# CCP-BioSim Bristol workshop: The Analysis of Biomolecular Simulation Data.

## Session 3: Applications of Principal Component Analysis.

### 1. Introduction.

In this session we will see how PCA can be used to assess convergence and sampling in MD and compare dynamics between different data sets.

The data you will be using comes from MD simulations on the mouse Major Urinary Protein (MUP), which is a popular test-bed for the investigation of protein-ligand recognition (see for example Roy and Laughton, Biophys J, 2010, **99**(1): 218-226).

In the *./data* folder you will find:

* A topology file for the protein: *Protein.gro*.
* Trajectory files for the protein, generated in the absence and presence of the pyrazine ligand: *apoProteinTraj1.dcd* and *holoProteinTraj1.dcd* respectively.
* An “index” file *apo+holo.ndx* – we will come back to this later.

### 2. Familiarisation.

If you want to have a quick look at the systems, load the *.gro* file into VMD. If you want to see more, load the trajectory files too!

### 3. Principal Component Analysis of the free protein using pyPcazip.

#### a) Using the pyPcazip package.

To get a quick guide to **pyPcazip** usage type:

% pyPcazip --help

#### b) Performing Principal Component Analysis.

The trajectory files have already been stripped of waters, ions, and the ligand; we could do PCA on the whole protein but analysis of just the dynamics of the C-alpha atoms may be a good place to start. So begin by running the PCA job itself:

% pyPcazip –i data/apoProteinTraj1.dcd –t data/Protein.gro –s “name CA” –p CA.pdb –o apo\_CA.pcz –vv

Two files will be produced: *CA.pdb* is a PDB format file containing just the atoms selected for analysis (good for checking that your selection string gave you what you intended), and *apo\_CA.pcz* is the PCA output file. This is a compressed binary file, readable using other tools from the **pyPcazip** package, so let’s do this next, using **pyPczdump**:

% pyPczdump –i apo\_CA.pcz –n

gives basic information about the contents of the file. **pyPczdump** also allows many other sorts of analysis – for a quick guide type:

% pyPczdump --help”

#### c) Looking at eigenvalues.

Now lets output a list of the eigenvalues:

% pyPczdump –i apo\_CA.pcz –l –o apo\_CA\_evals.dat

You can just scan through this in a terminal window, or input to your favourite graph-drawing package, e.g. **gnuplot**:

% gnuplot

gnuplot> plot 'apo\_CA\_evals.dat' w boxes

gnuplot> quit

#### d) Analysis of modes.

Clearly the majority of the dynamics of the free protein can be represented using a small number of collective modes (the eigenvectors). To get a feel for the sorts of motions these are, produce animations of them in the form of multi-model PDB format files, using the PDB file of the selection (CA.pdb, made in step b) above) as a ‘template’. Thus:

% pyPczdump –i apo\_CA.pcz --anim 0 --pdb CA.pdb –o apo\_CA\_ev0.pdb

Will produce an animation of the first (=0, python numbers from zero!) collective mode, which can be visualised using, e.g., VMD.

#### e) Analysis of convergence and sampling.

Having got a feel for the types of motion represented by each collective mode, we can see how these are sampled over the trajectory by outputting the time series of their projections, e.g.:

%pyPczdump –i apo\_CA.pcz --proj 0 –o apo\_CA\_proj0.dat

%pyPczdump –i apo\_CA.pcz --proj 1 –o apo\_CA\_proj1.dat

for modes 0 and 1. Again these can be visualised through, e.g. **gnuplot**.

#### f) Introducing the pczplot tool.

The process of running **pyPczdump** to produce an output file, and then turning this into some form of graph in a second step, is flexible but rather tedious. You have a chance to evaluate a trial version of an analysis tool that does both:

% src/pczplot.py apo\_CA.pcz

The start-up screen shows a plot of the eigenvalues, and summary data about the .pcz file. Press ‘1’ to show the time-series for the projections along the first PC. Pressing the ‘i’ and ‘k’ keys takes you to higher/lower PCs.

Press ‘2’ to produce histograms of the data. A PC that is sampled in a perfectly harmonic way will give a Gaussian distribution with a variance equal to the eigenvalue (shown by the dotted line); you can see how well (or not) your data fits this model. Use the ‘j’ and ‘l’ keys to change the selected PC.

Press ‘3’ to produce a ‘map’ showing how the trajectory moves over a 2D-subspace defined by two principal modes. Which two can be selected using the j/l and i/k keys. Press ‘a’ to animate the map.

Press ‘4’ to convert the map into a 2D histogram. This can presented in the form of a heat map or contour plot – press ‘c’ to toggle between them.

#### f) Further work.

Experiment with choosing different selections of atoms for the analysis (‘-s’ argument in step b)).

Repeat the whole process for the dataset from MUP bound to the pyrazine ligand (holo).

### 4. Comparative analysis of the dynamics of MUP in the presence and absence of the ligand.

#### a) Calculation of the dot product matrix using pyPczcomp.

If you visualise the first collective mode of MUP in the free state, and compare it with the same mode for MUP in the bound state, you should be able to see, even by eye, that they are not exactly the same. We can put this on a quantitative footing by calculating the dot product matrix between the eigenvectors found from PCA on the apo protein, with those found from PCA on the ligand-bound form.

% pyPczcomp –i apo\_CA.pcz holo\_CA.pcz --quick

#### b) Combined PCA on both trajectories.

To analyse the relative sampling and convergence of both simulations within a common subspace, we need to perform PCA on them both together:

% pyPcazip –i data/apoProteinTraj1.dcd data/holoProteinTraj1.dcd –t data/Protein.gro –s ‘name CA’ –o apo+holo\_CA.pcz –vv

(note how we give a list of trajectory files to be analysed in the ‘-i’ argument).

#### c) Visualisation of the data.

Use **pczplot** to visualize the results. In this case, also specify an index file, which provides information about which snapshots in the combined trajectory used for the PCA belong to which simulation:

% src/pczplot.py apo+holo\_CA.pcz data/apo+holo.ndx

Using the 2D map option (key ‘4’) see how the apo simulation samples an area of PC1/PC2 space that is not sampled in the holo simulation.