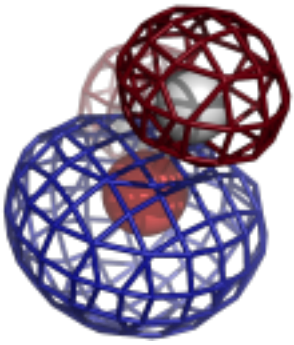
VOLUME THREE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXII

CARBOHYDRATES, SYMMETRY MATES, HYDROGENS

Table of Contents

- PHENIX News 1
- Crystallographic meetings 2
- Expert Advice 3
- FAQ 4
- Short Communications
  - phenix.find_alt_orig_sym_mate 5
  - Using force field generated models in the refinement of crystallographic structures containing carbohydrates 11
  - Viewing diffraction images in CCTBX 14
  - On the contribution of hydrogen atoms to X-ray scattering 18
- Articles
  - CCTBX tools for derivative-free optimization 22

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New releases

phaser.MRage: molecular replacement pipeline

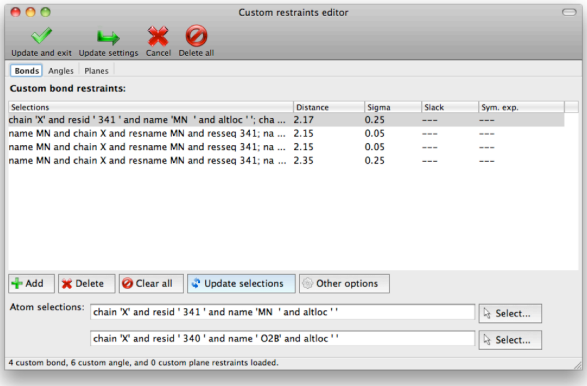
A versatile program for automatic molecular replacement, pronounced mirage, has been added to *PHENIX*. It integrates model search and generation utilities with the molecular replacement program *Phaser*. Search models can be specified in several stages. It is possible to input models that are used as they are provided, or models that are processed by the model editing programs *Sculptor* or *Ensembler*. However, minimal input can take the form of the target sequence (or alternatively, output from a homology search program, such as *BLAST*), in which case the homologues are first fetched from the PDB and then preprocessed. `phaser.MRage` has a novel search organization that can take advantage of clear solutions (identified by high translation function Z-score), and use these for quick evaluation of alternative models, attempt to complete them according to known multimeric structure, and terminate the search early. The parallelisation scheme can spread the search over multiple CPUs or nodes if a queuing system is used (currently SGE, LSF and PBS are supported). The program also employs a novel space group determination algorithm that prunes incorrect space groups incrementally and progresses potential space groups in the search until the correct one is established.

A GUI interface is currently available in alpha release and is planned to be the default in the near future.

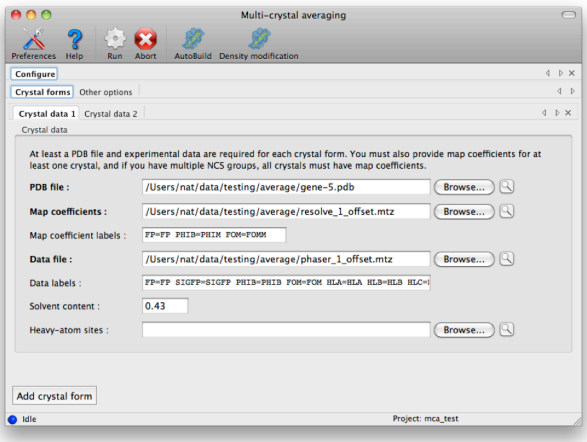
New features

Editing custom restraints

In response to a large volume of user feedback, an improved interface for viewing and editing custom bonds, angles, and planes (e.g. metal coordination parameters from `phenix.ready_set`) has been added to the `phenix.refine` GUI starting with build dev-964. The selection controls are linked to the graphical viewer, which automatically switches to picking a single atom at a time for bond and angle restraints. (Note that you can also clear all unwanted custom restraints at once by selecting "Clear custom restraints" from the Utilities menu.)



New GUIs in the nightly builds include `phenix.morph_model` (improvement & rebuilding of molecule replacement solutions) and `phenix.multi_crystal_average`.



Reference Model Restraints

To improve the success of refinement at low resolution, `phenix.refine` can now take a related model as a reference model. The reference model may be a related, higher resolution model, a theoretical homology model, or other structure with similarity in sequence in fold. The reference model is used to generate torsion restraints on matching torsions in the working model that are parameterized to allow for differences between the working model and reference model automatically. Matches between the reference model and working model are determined automatically, including flexibility for different numbers of NCS copies. Details may be found in a forthcoming issue of *Acta Crystallographica* Section D, in an article by Headd *et al.*, entitled '*Use of knowledge-based restraints in phenix.refine to improve macromolecular refinement at low resolution*'.

eLBOW and Mogul

eLBOW can generate geometries with internal coordinates taken from experimental geometries taken from the Cambridge Structure Database (CSD) using the *Mogul*[1] command file interface If you have *Mogul* from the Cambridge Crystallographic Data Centre, installed and accessible at the command-line, *eLBOW* can request that *Mogul* provide an experimentally accurate geometry. Simply use the `--mogul` option at the command-line and *eLBOW* will query *Mogul* for the geometry to use to generate a restraints file and a PDB file.

Furthermore, you can validate a geometry using *Mogul* in *eLBOW*. This useful in testing the validity of geometries calculated by a quantum chemical method. Adding the option `--validate` results in the output of an additional file containing validation results.

In addition, an XML file can be output using the `--xml` option. Information about the molecule is output in a simple XML format.

