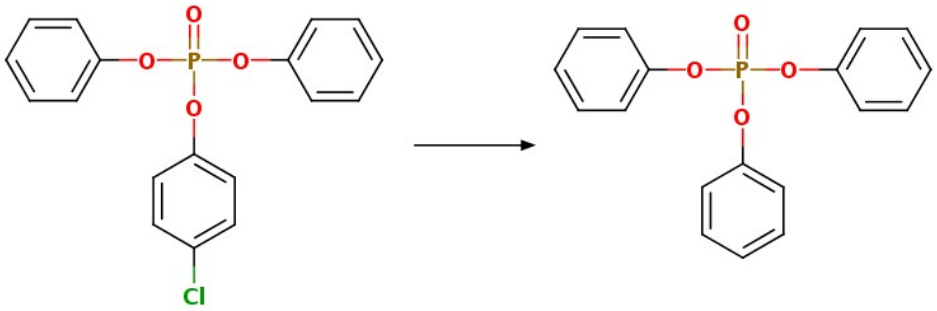
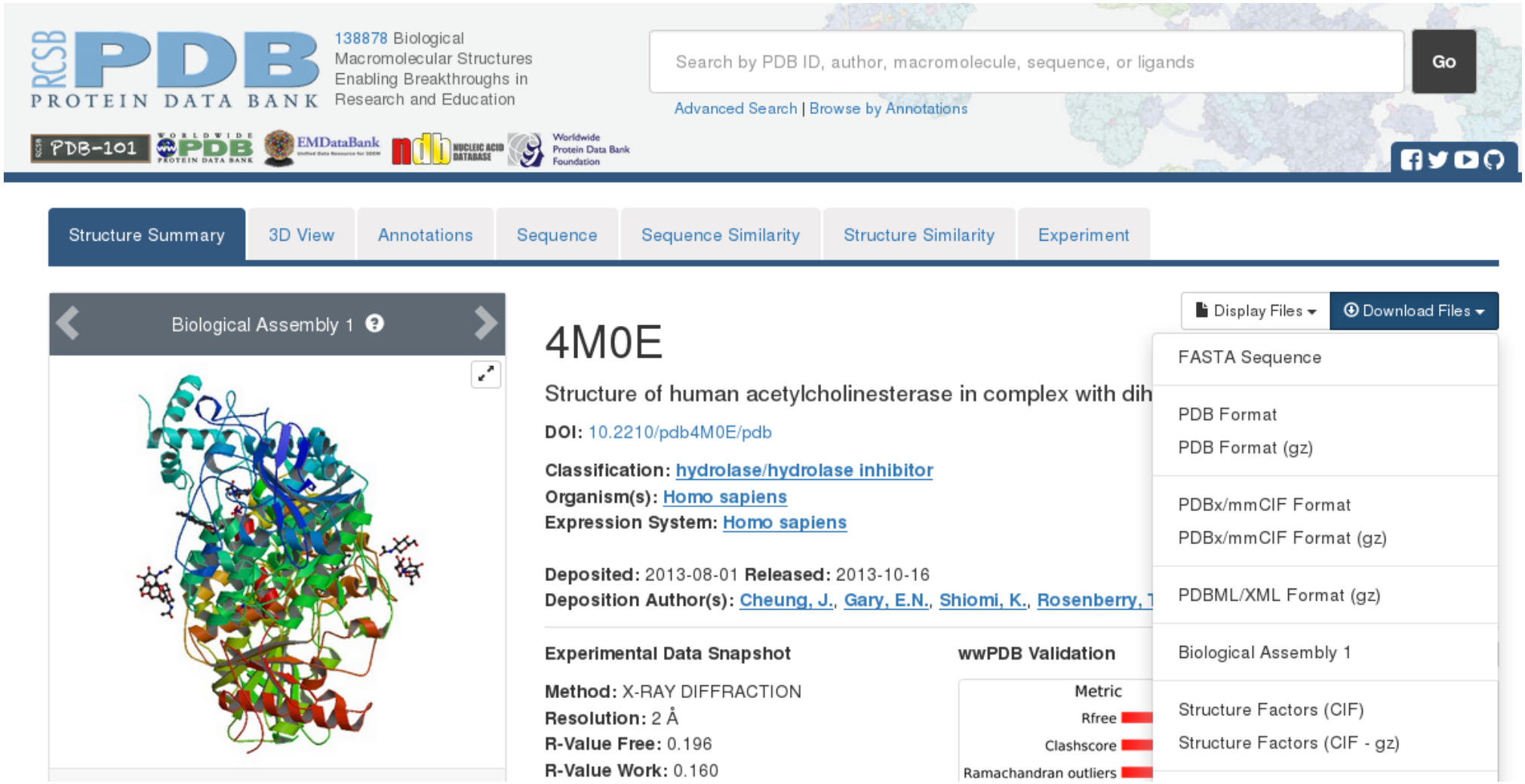
MCPro/FEP tutorial

Test Case #1 TPP/AChE



Download and Prepare a Protein Structure for Docking

Download 3D structure of Acetylcholinesterase (pdb id: 4m0e) from www.rcsb.org as pdb format (4m0e.pdb)



All proteins are categorized by a 4 digit alphanumeric code, known as the pdb ID
You can use this code to find your protein if you know it (you can also searches by name, ligand, author, etc.)

Prepare your protein for Docking using the Dockprep function of Chimera to add hydrogens, remove solvent ions, excess ligands, cofactors, or subunits, and repair incomplete side chains

Open 4m0e.pdb in chimera:

→Select → Chain → B

→Actions→ Atoms/Bond→Delete

#The protein is a dimer you will only need one chain so you are deleting the other

→Select → Residue→all nonstandard

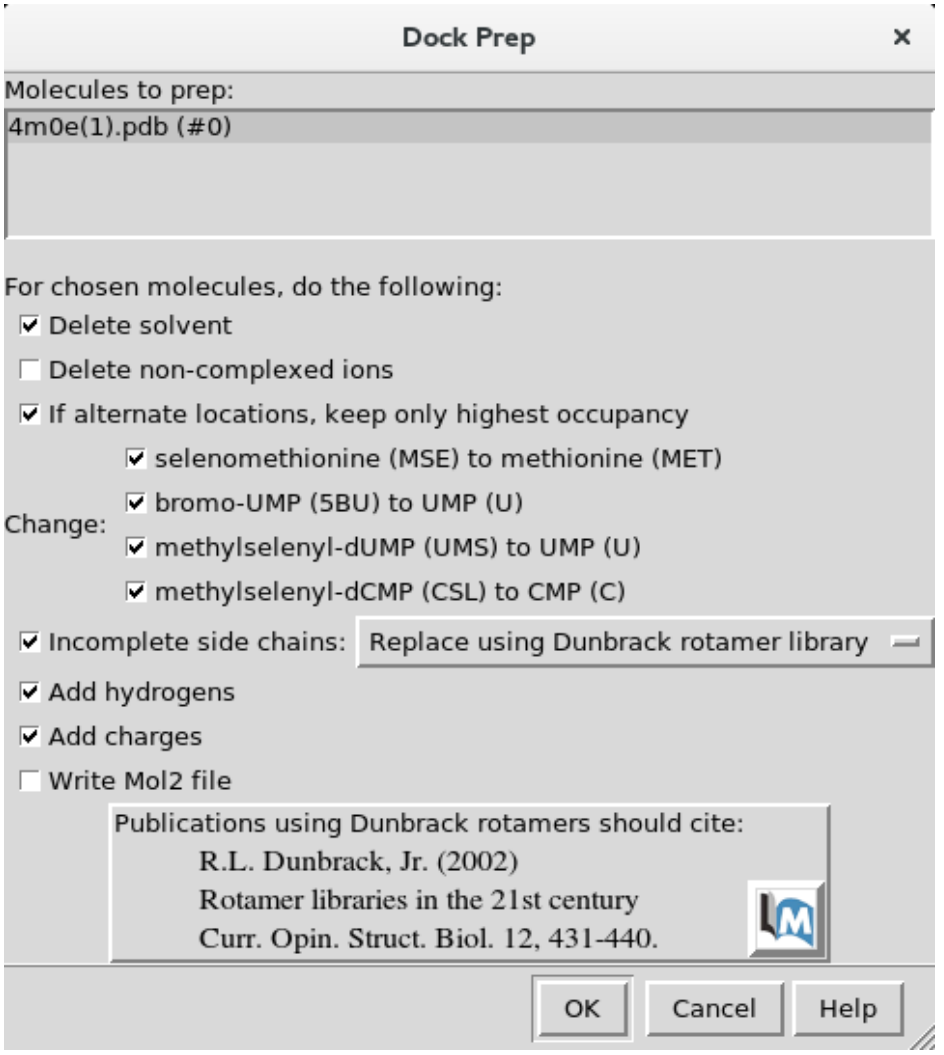
→Actions→ Atoms/Bond→Delete

#This will delete any ligands, ions, etc that are bound to the protein

→Tools→Structure Editing→DockPrep

#This brings up the menu to prepare the structure for docking: Deletes any solvent molecules, adds H's, charge, and fixes incomplete side chains

Uncheck Mol2 file, we will save the structure as a pdb to use in further prep for docking



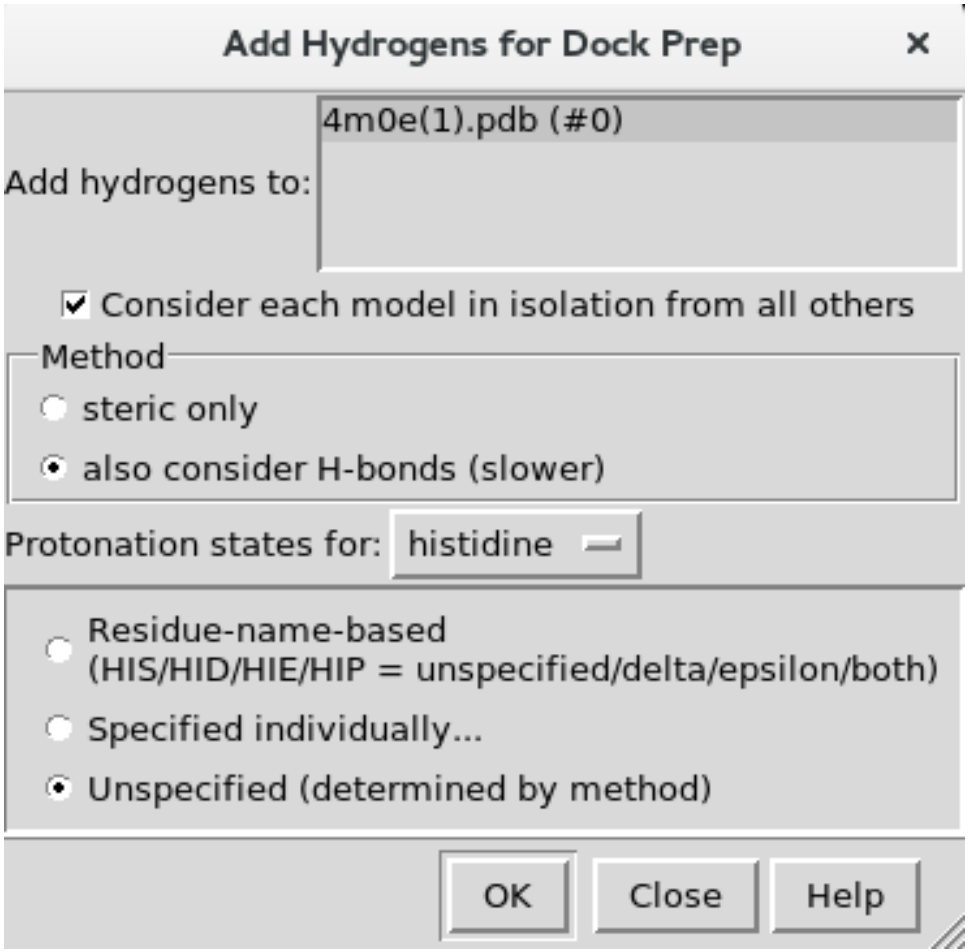
Prepare your protein for Docking using the Dockprep function of Chimera to add hydrogens, remove solvent ions, excess ligands, cofactors, or subunits, and repair incomplete side chains

