Chris
2016-07-23
Version 1

Tutorial for Refining an Icosahedral Model into Cryo-EM Density using COOT and PHENIX

Introduction

This protocol attempts to walk through the refinement of a protein model into an icosahedrally averaged cryo-em map. It makes use of Coot, Phenix, and Chimera. The procedure detailed here is complicated by three factors:

1. **The presence of non-protein ligands**

Most molecular modeling software that focus on proteins have very extensive libraries of parameters for protein molecules, which dictate appropriate bond lengths, bond angles, partial charges, etc. If you want to model a non-protein ligand, especially one that is completely novel, you need to supply the molecule parameters yourself. Neither Coot nor Phenix can accommodate non-protein ligands by default, but both provide the opportunity to easily import non-protein parameter files generated elsewhere. Phenix includes an interface to the eLBOW software, which takes a 3D model of your ligand, and generates a reasonable parameter file. However, the ligand structure should be monitored manually throughout the refinement to ensure that it is not adopting chemically unreasonable structures.

2. **The presence of icosahedral symmetry**

A significantly easier approach than the one shown here would be to segment out an asymmetric unit's (ASU) worth of density, and then follow existing Coot/Phenix protocols, treating the ASU as a standalone protein. The problem with this is that the refinement software is unaware that in the real molecule there is protein very close, in fact interacting, with the edges of the ASU. Unless you have truly atomic resolution of density you cannot segment the map perfectly, and during refinement the ASU model will tend to “spread out” into density that may actually belong to a neighboring ASU. Some options to deal with this would be to refine an entire icosahedral capsid asymmetrically, which is prohibitively computationally intensive. Another option is to refine a patch of several ASUs, and then take the coordinates from a centrally located ASU, in effect relying on “inertia” to ensure that the coordinates of the centrally located ASU are not spread out. However, the ideal way to deal with this problem is the one outlined here, where only a single ASU is refined into density, but the refinement software takes into account that there are symmetrically related molecules nearby.

3. **The limitations of the PDB file format**

The published PDB specification allocates a single row of a text file to an atom, with that atom's parameters in specific columns along the row. In the case of icosahedral viruses, two main problems that are encountered are, 1) that there are too many atoms, and parameters like Atom Number start taking up more columns that the formal PDB specification would allow, and 2) there are atoms that have identical properties except for the 3D coordinates, because of the symmetry. Different software packages deal with these problems differently, but often by writing non-standard PDBs. For example, using hexadecimals instead of integers, or using an additional column which is not supposed to be used for a given field. When refinement software encounters an element of a PDB file that it does not anticipate, it crashes. This protocol makes use of a custom Python module and associated scripts, to aid in the conversion of PDBs between the different types expected by each of the main pieces of software in the pipeline. Alternatively, you can edit the PDB text file yourself to satisfy the requirements for each step.

Starting Materials

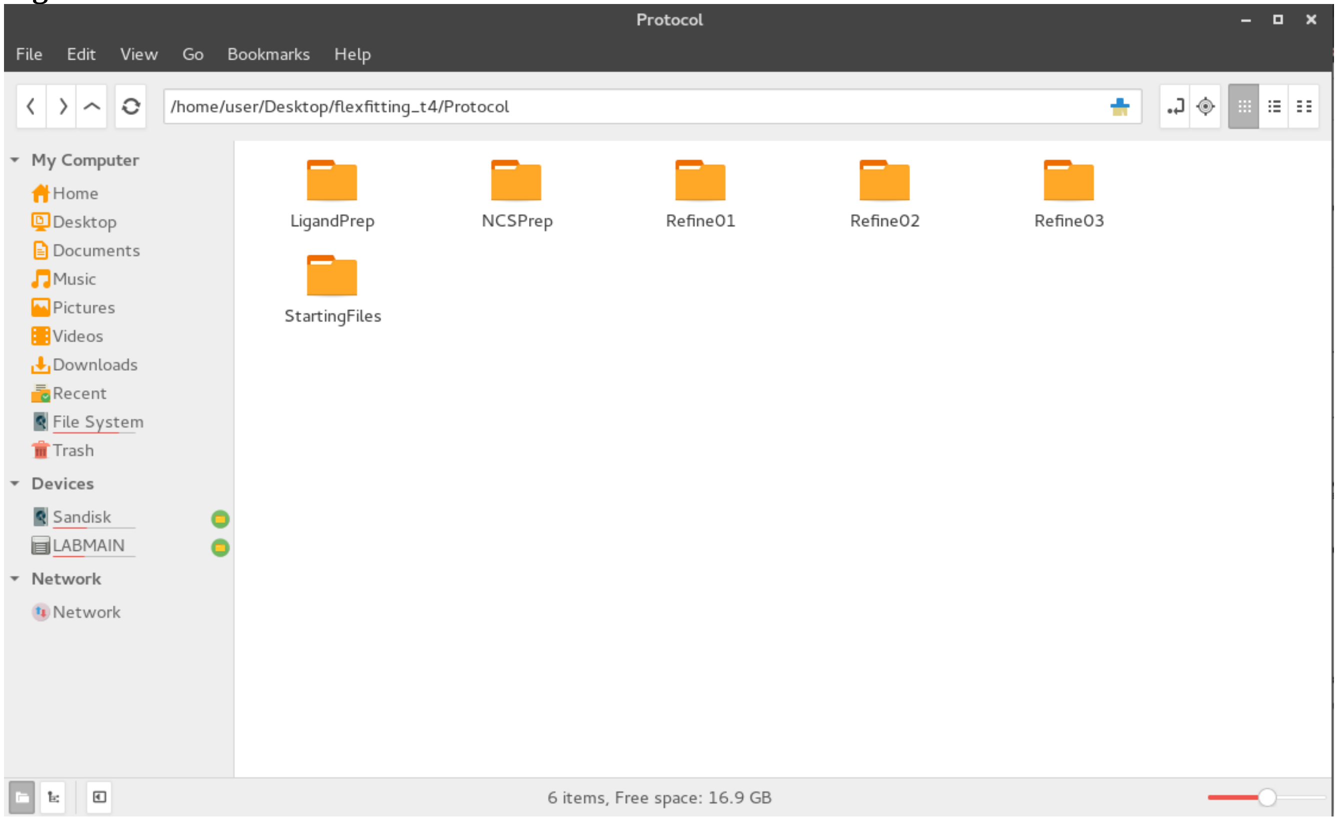
1. An icosahedral starting model of the protein that is close to the final result. For the case of HBV with drugs, these are readily available from previous crystal structures.
2. A 3D model of the ligand.
3. The density map
4. Coot and Phenix and Chimera installed.

It is also helpful to install Pymol. Chimera is better for visualizing the model with the density, but Pymol is much, much, much more tolerant of nonstandard PDB files than Chimera, so it can be useful for troubleshooting PDB problems. Also, most of the Chimera steps are not explained, so a good working knowledge of Chimera is helpful.

