NE451
Simulation Methods

LAMMPs Tutorial 1: Introduction to running a basic LAMMPs simulation on the c1 cluster for beginners

Agnes Katai
University of Waterloo
Canada

UNIVERSITY OF WATERLOO

Training Overview

Training Overview 1
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Description:
This hands-on lab introduces users to running classical molecular dynamics simulations with **LAMMPS**, a C++ based MD engine. The goal is to build, run, and analyze a basic structure relaxation on the BCC Li lattice.

Learning Objectives:

- Understand how to initialize atomic configurations in different ways
- Learn to define simulation parameters in a LAMMPS input script
- Practice running a simulation on an HPC cluster using Slurm
- Gain experience with analyzing simulation outputs

Logistics:

- **Type:** Self-paced hands-on lab
- **Audience:** Beginners to molecular dynamics
- **Prerequisite:** Review **Intro to HPC** training
- **Next Step:** Complete before advanced LAMMPS sessions (**diffusion**, **RDF**, **thermal conductivity**)

Basics of Running LAMMPS

Hands-on Workflow:

1. Initialize the atomic configuration
   - **Method A:** Write .data file using Python ASE
   - **Method B:** Convert .cif file from Materials Project
   - **Method C:** Build directly in input script
2. Define simulation parameters in `in.*` file
3. Run `lmp` executable with your input
4. Post-process with Python or OVITO

Input and Output Files

Hands-on Exercise: Identify Files

- **Input Files:**
  - Structure file (`*.data`)
  - Potential file (`*.meam`, `*.pair_style`)
  - Input script (`in.*`)
- **Output Files:**
  - `log.lammps` (thermo data, run info)
  - `*.dump` (atomic trajectory snapshots)

Task 1: Generate a Structure with ASE

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Activity:

1. Create and activate a Python virtual environment

```
$ python -m venv md-env
$ source md-env/bin/activate
```

2. Install ASE

```
$ pip install ase
```

3. Write a `bcc_Li.data` file with ASE (see ASE docs)

Task 1: Generate a Structure with ASE (Conda)

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Activity:

1. Create and activate a new Conda environment

```
$ conda create -n md-env python=3.10 -y
$ conda activate md-env
```

2. Install ASE

```
$ conda install -c conda-forge ase
```

3. Write a `bcc_Li.data` file with ASE (see ASE docs)

Task 2: Use Materials Project

Activity:

1. Go to Materials Project
2. Search for Lithium, download the `.cif` file
3. Convert to `bcc_Li.data` with OVITO or ASE

Task 3: Build Structure in LAMMPS Input Script

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix


```
lattice     bcc 3.439
region      box block 0 1 0 1 0 1 units lattice
create_box 1 box
create_atoms 1 box
```

Challenge: Replicate the system in 2x2x2 cells.

Task 4: Write Input Script for Relaxation

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Hands-on: Copy this block into `in.meam`

```
# ---- Relax BCC Li ----
units metal
boundary p p p
atom_style atomic
read_data bcc_Li.data
pair_style meam
pair_coeff * * library.meam Li Li.meam Li
min_style cg
minimize 1e-25 1e-25 5000 10000
```

Task 5: Prepare a Slurm Script


```
#!/bin/bash
#SBATCH --job-name=Li_relax
#SBATCH --time=0-00:30
#SBATCH --cpus-per-task=6
#SBATCH --mem=6G
module load class-simulations
/scic/app/cpu/lammps/bin/lmp -in in.meam
```

Command: Submit with `sbatch lammps_run.sh`

Task 6: Check the Results

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Expected log.lammps output:

```
Total energy (eV) = -3.299...
Number of atoms   = 2
Lattice constant  = 3.487
Cohesive energy   = -1.649 eV
All done!
```

Practice: Visualize `dump.lammpstrj` in OVITO.