Change the selection from “Residue-name-based” (default) to “Unspecified (determined by method)”

#Residue-name-based will simply assign a default protonation state based on the name of the residue

Ex) HIP = doubly protonated histidine

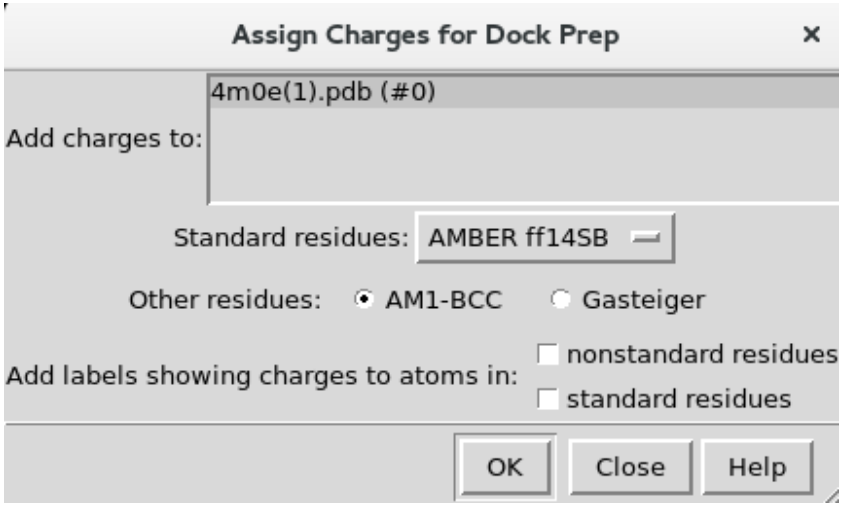
We want to instead calculate the protonation states

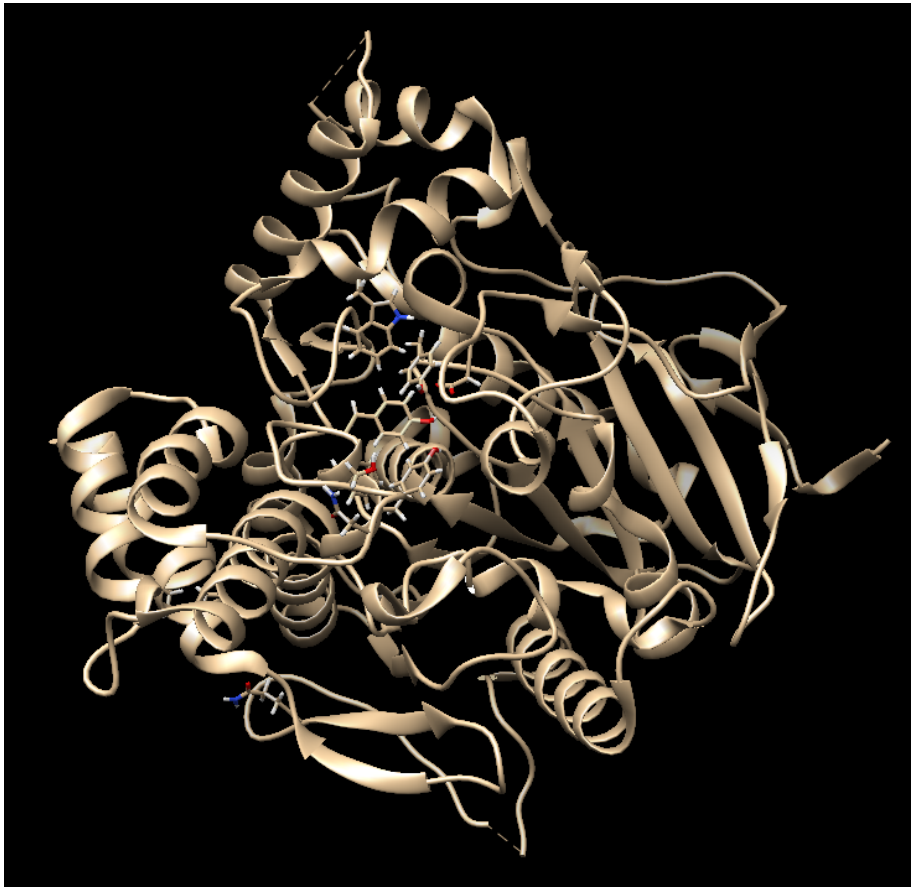


Prepare your protein for Docking using the Dockprep function of Chimera to add hydrogens, remove solvent ions, excess ligands, cofactors, or subunits, and repair incomplete side chains

Select the AM1-BCC charges

#These charges will not actually be used so you could really use either but the AM1-BCC charges should be more consistent with our forcefield.

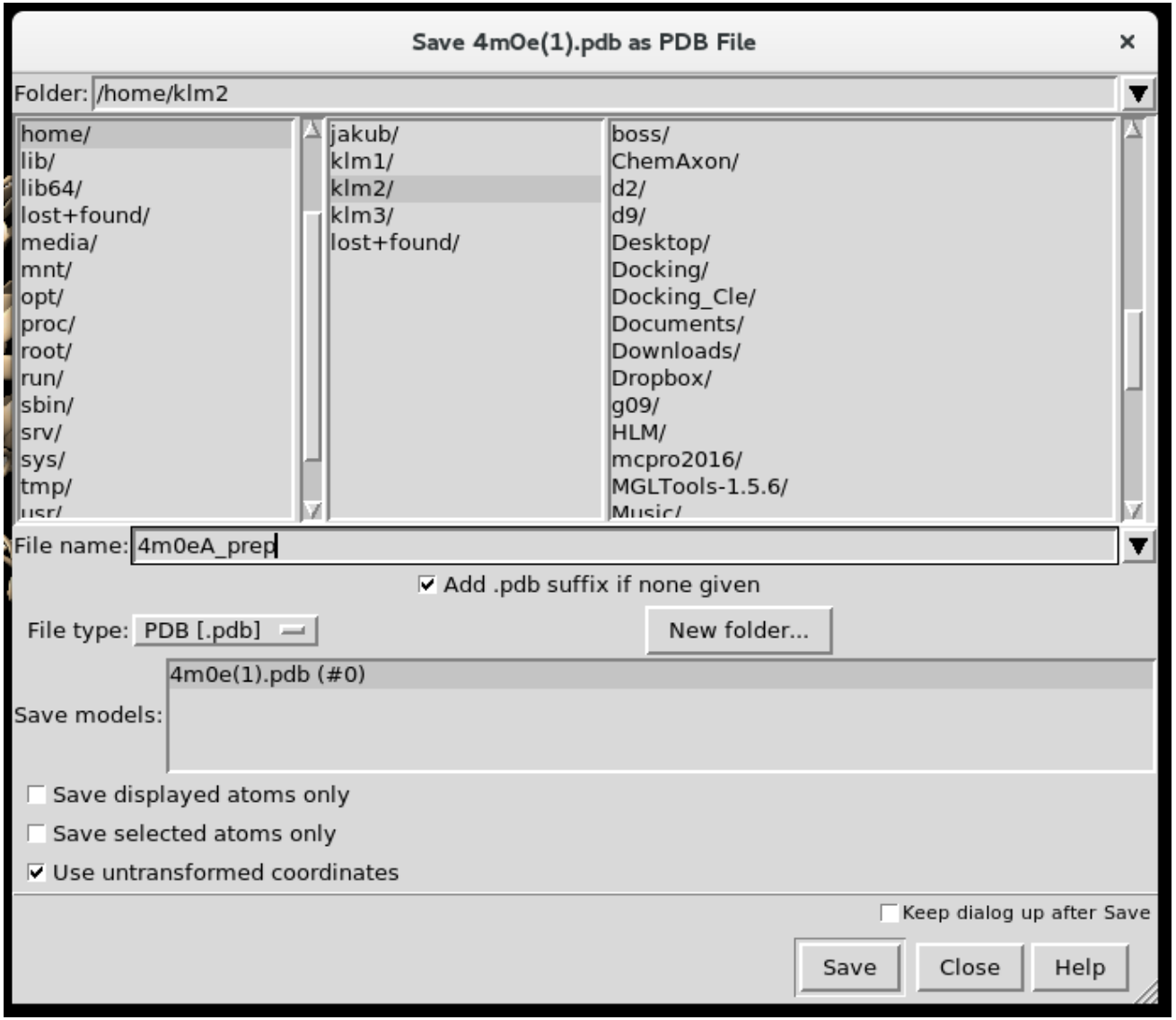




Save your prepped protein as a pdb

File -> Save PDB
4m0eAprep.pdb

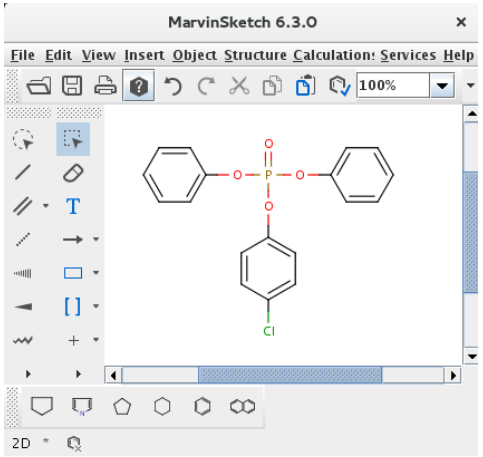
Pick a name that indicates the changes/prep you've done



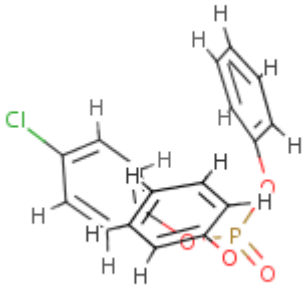
Prepare your chosen Ligand (TPP) for Docking

- You will need a PDB for docking there are several ways to do this:
  - 1) Draw your ligand in Marvin
  -
  - 2) Draw your ligand in LigParGen Server and download as a pdb

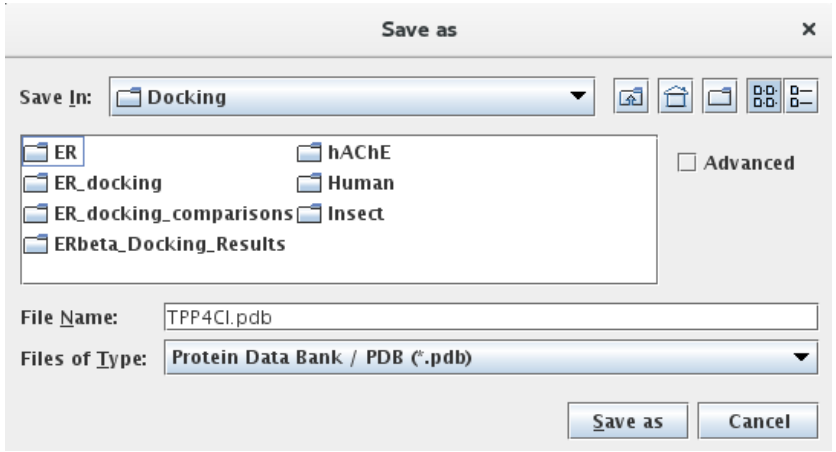
Preparing Ligand with Marvin



- **Open MarvinSketch and draw your structure:**
  - Triphenyl phosphate with a chlorine substituted in the para position
  -
  -