Setup

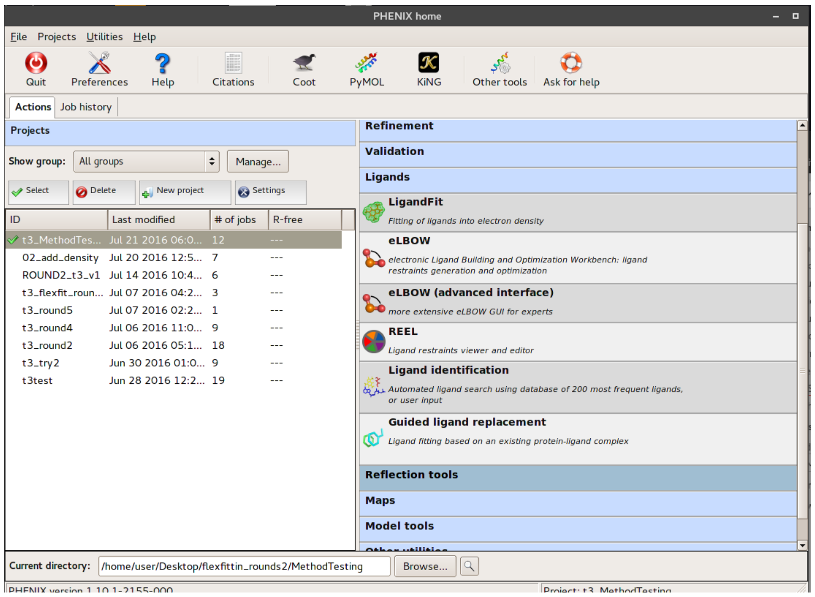
It is useful to use a consistent directory structure to avoid losing track of the large number of files that are generated. In this protocol I will use a directory structure as shown in Figure 1, with separate directories for Starting Files, Ligand Prep, and NCS (symmetry) Prep, followed by iterative refinement directories where each directory includes one round of Coot followed by Phenix, with the final refined PDB file copied to the subsequent iteration directory for another round of Coot and Phenix. If you do more than one round of Coot/Phenix in a single directory, it starts to become impossible to make sense of anything the next day.

Figure 1



eLBOW Ligand Parameterization

If we open the StartingFiles directory we can see our three starting files. The 5d7y.pdb is the HBV T=4 crystal structure in complex with HAP18. This is a good starting point because our ligand is HAP13 with a Tamra fluorophore, so it should have similar structure. The ligand_haptam.pdb file is a 3D model of our non-protein ligand, with all hydrogens and connect records (Chimera>Structure Editing>addH)



ID	Last modified	# of jobs	R-free
✓ t3_MethodTes...	Jul 21 2016 06:0...	12	---
02_add_density	Jul 20 2016 12:5...	7	---
ROUND2_t3_v1	Jul 14 2016 10:4...	6	---
t3_flexfit_roun...	Jul 07 2016 04:2...	3	---
t3_round5	Jul 07 2016 02:2...	1	---
t3_round4	Jul 06 2016 11:0...	9	---
t3_round2	Jul 06 2016 05:1...	18	---
t3_try2	Jun 30 2016 01:0...	9	---
t3test	Jun 28 2016 12:2...	19	---

LigandFit
Fitting of ligands into electron density

eLBOW
electronic Ligand Building and Optimization Workbench: ligand restraints generation and optimization

eLBOW (advanced interface)
more extensive eLBOW GUI for experts

REEL
Ligand restraints viewer and editor

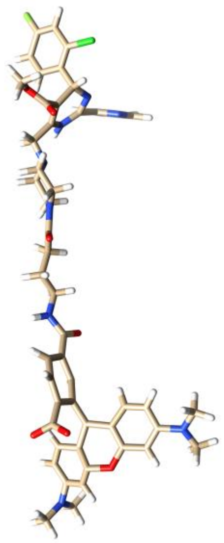
Ligand identification
Automated ligand search using database of 200 most frequent ligands, or user input

Guided ligand replacement
Ligand fitting based on an existing protein-ligand complex

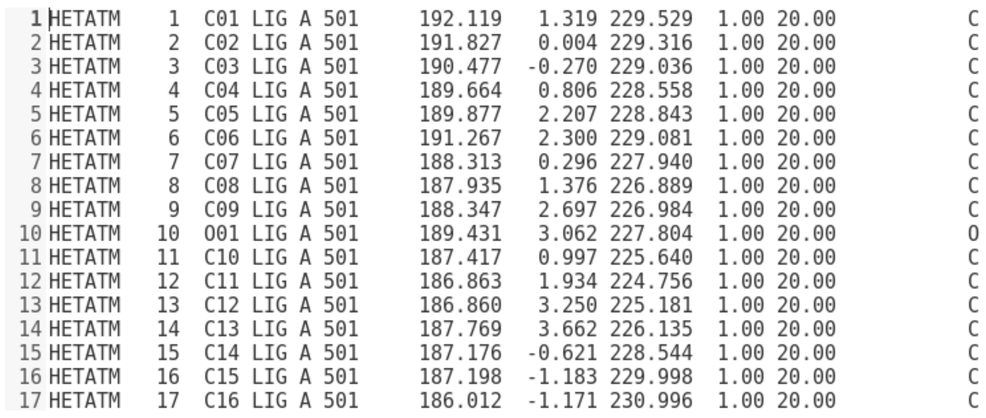
Current directory: /home/user/Desktop/flexfittin_rounds2/MethodTesting

PHENIX version 1.10.1-2155-000

Project: t3_MethodTesting



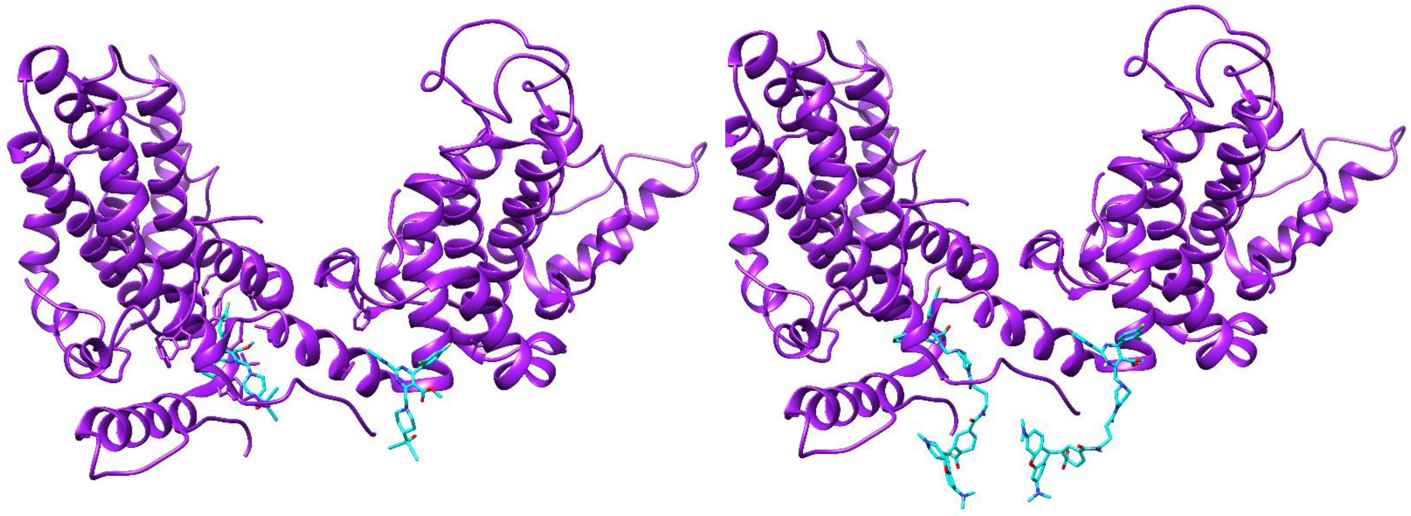
We will now run Phenix for the first time, I find it is helpful to always run Phenix from inside the top level directory, in this case the directory called “Protocol/” which contains StartingFiles, Refine01, Refine02...etc. After Phenix is started, select Ligands>eLBOW to start the eLBOW dialog. Click through the options, selecting “PDB file” as a source, and “simple optimization” as the method. Then browse to the ligand_haptam.pdb file as input, and also set the output directory to your LigandPrep directory. When this runs it will produce a folder called eLBOW_## where ## is the number of times you have run eLBOW. Inside the eLBOW directory you can open the elbow.ligand_haptam.pdb file to verify that you like the structure. In my case, I ran eLBOW several times before I got a structure I thought looked appropriate. Also inside the eLBOW directory is a .cif file. This is the parameter file that we will give to Coot/Phenix everytime we run them, so they know how to deal with the ligand. It is essential that the AtomName parameter (the third column of data in a PDB file) for each atom in your elbow.ligand_haptam.pdb and ligand.cif matches, and also that they do not conflict with any AtomNames currently used for proteins. Here is a sample of what my AtomNames look like.



