Rosetta Density-fitting Tutorial

Frank DiMaio, January 2010

This tutorial is intended to guide users to several different ways Rosetta may be used to solve various problems in structure fitting into low-resolution density data. It is not intended to replace the user's guide, available at http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/ or in the doc/ folder of the installed Rosetta application.

The tutorial is split up into several *scenarios* in which Rosetta may be used. In each scenario, one or more Rosetta application is used to improve a structure's fit to density. For each scenario, the inputs and outputs are briefly provided. Each command-line is highlighted in a shaded box. Following, a brief explanation of each of the flags, as well as other potentially relevant options is given. Important options (or at least options of which to take note) are boldfaced.

Also, this tutorial focuses on refinement of structures when density data is available, however, the vast majority of command lines given are perfectly valid when no density data is available. Generally, running each of these commands while omitting the flags beginning with '`-edensity::*`' will run the same protocol, using Rosetta's energy function *without* the fit-to-density terms.

Most of the applications described in this tutorial work as given in the current Rosetta release, version 3.1. However, there are several that make use of applications unavailable in the current version. These applications are pointed out in the document; they will be available in the next release of Rosetta, though option names are not guaranteed to stay the same.

Scenario 1: An introduction to some basics in Rosetta-density

In this scenario, we use the modeling of a small conformational change to introduce some basic Rosetta functionality. We refine the structure of 2JEL (chain P) into a synthesized density map of 1POH (chain A).

Generally, the first step in refining a structure in Rosetta is to *idealize* the structure; that is, to give the structure ideal bond lengths and bond angles. Idealization is done using the following command (see *scenario1_rosetta_basics/run1_idealize.sh*), which returns an ideal structure with minimal C-alpha deviation from the starting model.

```
bin/idealize.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in::file::s 2JEL_P.pdb \
      -overwrite
```

At the end of the run, idealize will print out the overall RMS from the initial model. This command will output a structure *2JEL_P_0001.pdb*. Verify that this model looks reasonable, and then rename it to *2JEL_P_idl.pdb.* The remainder of this scenario will work on this idealized structure.

The next step is to minimize the structure with respect to Rosetta's energy constrained to the experimental density data provided in the density map *1POH.5A.mrc*. This simplest way to do this is through the application *relax*. Relax makes relatively small backbone and sidechain torsional movements to find a nearby local minimum of Rosetta's energy function. A simple command line for doing this is shown below (see *scenario1_rosetta_basics/run2A_relax.sh*)

```
bin/relax.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in::file::s 2JEL_P_idl.pdb \
      -score:weights score13_env_hb \
      -relax::fast \
      -relax::fastrelax_repeats 1 \
      -relax::jump_move true \
      -edensity::mapfile 1POH.5A.mrc \
      -edensity::mapreso 5.0 \
      -edensity::grid_spacing 2.0 \
      -edensity::whole_structure_allatom_wt 0.1
```

Some flags of note are boldfaced above. First the flag `-relax::fast` runs a special fast mode of relax, where the backbone torsions are only minimized and not perturbed during the run. For fitting structures into density, this mode is highly recommended. Also, the flag `-relax::fastrelax_repeats` tells Rosetta for how long it should refine the structure. The default value is 8; for fitting structures into density, a smaller value (say 4) may be more reasonable (it is 1 above to show a relatively fast-running relax). Finally, the flag `-relax::jump_move true` tells relax it must take into account the rigid-body orientation of the whole molecule; *it must always be given when relaxing structures into density*.

Additionally, note the flags beginning with `-edensity::*`. These flags tell the relax application about the density map into which it is being fit. The name of the mapfile (in CCP4 or MRC format), the resolution of the map, the grid sampling of the map (*which should never be more than half the*

resolution), and the weights on the various fit-to-density scoring functions. These same flags are reused for many different protocols in addition to relax.

Rosetta features three different fit-to-density scoring functions:

elec_dens_whole_structure_ca
Recommended for very low resolution maps (6A+); Uses the correlation of the whole structure density versus the experimental data; structure density only uses C-alphas.

elec_dens_whole_structure_allatom
Recommended for low-medium resolution (3-8A); Uses the correlation of the whole structure density versus the experimental data; structure density only uses all heavy atoms.

elec_dens_window
Recommended for medium-high resolution (<4A); Uses the sum of correlations of a sliding window of residues versus the experimental data; structure density only uses all heavy atoms. *Unlike the other two scoring functions, this uses the fit-to-density score when choosing sidechain rotamers*.

These energy terms may be placed in a weights file like any other scoring term in Rosetta. For convenience, the following flags may also control the three scoring functions, respectively:

```
-edensity:whole_structure_ca_wt <wt>
-edensity:whole_structure_allatom_wt <wt>
-edensity:sliding_window_wt <wt>
```

If the sliding window scoring function is used, the additional flag `-edensity:sliding_window` *n* should also be provided, which gives the width (in residues) of the widow to use. This should always be an odd number. The command line in *scenario1_rosetta_basics/run2B_relax_slwin.sh* shows relaxing a structure using the sliding window weight.

