AMOEBA advanced potential energies workshop

Exercise 1 - Getting started with Tinker

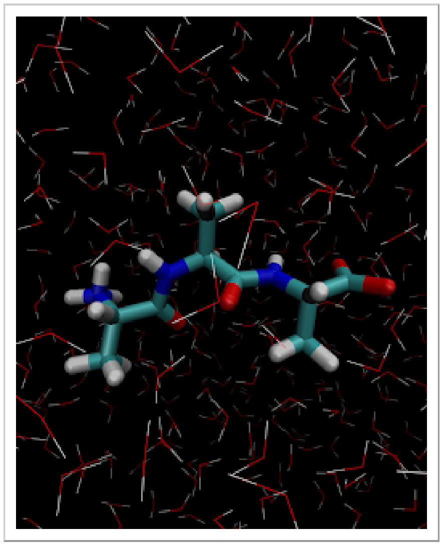
Contents

Why Tinker?

Tinker is a freely available molecular modelling package, primarily authored by Jay Ponder's lab at the University of Washington in St Louis. It has utilities for all sorts of biomolecular simulations, and includes parameters for many of the standard biomolecular force fields, but for learning how to use AMOEBA simulations it has 3 main advantages:

1. It's the canonical AMOEBA code - other software packages may be faster (more on that later!) but their implementations of AMOEBA will follow that of Tinker
2. As the canonical code, Tinker includes the latest AMOEBA parameter and methodology developments, for testing and investigating the effects of polarisation in different systems
3. It's free!

In this first tutorial we'll introduce the basic Tinker utilities for setting up and running a molecular dynamics simulation with AMOEBA, along with the file formats Tinker uses for topology, coordinates, and parameters of simulated systems. We'll be using a simple Ala_3 peptide in water as our system of interest, but the concepts are similar for any biomolecule.



Setting up Ala$_3$

Logging in

Log into Lyceum2, and type `ls`. You should see a folder called 'amoebaworkshop'. Change into it:

```
cd amoebaworkshop
```

Inside there will be many more folders, each corresponding to different exercises (you can see these if you type `ls` again). For now, change into the 'Exercise_1' folder, and list the files inside:

```
cd Exercise_1
ls
```

Of course, the first thing we need to set up a molecular dynamics simulation is an initial structure of the system we want to simulate. This might be, for example, an X-ray crystal structure of a protein from the PDB, properly protonated and solvated. In this case we've provided an initial file, `AAA_solv.pdb`. This is a linear Ala$_3$ peptide, which for the sake of speed we've already created and solvated in a box with a 10.0 Å buffer of water on all sides. There are many software packages that can perform this sort of set up, and of course Tinker has its own utilities too (see side note 1).

File formats

Tinker makes use of two main file formats to run its simulations:

1. **The 'xyz' file:** Contains the coordinates and topology of all the atoms in the system
2. **The 'key' file:** Contains the simulation conditions and input options, along with parameters for atoms, or paths to find them

Side note 1 - Tinker protein set up

Tinker has its own utilities for creating linear peptide and nucleic acid chains using the `protein` and `nucleic` tools, which you can run interactively to build initial structures in Tinker format. These can then be solvated with the `xyzedit` program (for details see exercise 2). However, for the most part, you might be interested in using initial structures (e.g. crystal structures) that already exist - in which case `pdbxyz` is the way to go. `Pdbxyz` is aware of all standard amino acids, nucleic acids, water and mono/divalent ion names and will convert and assign atom types to Tinker format, adding hydrogen atoms to crystal structures where necessary. Problems with `pdbxyz` are most often caused by discrepancies between atom names in different programs - take care with this in your own simulations.

We'll start by making a basic key file. Open up a new file called `PDB_to_xyz.key` in your favourite text editor - I use vi (e.g. type `vi PDB_to_xyz.key`). Add the following into the empty text file, and save it:

```
parameters
/home/***Enter_your_username_here***/tinker/params/amoebapro13.prm

a-axis 36.281
b-axis 33.399
c-axis 30.222
```

The first line (be sure to enter the temporary Southampton username you've been given correctly...) gives Tinker the location of the parameter file we want to use for this simulation. Tinker is packaged with a variety of force fields, here we'll use the Amoeba protein parameters of 2013 (named `amoebapro13.prm` in the Tinker '`params`' directory).

