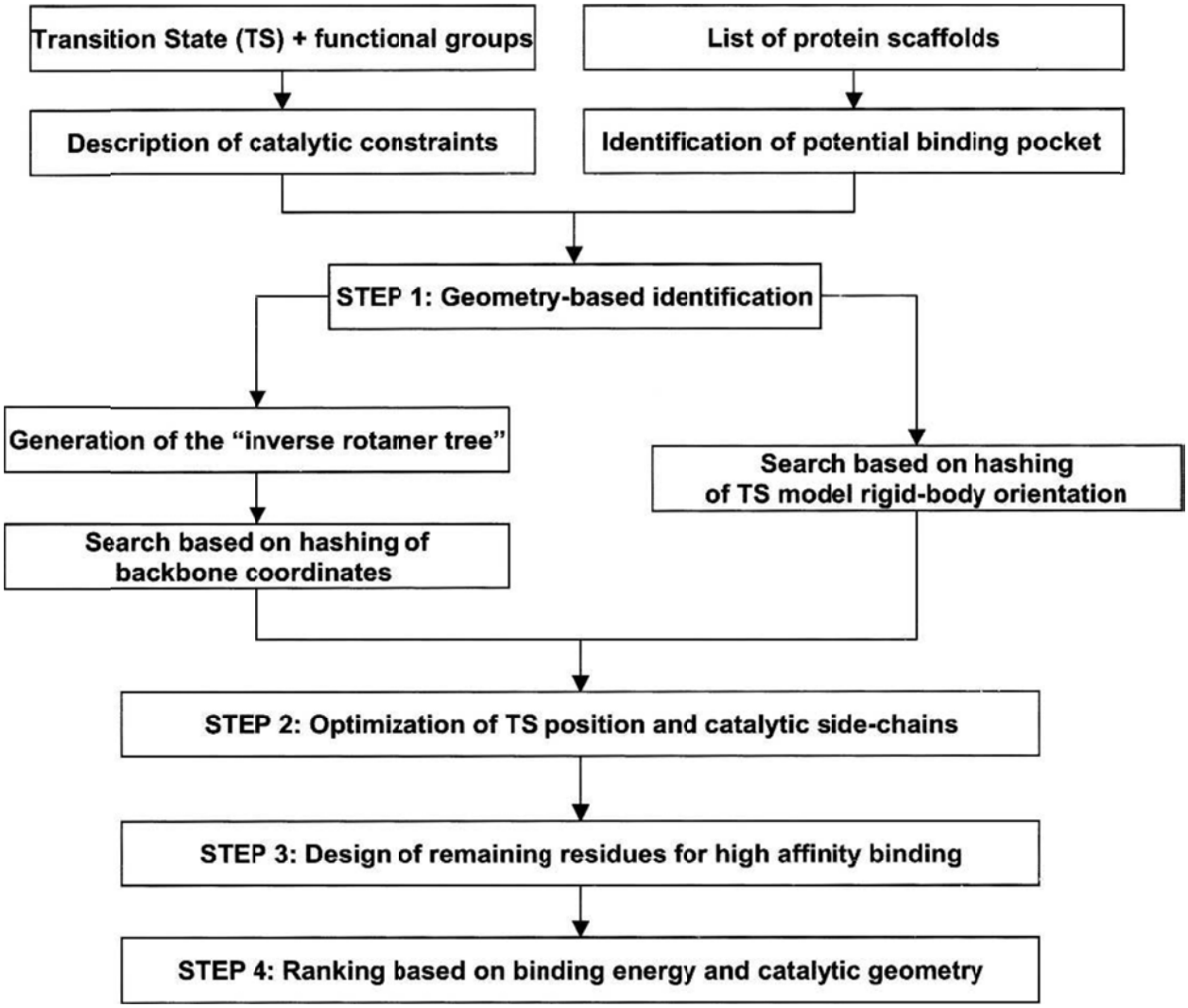
This document is a tutorial for the RosettaEnzyme application. This was solely written by Steven Combs and Dietmar Birzer, two people who don't work on enzyme design. It borrows heavily from the Rosetta 3.2 documentation. Florian Richter was consulted heavily when creating this document (a developer who codes for RosettaEnzyme). This tutorial assumes that you know and understand how RosettaDesign and RosettaLigand work. Without an understanding for those two applications, you will not be able to perform the RosettaEnzyme application. This tutorial will cover the theory behind enzyme design, the algorithm of enzyme design, and how to perform the kemp elimination seen in *Nature* **453**, 190-195 (8 May 2008). This tutorial will not discuss how to use RosettaMatch.