Basics of Running LAMMPs

Workflow for Running a LAMMPs Simulation:

1. Initialize the atomic configuration you wish to simulate
   - Use **ASE** package in Python to write a structure file in the lammps data file format. Ideal for custom configurations, i.e., 85-15% Ni-Cr alloy, add H adsorbate on Al(111) surface, etc.
   - Download .cif file from **The Materials Project** website and convert to lammps data file format using OVITO, OpenBabel, etc. Ideal for simple and/or well-known structures, i.e., pure crystals
   - Initialize simulation box and populate with atoms using lammps commands in input script (no .data file required)
2. Define simulation parameters in input script (in.* file)
3. Run lammps command
4. Post-processing with Python scripts

Basics of Running LAMMPs - Input Files

Input Files:

- Structure File: ***.data** (required **unless** you are initializing the atomic configuration in your lammps input script). Contains:
  - Total number of atoms and atom types
  - Box dimensions
  - Masses of atom types: <atom_type> <mass>
  - Atomic Coordinates: <atom_id> <atom_type> <x> <y> <z>

```
# LAMMPS data file written by Ovito 3.10.6

2 atoms
1 atom types

0.0 3.43931245 xlo xhi
0.0 3.43931245 ylo yhi
0.0 3.43931245 zlo zhi

Masses

1 6.941  # Li

Atoms  # atom_style atomic

1 1 0.0 0.0 0.0
2 1 1.719656225 1.719656225 1.719656225
```

Basics of Running LAMMPs - Input Files (Cont.)

- Potential File: ***.pair_style** (required). Contains:
  - Pairwise force field coefficients required to calculate the potential energy surface and forces on the atoms

```
rc = 3.6
delr = 0.1
augt1 = 0
erose_form = 2
ialloy = 2

zbl(1,1) = 0
nn2(1,1) = 1
attrac(1,1) = 0.05
repuls(1,1) = 0.05
Cmin(1,1,1) = 0.16
Cmax(1,1,1) = 2.8
Ec(1,1) = 1.65
re(1,1) = 3.02
```

  - You may find potential files you need from the **Interatomic Potentials Repository**: **https://www.ctcms.nist.gov/potentials/**
- Input Script: **in.*** (required). Contains:
  - Commands needed to define simulation settings and customize thermodynamic outputs
  - All commands are described in detail in the official **LAMMPs manual**: **https://docs.lammps.org/Commands.html**

Basics of Running LAMMPs - Output Files

Output Files:

- Log File: **log.lammps** (required). As soon as the simulation begins, lammps starts logging information to this file including:
  - Simulations settings used
  - Thermodynamic properties of the system
  - Performance, total run time, etc.
- Dump File: ***.dump** (generally required to see how your system behaves with time)
  - Stores snapshots of atomic coordinates and other selected quantities every custom number of timesteps during the simulation.
  - These files can be visualized using **OVITO (recommended)** to see changes in the atomic configuration with time.

1. Creating Structure File - ASE

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

- Create a Python virtual environment for your project to keep its dependencies separate from other projects. Activate the environment before installing any packages to ensure they are contained within this isolated space.

```
$ python -m venv path/to/virtual/environment
$ source <path/to/virtual/environment>/bin/activate
```

- Install ASE package

```
$ pip install ase
```

- Write lammps .data file to the path of your choice. See ASE documentation for classes and methods helpful for building molecules and alloys: **https://ase-lib.org/ase/build/build.html**