[1]http://www.ccdc.cam.ac.uk/products/csd_system/mogul

More tags and values will be added in the future.

phenix.ligand_identification

Methods have been implemented to generate custom ligand library based on parent protein's sequence and/or structure information using SCOP terms, CATH terms, Pfam accession numbers, GO accession numbers, or InterPro ID.

Crystallographic meetings and workshops

CCP4 School on Advanced X-ray crystal structure analysis, Australian Synchrotron, Melbourne, Australia, February 12-15, 2012

Paul Adams will be lecturing and giving tutorials throughout the meeting at the Australian Synchrotron.

PHENIX User's Workshop, University of Texas at Austin, Austin, TX, February 22, 2012

A *PHENIX* user's workshop is being planned in Austin, Texas on the 22nd of February for local area students, postdocs and other interested parties.

Advanced Course in Protein Crystallography. PHENIX State-of-the-art Software for Protein Structure Determination, National Autonomous University of Mexico, Mexico City, Mexico April 18-20, 2012

PHENIX developers will be giving lectures and available for questions in Mexico City April 18-20.

Gordon Research Conference on Diffraction Methods in Structural Biology, Bates College, Lewiston, ME, July 15-20, 2012

Jeff Headd will be presenting at the Gordon Conference on Diffraction Methods in the "Getting the Best Out of Your Data: Data Analysis" section.

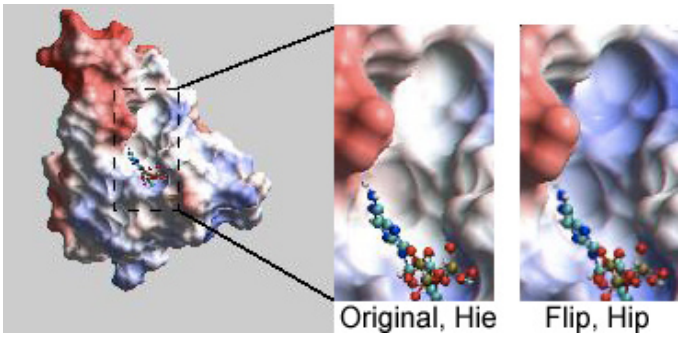


Figure 1: PDB entry `1mjh` showing electrostatic surface near `His40`.

Expert advice

Fitting Tips

Jane Richardson and Michael Prisant, Duke University

Some choices in model fitting make only negligible differences in the R factors or density matches even at good resolution, but may matter a great deal to end-users of a crystal structure. Of special significance in this category are 180° $\chi 2$ "flips" of histidine rings, which are frequently of functional importance. A `His` flip always changes hydrogen bonding, and over half the time changes assignment of the `His` protonation (H on Nd = "`Hid`", on Ne = "`Hie`", or on both = "`Hip`" with +1 charge). This crystallographically trivial difference can profoundly alter the inferred interaction properties of an active site, as in the electrostatic surface shown in figure 1, for `His40` in PDB entry `1mjh`.

`His` ring (and `Asn/Gln` sidechain amide) orientation is not well defined by electron density, since the N vs C (or N vs O) scattering is very similar and normal crystallographic refinement does not explore flips of 180° in the plane of the density. However, quite reliable assignment of flip orientation and `His` protonation can be made automatically (at resolutions better than about 2.7Å) by comparative evaluation of H-bonds and all-atom steric clashes. This is done by Reduce, in either MolProbity or PHENIX.

Figure 2 shows before-and-after states of

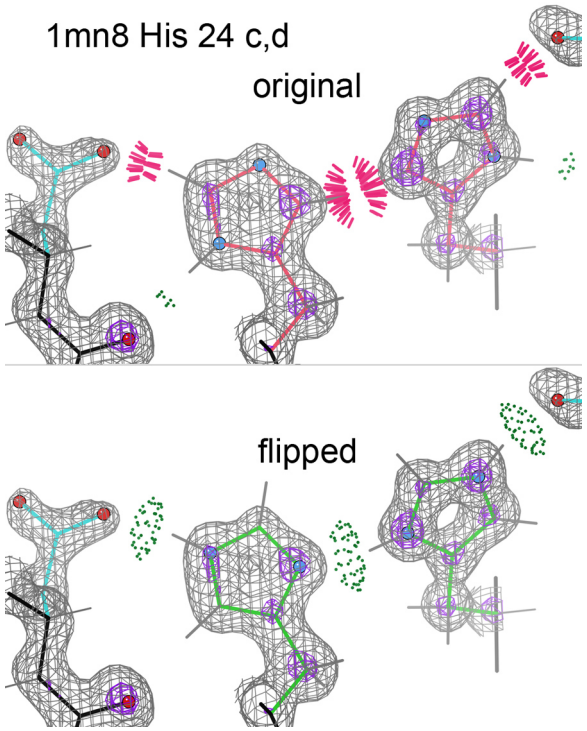


Figure 2: Improvements after flipping a residue.

another case where model-building and refinement gave the wrong answer in the deposited structure, even at 1.0Å resolution: this `Asp-His-His-Asp` H-bond network at a pseudo-2-fold interface shows 3 huge steric clashes (red spikes) and only weak H-bonds (green dots), with Nd protonation. Flipping the His rings gives 3 strong H-bonds and no clashes, with asymmetrical protonation of Ne on one chain and double protonation (`Hip`, +1 charged) on the other. Note that the clear, well-ordered 2Fo-mFc density places the ring atoms accurately, and their positions tell us that the molecule prefers His H-bonding with the `Asp` O's over H-bonding with the backbone carbonyl oxygens: the H-O distances are 1.9Å to Asp vs 2.3Å to carbonyl oxygen. At this resolution one can also see the flip assignment confirmed by higher 4σ (purple) peaks for the N atoms in the flipped state, even before re-refinement.