RosettaEnzyme Algorithm

The RosettaEnzyme algorithm is discussed in great detail in the paper Zanghellini et al. **Protein Science** (2006), 15:2785–2794.



In addition to understanding the algorithm, you must have the following:

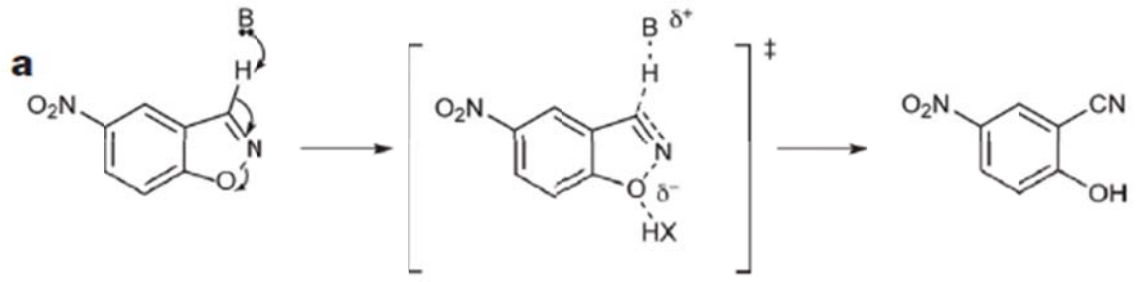
1. An enzyme reaction
2. An intimate knowledge of the mechanism for enzyme reaction
3. Known intermediates for the enzyme reaction
4. A protein scaffold (RosettaMatch can find one for you, but alas, we are not covering it)
5. Magic

All of these will be covered in the following sections.

Theozymes: Theoretical enzymes used for computational enzyme designs

Theozymes are theoretical enzymes that have an ideal arrangement of the transition state structure surrounded by catalytic functional groups. In general, a theozyme is created computational via quantum mechanical studies. In this process, a transition state is modeled and geometrically optimized. Once the transition state is geometrically optimized, catalytic functional groups are added to stabilize the TS. The actual amino acid side chain is not used, instead, ammonium is used for lysine, acetamide for aspargine, so forth so on. The placement of the functional groups are done through approximation of ideal distances/angles. Then, quantum mechanical studies are used to find the optimal geometric constraints. A review of such techniques can be found in the paper: Tantillo Current Opinion in Chemical Biology 1998, 2:743-750 A set of theozymes was created in this paper: Zhang ACS 2008. Both are from Kent Houk's group. Quantum mechanical studies done in the paper are done by Gaussian. This tutorial will not cover how to use Gaussian.

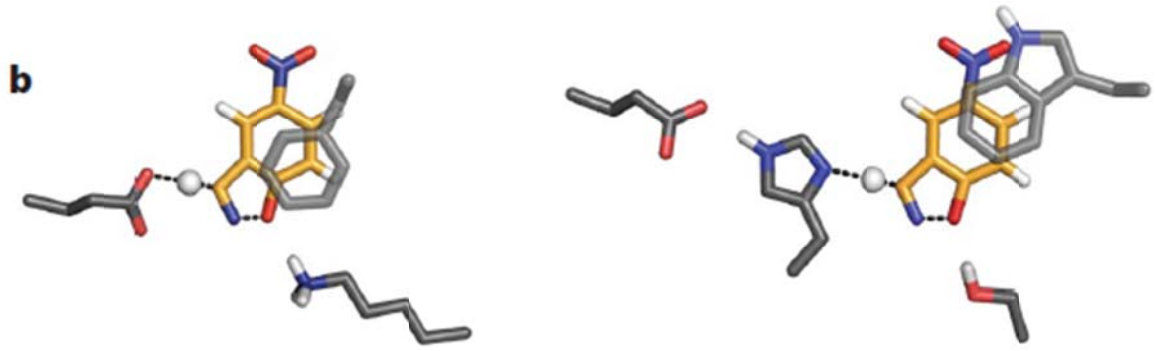
The Baker group uses idealized bonds/angles in their theozymes. For our purposes, we will follow the Baker group's strategy. In particular, this example will use the kemp elimination reaction that was previously made in the baker lab. A kemp elimination reaction "has been extensively studied as an activated model system for understanding the catalysis of proton abstraction from carbon—a process that is normally restricted by high activation-energy barriers." First, we need a reaction mechanism for the theozyme.



From the paper: The key step for the Kemp elimination is deprotonation of a carbon by a general base. We chose two different catalytic bases for this purpose: first, the carboxyl group of an aspartate or

glutamate side chain, and, second, the imidazole of a histidine positioned and polarized by the carboxyl group of an aspartate or glutamate (we refer to this combination as a His–Asp dyad).