1. Creating Structure File - ASE (Cont.)

Training Overview

Step 1: Preparing a Simulation

Step 2: Building a Structure

Step 3: Running the Simulation

Appendix

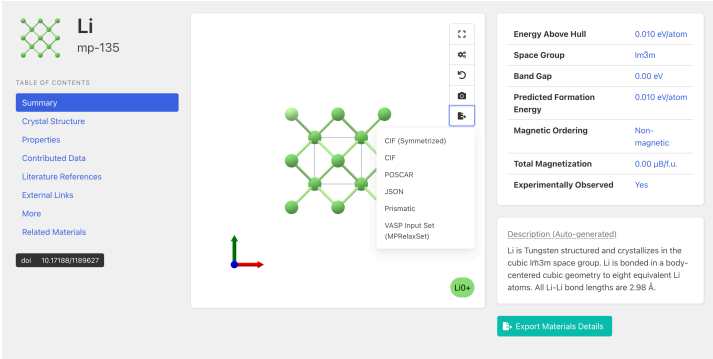
[1]:
```python
# Import relevant Python packages and modules within the ASE package

import numpy as np

from ase.build import *
from ase.visualize import view
from ase.io import lammpsdata
```

[2]:
```python
# Create an instance of an <Atoms> object. An <Atoms> object can represent an isolated molecule or periodic structure.
# In this case, we are building a pure BCC Li conventional unit cell.
Li = bulk('Li', 'bcc', 3.439, cubic=True)

# Create a 10x10x10 BCC Li supercell and visualize it
# Li = make_supercell(Li, np.diag([10]*3))
view(Li)

# Write the structure to a lammps .data file. You can include the masses in the file as well for convenience.
lammpsdata.write_lammps_data('bcc_Li.data', Li, masses=True)
```

```
2025-08-15 13:35:28.114 Python[1648:87041] +[IMKClient subclass]: chose IMKClient_Modern
2025-08-15 13:35:28.114 Python[1648:87041] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

1. Creating Structure File - Materials Project

- Go to **The Materials Project** website (you may need to create an account) - **https://next-gen.materialsproject.org/**
- Search for chemical system of interest and download **.cif** file.



- Convert the **.cif** file to a **lammps .data** file using **OVITO (recommended)**, OpenBabel or ASE in Python script

2. LAMMPs Input File - Initialize Simulation

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

Now we will begin discussing in greater detail the commands needed to run a basic structure relaxation with lammps. Each block of commands was taken from the lammps input script, **in.meam**.

- Start by initializing the simulation.

```
# ---- Initialize Simulation ---- #
clear
units             metal
dimension         3
boundary          p p p
atom_style        atomic
```

2. LAMMPs Input File - Initialize Simulation (Cont.)

- The `clear` command clears all memory
- The `units` command sets the style of units for all quantities specified in your input script and files (data file, potential file, etc.) and written to your output files.

  Please visit **https://next-gen.materialsproject.org/** for a description of all unit styles. The most common style is `metal` which uses Angstroms and eV for distance and energy, respectively. Picoseconds and Kelvin are used for the time and temperature, respectively.
  Note: Generally, potential files should have a **UNITS:** tag in the first line. Otherwise, you should get familiar with your potential and/or contact the authors to ascertain the unit style of the force field coefficients.
- The `boundary` command specifies the periodic boundary conditions in the x, y, and z directions, respectively.

2. LAMMPs Input File - Initialize Simulation (Cont.)

- The `atom_style` command specifies the per-atom attributes read from the data file and stored with those atoms. The `atom_style` is dependent on the type of potential (i.e., `pair_style`) used.
  The default is `atomic` and is used for most potentials.
- The `atom_modify` command dictates how atom IDs are assigned and how to retrieve specific atom IDs during the simulation.
  The default is `id yes`, meaning unique atom IDs are assigned to each atom in the system which are positive integers ordered consecutively from 1 to N atoms.
  By default, atomic styles do not use a map, however, creating a array-style map can often speed up calculations by storing a lookup table of length N in each processor, where N is the largest atom ID in the system.

2. LAMMPs Input File - Simulation Box

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

2 different methods can be used to build your chemical system:

1. Read from a **structure file** (as seen in **struc_relax_1** folder)

2. Use **commands in the lammps input script** to create chemical system (as seen in **struc_relax_2**).

LAMMPs Input File - Simulation Box (Option 1)

- One method of initializing the boundaries of your simulation box and the atomic positions is by creating a **lammps .data file** (see slides 4-6) and then reading the structure file using the `read_data` command in the input script.