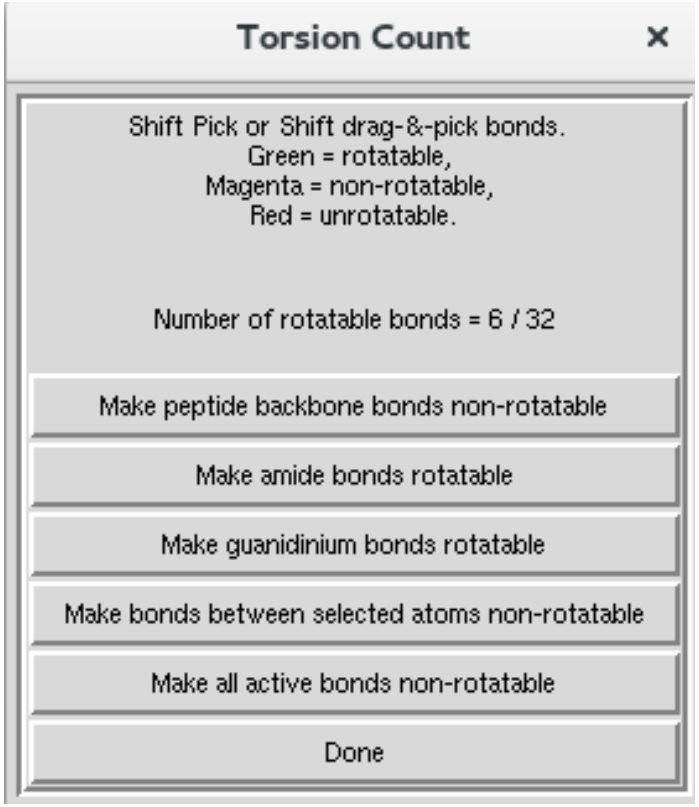


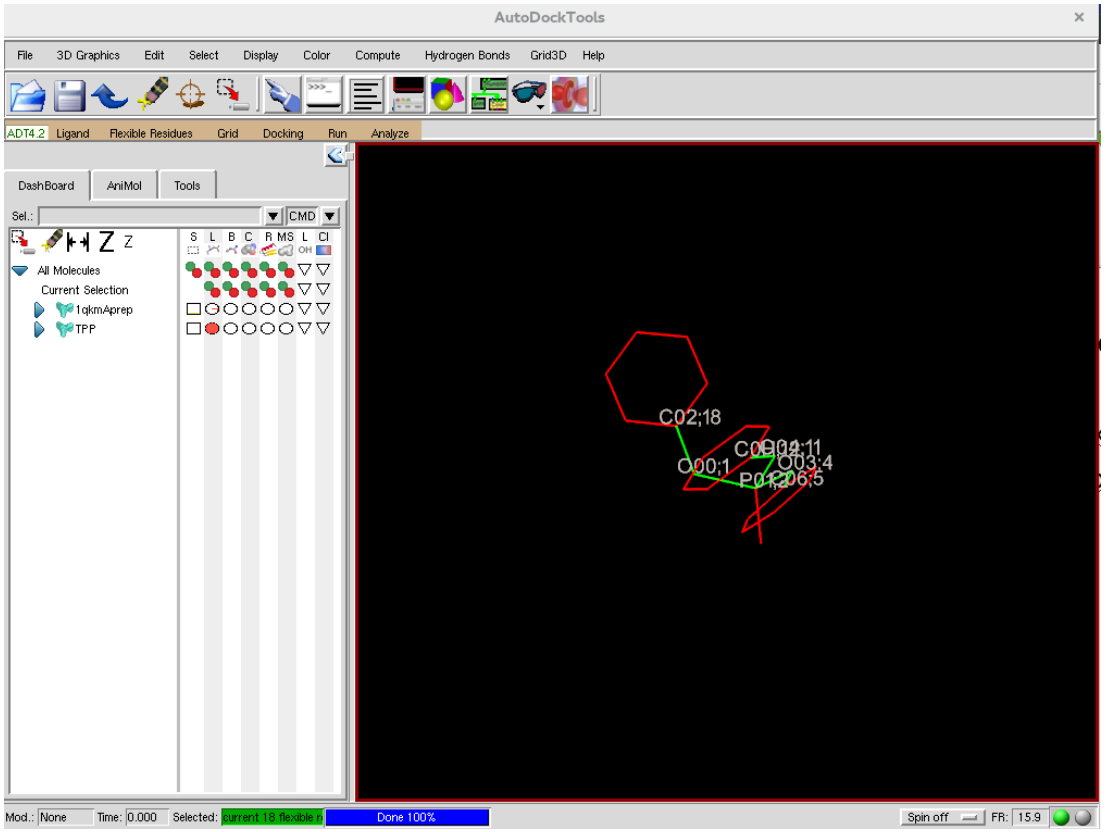
- **Use clean in 3D to get a loosely optimized structure:**
  - Structure → Clean 3D → Clean in 3D
  - (it will look crazy because its 3 dimensions shown in 2, don't worry)
  -
  -
- **Save as a pdb:**
  - File → Save as → “Ligand.pdb”
  -
  -
  -



Use autodock tools to prepare input file of ligand for docking with autodock Vina

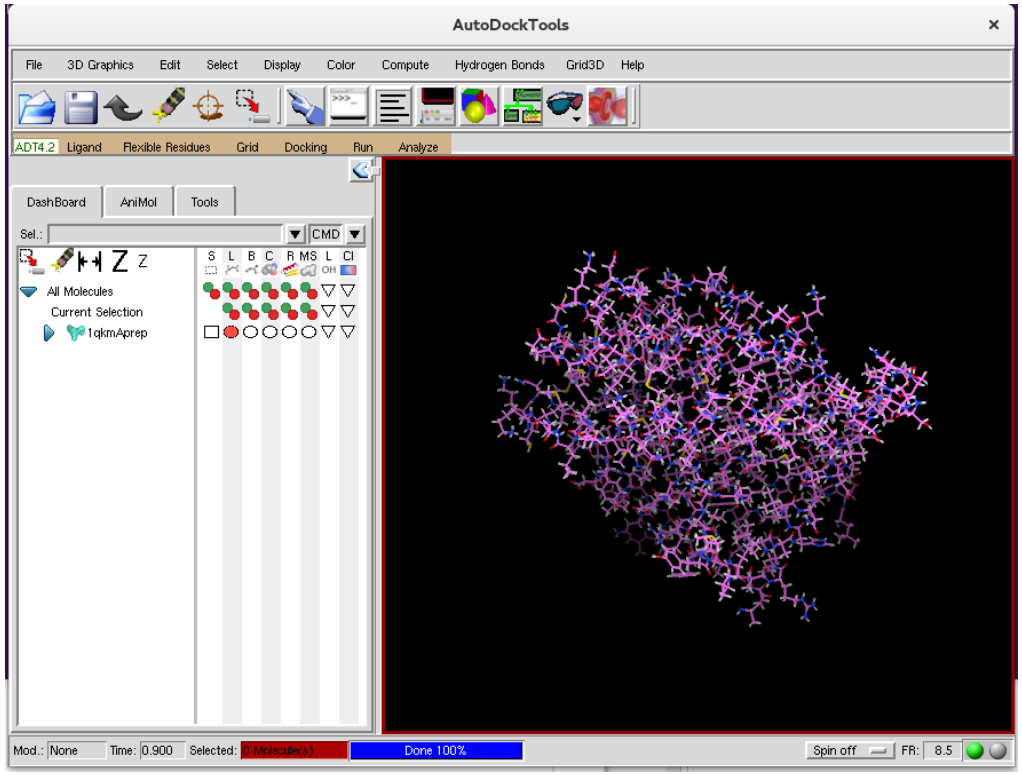
- Type “adt” in the command line to open autodock tools
- Ligand → Input → Open → TPP.pdb
- Ligand → Choose torsions (are they correct?) → Done
- Ligand → Output → TPP.pdb → save TPP.pdbqt
- Close autodock tools
-





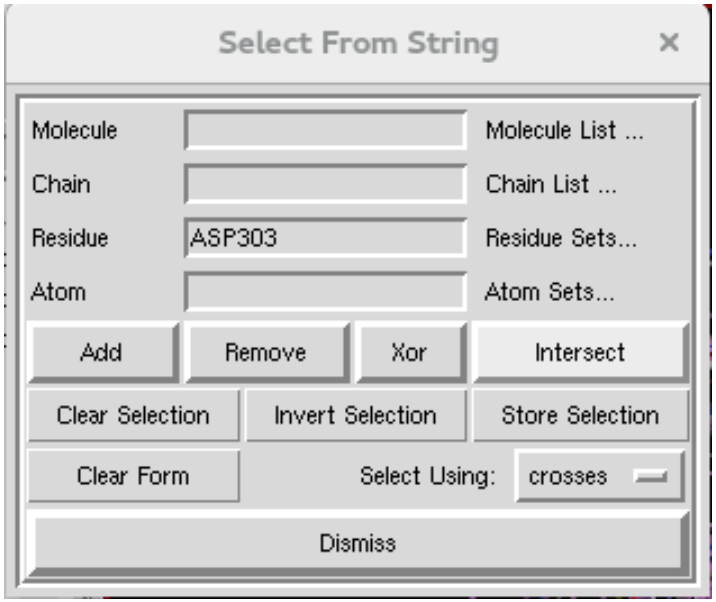
Use autodock tools to prepare input files from your prepped protein for autodock Vina