The theozymes needs a proton abstractor (base) and a proton donor (acid). To accomplish this, the acidic residues and basic residues are used. To stabilize the ligand in the pocket several other residues are used. These consist of a pi-stacking residues.



For both the carboxylate- and histidine-based mechanisms, we included additional functional groups in the idealized active sites to further facilitate catalysis using both quantum mechanical and classical methods. A hydrogen bond donor was used to stabilize the developing negative charge on the phenolic oxygen in the otherwise hydrophobic active site...Finally, because stabilization of the transition state by charge delocalization is a key factor in catalysis of the Kemp elimination5–7,10,14, we chose to stack aromatic amino acid side chains on the planar transition state using idealized p-stacking geometries

Now that we have an understanding of how a theozyme is made, we can take this information and convert it into constraints for Rosetta.

The constraints file!

Once we have chosen catalytic residues for our enzyme, we need to convert this information into constraints that Rosetta understands. We do this because from our previous calculations, we know what stabilizes the transition analog and we know what we need those residues to be within a certain distance of the TS. Rosetta uses a KBP and does not "understand" QM calculations. In order for us to fix a residue in a specific location so that it creates a TS stabilizing geometry we use constraints. Below is an example.

Overall organization of the constraints file:

```
#block 1 for proton abstracting histidine

CST::BEGIN
  TEMPLATE::   ATOM_MAP: 1 atom_name: C6 O4 O2
  TEMPLATE::   ATOM_MAP: 1 residue3: D2N


  TEMPLATE::   ATOM_MAP: 2 atom_type: Nhis,
  TEMPLATE::   ATOM_MAP: 2 residue1: H


  CONSTRAINT:: distanceAB:    2.00   0.30 100.00  1
  CONSTRAINT::    angle_A:  105.10   6.00 100.00  360.00
  CONSTRAINT::    angle_B:  116.90   5.00  50.00  360.00
  CONSTRAINT::  torsion_A:  105.00  10.00  50.00  360.00
  CONSTRAINT::  torsion_B:  180.00  10.00  25.00  180.00
  CONSTRAINT:: torsion_AB:    0.00   0.00   0.00  180.00
CST::END

#block 2 for aromatic binding

CST::BEGIN
  TEMPLATE::   ATOM_MAP: 1 atom_name: X1 C10 C12
  TEMPLATE::   ATOM_MAP: 1 residue3: D2N


  TEMPLATE::   ATOM_MAP: 2 atom_type: aroC,
  TEMPLATE::   ATOM_MAP: 2 residue1: WFY


  CONSTRAINT:: distanceAB:    3.50   0.20  50.00  0
  CONSTRAINT::    angle_A:   90.00   5.00  50.00  360.00
  CONSTRAINT::    angle_B:   90.00   5.00  50.00  360.00
  CONSTRAINT::  torsion_A:   90.00   5.00  50.00  180.00
  CONSTRAINT::  torsion_B:  180.00  15.00   0.00  120.00
  CONSTRAINT:: torsion_AB:    0.00   0.00   0.00  180.00
CST::END
```

Comments are seen in the constraints file via the # mark. This format is broken up into two blocks: catalytic histidine and aromatic binding. These blocks are defined by the `CST::BEGIN` and `CST::END` and everything that is contained within the notations that are indented. The residues that are being are followed by `TEMPLATE` designation. These are the two residues that will have the constraints applied to them. Below the `TEMPLATE` designation is the `CONSTRAINT` notation. Everything that are labeled `CONSTRAINT` are constraints that are placed upon your residues defined in `TEMPLATE`. Lets delve a little bit more into how to define residues in `TEMPLATE`.

TEMPLATE SECTION

```
#block 1 for proton abstracting histidine

  TEMPLATE::   ATOM_MAP: 1 atom_name: C6 O4 O2
  TEMPLATE::   ATOM_MAP: 1 residue3: D2N


  TEMPLATE::   ATOM_MAP: 2 atom_type: Nhis,
  TEMPLATE::   ATOM_MAP: 2 residue1: H

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

#block 2  for aromatic binding

  TEMPLATE::   ATOM_MAP: 1 atom_name: X1 C10 C12
  TEMPLATE::   ATOM_MAP: 1 residue3: D2N


  TEMPLATE::   ATOM_MAP: 2 atom_type: aroC,
  TEMPLATE::   ATOM_MAP: 2 residue1: WFY
```