The best strategy is to fit the ring plane manually or by automation that involves real-space refinement, followed by Reduce to determine flip and protonation. To add in your own scientific judgment, especially important at low resolution or extreme pH, do the Reduce run on the MolProbity site, which produces a "hisflip" kinemage that animates between flip states, with a preset view for each His and all the contact evidence shown explicitly. To see the density also (as in the figure above), either upload a map to MolProbity for on-line viewing or download the hisflip kinemage and view off-line in KiNG. If you disagree with a given flip, you can generate an output PDB file without that flip.

This type of issue can be vital when end-users want to infer chemical behavior from your deposited structure. For instance, relaxation of ring flips is difficult and protonation changes are impossible in the context of standard molecular dynamics.

To try out these His flip examples, run `1mjh` or `1mn8` on the MolProbity web site.

FAQ

What Molprobity clashscore is acceptable?

Some key validation criteria are listed in table 1 of the article by R. J. Read *et al.* entitled "*A New Generation of Crystallographic Validation Tools for the Protein Data Bank*" in Structure, 19, 1395-1412 published in 2011. The reader should read the article but a clashscore of better than 5 is considered ideal. When the PDB implements the new validation, percentile scores will be available for most criteria; in general, it's doable and worthwhile to achieve percentiles in the 90's (at least up to resolutions around 2.5Å), but it is not a good idea to try for 100% because there are usually a few real or intractable oddities.

Contributors

P. D. Adams, P. V. Afonine, G. Bunkóczi, G. Chen, N. Echols, B. L. Foley, R. J. Gildea, J. Hattne, J. J. Headd, R. W. Grosse-Kunstleve, H. Liu, N. W. Moriarty, R. D. Oeffner, B. Poon, M. Prisant, R. J. Read, J. S. Richardson, N. K. Sauter

phenix.find_alt_orig_sym_mate

Robert D. Oeffner, Gábor Bunkóczi and Randy J. Read[a]

[a]University of Cambridge, Department of Haematology, Cambridge Institute for Medical Research, Cambridge, CB2 0XY, UK

Correspondence email: rdo20@cam.ac.uk

`phenix.find_alt_orig_sym_mate` is a python script that allows the user to determine whether two different molecular replacement solutions in PDB files for the same dataset are equivalent to one another. It assumes that the PDB files consist of one chain only and that they represent the same component in the structure. Importantly, the two models need to be homologous but not identical to each other. All alternative origins and symmetry operations defined by the spacegroup are taken into consideration. The core part of the algorithm is using procedures outlined by Petrus Zwart and Nathaniel Echols based on the CCTBX (Große-Kunstleve, 1999) and the SSM algorithm (Krissinel, 2004).

Brief introduction

The choice of what point in a unit cell can serve as an origin is not unique, but is constrained by the symmetry operations of the space group. For instance, any point in the unit cell of a P 1 crystal can be chosen as an origin, but higher-symmetry space groups will have fewer allowed origins. For simplicity consider a two-dimensional crystal with a two-fold rotation axis at the origin of the unit cell as shown in figure 1. The combination of this two-fold with unit cell translations leads to further two-folds being generated at each corner of the unit cell as well as on the faces of the unit cell and in its centre. This in turn yields four different alternative origins coincident with a two-fold axis: (0, 0), (0, ½), (½ ,0) and (½, ½). Solving the structure relative to each of these origins yields equally valid solutions. But it is cumbersome for the user to work out whether two solutions of the same dataset actually match one another.

`phenix.find_alt_orig_sym_mate` will inspect if the structures in two PDB files derived from the same set of structure factors match one another irrespective of alternative origin and symmetry operations.

Procedure

Given a pair of PDB files that are molecular replacement solutions for the same dataset `phenix.find_alt_orig_sym_mate` will use one as a reference structure and the other as the moving structure.

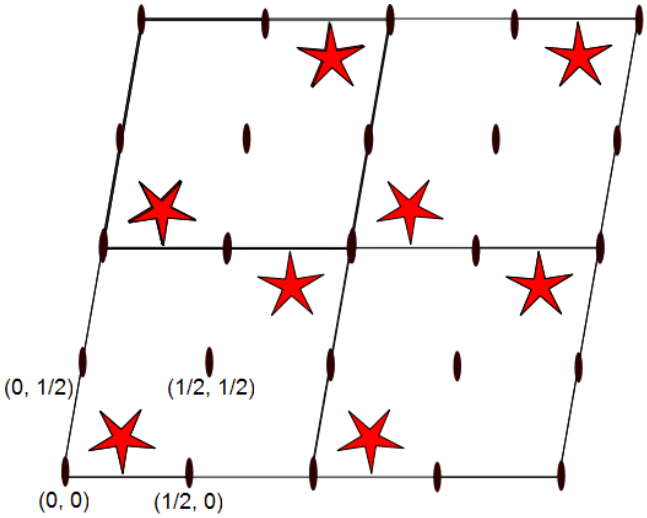


Figure 1: A two-dimensional crystal with a two-fold axis, symmetry generated copies as well as alternative origins.

It will then compute the SSM alignment between the reference structure and the moving structure. This produces a list of pairs of equivalent Cα atoms in the two structures. No coordinate transformation is applied however at this point.

Then `phenix.find_alt_orig_sym_mate` loops over all possible symmetry copies on alternative origins of the moving structure with respect to the spacegroup and cell dimensions as specified in the CRYST1 record in the PDB file of the moving structure. These copies of the moving structure are computed programmatically with the aid of the CCTBX. For each of these copies a unit-less score

value, termed MLAD, is calculated. The symmetry copy on an alternative origin that has the smallest MLAD score is selected as the best possible match between the two structures.

Finally a copy of the moving structure associated with the smallest MLAD score is saved as a PDB file together with a log file.

