AMOEBA advanced potential energies workshop

Search this site

Navigation

Home
Intro & setup
Exercise 1 - Getting started with Tinker
Exercise 2 part 1 - Parameterising a new molecule (electrostatics)
Exercise 2 part 2 - Parameterising a new molecule (valence & vdW)
Exercise 3 - Analysing trajectories and calculating free energies
Sitemap

Exercise 3 - Analysing trajectories and calculating free energies

Contents

Analyses in Tinker

As with the setup and dynamics tools, Tinker comes packaged with a variety of programs to perform specific analyses of generated structures or trajectories. These range from performing molecular superpositions & calculating RMSDs (`superpose`), to calculating radial distribution functions (`radial`), to calculating a power spectrum for a system from the velocity autocorrelation function (`spectrum`). Many analyses can be performed using the `analyze` program, which includes ways to evaluate both properties of the system coordinates/energies, and the system parameters (e.g. printing out a breakdown of potential energy by individual components for specified atoms, evaluating overall molecular dipole moments, calculating radius of gyration etc.). We'll demonstrate a couple in this exercise, and feel free to investigate more if there's time.

The main analysis we're interested in for our free energy calculations is the calculation of free energy differences between the adjacent lambda windows. This makes use of the Tinker `bar` program and will calculate both standard FEP and BAR free energies between any two trajectories and any two key files/parameter sets. More on that later - first, let's check our simulations have worked.

Analysing with `analyze`

Analysing energetics

Log back into Lyceum (if you haven't done so already) and change into the directory for `Exercise_3`. Inside we've provided an example set of trajectories, in both solution phase and gas phase, that were run using the inputs generated in exercise 2. Each individual simulation at each lambda window is available in subdirectories within the `solution` and `gas` folders. We'll run a couple of analyses on the trajectories from the starting point of the simulation, the 'completely on' solvated structure of ethanol. Change into the relevant directory:

```
cd solution/ele1.0
```

Inside you'll find all the inputs we created in exercise 2, along with an `ethanol_run.arc` trajectory, `ethanol_run.dyn` restart file and `ethanol_run.out` output file. As a first basic use of `analyze`, lets check the input & output structures for any unusual clashes. Run `analyze` on the input structure interactively:

```
analyze ethanol_run.xyz -k ethanol_run.key
```

You'll see a variety of options:

```
The TINKER Analysis Facility can Provide :

 General System and Force Field Information [G]
 Force Field Parameters for Interactions [P]
 Total Potential Energy and its Components [E]
 Energy Breakdown over Each of the Atoms [A]
 List of the Large Individual Interactions [L]
 Details for All Individual Interactions [D]
 Electrostatic Moments and Principle Axes [M]
 Internal Virial, dE/dV Values & Pressure [V]
 Connectivity Lists for Each of the Atoms [C]
```

```
 Enter the Desired Analysis Types [G,P,E,A,L,D,M,V,C] :
```

Any of these options can be chosen individually or in combination with one another. In this case we'd like to check our file for any structural/energetic clashes. Choose option 'L', which will list any pairwise atomic interactions larger than a default 5 kcal/mol for bonded interactions, or 10 kcal/mol for nonbonded. As an output you should see one interaction listed, a vdW contact between a water hydrogen and adjacent water oxygen atom, with a distance of around 1.66 Å. Although short, this is noting too unusual for a transient hydrogen bond. We'll perform the same analysis on the trajectory as a whole, redirecting the output to a new file:

```
analyze ethanol_run.arc -k ethanol_run.key l &> ethanol_large_interactions.out
```

If you open the resulting `ethanol_large_interactions.out` file in a text editor and browse though, you'll see a few water-water contacts similar to that above, but nothing worry inducing. This, of course, is a fairly basic analysis of simulation stability but demonstrates how analyze can be performed on both single structures and a whole trajectory. Other Tinker-aware analysis packages are available, see side note 1.

