Exercise 4 Calculation of the energy and geometry optimization for naphthalene

A good measure for the quality of the force field is perform a geometry optimization and compare the result to an accurate optimization with an *ab-initio* method or with an experimental structure. In principle all modeling efforts start by an optimization to ensure that one starts with the (local) minimum energy structure of that particular force field. Calculations using structures that are perhaps experimentally more accurate but are up in the potential well, lead to unreliable results. You will use different optimization algorithms to optimimize the crystal structure of naphthalene.

a) We will use the website `jupyterhub` for this. Login like last week and open a terminal. From here we will run LAMMPS. Since one of the files has changed, repeat steps e-f to get the update LAMMPS inputfile.

   LAMMPS is a very powerful Molecular Dynamics program, but it can also minimize structures and only perform energy calculations. It is used in many research groups for simulations. LAMMPS needs two input files: one with the commands which tells LAMMPS what to do and one with the system, which contains the position of the atoms and the force field parameters. For this you will need `minimize.in` and `naphtha.data`, respectively.

b) Open these files in the "Dir Tab" and read them. Try to understand what each line does. We use the Dreiding force field. The paper is in the directory as well. On page 8904 in the table, you can see Dreiding can be used with several functional forms and both all-atom and with implicit hydrogen atoms. Look in the file which combinations are used here.

c) Go back to the "Term Tab". Here we will need to tell the terminal were to find lammps. Type `source ~/MolecularModelling/setup`

d) Run your first simulations by typing `lmp_serial < minimize.in`. You will see some output appearing on the screen. This is also saved in `log.lammps`. You might want to rename this file to make sure that you do not overwrite it the next time.

e) You can download the `min.lammpstrj` file and open this with VMD to look at the changes in structure during the minimization.

f) Open the Jupyter Notebook file `plotOutputLAMMPS.ipynb`. It shows an example of how to use `logplot.py` to data from the `log.lammps` file. You can change the labels to plot, for instance, the different energy contributions. (Look in `log.lammps` for the labels) Which individual component contributed most the decrease in energy? Compare this to what you observe in the structure. Crystals are stable because of their long-range interactions. Check whether the total non-bonded contributions are indeed negative.

g) We have so far used conjugate gradient. Perform a new minimization with steepest decent. Which one is faster? And which reaches the lowest energy?

h) For both type of minimization methods, the simulation cell remains the same. Changes of the simulation cell may help to further lower the total energy. With a simulated annealing type of set-up we can also vary the box size. Remove the comment sign `#` in front of

```
velocity all create 300.0 4928459
fix 1 all npt temp 300 10 50 tri 1 1 500
timestep 0.5
run 50000
```

   This runs a simulation in which the pressure is kept constant and the temperature goes down from 300 K to 10 K. The keyword `tri` for triclinc allows changes in all cell lengths and angles independently. You want to increase the number after `thermo` to 100 and do the same in the `dump` command. It will now print data every 100 steps instead of every step. Press simulate again. This will take a longer time and you might want to continue to the next exercise and return here later.

   Once the simulation is done, check the `log.lammps` file again. What happened to the cell dimensions? Is the monoclinic nature of the structure preserved? Look at the different energy contributions.

Exercise 5 Implementing a Minimizer in a Computer Program

Minimization typically involves repetitive evaluation of values and derivatives of possibly rather complex multivariable functions. Such task are well suited for computers. Writing an efficient minimizer for problems commonly encountered in chemical calculations can be considered a challenging programming task even for expert programmers. In this tutorial we will illustrate how to program minimizers for a simple quadratic function without thinking much about the memory efficiency or the execution speed of the code.

General Principles

It is relatively straightforward to write a computer program that minimizes one-dimensional functions using the steepest descent or Newton-Raphson methods. In general, such a program must have the following parts:

- The definition of the function to be minimized. For example, in the case of a diatomic molecule that obeys Hooke's law, the energy function for bond stretching is a quadratic equation with two parameters: the harmonic force constant and the equilibrium distance.
- The prescription for calculating analytical or numerical derivatives of the function with respect to coordinates. In the steepest descent method, only first derivatives are needed; in the case of the Newton-Raphson method both the first and the second derivatives are required.
- The prescription for updating the geometry. The formula for updating the geometry depends on minimization method.
- A loop that carries out iterations either until the minimum is found or until a maximum number of search steps has been taken. A criterion for what constitutes a minimum (e.g. force smaller than 0.01) must be provided.
- Statements that output the information about the progress of the calculation.
- Any statements that are specific for a particular language, such as `end` in Matlab.

Exercise 5.1 Steepest descent minimizer

The code below illustrates a Matlab implementation of the steepest descent minimizer for the one-dimensional harmonic potential. This would represent a diatomic molecule.

```
%
% Steepest Descent minimizer: Harmonic Potential
% Dr. Herma Cuppen based on a Python mimimizer by
% Dr. Kalju Kahn, Chem 126 (UCSB)
%

% Level one: short, ugly, and rigid

% Potential Function Parameters
k       = 2743.0;                       % Harmonic force constant
r_eq    = 1.1283;                       % Equilibrium distance

% Initial Search Point
r_ini   = 1.55;

% Steepest Descent search
r = r_ini;
step = 0.0001;
E = 0.5 * k * (r - r_eq)^2;             % Energy
F = - k * (r - r_eq);                   % Force   (- first derivative)
fprintf ('Step\tDistance\tEnergy\t\tForce\t\tHessian\n')
fprintf ('0\t%g\t\t%8.6g\t%g\n', r, E, F)      % Initial values
for i = 1:50                            % Iterate up to 50 time
   r = r + step*F;                      % Steepest Descent update
   E = 0.5 * k * (r - r_eq)^2;
   F = - k * (r - r_eq);
   fprintf ('%d\t%g\t\t%8.6g\t%g\n', i, r, E, F)% Print results
   if (abs(F) < 0.01)                   % Are we at the stationary point already?
```

```
25        break
26     end
27 end
```

You can find this code as `HarmOscSD1.m` in course files. Read it carefully to see if you understand the code. Examine the output and verify that the minimum has been found. Change the equilibrium distance and the force constant and check how this affects the number of iterations needed to find the minimum.