Computational Design of Materials Lab 1

Andrew Sullivan

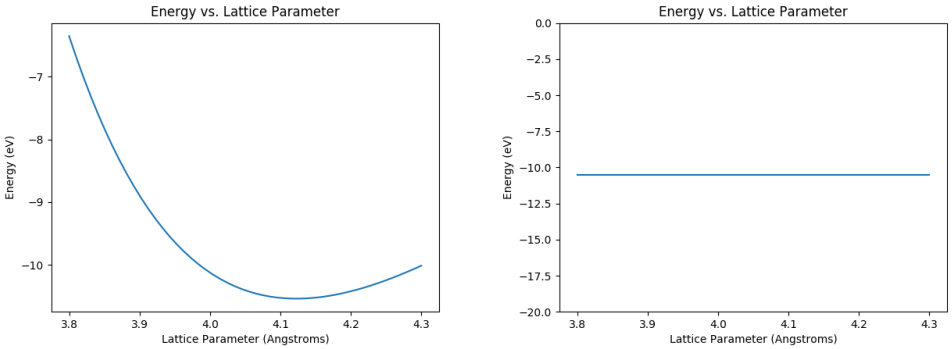
February 21, 2019

1 Problem 1: Lattice constants and energies

1.1 FCC Al with Lennard-Jones

In this section, we relax FCC Al using a Lennard-Jones potential with parameters $\varepsilon = 0.392eV$ and $\sigma = 2.62\mathring{A}$. This potential takes the form $V(r) = 4\varepsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}]$.

1.1.1 Manual Optimization



(a) Energy-lattice parameter plot. (b) Optimized energy-lattice parameter plot.

Figure 1: Optimization of Lattice Parameter using Lennard-Jones Potential for FCC Al.

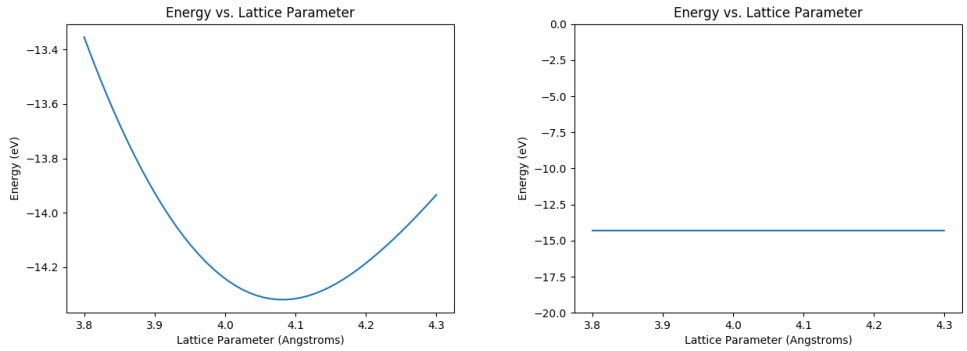
In order to determine the equilibrium lattice constant of FCC aluminum, we vary the lattice parameter between 3.8 and 4.3 $\mathring{A}$ using a total of 50 steps. This region should contain the minimum, which is around 4 $\mathring{A}$, and the steps are fine enough that the resultant curve is smooth. This analysis yields the above energy-distance plot in a), and the lattice constant is then the value which minimizes the energy, i.e. 4.127 $\mathring{A}$. This lattice constant corresponds to a minimum energy of -10.538 eV, yielding a cohesive energy of -2.635 eV/atom.

1.1.2 Automatic Optimization

To turn on the built-in structure optimizer, we uncomment three lines of code in the input template under the headings to include optimization of the unit cell parameter and enable atomic position and cell optimization. This yields the energy-distance plot above in b). As expected, here all input lattice constants yield the same energy as they are all optimized to the ideal value, with a corresponding energy of -10.538 eV, and the same cohesive energy as above. Checking the output file from LAMMPS reveals that the optimal lattice parameter is 4.123 $\mathring{A}$. This deviates slightly from the manual version above, though the resultant energies are the same out to 3 decimal places.