Before we begin our BAR free energy calculation we'll check our simulations have been run with properly scaled parameters.

Side note 1 - Analysis packages

Beyond the programs available in Tinker, analysis of Tinker archive files can be performed in other packages too. The cpptraj program in Amber has a wealth of available analyses, the Python package MDTraj will read Tinker archive files and can be used to manipulate sets of coordinates accordingly, and of course VMD may be used to visualise results and carry out other analyses. Many trajectory parsers only read the coordinate information, not the bonding information, from Tinker xyz files however, so a separate topology file may be needed to define how atoms are connected. For biomolecules this can often simply be a PDB, as amino acid residues have defined topology. You can create one using the Tinker `xyzpdb` program, for example.

Parameter scaling with lambda

Let's check that we didn't make any typos in our inputs and that the parameters of ethanol have been properly scaled with lambda in our different simulations. Still in the ele1.0 folder, run analyze interactively on the input structure and choose option P:

```
analyze ethanol_run.xyz -k
ethanol_run.key
```

Choose option 'P' when prompted, and enter the atoms 1 to 9 for output, as these are the atoms in the ethanol molecule **(Remember:** you can select a range of atoms in Tinker by specifying the first number as negative, e.g. '`-1 9`'). All the atomic parameters will be printed to screen. Now, re-perform the same analysis using the key file used for the **next** lambda window:

```
analyze ethanol_run.xyz -k
../ele0.9/ethanol_run.key
```

You should see how the multipole values and dipole polarisabilities (alpha) have been modified between the two simulations, and reduced to 0.9 of their initial values. If you wish (and if there's time) you can perform the same analysis with one of the lambda windows for vdW decoupling, however, remember that **the decoupling step does not scale the raw parameters, instead it scales the solute-solvent interactions**. As such, you should check for differences in the vdW energy between key files, not the vdW parameters (radius & epsilon) themselves.

Once you're happy to move on to the BAR analysis stage, carry on below.

BAR free energy estimates

Between lambda windows

We don't have time to delve deep into the theory of the calculations we're about to perform, but for those of you familiar with alchemical free energy calculations, the below equations should look familiar. For those of you who aren't, there are a number of excellent published reviews and introductions to the theory of free energy calculations, but the Alchemistry wiki is a good enough place to start. For today's purposes though, there are two main things to take from the below equations:

Exponential averaging:

$$\Delta A_{0->1} = -k_B T \ln \left\langle \exp[-\beta(U_1 - U_0)] \right\rangle_0$$

BAR:

$$\Delta A_{0->1} = k_B T \ln \frac{\left\langle f(U_0 - U_1 + C) \right\rangle_1}{\left\langle f(U_1 - U_0 - C) \right\rangle_0} + C$$

...where:

$$C = \Delta A_{0->1} - k_B T \ln \frac{n_0}{n_1}$$

Namely, to evaluate the free energy difference between state ***1*** and state ***0***, we need the potential energies at both states (***U_1***,***U_0***), evaluated over an ensemble average of structures from those states (***$\langle ... \rangle_0$***,***$\langle ... \rangle_1$***). It should be clear we already have the tools we need to do this - the key files for each lambda window describe the potential energy function for a given state, and the simulation structures represent the ensemble averages at each desired state. All we need is a program to evaluate the above expressions!

Fortunately, Tinker has this, in the shape of the bar program. Return to the `solution/ele1.0` directory we were analysing our simulations in above, and run the `bar` program interactively (as usual, just type `bar` on the command line). You'll be presented by a series of prompts that that will correspond to, in turn:

1. The coordinate file of state ***0*** (`ethanol_run.arc`)
2. The frames of state ***0*** you wish to analyse as part of the ensemble average (frames 1 to 200, at every step: `1 200 1`)
3. The temperature at state ***0*** (`300`)
4. The coordinate file of state ***1*** (`../ele0.9/ethanol_run.arc`)
5. The frames of state ***1*** you wish to analyse as part of the ensemble average (frames 1 to 200, at every step: `1 200 1`)
6. The temperature at state ***1*** (`300`)