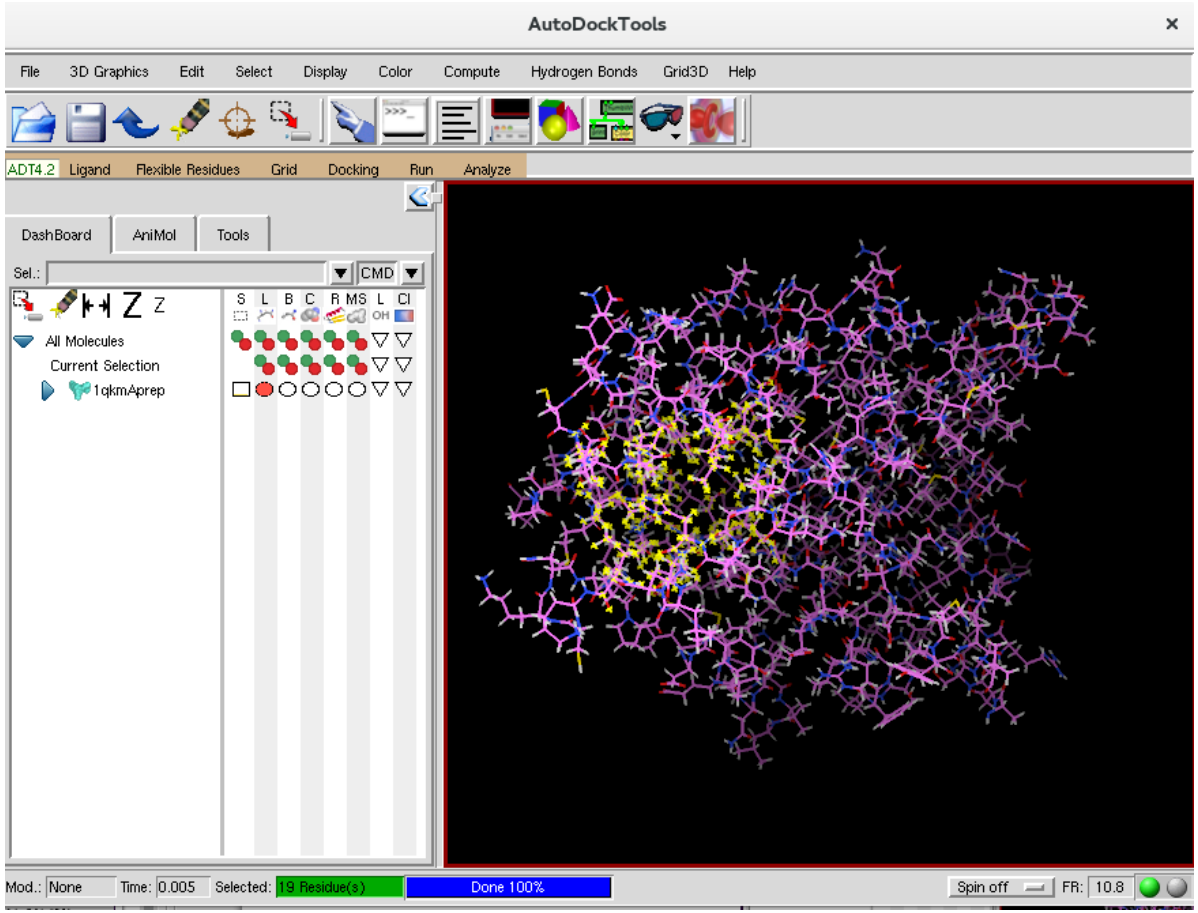
- Type “adt” in cmd line
- Open the pdb of your protein that you prepped in Chimera:
  - File → Read molecule → 4m0eAprep.pdb
  - Edit → Delete water (should already be done)
  - Edit → Hydrogens → Merge non-polar
  - Grid → Macromolecule → choose → 4m0eAprep.pdb
  - (creates 4m0eAprep.pdbqt)



Use autodock tools to prepare input files from your prepped protein for autodock Vina

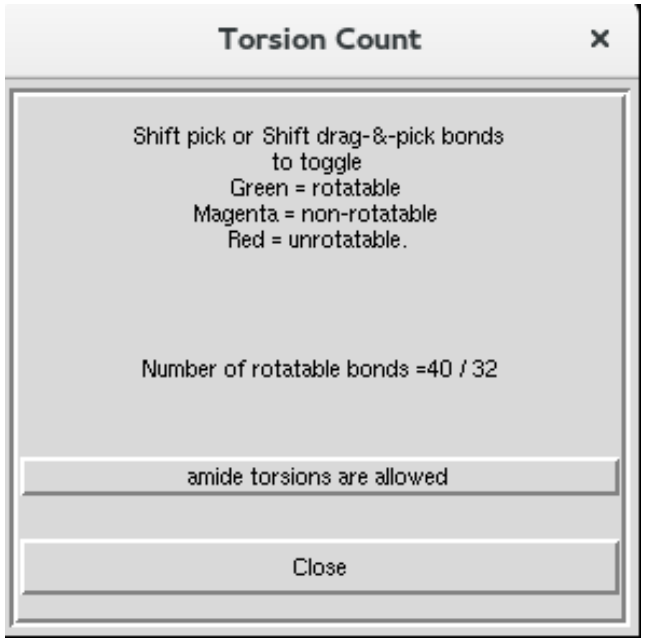
- Select Key Residues that will be allowed to rotate during docking:
  - Select → select from string:
    - MET85 TRP86 TYR124 TYR133 SER203 GLU202 PHE297 TRP236
  - PHE295 TYR337 TRP286 HIS447 PHE338 GLU450 TYR449 ILE451

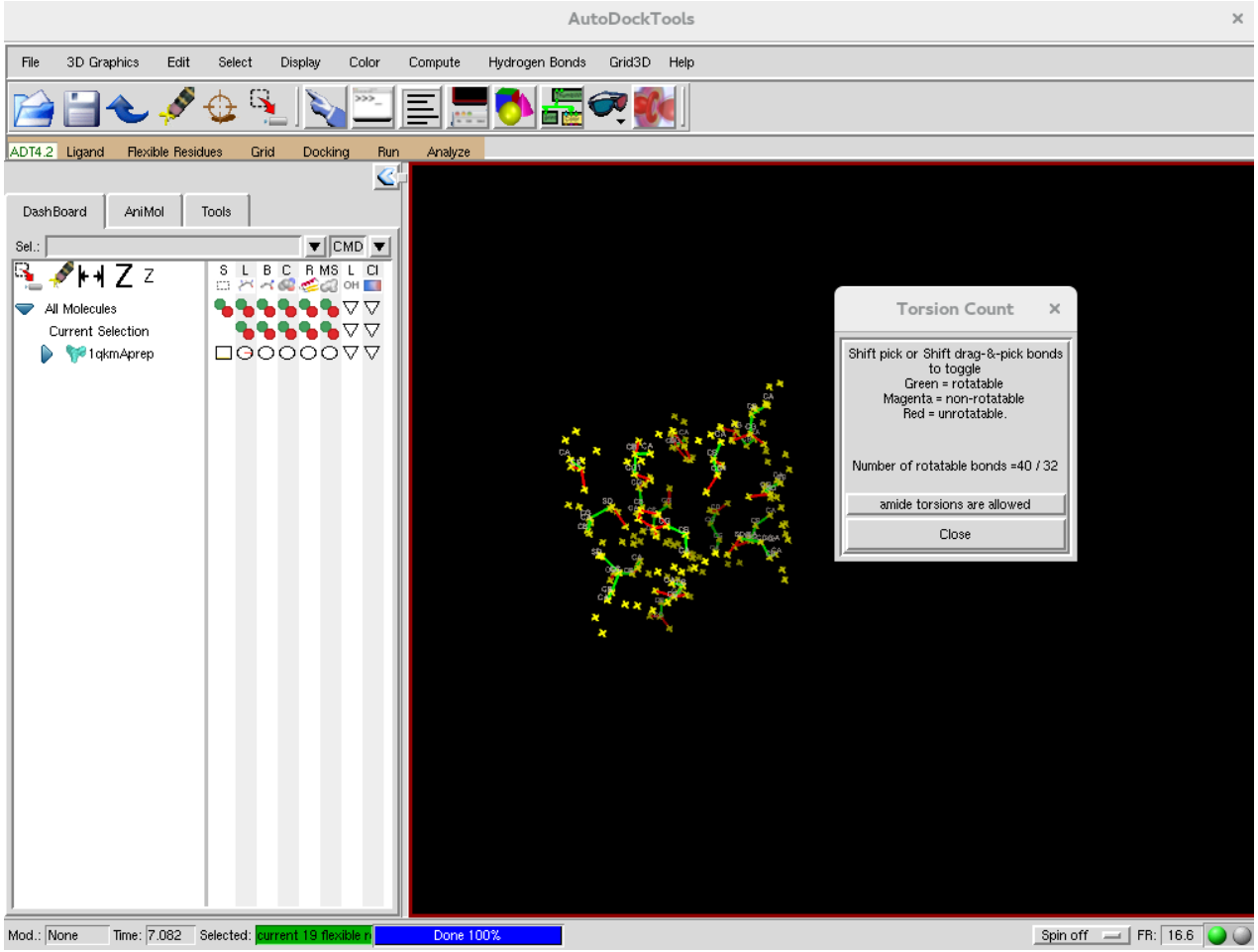




Use autodock tools to prepare input files from your prepped protein for autodock Vina

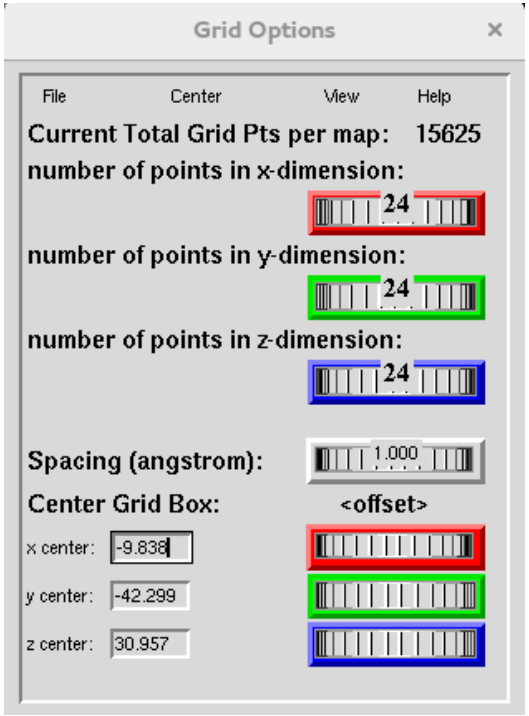
- Flexible residues → Input → Choose macromolecule → 4m0eAprep.pdbqt
- Flexible residues → Choose torsions
- Flexible residues → Output → Save Flexible PDBTQ (4m0eAprep_flex.pdbqt)
- Flexible residues → Output → Save Rigid PDBTQ (4m0eAprep_rigid.pdbqt)

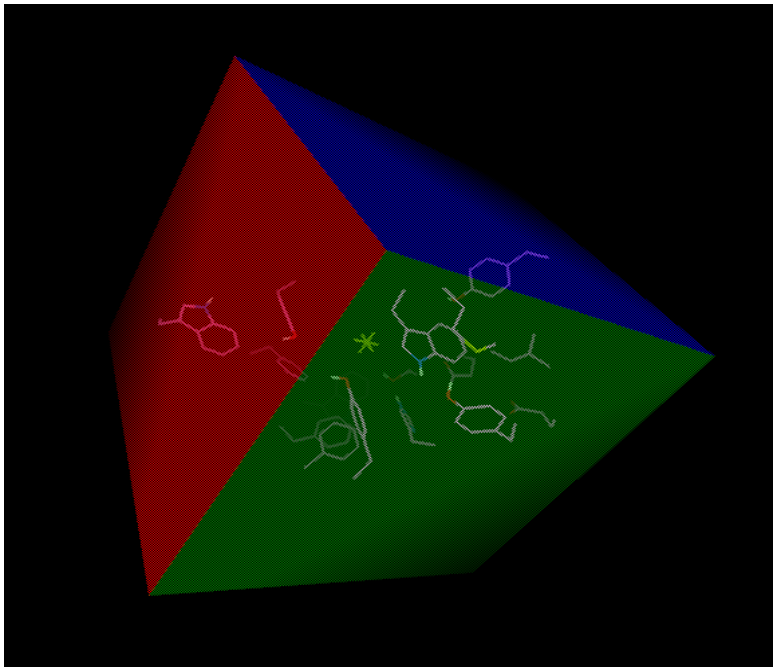




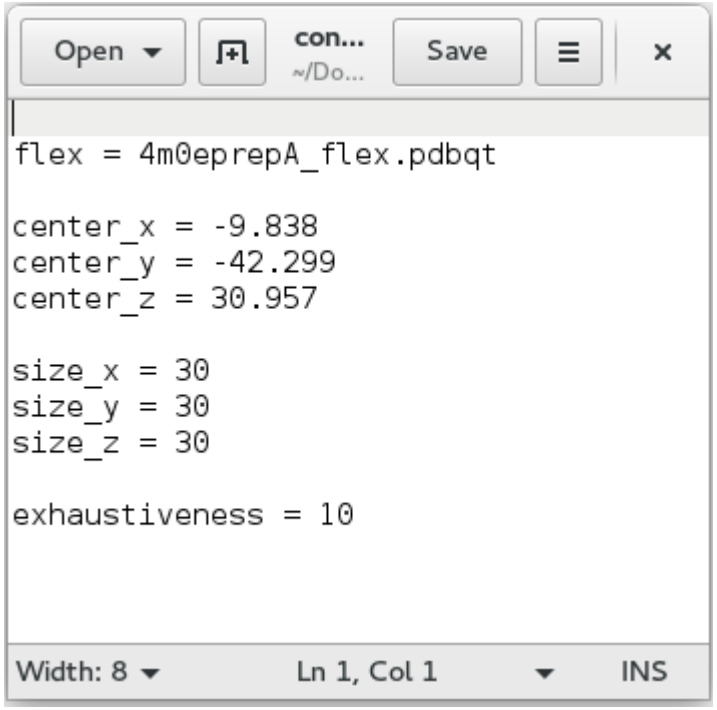
Assigning Dimensions for your Docking Search Space