MLAD

The score value MLAD is an acronym for the *mean log absolute deviation* that is calculated for pairs of Cα atoms as follows:

$$\text{MLAD}(\text{atom pairs}) = \frac{1}{\#\text{atom pairs}} \sum_{i=1}^{\text{atom pairs}} \log\left[\frac{|\Delta r_i|^2}{|\Delta r_{i,x}| + |\Delta r_{i,y}| + |\Delta r_{i,z}| + 0.9} + \max\left(0.9, \min(|\Delta r_i|^2, 1)\right)\right]$$

This is unlike the commonly known RMSD defined as:

$$\text{RMSD}(\text{atom pairs}) = \sqrt{\frac{\sum_{i=1}^{\text{atom pairs}} |\Delta r_i|^2}{\#\text{atom pairs}}}$$

That we employ MLAD rather than RMSD is justified as follows: Consider an alignment of ten pairs of atoms where a translation along the a-axis will perfectly superpose atoms in seven of the pairs or atoms in three of the pairs as illustrated in figure 2.

For a unit-less spacing of, say a=10, we can calculate the RMSD and MLAD when sliding the group of red atoms along the a-axis. This yields the graphs in figure 3.

We note that MLAD captures the minimum of superposing the seven pairs of atoms correctly and also reveals a local minimum for the three pairs of atoms. The RMSD on the other hand settles for an intermediate position that does not match the perfect superposition of neither the seven nor the three pairs of atoms. One might argue that a simpler function could have been used than MLAD, say

$$\text{MAD}(\text{atom pairs}) = \frac{\sum_{i=1}^{\text{atom pairs}} |\Delta r_i|}{\#\text{atom pairs}},$$

which will indeed capture the same minimum. However, an extensive number of calculations of MLAD, MAD and RMSD scores have shown that this function whilst superior to RMSD still fails to capture some of the good superpositions that the MLAD score correctly identifies. The superiority of MLAD over MAD is due to the log function that down-weights the scores of alignments of atoms that are

Figure 2: Ten blue atoms aligned with ten red atoms. Three red atoms are perfectly superposed on three blue atoms.

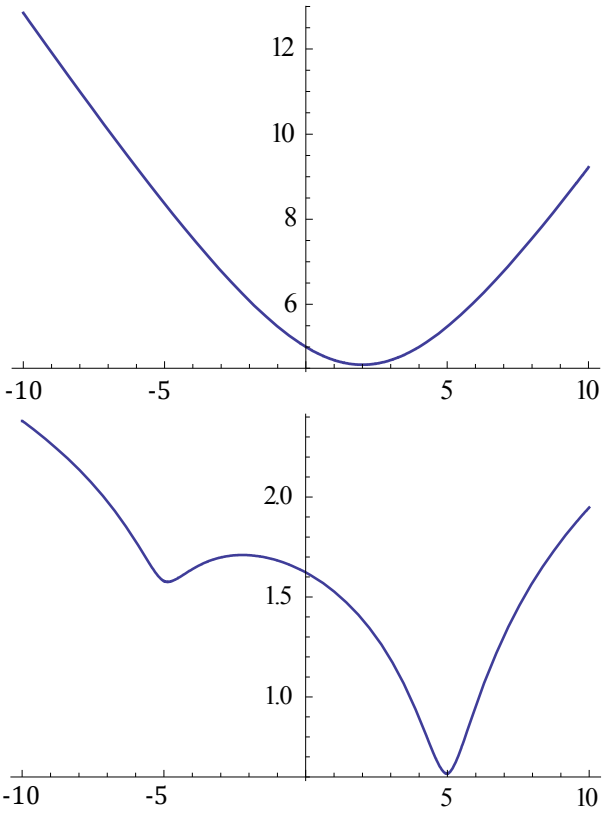


Figure 3: (Upper) The RMSD values of the hypothetical ten red and blue atoms in figure 2 as a function of sliding along the a-axis. (Lower) The MLAD values of the hypothetical ten red and blue atoms in figure 2 as a function of sliding along the a-axis.

spatially far apart. Such atoms are considered as outliers to be ignored whenever it is possible to superpose a substantial fraction of other atoms well on one another. In theory this could lead to local minima of very short alignments being identified as the best alignment for two structures. But we have not encountered this so far.

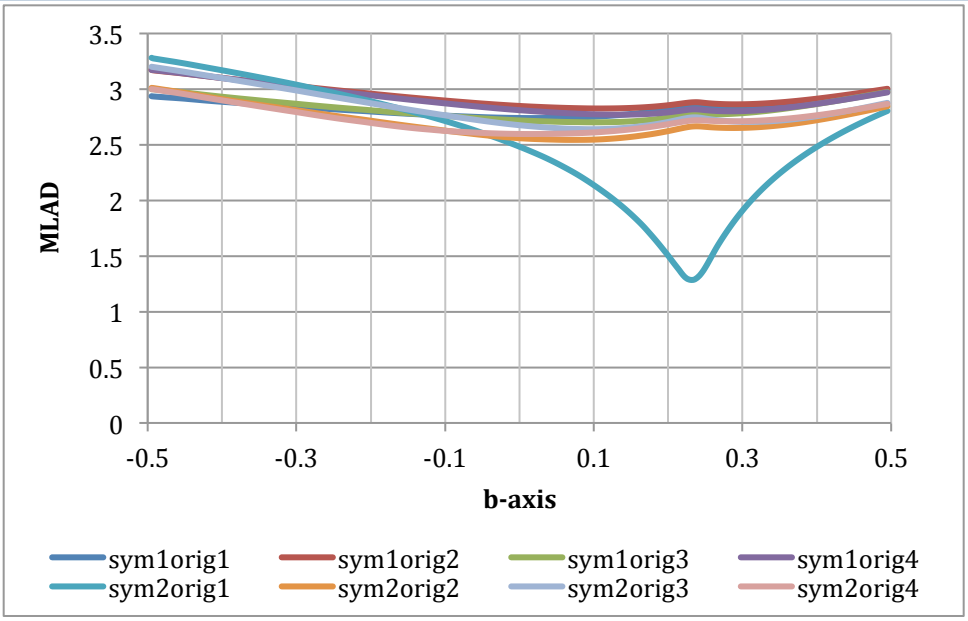


Figure 4: MLAD values between the structures with PDB code 1CM3 and a molecular replacement solution with a model derived from 1Y50 as a function of polar axis position in fractional coordinates for the eight combinations of symmetry transformation and choice of origin.