The template tag defines the residue to constrain, either protein side or ligand, and what atoms used to constrain. ATOM_MAP refers to what residue you are talking about. The number is either 1 or 2. Remember, you are constraining two residues. Residue one is identified by the ATOM_MAP: 1 tag and residue two is identified as `ATOM_MAP: 2 flag`. The type of residue to be constrained is identified by the residue3 or residue1 flag. The simplest case to use is the residue1 flag. residue1 is followed by the one letter amino acid code. If you do not care if pi-stacking interactions are made by either a W or F or Y you can place all three one letter amino acid names there. For example in block 2 residue 2, we don't care what pi-stacking interaction is made so we use residue1: WFY notation. Note that if you use this you will have to use the atom_type flag to identify the residues which is discussed below. The residue3 notation will always be used to identify the ligand as the ligand will always have a three letter identifier to it. In block 1 residue 1 we use the residue3: D2N to denote the ligand.

The type of atoms to be constrained can be identified by either the atom_name or atom_type flag. It is important to understand that both flags require 3 atoms to be identified. This is because we want to add angle constraints, which require three atoms. Specifically in block 1 for residue1, if you use the atom_name flag, the first atom name C6, the second O4, and third O2. Alternatively, you can use the atom_type flag which allows a more flexible definition of the constrained atoms. ASK KRISTIAN TO EXPLAIN BASE ATOMS.

The number in column 3 of these records indicates which catalytic residue the record relates to. It has to be either 1 or 2. The 'residue1' or 'residue3' tag specifies what type of residue is constrained. 'residue3' needs to be followed by the name of the residue in 3 letter abbreviation. 'residue1' needs to be followed by the name of the residue in 1 letter abbreviation. As a convenience, if several similar residue types can fulfill the constraint (i.e. ASP or GLU), the 'residue1' tag can be followed by a string of 1-letter codes of the allowed residues (i.e. ED for ASP/GLU, or ST for SER/THR).

It allows more flexible definition of the constrained atoms. It has to be followed by the Rosetta atom type of the first constrained atom of the residue. In case this tag is used, Rosetta will set the 2nd constrained atom as the base atom of the first constrained atom and the third constrained atom as the base atom of the 2nd constrained atom. (Note: the base atoms for each atom are defined in the ICOOR records of the .params file for that residue type). There are two advantages to using the 'atom_type' tag: first, it allows constraining different residue types with the same file. For example if a catalytic hydrogen bond is to be constrained, but the user doesn't care if it's mediated by a SER-OH or a THR-OH. Second, if a catalytic residue contains more than one atom of the same type (as in the case of ASP or GLU), but it doesn't matter which of these atoms mediates the constrained interaction, using this tag will cause Rosetta to evaluate the constraint for all of these atoms separately and pick the one with lowest score, i.e. the ambiguity of the constraint will automatically be resolved.

CONSTRAINT SECTION

```
CONSTRAINT:: distanceAB:    3.50   0.20  50.00  0
CONSTRAINT::    angle_A:   90.00   5.00  50.00  360.00
CONSTRAINT::    angle_B:   90.00   5.00  50.00  360.00
CONSTRAINT::  torsion_A:   90.00   5.00  50.00  180.00
CONSTRAINT::  torsion_B:  180.00  15.00   0.00  120.00
CONSTRAINT:: torsion_AB:    0.00   0.00   0.00  180.00
```

The constraint records identify the distance and angles of the residues' atoms to be used. This is the actual value and strength of the constraint applied between the two residues. A quick glance shows that the line begins with CONSTRAINT:: followed by a designation of A or B whether its a distance or angle. The designation of A or B refers to residue 1 for A and residue for B.