- **Use the grid box feature visualize what dimensions will encompasses the flexible residues you have selected and be appropriate to search for potential binding poses**
- Grid → grid box
  - Change Spacing to 1.000 (for Å)
  - Adjust coordinates and size so that box encompasses flexible residues (aka binding pocket)
  - Record dimensions and coordinates!!
    - These will define where the docking algorithm should look for potential binding poses
    - You will need to put them into your configuration file
  - Close ADT





Setting up your config file and executing a docking simulation with Vina



config_4m0eA.txt

Create a configuration file in your favorite text editor (as shown on the left)

Assign:
- -flex file
- -grid box coordinates
- -grid box size
- -exhaustiveness

Transfer files into Bound folder:
- -4m0eAprep_flex.pdbqt
- - 4m0eAprep_rigid.pdbqt
- -config_4m0eA.txt
- -TPP.pdbqt

Run your Docking Simulation in Vina!!

vina --receptor *4m0eAprep_rigid.pdbqt* --ligand *TPP.pdbqt* --config *config_4m0eA.txt* –log *TPP.log*
(or ./xRUNVINA 1qkm_Rigid.pdbqt conf.txt)

If “unknown option flex” error run

vina --receptor *4m0eAprep_rigid.pdbqt* –flex 4m0eAprep_flex.pdbqt
--ligand *TPP.pdbqt* --config *config_4m0eA.txt* –log *TPP.log*

Coordinates for each pose and flexible residues will be in TPP_out.pdbqt
Summary tables of the results are found in TPP.log
For a summary of all the flags in vina type “vina --help”

xFLEXRESPREP_v3 Check

Edit xFLEXRESPREP_v3 to make sure your UNK digit (either 0 or 1) matches your TPP_out.pdbqt file. If they do not match your TPP will not show up in your poses.

xFLEXRESPREP_v3

```
#!/bin/csh -e
# input flex file and protein pdb as arguments 1 and 2

set flexfile = ${argv[1]}
set protpdb = ${argv[2]}

alias MATH 'set \!:1 = `echo "\!:3-$" | bc -l`'

@ conf = `egrep MODEL ${flexfile} | wc -l` # no of poses
@ totres = `egrep -n "BEGIN_RES" ${flexfile} | wc -l`
egrep -n "BEGIN_RES" ${flexfile} | sed s/':'/' '/ > all.res
@ uniqres = ${totres} / ${conf}

head -$uniqres all.res > uniqres.tmp
set line = `egrep "BEGIN_RES" uniqres.tmp`
@ tot = ${#line}
@ count = ${tot} / 5

@ totlig = `egrep "UNK     0" ${flexfile} | wc -l`
egrep "UNK     0" ${flexfile} > uniqlig.tmp
@ uniqlig = ${totlig} / ${conf}
```

TPP_out.pdbqt

```
MODEL 1
REMARK VINA RESULT:     -11.1      0.000      0.000
REMARK  6 active torsions:
REMARK  status: ('A' for Active; 'I' for Inactive)
REMARK    1  A    between atoms: P_1  and  O_4
REMARK    2  A    between atoms: P_1  and  O_2
REMARK    3  A    between atoms: P_1  and  O_3
REMARK    4  A    between atoms: O_2  and  C_6
REMARK    5  A    between atoms: O_3  and  C_8
REMARK    6  A    between atoms: O_4  and  C_7
ROOT
HETATM    1  P   UNK     0     -13.226 -44.215  31.038
HETATM    2  O   UNK     0     -13.824 -44.118  32.395
ENDROOT
BRANCH   1   3
HETATM    3  O   UNK     0     -12.324 -42.871  31.008
BRANCH   3   4
HETATM    4  C   UNK     0     -11.286 -42.686  30.135
HETATM    5  C   UNK     0      -9.950 -42.814  30.558
HETATM    6  C   UNK     0      -8.885 -42.719  29.644
HETATM    7  C   UNK     0      -9.143 -42.385  28.301
HETATM    8 Cl   UNK     0      -7.876 -42.298  27.183
HETATM    9  C   UNK     0     -10.461 -42.087  27.905
HETATM   10  C   UNK     0     -11.511 -42.187  28.838
```

Combine the docking poses obtained from TPP4Cl_out.pdbqt with the Rigid receptor (4m0eAprep_rigid.pdbqt) to obtain a structure file for each pose bound to the receptor

- Use the xFLEXRESPREP script to add the coordinates of the ligand and flexible residues to the Rigid pdb for each pose:

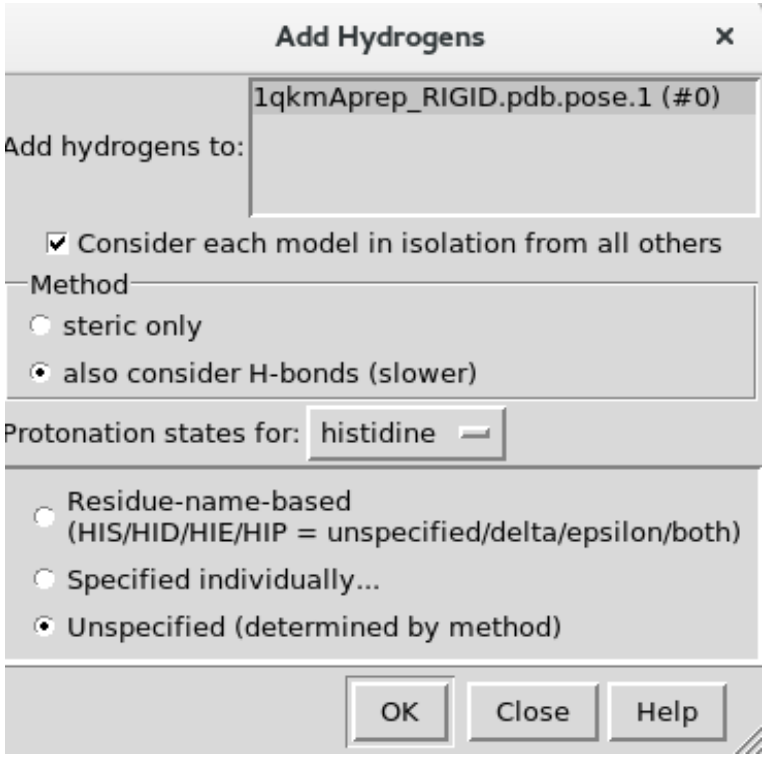
./xFLEXRESPREP_v3 TPP_out.pdbqt 4m0eAprep_rigid.pdbqt

Use Chimera to protonate the structure (for docking we merged all non-polar hydrogens)

Open resulting pdb's in Chimera:

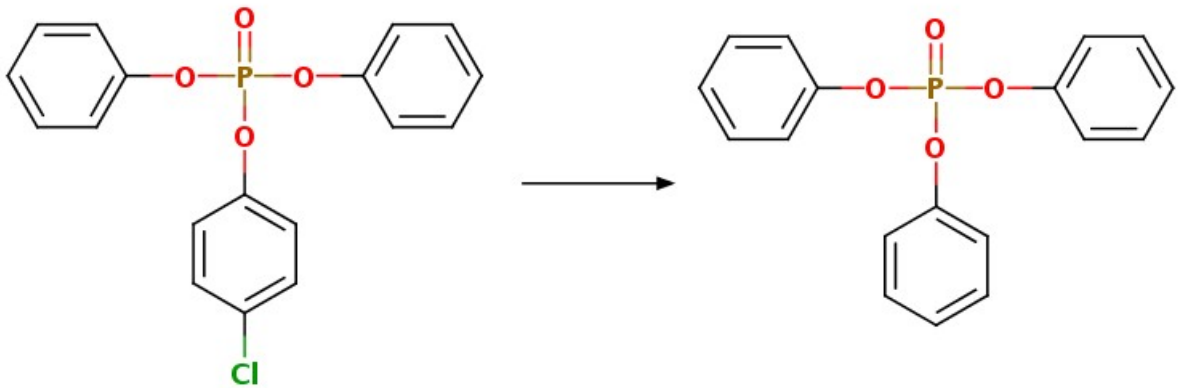
-Tools → Structure Editing → AddH

Check unspecified (determined by method)



Save structures as .pdb files (you will only need the top pose for this tutorial)

Make zmatrix for your FEP that includes parameters for both the final and initial structures of your ligand

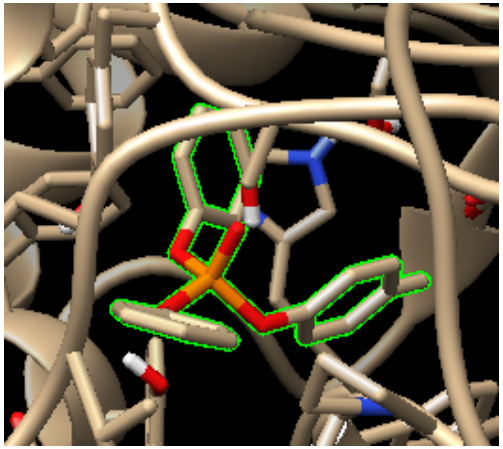


initial

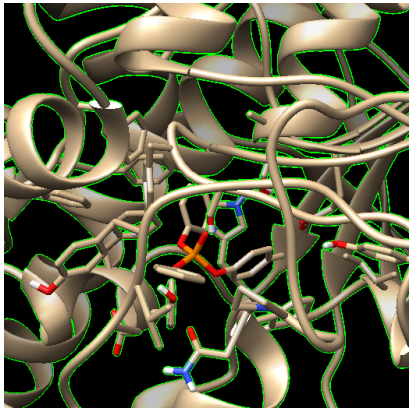
final

Step 1: Isolate your ligand in correct pose to obtain a pdb for your initial state

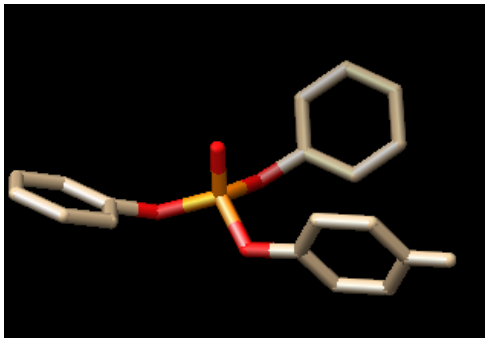
Save the structure of the ligand (in the correct pose) without the receptor as a pdb:
Open pdb file (with ligand, flex residues, and hydrogens) in Chimera
Select → Residue → UNK
Select → Invert (selected models)
Actions → Atoms/Bonds → delete
File → Save PDB… save as “TPP4Clpose1.pdb”



