# Module 2: LAMMPS Project - dislocation mobility in aluminum

### Project Brief

In this project, you will use MD simulation to examine the dislocation core structure in aluminum, and investigate the motion of the dislocation under an applied strain rate. This information can be used to determine input parameters to higher length scale models, such as discrete dislocation dynamics. This work will also help you prepare for the final project, where you will investigate the changes in dislocation behavior from the introduction of *solutes* into aluminum.

### Deliverables

You will produce a short report documenting your findings. Your report should contain a separate section for each of the tasks listed below, and provide the explicit deliverables requested for each task indicated by **[Report]** below. For the input scripts below, simply cut-and-paste your input script into your report. Your report should be formatted as a single pdf document comprising your report. You may wish to write your report in latex and convert using `pdflatex`, or in markdown and convert using `pandoc report.text --to latex --out report.pdf`; alternatively, you can put together a clearly formatted jupyter notebook. (`module load pandoc` and `module load texlive` to have the most up-to-date versions of each).

You should submit your report by creating a subdirectory called
`/class/mse404pla/sp26/<your_net_id>/Project2`
and copying your PDF into that directory by **11:59pm on 20 April 2025.** ***Late submissions will not be accepted*****; let me know in advance if you will have difficulty with completion..**

### Instructions

The LAMMPS Manual and commands list will be vital to your success. You are going to construct your own input files for running your `LAMMPS` calculations below, and will need to work through what commands are necessary. Broadly speaking, there are three parts to this project:

1. Relaxing an edge dislocation core ($T = 0$ structure).
2. Thermalizing at finite temperature ($T = 300$K structure).
3. Applying strain, identifying dislocation motion.

### Part 1. Relaxing an edge dislocation.

In anticipation of future work, you will use the Mendelev Al-Mg EAM potential: NIST repository information. This potential gives a lattice constant of 4.0453Å, and elastic constants of $C_{11} = 110.18$ GPa, $C_{12} = 61.37$ GPa, and $C_{44} = 32.56$ GPa. I have already created an initial dislocation structure for a perfect edge dislocation ($\frac{a}{2}[110](1\bar{1}1)$) using anisotropic elasticity. In the directory `/class/mse404pla/LAMMPS/Project/`, you will find a file called `Al-disloc.xyz`. This file is an "XYZ" file of an edge dislocation: the Burgers vector is $\frac{a}{2}[110]$, which points in the $x$ direction, the threading vector is $\frac{a}{2}[1\bar{1}2]$, which points in the $z$ direction (and will be periodic, with length $\sqrt{3/2}a \approx 4.954$Å), while the $y$ axis points perpendicular to the slip plane, in the $[\bar{1}11]$ direction. This geometry was generated for a cylindrical slab of radius 100Å. You can visualize this

geometry in `Ovito`; because of the displacement field, it looks like Pacman with an underbite: this is the Volterra construction of a perfect dislocation.

Also in the `/class/mse404pla/LAMMPS/Project/` is a python script called `xyz_to_lammps.py`. This is a very simple script that can read that XYZ file, and create a LAMMPS "data" file: the format is explained in the documentation for the read_data command. The script will read the XYZ file, and construct a data file with two regions (tagged by atom type): an "inner region" of all atoms inside the `r_in` radius, and an "outer region" that goes out to `r_out`; atoms outside of `r_out` will be removed. The two `boxsize` parameters specify the $x$ and $y$ dimensions; the $z$ dimension will be determined by the threading vector. You should choose `boxsize` to be larger than `r_out`, and ensure that $r_{\text{out}} - r_{\text{in}}$ is greater than, but close to, the cutoff of the potential (7.5Å). You are going to consider different sizes of `r_in`, but in the beginning to test your scripts, you will likely want to use a "small" value, such as 20Å.

Next, you will *construct input script*. You will start with the partially completed input script, `Al_dislocate_relax_template.in`. You will complete the input script by adding in missing commands; the input script also contains notes about what is missing.

- **[Report]** Complete the Initialization block by adding lines for `units`, `dimension`, `boundary`, and `atom_style`. *Be careful! What `units` and `atom_style` does the EAM potential require?* For `boundary`, you want to be "shrink-wrapped" in the $x$ and $y$ plane, but periodic along the $z$ direction.
- **[Report]** In the Atom Definition block, we will read our initial positions from a data file. The command `read_data` specifies the name of the file; you will need to construct this file using the python script `xyz_to_lammps.py`. Choose an inner radius of at least 20Å, and construct your cell. Use `Ovito` to check that your geometry looks correct: include a visualization of your geometry, color coded according to the "centrosymmetry" parameter (you will need to "add modification" corresponding to the centrosymmetry parameter, then "add modification" for color coding, according to that parameter. Choose a range of 0 to 6 to plot (though you may want to play with different values of the upper end) to see your initial dislocation core.
- **[Report]** In the Force Fields block, you will need to add the `pair_style` command (corresponding to `eam/fs`) and `pair_coeff` command. The `pair_coeff` command will look like this:

  ```
  pair_coeff * * [filename for potential] [chemistry for type 1] [chemi
  ```