```
# ---- Read Structure File ---- #
read_data         bcc_Li.data
```

2. LAMMPs Input File - Simulation Box (Option 2)

- Another method of creating your structure is by doing so within the lammps input script. Note that this lammps input script builds the exact same structure as seen in the test.ipynb script on slide 5.

```
# ---- Initialize Simulation Box ----
lattice          bcc 3.439
region           box block 0 1 0 1 0 1 units lattice
create_box       1 box
create_atoms     1 box
replicate        1 1 1
```

- The `lattice` command defines a lattice with a given phase (bcc, fcc, etc.) and lattice constant in the units you set previously in the input script with the `units` command.
- The `region` command defines the boundaries of your simulation box. In this case, `box` is the user-assigned name (or ID) for the region. `box` is a 3-dimensional `block` defined by dimensions `xlo xhi ylo yhi zlo zhi` in `lattice` units. In other words, the simulation region defined is a 3.439 Å $\times$ 3.439 Å $\times$ 3.439 Å box.
- The `create_box` command creates the simulation box defined earlier named `box`. `1` is the number of atom types in the system.
- The `create_atoms` command populates `box` with `1` type of atom. In this case, atoms are assigned to the `bcc` lattice points.
  Note: If you would like to assign velocities to these atoms before setting a thermostat, you would need to define a mass for each atom type.
- The `replicate` command multiplies the current system by a factor of `N`, in the x, y and z direction, respectively.

2. LAMMPs Input File - Interatomic Potential

- Define the interatomic potential parameters used to calculate the potential energy surface and the forces on the atoms at a given interatomic distance.

```
# ---- Define Interatomic Potential ----
pair_style      meam
pair_coeff      * * library.meam Li Li.meam Li
neighbor        2.0 bin
neigh_modify    delay 0 every 1 check yes
```

- The `pair_style` command dictates the type of interatomic potential used. It determines the format of the potential file, as well as the `atom_style` and other parameters used in the simulation. Most often, the `pair_style` is just the extension of the potential filename.
- The `pair_coeff` command defines the potential file name(s) and lists the element names whose pairwise force field coefficients are to be extracted from the potential file(s).

  Note: For the MEAM potential, **the list of element(s) between the two filenames, library.meam and *.meam, extracts lines from the library file and assigns indices to these elements irrespective of the order of atom types in your input script.** In other words, the elements are listed in the order that they appear in the library file. **The elements listed after the second file are mapped to the atom types (1...N_types) defined in your input script.** A wildcard asterisk is used to set the coefficients for multiple atom types. A leading asterisk sets the coefficients for atom types from 1 to N_types (inclusive).

2. LAMMPs Input File - Interatomic Potential (Cont.)

- The `neighbor` command sets the parameters used to build pairwise neighbor lists that speed up force calculations. These neighbor lists contain all atom pairs within a neighbor cutoff distance equal to the force cutoff (from potential file) plus the `skin` distance (a "buffer" zone set by the user). The larger the skin distance, the less frequently the neighbor lists need to be built, however, the more atom pairs will need to be checked for possible force interactions within each neighbor list at every timestep. By default, the `skin` distance is `2.0` Angstroms for `real` and `metal` units and the `bin` algorithm is used to build the list. Most often, the default values should suffice, however for some systems, the `skin` distance may need to be determined heuristically.

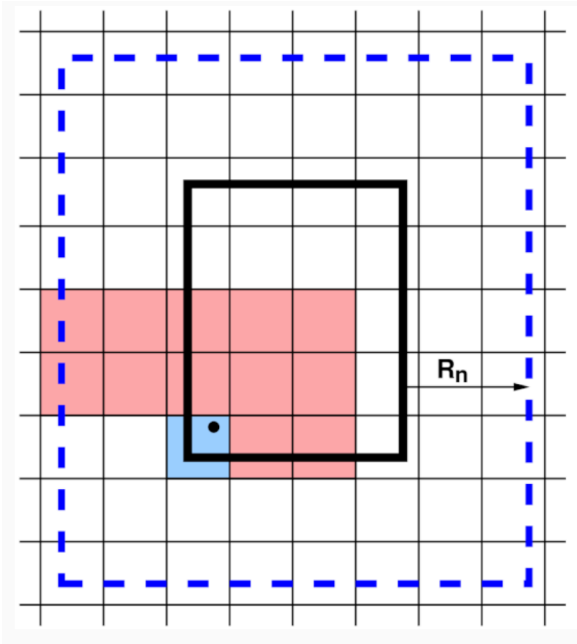


Figure: Half neighbor list stencils for a 2D model. The simulation space (in this case, 2D) is divided into a regular grid of neighbor bins (thin black lines). A single processor stores the set of neighbor bins that overlap the boundaries of its subdomain (thick black line) extended by the neighbor cutoff distance, R_n (dashed blue line). A list of integer offsets in the x, y, and z directions relative to the origin bin (blue bin) form the "stencil" which includes all bins close enough to have a neighbor atom j within R_n from any atom i in the origin bin. By building a "half" neighbor list, each i, j pair is stored only once.

2. LAMMPs Input File - Interatomic Potential (Cont.)

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

- The `neighbor_modify` command dictates how often pairwise neighbor lists are built during the simulation, affecting the accuracy and computational cost of the energy and force calculations. Generally, the greater the displacement of atoms from their initial positions in a timestep (diffusivity), the more often you want to rebuilt your neighbor lists. Although it may be the most computationally expensive, the safest option is to never delay the building of neighbor lists (`delay 0`), build new neighbor lists at every timestep (`every 1`) and always check if at least one atom has moved more than $\frac{1}{2}$ of the `skin` distance before rebuilding the neighbor list (`check yes`).

LAMMPs Input File - Define Custom Computations


