Tutorial 1A: Creating and simulating a box of water

Create a directory to work with the tutorial files (e.g., **mkdir ~/tutorial_1a**). This will create a directory called *tutorial_1a* in your home directory. We will use this naming convention for all of the workshop tutorials posted to the website.

For the purposes of this tutorial and all of the others that follow, we will additionally assume that:

(1) Your home directory is called */Users/name*
(2) TINKER has been installed in */Users/name/tinker*
(3) You have added */Users/name/tinker/bin* to your PATH environment variable

Modify these names and conventions as necessary!

Initial structure of water molecule

Use the molecule editor of your choice to save a PDB file of a single water molecule. One editor that we recommend is ***Avogadro*** (available at http://avogadro.cc/wiki/Main_Page). In ***Avogadro*** this task can be done in the following way:

a. Build → Insert → Fragment → water.cml
b. File → Save As → water.pdb (into the tutorial directory)

Using xyzedit to build a water box

In a terminal window, go to the relevant tutorial directory (e.g., **cd ~/tutorial_1a**).

Run the command ***pdbxyz*** to create a TINKER xyz file of the water molecule from the PDB file. This will bring up an interactive dialog. First, input the name of the PDB file (*water.pdb*). Next, input the name of the parameters file you would like to use; in this case we'll use the new 2013 AMOEBA parameters. Please note that this requires the full path to the parameters file (*/Users/name/tinker/params/amoebapro13.prm*). Now you will see two files: *water.xyz* and *water.seq*. The seq file is empty in this case and applies mainly to biomolecular simulations. The xyz file contains 3D coordinates and connectivity information for the water molecule.

Run the command ***xyzedit*** to create a box of water molecules. When prompted, input the appropriate xyz file (*water.xyz*) and the appropriate parameters file (same as above). Choose option 18 to create the box. You will now be prompted to input the number of molecules you want in the box (564) and box dimensions. In this case, we'll create a cubic box, but first we'll need to determine the appropriate dimensions.

At room temperature, water has a density of ~1 g/cm^3, but we want to convert this molecules per cubic Ångstrom (molecules/Å^3) so that we can determine an appropriate box size. Using the following formula, we find:

$$1\ \text{g/cm}^3 \times (10^{-8}\ \text{cm/Å})^3 \times (6.022 \times 10^{23}\ \text{molecules/mol}) \times 1/(18.015\ \text{g/mol}) = 0.0334\ \text{molecules/Å}^3$$

Therefore we need a box of volume 564 molecules/(0.0334 molecules/Å^3) = 16900 Å^3 to fit these molecules at the appropriate density. This implies a side length of 25.7 Å.

Inputting these side lengths, we obtain a new xyz file with 564 water molecules called *water.xyz_2*. **Note:** TINKER programs automatically append a numerical suffix to xyz files when they detect an existing xyz file with the same prefix name (e.g., *water.xyz*).

Energy minimization

In general energy minimization is a necessary prerequisite to running a molecular dynamics simulation. In this case minimization is absolutely essential because ***xyzedit*** built our water box without any regard to intermolecular spacing. Other more sophisticated programs (e.g., packmol) can build simulation boxes with a user-specified intermolecular spacing so as to avoid potential overlap problems. We review this option in later tutorials.

First, let's create a key file. Key files contain both force field parameters (or pointers to them) and simulation parameters. Using a text editor create a file *min.key* with the following lines:

```
parameters /Users/name/tinker/params/amoebapro13.prm

a-axis 27.2
b-axis 27.2
c-axis 27.2

neighbor-list
vdw-cutoff 9.0
vdw-correction
ewald
ewald-cutoff 7.0
pme-order 5
```

The first line of this key file specifies that we should use the AMOEBA 2013 parameters. The next three lines (a-axis, etc.) specify the size of the simulation box. **Note:** The box size here is 1.5 Å larger than the box size we initially calculated. This is because the water box was built with molecules that go right up to the edge of the box, and therefore we need to include a "buffer" of empty space to ensure that there are no overlapping molecules when we start our minimization (so that it remains numerically stable). This means that the density of the system will be below the desired density at the beginning of the simulation, but we will take care of that later.