  where there is one chemistry entry for each *type* of atom. Your data file has *two types*: type 1 is the "inner" and type 2 is the "outer," but you want to treat both of these as Al, so construct your `pair_coeff` line to correspond with that.
- **[Report]** In the Groups block, define two groups using the `group` command: identify one as the "mobile" atoms and the other as the "frozen" atoms. In your case, you can use the atom "type" to do this
- In the Settings block, there are three `compute`'s already defined: one for the centrosymmetry parameter, one for the potential energy of each atom, and finally one for the stress on each

atom. These get used in the dumps: one will be the relaxation "trajectory", while the other is a sequence of CFG files containing the compute information for each atom. `Ovito` will be able to parse this data and help us visualize the dislocation geometry.

- **[Report]** In the Relaxation block, you will need to add a `fix` command so that there is *no force* on the atoms that you have chosen to freeze. Add the appropriate command.
- Finally, you will want to make sure that the last line (`write_data`) is creating the filename that you want it to create: this relaxed geometry will be the input for the thermalization step!

**Next:** Relaxation. Run the relaxation to determine the equilibrium core geometry for this potential, and then visualize it in `Ovito`.

- **[Report]** How many steps did your relaxation require? What is the change in energy from the initial configuration to the final configuration?
- **[Report]** Take screen shots of the initial and final geometry of your dislocation by visualizing the centrosymmetry parameter, and estimate how far the partials have split based on the width of the stacking fault. **Note:** stacking fault is a locally "HCP"-like region in the dislocation core, which has a non-zero centrosymmetry value. You will need to adjust your ranges accordingly.
- **[Report]** Continue your visualization of the dislocation core by plotting the energy per atom, and the important stress fields: *xx* (atomStress1), *yy* (atomStress2), and *xy* (atomStress6). You may also want to investigate the other stress fields as well: *zz* is non-zero due to Poisson effects, while *yz* and *xz* stresses come from the *screw* components of the partials.

**Part 2. Thermalizing at finite temperature ($T = 300$K structure).**

Now that you have a relaxed dislocation geometry (corresponding to $T = 0$), you will perform a thermalization run to produce a dislocation geometry at a finite temperature (300K), and see if there are changes in the dislocation core geometry, as well as test the best way to visualize the position of the dislocation.

To do this, you will need to *construct an input script*. To do this, make a copy of your completed relaxation input script; next, you will edit it to (1) read the relaxed dislocation core, (2) thermostat the inner "mobile" atoms while keeping the outer atoms fixed, and (3) run molecular dynamics.

- **[Report]** In the Atom Definition block, you will need to change `read_data` to read the new dat file corresponding to your relaxed geometry. In addition, you will want to `replicate` it along the *z* direction, so that you dynamics are realistic. Choose to repeat at least 4 times (this corresponds to a length of almost 20Å, but you may want to go larger).
- **[Report]** In you Settings block, change the `dump` commands to (1) write every 100 steps instead of every step, and (2) write to different files than you used for your relaxation: you don't want to overwrite what you just did!
- **[Report]** Change the title of "Relaxation" block to "Dynamics" (this is purely for your reference). Then, you will need to (1) add a `timestep` command for 2fs timesteps, (2) add `velocity` commands and (3) new `fix` commands corresponding to the thermostat.

For your moving group, create velocities corresponding to *twice* the target temperature (so use 600K instead of 300K) so that thermalization will happen more rapidly. Be sure to zero out momenta and rotational torques. For the frozen group, set their velocities to zero.

Keep your `fix` to remove the forces on the frozen atoms, but add a new `fix` corresponding to an `nvt` (constant number, volume, and temperature) thermostat for 300K. A reasonable choice for the damping parameter is 1ps, and a good value for the drag is 1.0.

- **[Report]** Change the `thermo` command to output every 100 steps, change your `thermo_style` command to

  ```
  thermo_style custom step time cpu cpuremain pxx pyy pzz pxy pe temp
  ```

  remove the `min_style` and `minimize` command. Finally, add the command

  ```
  run 2000
  ```

  in order to run for 2000 timesteps.

- Finally, you will want to make sure that the last line (`write_data`) is creating a *new* file, corresponding to your thermalized dislocation. **Don't overwrite your relaxed dislocation geometry!**

**Next:** Molecular dynamics. Run your dynamics; you will get a copy of the output to the screen in the file `log.lammps` (so that you can see how your temperature stabilizes), and you can visualize your dislocation geometry in `Ovito`.

- **[Report]** Capture the temperature vs. time-step data in your `log.lammps` file, and make a plot of the temperature vs. time in `MATLAB`. **Note:** the temperature is based on the average kinetic energy of *all the atoms in your system*. Since a fraction of your atoms are frozen, you will want to scale the temperature so that it represents the temperature of your mobile atoms. Does your system look well converged in temperature? If not, consider rerunning for a longer time.