Floating origin

Datasets with a space group that have a polar axis comprises about 25% of all the structures in the PDB. The algorithm treats such cases slightly differently than space groups with fixed origins: For each symmetry copy on an alternative origin `phenix.find_alt_orig_sym_mate` will do a line minimization of the MLAD score along the polar axis. The minimiser employed is the golden section search from the CCTBX that is fast and requires typically no more than 34 steps. When the minimiser returns with the best MLAD score for that symmetry copy on an alternative origin the MLAD score of this symmetry copy is compared to score values of all the other symmetry copies. Eventually the best MLAD score with the corresponding symmetry copy on an alternative origin is found.

Examples

To illustrate the efficacy of the algorithm we show graphs (figure 4) of MLAD score values for comparing PDB entry `1CM3` with the molecular replacement solution for the same dataset but with the model derived from PDB entry `1Y50`. The space group is P 1 21 1 implying that the b-axis of the unit cell is a polar axis. Using the `--debug` keyword `phenix.find_alt_orig_sym_mate` will include a table of MLAD scores for each symmetry copy on an alternative origin as it is moved along the polar axis. As this space group has 2 symmetry operations and 4 alternative (floating) origins the algorithm will calculate the MLAD scores along the polar axis for eight combinations of symmetry transformation and choice of origin.

The symmetry copy with the minimum MLAD value of 1.2842 is clearly identified as the relatively steep trench for the graph of the second symmetry operation, (-x, y+1/2,-z), on the first alternative origin, (0, y, 1/2) with the legend sym2orig1. This is at the position 0.2323 fractional cell units along the b-axis. We note how the graphs of MLAD values for the other symmetry copies on alternative origins feature no minimum near the same location let alone with a similar depth or curvature. This is the typical topology for MLAD values as a function of symmetry operations on alternative origins. In figure 5 we show how the MR solution with `1Y50` superposes fairly well onto this symmetry copy of the structure `1CM3`.

This is in stark contrast to what would have

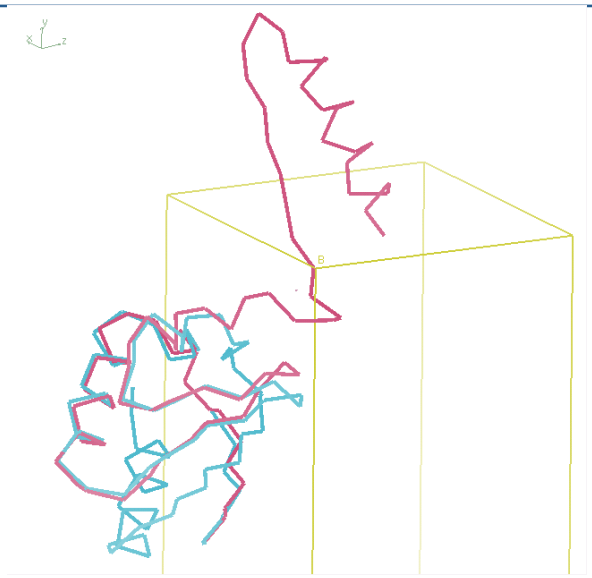


Figure 5: Cα-trace of the structures with PDB code 1CM3 (red) and a molecular replacement solution with a model derived from 1Y50 (blue) where the position along the polar axis was found to yield the smallest MLAD score among all possible symmetry copies on alternative origin for this crystal.

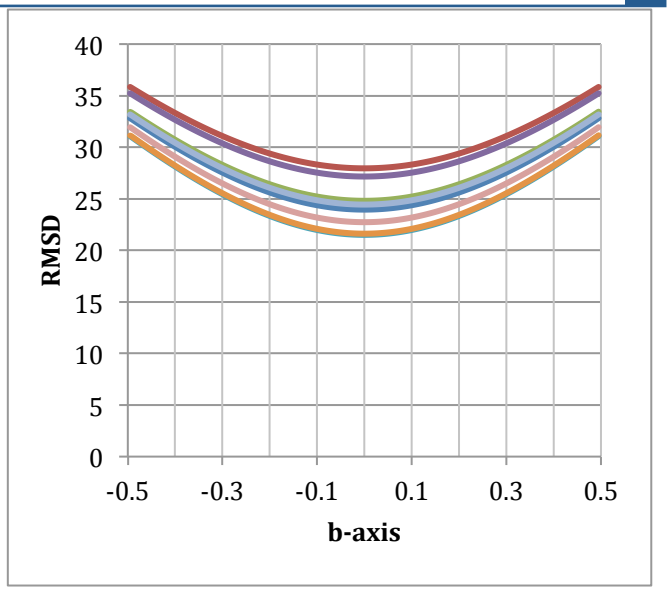


Figure 6: RMSD values between the structures with PDB code 1CM3 and a molecular replacement solution with a model derived from 1Y50 as a function of polar axis position in fractional coordinates for the eight combinations of symmetry transformation and choice of origin. Same legend as figure 4.

happened had we used the classical RMSD as a score indicator of the best superposition of a symmetry copy on an alternative origin with another structure.

In this case figure 6 shows that the correct symmetry operation on alternative origin, sym2orig1, has not been identified as having the smallest RMSD value, let alone identified its correct translation along the polar axis. In figure 7 we show how the MR solution with `1Y50` fails to superpose onto this symmetry copy of the structure `1CM3` when `phenix.find_alt_orig_sym_mate` is using RMSD rather than MLAD to predict the best symmetry copy of the structure `1CM3`.