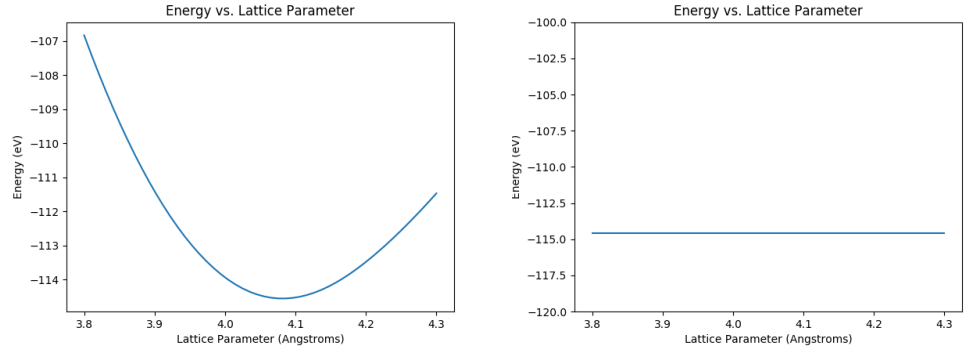
1.2 FCC Al with EAM

Rather than using a Lennard-Jones potential, in which the energy of each bond is independent of the coordination, here we use the embedded atom method (EAM) potential, $\sum_i F_i(\sum_{i \neq j} f(r_{ij})) + \frac{1}{2}\sum_{i \neq j} V(r_{ij})$. Here, the embdedding function, F, is an empirically obtained nonlinear function of the coordination. Using manual optimization, we obtain a lattice parameter of 4.086 $\mathring{A}$ and a corresponding minimum energy of -14.320 eV, yielding a cohesive energy of -3.58 eV/atom. When we include the automatic optimization as defined above, the lattice parameter changes slightly to 4.082 $\mathring{A}$, and the total and cohesive energy remains the same out to 3 decimal places.



(a) Energy-lattice parameter plot. (b) Optimized energy-lattice parameter plot.

Figure 2: Optimization of Lattice Parameter using Embedded-Atom Method for FCC Al.



(a) Energy-lattice parameter plot. (b) Optimized energy-lattice parameter plot.

Figure 3: Optimization of Lattice Parameter using Embedded-Atom Method for 2x2x2 FCC Al Supercell.

1.3 2x2x2 FCC Al Supercell with EAM

If we instead construct a 2x2x2 supercell of the structure with the EAM method, we find a similar lattice parameter without optimization of 4.086 $\mathring{A}$ and a new total energy of -114.559 eV. This new structure has 32 total atoms, and so the cohesive energy remains constant to that obtained above at -3.58 eV/atom. Interestingly, if we turn on the lattice optimization function, the energy obtained is nearly identical, -114.560 eV, but the returned lattice parameter in the LAMMPS output file is 8.163 $\mathring{A}$, double that obtained in previous parts. This suggests that the optimizer obtains the parameter of the entire supercell, rather than an individual unit cell. Indeed, if we increase the supercell size to 3x3x3, the optimizer returns a value of 12.245 $\mathring{A}$, three times that of the unit cell.

1.4 Discussion

Clearly, the results from the EAM potential are closer to the experimental value than those obtained using the LJ potential. In all cases, including the optimizer did not produce a large change from the manual version, likely due to the large number of subdivisions used in a relatively small range of parameters. These results are expected, as the LJ potential tends to fail in metallic systems, since pair potentials such as LJ predict a cohesive energy that is proportional to the coordination number, while experiment instead predicts a cohesive energy proportional to the square root of the coordination due to electronic delocalization. The lattice energy supports the hypothesis that EAM potentials will provide a more stable structure, as the resultant total lattice energy and cohesive energies are both more negative for EAM, suggesting that the atoms are in more stable positions. The lattice energy of the supercell is expected, as it simply reflects that

obtained for the single unit cell with an EAM potential over a total of 8 cells.

2 Problem 2: Vacancy formation energy

2.1 Vacancy Formation Energy in FCC Al with LJ Potential

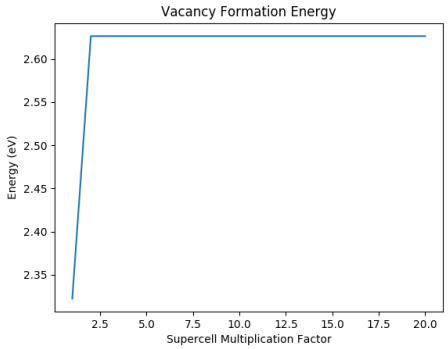


Figure 4: Vacancy Formation Energy of FCC Al using Lennard-Jones Potential.