- **[Report]** In `Ovito`, we want to use the centrosymmetry parameter to not just color code the atoms corresponding to the dislocation, but ideally to *remove* the atoms that are not of interest. To do this, you will need to add a few modifications:

  - First, add an "Expression select" modification: this will allow you to *select* atoms that satisfy a given expression. In this case, we want to remove all of the atoms whose centrosymmetry parameter (`c_csym`) are below a threshold; as an initial guess, you may try 1, but feel free to experiment with other values. This expression would be `c_csym < 1`. You can *also* select the atoms in the outer boundary; these are atoms where `(Position.X^2 + Position.Y^2)` is greater than your cutoff distance squared; you can use `||` for logical or.
  - Having selected atoms, you will need to add "Delete selected particles" modification. This will remove them from the visualization.
  - **Note:** the *order* of the modification is important. You need to have the "delete" modification listed *above* the "expression select" modification. You can use the up and down arrows on the right side to make sure they're in the right order.

- Now, take a screen shot of your dislocation core!
- For funsies, you can also make a *movie* of your dislocation. Select the "render" settings (its an icon of a small picture on the top right, between the "pencil" icon and the "stack" icon). You can then select "complete animation", "save to file" (and choose the file). Clicked on the viewport you want to render (e.g., "Perspective" viewport), then you can click the "Render Active Viewport" button at the top right to create your movie.

**Part 3. Applying strain, identifying dislocation motion.**

Now with your thermalized dislocation, you will want to apply strain to cause it to move. Your goal is to identify the necessary strain to move the dislocation by monitoring the stress in the system and verifying the motion of the dislocation via visualization.

To do this, you will need to *construct an input script*. To do this, make a copy of your completed thermalization input script; next, you will edit it to (1) read the thermalized dislocation core, (2) apply increasing strain on the outer atoms, and (3) run molecular dynamics.

- **[Report]** In the Atom Definition block, you will need to change `read_data` to read the new dat file corresponding to your thermalized geometry. You will want to turn off any `replication` you did in the previous step.
- **[Report]** In the Settings block, change your `dump` commands to write to different files than you did for thermalization. You don't want to overwrite what you just did!
- **[Report]** In your Dynamics block, you should (1) reduce your time step to 1fs to increase the accuracy, (2) *remove* the velocity commands—your dat file already has velocities for all of the atoms, so you don't need to replace those velocities, and (3) add a new `fix` that will correspond to the shear on your frozen atoms.

  *Assuming that your frozen group is called `frozen`*, the following commands will allow you to add in a velocity to the frozen atoms:

```
### new velocities, corresponding to shear
variable   vperAng equal 0.001
variable   VX atom ${vperAng}*(y-ly/2)
variable   VY atom 0
variable   VZ atom 0
fix        3 frozen move variable NULL NULL NULL v_VX v_VY v_VZ
```

  The label "3" for fix assumes that this is the third fix you have; if you are using a different labeling scheme, change this to correspond to your choice. This will apply a constant velocity to the atoms on the outside, where the $x$ velocity is proportional the distance from slip plane; this is an *xy* shear strain. The value of `0.001` corresponds to a shear strain rate of 0.001/ps, you can change this to lower or higher values, but it is a reasonable starting place.
- You do not need to write out the data unless you wish to restart the calculation to continue.

**Next:** Molecular dynamics. Run your dynamics; you will get a copy of the output to the screen in the file `log.lammps` (so that you can see how your stress changes with time, which corresponds

to the change in strain), and you can visualize your dislocation geometry in `Ovito`.

- **[Report]** Capture the *xy* stress vs. time-step data in your `log.lammps` file, and make a plot of stress vs. strain in `MATLAB`. **Note:** You will need to convert your time step into a strain by multiplying the time by the corresponding strain rate. You stress is output in "bar" (`metal` units); convert to MPa. Make sure your units make sense! Identify the point where the stress stops increasing with strain: this is an indication of the dislocation moving.
- **[Report]** Visualize your dislocation trajectory. Using the same methods you did for thermalization, watch to see the dislocation glide in the positive *x* direction. Does this correspond with where you saw the stress drop in your stress/strain plot? Make screen shots of the dislocation at the time step before motion and after motion.
- Estimate your Peierls stress at finite temperature from your single simulation: does your result seem reasonable to you? You may want to change the strain rate (which may necessitate changes to the trajectory length, too), or how often you output data, to further investigate.

**Part 4. Finite size effects**

By this point, you should have a reasonable set of input scripts that take you from relaxation to thermalization, to the application of stress. These have now been tested for the "small" simulation size you initially chose. You should repeat your calculation with a cell that is at least 40Å in radius and 40Å along the threading vector.

- **[Report]** How does your relaxed dislocation core change with the larger radius?
- **[Report]** Does your thermalization run require changes to produce a thermalized trajectory? Provide evidence for your conclusion.
- **[Report]** How is your Peierls stress estimate changed with the larger cell? Provide appropriate plots to back up your conclusion.