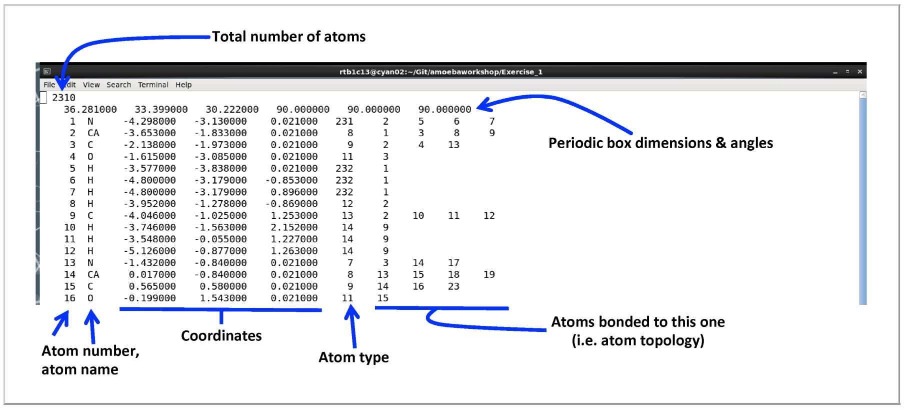
The next three lines (starting `a-axis`, `b-axis` & `c-axis`) are the X, Y & Z box dimensions in Ångstrom of the periodic system. We've taken these straight from the CRYST records at the top of the PDB file.

Converting a PDB to Tinker format

Now we have a basic key file we can use it to create a Tinker xyz file from our initial PDB file. Tinker has a built-in program to do this, called `pdbxyz`. Back in the terminal window, type:

```
pdbxyz AAA_solv.pdb -k PDB_to_xyz.key
ls
```

You'll see that two new files have been created, `AAA_solv.seq` and `AAA_solv.xyz`. Have a look at the `.seq` file first, again in your favourite text editor. You'll see it just contains the line "`1  ALA ALA ALA`". This is the peptide sequence that `pdbxyz` has interpreted and converted. Next, open up `AAA_solv.xyz`. It'll have lines that look like this:


Total number of atoms
rtb1c13@cyan02:~/Git/amoebaworkshop/Exercise_1
File Edit View Search Terminal Help
Periodic box dimensions & angles
Coordinates
Atom number, atom name
Atom type
Atoms bonded to this one (i.e. atom topology)

```
 2310
   36.281000   33.399000   30.222000   90.000000   90.000000   90.000000
     1  N     -4.298000   -3.130000    0.021000   231     2     5     6     7
     2  CA    -3.653000   -1.833000    0.021000     8     1     3     8     9
     3  C     -2.138000   -1.973000    0.021000     9     2     4    13
     4  O     -1.615000   -3.085000    0.021000    11     3
     5  H     -3.577000   -3.838000    0.021000   232     1
     6  H     -4.800000   -3.179000   -0.853000   232     1
     7  H     -4.800000   -3.179000    0.896000   232     1
     8  H     -3.952000   -1.278000   -0.869000    12     2
     9  C     -4.046000   -1.025000    1.253000    13     2    10    11    12
    10  H     -3.746000   -1.563000    2.152000    14     9
    11  H     -3.548000   -0.055000    1.227000    14     9
    12  H     -5.126000   -0.877000    1.263000    14     9
    13  N     -1.432000   -0.840000    0.021000     7     3    14    17
    14  CA     0.017000   -0.840000    0.021000     8    13    15    18    19
    15  C      0.565000    0.580000    0.021000     9    14    16    23
    16  O     -0.199000    1.543000    0.021000    11    15
```

As you can see, the xyz file contains the coordinates, topology, and atom type information Tinker needs to describe the system under investigation. We can now use this file to actually start some simulation...