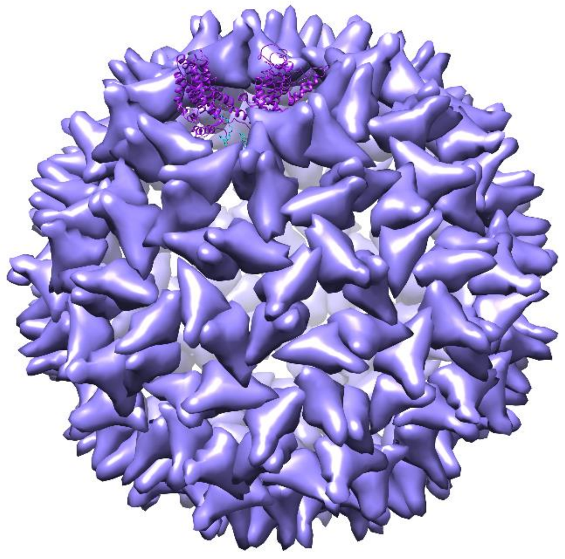
```
HETATM    1  C01 LIG A 501     192.119   1.319 229.529  1.00 20.00           C
HETATM    2  C02 LIG A 501     191.827   0.004 229.316  1.00 20.00           C
HETATM    3  C03 LIG A 501     190.477  -0.270 229.036  1.00 20.00           C
HETATM    4  C04 LIG A 501     189.664   0.806 228.558  1.00 20.00           C
HETATM    5  C05 LIG A 501     189.877   2.207 228.843  1.00 20.00           C
HETATM    6  C06 LIG A 501     191.267   2.300 229.081  1.00 20.00           C
HETATM    7  C07 LIG A 501     188.313   0.296 227.940  1.00 20.00           C
HETATM    8  C08 LIG A 501     187.935   1.376 226.889  1.00 20.00           C
HETATM    9  C09 LIG A 501     188.347   2.697 226.984  1.00 20.00           C
HETATM   10  O01 LIG A 501     189.431   3.062 227.804  1.00 20.00           O
HETATM   11  C10 LIG A 501     187.417   0.997 225.640  1.00 20.00           C
HETATM   12  C11 LIG A 501     186.863   1.934 224.756  1.00 20.00           C
HETATM   13  C12 LIG A 501     186.860   3.250 225.181  1.00 20.00           C
HETATM   14  C13 LIG A 501     187.769   3.662 226.135  1.00 20.00           C
HETATM   15  C14 LIG A 501     187.176  -0.621 228.544  1.00 20.00           C
HETATM   16  C15 LIG A 501     187.198  -1.183 229.998  1.00 20.00           C
HETATM   17  C16 LIG A 501     186.012  -1.171 230.996  1.00 20.00           C
```

Building Initial Model

In the StartingFiles directory, the 5d7y.pdb is the HBV T=4 crystal structure in complex with HAP18. This is a good starting point because our ligand (HATPTAM) is closely related, so it should have similar structure. Using Chimera, we replace each of the HAP18 ligands manually with a copy of the ligand_haptam.pdb molecule, as shown below.

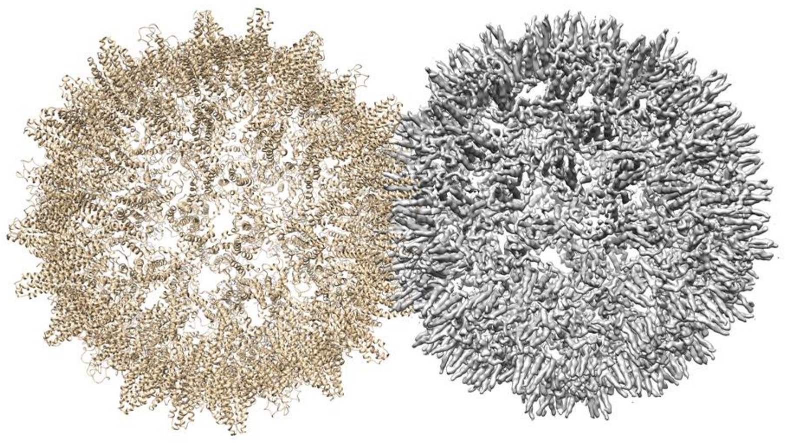


We can then apply the biomolecule symmetry operators found in the 5d7y.pdb using the “biological unit” feature in Chimera, to create an icosahedral capsid with the new ligand.

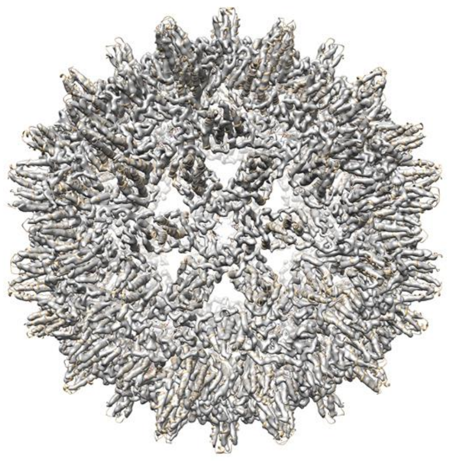


***Note that this requires you to have the symmetry operators already, in this case from a similar pdb file.*

Save this entire capsid as a single pdb using Chimera. We will call it t4_capsid_initial.pdb, then close and reopen it in Chimera to verify that we saved what we want. We can then also open our density map, and see that the capsid has a different origin than the map, causing the capsid to be out of density. This is bad, so we move the capsid into the map density using Chimera, being careful not to move the map at all. This way, if we want to generate new maps in the future, our refined model should fit inside them without requiring any movement. Save the model **relative to the map** once you have it rigidly fit into density. I called this t4_capsid_indensity.pdb

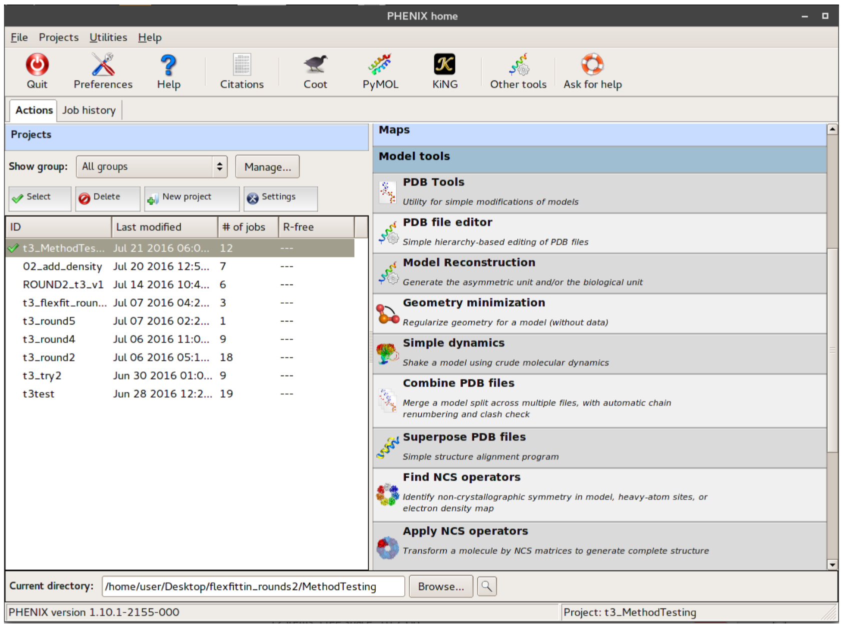


Move the **model** into the **map…..**save model relative to map.



