**Docking preparation procedure**

The expected output:

1. Initial PDB file – 2btr.pdb

2. Protonated protein structure in PDB with all hydrogens explicit – 2btr\_protein.pdb

3. The same structure in PDBQT format - 2btr\_protein.pdbqt

4. Protonated ligand structure in PDB – 2btr\_ligand.pdb

5. Protonated ligand structure in MOL(SDF) – 2btr\_ligand.mol

6. The docking box is only created if the structure in the first one. If structures of this protein were previously prepared the old docking box can be used, because all structures should be pre-aligned before storage.

PDB download

**By script:**

python scripts/get\_pdb\_fasta\_mol\_bypdbid.py -i 5tgz 5u09 -o P21554

*it returns:*

P21554/

ligands\_frompdb.smi (can use for pdb2mol script)

P21554/5tgz/

*5tgz.pdb*

*5tgz.fasta*

*5tgz\_ligands\_frompdb.sdf*

*5tgz\_ligands\_frompdb.smi*

*ligands\_list.log*

**or by PDB downloader**

Usually can be directly fetched in Chimera

Target preparation

**1) Open Chimera**

**2) Fetch PDBIDs (space as separator)**

*File → Fetch by ID*

**3) Select chosen chains (remove other chains)**

*Select* menu

*Select chain (if chains: Select→ Selection mode → append) → Invert (all models) → Actions → Atoms/Bonds → delete*

**4) Align structures**

*Tools → Structure comparison → MatchMaker*

This is an optional step. It is required if at least one structure of the same protein is already present in prepared files. In such a case select the first structure from the prepared ones by alphabetic order and use it as a reference to align a new one. This will simplify analysis of docking to different X-ray structures of the same protein.

**5) Check structures for missing residues (sequence information)**

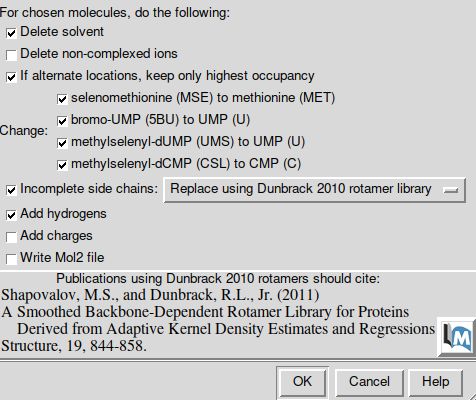
If it necessary build it by *Tools → Structure editing → Model/Refine loops*

Rebuild only non-terminal missing sequences. Check them carefully if they are close or within the binding site.

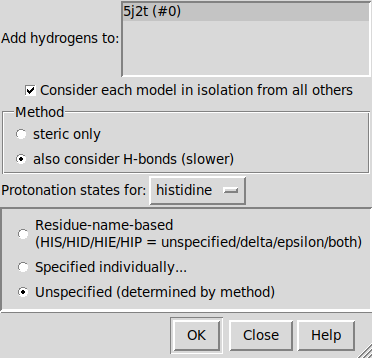
**6) Prepare structure**

*Tools → Structure editing → Dock prep*

Decide if you need to save solvent or ions.



For histidines set to determine protonation state by the method. For other residues this is optional.



**7) check protonation states of a protein**

Check protonation states of polar amino acids, especially histidines. If some hydrogens were incorrectly added (sometimes there are 2 H atoms on histidines making them positively charged, that may be incorrect) they should be removed.