```
# ---- Define Settings ----
compute eng all pe/atom
compute eatoms all reduce sum c_eng
```

Syntax: `compute ID group-ID style args`

where `ID` is user-assigned name, `group-ID` is the group of atoms to perform the calculation on and `style` is one of several `compute` styles - see LAMMPs manual.

Description:

- The `compute` command defines a custom computation to be performed on a group of atoms. In this case, the name `eng` defines the computation of per-atom potential energy to be performed on `all` atoms in the simulation box. `eatoms` stores the sum of all `eng` values.

  Note: the `compute` command does not actually perform the calculation. A given `compute` needs to be invoked by other commands in the input script.

2. LAMMPs Input File - Structure Relaxation Settings

- Relax the structure to find the equilibrium configuration using the conjugate gradient algorithm

```
# ---- Find minimum energy configuration ---
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press p
min_style cg
minimize 1e-25 1e-25 5000 10000
```

- The `reset_timestep` resets the current timestep to 0.

2. LAMMPs Input File - Structure Relaxation Settings (Cont.)

Syntax: `fix ID group-ID style args`

where `ID` is user-assigned name, `group-ID` is the group of atoms to apply the fix to and `style` is one of several `fix` styles - see LAMMPs manual.

Description:

- The `fix` command sets a thermostat or an operation that is applied to a group of atoms at each timestep or minimization step. In this case, all 3 dimensions of `box` are dilated/contracted together by a max allowed volume change of `0.001` distance units^3 in one iteration. The volume is iteratively changed until the potential energy is minimized and the hydrostatic pressure is close to `0.0` pressure `units`.

2. LAMMPs Input File - Structure Relaxation Settings (Cont.)

- The `thermo` command dictates how often a custom set of thermodynamic quantities need to be outputted to the log file. In other words, the thermodynamic quantities need to be written at every `N` timesteps.
- The `thermo_style` command enables customization of thermodynamic quantities to be outputted during the simulation. In this case, the timestep, potential energy, the length of `box` in the x, y, and z directions, and pressure will be written to the log file at every 10th timestep during the simulation.

2. LAMMPs Input File - Structure Relaxation Settings (Cont.)

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

- The `min_style` command sets the algorithm used for the iterative energy minimization of the system.
- The `minimize` command sets the stopping tolerances for energy and force, respectively, and the maximum number of iterations for the minimization. The minimization is terminated when one of the stopping tolerances are met at which the system is presumed to be at a local potential energy minimum.

2. LAMMPs Input File - Define Variables