This illustrates that RMSD only predicts the spatial difference between the centroids of the molecules.

General test cases

To present the user with some guidance for what is a good MLAD score we have used `phenix.find_alt_orig_sym_mate` on 520 random target structures of MR calculations and model structures. These vary from being totally incorrect solutions to being very good solutions. A measure of "correctness" of an MR solution is the correlation coefficient between the electron density of the target structure and the electron density of the MR

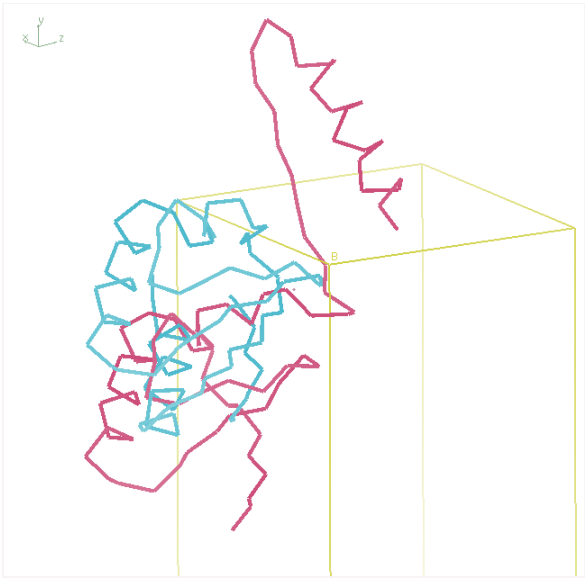


Figure 7: Cα-traces of the structures with PDB code 1CM3 (red) and a molecular replacement solution with a model derived from 1Y50 (blue) where the position along the polar axis was found to yield the smallest RMSD score among all possible symmetry copies on alternative origin for this crystal.

solution. An electron density is readily computed by using a structure together with the associated dataset in a zero cycle rigid-body refinement. Subsequently we used `phenix.get_cc_mtz_mtz` to calculate the correlation coefficient. We have computed correlation coefficients for these solutions and present them below with their associated MLAD scores (see figure 8).

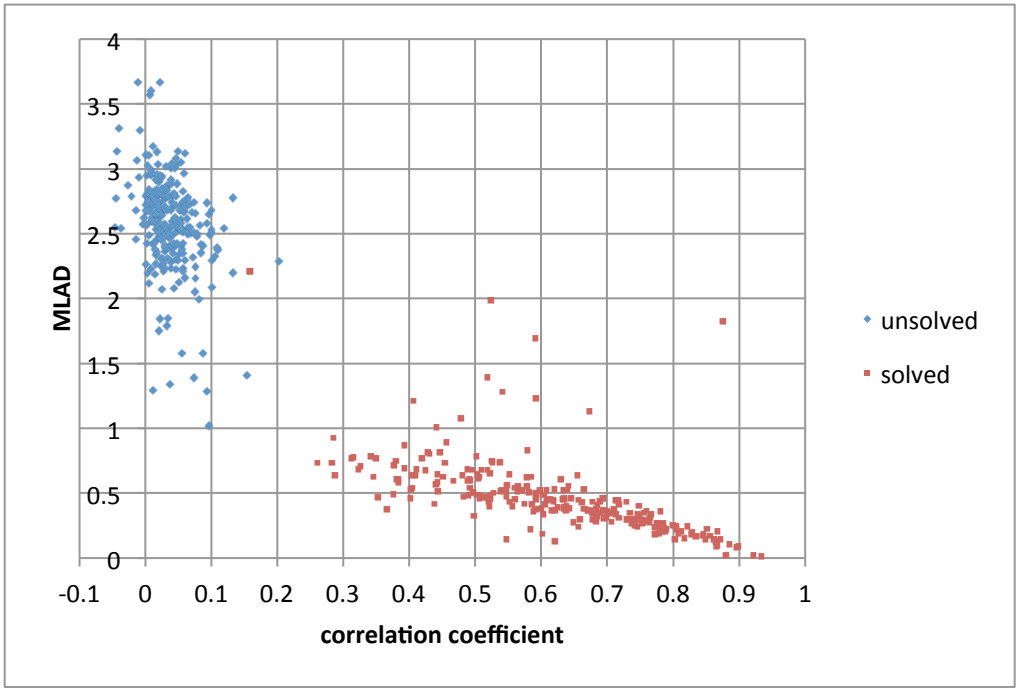


Figure 8: Map correlation coefficients and the corresponding MLAD values for 520 random molecular replacement calculations between target structures and their solutions.

We note that for correct solutions MLAD values are typically below 1.5 and for incorrect solutions MLAD values are typically above 2. This is a rule of thumb as exceptions do occur. This might be if SSM fails to identify an adequate alignment between the moving structure and the reference structure. In any case a visual inspection of the PDB file produced by `phenix.find_alt_orig_sym_mate` together with the reference structure will show if the structures have been superposed correctly.