Another useful flag when running relax is `-relax::constrain_relax_to_start_coords`. If relax is given a structure with very poor energy (for example, a homology model with serious steric clashes), the output model may move significantly, often in an undesirable way. This flag will place constraints on the starting C-alpha positions. See the command *scenario1_rosetta_basics/run2C_relax_cst.sh* for an example.

Finally, one may wish to simply score (or rescore) some set of models using Rosetta. This is simply accomplished by using the *score* application. A sample command line to rescore the relaxed structures is given in *scenario1_rosetta_basics/run3_rescore.sh*. Like the relax command line, it uses the various `-edensity` flags to provide Rosetta with experimental density information.

```
bin/score.linuxgccrelease \
 -database ~/rosetta_database/ \
 -in::file::s 2JEL*.pdb 1POH_A.pdb \
 -in::file::native 1POH_A.pdb \
 -score:weights score13_env_hb \
 -ignore_unrecognized_res \
 -edensity::mapfile 1POH.5A.mrc \
 -edensity::mapreso 5.0 \
 -edensity::grid_spacing 2.0 \
 -edensity::whole_structure_allatom_wt 0.1
```

This command line outputs a score file *default.sc* that gives – for each structure specified with `-in::file::s` – the score with respect to each term in Rosetta's energy function. The meaning of some of the more important terms is shown below.

fa_atr, fa_rep
: Lennard-Jones attractive, repulsive energies

fa_sol
: Lazaridis-Karplus solvation energy

pro_close
: proline ring closure energy

fa_pair
: statistics based pair term; models things like salt bridges

hbond_*
: hydrogen bond energy terms

dslf_*
: disulfide bond energy terms

rama, omega
: Ramachandran preferences**,** omega angle preferences

fa_dun
: internal energy of sidechain rotamers as derived from Dunbrack's statistics.

p_aa_pp
: probability of observing a particular amino acid given phi/psi angles

ref
: reference energy for each amino acid

Scenario 2: Building a model from a close homologue

In this scenario, we refine a near homology model into density. We refine a model of 1ONC that has been threaded onto template 1OJ8 into synthetic density data corresponding to 1ONC. The file *1ONC_threaded.pdb* contains the backbone of coordinates of 1OJ8 with the corresponding amino-acid identities from 1ONC. It is not necessary for this model to contain any sidechain atoms.

Since the threaded model covers the entire sequence of 1ONC (that is, we do not have to rebuild any of the backbone atoms) we can simply use the relax application. The command line is very similar to that of Scenario 1, and is given in *scenario2_close_homology/run1_relax.sh*:

```
bin/relax.linuxgccrelease \
      -database /opt/rosetta-3.1/rosetta_database/ \
      -in::file::s 1ONC_threaded.pdb \
      -score:weights score13_env_hb \
      -relax::fast \
      -relax::fastrelax_repeats 1 \
      -relax::jump_move true \
      -edensity::mapfile 1ONC.5A.mrc \
      -edensity::mapreso 5.0 \
      -edensity::grid_spacing 2.0 \
      -edensity::whole_structure_allatom_wt 0.1
```

Additionally, using an input mode *not available in Rosetta 3.1*, we do not even have to provide the threaded model. Instead, we simply need to give Rosetta an alignment (as in *1onc_1oj8.fasta*):

```
1onc_.pdb 1
   DWLTFQKKHITNTRDVDCDNIMSTNLFHCKDKNTFIYSRPEPVKAICKGIIASKNVLTTSEFYLSDCNVTSRPCKYKLK
   KSTNKFCVTCENQAPVHFVGVGSC
1OJ8.pdb  2
   DWDTFQKKHLTDTKKVKCDVEMKKALFDCKKTNTFIFARPPRVQALCKNIKNNTNVLSRDVFYLPQCNRKKLPCHYRLD
   GSTNTICLTCMKELPIHFAGVGKC
```

Then (see *scenario2_close_homology/run2_relax_cm_inputs.sh*) we use the follow flags in lieu of `-in::file::s`:

```
      -in::file::extended_pose 1 \
      -in::file::fasta 1onc.fasta \
      -in::file::alignment 1onc_1oj8.fasta \
      -in::file::template_pdb 1OJ8.pdb \
```

Scenario 3: Using "relax" to model large conformational variation

In this scenario, we use relax to model a large conformational change guided by density. This shows the power of relax when combined with low-resolution experimental density. Additionally, we introduce some non-standard flags that may be useful for difficult density-fitting cases.

Chain L of 1QFK and 2PUQ represent different states of the same molecule, and show large conformational variation (8A C-alpha RMS). In this scenario, we refine each model into synthesized density from the other.