In order to obtain the vacancy formation energy in FCC Al, we first use an LJ potential with the previously specified parameters and construct a supercell of varying size with a vacancy created by removing the atom with index zero. In this and all following calculations, unrelaxed lattice parameters are set to 4.0857 $\mathring{A}$. Due to periodic boundary conditions and the fact that all atoms are identical, it should not matter which atom is removed. The vacancy formation energy is obtained as $E_v = E_{n-1} - \frac{n-1}{n} E_{perfect}$, where the first term on the right-hand side is the energy of the supercell with the vacancy, the second energy term is the corresponding energy of a perfect lattice with the same supercell size, and the scaling factor accounts for the 1-atom difference between the two structures. Supercells ranging from size 1 to size 20 were constructed and their vacancy formation energies calculated. The plot of formation energy versus supercell size shows clear saturation by a size of 3x3x3, and the corresponding energy is 2.626 eV, approximately equal to the cohesive energy. This result is expected for Lennard-Jones potentials, but is in constrast with experiment, where in metallic systems vacancy formation energies are often less than half of the cohesive energy.

2.2 Vacancy Formation Energy in FCC Al with LJ Potential and Relaxation

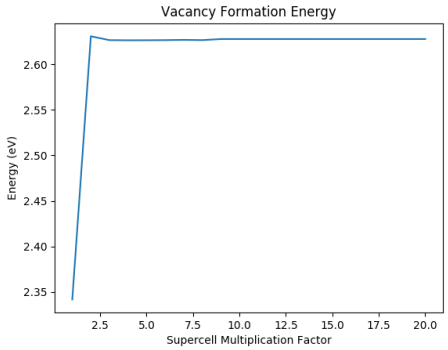
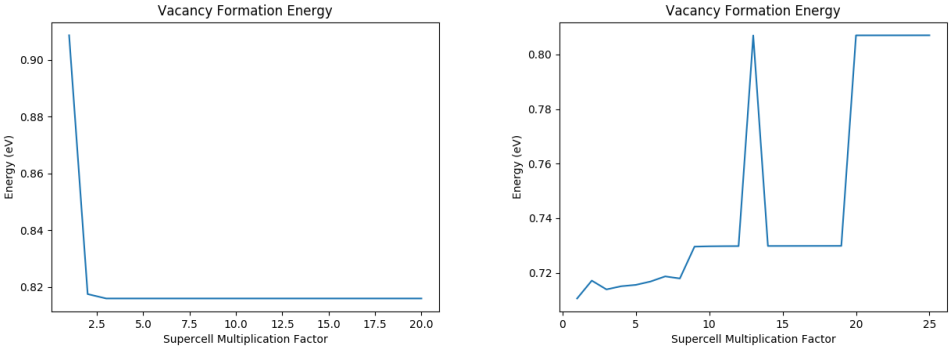


Figure 5: Vacancy Formation Energy of FCC Al using Lennard-Jones Potential with Relaxation.

Including relaxation after creation of the vacancy using the uncommenting method discussed above does not change the results dramatically, yielding a vacancy formation energy of 2.628 eV. This small change is

consistent with the other results of relaxation, namely, a small change in the lattice parameter and essentially no change in the total lattice or cohesive energies. Even so, it is interesting that the vacancy formation energy is slightly higher when relaxations are included, as one would expect relaxations to reduce the final energy of the system slightly as atoms rearrange slightly in response to the stress field induced by the vacancy, rather than increase the final energy.

2.3 Vacancy Formation Energy of FCC Al using EAM Potential



(a) Vacancy Formation Energy of FCC Al using Embedded Atom Method.

(b) Vacancy Formation Energy of FCC Al using Embedded Atom Method with Relaxation.

Figure 6: EAM Vacancy Formation Energy.