Energy minimisation

Defining simulation conditions in a key file

The first step in any MD protocol is to relax the initial structure to remove any steric clashes or poor configurations arising from our setup/solvation procedure. Tinker includes multiple different minimisation algorithms, but here we'll use the program `minimize` with the rapid but simple steepest descent algorithm.

Start by making a new copy of both our xyz file and key file, for tidiness' sake:

```
cp PDB_to_xyz.key AAA_min.key
cp AAA_solv.xyz AAA_min.xyz
```

Then, edit `AAA_min.key` so it contains the following (the comments, starting with `#`, aren't necessary):

```
parameters /home/***Enter_your_username_here***/tinker/params/amoebapro13.prm

a-axis 36.281
b-axis 33.399
c-axis 30.222

openmp-threads 4           # No. of parallel CPU threads to use

cutoff 9.0                 # Nonbonded interactions direct cutoff
ewald                      # Switch on PME
ewald-cutoff 7.0           # PME real space cutoff (overrides the 9A above)
vdw-correction             # Switch on analytical long-range vdW correction
polar-eps 0.01             # Dipole convergence criterion in RMS Debye/atm

neighbor-list              # Use pairwise neighbor list for calculating
                           # nonbonded interactions (improves speed)

maxiter 2000               # Maximum number of minimisation steps
steepest-descent           # Use the SD minimisation algorithm

printout 100               # Interval at which to print out energies
```

We've added some keywords defining the type of simulation we want to perform - a steepest descent minimisation of maximum 2000 steps, with a 9.0 Å vdW cutoff, 7.0 Å real space electrostatics cutoff, and PME and an analytical correction for long range electrostatics and vdW respectively. Additionally we've specified (with the `polar-eps` keyword) that at every step we wish our induced dipoles to be iterated until the RMS change in atomic dipoles is less than 0.01 Debye/atom. This is a fairly ***loose*** convergence criterion, and is at the limits of conserving energy in an AMOEBA MD simulation, but we use it here for speed. In practice it's preferred to use a convergence criterion of 1e-5 Debye/atom or below.

Running a minimisation job

To run this minimisation we'll need to submit a job to the Lyceum cluster. A script to do so has been ready prepared for you, called `AAA_min.x`. Open it up, and you'll see the command line used to run a Tinker minimisation, along with some comments describing how it's used:

```
# Command for using Tinker minimize:
# minimize [xyz file] -k [key file] [gradient convergence criterion]

minimize AAA_min.xyz -k AAA_min.key 0.01 > AAA_min.out
```

The gradient convergence criterion here is NOT the same as the dipole convergence criterion above. It is the RMS gradient in the energy at which we wish to stop the minimisation. If this gradient is reached before the maximum number of minimisation steps have been performed, the minimisation will stop, as the system is well minimised anyway. 0.01 kcal/mol/A per atom is the default value, and we use it here.

Change any filenames in `AAA_min.x` to be correct for the xyz file and key file you have, then submit the job to the cluster:

```
qsub AAA_min.x
```

This will take about 3 minutes to run. You can have a look at the output in real time by following `AAA_min.out`, e.g. `tail -f AAA_min.out`

Once complete, it's time to run some molecular dynamics...

Heating to 300K - NVT simulations

Additions to the key file for dynamics

Once your minimisation job is complete, you'll have a new file in your directory: `AAA_min.xyz_2`. This is the final structure of the solvated Ala$_3$ system after the minimisation. By default, Tinker appends output files with sequential numbers, so if you wanted to start another minimisation from `AAA_min.xyz_2`, the output would be called `AAA_min.xyz_3`. However, this might get very confusing very quickly, so instead we'll rename our output file to something more sensible, and make a new key file for the next simulation:

```
mv AAA_min.xyz_2 AAA_eq_nvt.xyz
cp AAA_min.key AAA_eq_nvt.key
```

As you might have guessed our next simulation will involve heating and equilibrating the system at 300 K, under constant volume (NVT) conditions. Open up the `AAA_eq_nvt.key` file and edit it to look like the following. There are only a few changes that need to be made at the bottom of the file:

```
parameters /home/***Enter_your_username_here***/tinker/params/amoebapro13.prm

a-axis 36.281
b-axis 33.399
c-axis 30.222

openmp-threads 8

cutoff 9.0
ewald
ewald-cutoff 7.0
vdw-correction
polar-eps 0.01

neighbor-list

printout 500

thermostat andersen        # Switch on Andersen thermostat
integrator verlet          # Use a velocity-Verlet integrator

archive                    # Create a single trajectory file with all
                           # MD snapshots concatenated in sequence
```