First, we fit 1QFK into the density of 2PUQ (using the command line from *scenario3_conformational_change/run1_flexfit_1qfk_to_2puq.sh*).

```
bin/relax.linuxgccrelease \
      -database /opt/rosetta-3.1/rosetta_database/ \
      -in::file::s 1QFK_L.pdb \
      -score:weights score13_env_hb \
      -relax::fast \
      -relax::fastrelax_repeats 1 \
      -relax::fastrelax_rampcycles 10 \
      -relax::jump_move true \
      -edensity::mapfile 2PUQ.5A.mrc \
      -edensity::mapreso 12.0 \
      -edensity::grid_spacing 6.0 \
      -edensity::ca_mask 20.0 \
      -edensity::whole_structure_ca_wt 0.1
```

Flags to note are indicated in boldface. The first of these flags `-relax::fastrelax_rampcycles`, controls the behavior of the relax application itself. By default, each cycle of fast-relax consists of 5 repack-minimization steps. In each step, a higher weight is given to the repulsive energy (fa_rep). By increasing this to 10, it gives the structure more time to make a larger conformational change to better fit the density before the increased repulsive weight restricts motion too much. Aside from density fitting, this increased number of ramp-cycles is also useful (especially combined with `-relax::constrain_relax_to_start_coords` from Scenario 1) when refining structures with serious clashes.

Several density flags are changed from the default. First to give a much smoother energy landscape, the density map is resampled on a 6A grid, and the resolution is computed to only 12A. To further smooth things out (as well as reduce the running time) the calculated density used for scoring the model only uses C-alpha positions (that is, we use `whole_structure_ca_wt` instead of `whole_structure_allatom_wt`). Finally, because of the large conformational change, the C-alpha's in the starting structure need to "see" the density of the alternate conformation. Thus, we increase the mask around each C-alpha to 20A with the flag `-edensity::ca_mask`. The default value is resolution-dependent but is generally much smaller. Were we to use `whole_structure_allatom_wt`, then we would need to specify an atom mask distance instead, with `-edensity::atom_mask`.

The result is a very accurate model of the conformation in 2PUQ, about 1A C-alpha RMSd. Fitting 2PUQ to the density of 1QFK (as in *scenario3_conformational_change/run1_flexfit_2puq_to_1qfk.sh*) yields similar accuracy. Further refinement using the full-resolution density data might yield additional improvement.

Scenario 4: Using "loopmodel" for selective rebuilding

In this scenario, we introduce the *loopmodel* application as a tool for modeling large – but localized – backbone conformational changes. We also introduce the *loops_from_density* application as a way of identifying candidate regions to rebuild, given experimental density information. Here, we try to infer the structure of 1CID from a threaded homology model, given synthesized electron density data.

Given this threaded model – and the density data – we first pick out the residues we are going to aggressively rebuild. Loops_from_density is a straightforward application that uses a model's secondary structure and it's fit to density to select candidate residues for rebuilding. The application is run using the command line in *scenario4_loop_remodel/run1_make_loopfile.sh*.

```
bin/loops_from_density.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in::file::s 1cid_threaded.pdb \
      -edensity::mapfile 1cid_5A.mrc \
      -edensity::whole_structure_allatom_wt 1.0 \
      -edensity::realign min \
      -edensity::sliding_window 9 \
      -edensity::mapreso 4 \
      -edensity::grid_spacing 2 \
      -max_helix_melt -1 \
      -max_strand_melt 3 \
      -frac_loop 0.2
```

Again, important flags are boldfaced. The flag `-edensity::realign min` tells Rosetta that the model should be rigid-body-minimized into the density map (using the density weight provided on the command line) before running the protocol. Also, for *loops_from_density*, the sliding-window weight is always used to identify regions with a poor local fit to the density, so a sliding-window width must always be provided. The value given here (9) seems to work reasonably well for maps in the 4-8A range; lower-resolution maps may want to use a wider window.

The flags `-max_helix_melt` and `-max_strand_melt` tell the application not to rebuild more than the specified number of residues into the corresponding secondary structure element. The default, -1, ignores the corresponding secondary structure when selecting residues to rebuild. Finally, `-frac_loop` identifies the fraction of the structure to rebuild (in this case 20%).

There are a couple things to note about this application. First, the names "loop-building" and "loop-model" are a bit of a misnomer; Rosetta's loopmodel application will happily rebuild secondary structure elements. In many cases, it may be beneficial to rebuild a helix. Secondly, the *loops_from_density* application is not actually rebuilding segments of the structure. It merely generates an input file for the *loopmodel* application, which does the actual rebuilding.

After running this script, the application produces the file *scenario4_loop_remodel/ 1cid_threaded.loopfile*. The header of this file contains the per-residue density correlation. The bottom of this file contains the loopfile, used by the *loopmodel* app:

```
LOOP  23 26  0 0
LOOP  30 42  0 0
```

Each line in this file contains five columns. The first is the keyword "LOOP". The next two are a residue range. ***Notice this residue range ignores PDB numbering.*** Instead, *Rosetta numbering* is used, where the first residue is "1" and remaining residues are numbered consecutively. The fourth and fifth columns may be left as 0. After running this application, you will want to investigate this loopfile by hand, to see if it makes sense.