Using an EAM potential rather than LJ produces exactly what would be expected for the unrelaxed calculations. We now obtain a vacancy formation energy of 0.816 eV and a ratio of vacancy-to-cohesive energies of 0.228, which is in accordance with what would be expected from experimental results. Including optimization, however, produces the curve on the right, which does not display the convergence behavior seen in the unrelaxed case. Even including supercells up to size 35x35x35 still produces jumps between the maximum value of 0.807 eV and minimum of 0.730 eV seen on the right side of the above plot. It is possible that the curve will truly converge at larger values of the structure, but due to time and computational cost, this was not investigated.

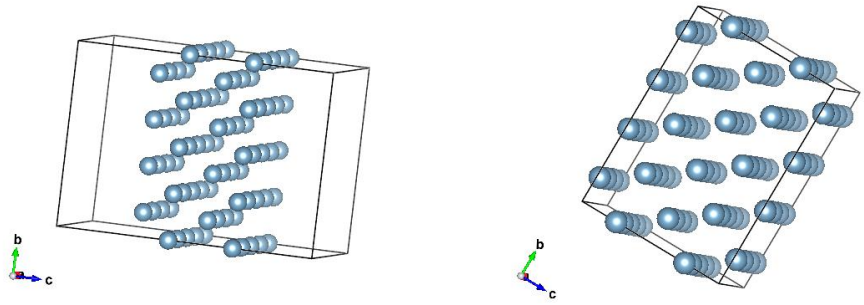
2.4 Discussion

Consistent with the results from problem 1, the EAM potential provides a result that is much closer to the experimental findings, with errors of less than 0.2 eV in comparison to almost 2.0 eV for the LJ cases. Including of parameter optimization also highlights the tradeoff between accuracy and speed. For the LJ case, complete convergence does not occur until the supercell is 10x10x10, though the change between this and a 4x4x4 supercell is quite small. In contrast, the optimized EAM case, convergence is not seen even up to a 35x35x35 supercell. However, both values that the energy seems to jump between at larger supercell sizes are even closer to the true value of 0.66 eV than any of the other tested methods, with the lower of the two being off by only 0.07 eV.

3 Problem 3: Surface energy of FCC Al

3.1 Surface Energy of FCC Al with Lennard-Jones Potential

The surface energy of FCC Al is obtained by constructing to comparable slabs and comparing their energies. Initially, a slab created using the fcc100 function from the ase.build library was compared to a supercell with the same inputs (i.e. slab size was specified as 4x4xi and a supercell multiplier from an identity matrix



(a) Slab Constructed with Vacuum = 5.0.

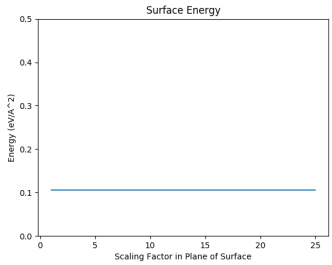
(b) Reference Slab Constructed with No Vacuum and Same Dimensions.

Figure 7: Slabs for Surface Energy Calculation.

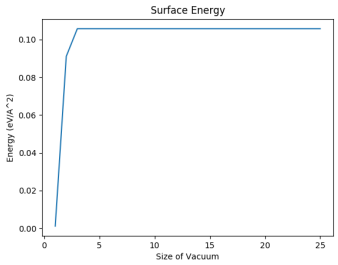
with diagonal elements 4, 4, and i), but these functions did not produce structures with identical numbers of atoms, and thus, could not be easily compared. An alternative method would be to represent the energy of a corresponding supercell as a simple product, $E_{supercell} = NE_{coh}$ using the above cohesive energy. Instead, the fcc100 function was used to create both the slab with the surface (specified by a vacuum not equal to None) and one with the same size but no vacuum. Unfortunately, having a vacuum of None (the default value) is incompatible with the LAMMPS energy functions, as it produces a dimension perpendicular to the surface of size 0. The ASE documentation for the surface-creating functions mentions that these functions should accept an optional parameter "periodic," which is set to False by default. However, the version of these functions included in the ase.build library seem to be outdated, as they do not accept this parameter. To account for this, the surface2.py file was created, which simply contains the source code from the ASE documentation for all the surface-creating functions and the $\_surface$ class itself. In the code used for this question, the "$fcc100\_2$" function is the updated version which accepts the periodic argument and allows for direct comparison due to the presence of the vacuum/surface.

