under the *Modification* subcategory. Leave it such that the modifier is operating on Particles, and type whatever name is desired under the *Output property* section. Then in the Expression field type **Timestep** and hit enter - now there should be a new column present that calculates the timestep for each particle throughout the entire simulation.

The next modifier I add is called **Slice**, which allows us to cutoff portions of the quartz sample that we don't care about, especially towards the edges of our sample (since the edges of our input crystal don't experience physics that matches the real world, despite our efforts to shrink-wrap the boundaries as tightly as possible to try and mitigate these effects). The *Slice* modifier is also located under the *Modification* subcategory, and I recommend adding 3 slices in each planar direction. The slab width and distance can be adjusted so that just the edges of the crystal sample are chopped off. These will need to be adjusted for each unique crystal sample loaded in, but are generally quick to find and add.

Watching the damage evolve in the entire quartz is nice, but it is a lot more beneficial to highlight only the displaced atoms and watch the damage evolve in this manner. In order to delete all stationary particles and just leave displaced atoms, add the following modifiers *** in the correct order *** (this in particular applies to quartz, so you may have

to modify for different crystal structures as needed). *Note: the order displayed shows how the modifiers need to be added in OVITO itself, with the top modifier representing the last step applied and bottom ones applied first.*

- **Delete Selected, under Modification**
- **Slice, under Modification for X-axis example**
  1. Distance: 5
  2. Normal (x): 1
  3. Normal (y): 0
  4. Normal (z): 0
  5. Slab width: 100
- **Expression select, under Selection**
  1. Operate on: Particles
  2. `(ParticleType == 1 && Coordination == 4) || (ParticleType == 2 && Coordination == 2)`
- **Coordination Analysis Modifier, under Analysis**
  1. Cutoff radius: 2
  2. Number of histogram bins: 200
- **Compute Property, under Modification**
  1. Operate on: Particles
  2. Output property: Timestep
  3. Central expression: Timestep

If these were added correctly, you should see all particles deleted at the first timestep and particles should begin to appear on screen as damage propagates and evolves overtime. This allows us to clearly see which particles have been disturbed and roughly what areas present vacancies.

4.3 Exercises for Becoming Familiar with OVITO

Congrats! You have now visualized your first successful LAMMPS run. Here are some exercises to complete to get comfortable with the above steps.

- Experiment with the other compute property elements - what else can you track as the simulation evolves?

- Try finding a modifier that can modifying the colors corresponding to different atoms in the simulation.
- Try playing around with the expression selection modifier and only show the type 1 atoms? Type 2 atoms?
- Try deleting all atoms except for the inbound gold ion. Now make the gold ion displayed as larger (hint - you don't need a modifier for this. Try exploring the *Visual elements* portion of choices.
- Try visualizing the forces the atoms within the crystal feel (hint - you don't need a modifier for this. Try exploring the *Visual elements* portion of choices.
- *** Bonus : try loading in the different simulations you ran in the previous exercises and see if you can confirm that adding or removing different potentials influences how the simulation evolves. Try pushing these potentials and their parameters to extreme limits to confirm the potentials are actually making an impact.

4.4 Exporting OVITO Data

Now that you've selected the desired particles and have the results displayed, we need to export this data for further analysis on the vacancy densities and to calculate the approximate track width created as a function of depth into the crystal sample. Navigate to the final timestep (or other desired timestep during the simulation). Click **File** → **Export File** and name it to whatever desired, ensuring the *Files of type:* is selected to *XYZ(*)*. Make sure the following particle properties to export are selected and the option *Current frame only* at the top is selected before hitting the OK button in the bottom right corner.

- Particle Identifier
- Particle Type
- Position.X
- Position.Y
- Position.Z
- Coordination
- Timestep

Once this has been exported, open the file. It should look like the following:

```
100000
Lattice="3000.0 0.0 0.0 0.0 212.0 0.0 0.0 0.0 212.0" Origin="-1000.0 -100.0 -101.0"
Properties=id:I:1:species:S:1:pos:R:3:n_Neighb:I:1:Timestep:R:1
```

```
1 2 -50.0 -100.0 -50.0 0.0
2 2 -50.0 -100.0 -45.0 0.0
...
```

Delete the first 3 lines of this script and replace it with the following, so the output file now should look like the following:

```
id type x y z coodination step
1 2 -50.0 -100.0 -50.0 0.0
2 2 -50.0 -100.0 -45.0 0.0
...
```

Now the input script can be loaded into the final python script for further analysis.