Once the loopfile has been verified, we are ready to remodel the loops. The command-line for this is in *scenario4_loop_remodel/run2_loopmodel.sh.*

```
bin/loopmodel.linuxgccrelease \
      -database ~/rosetta_database/ \
      -loops::input_pdb 1cid_threaded.pdb \
      -loops::loop_file 1cid_threaded.loopfile \
      -nstruct 1 \
      -in::file::fullatom \
      -loops::remodel quick_ccd_moves \
      -loops::intermedrelax no \
      -loops::refine no \
      -loops::relax fastrelax \
      -loops::frag_sizes 9 3 1 \
      -loops::frag_files aa1cid_09_05.200_v1_3.gz  aa1cid_03_05.200_v1_3.gz none \
      -loops::extended \
      -loops::random_grow_loops_by 4 \
      -relax::fastrelax_repeats 1 \
      -relax::jump_move true \
      -edensity::mapfile 1cid_5A.mrc \
      -edensity::realign min \
      -edensity::mapreso 5.0 \
      -edensity::grid_spacing 2.0 \
      -edensity::whole_structure_allatom_wt 0.05 \
      -score:weights score13_env_hb
```

The input file and loop file are given with the flags `-loops::input_pdb` and `-loops::loop_file`, respectively.

The loopmodel application runs a highly configurable four-stage protocol. The first stage performs localized fragment insertion using Rosetta's low-resolution potential (where sidechains are modeled using a single interaction center). Four flags control the four stages (omitted flags default to 'no'):

-loops::remodel { no | perturb_kic | quick_ccd | quick_ccd_moves }
The initial centroid-mode loopbuilding protocol. For performance reasons, *quick_ccd_moves* is recommended when refining into density.

-loops::intermedrelax { no | fastrelax | fullrelax }
An intermediate fullatom relax.

-loops::refine { no | refine_ccd | refine_kic }
Fullatom refinement, with backbone motion restricted to the loop only.

-loops::relax { no | fastrelax | fullrelax }
A final fullatom relax.

Generally, only one of `-loops::refine` or `-loops::relax` will be specified, depending on whether backbone motions outside the "loop" regions are desired. If, for example, in homology modeling we wanted to close a gap without moving the backbone of the remainder of the model, we would specify:

```
-loops::remodel quick_ccd_moves
-loops::intermedrelax no
-loops::refine refine_ccd
-loops::relax no
```

The flag `-nstruct` specifies the number of models to build using the protocol. Unlike the relax application – where 10 models is usually more than sufficient – generally hundreds (for building a single relatively short loop) to tens of thousands of models (for rebuilding multiple loops or long loops) is necessary to reasonable sample the search space.

One other thing to notice with the loopmodel application is that backbone fragment files have to be provided with `-loops::frag_files`. These fragment files are specific to a particular sequence, and are required for loop modeling. The easiest way to make fragment files is to submit your sequence at http://robetta.bakerlab.org/. Alternatively, the Rosetta manual (see http://www.rosettacommons.org/manuals/archive/rosetta3.1_user_guide/file_fragments.html) describes how you can make your own fragment files.

Finally, if relax is enabled as part of the protocol, the standard relax flags described in the previous scenarios are all applicable. The standard density flags are also applicable.

Scenario 5: Distant homology modeling using "cm"

This scenario contains code not available in the 3.1 release

In this scenario, we use the *cm* application to automation the construction of homology models from a template and an alignment. The application automatically performs the threading and calls *loopmodel* and *relax* as needed, without needing to set up loop files.

This scenari, which builds a model from 1XVQ from an alignment to 2BMX, is run using the command line in *scenario5_comparative_modeling/run_cm_into_density.sh*.

```
bin/cm.linuxgccrelease \
      -database /opt/rosetta-3.1/rosetta_database/ \
      -in::file::extended_pose 1 \
      -in::file::fasta 1XVQ.fasta \
      -in::file::alignment 1xvq_2bmx.fasta \
      -in::file::template_pdb 2bmxA.pdb \
      -cm::min_loop_size 4 \
      -cm::loop_close_level 0 \
      -cm::loop_rebuild_filter 50 \
      -loops::frag_sizes 9 3 1 \
      -loops::frag_files aaxvqn_09_05.200_v1_3.gz aaxvqn_03_05.200_v1_3.gz none \
      -loops::remodel quick_ccd_moves \
      -loops::idealize_after_loop_close \
      -loops::relax fastrelax \
      -relax::fastrelax_repeats 1 \
      -relax::jump_move true \
      -edensity::mapfile 1XVQ.5A.mrc \
      -edensity::realign min \
      -edensity::mapreso 5.0 \
      -edensity::grid_spacing 2.5 \
      -edensity::whole_structure_allatom_wt 0.05 \
      -score:weights score13_env_hb \
      -out::pdb
```

Most of the flags here have the same meaning as in the loopmodel application of the previous section. New flags are highlighted in boldface. The alignment and template are provided with the arguments `-in::file::alignment` and `-in::file::template_pdb`. The flag `-cm::min_loop_size` pads insertions smaller than the specified size so that rebuilding is at least this many residues (the value given here, 4, seems reasonable for most applications).