Once the structures have been built, the total energy is calculated for each as a function of three parameters- the vacuum size, the slab thickness, and the slab dimensions in the plane of the surface. This final case is merely to validate the claim that surface energy is independent of these dimensions, as the periodic boundary conditions should remove the effect of any changes along these directions. The surface formation energy is calculated as $E_{surf} = \frac{E_{vacuum} - E_{novacuum}}{2A}$, where the numerator accounts for the difference between the slab with the vacuum (and thus, the surface) and the one without (corresponding to the bulk), and the denominator accounts for the area of the surface, A, and the fact that 2 surfaces are created due to periodic boundary conditions on either side of the slab. Given that the input size to the fcc100 functions does not appear to directly correspond to the size of a single unit cell, the length along the dimensions in the plane of the surface was extracted from the LAMMPS output file with these dimensions set to 1 (2.863782 $\mathring{A}$), and this value was squared and multiplied by the square of the input dimensions. For example, a structure with size (4,4,x) would have an area of $16 * 2.863782^2 \mathring{A}^2$. As discussed later (see problem 5), this dimension is close to the nearest-neighbor distance obtained using the relation $r = a\sqrt{(2)}/2$ for an FCC unit cell, and thus it appears that this function increments by single layers of atoms, rather than whole unit cells.

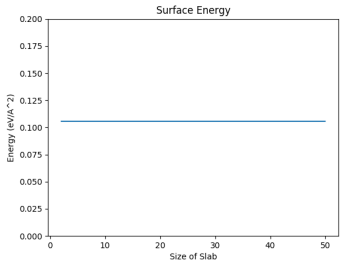
Using the above methodology, we obtain the 3 curves in Figure 8. As expected, the surface energy is independent of the dimensions within the plane of the slab. Interestingly, the value also appears to be essentially independent of the thickness of the slab, suggesting that the structure converges immediately. Convergence of the energy as a function of the vacuum size occurs rapidly as well, by a value of 4.0. When all three parameters have converged (here, we use in-plane dimensions of 4x4 and a thickness of 10 with a vacuum size of 5.0), a surface energy of 0.1057 $eV/\mathring{A}^2$ is obtained. Including relaxations increases this value of 0.1069 $eV/\mathring{A}^2$, consistent with what was observed in problem 2, wherein including optimization with LJ potentials increased the energy required to produce a defect.



(a) Energy as a Function of in-plane Dimensions.

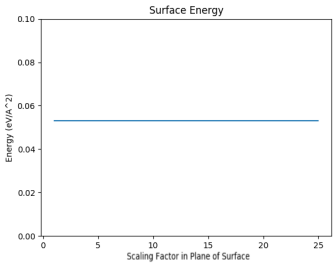


(b) Energy as a Function of Vacuum Size.

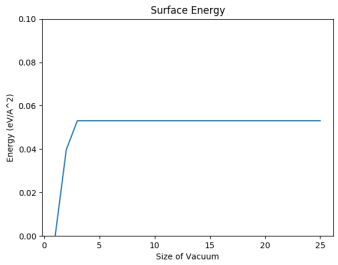


(c) Energy as a Function of Slab Thickness.

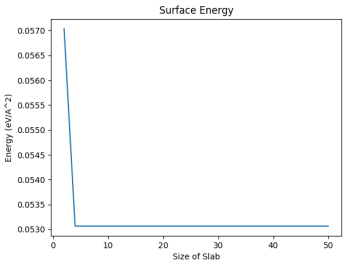
Figure 8: Convergence Properties of FCC Al Surface Energy with Lennard-Jones Potential.



(a) Energy as a Function of in-plane Dimensions.



(b) Energy as a Function of Vacuum Size.



(c) Energy as a Function of Slab Thickness.

Figure 9: Convergence Properties of FCC Al Surface Energy with Embedded-Atom Method.