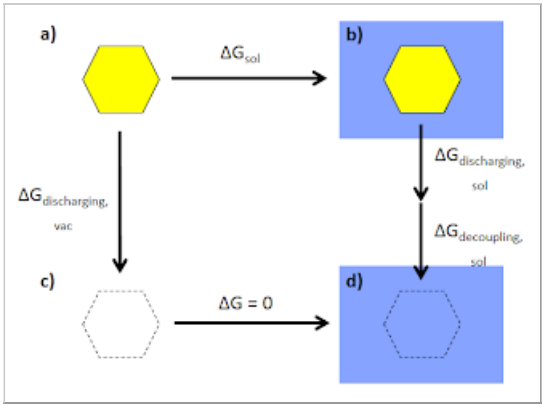
Tinker automatically finds any key files with the same prefix as you use for the xyz files, so reads in `ethanol_run.arc` with `ethanol_run.key` and `../ele0.9/ethanol_run.arc` with `../ele_0.9/ethanol_run.key`.

Once you've entered all the inputs, after a few seconds you should see some results pop up. These are the forward & backward FEP (i.e. exponential averaging) free energies, and the BAR free energy, between lambda windows 1.0 and 0.9 of the discharging step.

As you can imagine, running this interactively between every lambda window would be very time consuming, so instead we'll use a script for the rest of the free energy calculations.

Overall hydration free energy

As a reminder, the thermodynamic cycle we use to divide the hydration free energy up into steps is as follows:



This means that the overall hydration free energy, ΔG_{sol}, is given by:

$$\Delta G_{sol} = \Delta G_{discharging,vac} - \Delta G_{decoupling,sol} - \Delta G_{discharging,sol}$$

A script has been prepared to run all of the bar analyses and write the results to separate files, and is available as `BAR_analysis.x` in the main `Exercise_3` directory. Submit this to Lyceum with:

```
qsub BAR_analysis.x
```

It should take about 15 minutes to run, and if you want to see the underlying script, it's available as `run_bar.py`. It's very basic, and is just a loop over multiple calls of `bar`, e.g.:

```
bar ethanol_run.arc 1 200 1 300 ../ele0.9/ethanol_run.arc 1 200 1 300 >
sol_ele1.0-0.9_bar.out
```

Once complete, you'll have multiple files of the names "`[sol/gas]_[ele/vdw][lambda0]-[lambda1]_bar.out`", each of which contains free energy estimates calculated with FEP/exponential averaging, BAR and a re-estimation of the BAR free energy from bootstrapping the underlying ensemble. The free energy for the discharging/decoupling steps is simply the sum of the individual free energies for each of the lambda windows in that step, and the overall free energy of hydration can then be calculated from the equation above (be sure to get the signs correct!)

Try to have a go at this by extracting the data from the `bar` output files (hint, you may wish to `grep` or `awk` out the individual free energies and associated error bars into a more amenable format first, e.g.:

```
awk '/BAR\ Bootstrap\ Free\ Energy/ {print $5,$7}' sol_ele* >
sol_ele_energies.txt
```

Or if there's little time then summary files are available in the `outputs` folder. If you've summed the individual steps correctly then you should end up with a ΔG_{sol} of around -6.1 kcal/mol, compared with the experimental value of -5.0 kcal/mol, and a value estimated with the GAFF fixed charge force field of -3.45 kcal/mol.

While this agreement with experiment is good, it's not perfect, and may reflect the short simulation times and short equilibration times we've used here. If you wish to explore further then the length of the simulations would be a good place to start, as would the sensitivity of the free energies to small changes in parameterisation protocol, or even empirical scaling of the -OH group multipoles. For now, though, we hope we've demonstrated a starting point of how to use AMOEBA, and Tinker, for some simple applications that are often of interest in biomolecular simulation.

註解

您沒有新增註解的權限。