Finally, the flag `-cm::loop_rebuild_filter` continues rebuilding loops (in centroid mode) until Rosetta's low-resolution energy function is below the specified value. This value is *very* structure-specific, so it is typically necessary to do a couple test runs before deciding on a reasonable cutoff value. A value of 50 here is probably a reasonable starting point.

Scenario 6: Building an all-atom model guided by an approximate C-alpha trace

This scenario introduces another application. This application, *ca_to_allatom*, is intended to aid in building a full-atom model from a low-resolution C-alpha-only trace (such as hand-traced models from cryo-electron microscopy).

The input to the protocol includes the initial C-alpha trace and a rigid-body segmentation file, which identifies secondary-structure elements in the initial trace. The protocol is divded into 4 phases:

Fragment insertion -- variable-length fragments are inserted at each secondary structure element. The fragments are dynamically selected based on RMS to the trace, and the insertion ensures that the fragment is in the same relative orientation as the original trace.

Rigid-body perturbation -- Individual secondary structure elements are perturbed in a Monte-Carlo trajectory. Optionally, sequence-shifting moves explore alternate threadings of the model.

(*optional*) **Loop remodeling**

(*optional*) **Relax**

We build up a model of 1bbh from a threaded model , using synthesized density data. The command line is at *scenario6_model_from_ca_trace/run1_bootstrap_ca_cen.sh*.

```
bin/ca_to_allatom.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in:file:s 1bbh_threaded.pdb \
      -in:file:centroid_input \
      -in:file:native 1bbh.pdb \
      @density_flags \
      @loopmodel_flags \
      -RBSegmentRelax::cst_wt 0.1 \
      -RBSegmentRelax::cst_width 2.0 \
      -RBSegmentRelax::rb_scorefxn rb_cen.wts \
      -RBSegmentRelax::rb_file 1bbh.rbsegs \
      -RBSegmentRelax::nrbmoves 200 \
      -RBSegmentRelax::helical_movement_params 30.0 0.5 2.0 0.5 \
      -loops::vall_file ~/rosetta_fragments/nnmake_database/vall.dat.2006-05-05
```

As applications sample more and more aggressively, the command lines become more complicated. To simplify things somewhat, Rosetta uses the `@` symbol to reference files which contain lists of flags. here, the file '*density_flags*' contains all the `-edensity::*` flags and '*loopmodel_flags*' contains all the `-loops::*` and `-relax::*` flags (the complete command line is given in the script file).

The `-RBSegmentRelax::*` flags control the fragment insertions and rigid-body perturbation. The first two flags, `cst_wt` and `cst_width` control the constraints placed on C-alpha atoms in the starting model. Models with C-alpha deviations more that *cst_width* will be penalized during rigid-body perturbation.

The score function used (during the initial rigid-body prerturbation) is given by the flag `-RBSegmentRelax::rb_scorefxn`. Here's we're using a custom scoring function (*rb_cen.wts*) that combines Rosetta's low-resolution (centroid) energy terms with the full-atom backbone hydrogen bonding terms. This score function seems to work well at rebuilding beta sheets.

```
env         1.0
pair        1.0
cbeta       1.0
vdw         1.0
hbond_lr_bb 4.0
hbond_sr_bb 1.0
rama        0.2
rg          2.0
```

The flag `-RBSegmentRelax::nrbmoves` tells Rosetta how many steps to make during the first stage of the protocol (for large structures in centroid mode, several thousand may not be unreasonable).

The flag `-RBSegmentRelax::helical_movement_params` (and the related `strand_movement_params`) tell Rosetta the magnitude of the perturbations. The first two numbers are the angle and translation along the helical axis (or along the strand) while the second two are the rotation/translation normal to this axis.

The flag `-loops::vall_file` gives the path to the *vall_file*, needed by the protocol, and included with the Rosetta distribution.

Finally, for some all-helical proteins, it may make more sense to run the rigid-body perturbation mode using an all-atom model and Rosetta's high-resolution energy function. This can be done by replacing the flag `-in:file:centroid_input` with `-in:file:fullatom`, and giving `-RBSegmentRelax::rb_scorefxn` a full-atom score function (like *score13_env_hb*). See the command line at *scenario6_model_from_ca_trace/run2_bootstrap_ca_allatom.sh*.

Scenario 7: Modeling systems with internal symmetry

This scenario contains code not available in the 3.1 release

This scenario describes the use of Rosetta's symmetry mode for modeling symmetric protein complexes using both the *relax* and *looprelax* applications. The sample case involves refinement of an ab initio model of the (C4) tetramer 1K4C into synthesized density data.

