1 Parameterisation of modified cysteine

1.1 Preparation of the Cys-lig structure

First, The cysteine-ligand complex coordinates must be acquired. Either crystal structure data or a CovDock output will suffice. There must be a formal bond between the cysteine and the ligand so at this stage it would be recommended to export as a mol2 file and check the connectivity if you are unsure. The mol2 file output should be modifed such that every atom has the same residue/molecule code.

The structure must be capped at the terminals of the cysteine. An amide structure must be recreated for the correct charge determination. The N-terminal should be capped with an acetyl group ($COCH_3$) and the C-terminal capped with an N-methyl group ($NHCH_3$). At this point, the structure should be minimised, and a RESP calculation should be used for the charge determination. We will be using gaff2 for the parameterisation and RESP works best for these parameters.

Once charges are calculated, the generated charges should be copied into the coordinates from the crystal/docked structure.

1.1.1 Summary

- the cys-lig complex should have a single residue code for all atoms, and an explicit bond between the sulphur and ligand.
- structures should be capped with ACE and NME moieties, prior to charge calculations.
- charge calculations should be carried out using the RESP method

1.2 Parameterisation

Using antechamber, the mol2 file containg the original coordinates and updated charges will be used to generate an ac file with the following command.

```
antechamber -fi mol2 -i cys-lig.mol2 -fo ac -o cys-lig.ac -at gaff2 -an n
```

here, `cys-lig.mol2` is the file containg updated charges, the '`-at gaff2`' input defines the molecular atom types as gaff2, and the '`-an n`' input prevents the atom names being changed **(VERY IMPORTANT)**. `cys-lig.ac` is the output file and will be used to generate a prepin file.

It is at this point that a .mc file must be created. An example of this can be seen below. the .mc file defines the **M**ain**C**hain of the new resiude, along with any atoms that must be removed, connectivity information, and charge information. This should be very similar for all structures.

```
HEAD_NAME N
TAIL_NAME C
MAIN_CHAIN CA
OMIT_NAME C50
OMIT_NAME N44
OMIT_NAME H51
OMIT_NAME H52
OMIT_NAME H53
OMIT_NAME H54
OMIT_NAME C42
OMIT_NAME O46
OMIT_NAME C45
OMIT_NAME H47
OMIT_NAME H48
OMIT_NAME H49
PRE_HEAD_TYPE C
POST_TAIL_TYPE N
CHARGE 0.0
```

Figure 1: A example of what the .mc file should look like. all options in the first column must be included.

The `ac` file generated from the previous command, and the `mc` file that has been created manually are now used together in the following command.

```
prepgen -i cys-lig.ac -o cys-lig.prepin -m cys-lig.mc -rn TLC
```

Where 'TLC' is the '**T**hree **L**etter **C**ode' as it is in the mol2 file generated from the docked/crystal structure.
The parameters for the modified cysteine can now be generated.

```
parmchk2 -i cys-lig.prepin -f prepi -o frcmod.cys-lig -a Y -s gaff2
```

The '`-a Y`' input writes all parameters required in the frcmod file, and the '`-s gaff2`' input ensures the parameters written are from the gaff2 parameter set, in order to match with the gaff2 atom types generated from the antechamber step.

At this point the frcmod file is missing the parameters for mixed Amber-gaff2 bonds, angles, and dihedrals. All missing parameters for the Amber-gaff2 interface are in the `frcmod.cov` file.

You should now possess all of the files required. The original PDB file from which the coordinates were acquired must be modified prior to simulation. the modified cysteine and ligand coordinates should now be combined into a single residue as if it were an amino acid. the three letter residue code in the pdb should match the code you have decided to use until this point. all CONNECT records in the PDB file should be deleted.

```
loadamberparams params/frcmod.cov
loadAmberPrep params/NTA.prepin
loadAmberParams params/frcmod.NTA
```

Figure 2: the three generated files that must be included in `leap.in`