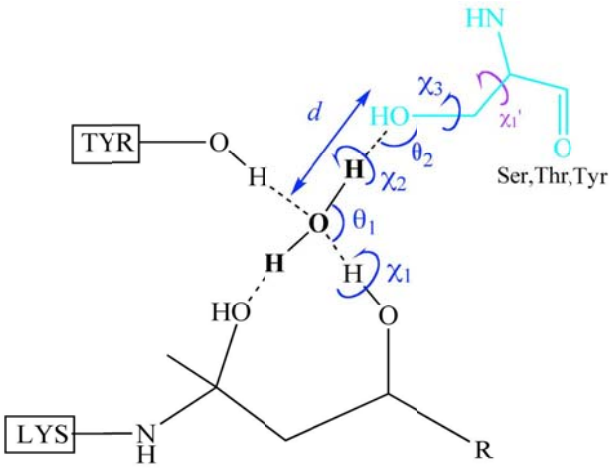
The distanceAB refers to a distance constraint between residue 1, atom 1 and residue 2, atom 2. After the distanceAB comes four numbers. For the distanceAB case, the four numbers are 3.50 0.20 50.00 0. The first number 3.50 specifies the optimum distance between residue1, atom 1 and residue 2, atom 1. Tolerance for the given distance (our example 3.50) is defined in the second column. The number 0.20 identifies the "distance" that we can move from our 3.50 goal. During the design process that means we can move 3.3 or 3.8 away from the optimal distance. The fourth number, 50.00 is the "force" of our constraint. This basically means how often do you allow breaking of our constraint. The higher the number in this column means that models will not be output if the constraint is broken. There is an equation associated with this column. If x is the actual distance (during enzdes run), x0 the optimal distance (defined in constraints) and k = value in 3rd column (force constant, parameter for strength of the constraint) then the score penalty applied will be: 0 if |x - x0| < xtol and k * (|x - x0| - xtol)^2 else. This is, of course, an optimization problem. You will probably have to play around with this number to get it to work. The last number can either be 1 or 0, where 1 means covalent interaction and 0 means non-covalent interaction for the interacting atoms.

The second constraint that can be defined are the angle constraints. These are defined by the angle_A or angle_B constraint. angle_A refers to the angle between Res1:Atom2 - Res1:Atom1 - Res2:Atom1. angle_b refers to the angle Res1:Atom1 - Res2:Atom1 - Res2:Atom2. Similar to the distanceAB, there are

four numbers after the angle_a, angle_b flag: 90.00 5.00 50.00 360.00 (angle_a numbers shown for simplicity). The first number is the optimal angle, which is 90 in this case. The second number is how much deviation which is 5 degrees. The third number is defined above. The fourth numbers adds periodicity to your degrees. For example: If x0 (optimal distance) is 120 and per is 360, the constraint function will have a its minimum at 120 degrees. If x0 is 120 and per is 180, the constraint function will have two minima, one at 120 degrees and one at 300 degrees. If x0 is 120 and per is 120, the constraint function will have 3 minima, at 120, 240, and 360 degrees.

The final set of information that can be constrained are the torsion angles. torsion_A is the dihedral Res1:Atom3 - Res1:Atom2 - Res1:Atom1 - Res2:Atom1, torsion_B is the dihedral Res1:Atom1 - Res2:Atom1 - Res2:Atom2 - Res2:Atom3, torsion_AB is the dihedral Res1:Atom2 - Res1:Atom1 - Res2:Atom1 - Res2:Atom2

To depict the actual constraints in a diagram, see below:



	ideal value	deviation	# of conformations
d	3.0Å	+/-0.2Å [a]	1
θ_1	110°	+/-10° [a]	1
θ_2	120°	+/-20° [a]	1
χ_1	-90° [b]	+/-20°	3
χ_2		every 60°	6
χ_3		every 60°	6
Ser χ_1'	69.6° [c]	+/-12.3°	
Ser χ_1'			3x3
Thr χ_1'			6x3
Tyr $\chi_1' - \chi_2'$			6x9
all			8,748

The description of each section is explained in the following table.

symbol	constraint flag
d	distanceAB
theta1	angle_A
theta2	angle_B
chi1	torsion_A
chi2	torsion_B
chi3	torsion_AB

There is an additional step that is needed to finish everything for the constraints file. The scaffold to be used by your constraints file needs to be edited. In case there is not a scaffold chosen, please look at how to run RosettaMatch. The input pdb is edited to contain the following line (at the top):

REMARK 0 BONE TEMPLATE X D2N 900 MATCH MOTIF A HIS 37 1

From the above line, the REMARK line indicates that this is the start of the constraint information. Your pdb must be cleaned prior to this process with all REMARK lines removed except for this line (use /sb/scripts/get_pdb_new.py to clean your pdb). Ignore the 0 BONE and line for now...this will be discussed in a little.

The TEMPLATE X D2N 900 words are used to describe the ligand. The word TEMPLATE tells Rosetta to start taking information. The X D2N 900 has to match the chain id, ligand name, and residue number of the ligand.

For now, ignore the MATCH MOTIF words. This will be discussed at the end of the section.

The A HIS 37 1 refers to the catalytic residue. The A HIS 37 is the chain id, residue name, and residue number in the pdb for the catalytic residue. The last column 1 refers to the first block of catalytic constraints. Remember, in our example above we had two blocks of catalytic constraints. You can have multiple REMARK lines that refer to all the blocks in your constraints file. The reason for declaring the constraints and the exact residues to constrain in two different places is the following: it allows to use the same .cstfile for any number of matches that have the same type of active site but are in different scaffolds or at different attachment points in the same scaffold.