The information that Rosetta needs to know about a symmetric system is encoded in the *symmetry definition file.* Its tells Rosetta: (a) how to score a structure, (b) how to maintain symmetry in rigid body perturbations, (c) what degrees of freedom are allowed to move, (d) how to initially setup the system and (e) how to perturb the system to preserve the symmetry and absolute coordinate frame of the input protein assembly.

To aid in creating a symmetry definition file from a symmetric (or near-symmetric) PDB, an application, *make_symmdef_file.pl*, has been included in src/apps/public/symmetry. To generate the symmetry definition file for 1K4C, we run the command in *scenario7_symmetric_modeling/ run1_make_symmdef.sh*.

```
~/rosetta_source/src/apps/public/symmetry/make_symmdef_file.pl \
      -m NCS -a A -i B \
      -p 1K4C_mem_abrelax.pdb > 1K4C.symm
```

This script needs minimal information: the type of symmetry to generate (here NCS), the primary chain (here A), and an adjacent chain in each symmetry group, separated by spaces (here just B). For C*n* symmetries, only one adjacent chain is given; for D*n*, two are given. If the input system is asymmetric, the script will make a symmetrical version of it (sometimes significantly perturbing it in the process).

In addition to the definition file written to STDOUT, the script writes a few other files:

1K4C_mem_abrelax_symm.pdb
: the symmetrized version of the input file, showing the complete point symmetry group.

1K4C_mem_abrelax_model_AB.pdb
: the same as above, but only showing chains that form an interface (where interface is defined by having a Ca-Ca distance less than 8A; or another value specified with -r) with chain A

1K4C_mem_abrelax_INPUT.pdb
: the input PDB to Rosetta's symmetry modelling. A single chain in the symmetric complex.

1K4C_mem_abrelax.kin
: a KineMage image showing the connectivity of subunits in Rosetta

The symmetry definition file looks something like this:

```
symmetry_name 1K4C_mem_abrelax__4
E = 4*VRT0_base + 4*(VRT0_base:VRT3_base) + 2*(VRT0_base:VRT2_base)
anchor_residue 54
…
set_dof JUMP0_to_com x(11.7023996817515)
set_dof JUMP0_to_subunit angle_x angle_y angle_z
…
```

The omitted sections describe a system of virtual residues that maintain the symmetry of the system, and they generally should remain unedited. The *set_dof* lines are what the user might want to edit. For fitting structures into density, in addition to the symmetric degrees of freedom, we want to allow the rigid body orientation of the entire system to move as well. Thus, we need to add the following line (see *scenario7_symmetric_modeling/1K4C_edited.symm*):

```
set_dof JUMP0 x y z angle_x angle_y angle_z
```

This input file can now be used to refine 1K4C symmetrically. For example, we can relax our model using the command line in *scenario7_symmetric_modeling/run2_symm_relax.sh*:

```
bin/relax.linuxgccrelease \
      -database ~/rosetta_database \
      -in:file:s 1K4C_mem_abrelax_INPUT.pdb \
      -symmetry:symmetry_definition 1K4C.symm \
      -symmetry::initialize_rigid_body_dofs \
      -score::weights score12_nosol.wts \
      @relax_options \
      @density_options \
      -edensity::score_symm_complex true
```

Only a couple flags are different from the nonsymmetric version of relax. The two `-symmetry::*` flags tell the relax application to run in symmetric mode. The flag `initialize_rigid_body_dofs` tells Rosetta to use the parenthesized values in the *set_dof* lines to initialize the symmetric model.

Because this is a membrane protein we are using a modified score function, where the solvation term *fa_sol* is turned off.

Finally, the flag `-edensity::score_symm_complex true` tells Rosetta to score the entire structure's fit-to-density as opposed to a single subunit. It is using this flag that Rosetta may fit a symmetric model into asymmetric density data, simultaneously inferring the conformation and symmetric operations.

The *loopmodel* application also understands symmetry. The command line given in *scenario7_symmetric_modeling/run3_symm_looprelax.sh* illustrates:

```
bin/loopmodel.linuxgccrelease \
      -database ~/rosetta_database \
      -loopmodel_app::viewer true \
      -symmetry:symmetry_definition 1K4C_edited.symm \
      -symmetry::initialize_rigid_body_dofs \
      -loops::input_pdb 1K4C_mem_abrelax_INPUT.pdb \
      -loops:loop_file 1K4C.loopfile \
      @loopmodel_options \
      @density_options \
      -loops::build_attempts 10 \
      -edensity::score_symm_complex true
```

The flags are the same as in Scenario 4. The only recommended exceptions are if one is rebuilding self-interacting loops, as in this scenario. In these cases Rosetta may have difficulty closing loops, and so allowing Rosetta more chances to build the loop, with `-loops::build_attempts 10` (the default is

3) is recommended. There is also some evidence that using `-loops::remodel quick_ccd` instead of `-loops::remodel quick_ccd_moves` may also help.