```
# ---- Define Global Variables ----
variable N_atoms equal "count(all)"
variable total_e equal "c_eatoms"
variable length equal "lx"
variable coh_e equal "v_total_e/v_N_atoms"
```

- The `variable` command is used to define a global variable within the scope of the lammps input script. The style `equal` is used to define numbers and/or operations that involve numbers. In this case, the identifier `N_atoms` is assigned to a variable representing the total number of atoms in the system.

  `c_` and `v_` are prefixes used to access data from a user defined `compute` and `variable`, respectively. As a general rule, **the first letter of the prefix** corresponds to the **name of the command** from which you wish to extract data from. For example, `c_` $\rightarrow$ `compute`, `v_` $\rightarrow$ `variable`, `f_` $\rightarrow$ `fix`.

  `c_eatoms` returns the sum of all per-atom potential energies from the previously defined `compute` named `eatoms` while `v_total_e` returns the floating point value of the user-defined `total_e` variable.

2. Printing to Log File

- Finally, the following commands print the desired physical quantities to the log file.

```
print "Total energy (eV) = ${total_e};"
print "Number of atoms = ${N_atoms};"
print "Lattice constant (Angstoms) = ${length};
print "Cohesive energy (eV) = ${coh_e};"

print "All done!"
```

3. Running LAMMPs on c1 cluster

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

- Now that you've prepared the lammps input script, it's time to run the calculation.
- Since we're running the lammps executable on the c1 cluster, we need to prepare a Slurm file
- The following Slurm file requests 6 cores (processors) and 6G of memory per node from the cluster, which is suitable for simple calculations

```
#!/bin/bash
# this example is auto generated by the "guide" command
#SBATCH --job-name=Li_relax
#SBATCH --time=0-00:30       # Max time (DD-HH:MM)
#SBATCH --cpus-per-task=6    # Number of CPUs
#SBATCH --mem=6G             # Memory per node
module load class-simulations

/scic/app/cpu/lammps/2025-04-02/bin/lmp -in in.meam
```

3. Running LAMMPs on c1 cluster (Cont.)

Training Overview
Step 1: Preparing a Simulation
Step 2: Building a Structure
Step 3: Running the Simulation
Appendix

- Now it's time to submit the job to the cluster with the following command: `sbatch lammps_run.sh`
- If the calculation was performed correctly, you should see the following written to your log.lammps and slurm*.out file:

```
Total # of neighbors = 52
Ave neighs/atom = 26
Neighbor list builds = 0
Dangerous builds = 0
Total energy (eV) = -3.2994903358043 5;
Number of atoms = 2;
Lattice constant (Angstoms) = 3.48756708615128;
Cohesive energy (eV) = -1.64974516790218;
All done!
Total wall time: 0:00:00
```

References

This LAMMPs input script has been reproduced from **https://www.cavs.msstate.edu/icme/code/lammps/tutorials/lammps/tutorial1.php.**

The descriptions for the LAMMPs commands have been adapted from the offical LAMMPs code manual: **https://docs.lammps.org/Manual.html**

If you would like more detail on how to run lammps calculations or are at any point confused by any of the commands, it is recommended to supplement your reading with the following resources.