NCSPrep

Here we will use the capsid we just created as a source for the NCS symmetry. Phenix has a function called “FindNCS” where it takes a symmetric model and automatically determines the symmetry operators, and generates a .ncs_spec file. You can then later use this .ncs_spec file and the Phenix function “ApplyNCS” to generate a symmetric structure from an asymmetric unit.



ID	Last modified	# of jobs	R-free
t3_MethodTes...	Jul 21 2016 06:0...	12	---
02_add_density	Jul 20 2016 12:5...	7	---
ROUND2_t3_v1	Jul 14 2016 10:4...	6	---
t3_flexfit_roun...	Jul 07 2016 04:2...	3	---
t3_round5	Jul 07 2016 02:2...	1	---
t3_round4	Jul 06 2016 11:0...	9	---
t3_round2	Jul 06 2016 05:1...	18	---
t3_try2	Jun 30 2016 01:0...	9	---
t3test	Jun 28 2016 12:2...	19	---

PDB Tools Utility for simple modifications of models

PDB file editor Simple hierarchy-based editing of PDB files

Model Reconstruction Generate the asymmetric unit and/or the biological unit

Geometry minimization Regularize geometry for a model (without data)

Simple dynamics Shake a model using crude molecular dynamics

Combine PDB files Merge a model split across multiple files, with automatic chain renumbering and clash check

Superpose PDB files Simple structure alignment program

Find NCS operators Identify non-crystallographic symmetry in model, heavy-atom sites, or electron density map

Apply NCS operators Transform a molecule by NCS matrices to generate complete structure

Current directory: /home/user/Desktop/flexfittin_rounds2/MethodTesting

PHENIX version 1.10.1-2155-000   Project: t3_MethodTesting

Using “FindNCS”, give it the location of your t4_capsid_indensity.pdb file, and set the output directory to your NCSPrep directory. If you then click run, Phenix will choke on the pdb and crash. This is because Chimera saves each ASU as separate model using MODEL/ENDMDL tags, and Phenix does not support this. This is the first instance where we will make use of the conversion script, 'pdbRefactor'. For precise usage of the script and a description of the typical PDB formatting problems encountered, refer to the Appendix. In short, pdbRefactor takes as arguments 1)pdb file, 2) an output type, and 3) an optional extract argument for extracting subsections. In this case, we don't want to extract anything, so we type:

pdbRefactor t4_capsid_indensity.pdb type=phenix

After running the script on the file, you will have a t4_capsid_indensity_PHENIX.pdb file. Now you should be able to supply this file to the “FindNCS” function, and if it is successful, you will have a FindNCS folder inside your NCSPrep folder. Inside this FindNCS folder is the .ncs_spec file, which is the goal of this step. A brief summary of the desired directory structure is shown below

```
|- - - - - >TopLevelDirectory
            |- - - - - - - - - - ->LigandPrep
            |- - - - - - - - - - ->StartingFiles
            | - - - - - - - - - - >NCSPrep
                      |- - - - - - - - - - - - >FindNCS_1
                      |- - - - - - - - - - - - >*.ncs_spec
                      | - - - - - - - - - - - ->*.phil
                      |- - - - - - - - - - - - >*.resolve
            | - - - - - - - - - - >Refine01
            | - - - - - - - - - - >etc….
```

It is useful to pause and make sure that the .ncs_spec file that is generated can be used to create a new capsid. To do this, we first need an ASU from the t4_capsid_indensity_PHENIX.pdb file. You could just copy and paste the top ASU from the file into a new file, or you can use the script. This will take any capsid PDB and generate a new pdb file that is only the ASU.

pdbRefactor t4_capsid_indensity_PHENIX.pdb type=phenix extract=asu

The resulting file would be called 't4_capsid_indensity_PHENIX_extractedASU_PHENIX.pdb'. If you try to open it in Chimera, it will probably work, but look weird and be numbered in a frustrating way. If our goal was an ASU to open in Chimera for visualization, then we should have used 'type=chimera'. However, in this case we are about to run Phenix, so what is shown above was correct.

Open Phenix and click through the "ApplyNCS" dialog, being careful to supply the correct input files, .ncs_spec file, and output directory. In this case, we want to use the extracted ASU as input (which should be in the StartingFiles directory), the .ncs_spec file we just generated, and set NCSPrep as the output directory. Once this runs, there will be a new directory in the NCSPrep folder, called ApplyNCS_1 (or some other number if you run this multiple times). Inside that ApplyNCS directory, is what should be a full capsid pdb. However, it is in Phenix format, so if you open it in Chimera it will look fragmented, or it will not open at all. First use the script to do something like:

pdbRefactor *titlehere*_apply_ncs_1.pdb type=chimera

Then, open the resulting *titlehere*_apply_ncs_1_CHIMERA.pdb file in Chimera. You should have what looks like a capsid. What is now in place is a mechanism for taking any ASU you generate, and forming it into a capsid. You will continue to use the same .ncs_spec file to do this throughout refinement.

******If at any point during this process you have a problem, try opening your file in Pymol (the most tolerant program). Then you can at least look at what you have, which helps with troubleshooting*

