*Part 3: Generating Parameters for Modified Bases (advanced)*

Note that there are instructions, in particular the R.E.D. website (<http://q4md-forcefieldtools.org/Tutorial/Tutorial-1.php>) may be a useful supplement to the following instructions. Unfortunately, it is not a great read and I hope the instructions below are a little better.

*Obtain Charges for the Modified Base Using the R.E.D. program*

1. You should begin with geometry optimized log files (not frequencies or single points) for all conformations of interest, for every molecule to be included in the calculation. For example, if you are calculating charges for a C8 adduct modified nucleobase, you need a log file for each of the four nucleobases and two conformations (anti/syn) of the modified base you are interested in. Convert each log file to the pdb format using, for example, the Open Babel program. *The atom order in each conformation of the same molecule must be identical.*
2. Run the Ante Red script on each pdb file from Step 1 to generate .p2n files, which are needed to obtain charges using the R.E.D. program:

perl Ant e\_RED-1.4.pl **file.pdb** > Ante\_RED-1.4.log (repeat for each file.pdb in Step 1)

1. The script will have generated a number of output files. The important one is the p2n file, which will have the same atomic coordinates as your log file, but the atom order will be rearranged. *Use the atom order of the p2n coordinates to re-run the Gaussian optimization.* This is because you need the optimization log file to have **the same atom order** as the p2n file. In fact, there is a Gaussian input file included in the output of the Ante RED script with the correct atomic order, so you can use that to re-run your optimizations. But first….

In addition, you need to modify both the p2n and the optimization file coordinates so that **any atoms being constrained or equivalenced later in the charge calculation are in the same place (order) in every file**. So for a modified nucleoside example, now would be a good time to move the O3’/O5’ atoms to the top of the atom coordinates list in every file, for both the p2n and com files, followed by the sugar atoms. Now, run the Gaussian optimization with the correct atom order and **repeat Step 1 and 2**, so that you run the Ante\_RED script again to get the correct p2n file that you will now use to generate charges.

See this example p2n file with the correct ordering of the sugar atoms (missing the connectivity information and in brackets the specific atoms have been added, the bracketed items should not be in the file created)