3.2 Surface Energy of FCC Al with EAM Potential

Figure 9 displays the results for the same procedure as above with the EAM potential. One notable difference is the scale of the horizontal axis for the thickness of the slab, which is doubled relative to the other plots. As shown in Figure 10, running the EAM energy calculation with the normal range of 1 to 25 produces a periodic behavior that fluctuates between two values. In order to avoid this fluctuation, which may result from the fact that the input to the fcc100 function does not correspond to integer values of the unit cell size and may thus disrupt lattice periodicity in the direction of the surface, we only sample every 2 structures. This is consistent with the above observation that each increment only corresponds to a single nearest-neighbor distance, rather than an entire cell (which would be double along the $< 110 >$ closest-packed directions). Using this method, we obtain a surface energy of 0.053 $eV/\mathring{A}^2$. Including relaxations does not appear to affect the resultant energy, increasing it slightly to 0.054 $eV/\mathring{A}^2$.

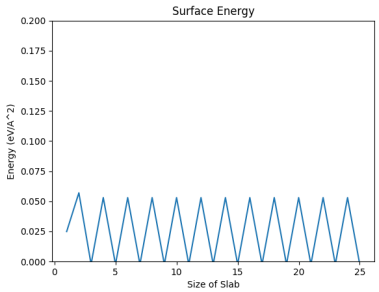


Figure 10: Oscillations in Energy when Increment of 1 is Used for Slab Thickness.

4 Problem 4: Short answer

4.1

Pair potentials like Lennard-Jones are suitable for the more simplistic atomic or molecular environments, such as gases, especially noble gases, certain fluids, and adsorbates. Ultimately, the LJ potential is fairly accurate at capturing inter-molecular or inter-atomic interactions of neutral atoms and molecules, but begins to fail when used within a given structure, such as a molecule or a crystal due to a lack of angular dependence and incorrect scaling with coordination, respectively.

4.2

Pair potential methods rapidly lose their utility in metallic solids and are unable to predict crystal structure, owing to the linear relationship between energy and coordination. In metals, the delocalization of electrons produces a cohesive energy that scales with the square root of the coordination, while pair potentials like LJ scale linearly. EAM is specifically suitable for metallic solids, and predicts results that are more closely aligned with experiment than those achievable by LJ while not drastically increasing the computational cost. It is suitable for more complex metallic environments as well, such as those including defects like vacancies, grain boundaries, or cracks.

4.3

Despite the utility of the EAM potential for capturing many properties of metallic solids, both EAM and pair potential methods fail to accurately describe most other atomic or molecular systems. Given that the EAM potential assumes spherical bonding and thus is independent of the angle of the bond, it is unsuitable for any covalent interactions, such as directional bonding in silicon, which requires the introduction of 3-body potentials such as Stillinger-Weber. Coulombic/electrostatic interactions are also not present in the above methods, and thus require either a modification or a completely new potential to deal with ionic solids. The presence of charge transfer, intermolecular interactions, and the formation and breaking of bonds further complicate the situation and require the introduction of new force fields.

5 Problem 5: Short answer

To find the minimum of the Lennard-Jones potential, we differentiate the potential with respect to the distance and set the result equal to zero.

$$
\begin{aligned}
V(r) &= 4\varepsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \\
\frac{dV(r_0)}{dr} &= 4\varepsilon[-12(\frac{\sigma^{12}}{r_0^{13}}) + 6(\frac{\sigma^{6}}{r_0^{7}})] = 0 \\
&= 24\varepsilon\sigma^{6}[\frac{-2\sigma^{6}}{r_0^{13}} + \frac{1}{r_0^{7}}] = 0 \\
&= -24\varepsilon\sigma^{6}[\frac{2\sigma^{6} - r_0^{6}}{r_0^{13}}] = 0 \\
r_0 &= \sqrt[6]{2}\sigma
\end{aligned}
$$

Using this equation, we obtain a lattice parameter of 2.94 $\mathring{A}$, which is quite different from the computational or experimental lattice parameters obtained in problem 1. However, this makes sense when the crystal structure is considered, as the lattice parameter does not necessarily correspond to the minimum on the energy curve. This minimum value should correspond to the center-to-center distance of nearest neighbor atoms. For a face-centered cubic structure, nearest neighbors are located along the diagonal of each face. The face diagonal covers a distance $a\sqrt{(2)}$, where a is the lattice parameter, and this is equivalent to four

atomic radii, with 1 contributed from each corner atom and 2 from the atom in the center of the face. Thus, the center-to-center distance, which corresponds to 1 atomic diameter, is equal to $a\sqrt{(2)}/2$. If we use the lattice parameter obtained in the first part of problem 1, 4.127 $\mathring{A}$, this yields a minimum distance of 2.92 $\mathring{A}$, quite close to the value obtained from the equation. However, this small difference suggests that, when half the nearest neighbors are removed during construction of a surface, the outer atoms will relax to their equilibrium positions, moving 0.02 $\mathring{A}$ outward, rather than inward, in contrast to experiment.