Now, lets take a look at the 0 BONE and the MATCH MOTIF words. I had no clue what these ment so I wrote Florian. This is his response:

There is only historical significance to the way the REMARK lines are structured. in the old R++ matcher, the lines used to read REMARK BACKBONE TEMPLATE X D2N 900 MATCH MOTIF A HIS 37 1

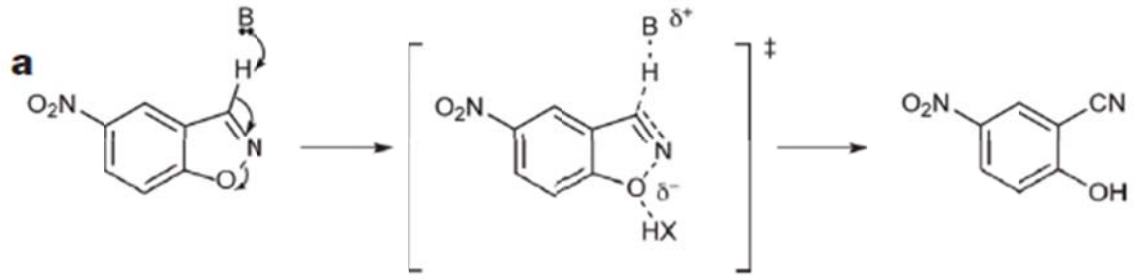
this however didn't conform to pdb format, wehre columns 7-10 or so are reserved for the number of the remark, i.e. mini chopped the backbone to bone.

since I ported the design code before the matcher made it into mini (i.e. the mini code needed to understand r++ matcher output), the function that reads these lines needs the BONE TEMPLATE tag to know it's a line containing the catalytic residues, and there need to be two columns (MATCH MOTIF) between the catres infos.

the numbering is also an historical artefact. the old matcher spit out 0 as the seqpos of the ligand. so in mini, everytime a residue is given as number 0 in the remark lines, this is ok as long as there is only one residue with this name3 in the pose. but you could just as well put in the actual ligand seqpos.

The kemp elimination tutorial!

We will work through a simple example of the Kemp Elimination with enzyme design. We will forgo using RosettaMatch and use the protein HisF (pdb code: 1thf) as our template structure. The first step is to understand the reaction of a kemp elimination.



From this, we see that a base abstracts a hydrogen from the isoxazole ring at the C=N bond. This results in opening of the ring. The transition state can be stabilized via aromatic ring pi-stacking (YFW). Proton donation can occur via any proton donor residue (STKRH). Thanks to this reaction already being done, we know that there are several different combinations of catalytic residues that we can use to get enzymatic activity with 1THF. Below is a table

Table 1| Kinetic parameters of designed enzymes

Design	PDB code of template	Base	Hydrogen-bond donor	π-stack	k_{cat} (s^{-1}; mean ± s.d.)
KE07	1thf	E101	K222	W50	0.018 ± 0.001
KE07	1thf	E101A	K222	W50	0.0009 ± 0.0004
KE07	1thf	E101	K222A	W50	0.030 ± 0.004
KE10	1a53	E178	None	W210	NA
KE10	1a53	E178Q	None	W210	NA
KE15	1thf	D48	None	Y126	0.022 ± 0.003
KE15	1thf	D48A	None	Y126	NA
KE16	1thf	D48	K201	Y126	0.006 ± 0.001
KE16	1thf	D48A	K201	Y126	ND

If you notice from the full table in the paper, 1a53 actually has a better kcat...but we are going to use the 1thf protein because of its history with our lab. For this tutorial we will choose the first KE07 design with E101 as base, K222 as hydrogen-bond donor and W50 for Pi-stacking. So what you need in order to start an EnzDes run are 5 files:

- ligand params file (LG.params)
- ligand conformers pdb file (LG1_confs.pdb)
- protein pdb file (1thf.pdb)
- constraints cst file (kemp.cst)
- flags file

The files can be generated using the tutorial for ligand docking. Please complete this step.

Lets go ahead and make the transition state molecule. A complete tutorial on how to make ligands usable in RosettaLigand can be found here. In general, follow these steps.