Finally, *fragment files should be generated and loop files should be defined over the monomer*.

Scenario 8: Iterative loop-modeling for extremely distant homology modeling

In this scenario, we briefly describe iterative loop-modeling, a very powerful tool when performing homology models with low sequence identity. We show an example building homology models for 1Q0P from 1IDO. Most of the techniques have been described in previous sections; this scenario shows how these methods can be combined to tackle some very difficult modeling problems.

The basic idea of iterative loop modeling is to use a large number of *loopmodel* trajectories to explore conformational space. Then, from among these trajectories, select some small subset. Generally, to maintain diversity, the lowest energy models are clustered; no more than one model from each cluster is selected. These selected models are then carried to the next generation, and are used as the starting points for another round of *loopmodel* trajectories.

As an example, 80 low-energy models for 1Q0P are in the folder *scenario8_iterative_loopmodel/generation1*. We can cluster these models using the command line in *scenario8_iterative_loopmodel/run1_cluster_gen1.sh:*

```
cd generation1

bin/cluster.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in::file::s *.pdb \
      -radius 2.5
```

The main parameter for clustering is **-radius**, which describes how different models have to be before they are placed in a new cluster. The clustering produces files named c.*m*.*n*.pdb, where *m* refers to the cluster number, and *n*=0 is the centroid model with the remainder sorted by energy. For example, the following command copies the lowest-energy models from each cluster to the next generation.

```
cp c.*.1.pdb ../generation2
```

We then run another generation of *loopmodel* starting with each cluster center. First we identify regions to aggressively rebuild (see *scenario8_iterative_loopmodel/run2_make_gen2_loopfiles.sh*):

```
cd generation2

bin/loops_from_density.linuxgccrelease \
      -database ~/rosetta_database/ \
      -in::file::s *.pdb \
      @density_flags \
      -edensity::sliding_window 9 \
      -max_helix_melt -1 \
      -max_strand_melt 2 \
      -frac_loop 0.4
```

Then the actual *loopmodel* application (see *scenario8_iterative_loopmodel/run3_looprelax_gen2.sh*):

```
cd generation2

for file in *.pdb
do
```

```
stem=`echo $file | sed 's/\.pdb//'`

bin/loopmodel_viewer.linuxgccrelease \
      -database /opt/rosetta-3.1/rosetta_database/ \
      -loops::input_pdb $stem.pdb \
      -loops::loop_file $stem.loopfile \
      -nstruct 1 \
      -in::file::fullatom \
      -loops::frag_sizes 9 3 1 \
      -loops::frag_files ../aa1q0p_09_05.200_v1_3.gz  \
                         ../aa1q0p_03_05.200_v1_3.gz none \
      -loops::extended \
      -loops::random_grow_loops_by 4 \
      -relax::fastrelax_repeats 1 \
      -relax::jump_move true \
      -edensity::mapfile ../1q0p.5A.mrc \
      -edensity::realign min \
      -edensity::mapreso 5.0 \
      -edensity::grid_spacing 2.0 \
      -edensity::whole_structure_allatom_wt 0.05 \
      -score:weights score13_env_hb \
      -overwrite &> /dev/null &

done
```

Scenario 9: Ab-initio modeling for completing a partial structure

This scenario contains code not available in the 3.1 release

This scenario briefly introduces the topology broker, the most recent version of Rosetta's ab-initio protocol. In this case, we use the topology broker to repair a model of 1NSF that was over-refined against low-resolution crystallographic data.

For this case, the helices in the structure were well resolved. However, the beta sheet geometry was destroyed. Loopmodeling is insufficient in this case since fixing the beta sheet requires the concerted movement of many contiguous segments. Here we show how the topology broker may be used to rebuild 1NSF while keeping the helices fixed.

The command line we pass the broker is given by the command line in *scenario9_density_ab_initio/ run1_jumping_abinitio.sh*:

```
bin/r_broker.linuxgccrelease \
      -database /opt/rosetta-3.1/rosetta_database/ \
      -broker:setup setup_dens.tpb \
      -database /work/dimaio/minirosetta_database \
      -skip_stages 1 2 \
      -seq_sep_stages 1 1 1 \
      -short_frag_cycles 1 \
      -scored_frag_cycles 1 \
      -increase_cycles 0.1 \
      -ramp_chainbreaks \
      -sep_switch_accelerate 0.8 \
      -skip_convergence_check \
      -overlap_chainbreak \
      -fail_on_bad_hbond false \
      -edensity::mapfile 1nsf.5A.mrc \
      -edensity::mapreso 8.0 \
      -edensity::grid_spacing 4.0 \
      -abinitio::stage3a_patch ./phase3.patch \
      -abinitio::stage3b_patch ./phase3.patch \
      -abinitio::stage4_patch  ./phase4.patch
```

While a discussion of all the flags used by the broker is beyond the scope of this tutorial, the key ones are boldfaced. First, the flag `-broker:setup` specifies the file used to describe the topology of the system, what parts may move, how those parts may move, and the direction of the chain towards which those moves are propagated. This is done through a series of claimers, while the broker is used to determine which claimer may access which part of the structure.