6 Appendix

The following is the modified code used for the calculations above. Important differences include the "type" structure used in the main function body, which call different versions of $lattice\_scan()$ depending on the question being answered. Inputs to $compute\_energy()$ and $make\_struc()$ functions control the size of supercells being constructed, whether a vacancy is formed by popping the atom with index 0, and locations to save the resulting files. Another distinction is the use of the $fcc100\_2$ function, which is an updated version of the fcc100 function provided in the initial code. This function was constructed using the documentation from the ASE website under the "Surfaces" page. The source code was copied to a new file, surface2.py, and imported to include the updated function and $\_structure$ class. This updated function accepts the optional argument "periodic," which is set to True in question 3 to allow for construction of an equivalent slab with zero vacuum to compare to the vacuum slab for surface energy calculation, as direct comparison between the slab and a constructed supercell was not straightforward.

```
from my_labutil.src.plugins.lammps import *
from ase.spacegroup import crystal
from ase.build import *
from surface2 import *
import numpy
import matplotlib.pyplot as plt
from ase.io import write


input_template = """
# ---------- 1. Initialize simulation ---------------------
units metal
atom_style atomic
dimension  3
boundary   p p p
read_data $DATAINPUT

# ---------- 2. Specify interatomic potential -------------------
pair_style eam/alloy
pair_coeff * * $POTENTIAL  Al

#pair_style lj/cut 4.5
#pair_coeff 1 1 0.392 2.620 4.5

# ---------- 3. Run single point calculation  -------------------
thermo_style custom step pe lx ly lz press pxx pyy pzz
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 (Angstroms) = ${length}"
      """

def make_struc(i, vac, alat, type):
    """
    Creates the crystal structure using ASE.
    :param alat: Lattice parameter in angstrom
    :return: structure object converted from ase
```

```
    """
    unitcell = crystal('Al', [(0, 0, 0)], spacegroup=225, cellpar=[alat, alat, alat, 90, 90,
     90])
    multiplier = numpy.identity(3) * i
    ase_supercell = make_supercell(unitcell, multiplier)
    #if i == 2:
        #from ase.visualize import view
        #view(ase_supercell)
    if vac == 1:
        ase_supercell.pop(0)
    structure = Struc(ase2struc(ase_supercell))
    return structure


def compute_energy(alat, template, i, vac, type):
    """
    Make an input template and select potential and structure, and the path where to run
    """
    if type == 1:
        potpath = os.path.join(os.environ['LAMMPS_POTENTIALS'], 'Al_zhou.eam.alloy')
        potential = ClassicalPotential(path=potpath, ptype='eam', element=["Al"])
        runpath = Dir(path=os.path.join(os.environ['WORKDIR'], "Lab1", "1C", "1_EAMOpt", str(
     i), str(alat)))
        struc = make_struc(i=i, vac=vac, alat=alat, type = type)
        output_file = lammps_run(struc=struc, runpath=runpath, potential=potential,
     intemplate=template, inparam={})
        energy, lattice = get_lammps_energy(outfile=output_file)
        return energy, lattice
    if type == 2:
        potpath = os.path.join(os.environ['LAMMPS_POTENTIALS'], 'Al_zhou.eam.alloy')
        potential = ClassicalPotential(path=potpath, ptype='eam', element=["Al"])
        runpath = Dir(path=os.path.join(os.environ['WORKDIR'], "Lab1", "2C", "2_EAMOpt",str(
     vac), str(i), str(alat)))
        struc = make_struc(i=i, vac=vac, alat=alat, type = type)
        output_file = lammps_run(struc=struc, runpath=runpath, potential=potential,
     intemplate=template, inparam={})
        energy, lattice = get_lammps_energy(outfile=output_file)
        return energy, lattice
    if type > 2:
        potpath = os.path.join(os.environ['LAMMPS_POTENTIALS'], 'Al_zhou.eam.alloy')