The remaining lines of the key file specify that we should use:
(1) TINKER's parallel neighbor list
(2) a direct space van der Waals (vdW) cutoff of 9.0 Å
(3) the long-range vdW correction for interactions beyond the cutoff
(4) Ewald summation for long-range electrostatic interactions
(5) a direct space electrostatic cutoff of 7.0 Å
(6) 5^{th} order spline polynomials for the Ewald summation (a technical detail, but this number must be at least 5 or higher for simulations with multipole electrostatics)

Make a copy of *min.key* called *min_noelec.key*. This key file will do a "dumb" minimization (i.e., without electrostatics turned on) to clean up the initial box created by xyz. Why are we doing this? If atoms are too close to each other in space, the induced dipole solver will not be able to

converge. Minimizing without electrostatics turned on will enable molecules with overlapping atoms to get out of each other's way. Add the following lines to *min_noelec.key*:

```
chargeterm none
mpoleterm none
polarizeterm none
```

In the interests of keeping things organized, make a subdirectory for the minimization files (e.g., ***mkdir minimization***) and copy both of the key files to that directory (e.g., ***cp min*.key minimization***). Then copy *water.xyz_2* to that subdirectory with the new name *min_noelec.xyz*.

Changing into the minimization directory, we can now run the energy minimization from the command line:

minimize min_noelec.xyz –k min_noelec.key 10.0
cp min_noelec.xyz_2 min.xyz
minimize min.xyz –k min.key 1.0

The last number in the minimize syntax is the energy convergence criterion. In this case the no-electrostatics minimization will terminate when it reaches an RMS gradient of 10.0 kcal/mol/Å per atom, while the minimization with the full potential energy function will terminate at 1.0 kcal/mol/Å. The latter number is safe from a numerical stability point of view. Note that we made a copy of the first minimization's output xyz file to be the input xyz file for the second minimization.

Heating

Now we want to bring our simulation box up to the desired temperature. In this case we'll run the simulation at room temperature (298.15 K). Create a new subdirectory in your tutorial directory for this part of the simulation, and then create a key file in that subdirectory called *heat.key* with the following content:

```
parameters /Users/name/tinker/params/amoebapro13.prm

a-axis 27.2
b-axis 27.2
c-axis 27.2

archive

neighbor-list
vdw-cutoff 9.0
vdw-correction
ewald
ewald-cutoff 7.0
pme-order 5
polar-predict

thermostat bussi
tau-temperature 0.1

integrator verlet
```

Note that many of the same terms are present in this key file. We see the appearance of a few new terms:

(1) "archive" – this is a trajectory file that contains a series of concatenated xyz files (this file will have the extension arc)
(2) "polar-predict" – this uses a predictor-correcter algorithm to predict the induced dipoles at each time step based on the last several time steps (resulting in a simulation speed-up of ~10%)
(3) "thermostat bussi" – this activates the Bussi thermostat (a successor to the Berendsen thermostat with a random noise component to ensure a statistically correct ensemble)
(4) "tau-temperature 0.1" – this sets the time constant for the thermostat to 0.1 ps
(5) "integrator verlet" – this uses a velocity Verlet integration algorithm

As before, let's copy the most recently created xyz file (*min.xyz_2*) to this subdirectory with the new name *heat.xyz*. Run the NVT simulation with the following command:

dynamic heat.xyz –k heat.key 5000 1.0 0.1 2 298.15 > heat.out

The numbers after the key file name are important. In order, they are:

(1) 5000 – the number of MD steps to be performed (i.e., 5000 steps)
(2) 1.0 – the length of the MD time step in femtoseconds (i.e., 1.0 fs)
(3) 0.1 – how frequently to write out the MD trajectory and energies in picoseconds (i.e., once every 0.1 ps or 100 steps)
(4) 2 – what type of thermodynamic ensemble to run the simulation in (i.e., 2 = NVT)
(5) 298.15 – the desired temperature for the simulation in Kelvin (i.e., 298.15 K)

In addition to the trajectory (arc) file, the simulation will also generate a restart (dyn) file that contains the final positions, velocities, and accelerations of the system at the conclusion of the simulation. This can (and should!) be used as a starting point for additional simulations.

Equilibration

Now that our system is up to temperature, we can bring it to the correct pressure/density. By now you've gotten the hang of the organization scheme we're using. Create an appropriate subdirectory and a new key file called *eq.key* with the following content:

```
parameters /Users/name/tinker/params/amoebapro13.prm

archive

neighbor-list
vdw-cutoff 9.0
vdw-correction
ewald
ewald-cutoff 7.0
pme-order 5
polar-predict

thermostat bussi
tau-temperature 0.1
```

```
barostat berendsen
tau-pressure 1.0

integrator verlet
```

This key file contains only two new terms:
(1) "barostat berendsen" – this activates the Berendsen barostat (similar in principle to the Berendsen thermostat)
(2) "tau-pressure 1.0" – this sets the time constant for the barostat to 1.0 ps (a fairly common value)

Note that we also were able to exclude the a-axis, b-axis, and c-axis statements. This is because the dimensions of the simulation box are read from the restart (dyn) file.