Usage

The program can be invoked from the command line with the PDB files specified explicitly:

```
phenix.find_alt_orig_sym_mate --moving_pdb=1oi1.pdb --reference_pdb=mrsol.pdb.
```

Using the solution file from Phaser together with the model is also an option:

```
phenix.find_alt_orig_sym_mate --moving_pdb=1ahn.pdb \
                          --model_pdb=sculpted_1OBV_A.pdb \
                          --use_solution=1OBV_A.sol.
```

Within PHENIX a pickle file specifying the molecular replacement solution for the model specified in a phil file may also be submitted:

```
phenix.find_alt_orig_sym_mate --moving_pdb=1ahn.pdb \
                          --use_pickle=phaser_mr_1.pkl \
                          --use_phil=job_1.phil.
```

The last two commands will prompt `phenix.find_alt_orig_sym_mate` to generate intermediate PDB files of the specified models placed and oriented according to the solutions in the .sol or the `.pkl` file. `phenix.find_alt_orig_sym_mate` will then find the best symmetry mate on an alternative origin for each of the solutions. Additional flags are the `--only_use_origin`, `--debug` and the `--no_symmetry_operations`, keywords. For instance

```
phenix.find_alt_orig_sym_mate --moving_pdb=1oi1.pdb \
                          --reference_pdb=mrsol.pdb \
                          --only_use_origin="2/3, 1/3, 1/2" \
                          --debug
```

would try and match `1oi1.pdb` on `mrsol.pdb` offset at the alternative origin (2/3, 1/3, 1/2).

The `--debug` flag saves pdb files at each of the symmetry operations of the Cα atoms taking part in the SSM alignment of the moving structure and also the one file of Cα atoms taking part in the SSM alignment of the reference structure. It also saves files with all symmetry copies on alternative origins of the moving structure dumped into one pdb file but labelled with different chain ID. Depending on the number of operations several pdb files may be generated.

In the rare instance when `phenix.find_alt_orig_sym_mate` fails a remedy might be to specify the `--use_all_SSM` flag. The SSM algorithm will then attempt to locate other alignments between the two structures which will be used in the subsequent MLAD scoring. It is helpful to inspect the alignments generated by the `--debug` keyword to see if SSM really did get it wrong. If not, then the molecular replacement solutions being tested are likely to be unrelated.

References

Große-Kunstleve, R. W. (1999). *Algorithms for deriving crystallographic space-group information.* Acta Cryst., A55, 383-395.

Krissinel, K. E. (2004). *Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions*. Acta Cryst., D60, 2256-2268.

Using force field generated models in the refinement of crystallographic structures containing carbohydrates

B. Lachele Foley

Complex Carbohydrate Research Center, The University of Georgia, Athens, GA USA

Email: lfoley@uga.edu

It is difficult to ensure proper assignment of carbohydrate structures using the widely available crystallographic methods. Some evidence for this is presented here, but there are other sources. For example, in 2004, Lütteke, Frank and von der Lieth reported (Lütteke, 2004) that about 30% of carbohydrate-containing entries in the PDB contained one or more errors. They considered only the most obvious errors: incorrect naming, incorrect bonding and obvious omissions or insertions of atoms. Had they also considered, for example, structures whose geometries are energetically very unlikely, they would have found more. Persons wishing to investigate these claims and other aspects of carbohydrate-containing structures in the PDB might find useful the tools available at http://glycosciences.de, a web site begun by the authors of the aforementioned study.

Misidentifications occur mainly because carbohydrate structures are complex and their nomenclature is more historically-based than systematic. Names such as "mannose" and "glucose" offer no clues regarding their actual structures except via memorization (Figure 1). Recognition is further complicated by the presence of mirror images, alternate ring conformations, changes in substituents, etc. For example, particularly within a much larger system, one might accidentally count an oxygen atom as "up" when it is actually "down". Highly distorted structures can exacerbate identification as well since visual clues normally considered, such as axial or equatorial orientation, might no longer be relevant.

The assignment of unrealistic geometries occurs when the force field lacks sufficient

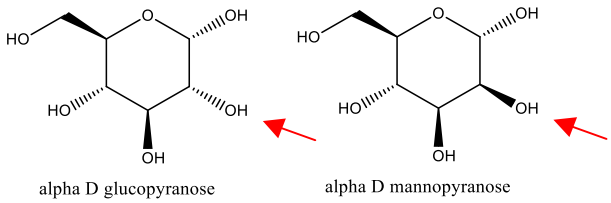


Figure 1. The position of a single hydroxyl group differentiates between α-D-glucopyranose and α-D-mannopyranose.

internal information about the residues it treats. Two examples from the PDB are shown in Figure 2. In both cases, the assigned geometries are very high energy structures. While high-energy geometries might occasionally be appropriate, they should be strongly defended. Here, the energies due to the highlighted anomalies are about 20 (Figure 2-A) and 10 (Figure 2-B) kcal/mol above the energies for the expected geometries, making the structures extremely unlikely. In each of these cases, the carbohydrate was present within a much larger and more complex structure, but was not itself of direct interest to the study (Gamblin, 2004 and Huntingdon, 2003). Although these represent only two anecdotes, situations such as this might explain why unlikely structures seem so readily to pass the notice of researchers and reviewers alike. Whatever the cause, the researchers would certainly have benefited from software better able to differentiate reasonable structures from less likely ones.

Any force field used for proteins almost certainly contains information regarding the proper geometry for an N-acetyl group such as that in Figure 2-A. However, the atoms were not recognized outside their usual context. The example in Figure 2-B illustrates lack of attention to a more subtle situation of great importance to carbohydrate structure, the exo-anomeric effect. There are many