For this application the broker setup file is in *scenario9_density_ab_initio/setup_dens.tpb*:

```
CLAIMER SequenceClaimer
LABEL DEFAULT
FILE 1nsf.fasta
END_CLAIMER

ABINITIO_FRAGS
LARGE aansfd_09_05.200_v1_3.gz
SMALL aansfd_03_05.200_v1_3.gz
END_ABINITO
```

The first block of code in this file sets up a very simple ab initio model. The sequence claimer sets up an extended chain using the sequence in *1nsf.fasta*. The second claimer is where the fragment files are given, and it tells the broker to allow fragment insertion anywhere in the model where no claimers otherwise prevent it.

```
CLAIMER DensityScoringClaimer
anchor 98
END_CLAIMER
```

The next claimer sets the model up for density scoring. It tells Rosetta: (a) that the overall rigid-body orientation of the system is important, and (b) that the rigid body orientation of the system is maintained by propagating all fragment insertions outward from residue 98.

```
CLAIMER CoordConstraintClaimer
PDB_FILE 1nsf_bad_refine.pdb
ASK_FOR_ROOT ALL
POTENTIAL BOUNDED 0.0 4 1 xyz
END_CLAIMER
```

This claimer sets coordinate constraints on the C-alpha atoms from *1nsf_bad_refine.pdb*. The constraints are bounded constraints, of width 4 and weight 1.

```
CLAIMER RigidChunkClaimer
pdb 1nsf_bad_refine.pdb
REGION
RIGID 1 5 0 0
RIGID 24 42 0 0
RIGID 61 68 0 0
RIGID 90 105 0 0
RIGID 135 143 0 0
RIGID 161 171 0 0
RIGID 185 195 0 0
RIGID 200 207 0 0
RIGID 219 229 0 0
RIGID 234 247 0 0
END_REGION
END_CLAIMER
```

This last block is where most of the work is done. The rigid-chunk claimer sets up regions where no fragment insertions may take place. In between these regions, cuts are introduced in the chain; all fragment insertions in these regions propagate toward the cut. For this scenario these regions correspond to the helices and the N-terminus, which we want to remain fixed.

While this protocol is extremely powerful for very difficult modeling problems, the sampling required is very significant; in general tens to hundreds of thousands of models are required to accurately sample conformational space.

Scenario 10: Modeling a ligand-containing structure into density

The final scenario shows how ligands may be added to proteins modeled in Rosetta. Here, we again visit the case of 1BBH, this time also modeling a heme present. There are many options for modeling and docking small ligands in Rosetta; this demo is only showing a small part of what is possible.

New ligands are described to Rosetta using a params file, which defines ligand topology, rotatable bonds, atom types, partial charges, etc. A script called *molfile_to_params.py* has been supplied in *src/python/apps/* to help in producing these files from a typical small molecule format (.mol, .sdf, or .mol2).

For the heme case, we begin with the heme molecule in HEM.mol2, the run the command line in *scenario10_ligand_modeling/run1_mol2params.sh*:

```
python ~/rosetta_source/src/python/apps/public/molfile_to_params.py \
      --keep-names --clobber HEM.mol2 -p HEM -n HEM
```

This outputs the param file HEM.params. This will only create a single conformation of the ligand. If multiple conformations of the ligand are desired, a multi-model PDB of conformers must be created (using a program like OpenEye's Omega) and added to the *.param file under the keyword PDB_ROTAMERS.

The params file is given to Rosetta with the flag `-extra_res_fa`, as in the command line at *scenario10_ligand_modeling/run2_relax_with_heme.sh*:

```
bin/relax.linuxgccrelease \
       -database /opt/rosetta-3.1/rosetta_database/ \
       -in::file::s 1bbh_threaded_with_heme.pdb \
       -score:weights score13_env_hb \
       -relax_app::viewer \
       -relax::fast \
       -relax::fastrelax_repeats 1 \
       -relax::jump_move true \
       -extra_res_fa HEM.params \
       -edensity::mapfile 1bbh.5A.mrc \
       -edensity::mapreso 5.0 \
       -edensity::grid_spacing 2.0 \
       -edensity::whole_structure_allatom_wt 0.1
```

The output is a relaxed model with the ligand present. Since no PDB_ROTAMERS line is provided, the ligand moves very little; the rotatable bonds may minimize but will not be repacked. The ligand's fit to density is used, however. In addition, while some terms of Rosetta's energy are applicable to ligand conformations, many of the statistical terms are not. So it may be necessary to constrain the ligand to maintain proper binding geometry.