**Tutorial for MSLD using a Common Core**

WrittenBy: Luis F. Cervantes

This tutorial was written for interrogating ligand binding as the thermodynamic process, but can also be used as a general guide to running MSLD with a common core (single topology) setup.

All directories needed for this tutorial can be found in:

~luiscv/Tools/MSLD/MSLD\_Template

# **Ligand Preparation**

## **Alignment to Ligand in the Binding Site**

### Once the holo protein PDB structure that will be used for MSLD has been prepared (solvated and minimized), extract ligand coordinates from it and create a new .pdb file for the ligand only. NOTE: keep track of the solvation box size for the protein setup, as this number is necessary in subsequent steps.

### Draw the ligands that will be interrogated using a visualization software and subject them to a small minimization. Save them as a .mol2 file. Chimera is typically used for most of the ligand preparation, but any software that has the ability to create a .mol2 file should suffice. Make sure atom names within each molecule are unique.

### In Chimera, align the drawn ligands to the ligand in the binding site using the match command.

### Save the aligned ligands with a .mol2 extension

## **Identification of Common Core**

### Using ParamChem or the CGenFF module in the cluster, generate stream files with charges and parameters for each ligand. Then, separate the stream files into corresponding .rtf and .prm files. Make sure to name these files the same way the .mol2 file is named.

### Place the .mol2 files and corresponding .rtf and .prm files for each ligand into the **py\_prep** directory

### For the scripts in the **py\_prep** directory to work, Python 3 will be needed. Usually the Anaconda 3.5.3.0 module is loaded before running these scripts.

### Update the file named *mol\_list.txt* by typing the names of the ligands that will be used. Each line corresponds to a specific ligand name.

### Open *py\_prep.py*. Update the *sysname* variable with the name of your system. If CGenFF parameters are being used, set the *cgenff* variable to True.

### Run the *py\_prep.py* script up to the first *quit* statement in step (3) of the script. This should generate a file called *MCS\_for\_MSLD.txt*, which contains several things:

#### The line beginning with NSUBS contains information about how many different substituents are at each site.

#### The line beginning with REFLIG includes the name of the ligand that is used as a reference to match every other ligand’s atoms to. This is not to be confused with the reference ligand used for the free energy calculations, although they could be the same.

#### The CORE section includes a list of all of the ligands and the corresponding atom names that are in the core. The matched atoms are spatially aligned across ligands, meaning that each column corresponds to the atom names of all the ligands whose atoms are a match (or the ‘same’).

#### The ANCHOR atom section corresponds to the atoms that connect the core to the different sites. Therefore, the first column after the ligand name column corresponds to the anchor atoms for each ligand in the first site, the second column to those in the second site, and so on. The anchor atoms should appear in the CORE section. Following charge renormalization these atoms will become part of the fragment. If more than two atoms serve as anchors to a specific site, the variable DUM will be used as a placeholder (meaning dummy atom). In this case, the two atoms that serve as anchors should be placed in the SITE N FRAGMENTS portion instead.

#### Finally, the SITE N FRAGMENTS section (where N corresponds to the site number) includes the unique fragments that are within each site across all of the queried ligands.

### To visualize each of the ligands and the corresponding core and sites identified, open PyMol in a directory containing the ligand .mol2 files, the *MCS\_for\_MSLD.txt* file, and the *vis\_check.py* script.

### Run the *vis\_check.py* script within PyMol. All of the ligands should load with the common core atoms represented as hard spheres. Check each ligand to make sure the desired core has been identified.

## **Charge Renormalization**

### Once a common core has been identified via the *MCS\_for MSLD.txt* output file, open *py\_prep.py*. Comment out steps (2) and (3) of the script and uncomment the rest of the script.

### Run the *py\_prep.py* script once more (this time for steps (4) and (5)). This should generate a directory called **build.*sysname****,* where the core and site specific fragments’ pdb, rtf and prm files are generated, as well as input scripts for CHARMM to perform MSLD.

### Make sure that charge renormalization is not changing the original atom charges too much by looking at the percentage charge difference between the charge renormalized and original atom charges that should print out as the script is running. Any charge renormalized ligand that shows more than a 5% charge difference from its original charge will be flagged.

1. If a large charge difference is observed, then this usually means that the number of atoms in the core should be reduced to accommodate for that charge difference in the fragments.

### Visualize the fragments and the core pdb files to make sure that the correct core was generated, as well as the desired fragments for each site.

#### The coordinates for the core are taken from the REFLIG (reference ligand) to which everything was aligned to during the MCS search. Therefore, some of the ligands may look distorted. Depending on the visualizing software being used, this might result in incorrect assignment of bonds for visual inspection. However, the connectivity is specified in the respective .rtf, so this should not be an issue for the simulations. This distortion will be fixed in Section III following minimization of the system.

### Check the .rtf files generated to make sure that the charges for all fragments at a specific site are the same and that the sum of the charges at each site and the core add up to the net charge of the ligands.