1. Draw the transition state in your favorite editor (moe, chemdraw) as it can be seen in the figure above.
    1. sbset moe
    2. moe
2. Save the (minimized) molecule as mol file in 3D not 2D (LG1.mol). In order to be closer to the original designs you might want to take the ligand from one of the structures given in the supplementary data of the Kemp elimination paper, but in this tutorial we will just go on with ligand created above.
3. For the pi-stacking constraint you need to add a virtual atom to the middle of the pi ring

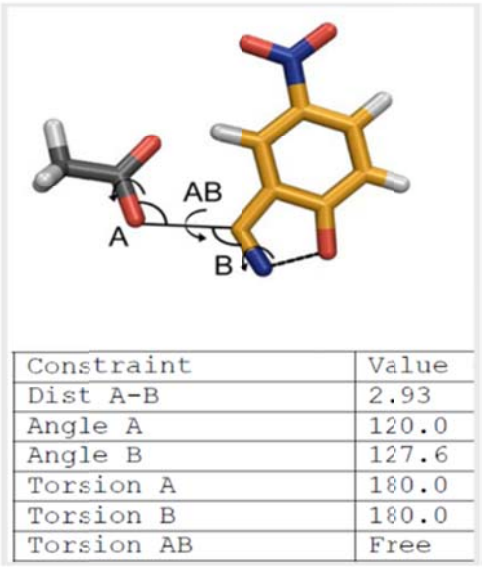
1. Open LG1.mol into pymol and save as LG1.pdb
2. Calculate center of pi ring from two opposing ring atoms
3. Add a virtual atom (X) to the pdb file with these coordinates
4. Assure that the chainID is X and the residue name is LG1
5. Open pdb file and save it again as mol file. Conversion back to mol file does not work with moe as it can not handle virtual atoms! Use pymol instead.

Convert your ligand into a parameter file.

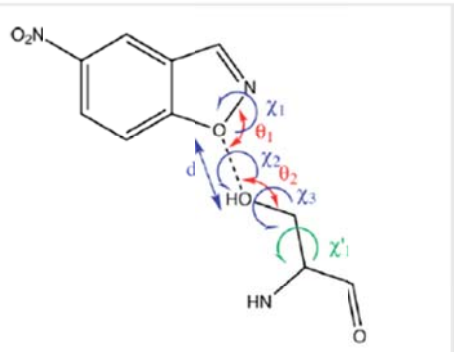
Now there are three things to do with the protein pdb file:

1. Mutate the residues needed for the reaction (E101, K222, W50). This can be done with pymol->Wizard->Mutagenesis. Done worry about which rotamer to select, Rosetta will take care of that.
2. Append a ligand conformer taken from LG1_confs.pdb as starting position for the algorithm. Depending on the relative position of protein and ligand you might have to move the ligand closer to the active site of the protein because otherwise there won't be any residues within the distance cutoffs (see flags section below). You can use pymol to do this, although there certainly are several better ways to tackle this task.
3. Add REMARK statements for the constraints at the top of the file. Here we have three statements, one for E101, one for W50 and one for K222.
   REMARK 0 BONE TEMPLATE X LG1 1 MATCH MOTIF D GLU 101 1
   REMARK 0 BONE TEMPLATE X LG1 1 MATCH MOTIF D TRP 50 2
   REMARK 0 BONE TEMPLATE X LG1 1 MATCH MOTIF D LYS 222 3
   These statements correspond to the constraints in the kemp.cst file that we will set up in the next step.

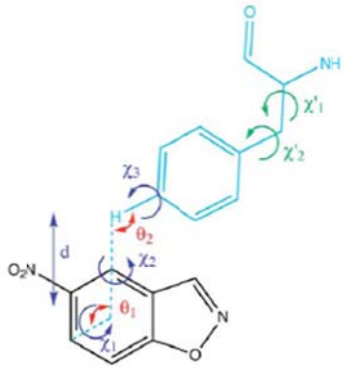
So, in order to create a constraints file the first thing is to determine what angles/lengths to use for our constraints of amino acids. To find these, we would have to use quantum mechanics. Fortunately for us, the distances are listed in the supplementary paper. Also, there is a paper about the Kemp reaction that used quantum mechanics that you could take the values from (J. Am. Chem. Soc. 1996, 118, 6462-6471). We will use the constraints listed in the selection of tables from the Kemp elimination paper below. Unfortunately the tables do not provide all the information needed to set up the constraints. So we will just give it a try and make up something from what we get from the Kemp elimination and from the example cst file that comes along with Rosetta (mini/test/integration/tests/enzdes/inputs/Est_ha_d2n.cst):



Constraint	Value
Dist A-B	2.93
Angle A	120.0
Angle B	127.6
Torsion A	180.0
Torsion B	180.0
Torsion AB	Free



