Lab 1: Empirical Energy Methods

Due: 2/19/18

Getting started

To get the lab libraries and samples go to the ~/Software directory and type:

```
bond@vmint ~/Software $ git clone https://github.com/bkoz37/labutil.git
```

To get the latest updates to the scripts and libraries in the future go into the 'labutil' director and type 'git pull'. Next, copy the sample files to your working directory.

```
bond@vmint ~/Software/labutil/lab1_samples $ cp * ~/WORK/Lab1/
```

Note: If you modify the plugins and samples in this directory while working on the assignment, the git command will notice this and will prompt you to 'merge' your changes when you try to pull or update the repository. Do not do this. Instead, backup any of your changes by renaming this folder if you ever want to edit any libraries or scripts. For instance:

```
bond@vmint ~/Software/labutil $ cd ..
bond@vmint ~/Software $ mv labutil/ my_labutil
```

Then repeat the initial procedure steps to set up a fresh repository (see above).

Preparing, manipulating and visualizing crystal structures with ASE

For preparing and manipulating crystal structures we will be using the ASE Python library. The documentation is quite accessible and even includes a refresher of Python basics. You can read it here https://wiki.fysik.dtu.dk/ase/python.html.

In the labs we will explore various functionalities of ASE, particularly how to make crystal structures, generate supercells, remove an atom to form a vacancy, and change an atom's position. Do not worry about 'calculators', we will not be using them. To learn how to use ASE we will use the Jupyter notebook file included in the lab materials. Start Jupyter as follows:

```
bond@vmint ~/WORK/Lab1/ $  jupyter notebook
```

General remarks on scientific scripting

Scientific scripting for managing the input and output data is an important component of modern materials computations, and you will be learning the basic tools while working on the lab exercises. For running computational codes we will be using simple Python functions as wrappers. The advantage of Python is that it has convenient data processing and text parsing tools built-in and more complex numerical operations can be also performed by importing common libraries, such as NumPy and SciPy. This makes it easy to compose rather complex workflow by connecting functions that each perform a specific task: making a crystal structure, preparing an input file, running the actual code, parsing and analyzing results, etc. Instead of hard-coding the scripts we will be using the data-flow programming model.

Understanding the LAMMPS input file

For classical potential computations we will use LAMMPS, one of the leading modern classical molecular dynamics codes, developed and maintained by the Sandia National Lab http://lammps.sandia.gov/. You will familiarize yourself with LAMMPS and input/output file formats as part of exercises below. First, let us take a look at the input file for calculating the total energy of an Al crystal unit cell. By default we name the input file '*lammps.in*'.

Note that the '#' sign indicates a comment line that LAMMPS ignores. Please take a look at the full input documentation here: http://lammps.sandia.gov/doc/Section_commands.html

Block 1 defines the basic parameters of the calculation, such as units, what types of atoms are used, periodicity, and the location of the structure file '*lammps.data'*.

Block 2 describes the potential. In this example we have explicit LJ parameters and the cutoff radius. We can also specify a potential from a filename (see lines commented out). You can find many potentials in the LAMMPS /potentials/ directory, and on the NIST page http://www.ctcms.nist.gov/potentials/

Block 3 first line tells the code how to print the results to the output file. Second line 'run 0' says to only run a single-step energy calculation, without any dynamics. The commented lines below are needed to perform lattice parameter and atomic optimization using the conjugate gradient minimization algorithm.

Block 4 asks the code to define and output several useful computed quantities in a pretty way.

```
# ---------- 1. Initialize simulation ---------------------
units metal
atom_style atomic
dimension  3
boundary   p p p
read_data /home/bond/WORK/Lab1/work/3.9/lammps.data

# ---------- 2. Specify interatomic potential -------------
pair_style lj/cut 4.5
pair_coeff 1 1 0.392 2.620 4.5

# pair_style eam/alloy
# pair_coeff * * /<path>/Al_zhou.eam.alloy  Al

# ---------- 3. Run the calculation ----------------
# -- perform a single-point energy calculation only
run 0

# -- include optimization of the unit cell parameter
# fix 1 all box/relax iso 0.0 vmax 0.001

# -- enable optimization of atomic positions (and the cell)
# min_style cg
# minimize 1e-10 1e-10 1000 10000

# ---- 4. Define and print useful variables -------------
variable natoms equal "count(all)"
variable totenergy equal "pe"
variable length equal "lx"

print "Total energy (eV) = ${totenergy}"
print "Number of atoms = ${natoms}"
print "Lattice constant (Angstoms) = ${length}"

# --------------------------------
```

A sample input fine is in the /labutil/lab1_samples/ folder. You can run LAMMPS with this input file directly from command line like so:

```
$ lmp_ubuntu < lammps.in > lammps.out
```

But running codes manually is an outdated way of computing, so you will be learning how to automate common tasks using Python workflows. You will be given example scripts and the wrapper library to help you get started.