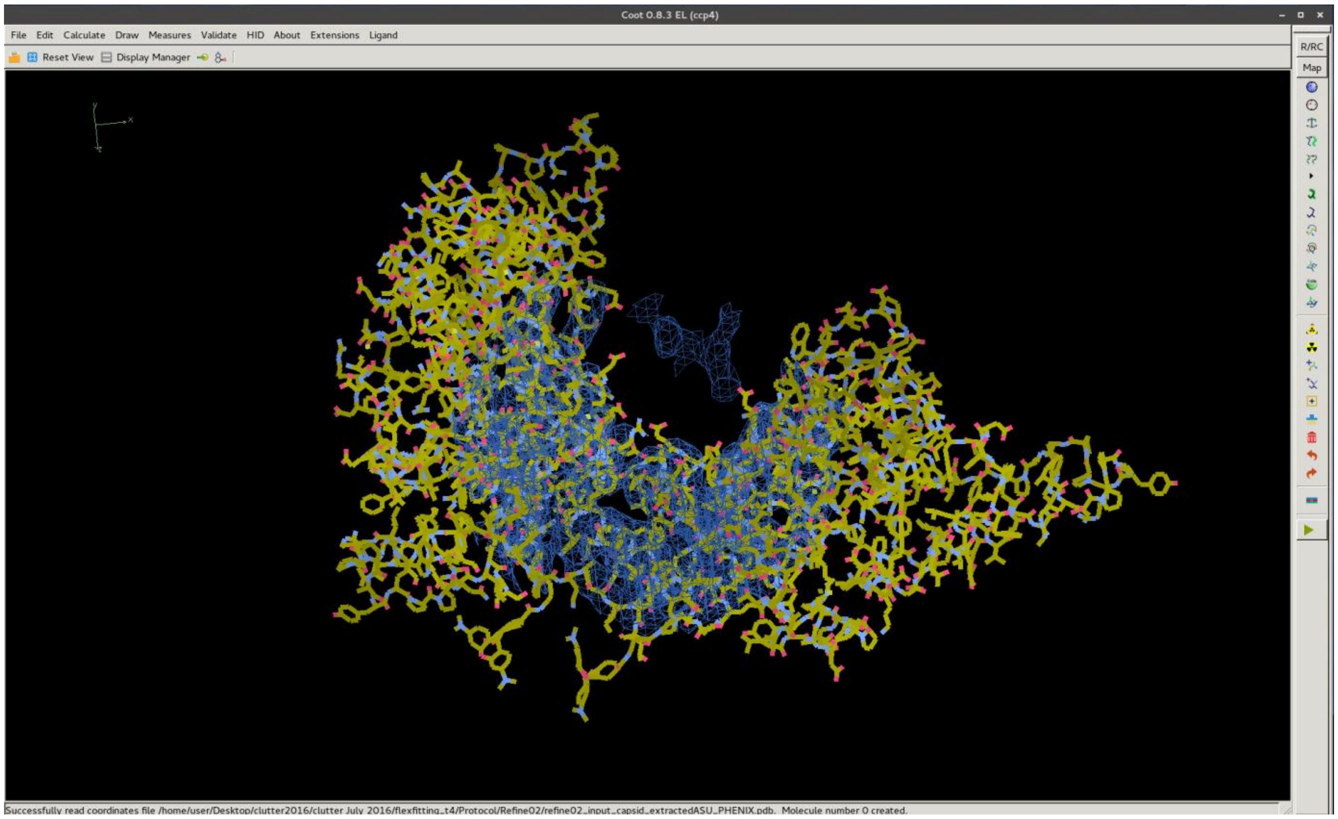
Refinement Iteration 01

Now we are going to move to the Refine01 directory. We need to copy two things into this directory, 1) your density map, 2) the ASU pdb file you generated in the previous step. You will probably also want to rename the pdb file, because since all the programs used here append things to the end of the filename, it gets very long very fast. I suggest something like 'refine01_input_asu.pdb' or something like that. Something that identifies it was the starting point for refinement.

Coot

Now we want to use Coot to manually refine our ASU model into density. Coot will open Phenix formatted files, but the residues will be numbered 0-N over the whole asymmetric unit which is frustrating. Also, it won't recognize the terminal regions between 0 and N, so you will not be able to use the “Add Terminal Residue” functionality. Coot will not open Chimera files as they are created by default by the pdbRefactor script, but if you use a text editor to remove all references to MODEL or ENDMDL lines, then it should work with Coot. Alternatively, you can use 'pdbRefactor type=coot', which just creates a Chimera PDB, and then removes the MODEL/ENDMDL tags for you.

Run Coot from the Refine01 directory, and **File>Open** the **map**, the **ASU pdb** file, and navigate to the **.cif file**, which should be one directory up, and then somewhere inside the LigandPrep directory. Once you are inside Coot, you more or less are using Coot the normal way. Just keep in mind that Coot has a capsid density map, but only an ASU model, so be careful of the model moving into territory that will be occupied by symmetric neighbors in a capsid. The NCS symmetry will not be taken into account until the Phenix refinement. Once you are done in Coot, save the coordinates, and continue to the next section.


Coot 0.8.3 EL (ccp4)
File Edit Calculate Draw Measures Validate HID About Extensions Ligand
Reset View
Display Manager
R/RC
Map

```
Successfully read coordinates file /home/user/Desktop/clutter2016/clutter July 2016/flexfitting_t4/Protocol/Refine02/refine02_input_capsid_extractedASU_PHENIX.pdb.  Molecule number 0 created.
```

***If the .cif file fails to load, it is probably because the AtomName fields in the cif file no longer match the AtomName field in the pdb file. You will need to fix the pdb.*

PHENIX

When you saved you coordinates from Coot, it probably appended something like coot-0.pdb to the end. This is the file you want to give to Phenix. We have to give to Phenix a full capsid that incorporates the changes just made in Coot. To do this we will use the Phenix ApplyNCS function again. Make the input file the ASU pdb you just made in Coot, the .ncs_spec file is in the NCSPrep folder, and the output directory is Refine01.

After this has run, there should be a new ApplyNCS folder inside of the Refine01 directory. Inside there is the capsid pdb you just created. First, use pdbRefactor to convert it to Chimera, and open it in Chimera, possibly with your map, to make sure it looks ok. Next, use pdbRefactor again to convert it to Phenix type, this PHENIX.pdb file will be our input into the Phenix real space refinement.

Change directory in the terminal so you are inside Refine01. The three main inputs you need are the capsid pdb, the density map, and the .cif file so Phenix knows how to handle your ligand. Assuming the filenames are:

apply_ncs_PHENIX.pdb
densitymap.mrc
../LigandPrep/eLBOW_1/elbow.molecule_pdb.001.cif

Then run Phenix by entering the following at the command line:

```
phenix.real_space_refine apply_ncs_PHENIX.pdb  densitymap.mrc
elbow.haptam_molecule_pdb.001.cif ncs_constraints=True resolution=5.0
```

The above command is all one line, it has the three input files and two additional arguments; 'ncs_constraints=True' tells Phenix to detect symmetry related atoms, and 'resolution=5.0' is the resolution at which the model and map are compared. For this value I have been using my FSC value from the reconstruction, or in other words, the highest resolution possible for a given map.

When you run this line, Phenix will read in the files, detect the NCS, and then by default do 5 cycles of refinement. This will take several hours. If it is successful, it will write several files to the current directory, the most important of which will be '*_PHENIX_real_space_refined.pdb'. This is a full capsid file, and it will be the file you carry over to the Refine02 directory for another iteration of Coot/Phenix.

*****Phenix reading the pdb file was by far the most common point of failure when I was first doing this, and was the main motivation for creating pdbRefactor. If it fails, take a look at the Appendix for some common errors.*

Refinement Iteration 02

Copy your *_real_space_refined.pdb from phenix and your density map into the Refine02 directory. From here everything is exactly the same as in iteration 1.

1. Use pdbRefactor to extract an ASU
2. Use coot to manually refine the ASU
3. Build a capsid using ApplyNCS
4. Run Phenix real space refine
5. Copy *_real_space_refined.pdb to a new refinement directory

Iterate as many times as you want.

Appendix I

PDB Requirements

In the event that pdbRefactor does not work, or you don't want to use it, or you are just curious… I have listed here some "requirements" of a pdb to be opened in each piece of software. These requirements were found by trial and error, and it may be that not all of them are actually required. However, this is what I found to work.

Phenix

1. All atoms must have a unique AtomNumber. This necessitates using hexadecimals for an HBV capsid.
2. Residues must be numbered 1-N where N is the total number of residues in an asymmetric unit. So each residue number should only occur 60 times in an icosahedral capsid, regardless of the number of monomers in the ASU.
3. Each asymmetric unit must have a unique Chain ID, so there should be 60 chain IDs total
4. There are no MODEL/ENDMDL tags, and no TER entries.

Chimera

1. Hexadecimals are not supported. Instead, each ASU must have atoms numbered 1-N where N is the number of atoms in the ASU. Each ASU is then separated by a MODEL/ENDMDL tag.
2. In order to easily select all of one kind monomer, each monomer gets a unique Chain ID. This means there are only a couple of unique Chain IDs, that are repeated in each ASU.
3. The residues are numbered 1-N where N is the number of residues in a **monomer**. This makes visualization much easier because the residue numbers are what you would expect based on the sequence.
4. TER entries are required at the end of each **monomer** to prevent chimera drawing a bond between N and C termini.

Coot

1. Coot can open Phenix-type files without modification, but the residue numbering will be confusing to use in Coot. Coot can also open Chimera-type files, but the MODEL/ENDMDL tags must be removed first.