5 Analyzing Vacancy Density Data

The following python notebook contains all post-processing progress currently for the MD efforts. The goal of this notebook is to calculate the number of vacancies were created, find the vacancy density vs depth into the sample, and calculate the approximate track width as a depth into the sample. Previously when we irradiated our olivine sample with 15 MeV gold ions we crafted the following plot - the goal with these LAMMPS simulations is to recreate these types of plots, confirm with real data, and then predict track widths for future paleodetector samples and efforts.

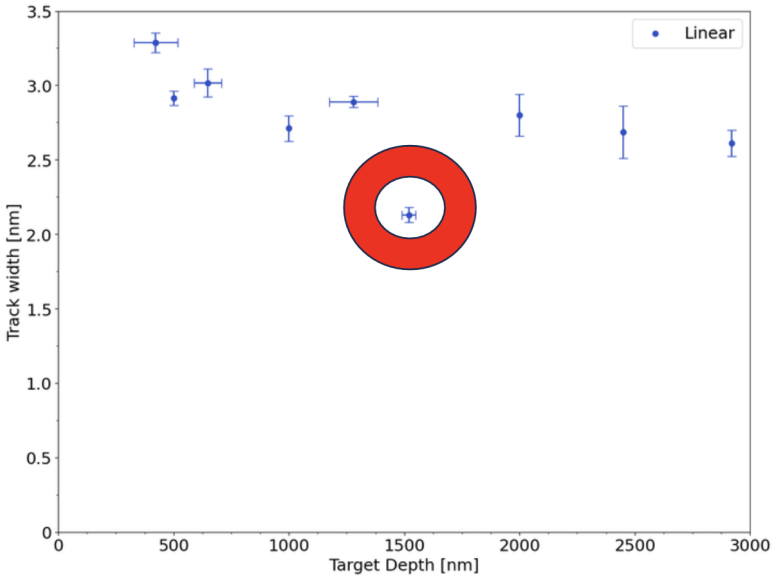

Figure 1: Resulting data from May 2024 involving multiple trials of irradiating an acquired sample of Olivine at different depths into the sample with 15 MeV gold ions. Note the sample circled is because it deviates from the expected pattern, which requires more data from Kai to confirm the reason for such a disagreement.

When you load in sample data into this notebook, here are the three important plots and results you will recover:

- Visualize the vacancies with the appropriate scales to see track length.
- Calculate and visualize the vacancy density vs depth relationship as we go further into the sample.
- Calculate and visualize the track width vs depth relationship as we go further into the sample.

Here are some example plots showcasing each item listed above:

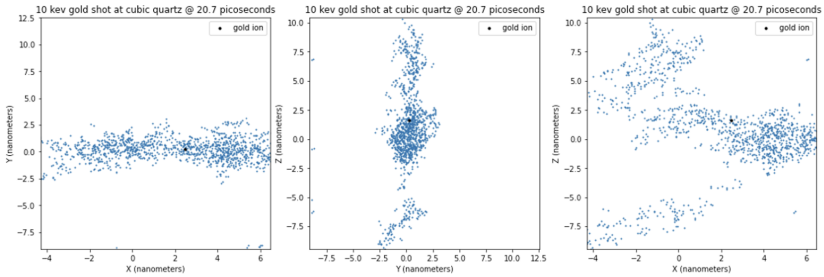

Figure 2: Outputted 3D data from OVITO displayed in the xy, yz, and xz 2D planes with the gold ion highlighted in black.

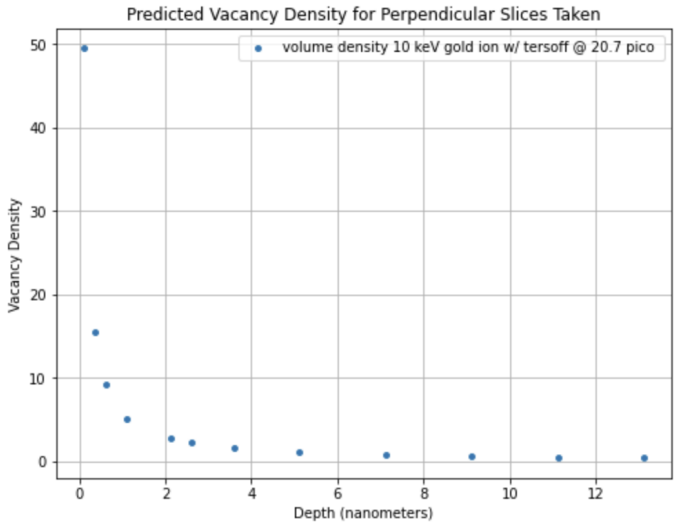

Figure 3: Calculated vacancy density values as a function of depth into the quartz sample, found by taking volume slices perpendicular to the direction of the inbound gold ion and count the number of vacancies in the defined region.