Problem 1: (20 points) Lattice constants and energies

Goal: Find the ground-state energy and lattice parameter of FCC Al.

In nature, a crystal in equilibrium is “automatically” in the lowest energy configuration (ground-state). That is, the lattice energy being at a minimum defines the lattice geometry and atomic positions.

The exact crystal geometry is generally not known *a priori* in simulations. The best we can do is to start with a reasonable guess and try to find the lowest energy configuration from there. If the initial guess is close enough to the ground state, a feasible approach is to slightly adjust the atomic positions and lattice parameter until an energy minimum is found. This is commonly called “relaxation”, or “optimization”.

A. Relaxing FCC Al using a Lennard-Jones (LJ) potential with ε=0.392eV and σ=2.62 A
    a. Find the equilibrium lattice constant by **manually** adjusting the lattice parameter. Start with a lattice parameter close to the relaxed lattice parameter (e.g. 3.9 A). Plot the crystal energy as a function of lattice parameter (from above to below the optimized value) and identify the equilibrium lattice constant. Before running the calculations think of a reasonable range for the lattice constant, and how many steps to include.

    b. Calculate the lattice constant and cohesive energy (energy per atom in eV for the ideal crystal) for FCC Al using the LJ potential and the built-in structure optimizer. See notes above on the input command to turn on lattice optimization.

B. Repeat the calculations by using the supplied Embedded-Atom-Method (EAM) potential (potential file: *Al_zhou.eam.alloy* in the LAMMPS /potential directory).

C. Make a 2x2x2 supercell of the structure and compute the cohesive energy again as in part B. How do the results change, and why?

D. The experimental lattice constant of Al is 4.046 A. How do the calculated lattice constants (LJ vs. EAM) compare to experiment? Is this expected? Comment on the absolute values of the lattice energy and their significance.

Problem 2: (30 points) Vacancy formation energy

Goal: Compute the vacancy formation energy in FCC Al. Learn about practical aspects of total energy calculations such as: supercell convergence, effect of relaxations, and shortcomings of some potentials.

The vacancy formation energy E_v is defined as the energetic cost to remove an atom from the crystal and put it into the bulk:

$$E_v = E_{n-1} - \frac{n-1}{n} E_{perfect}$$

where $E_{perfect}$ and E_{n-1} are the energies of the perfect supercell with *n* atoms and of the defect cell, respectively. Because $E_{perfect}$ scales with the number of atoms in the supercell, we can scale it with a multiplier (n-1)/n so that we're comparing the energy of a defective cell with *n - 1* atoms to that of a perfect cell with *n – 1* atoms.

A. Compute the vacancy formation energy with the provided LJ potential as a function of different supercell sizes, at the lattice parameter you found in Problem 1. Use ASE to build the supercell and to remove an atom. Do not relax the atoms after you took out an atom. What is the ratio of the vacancy formation energy to the cohesive energy? Is this what you would expect?
B. Repeat your calculation, but this time relax the cell after creating the vacancy. How does the vacancy formation energy change compared to the unrelaxed calculations?
C. Compute the vacancy formation energy using the provided EAM potential. Do it as accurately as you can. Use what you learned above. You may do it any way you want, but document your work.
D. The experimental vacancy formation energy in fcc Al is 0.66 eV. How do the final results from B and C compare? Comment on your result in the light of your findings from problem 1.

Problem 3: (40 points) Surface energy of FCC Al

Goal: Calculation of surface energies. Apply your know-how regarding relaxation and convergence to a different problem.

A. Compute the surface energy of a solid (100) surface of FCC Al using the LJ potential. How you do this is up to you, but consider the issues of supercell sizes in all directions and the effect of relaxations. Document your approach and write down and explain the formulas you use for the surface energy. Give your result in eV/A^2.

   Hints: Think about which dimension supercell you use, keeping in mind periodic boundary conditions. Be computationally efficient. There are two size convergence issues to consider.

B. Compute the surface energy of a solid (100) surface of Al using an EAM potential. Again, you may do this any way you want, but document your work. In the interest of time, you may skip relaxations.

Problem 4: (5 points) Short answer

When using potentials, the choice of potential is important. In this problem, we used the Lennard-Jones and EAM energy methods.

A. For what other types of problems would you use Lennard-Jones potentials?
B. When would you use EAM?
C. When would neither be appropriate?

Problem 5: (5 points) Short answer

Given the simple form of the Lennard-Jones potential, why use computers to solve what at first glance appears to be a pencil and paper type problem? The following exercise should clarify the issue. Calculate the distance where the Lennard-Jones potential reaches the minimum value.

$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

(In other words, express r_0 in terms of ε and σ.) Plug in the values for ε and σ given for the Al Lennard-Jones potential Problem 1 and compare the result with your computed lattice parameter. Explain any discrepancies. In light of your finding, would you expect a top surface layer to relax inward or outward?