Notice we've removed the 'maxiter' and 'steepest-descent' keywords as we're no longer performing a minimisation. Instead they've been replaced with keywords to set a thermostat and integrator. Finally the 'archive' keyword will concatenate all of our output files (snapshots of the simulation coordinates) into a single trajectory. Otherwise Tinker prints out every frame as a separate, sequentially numbered file, as it did with the minimisation output above.

How to use 'dynamic', the MD engine

Next, we'll submit an MD job to the Lyceum cluster - the script to do so is called AAA_eq_nvt.x. It contains the following command line, which uses the program dynamic, Tinker's main MD engine:

```
dynamic AAA_eq_nvt.xyz -k
AAA_eq_nvt.key 5000 1.0 0.5 2 300 >
AAA_eq_nvt.out
```

Side note 2 - interactive running

All Tinker programs, minimize and dynamic included, are also set up to run and receive input interactively. Simply running 'dynamic' from the command line will prompt the user for all required input data, from input filenames to choice of ensemble to temperatures and pressures. This will also happen if only a partial command line is used, e.g. if you want to run an NVT simulation but forget to specify a desired temperature.

The numbers following the usual xyz and key file inputs correspond to the following:

1. The number of MD steps to perform (here 5000 steps)
2. The timestep, in fs (here 1.0 fs)
3. The time interval to print out coordinate information, in ps (here 0.5 ps, or 500 steps)
4. The type of simulation to run/which ensemble to run in (here 2 = NVT)
5. The desired simulation temperature, in K (here 300 K)

Change any of the filenames in AAA_eq_nvt.x to be correct for your own system, and submit the job to Lyceum:

```
qsub AAA_eq_nvt.x
```

This should take about 6-7 minutes to run.

Equilibrating to 1 atm - NPT simulations

Restarting a simulation

Open up the output file (AAA_eq_nvt.out) from the heating simulation. The last few lines should include a section like this:

```
Average Values for the Last   500 Out of     5000 Dynamics Steps

Simulation Time            5.0000 Picosecond
Total Energy           -4860.8933 Kcal/mole   (+/-   5.4696)
Potential Energy       -6761.2395 Kcal/mole   (+/-  31.1456)
Kinetic Energy          1900.3462 Kcal/mole   (+/-  28.3130)
Temperature                275.99 Kelvin      (+/-     4.11)
Pressure                  -770.24 Atmosphere  (+/-   533.81)
Density                    0.6305 Grams/cc    (+/-   0.0000)
```

This file contains the average and instantaneous system energies, temperatures, pressures etc. If this were a real simulation you may wish to visually check equilibration by writing a small script to extract and plot the progress of these properties over time. From the above we can already see this NVT simulation is too short, as the system temperature hasn't quite reached 300 K. We could change the collision frequency of the Andersen thermostat to speed this up (using the 'collision' keyword - see the Tinker manual), but for now we'll carry on with a pressure equilibration just to demonstrate how it works.

Type ls and you'll see a couple more new files have been created as outputs from the simulation. The first, AAA_eq_nvt.arc, is the 'archive' file, containing all the coordinates of snapshots printed out during the trajectory as a single file (remember, this is why we specified the 'archive' keyword in AAA_eq_nvt.key). The second, AAA_eq_nvt.dyn, is the 'dynamics' file and contains the current atomic positions, velocities, box dimensions and accelerations at the last trajectory snapshot. The dyn file is therefore the way to perform a (non-binary) restart of a Tinker MD simulation:

If dynamic finds a dyn file with the same prefix as the specified starting xyz file, then coordinates, velocities and box dimensions are restarted from the dyn, overriding anything specified in other inputs

So, in theory, we could use exactly the same command line for dynamic as we did above and the simulation would continue for another 5 ps from where it left off, as Tinker would recognise the existence of the dyn file. However, for tidiness' sake we will again make copies of our inputs and start our NPT equilibration from there.