## **Create Ligand in Water Input Scripts**

### While this directory contains the prepared ligands for MSLD for the ligand in protein, we still need to generate solvated ligand files to perform the simulations for the ligand in water, as per the ligand binding thermodynamic cycle. To this end, load the MMTSB module.

### Within the **py\_prep** directory, run the *Lg\_Solvate.sh* bash script. This will generate a directory called **solv\_prep,** which contains the same rtf and prm files, but different pdb files corresponding to the coordinates of the ligands in water. A solvent.pdb file is also generated. Make sure to make note of the box size, which should be printed when the script is run.

### We have now generated CHARMM input files for MSLD.

# **ALF Setup**

## **Create two separate ALF directories for ligand in protein and water simulations**

### Make two copies of the *ALF\_template* directory and rename them to differentiate the ligand in protein simulation from the ligand in water simulation. Steps 1-2 correspond to changes that only involve one of the ALF directories. Steps 3-8 correspond to changes that will be the same in both directories.

#### If less than ~8 substituents are being perturbed per site, then steps 4-5 should already have been automatically been done by the charge renormalization scripts. Simply copy the nsubs, name, and nblocks files from the build directory onto the ALF\_template directory.

#### If more than ~8 substituents are being perturbed per site, then subsets should be made to perturb less substituents in order to allow for convergence within a reasonable time, so steps 4-5 should be changed according to the number of substituents that will be perturbed for that specific subset.

### In the water ALF directory, copy the contents of the **solv\_prep** directory generated in Section I. Modify the last line in the *nbond.str* file to reflect an approximate size of the water box with integers that are greater than the box size and that only have prime factors of 2, 3 and 5. For example, if the box size is 37, then the line would look like “fftx 40 ffty 40 fftz 40”. This is a requirement for the way in which domain decomposition (domdec) creates the simulation grid for calculations involving GPU’s.

### In the protein ALF directory, copy the contents of the **build.*sysname*** directory generated in Section I. Modify the last line in the *nbond.str* file to reflect an approximate size of the solvated protein box with integers with the same criteria as those in step 7.

### In the *ALF\_template* directory (not the *prep* directory): change the *name* file to the system name. Change the *nsubs* file to reflect the space delimited number of fragments that will be used at each site. The first number corresponds to the number of fragments at site 1, the second to number of fragments at site 2, and so on.

### In the *ALF\_template* directory (not the *prep* directory): Change the *nblocks* file to the sum of *nsubs* (the total number of substituents across all sites).

### If VB or BP-REX is being used, then the **nbshift**directory files should be modified (beyond the scope of this tutorial).

### Go to the **analysis0** directory and run the *InitVars.py* python script. This should generate the *variables1.inp* file in the respective ALF directory, which initializes the fixed and variable biases based on the files modified in steps 4-5. This script should also populate the **nbshift** directory file *b\_shift.dat* as an nblocks long zero vector, and the files *c\_shift.dat, s\_shift.dat,* and *x\_shift.dat* as an *nblocks x nblocks* zero matrix.

## **Modifying the *sysname*.inp CHARMM input file.**

### Open the *sysname*.inp input file in the prep directory of the water ALF directory. Steps 2-12 will involve steps that are common in both the water and protein ALF directories.

#### The same explanation as that in steps II.A.1.a-b applies to the subsequent steps.

### Set the nsites variable in the script to the number of sites of the ligand. Set nsubs1 to the number of fragments in site 1 and so on for the number of sites.

### Make sure *core.rtf*, *full\_ligand.prm*, *core.pdb*, as well as the .rtf and .pdb files for each of the fragments at each site are loaded in the script (should already have been written by the *py\_prep.py* scripts, but can be modified according to the number of perturbations that will be made). If lone pairs are used through CGenFF, make sure *lpsites.inp* is also being loaded in in the script.

### Check to make sure the definitions of the different groups at each site (e.g. site1\_sub1, site2\_sub1, etc.) are defined properly with the correct atom names for each substituent. These definitions will be used in the BLOCK section of the input script.

### If linear groups are present, uncomment the respective section and specify the atom name that introduces linearity in order to delete dihedrals.

### The first substituent defined in the BLOCK section for each site will be used as the reference ligand. Modify the ‘Call’ section in BLOCK to define each substituent at each site as per the atom definitions made in Step 4.

### For the ‘ldin’ section in BLOCK, the third column value should be modified to be 1/*nsubs* for each site. For example, if two substituents are present in site 1 and 3 in site 2, then the first two ‘ldin’ lines (corresponding to site 1) should start with ‘ldin [block number] 0.5 …”. The next three ‘ldin’ lines (corresponding to site 2) should start with ‘ldin [block number] 0.3333 …”. Make sure to specify the substituent requested at each block as well. Block 1 corresponds to the environment, so no modifications are needed.

### The ‘set excl’ section involves space delimited definitions of the exclusions necessary in order to define which blocks will not ‘see’ or interact with each other. It involves permutations of the blocks that correspond to a specific site. Using the example in Step 7, where blocks 2 and 3 are the two substituents in site 1 and blocks 4-6 are the three substituents in site 2, this section should look like this:

#### set excl1 = 2 3 4 5 4 6 5 6

### Call all of these definitions in the ‘excl’ line.

### The ‘rmla’ line involves the different parameter terms that will not be scaled by lambda. Currently, bonds, angles and impropers are not scaled by lambda.