Copy two files (*heat.xyz* and *heat.dyn*) from the heating simulation directory to this one and give them appropriate names (*eq.xyz* and *eq.dyn*). Run the NPT simulation with the following command:

dynamic eq.xyz –k eq.key 25000 1.0 0.5 4 298.15 1.0 > eq.out

Note that the fourth number after the key file is "4", signifying an NPT simulation. There is also a sixth number (i.e., 1.0) that signifies the desired pressure of the simulation in atmospheres.

This simulation will take quite a bit more time to run. On a relatively new 27" iMac (4-core Intel Core i5 3.2 GHz), this took ~35 minutes to complete. We have provided output files on the workshop's GitHub repo if you do not want to wait this long!

Let's analyze the output file to monitor the convergence of the density and potential energy of the system. On the command line, input the following two commands:

grep "Density" eq.out | awk '{ print $2 }' > density.txt
grep "Current Potential" eq.out | awk '{ print $3 }' > potential_energy.txt

The ***grep*** command outputs lines in a file (or files) that contain the expression in quotes. (Strictly speaking, quotes aren't necessary if the expression is a single word, number, etc.) The ***awk*** command is quite versatile in parsing text files; here we've used it to print out either the 2nd (for density) or 3rd column (for energy) of the grep output. Plot the resulting files, *density.txt* and *potential_energy.txt*, using the tool of your choice and examine the plots for evidence of convergence. You should see that both the energy and density of the system reach their equilibrium values within ~10 ps. (It makes sense that these quantities are correlated for a simple homogeneous system like neat water.)

Computing the enthalpy of vaporization

In addition to ensuring that our simulation converges to the physically correct density for liquid water at 298.15 K, you may also be curious about the accuracy of the intermolecular interactions. One way to measure this is to use the potential energy information of our liquid simulation combined with the potential energy data from a gas phase simulation to calculate the enthalpy of vaporization.

To run a gas phase simulation of water, make a new subdirectory *gas* and copy the original *water.xyz* file to that directory. Create a new key file (*heat_gas.key*) that contains just a few lines:

```
parameters /Users/name/tinker/params/amoebapro13.prm

archive

thermostat bussi
tau-temperature 0.1

integrator verlet
```

Run the gas phase simulation:

dynamic water.xyz -k heat_gas.key 1000000 1.0 0.5 2 298.15 > heat_gas.out

This simulation should run very quickly, even on modest hardware. Extract the potential energies from *heat_gas.out* as before.

With the gas phase potential energies and the liquid phase potential energies, we can now calculate the enthalpy of vaporization as:

$$\Delta H_{vap} = U_{gas} - U_{liquid} + RT$$

where U_{gas} is the gas phase potential energy, U_{liquid} is the liquid phase potential energy per molecule, R is the universal gas constant, and T is the temperature. The experimental value for the enthalpy of vaporization of water at 298.15 K and 1.0 atm is 10.52 kcal/mol. See how your simulation data compare! (Be careful to only consider data from the equilibrated parts of your simulations.)

Radial distribution function

One final analysis we can do to check if our water box is physically reasonable is to calculate the radial distribution function (rdf) for the O-O (or O-H or H-H) distances in the water. This will give us some insight into the structure of our bulk water at the molecular scale. This analysis is relatively simple to execute:

radial eq.arc 101 250 1 O O 0.05 N > O-O_rdf.txt

Let's unpack this syntax a bit. As specified, this syntax has the ***radial*** program analyze the trajectory file *eq.arc* from frame 101 to 250, not skipping any frames. It will compute the rdf between atom types O and O, use a bin width of 0.05 Å for the resulting histogram, and ignore any intramolecular O-O pairs (although there aren't any here). (As always, these options can be accessed interactively.) You will need to clean up the output in *O-O_rdf.txt*, but plot the "Raw g(r)" column vs. the "Distance" column using the tool of your choice and compare your plot to experimental data (e.g., http://pubs.acs.org/doi/abs/10.1021/cr0006831).