REMARK

REMARK TITLE ortho-g-adduct

REMARK CHARGE-VALUE 0

REMARK MULTIPLICITY-VALUE 1

REMARK REORIENT 1 8 15 | 15 8 1 | 42 35 23 | 23 35 42

REMARK x90-t210

ATOM 1 O1 MOL 0 -3.351 -2.549 -1.046 O *(O5’ sug)*

ATOM 2 H1 MOL 0 -3.995 -2.549 -1.769 H *(H5’ sug)*

ATOM 3 O2 MOL 0 -0.508 -3.895 1.737 O *(O3’ sug)*

ATOM 4 H2 MOL 0 -0.701 -3.824 2.685 H *(H3’ sug)*

ATOM 5 CT3 MOL 0 -2.289 -3.426 -1.404 C *(C5’ sug)*

ATOM 6 H3 MOL 0 -1.944 -3.244 -2.432 H *(H5’ sug)*

ATOM 7 H3 MOL 0 -2.603 -4.480 -1.332 H *(H5’ sug)*

ATOM 8 C4 MOL 0 -1.113 -3.211 -0.464 C *(C4’ sug)*

ATOM 9 H4 MOL 0 -0.406 -4.041 -0.595 H *(H4’ sug)*

ATOM 10 O5 MOL 0 -0.456 -1.973 -0.795 O *(O4’ sug)*

ATOM 11 C6 MOL 0 -0.217 -1.241 0.407 C *(C1’ sug)*

ATOM 12 H6 MOL 0 0.723 -1.572 0.859 H *(H1’ sug)*

ATOM 13 C7 MOL 0 -1.487 -3.127 1.034 C *(C3’ sug)*

ATOM 14 H7 MOL 0 -2.496 -3.511 1.223 H *(H3’ sug)*

ATOM 15 CT8 MOL 0 -1.381 -1.625 1.324 C *(C2’ sug)*

ATOM 16 H8 MOL 0 -2.292 -1.117 1.012 H *(H2’ sug)*

ATOM 17 H8 MOL 0 -1.178 -1.407 2.378 H *(H2’ sug)*

ATOM 18 N9 MOL 0 -0.034 0.166 0.091 N *(nb)*

ATOM 19 C10 MOL 0 1.203 0.802 -0.105 C *(nb)*

ATOM 20 C11 MOL 0 -0.990 1.170 0.061 C *(nb)*

ATOM 21 N12 MOL 0 1.057 2.114 -0.198 N *(nb)*

ATOM 22 C13 MOL 0 -0.288 2.359 -0.101 C *(nb)*

ATOM 23 N14 MOL 0 -2.336 1.014 0.162 N *(nb)*

ATOM 24 C15 MOL 0 -0.996 3.612 -0.140 C *(nb)*

ATOM 25 C16 MOL 0 -3.000 2.144 0.124 C *(nb)*

ATOM 26 O17 MOL 0 -0.586 4.755 -0.244 O *(nb)*

ATOM 27 N18 MOL 0 -2.405 3.370 -0.018 N *(nb)*

ATOM 28 H18 MOL 0 -2.961 4.217 0.029 H *(nb)*

ATOM 29 N19 MOL 0 -4.370 2.129 0.286 N *(nb)*

ATOM 30 H19 MOL 0 -4.900 2.826 -0.222 H *(nb)*

ATOM 31 H19 MOL 0 -4.755 1.196 0.210 H *(nb)*

ATOM 32 C20 MOL 0 2.518 0.161 -0.184 C *(nb)*

ATOM 33 C21 MOL 0 2.707 -1.169 -0.613 C *(nb)*

ATOM 34 H21 MOL 0 1.855 -1.737 -0.970 H *(nb)*

ATOM 35 C22 MOL 0 3.668 0.935 0.148 C *(nb)*

ATOM 36 C23 MOL 0 3.970 -1.749 -0.636 C *(nb)*

ATOM 37 H23 MOL 0 4.088 -2.774 -0.973 H *(nb)*

ATOM 38 C24 MOL 0 4.933 0.328 0.144 C *(nb)*

ATOM 39 H24 MOL 0 5.785 0.939 0.424 H *(nb)*

ATOM 40 C25 MOL 0 5.082 -0.999 -0.236 C *(nb)*

ATOM 41 H25 MOL 0 6.073 -1.446 -0.244 H *(nb)*

ATOM 42 O26 MOL 0 3.613 2.240 0.477 O *(nb)*

ATOM 43 H26 MOL 0 2.702 2.572 0.269 H *(nb)*

1. For each molecule involved in the charge calculation, you should have a (single) separate p2n file and a single log file that contains the coordinates of every conformation you are including in the charge calculation. **Append all conformations of each molecule into one .p2n file and one .log file, and ensure that the conformations are in the same order in the p2n and log files:**

cat conformation1.p2n conformation2.p2n conformation3.p2n ….. conformationN.p2n > Mol\_red1.p2n

ie. cat anti.p2n syn.p2n > Mol\_red6.p2n

cat conformation1.log conformation2.log conformation3.log ….. conformationN.log > Mol\_red1.log

ie. cat anti.log syn.log > Mol\_red6.log

1. Repeat Step 4 for every molecule. Note that you MUST rename files to “**Mol\_red1.p2n** and **Mol\_red1.log**” where **1** is the molecule number. If you plan to use charge equivalencing, then the molecule assigned to be number 1 should be the one *donating* the chemical group (e.g.; dimethylphosphate should be molecule 1 if attaching to a nucleoside, where the nucleoside(s) would be Mol\_red2, Mol\_red3, Mol\_red4, etc.). You should then further edit the p2n files manually. Separate each conformation in the p2n file with a TER card (Remove the END card that is present). Only the first conformation in the file needs the REMARK lines at the top of the conformation (except for a REMARK line with a description of each conformation, for your own benefit). Only the first conformation in the file requires topology information following the Cartesian coordinates. Note that log files require no further editing, but *cannot* contain frequency calculation information, and MUST be optimization logs, not single points.
2. Edit the “TITLE” of the molecule in .p2n to something that makes sense (see example p2n file above). Check that the charge and multiplicity are correct. For the first conformation in each file, tell the program what reorientation procedure to use when calculating the charge on each atom and place this on the top of every file:

REMARK REORIENT 1 2 3 | 3 2 1 | 4 5 6 | 6 5 4

It is a good idea to reverse the order of atoms you have chosen to cancel some error. Only choose heavy atoms for reorientation, not hydrogens. Any heavy atoms can be chosen. I chose 4 orientations, the more orientations you choose the more reproducible the numbers are.

In the atom names column, *equivalent atoms must have identical names*. This may require further manual editing. Note that atoms with a “T” after the name will be treated in two-stage RESP generation.