**8) Assign correct labels for histidines (strictly required for MD)**

Proper labels to histidines can be assigned based on their protonation states. Call the following lines in a console:

setattr r type HID :HIS@HD1,DD1,TD1,HND

setattr r type HIP :HID@HE2,DE2,TE2

setattr r type HIE :HIS@HE2

These commands can be combined in a single command by concatenation using “;”

setattr r type HID :HIS@HD1,DD1,TD1,HND; setattr r type HIP :HID@HE2,DE2,TE2; setattr r type HIE :HIS@HE2

or you can skip this step and do it later by script chimera\_assignHIS.py:

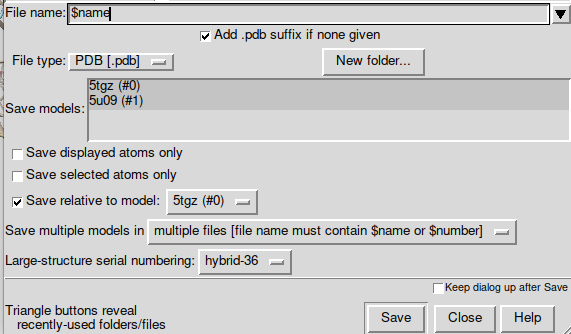
~/.local/UCSF-Chimera64-1.14/bin/chimera --nogui --script " scripts/chimera\_assignHIS.py --input 5tgz.pdb "

**9) Save proteins as pdb**

This will remain a fully corrected structure which can be used as a source for preparation for future simulations.

Saving ligands and water is optional at this stage.

File → Save PDB



Save ligand from pdb as mol file

Strictly required for MD.

**1) Split protein and ligands by script:**

~/.local/UCSF-Chimera64-1.14/bin/chimera --script "scripts/split\_lig\_protein.py --input 5tgz\_H\_HIS.pdb --out 5tgz " --nogui

**script returns:**

5tgz\_protein.pdb – only protein and ions (if present)

5tgz\_FMN\_ligand.pdb – ligand from complex

If complex consists of ligand and cofactor you can set *--resn* FMN argument, so each ligand will be returned, but 5tgz\_protein.pdb will consist of protein, ions (if present) and cofactor to be used for molecular docking. Other file - 5tgz\_protein\_MD.pdb will have only protein and ions (if present) and can be used for MD simulations.

Alternatively the ligand file can be directly saved from Chimera in the pdb format.

**2) Convert pdb to mol**

python scripts/pdb2mol.py -i \*\_ligand.pdb --smi ligands\_frompdb.smi

Convert pdb to pdbqt

**Targets:**

Cmd : *for i in \*.pdb; do ~/bin/mgltools\_x86\_64Linux2\_1.5.6/bin/pythonsh ~/bin/mgltools\_x86\_64Linux2\_1.5.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare\_receptor4.py -r $i -U 'nphs\_lps' -o $i'qt';done*

Important:

1. The default for repairs is to do no repairs. Nonetheless, all hydrogens must be added to the receptor if they have not already been added. The rationale is that the user probably has already added hydrogens. If not, the 'hydrogens' repair option must be used.  
2. Formatting the receptor involves building bonds between atoms which are within vanDerWaals radii of each other to determine the torsionTree. For this reason, some input atom coordinates cannot be processed without using the -A 'bonds' repair option.

prepare\_receptor4.py -r filename

**Input**

The receptor file should contain only one molecule, the one to be formatted as the receptor. It should already have all hydrogens added [see below]