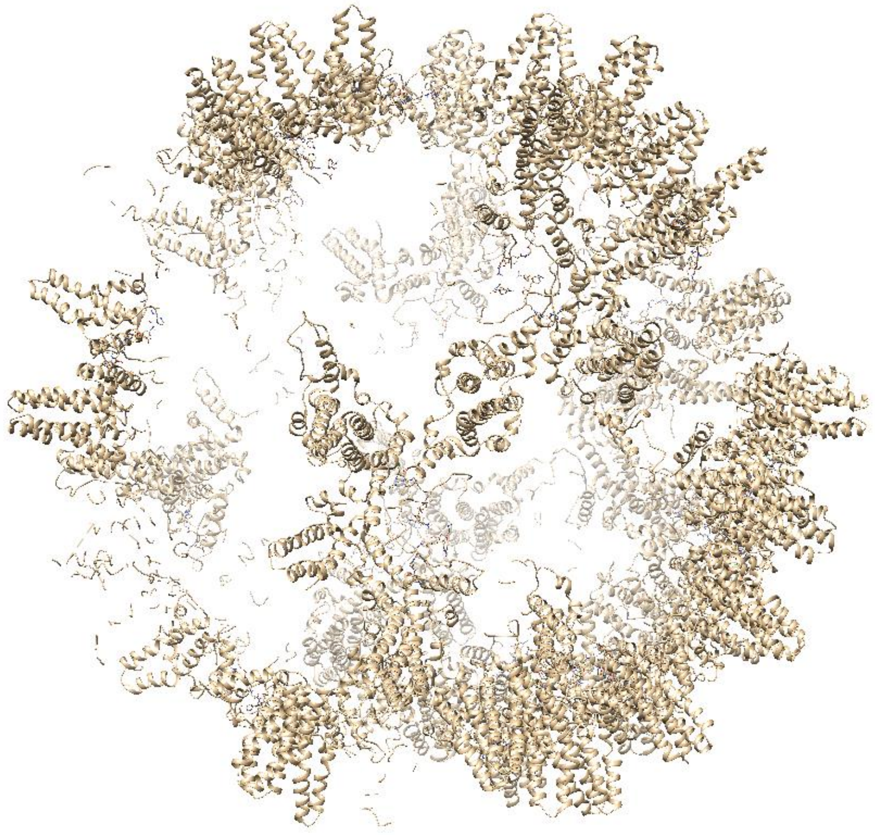
Appendix II

Troubleshooting

Here are some commonly encountered problems:

Problem 1:

Chimera is not opening icosahedral pdbs correctly

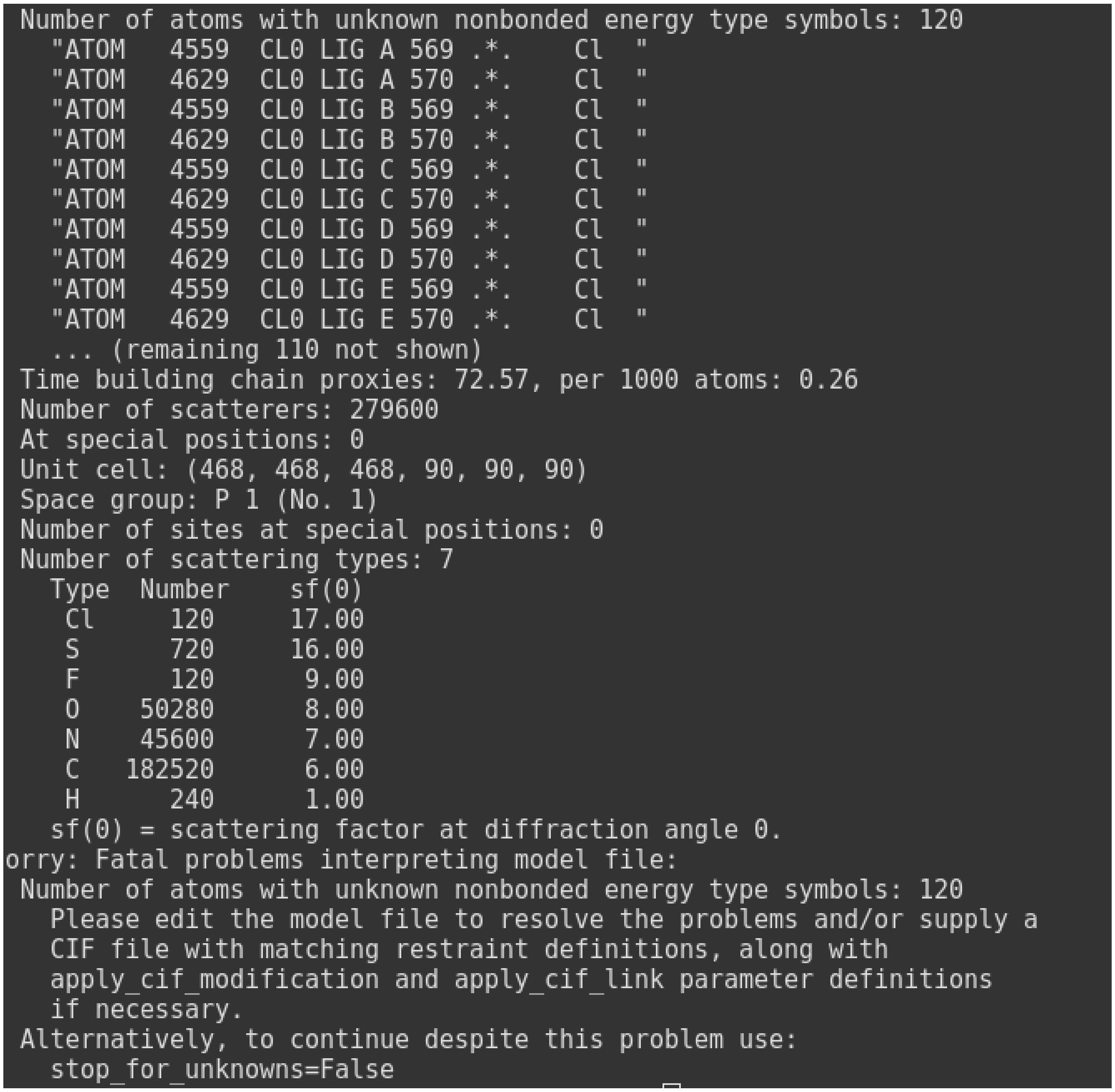


Cause: Your pdb is not using MODEL/ENDMDL tags correctly to separate ASUs. When Chimera encounters duplicate atoms, it just doesn't show them. If you opened this in Pymol, it would probably look alright. The only way to resolve the conflicts are to 1) Use the MODEL/ENDMDL tags, or 2) Use hex codes for AtomNumber. Since you want to look at your model in Chimera, your only option is to use the MODEL tags.