Select ligand “UNK”



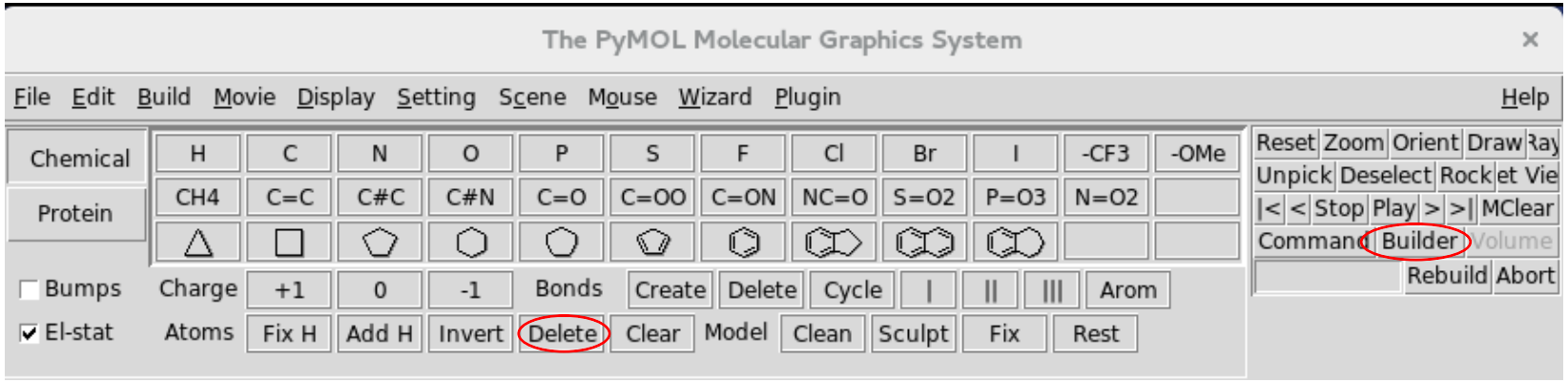
Invert selection so that everything but the ligand is selected



Delete selected atoms leaving only the ligand

Step 1: Use the ligand of your initial state in the correct pose to obtain a pdb for your final state

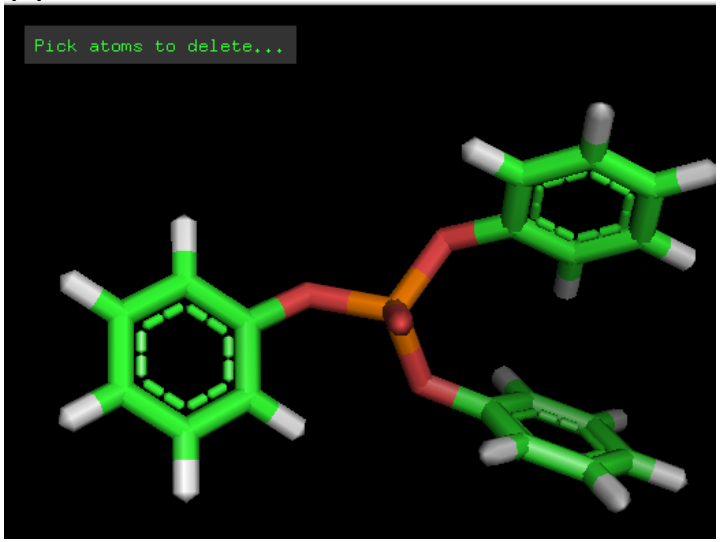
Use pymol to change the Cl atom to an H atom and save as the pdb for your final state:
Click on “Builder” in the top right corner to pull up the builder menu



Then select the “Delete” button under the “Atoms” menu

Note: make sure your rings are cyclized or it will not correctly protonate when the Cl is deleted

Click on the Chlorine atom to delete (replace with H)



Save your final state (TPP) as a pdb:
File → Save molecule → Select “TPP4Clpose1” as object you'd like to save → Save as “TPP4Hpose1.pdb”

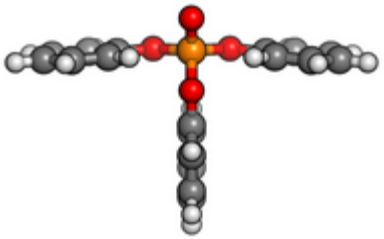
Close pymol

Generating zmatrix coordinates and parameters for the pdb files of your intial and final ligand structures

There are 2 ways (at least) to do this:

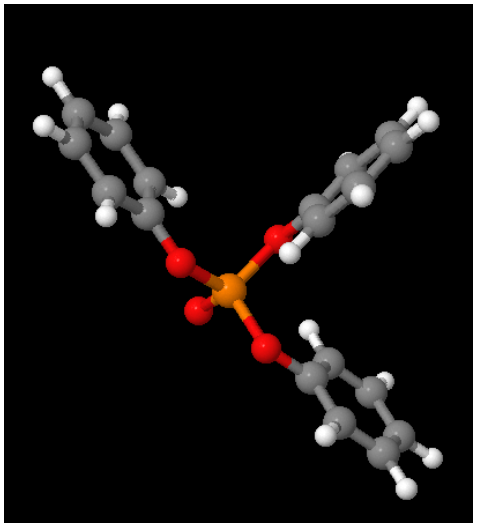
1) Ligpargen server: http://zarbi.chem.yale.edu/ligpargen/index.html

Upload the pdb file of your initial state
Click “submit molecule”
Download as a BOSS/MCPRO ZMAT (and rename “TPP4Clpose1.z)
Repeat for final state



Ligand submitted

Confirm that the structure of your ligand is correct



You can rotate to see the entire geometry in the Jsmol window

BOSS/MCPRO

ZMAT

Download as an MCPRO/BOSS compatible zmatrix

Generating zmatrix coordinates and parameters for the pdb files of your intial and final ligand structures

There are 2 ways (at least) to do this:

2) xPDBMCP script (use this method if the server is down)

Execute the script by typing “./xPDBMCP TPP4Clpose1
Check that the structure is reasonable via the .plt file that is output
Repeat for both states

Open the .z files in a text editor like vim and add an extra blank line at the end of the file

*this is necessary for compatibility with the python script we will use to generate our FEP zmatrix
*it is written to take the zmatrices produced by ligpargen as inputs which are formatted to have an extra line at the end of the file

Generate a single FEP zmat with parameters from both initial and final zmatrices of your ligand

The python script “make_single_topology will produce the FEP zmat with both parameters

Atom type codes:
800 = initial zmat
9500 = final zmat

Format: python make_single_topology -i initial.z -f final.z -n new.z

Ex) **python make_single_topology -i TPP4Cl.z -f TPP.z -n TPP4ClToH.z**

```
BOSS Z-Matrix                                                Tot. E =     53.3
   1 DUM    -1    0    0    0.000000   0    0.000000   0    0.000000 UNK    1
   2 DUM    -1    0    1    1.000000   0    0.000000   0    0.000000 UNK    1
   3 O     800 9500    2    1.000000   1   90.000000   0    0.000000 UNK    1
   4 P     801 9501    3    1.722305   2   90.000000   1    0.000000 UNK    1
   5 C     802 9502    3    1.401932   2  145.439580   4 -179.822120 UNK    1
   6 O     803 9503    4    1.722075   3  109.609870   5  -59.927370 UNK    1
   7 O     804 9504    4    1.722054   3  109.636310   6  119.611260 UNK    1
   8 O     805 9505    4    1.511873   3  109.618900   6 -120.195160 UNK    1
   9 C     806 9506    6    1.402610   4  124.699620   3  -54.639400 UNK    1
  10 C     807 9507    9    1.410299   6  122.957420   4   -2.982750 UNK    1
  11 C     808 9508    9    1.401209   6  117.995850  10  179.897830 UNK    1
  12 C     809 9509   10    1.398758   9  120.215680   6  179.667940 UNK    1
  13 H     810 9510   10    1.080165   9  121.491040  12  179.843560 UNK    1
  14 C     811 9511   12    1.396905  10  120.154390   9    0.113330 UNK    1
  15 H     812 9512   12    1.082383  10  119.949520  14  179.955670 UNK    1
  16 C     813 9513   14    1.396681  12  119.965060  10    0.044420 UNK    1
  17 H     814 9514   14    1.082433  12  120.027950  16  179.972890 UNK    1
  18 H     815 9515   16    1.082341  14  119.968440  12 -179.994760 UNK    1
  19 H     816 9516   11    1.082800   9  119.908790   6    0.087420 UNK    1
  20 C     817 9517    7    1.402508   4  124.709820   3   54.568680 UNK    1
  21 C     818 9518   20    1.410049   7  122.962760   4    2.825260 UNK    1
  22 C     819 9519   20    1.400869   7  117.987340  21 -179.893490 UNK    1
  23 C     820 9520   21    1.398484  20  120.234310   7 -179.789680 UNK    1
  24 H     821 9521   21    1.080068  20  121.475700  23 -179.711220 UNK    1
  25 C     822 9522   23    1.398074  21  120.184220  20   -0.013600 UNK    1
  26 H     823 9523   23    1.082683  21  119.690970  25 -179.957530 UNK    1
  27 C     824 9524   25    1.397808  23  119.862910  21   -0.047910 UNK    1
  28 ClH   825 9525   25    1.754200  23  120.068360  27 -179.967620 UNK    1
  29 H     826 9526   27    1.082617  25  120.205370  23 -179.939350 UNK    1
  30 H     827 9527   22    1.082652  20  119.891380   7   -0.127190 UNK    1
  31 C     828 9528    5    1.410130   3  122.927240   4   -0.128480 UNK    1
  32 C     829 9529    5    1.401046   3  118.000490  31 -179.965350 UNK    1
  33 C     830 9530   31    1.398553   5  120.198990   3 -179.954260 UNK    1
  34 H     831 9531   31    1.080164   5  121.488040  33 -179.985950 UNK    1
  35 C     832 9532   33    1.396944  31  120.158590   5   -0.017330 UNK    1
  36 H     833 9533   33    1.082424  31  119.942960  35  179.968440 UNK    1
  37 C     834 9534   35    1.396664  33  119.968160  31   -0.009940 UNK    1
  38 H     835 9535   35    1.082345  33  120.017610  37  179.994740 UNK    1
  39 H     836 9536   37    1.082276  35  119.981900  33  179.947850 UNK    1
  40 H     837 9537   32    1.082606   5  119.904440   3    0.069400 UNK    1
                     Geometry Variations follow    (2I4,F12.6)
00280001 1.08
                     Geometry Variations follow    (2I4,F12.6)
00280001 1.08
```

FEP ZMAT

Nonbond parameters for final zmat

9513	6	CA	-0.103092	3.550000	0.070000
9514	1	HA	0.153422	2.420000	0.030000
9515	1	HA	0.158838	2.420000	0.030000
9516	1	HA	0.166533	2.420000	0.030000
9517	6	CA	0.177581	3.550000	0.070000
9518	6	CA	-0.221176	3.550000	0.070000
9519	6	CA	-0.156880	3.550000	0.070000
9520	6	CA	-0.103108	3.550000	0.070000
9521	1	HA	0.166524	2.420000	0.030000
9522	6	CA	-0.172436	3.550000	0.070000
9523	1	HA	0.158841	2.420000	0.030000
9524	6	CA	-0.117235	3.550000	0.070000
9525	1	HA	0.153427	2.420000	0.030000
9526	1	HA	0.156751	2.420000	0.030000
9527	1	HA	0.165926	2.420000	0.030000
9528	6	CA	-0.238937	3.550000	0.070000
9529	6	CA	-0.150805	3.550000	0.070000
9530	6	CA	-0.097692	3.550000	0.070000
9531	1	HA	0.151176	2.420000	0.030000
9532	6	CA	-0.170524	3.550000	0.070000
9533	1	HA	0.159759	2.420000	0.030000
9534	6	CA	-0.112415	3.550000	0.070000
9535	1	HA	0.156151	2.420000	0.030000
9536	1	HA	0.159566	2.420000	0.030000
9537	1	HA	0.170999	2.420000	0.030000