### The line starting with ‘msld’ includes space delimited integers corresponding to the site for each substituent. Using the same example as in Steps 7-8, this line should look like this:

#### msld 0 1 1 2 2 2 fnex @fnex

### Usually, no NOE restraints are used for common core, but the option is available at the end of the input script in order to introduce restraints.

### For the ligand in water input script, change the box variable to the exact box size. Then, load the solvent pdb that was generated in the **solv\_prep** directory.

### For the ligand in protein input script, change the box variable to the exact box size. Then, load the solvent pdb corresponding to that of the protein in water. Add ions separately with their respective pdb files.

### Once these modifications have been made to the input script, the directories are ready in order to begin bias optimization.

# **Running ALF and MSLD**

## **Obtaining Optimized Biases**

### Usually, 50 100ps simulations are run in order to get a rough estimate of the biases. This is followed by ~14 1ns simulations in order to further refine the biases. Additional 5ns simulations are performed in five duplicates in order to make sure that all of the substituents exhibit enough sampling.

### The first 50 100ps simulations can be performed by modifying the *runset2.sh* script. The *ini* variable in this script corresponds to the initial run that will be performed as part of this set of short simulations. The *iri* variable is the run that will be used as a cutoff for the previous restart files that can be used. For example, if one is running this script with an *ini* variable of 10 and an *iri* variable of 5, ALF will only be able to randomly choose a restart file from runs 5-9, but not the first five, for run 10. Since this is the beginning of the ALF run, the *iri* variable is set to 1. The *ifi* variable marks the last run that will be performed, which in this case will be set to 50.

### Once these modifications are made to *runset2.sh*, run the bash script *subset2.sh* which submits these as slurm jobs.

### The sampling of each substituent can be checked at the end of the output file *output\_0* in the SUMMARY STATISTICS section.

### It is also necessary to visualize the trajectories in VMD or any other visualization software by loading the dcd file in the *dcd* directory and the *minimized.psf* file that is generated in the *prep* directory.

### Another useful tool is the *PlotFreeEnergy5.m* matlab script that can be copied from the *ALF* directory into any analysis directory and run from there in order to visualize pairwise, as well as general free energy profiles to check that the free energies in lambda space for each pair of substituents at a given site are flattened.

### Once the first 50 100ps simulations have been completed, the subsequent 14 1ns simulations can be performed by modifying the *runset3.sh* script following the same specifications as those shown in Step 2. However, we set the *ini* variable to 51, the *iri* variable to 46, and the *ifi* variable to 64.

### Once these modifications are made to *runset3.sh*, run the bash script *subset3.sh* which submits these as slurm jobs.

### Make sure these simulations are completed and check the sampling again. If no sampling is observed, *subset2.sh* or *subset3.sh* can be extended using the same steps as above, modifying the *ini*, *iri*, and *ifi* variables.

### In order to run an additional 5ns production in 5 duplicates, open the *subset4.sh* script. Define the *ini* variable as appropriate (if the conventional 50 100ps and 14 1ns simulations have been successful for sampling, then the *ini* variable should be set to 65). Within the ‘sbatch’ line, modify the ‘--array’ flag to ‘-- array 1-5%1’, where 1-5 indicates that only 5ns will be run. Run the *subset4.sh* script.

### Once the simulations have completed, the results need to be postprocessed in order to account for the 5 duplicates. To this end, open the *subsetP.sh* script.

### In this script, the variable *i* corresponds to the run that will be postprocessed, *eqS* indicates the nanoseconds that will be used as equilibration time (i.e. will not be used to calculate statistics), *S* is the total number of nanoseconds run and *N* is the number of duplicates. Therefore, to postprocess our short 5ns production, we will specify *i=65*, *eqS=1, S=5,* and *N=5*. Run *subsetP.sh.*

### The *analysis* directory generated after running this script should contain a *Result.txt* file that specifies the relative free energies that were calculated averaged across the different duplicates for each of the different blocks at each site, as well as their respective uncertainties.

## **Production using optimized biases obtained from ALF**

### Assuming uncertainties are low, the free energy profiles are flat, and enough sampling is observed, production can be started.

### For the ligand in water, a 20ns production is run in 5 duplicates due to the observed increased sampling in water only using the same scripts as in Steps 10-14, but now specifying 20ns. 5ns are used as equilibration time.

### For the ligand in protein, a 50ns production is run in 5 duplicates using the same scripts as in Steps 10-14, but now specifying 50ns for production. 5ns are used as equilibration time.

### The *Results.txt* files from both the ligand in water and ligand in protein sides are then transferred into an Excel sheet. The free energies for a corresponding ligand compared to the reference from the water side are subtracted from those in the protein side. The resulting relative free energy values correspond to the relative free energies of ligand binding, as per the ligand binding thermodynamic cycle.