****If you try to open a PDB with hex codes in Chimera, it will freeze.*

Problem 2

Phenix fails reading the cif file

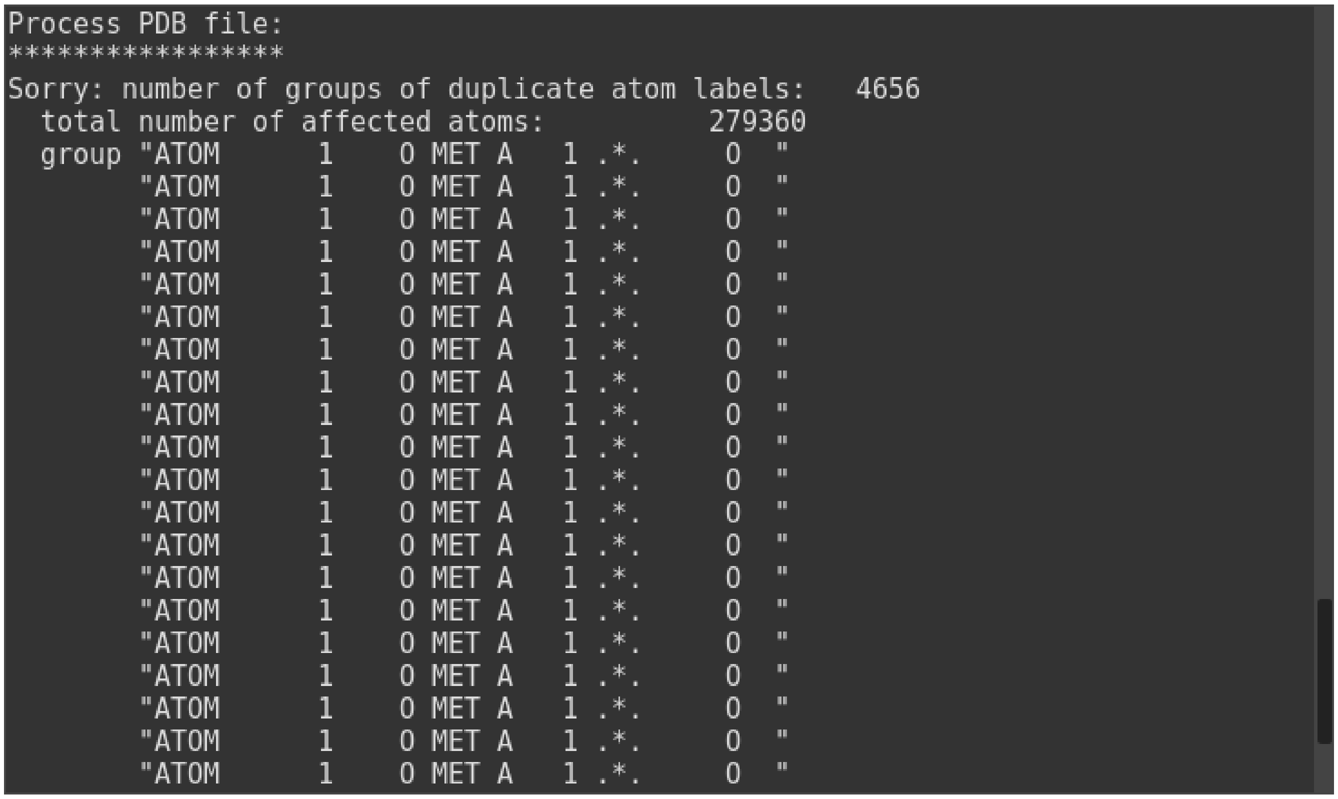


```
 Number of atoms with unknown nonbonded energy type symbols: 120
   "ATOM   4559  CL0 LIG A 569 .*.    Cl  "
   "ATOM   4629  CL0 LIG A 570 .*.    Cl  "
   "ATOM   4559  CL0 LIG B 569 .*.    Cl  "
   "ATOM   4629  CL0 LIG B 570 .*.    Cl  "
   "ATOM   4559  CL0 LIG C 569 .*.    Cl  "
   "ATOM   4629  CL0 LIG C 570 .*.    Cl  "
   "ATOM   4559  CL0 LIG D 569 .*.    Cl  "
   "ATOM   4629  CL0 LIG D 570 .*.    Cl  "
   "ATOM   4559  CL0 LIG E 569 .*.    Cl  "
   "ATOM   4629  CL0 LIG E 570 .*.    Cl  "
   ... (remaining 110 not shown)
 Time building chain proxies: 72.57, per 1000 atoms: 0.26
 Number of scatterers: 279600
 At special positions: 0
 Unit cell: (468, 468, 468, 90, 90, 90)
 Space group: P 1 (No. 1)
 Number of sites at special positions: 0
 Number of scattering types: 7
   Type  Number    sf(0)
    Cl     120     17.00
    S      720     16.00
    F      120      9.00
    O    50280      8.00
    N    45600      7.00
    C   182520      6.00
    H      240      1.00
   sf(0) = scattering factor at diffraction angle 0.
orry: Fatal problems interpreting model file:
 Number of atoms with unknown nonbonded energy type symbols: 120
   Please edit the model file to resolve the problems and/or supply a
   CIF file with matching restraint definitions, along with
   apply_cif_modification and apply_cif_link parameter definitions
   if necessary.
 Alternatively, to continue despite this problem use:
   stop_for_unknowns=False
```

Cause: The AtomName fields in the PDB do not match those in the CIF file. In this particular case, the CIF file has ‘CL01’ for the first chlorine atom, but the PDB file has ‘CL0’, and the “1” got truncated somehow. The PDB always has to match the cif file.

Problem 3

Phenix fails reading the PDB

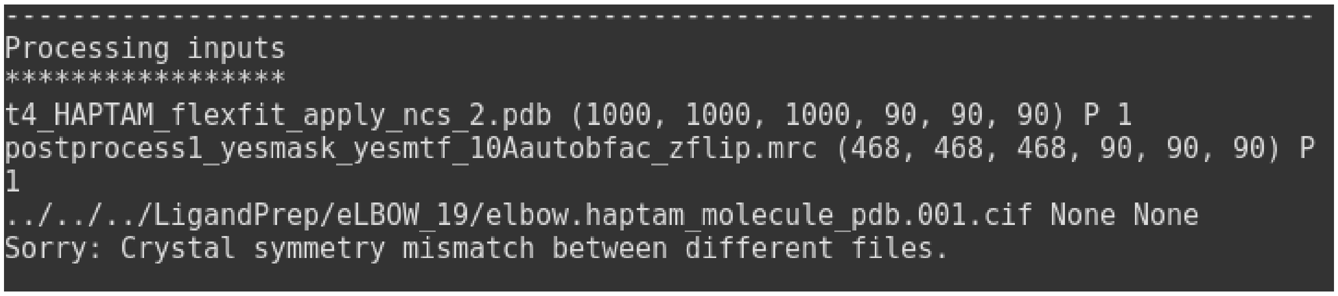


```
Process PDB file:
*****************
Sorry: number of groups of duplicate atom labels:   4656
  total number of affected atoms:          279360
  group "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
        "ATOM      1    O MET A   1 .*.     O  "
```

Cause: There are atoms which Phenix thinks are identical, but are in reality different but symmetry related. This is what happens if you try to run Phenix on a Chimera-type file, for instance. To fix these types of problems, I started following the requirements detailed in Appendix I. If you run pdbRefactor with type=phenix (and it runs successfully), you should not have this problem.