1. If intermolecular charge constraints or intramolecular charge constraints or intermolecular charge equivalencing is required, insert the appropriate command in the p2n file remarks for the **first** conformation:

Ex. 1: REMARK INTER-MCC 0.0 | 1 2 | 14 15 | 6 7 8 9

This is read as: REMARK intermolecular\_charge\_constraint charge constraint 0.0 | molecules 1 and 2 involved | atoms from molecule 1 involved are 14 and 15 | atoms from molecule 2 involved are 6, 7,8 and 9

Atoms 14 and 15 in Molecule 1 and atoms 6 7 8 9 in Molecule 2 will be constrained to a charge of zero so that these atoms can be deleted and Molecule 1 and 2 can be connected, such as a phosphate group joined to a nucleoside to form a nucleotide.

e.g. 2. REMARK INTER-MEQA 2 3 4 5 6 | 1 2 3 4 5 6 7 8 9 10 13 14 15 16 17

This means Charge Equivalencing between molecules 2, 3, 4, 5, 6, where atoms 1-10 and 13-17 (in this example, the sugar atoms in each of the 5 nucleosides) are held equivalent (*atom order in each file must be identical*). Note atoms 11 and 12 are not held equivalent because these are C1’ and H1’ and allowed to have different charges.

These two commands must be present in the REMARK lines of only **one** of the p2n files when multiple molecules are being considered. However, reorientation remarks are required for each individual p2n file (each molecule).

1. Now you can run the RED script to generate charges for all atoms, and .mol2 files. Ensure that you load AMBER before you run this.

perl RED-vIII.4.pl > RED-vIII.4.log

1. It should take less than an hour to run the script successfully. You will get back a DATA-RED directory with a subdirectory for each molecule involved in the run, plus a **Mol\_MM** directory for the molecular fragments (nucleotides) generated by the charge-equivalencing process. In each whole molecule subdirectory is a mol2 file containing RESP charges for the whole molecule. In the Mol\_MM directory are mol2 files for every fragment generated. The mol2 files contain the atomic coordinates and a column with the charges you have now calculated. There is also space for a column containing atom types which we will generate next. We will use these mol2 files to convert to an AMBER library file.

*Generate Atom Types and Missing Parameters for your modified base*

1. Choose the mol2 output file you want to use in AMBER (either the nucleoside or nucleotide fragment), and run antechamber to generate (amber) atom types in your file. When naming output.mol2, use 3 letter code as this will be needed later (eg. t4m could represent o4-methylthymine)

antechamber -i input.mol2 –fi mol2 –o output.mol2 –fo mol2 –at amber

The “-at” flag specifies the kind of atom type. The default is gaff. (I modified my phenol carbon with the OH group to be type “C” instead of “CA.”) Antechamber also modifies the connectivities, so I undo those changes.

**Editing mol2 BEFORE frcmod files:**

Correct atom types (after coordinates)

Change all \* to ‘

UNL should be your file name (eg. t4m)

Fix first row of atoms (eg: N could be N1, and CH3 could be C7,H71,H72.H73)

1. Then run parmchk to check the parameters required by your molecule against the parameters provided by your forcefield to find out what parameters are missing. First make sure to copy the parameter file from within the Amber folders to the working directory (in this case, we are using the parm99SB forcefield, so the parameters are contained in the file parm10.dat).

parmchk –i output.mol2 –f mol2 –o output.frcmod –p parm10.dat

Parmchk will assign missing parameters if there is something similar enough in the forcefield. It will print these to the frcmod file. It will also list any parameters that for which no approximation could be found. Check for any that say ‘ATTN, need revision’ and correct these by repeating step 11 with the gaff atom types and parameter files and generating another frcmod file:

parmchk –i output.mol2 –f mol2 –o output-gaff.frcmod –p gaff.dat

Then combine these by copying and pasting the corresponding coordinates/description into your output.frcmod

*Creating Amber library files*

In the case of a WHOLE MOLECULE, the following instructions can be followed without further editing. If creating a nucleotide fragment, please see Step 17 prior to following these directions.

1. Now we will convert the mol2 file to an amber library file. Load your force field in tleap with the following commands, replacing output.frcmod and output.mol2 with your output files from the previous step, and **file** in **file**.lib with what you are calling it.

Copy the following lines into notepad, and replace output.frcmod and output.mol2 with your files from steps 10 and 11, and the replace file in file.lib with your 3-letter title

tleap -s -f /global/software/amber/amber14p13\_at14p27/dat/leap/cmd/ leaprc.gaff

source leaprc.ff14SB

loadamberparams output.frcmod

LIG=loadmol2 output.mol2

savepdb LIG lig.pdb

set LIG head LIG.1.P

set LIG tail LIG.1.5

set LIG.1 connect0 LIG.1.P

set LIG.1 connect1 LIG.1.5

saveoff LIG **file**.lib

bond pyr.7.O3 pyr.8.P

Copy and paste this list of commands into the command line all at once. Once they have all gone through, hit ctrl c to quit. Your library file should be made.

