UNIVERSITY OF CALIFORNIA BERKELEY

Numerical Algorithms Applied to Computational Quantum Chemistry
Homework 1: Potential Energy Functions, Forces and Local Optimization

Prof. Martin Head-Gordon, with Zhe Wang (C279) and Xiao Liu (C179)

Due 11:59PM, Wednesday Jan. 31, 2024

1 THE LENNARD-JONES ENERGY OF A SET OF ATOMS.

Implement a C++ code to evaluate the energy of a cluster of gold (Au) atoms, at positions $\mathbf{R}_i$, using a Lennard-Jones (LJ) potential:

$$E_{LJ} = \sum_{i<j} E_{ij} \tag{1.1}$$

$$E_{ij} = \epsilon_{ij} \left\{ \left(\frac{\sigma}{R_{ij}} \right)^{12} - 2 \left(\frac{\sigma}{R_{ij}} \right)^{6} \right\} \tag{1.2}$$

Here R_{ij} is the distance between atoms i and j, and ϵ_{ij} and σ_{ij} are problem-dependent constants, which are usually defined as the geometric mean of the corresponding parameters for the atoms:

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \tag{1.3}$$

$$\sigma_{ij} = \sqrt{\sigma_i \sigma_j} \tag{1.4}$$

For the special case of Au atoms, a good set of constants are (see H. Heinz *et. al.*, J. Phys. Chem. C 112, 17281 (2008); doi: 10.1021/jp801931d):

$$\epsilon_{\text{Au}} = 2.951\ \text{Å} \tag{1.5}$$

$$\sigma_{\text{Au}} = 5.29\ \text{kcal mol}^{-1} \tag{1.6}$$

While the LJ potential is incredibly simple, and was first developed to describe noble gases such as neon or argon (see ICC, ch. 2: "The Van der Waals energy"), it has also been used with some success for predicting the properties of a range of molecules and materials, including gold! There are even recent papers using LJ potentials to describe metals, alloys and their interactions with molecules (see K. Kanhaiya *et. al.*, Npj Comput. Mater. 7:17 (2021); doi:

10.1038/s41524-020-00478-1). Hence writing a code to implement the LJ potential is a useful exercise!

Remember principles of good software design even as you do a small project like this. Aim for modular code so that your components can be re-used for later projects.

Your code should:

- Read in the coordinates of some atoms from a text file containing information about one atom per line, in the format: `A x y z`. Here `A` is the atomic number, and `x y z` are the Cartesian coordinates of the atoms. Your TA will specify any requirements for the file format during the first discussion class.
- Echo the input to the output, and then crash with an error message if there are any atoms other than Au atoms.
- Return the energy of the cluster.

Demonstrate that your code is correct by evaluating against the two test cases that will be provided.

2 Finite difference versus analytical forces

For a simple energy function such as the LJ potential used in the previous exercise, it is not too difficult to evaluate the analytical forces, defined as:

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{R}_i} \tag{2.1}$$

Extend your LJ energy capability to evaluate the analytical force by working out that algebra associated with the above equation using E_{LJ}, and implementing the force within your small program above.

If the analytical force is cumbersome to evaluate, then it may be a good idea to test the implementation by performing finite differences of the energy, using either a forward difference approximation:

$$\mathbf{F}_i \approxeq -h^{-1}\left[E(\mathbf{R}_i + h) - E(\mathbf{R}_i)\right] \tag{2.2}$$

or, with higher accuracy, a central difference approximation:

$$\mathbf{F}_i \approxeq -(2h)^{-1}\left[E(\mathbf{R}_i + h) - E(\mathbf{R}_i - h)\right] \tag{2.3}$$

Implement a finite difference evaluation of the force using the forward and central difference equations. Use a step size of 0.01 Å and make sure your forces match finite differences (e.g. for Au_2).

When your analytical force code is correct, investigate the truncation error of the forward and central difference expressions as a function of h, trying $h = 0.1, 0.01, 0.001, 0.0001$, and making a plot of the log of the error (vs your analytical gradient code) versus the log of the step size. Report your slope for each of the two approximations and discuss.

3 Optimizing the geometry of a small gold cluster

Your program now has energy and forces for the LJ potential of a collection of Au atoms. With the addition of some code to take steps in the nuclear coordinates given the energy and force, we can optimize the structures of some gold clusters!

So, implement a local optimizer that works by steepest descent with line searches. The idea is as follows. The steepest descent step is taken in the direction of the gradient. But how far to step? The answer is resolved by repeated evaluation of the energy in the direction of the gradient to find a minimum, which is a 1-dimensional search. Read about 1-d function optimization in Numerical Recipes and implement a simple algorithm for yourself. Convergence should be based on the energy change.

Test your simple optimizer on the simplest gold cluster, Au_2. The steepest descent direction is along the Au-Au bond vector, which is the only degree of freedom in your problem. Hence this is in principle only a test of your linesearch code, which should converge in one steepest descent step!

Then pick a gold cluster with more atoms. If you choose 3 or 4 atoms, then do at least three clusters of different sizes! Report your initial guess and optimized structures. Maybe try more than one initial guess and see what happens.

4 Going further, but only if you want to!

- **Higher order finite difference expressions** Whilst central differences are useful, you can also read (in Numerical Recipes of course) about methods that have advantages for higher order finite difference approximations. For instance, Chebyshev approximations are very effective for accurate numerical derivatives of smooth functions.

- **Investigate the Tersoff potential.** The LJ potential is very simple, and cannot describe directional, covalent bonding between atoms such as Si or C. Read about the Tersoff potential as described by its originator: J. Tersoff, Phys. Rev. Lett. 56, 632 (1986). If you are inspired, then implement it for either Si or C clusters (for C, see: J. Tersoff, Phys. Rev. Lett. 61, 2879 (1988)), and perhaps optimize some structures (graphene-like molecules, anyone?).

- **Clusters that are metal alloys.** You could extend your LJ code to treat atoms of more than one type. For instance, how about an alloy of Au and Ag? The parameters for Ag are available in the article mentioned previously (H. Heinz *et. al.*, J. Phys. Chem. C 112, 17281 (2008)). You could do some numerical experiments on such clusters.

- **Simulated annealing for optimization.** An interesting alternative to steepest descent is simulated annealing which the C279 students implemented in Python last semester if my spies have informed me correctly. This is a powerful method that can help you find the global minimum of a cluster of a given number of atoms by exploring different initial guesses and cooling schedules.