Problem 4

Phenix error: Crystal Symmetry mismatch

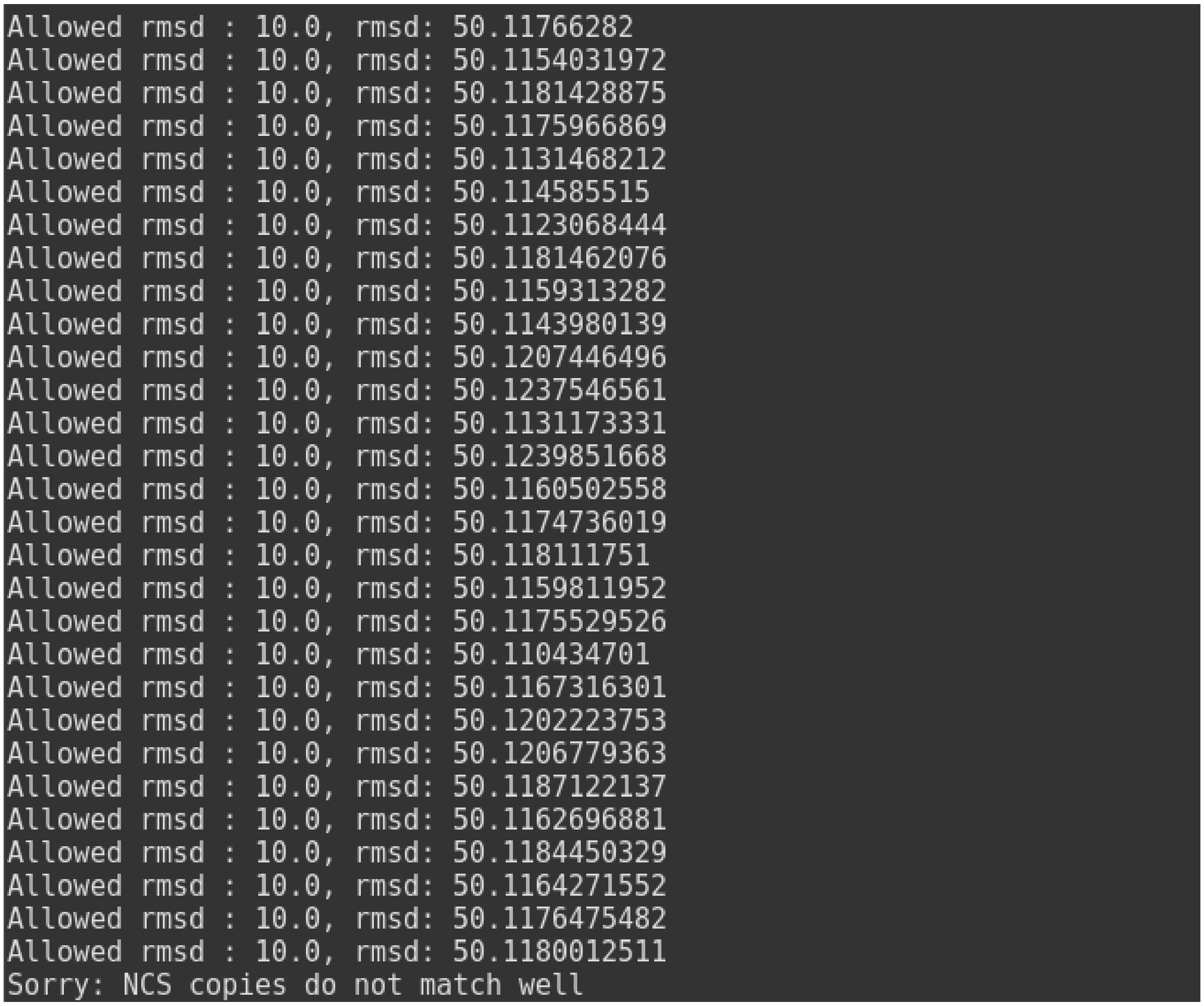


```
-------------------------------------------------------------------------------
Processing inputs
****************
t4_HAPTAM_flexfit_apply_ncs_2.pdb (1000, 1000, 1000, 90, 90, 90) P 1
postprocess1_yesmask_yesmtf_10Aautobfac_zflip.mrc (468, 468, 468, 90, 90, 90) P
1
../../../LigandPrep/eLBOW_19/elbow.haptam_molecule_pdb.001.cif None None
Sorry: Crystal symmetry mismatch between different files.
```

Cause: This is an easy one. Open the pdb and remove the line that specifies crystal symmetry, since that is meaningless here. Also, since pdbRefactor does not ever write metadata like this, you could just use the script to refactor to the Phenix type.

Problem 5

Phenix error: NCS copies do not match well



```
Allowed rmsd : 10.0, rmsd: 50.11766282
Allowed rmsd : 10.0, rmsd: 50.1154031972
Allowed rmsd : 10.0, rmsd: 50.1181428875
Allowed rmsd : 10.0, rmsd: 50.1175966869
Allowed rmsd : 10.0, rmsd: 50.1131468212
Allowed rmsd : 10.0, rmsd: 50.114585515
Allowed rmsd : 10.0, rmsd: 50.1123068444
Allowed rmsd : 10.0, rmsd: 50.1181462076
Allowed rmsd : 10.0, rmsd: 50.1159313282
Allowed rmsd : 10.0, rmsd: 50.1143980139
Allowed rmsd : 10.0, rmsd: 50.1207446496
Allowed rmsd : 10.0, rmsd: 50.1237546561
Allowed rmsd : 10.0, rmsd: 50.1131173331
Allowed rmsd : 10.0, rmsd: 50.1239851668
Allowed rmsd : 10.0, rmsd: 50.1160502558
Allowed rmsd : 10.0, rmsd: 50.1174736019
Allowed rmsd : 10.0, rmsd: 50.118111751
Allowed rmsd : 10.0, rmsd: 50.1159811952
Allowed rmsd : 10.0, rmsd: 50.1175529526
Allowed rmsd : 10.0, rmsd: 50.110434701
Allowed rmsd : 10.0, rmsd: 50.1167316301
Allowed rmsd : 10.0, rmsd: 50.1202223753
Allowed rmsd : 10.0, rmsd: 50.1206779363
Allowed rmsd : 10.0, rmsd: 50.1187122137
Allowed rmsd : 10.0, rmsd: 50.1162696881
Allowed rmsd : 10.0, rmsd: 50.1184450329
Allowed rmsd : 10.0, rmsd: 50.1164271552
Allowed rmsd : 10.0, rmsd: 50.1176475482
Allowed rmsd : 10.0, rmsd: 50.1180012511
Sorry: NCS copies do not match well
```

Cause: You are probably trying to use the raw output of “ApplyNCS” as input for Phenix refine. This does not work, because the residues are not numbered by ASU. Once again, running ‘pdbRefactor type=phenix’ should fix this problem.

Appendix III

pdbRefactor

To aid in quickly and consistently reformatting pdb files, I wrote a small Python module called PDBModule. This provides general classes like PDBFile, Atom and Residue, as well as methods for doing basic operations like grouping residues, renumbering atoms, calculating center of mass, etc. The long term goal of this module is to be generally useful for working with any pdb file.

The script pdbRefactor makes use of PDBModule, and is more specialized towards dealing with the exact type of problems encountered when doing Coot/Phenix refinement. It is meant to be run from the command line.

Installation

This is a quick and dirty installation for Linux. At some point I will try to package PDBModule as a proper Python module. For now:

1. Put the PDBModule folder anywhere on your computer
2. Edit the pdbRefactor.py file, and change the sys.path.append() field so that it matches the location of the PDBModule folder
3. Create a sym link from pdbRefactor.py to somewhere in your path, ie sudo ln -s ~/PDBModule/pdbRefactor.py /usr/bin/pdbRefactor

Here is the output of ‘pdbRefactor –help

```
Usage: pdbRefactor input.pdb type=TYPE  extract=SECTION
Where 'TYPE' is the output type 'phenix' or 'chimera' or 'coot'
And 'SECTION' is optional, can be 'asu' or 'patch'

Also required is a parameter file located either in the working directory, or one or two directories up.
parameter file should be exactly 'pdbparams.txt' and should include these parameters:
```

monomerlen = integer   # The length of protein residues in a monomer.
asusize = integer   # The number of monomers in an asu. Note that for this program, all monomers MUST be an identical length
numligands = integer   # The number of non-protein ligands per asymmetric unit. (specified with LIG as the residue ID)
endsequence = string   # Several letters of protein sequence to identify the end of the monomer chain. ie: PNAPILST
patchsize = integer   # The number of nearby asymmetric units to extract when using 'extract=patch'. You will probably need to adjust this. For HBV T=4, patchsize=4 extracts a 6-fold.

These parameters are used to divide up a pdb into symmetric sections, assuming the order they occur in the pdb is Asu1, Asu2, Asu3, etc.
Different orderings will cause this to fail. For example, putting all the ligands together at the end of the file. To generate a sample parameter file run pdbRefactor -initialize in the directory you want the file.