Setup the FEP calculation for unbound ligand

```
            Geometry Variations follow    (2I4,F12.6)
00280001 1.08
            Variable Bonds follow        (I4)
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 29
```

Remove atom to be mutated from variable bonds

Create a folder: **TPP4ClItoH.fep**
Copy the following items into **TPP4ClItoH.fep:**

- **TPP4ClItoH.z**
- fepcmd
- feppar
- feppar0

Assign **TPP4ClItoH.z** as your zmatrix in fepcmd file:

setenv ZMATRIX TPP4ClItoH.z

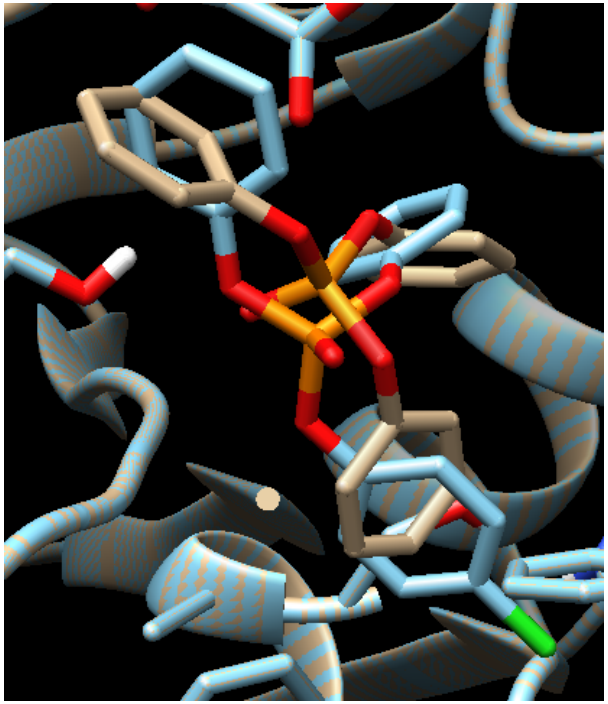
Execute calculation in Separate folder:

csh fepcmd >& log &

CLU
(complex ligand utility)

clu -t 4m0eTPPClpose1.pdb -r TPP4Clpose1toH.z -n 4m0eTPP4Clpose1_cplx.pdb

Replaces ligand (TPP) in pdb specified by “-t” flag with that specified by “-r” flag creating a new pdb specified by the “-n” flag



****Be sure to overlay structures in Chimera to see if your ligand is not in a substantially different position**

→ If structures match ignore the “Unmatched record in replacement ligand file :
Atom C UNK 1. -0.8. 1. 0” code

For more options and info see “Users guide to clu” in MCPRO manual

Chop

Execute chop on in text mode on pdb

$home/username/mcpro2016/mcpro/miscexec/chop -u -i 4m0eTPP4Clpose1_cplx.pdb
chop> add center :UNK **adds center of ligand (:C01@C01)**
chop> set cap origin :c01 **makes this center the origin of water cap**
chop> set cut origin ligand **define cut from all atoms in ligand**
chop> set cut size 15 **cuts residues within 15Å**

213 Residues, Total Charge = -8 ##for tutorial okay if it doesn't match (-4 for mine)
7ASP(-), 11GLU(-), 2LYS(+), 8ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)

chop> fix chains **completes chains**

232 Residues, Total Charge = -9
9ASP(-), 12GLU(-), 2LYS(+), 10ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)
227 Residues, Total Charge = -9
9ASP(-), 12GLU(-), 2LYS(+), 10ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)
Warning: not all chains could be fixed automagically, please do it manually
chain #7 [ARG(247A)] is only 1 residues long, chain #14 [TRP(480A)] is only 1 residues long, chain #15 [LEU(524A)-ARG(525A)] is only 2 residues long

Chop

Warning: not all chains could be fixed automagically, please do it manually
chain #7 [ARG(247A)] is only 1 residues long, chain #14 [TRP(480A)]
is only 1 residues long, chain #15 [LEU(524A)-ARG(525A)] is only 2 residues long

chop> delete cut :247a **deletes residue 247A**

226 Residues, Total Charge = -10
9ASP(-), 12GLU(-), 2LYS(+), 9ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)

chop> delete cut :524a

225 Residues, Total Charge = -10
9ASP(-), 12GLU(-), 2LYS(+), 9ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)

chop> delete cut :525a

224 Residues, Total Charge = -11
9ASP(-), 12GLU(-), 2LYS(+), 8ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)

chop> delete cut :480a

223 Residues, Total Charge = -11
9ASP(-), 12GLU(-), 2LYS(+), 8ARG(+), 0HIP(+)
0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 6HIS(0), 0HID(0), 0HIE(0)

Chop

```
chop> fix chains                                    completes chains
chop> cap all                                       adds Ace and Ame neutral caps
  223 Residues, Total Charge = -12
  8ASP(-), 12GLU(-), 2LYS(+), 6ARG(+), 0HIP(+)
  0ASH(0), 0GLH(0), 0LYN(0), 0ARN(0), 5HIS(0), 0HID(0),
0HIE(0)
chop> set variable origin ligand                    defines variables from all atoms in ligand
chop> set variable size 10                          makes residues beyond 10Å fixed
chop> fix charge +0                                 neutralizes enough residues to reach the
  Target charge = +0, Current charge = -12          target charge given. Target charge should be
  isOK = 1                                          assigned so that the charge of the ligand +
  223 Residues, Total Charge = +0                   protein = 0
chop> write pdb 4m0eTPP4Clpose1.chop.pdb            writes the new pdb file
chop> write pepz all 4m0eTPP4Clpose1.chop.all.in    writes pepz input for minimization
chop> write pepz variable 4m0eTPP4Clpose1.chop.var.in writes pepz input for simulation
chop> write translation 4m0eTPP4Clpose1.chop.tt     Writes translation table file
chop> exit                                          stops chop
```

Use script to check for missing atoms:

./xRESINTAN_v5

AA residue integrity analyzer
****
USE ON CHOP GENERATED PDBS ONLY

Input pdb file to examine:
4m0eTPP4Clpose1.chop.pdb

If you do not receive an output message, all residues are intact

cp 4m0eTPP4Clpose1.chop.all.in 4m0eTPP4Clpose1.all.in
cp 4m0eTPP4Clpose1.chop.var.in 4m0eTPP4Clpose1.var.in

4m0eTPP4Clpose1.all.in:

```
$ title [ADD YOUR TITLE HERE]
$ read database  $MCPRODir/AA/oplsaa.db
$ read dihedrals $MCPRODir/AA/dihedrals.aa
$ read parameter $MCPRODir/AA/oplsaa.par
$ read boss [WRITE NAME OF YOUR solute z-matrix FILE]
$ set parameter type ALL *
$ set override domain 1-223
$ sequence
ACE GLN SER VAL CYS TYR GLN TYR VAL ASP THR LEU TYR PRO GLY
PHE GLU GLY THR GLU MET TRP ASN PRO ASN ARG GLH LEU SER GLH
ASH CYS LEU TYR LEU ASN VAL AME TER ACE TRP ILE TYR GLY GLY
GLY PHE TYR SER GLY ALA SER SER LEU ASP VAL TYR ASH GLY ARG
PHE AME TER ACE VAL SER MET ASN TYR ARG VAL GLY ALA PHE GLY
AME TER ACE VAL GLY LEU LEU ASH AME TER ACE GLY GLU SER ALA
GLY ALA ALA SER VAL GLY MET AME TER ACE LEU GLN SER GLY ALA
PRO ASN GLY PRO TRP ALA THR AME TER ACE GLN VAL LEU VAL ASN
HIS GLH TRP HIS VAL LEU PRO GLN GLH SER VAL PHE ARG PHE SER
PHE VAL PRO AME TER ACE GLY VAL VAL LYS ASH GLU GLY SER TYR
PHE LEU VAL TYR GLY ALA PRO GLY PHE SER LYS AME TER ACE VAL
ARG VAL GLY VAL AME TER ACE GLH ALA LEU SER ASH VAL VAL GLY
ASH HIS ASN VAL VAL CYS PRO VAL ALA GLN AME TER ACE VAL PHE
GLH HIS ARG ALA SER THR LEU SER TRP PRO LEU TRP MET GLY VAL
PRO HIS GLY TYR GLU ILE GLU PHE ILE PHE GLY ILE AME TER ACE
ASN AME TER ACE ALA PHE TRP AME TER UNK TER CAP
$ center
$ set variable all 1-222
$ read pdb 4m0eTPP4Cl.chop.pdb
$ write pdb [NAME OF pdb file TO BE WRITTEN]
$ write zmatrix [NAME OF THE z-matrix TO BE WRITTEN]
```

AchE/TPP4CltoHpose1

TPP4CltoH.z

Delete this line

AChETPP4CltoHpose1all.z
(AChETPP4CltoHpose1var.z)

Make the same edits to AChETPP4CltoHpose1.var.in but change name of zmat to be written(all→var)

PEPZ

./xPEPZ filename.all → example: ./xPEPZ 4m0eTPP4CltoHpose1.all

```
if ( ! ( 1 ) ) then
if ( ! ( -e 4m0eTPP4Cl.all.in ) ) then
/home/klm2/mcpro2016/mcpro/miscexec/pez.old -i 4m0eTPP4Cl.all.in -o 4m0eTPP4Cl.all.out
   WARNING: residue 214(ACE) cannot be moved in a conrotatory fashion but it contains backbone variables; MC moves will be used
   WARNING: residue 215(ASN) cannot be moved in a conrotatory fashion but it contains backbone variables; MC moves will be used
   WARNING: residue 216(AME) cannot be moved in a conrotatory fashion but it contains backbone variables; MC moves will be used
   WARNING: residue 222(UNK) cannot be moved in a conrotatory fashion but it contains backbone variables; MC moves will be used
   WARNING: residue 223(CAP) cannot be moved in a conrotatory fashion but it contains backbone variables; MC moves will be used
exit
```

PEPZ writes a zmatrix with all degrees of freedom variable based on the instructions in 4m0eTPP4Clpose1.all.in

./xPEPZ filename.var → example: ./xPEPZ 4m0eTPP4CltoHpose1.var

PEPZ writes a zmatrix with only the degrees of freedom selected in chop variable based on the instructions in 4m0eTPP4Clpose1.var.in

Relax “Chopped” Complex

We need to optimize the zmatrix of the ligand/protein cplx we just created before FEP calculations

Delete all Geometry variations from AChETPP4CltoHpose1all.z:

```
                    Geometry Variations follow    (2I4,F12.6)
  3140   1    1.080000
```

These lines specify the geometry changes for the FEP (Cl→H)