	Ideal Value	Deviation
d	2.98Å	+/-0.2Å
θ_1	103.7°	+/-10°
θ_2	113.0	+/-10°
χ_1	180°	+/-10°
χ_2	Free	Every 15°
χ_3	180°	+/-120°
χ'_1	69.6°	+/-12.3



	Ideal Value	Deviation
d	4.0Å	N/A
θ_1	90°	N/A
θ_2	90°	N/A
χ_1	90°	+/-180°
χ_2		Every 60°
χ_3	90°	N/A

So you can either try to derive a cst file (kemp.cst) yourself (recommended) or use the blocks given below. Each block addresses a different constraint. The ligand atoms are given according to their names in the mol file whereas the side chain atoms are given as atom types, which is OOC for the glutamate oxygens and Nlys for the lysine nitrogen.

Block 1

block 1 for Glu

CST::BEGIN

#ligand

TEMPLATE:: ATOM_MAP: 1 atom_name: C3 N2 C5

TEMPLATE:: ATOM_MAP: 1 residue3: LG1

#protein

TEMPLATE:: ATOM_MAP: 2 atom_type: OOC

TEMPLATE:: ATOM_MAP: 2 residue1: E

CONSTRAINT:: distanceAB: 2.93 0.2 100.0 1

CONSTRAINT:: angle_A: 127.6 10.0 100.0 360.0

CONSTRAINT:: angle_B: 120.0 10.0 50.0 360.0

CONSTRAINT:: torsion_A: 180.0 10.0 50.0 360.0

CONSTRAINT:: torsion_B: 180.0 180.0 25.0 360.0

CONSTRAINT:: torsion_AB: 0.0 0.0 0.0 180.0

CST::END

- Block 2

#block 2 for pi stacking

CST::BEGIN

#ligand

TEMPLATE:: ATOM_MAP: 1 atom_name: X1 C7 C1

TEMPLATE:: ATOM_MAP: 1 residue3: LG1

#protein

TEMPLATE:: ATOM_MAP: 2 atom_type: aroC,

TEMPLATE:: ATOM_MAP: 2 residue1: WFY

CONSTRAINT:: distanceAB: 4.0 0.2 50.0 0

CONSTRAINT:: angle_A: 90.0 5.0 50.0 360.0

CONSTRAINT:: angle_B: 90.0 5.0 50.0 360.0

CONSTRAINT:: torsion_A: 90.0 5.0 50.0 180.0

CONSTRAINT:: torsion_B: 90.0 15.0 0.0 360.0

CONSTRAINT:: torsion_AB: 0.0 0.0 0.0 60.0

CST::END

- Block 3

#block 3 for hydrogen bond

CST::BEGIN

#ligand

TEMPLATE:: ATOM_MAP: 1 atom_name: O3 N2 C6

TEMPLATE:: ATOM_MAP: 1 residue3: LG1

#protein

TEMPLATE:: ATOM_MAP: 2 atom_type: Nlys

TEMPLATE:: ATOM_MAP: 2 residue1: K

CONSTRAINT:: distanceAB: 2.98 0.2 100.0 0

CONSTRAINT:: angle_A: 103.7 10.0 100.0 360.0

CONSTRAINT:: angle_B: 113.0 10.0 50.0 360.0

CONSTRAINT:: torsion_A: 180.0 10.0 50.0 360.0

CONSTRAINT:: torsion_B: 180.0 120.0 25.0 360.0

CONSTRAINT:: torsion_AB: 0.0 0.0 0.0 15.0

CST::END

The final preparation step is to set up a flags file (flags). In this tutorial we simply use the one given in RosettaMini mini/test/integration/tests/enzdes/flags that you can also find

-s ./1thf.pdb

-extra_res_fa ./LG.params

-enzdes:detect_design_interface

-enzdes:cut1 6.0

-enzdes:cut2 8.0

-enzdes:cut3 10.0

-enzdes:cut4 12.0

-enzdes:cst_opt

-enzdes:cst_design

-enzdes:cst_min

-enzdes:cstfile ./kemp.cst

-enzdes:bb_min

-enzdes:chi_min

-enzdes:final_repack_without_ligand

-out:file:o enz_score.out

-unboundrot ./1thf.pdb

-mute core.io.database

If you want to learn more about EnzDes flags, check out the RosettaCommons site.

So now that everything is prepared we can start an EnzDes run. This is done by calling:

mini/bin/enzyme_design.linuxgccrelease @flags -database minirosetta_database/ -run:constant_seed -nodelay

where minirosetta_database represents the full path to a minirosetta_database.