In the library file, modify wherever it says ‘LIB’ to the title of your output file name (eg. From above: LIB would be t4m) in one of two ways:

The sed command, replacing ‘PYR’ with your 3-digit code:

sed –i ‘s/LIG/PYR/g’ \*.lib

or

Open the file on Notepad++, hit ctrl f, hit the replace tab and replace LIG with your title (eg. t4m)

**If you have the above correctly, skip to step 16, as steps 13-15 are another way to create this file.**

tleap -s -f /opt/amber11p6.X11/dat/leap/cmd/leaprc.ff99SB

To load an additional force field (if you have a mixture of gaff and amber atom types and parameters) use for example:

source leaprc.gaff

Next, load your new frcmod file:

loadamberparams file.frcmod

**Aside:** How to use the set of parameters parmbsc0

------------------------------------------

0) source leaprc.ff99bsc0, (or just open leap with this force field loaded: tleap -s -f $AMBERHOME/dat/leap/cmd/leaprc.ff99bsc0)

1) It should load DNA\_CI.lib, a library that has all DNA nucleotides but having changed

one atom type: atomname C5' is changed from atomtype CT --> CI

2) load an extra parameters file (frcmod.parmbsc0) that includes all the new parameters needed due to the new atom type definition and the new QM-fitted parameters for torsion alfa and gamma

3) load any additional forcefields or frcmod files you have created for modified bases, as in step 12. Then load the mol2 of the structure that you have made, as in Step 13 below.

The most recently loaded force field or force field modification will be applied, if they conflict.

Important: Note that this procedure will only apply the new parameters to DNA (or RNA if the 2nd library is loaded) residues with canonical bases i.e. that follow amber atom name definition. For your modified base, **manually** change C5′ to atom type CI in order for the force field to recognize it.

*These steps for loading the forcefield can also be referred to later, when you are creating your MD input files with your pdb instead of your mol2.*

1. Now all parameters required to describe your molecule should be loaded in leap. You can now load your mol2 file with charges and atom types:

TEST=loadmol2 file.mol2

(NOTE: Here I name the unit “TEST” as an example, but you must give the unit the same name as you gave your residue; e.g. LIG.)

1. You can set the head and tail atoms using the ‘set UNIT head UNIT.residue.atom’ commands, and set the residue type using the ‘set TEST restype nucleic’ command. If it is a fragment, also specify the connect1/connect0 atoms.

e.g.; set TEST head TEST.1.P

set TEST tail TEST.1.5

set TEST.1 connect0 TEST.1.P

set TEST.1 connect1 TEST.1.5

Means that P and O3 (atom 5) are the terminal atoms in this newly defined residue, and that other residues will connect to these atoms if you use them to build a chain.

1. You can save this as a library file:

saveoff TEST file.lib

You should not have received any warnings or errors in this process (unless they make sense!!). To make sure, use the ‘check’ command (check TEST). It shouldn’t find anything missing, and it should say the unit is ok. You have now created a new residue that can be used to run MD simulations.

1. If you are running MD on the nucleoside, then you can save the prmtop and inpcrd files, or alternatively, add waters and ions and then save the prmtop and inpcrd files. (saveamberparm TEST prmtop inpcrd). Again, there should be no errors or warnings in the output across your screen.

You can now use these files to run MD for a whole molecule. Follow the instructions from <http://ambermd.org/tutorials/basic/tutorial1/section6.htm>.

Creating a strand using the Nucleic Acid Builder (nab) problem in Amber (MD tutorial section 1) in linux, renaming pyr to what you would like to call your file:

vi pyr.nab

In this file, copy and paste:

molecule m;

m = fd\_helix( "adna", "aaaaaaaaaa", "dna" );

putpdb( "a-dna.nab.pdb", m, "-wwpdb");

Where “aaaaaaaaaa” is replaced by your 5’ 🡪 3’ strand sequence using letters a,g,c,t.

Save this file and quit; to run nab, copy the following into the command line:

nab pyr.nab

./a.out

This should create pyr.nab.pdb, which you can open in Gaussview or PyMol to check.

Ignore the rest.

1. FOR A FRAGMENT (nucleotide with dangling ends)

Before you build the library for your residue, you need to change the atom names (not types) in the mol2 file to match what they will be in the pdb of the whole system you are running MD on. That way, you can insert your residue in a sequence. (For example, a nucleotide in dsDNA.) To obtain the pdb file, create the natural sequence pdb, and modify the structure to contain your residue.