**Output**

The output filename default is the input file *stem* plus .pdbqt. For example, 'hsg1.pdb' by default is written to 'hsg1.pdbqt'.  
Options  
-r receptor\_filename  
Optional parameters:  
[-v] verbose output (default is minimal output)  
[-o pdbqt\_filename] (default is 'molecule\_name.pdbqt')  
[-A] type(s) of repairs to make:   
'bonds\_hydrogens': build bonds and add hydrogens   
'bonds': build a single bond from each atom with no bonds to its closest neighbor  
'hydrogens': add hydrogens  
'checkhydrogens': add hydrogens only if there are none already  
'None': do not make any repairs   
(default is 'None': do not to make any repairs)  
[-C] preserve all input charges ie do not add new charges   
(default is addition of gasteiger charges)  
[-p] preserve input charges on specific atom types, eg -p Zn -p Fe  
[-U] cleanup type:  
'nphs': merge charges and remove non-polar hydrogens  
'lps': merge charges and remove lone pairs  
'waters': remove water residues  
'nonstdres': remove chains composed entirely of residues of  
types other than the standard 20 amino acids  
'deleteAltB': remove XX@B atoms and rename XX@A atoms->XX  
(default is 'nphs\_lps\_waters\_nonstdres')   
[-e] delete every nonstd residue from any chain  
'True': any residue whose name is not in this list:  
['CYS', 'ILE', 'SER', 'VAL', 'GLN', 'LYS', 'ASN',   
'PRO', 'THR', 'PHE', 'ALA', 'HIS', 'GLY', 'ASP',   
'LEU', 'ARG', 'TRP', 'GLU', 'TYR','MET']  
will be deleted from any chain. NB: there are no   
nucleic acid residue names at all in the list.   
(default is False which means not to do this)  
[-M] interactive   
(default is 'automatic': outputfile is written with no further user input)

Ligands:

*Cmd: for file in \*.pdb;do ~/bin/mgltools\_x86\_64Linux2\_1.5.6/bin/pythonsh ~/bin/mgltools\_x86\_64Linux2\_1.5.6/MGLToolsPckgs/AutoDockTools/Utilities24/prepare\_ligand4.py -l $file -o ${file##\*/}'qt'; done*

Important:

1. The default for repairs is not to do repairs. Nonetheless, all hydrogens must be added to the ligand if they have not already been added. The rationale is that the user probably has already added hydrogens. If not, the 'hydrogens' repair option must be used.  
2. Formatting the ligand involves building bonds between atoms which are within the van der Waals radii of each other to determine the torsion tree. For this reason, some input atom coordinates cannot be processed without using the -A 'bonds' repair option.

prepare\_ligand4.py -l filename

Input

The input file should contain only one molecule, which will be formatted as the ligand. Ideally, it should already have all hydrogens added (but see option '-A' below).

**Output**

The output filename default is the input file stem plus .pdbqt. For example, 'ind.pdb' by default is written to 'ind.pdbqt'.

Options

-A <option>  
-A 'hydrogens'  
adds hydrogens. PyBabel is used for adding all hydrogens, not just polar-hydrogens  
-A 'bonds'  
if, after having built bonds by-distance, there are any atoms that do not have any bonds, this builds a bond between each of these atoms and the atom nearest to it. This is necessary for building the torsion tree.  
-A 'hydrogens\_bonds'  
adds hydrogens and builds bonds to any non-bonded atoms.  
(The default is not to perform any repairs, i.e. to not add hydrogens and not build bonds.)  
  
-C  
does not add Gasteiger partial atomic charges. If this option is used, the input ligand should already have partial atomic charges; the best input formats here would be SYBYL mol2 or AutoDock 3 PDBQ, since these store charges.  
  
-p atomtype  
preserves the input charge on a specific atom type; this is useful for metals where the charge has already been set, e.g. -p Zn

-U <option>  
performs various kinds of clean-up.  
-U 'nphs'  
merges non-polar hydrogens by adding the charge of each non-polar hydrogen to the carbon to which it is bonded and then removes the non-polar hydrogen from the ligand molecule, thus implementing the 'United-Atom' model.  
-U 'lps'  
merges lone-pairs by adding the charge of each lone pair to the atom to which it is 'bonded' and then removes the lone-pair.  
-U 'nphs\_lps'  
merges both non-polar hydrogens and lone pairs. (This is the default.)  
-U ''  
(this is two quote characters, not one double-quote) Does not perform any clean-up   
(The default is 'nphs\_lps')  
  
-B <option>  
defines which types of bonds to allow to rotate  
-B 'backbone'  
allows peptide-backbone-bonds (i.e. phi and psi) are rotatable. (This is the default).  
-B 'amide'  
allows amide bonds to be rotatable. (They are non-rotatable by default).  
-B 'guanidinium'  
allows guanidinium bonds to be rotatable. (They are non-rotatable by default).  
(The default is 'backbone')