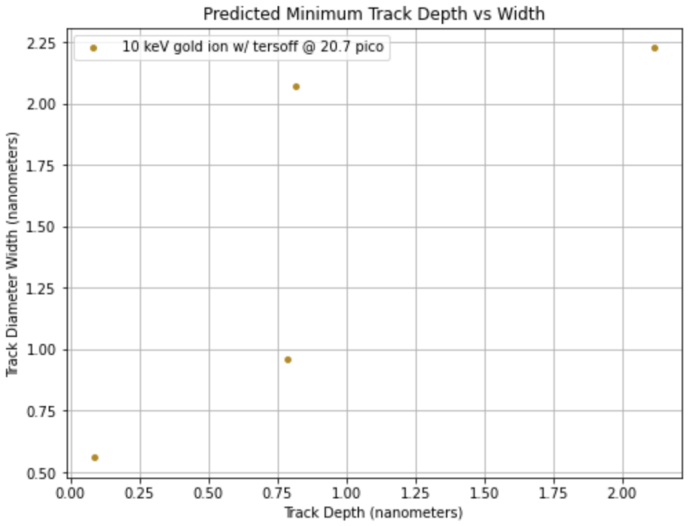

Figure 4: Calculated track width values as a function of depth into the quartz sample, found by performing a complete 360 degree rotation along the plane which the ion was shot in to find the atoms which have the shortest radius at that particular point (with a specified small tolerance value), then multiplied by 2 to get the complete width.

6 Where Progress is Currently

The current stage of progress as of this most recent edit is we are trying to increase the energy of the inbound gold ion in the MD simulations up to 100 keV. We want 100 keV because we can reasonably justify actually irradiating a sample of quartz on North Campus with 100 keV gold and we can compare the experimental results with the MD simulation predictions also cross-referenced with SRIM outputs. The issues we currently face is that for each run we've completed (2 keV, 10 keV, and 20 keV) we are counting more vacancies than what is predicted - sometimes by a whole order of magnitude - however the track length matches similarly to what is predicted by SRIM. Additionally we need to increase the size of our input quartz crystal, however LAMMPS does not like working with a very large sample because it is too computationally expensive. Currently it seems like our limit might be around 2,000,000 input atoms and reaching up to 25nm x 40nm x 40nm, which is *potentially* big enough length wise for 100 keV ions but is unclear for the time being. As such we may have to pivot our approach to potentials or something else - honestly I'm not sure what this solution will look like at the time being since there are very few MD simulations which have reached our energy threshold with the intent of studying ion bombardment, if any.

6.1 Things We've Tried in the Past that have Failed

If you add in various potentials the simulation goes crazy, there are two solutions. First, try commenting out all of the potentials and run a dummy trial to ensure no moment occurs - this way you know that the movement/behavior was caused by the potentials and not the boundaries. Then start uncommenting the potentials one at a time and watch them evolve individually - sometimes individual potentials go awire. Then, start adding potentials back two at a time. Repeat until you discover the wrong combination - this potential may have the incorrect parameters or might not be the correct potential to add in that particular combination. **Example: DO NOT add coulombic potential to a tersoff interatomic potential file, but DO add it to a vashistha interatomic potential file.**

For the love of god make sure that the box boundaries are LARGER than the specified input crystal. If this is not true, either the quartz will look insane because it was built over itself when the simulation compiled or an error will arise in the terminal saying atoms were lost/couldn't be computed.

7 Concluding Remarks

There is a folder given to you with all of the respective python notebooks and a README.txt file that walks through the correct order and reason to use each file for brevity and clarity-sake. If there are any questions, please don't hesitate to reach out! Additionally these simulations can sometimes be aggravating because you don't know why something is going awry - don't fret! Just remember to comment out literally everything that could cause the simulation to go crazy, uncomment one line at a time, and boom there's your problem. Within the larger paleo group as well there is a larger effort being devoted to MD so there are resources available. Additionally there is a UM professor of Chemical Engineering named Rebecca Lindsey (email here: rklinds@umich.edu that has been super helpful, as well as the MD effort at University of Maryland with whom we've been collaborating with. Don't ever feel held back to reach out to these people or find your own who have lots of experience with MD and LAMMPS! As a final note, remember we're in a sense pioneering what the limits are for MD and ion irradiation so have fun with it! - Katie