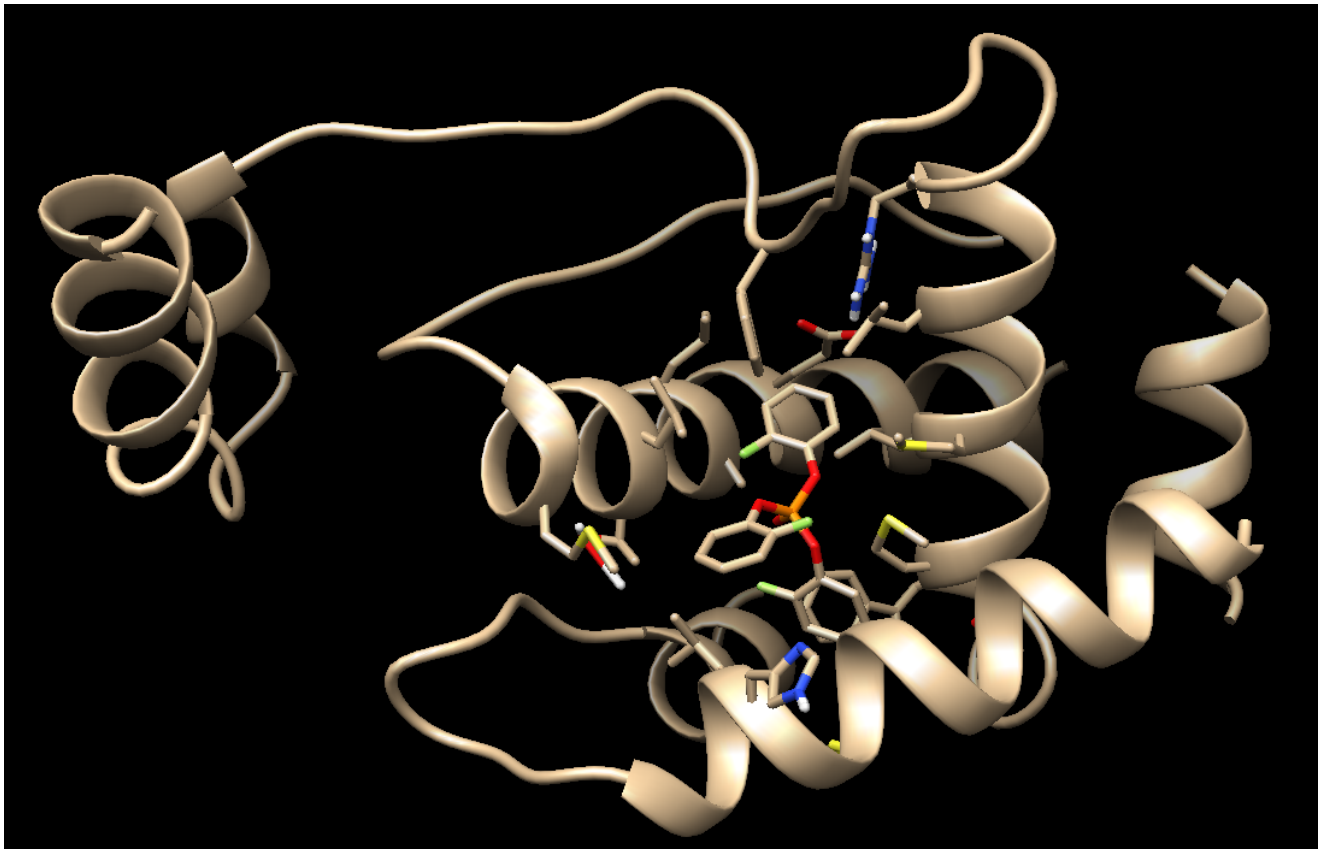
They will cause problems in our optimization if left in

Relax protein/ligand complex with 30 steps of conjugate gradient minimization

Move OPTCG9cmd, OPLSCH9par to folder from test job cdk2
(will get error if not copied)

./xOPTCG9 AChETPP4CItoHpose1all

Output: AChETPP4CItoHpose1all.out (output file)
AChETPP4CItoHpose1all.sum (optimized z-matrix)
AChETPP4CItoHpose1all.pdb (pdb formatted structure)



- Use pdb to visually check structure
- Find a centrally located residue/atom of the protein and record it's atom id (ex. Ser48/OG)

Replace the coordinates of the **AChETPP4CItoHpose1var.z** with those from the sum file and save as **AChETPP4CItoHpose1cap.z**

Open AChETPP4CItoHpose1cap.z in a text editor

- **Delete all "TERZ" except the last 2**
- **Find the residue/atomid you recorded earlier:**

723 OG 154 154 722 1.417039 719 114.213488 717 77.001169 SER 48

- **Record atom # (723)**
- **Record the atom number for a central atom of the ligand (ex "P" 3116)**
- **Remove the atoms you are mutating from from the variable bonds list:**

```
                Geometry Variations follow    (2I4,F12.6)
3140   1    1.080000
                Variable Bonds follow         (I4)
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3141
```

Delete 3140

JAWS (Just Add Water Molecules)

Use JAWS to generate solvent coordinate input file (this gives you buried waters that other methods may not include)

•

Create a folder: **AChETPP4CltoHpose1cap**

Copy the following items into **AChETPP4CltoHpose1cap:**

- **AChETPP4CltoHpose1cap.z**
- Jaws_cmd
- phase0.par
- phase1.par
- phase2.par

Edit **par** files phase0.par, phase1.par, phase2.par as indicated in red:

```
NCENT1, NCENT2  (THE ATOMS USED TO DEFINE THE CENTER OF THE SOLUTE(S))
  07233116                        INTEGERS IN I4 UNLESS NOTED.
NROTA1, NROTA2  (THE ATOMS SOLUTES 1 & 2 ARE ROTATED ABOUT - NOTE:
  07233116          ROTATIONS DO NOT CHANGE THEIR RELATIVE POSITIONS)
```

Atom #'s recorded earlier (Ser48/OG & P atom of ligand)

```
GSTEP, GSIZE, NGSKIP, NGRESTR, NTARGET, GRIDDIFF, NGRIDATOMS (-1 means solute 2)
  1.0      3.0     0  0  0  0  -1
```

JAWS

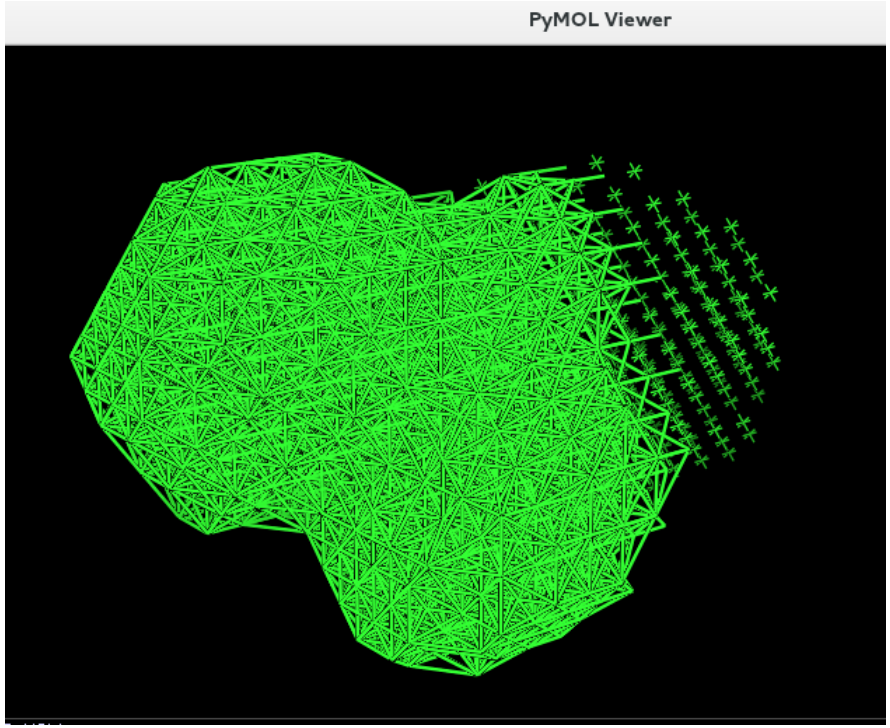
Give the command file (Jaws_cmd) the name of your zmat:

Name of the Z-matrix file:

setenv ZMATRIX **AChETPP4CltoHpose1cap.z**

Run JAWS:**csh Jaws_cmd >& Jaws.log &**

See "ReadMe file in $MCPRODir/testjobs/JAWS/neuraminidase for more details about input/output



GR-phase0.001.pdb
Gridpoints where waters were observed

# of predicted waters (record this number)

JAWS-all.in
~/mcpro2016/mcpro/testjobs/JAWS/neuraminidase/AChETPPcap

```
 435 3306     3   644    37   6165000000028343440   419642       21
   25.00      1.00 500.00000000 500.00000000 500.00000000
-0.806518011420D+04 -0.536206472792D+01 -0.407229608036D+04 -0.407229608036D+04
-0.407229608036D+04 -0.596761664913D+02 -0.596761664913D+02 -0.596761664913D+02
 0.199658274269D+03  0.199658274269D+03  0.199658274269D+03  0.000000000000D+00
 0.000000000000D+00  0.000000000000D+00  0.000000000000D+00  0.000000000000D+00
 0.000000000000D+00  0.167932107685D+02  0.167932107685D+02  0.167932107685D+02
 0.343484429314D+03  0.343484429314D+03  0.343484429314D+03
 0.384864605541D+02  0.146135637552D+02  0.188268226866D+02  0.210789755709D+02
-0.571049541960D+01 -0.830890039673D+00 -0.200911899391D+02 -0.395849765522D+02
 0.000000000000D+00  0.000000000000D+00  0.000000000000D+00
  0.00000000   1.00000000   1.00000000  -3.38213267  -2.97722204  -4.06465262
 -3.50314212  -2.37996796  -2.43612960  -5.04551532  -5.74596983  -6.39104425
```

FEP setup (bound)

Create a folder: **AChETPP4CItoHcap.fep**
Copy the following items into **AChETPP4CItoHcap.fep:**

- **AChETPP4CItoHcap.z**
- JAWS-all.in
- FEP_q.cmd
- CAPpar
- SLVpar

Edit SLVpar & CAPpar as follows:

NMOL **# of water molecules from JAWS-all.in**
435

```
NCENT1, NCENT2  (THE ATOMS USED TO DEFINE THE CENTER OF THE SOLUTE(S))
  07233116                    INTEGERS IN I4 UNLESS NOTED.
NROTA1, NROTA2  (THE ATOMS SOLUTES 1 & 2 ARE ROTATED ABOUT - NOTE:
  07233116          ROTATIONS DO NOT CHANGE THEIR RELATIVE POSITIONS)
```

Give the command file (FEP_q.cmd) the name of your zmat and make sure it is set to 11 windows with double-wide sampling:

Name of the Z-matrix file:

setenv ZMATRIX **AChETPPCItoHpose1cap.z**

***Get edited version of FEP_q.cmd from me**

set numwin = 28
@ dw = 1

FEP setup (bound)

Sign onto ColonialOne

Execute command file: **csh FEP_q.cmd_JK >& FEP_q.log &**

This will create a directories with all the necessary files for each window of the FEP

Create submission script with xDO2ALL_v7:
xDO2ALL_v9 mc2q 0-20 fepcmd (fix paths)

Execute!
./ALL2Q (make ALL2Q and submit files executable)

Check Status of your calculation:
Type "squeue"

Use xDELG to get ΔΔG: ./xDELG ERTPP3x2FtoHpose1cap

****(move outputs for each window to parent folder first "mv l*/ *.")**

You may delete *.in *.up *.sv and *.av files but be sure to keep *.sum *.out and *.pdb

Use Chimera, Pymol, or VMD to visualize the pdb files for each window to be sure the correct mutation has occured