Running under constant pressure

Start by making copies of the key file and dyn file:

```
cp AAA_eq_nvt.key AAA_eq2_npt.key
cp AAA_eq_nvt.dyn AAA_eq2_npt.dyn
```

And now add the following to the bottom of the `AAA_eq2_npt.key` key file:

```
barostat berendsen          # Switch on Berendsen barostat
tau-pressure 1.0            # Set pressure coupling time to 1.0 ps
```

There are other, rigorous, barostats (e.g. Monte-Carlo, Nose-Hoover) available in Tinker, but for simplicity we'll use the Berendsen one here.

Next we'll extract the last frame of the NVT trajectory to use as the input of the NPT simulation. Again, thanks to the dyn file we don't *need* to do this, but it is sensible for record-keeping and illustrates how to manipulate trajectories. The Tinker `archive` program is used to do this, and we'll use it interactively as suggested in Side Note 2. Type:

```
archive AAA_eq_nvt.arc
```

...and follow the on-screen instructions. Here we want option 2, to extract an individual frame. You'll then be asked to select the first and last frame to extract, and the stride between them. Here we want the 10th frame only (the last in our 5 ps trajectory), so type `10 10 1` at the prompt. Once complete, press Enter to quit. Type `ls` again and you'll notice a new file has been created, `AAA_eq_nvt.010`. This is the 10th frame of the `AAA_eq_nvt.arc` archive file - the one we want. Rename it as the input for our upcoming NPT simulation, **making sure it has the exact same prefix as the dyn file we want to restart from**:

```
mv AAA_eq_nvt.010 AAA_eq2_npt.xyz
```

Finally we're ready to run. The script to submit is `AAA_eq2_npt.x`, inside you'll find the command line:

```
dynamic AAA_eq2_npt.xyz -k AAA_eq2_npt.key 5000 1.0 0.5 4 300 1.0 >
AAA_eq2_npt.out
```

Notice there are two changes from the NVT simulation. First we've changed from ensemble option 2 (NVT) to ensemble option 4 (NPT). Second, there is a final number on the command line specifying the pressure we want to run at, in atmospheres (here 1.0 atm). Make any filename changes you need to and submit the job:

```
qsub AAA_eq2_npt.x
```

This should again take about 6-7 minutes to run. Once complete, have a look at the outputs again - you should see the box dimensions and density begin to change as the barostat does its job.

Extras:

Visualising output

Obviously with these short simulation times our system is nowhere near equilibrated just yet. However, we can still take a look at our trajectories to ensure there are no clashes or unphysical structures created during the dynamics so far. VMD is often used for the visualisation of molecular dynamics simulations, and we'll use it here too.

However, VMD currently (version 1.9.2) has a small bug, in that it will only correctly visualise Tinker archive files that **don't contain any periodic box information**, as older versions of Tinker did not include box dimensions in their archive files. If the periodic boundaries are not crucial for your analysis, you can strip the trajectories of box information with a simple `sed` command, e.g.

```
sed '/90.000000\ \ \ 90.000000\ \ \ 90.000000/d' AAA_eq2_npt.arc >
AAA_eq2_stripped.arc
```

If you wish, strip the trajectories of their box information, copy the archive files (`*.arc`) back to your local machine using WinSCP as detailed in the introduction, and visualise in VMD using the file type 'Tinker'.

Analysing simulations

Tinker comes with a whole suite of programs for analysing Tinker xyz or trajectory files, the most useful of which is probably the '`analyze`' program. You can run analysis interactively simply by feeding it an input structure and key file (e.g. `analyze AAA_eq2_npt.arc -k AAA_eq2_npt.key`) and following the on screen instructions. Alternatively, many other analysis packages, such as MDTraj, Amber cpptraj, or Parmed, are Tinker-aware and can read and manipulate Tinker coordinate files to varying extents. With a little scripting, this allows a diverse range of analysis of AMOEBA simulations beyond what's available in Tinker. More on that, however, in Exercise 3.

註解

您沒有新增註解的權限。