-R <integer>  
defines the 0-based index of the atom that will become the root of the torsion tree.  
(The default is to find the root automatically: this will be the atom with the smallest-largest-subtree)  
-F  
checks for and uses largest non-bonded molecule or fragment in the input. Some files contain more than one molecule. By default, the first found is processed. If this option is used, the largest molecule found is used.  
-M  
interactive mode  
This option processes the molecule according to the input options but does not write the outputfile. For example:

pythonsh -i prepare\_ligand4.py -l ind.pdb -M

(The default is automatic mode which exits after writing the output file.)  
-I <string>  
defines the string of bonds to inactivate, defined using a string of zero-based atom indices separated by underscore characters. For example:  
-I '5\_13\_2\_10'  
will inactivate the bond between atoms[5] and atoms[13] and the bond between atoms[2] and atoms[10]   
(Note this flag is the capital i)  
-Z  
inactivates all active torsions. This results in a rigid ligand molecule.

Configuration file

https://pymolwiki.org/index.php/Autodock\_plugin

**1) Open pymol**

**2) Visualize cavites and pockets**

Menu

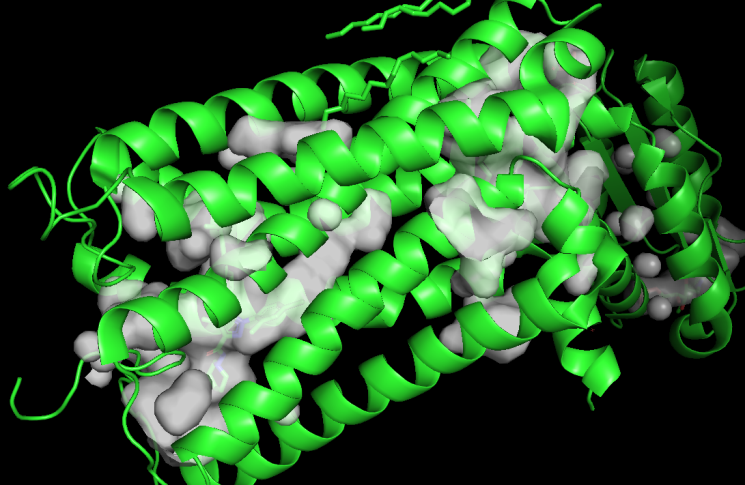
*Setting → Surface → Cavites and Pockets Only*

*Setting → Surface →Color → White*

*Setting → Transparency → Surface → 20%*

Side panel

*S (Show) → Surface*

**

**3) Vina Grid**

Menu

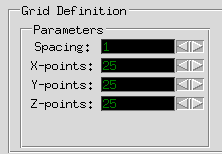
*Plugin → Legacy Plugins → Autodock/Vina*

Grid setting

**1. Grid definition:**

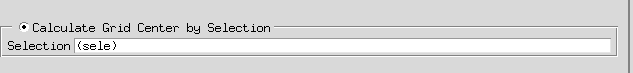
Spacing 1

X, Y, Z to start with enter 25 ( size)



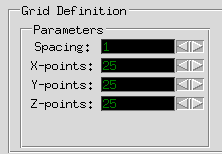
**2. Set coordinates of the grid**

*Select ligand (if we choose box by ligand into active site) → enter (sele) into Grid setting form → press enter*



**3. Set parameters of the grid**

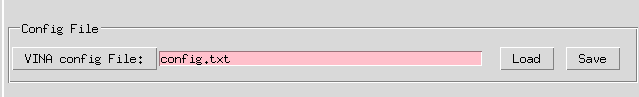
- Set sizes by X- Y- Z- points: (optimal is around 25, but it is dependent from task and active site).



- And set center of the grid by X Y Z ← →

4. Save (or load) config file

Set path and filename by clicking VINA config File (.txt format)



Click Save

\* You can also load previously created config file by Load button.