        potential = ClassicalPotential(path=potpath, ptype='eam', element=["Al"])
        if type == 3:
            runpath = Dir(path=os.path.join(os.environ['WORKDIR'], "Lab1", "3B", "Slab_opt",
     str(vac), str(i), str(alat)))
            slab = fcc100_2('Al', size=(4, 4, i), vacuum=5, periodic = True)
        if type == 4:
            runpath = Dir(path=os.path.join(os.environ['WORKDIR'], "Lab1", "3B", "Bulk", str(
     vac), str(i), str(alat)))
            slab = fcc100_2('Al', size=(4, 4, i), periodic = True)
        if i == 4 and type == 3:
            #from ase.visualize import view
            #view(slab)
            write('slab.cif', slab)
        struc = Struc(ase2struc(slab))
        output_file = lammps_run(struc=struc, runpath=runpath, potential=potential,
     intemplate=template, inparam={})
        energy, lattice = get_lammps_energy(outfile=output_file)
        return energy, lattice

def lattice_scan(i, vac, type):
    if type > 1:
        alat = 4.0857
        energy_list = [compute_energy(i=i, vac=vac, alat=alat, template=input_template, type=
     type)[0]]
        return min(energy_list)
    else:
        alat_list = numpy.linspace(3.8, 4.3, 50)
        energy_list = [compute_energy(i=i, vac=vac, alat=a, template=input_template, type=
```

```
    type)[0] for a in alat_list]
110         lattice = [compute_energy(i=i, vac=vac, alat=a, template=input_template, type=type)
    [1]for a in alat_list]
111         ind = energy_list.index(min(energy_list))
112         print("lattice parameter =", alat_list[ind])
113         print("minimum energy =", min(energy_list))
114         plt.plot(alat_list, energy_list)
115         plt.xlabel('Lattice Parameter (Angstroms)')
116         plt.ylabel('Energy (eV)')
117         plt.title("Energy vs. Lattice Parameter")
118         axes = plt.gca()
119         axes.set_ylim([-120, -100])
120         plt.show()
121
122
123
124 if __name__ == '__main__':
125     type = 2                                                    #if 1, varies energy over
    linspace with fixed supercell size i
126                                                                   #if 2, uses fixed lattice
    parameter with variable supercell size for vacancy formation
127     if type == 2:                                               #if 3, constructs slabs with
    specified dimensions and vacuum and with the same dimension but no vacuum
128         min_E_n1 = []
129         min_E_perf = []
130         E_coh = []
131         Ev = []
132         for i in range(1,26):
133             min_E_perf.append(lattice_scan(i, 0, type))
134             E_coh.append(lattice_scan(i,0,type)/(4*(i**3)))
135
136         for i in range(1,26):
137             min_E_n1.append(lattice_scan(i, 1, type))
138
139         for i in range(1,26):
140            Ev.append(min_E_n1[i-1] - ((4*i**3-1)/(4*i**3))*min_E_perf[i-1])
141
142         print(min_E_perf)
143         print(E_coh)
144         plt.plot(range(1,26), Ev)
145         plt.xlabel('Supercell Multiplication Factor')
146         plt.ylabel('Energy (eV)')
147         plt.title("Vacancy Formation Energy")
148         axes = plt.gca()
149         #axes.set_ylim([-5, 0])
150         plt.show()
151
152     if type == 1:
153         lattice_scan(i=1, vac=0, type=type)
154
155     if type == 3:
156         Esurf=[]
157         alat = 4.0857
158         Etemp=[]
159         Etemp2=[]
160         r=[]
161         for i in range(1,26):
162             Etemp.append(lattice_scan(i=i,vac=0,type=type))
163             Etemp2.append(lattice_scan(i=i,vac=0,type=4))
164             Esurf.append((Etemp[i-1]-Etemp2[i-1])/(2*(2.8637825**2)*(4**2)))
165             r.append(2*i)
166
167         plt.plot(range(1,26),Esurf)
168         plt.xlabel('Size of Slab')
169         plt.ylabel('Energy (eV/A^2)')
170         axes = plt.gca()
171         axes.set_ylim([0, 0.2])
172         plt.title("Surface Energy")
```

```
173         plt.show()
174
175
```