Try building the natural strand you want in Amber using the nab program (See MD tutorial section 1), then open it in GaussView to modify it. Then save the file as a .mol2, and convert it back to a pdb using Open Babel. It will then require editing to become recognizable to Amber again. You can delete all the connectivity at the bottom of the file, and delete the first two header lines of the pdb. The sed command is useful to search and replace certain things:

sed ‘s/oldstring/newstring/g’ filename > newfile

You can copy and paste these commands to make sure you have the correct number of spaces:

sed ‘s/ A / /g’ file1.pdb > file2.pdb *Removes the chain labels*

sed ‘s/ B / /g’ file2.pdb > file3.pdb

sed ‘s/ G/DG/g’ file3.pdb > file4.pdb *Correctly names the residues. Don’t include Cyt*

sed ‘s/ A/ DA/g’ file4.pdb > file5.pdb *because it will change all of your carbon names.*

sed ‘s/ T/DT/g’ file5.pdb > file6.pdb

sed ‘s/DATOM/ATOM/g’ file1.pdb > file2.pdb *Corrects atom name from DATOM to ATOM*

sed ‘s/5M/7/g’ file6.pdb > file7.pdb *Correct atom name for Thy.*

sed ‘s/OTM/OTD/g’ file7.pdb > file8.pdb *Renames the unknown residue to your ligand.*

*Make sure the name (LIG) matches your mol2*

Note that your PDB file must be carefully formatted:

* Change the name of Cyt from C to DC manually.
* Atom names are supposed to be in columns 13-16, with one, two and three character names starting in column 14, and four-character names (like H2'') starting in column 13.
* Residue names should be in columns 18-20; two character names (like DA) are right justified (columns 19-20) in the PDB, although I think Amber will accept left justification as well.
* Further editing may be needed or you will get error messages later when you load your pdb in Amber. For example, the terminal atoms must have atoms names “**HO5’**” and “HO3’” to be recognized, although \* is interchangeable for ‘ in all other atom names.
* You must also correct the *numbering* of the residues in column 26 (not the atoms) (it will be messed up around your modified residue, and it will restart at 1 in the second strand of DNA, but it should not.)
* Also, H3 in any thymine residue will be incorrectly labeled as H (due to the distance of the C-H bond in GaussView).
* There should be a “TER” card separating the two strands.
* If the editing seems tedious, feel free to look for other (more efficient?) ways to convert your file to a pdb (or skip using GaussView altogether?). But be cautious that your files are *really* what you want them to be, and no information has been lost!
* Note that if you have added atoms in GaussView to the modified part of your DNA strand, they will appear at the very bottom of the pdb file, separate from the rest of the residue. Don’t forget to edit these atoms and change the atom names to match the mol2 (and the numbering!), and copy and paste them into the correct place in the pdb, or they won’t be recognized.

Below is a few examples of the spacing and names which you should have in your PDB file (note this only shows the first few column, no editing is required to columns further right)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| A | T | O | M |  |  |  |  |  |  | 1 |  |  | H | O | 5 |  |  | D | G |  |  |  |  |  | 1 |  |
| A | T | O | M |  |  |  |  |  | 6 | 8 |  |  | H | 3 | \* |  |  | D | T |  |  |  |  |  | 3 |  |
| A | T | O | M |  |  |  |  | 1 | 0 | 2 |  | H | 5 | \* | 1 |  |  | D | A |  |  |  |  |  | 5 |  |
| A | T | O | M |  |  |  |  | 1 | 9 | 3 |  |  | C | 4 |  |  | L | I | G |  |  |  |  |  | 6 |  |

When you are finished editing, follow Steps 12 to 15 to generate the library file for your residue.

1. Then, to create the strand, open leap again, re-load your force field(s) and frcmod file(s), then, load the library file that you made earlier:

loadoff file.lib

Type ‘list’ to see what residues are defined and make sure yours is there. Now, load your pdb file of the full sequence (TEST=loadpdb FULL\_STRAND.pdb). Your modified residue should be recognized by the program. Check the unit to make sure all parameters are present (check TEST).

Now, you can save the prmtop and inpcrd files as in Step 16, or use the solvateoct and addions commands, and then save the prmtop and inpcrd files. I recommend saving the files at each step. (Follow <http://ambermd.org/tutorials/basic/tutorial1/section6.htm> for instructions on these commands and running minimization and equilibration procedures).