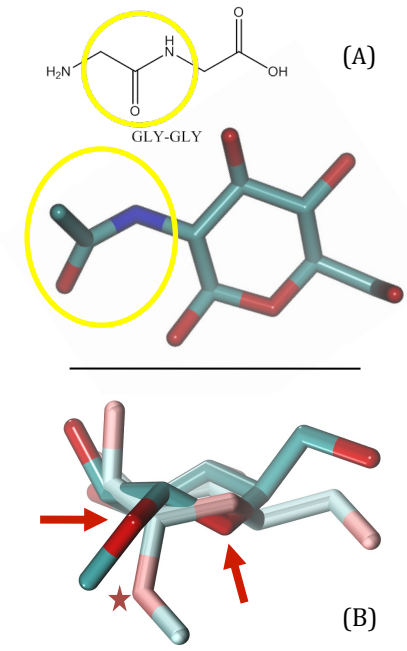


Figure 2. Two examples of unrealistic geometries assigned to carbohydrates in the PDB. In the molecular models, hydrogen atoms are omitted, carbon is cyan, oxygen is red and nitrogen is blue. (A) β-D-N-acetyl glucosamine, residue 3321(NAG) in chain A of PDB ID 1RVZ. The bend at the nitrogen atom breaks the expected planarity. The atoms attached to that nitrogen should lie in the same plane as the atoms attached to the carbon bonded to the left. The yellow circles highlight the chemical equivalence to a peptide-peptide bond, well known to prefer planar geometry. (B) α-D-mannopyranose, residue 6 (MAN) in PDB ID 1LQ8 (bold) superimposed on a lower energy conformation generated using the Glycam_06g force field (pastel). The PDB structure violates the geometry expected on the basis of the exo-anomeric effect. Red arrows point to two oxygen atoms whose regions of enhanced electrostatic effect (often called lone pairs) are in close proximity. In other words, the "elbows" at the oxygen atoms point nearly the same direction and are only one atom removed from each other. In contrast, the orientation of the glycosidic oxygen (star) in the Glycam_06g model better separates the repulsive regions. The ring in the PDB structure is also in a relatively high energy configuration, likely distorted by the software during fitting.

papers describing the effect, and a number of them are referenced within a recent review (Foley, 2011) that also contains a brief discussion of the effect. At its simplest, the effect is due to repulsions between regions traditionally represented as lone pairs. These effects are rarely modeled well by force fields unless they have been specifically designed to do so. Of course, there are other subtleties, such as preferred ring geometry, which are also better handled by a more appropriate force field.

In almost all cases, the use of models generated by force fields designed for molecular modeling could significantly increase the chances that structures in crystallographically generated models are properly identified and are in realistic geometries. Force fields designed for molecular modeling are more advantageous in these circumstances because they describe structures in greater detail than do the current force fields designed for crystallography. In the case of carbohydrates, internal interactions such as the exo-anomeric effect (Figure 2b) are not easily treated without such a detailed description. Even if structures continue to be initially determined using traditional crystallographic force fields, the molecular modeling force fields could be used to discard highly unlikely geometries.

While it might not immediately be apparent that a force field could assist with identification, some force fields can. In the Glycam force fields (Foley, 2011 and Kirschner, 2008), for example, charges are defined at the residue level, as are reasonable initial residue geometries. If a force field such as this is employed, then proposed structures can be compared to the force field's corresponding template. If there is a mismatch, then either the proposed structure is wrong, or it has been named incorrectly. The challenge in this is to map the contents of a file to a template when the file's internal naming, atom ordering, etc., might differ greatly from that of the template. There is also the problem of identifying structures that do not already have templates within the force field. Here, the force field might be used as a guide, but a more advanced recognition algorithm would be required.

Most other geometrical concerns can also be treated with proper application of a force field. A comparison between energies of the two structures shown in Figure 2 and their force field generated counterparts would have shown them to be unlikely. A force field with

residue templates could also be used to quickly identify structures with unexpected bonding, perhaps where the CONECT cards had been written incorrectly. Even a molecular dynamics force field without templates could identify many of these issues.

References

B. L. Foley, et al. *WIREs Comput Mol Sci* (2011) doi: 10.1002/wcms.89
S. J. Gamblin et al. *Science* 303 (2004) 1838-1842
J. A. Huntingdon, et al. *Structure* 11 (2003) 205-215
K. N. Kirschner, et al. *J. Computational Chemistry* 29 (2008) 622-655
T. Lütteke et al. *Carbohydrate Research* 339 (2004) 1015–1020

Viewing diffraction images in CCTBX

Nathaniel Echols, Johan Hattne, Richard J. Gildea, Paul D. Adams, and Nicholas K. Sauter
Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Correspondence email: NKSauter@LBL.Gov

Introduction

To facilitate ongoing efforts to analyze and process problematic diffraction images, we have developed a simple, embeddable application in CCTBX (Grosse-Kunstleve *et al.,* 2002) to view a wide variety of detector formats using the wxPython library (http://www.wxpython.org). In the past, *LABELIT* (Sauter 2011) was capable of generating image files containing annotated sections of the diffraction pattern, but this was limited to non-interactive use. Debugging of indexing and integration code necessitated a more robust and configurable interface capable of displaying images along with detailed visual feedback on the performance of the processing algorithms.

Program description

The standalone program can be run as *phenix.image_viewer* (also available in the PHENIX GUI under "Utilities"), and supports most common file formats (including ADSC, MAR, RAXIS, and CBF). The interface (Figure 1) is similar to the versatile X11-based image viewer *adxv* (http://www.scripps.edu/~arvai/adxv.html) in many respects, although we have diverged from this design where appropriate. (In particular, the functions of the left and middle mouse buttons have been swapped to simplify use on Macintosh laptops.) The image display itself occupies most of the main window, with view controls in a separate floating window. To aid navigation of large images at high magnification, the control window incorporates a thumbnail view of the overall image; clicking in this view recenters the main viewport. The thumbnail image incorporates a feature previously implemented in *LABELIT*: when sampling the raw image to produce the thumbnail, multiple image pixels are polled to produce each thumbnail pixel, with the highest intensity value being taken instead of the average value. This results in a much higher contrast that allows the lattice diffraction to be clearly visible in the thumbnail even for relatively weak images (Figure 1).

Internally, each detector image is essentially an integer array indexable by X,Y coordinates. This is converted to an ASCII string representing the image in a generic format, with the desired brightness and color scheme applied in place of the raw intensity values. The conversion is applied only when these settings are modified; scaling and panning is performed entirely by the built-in image manipulation functions in wxPython. Actual display is accomplished by rendering the appropriate section of the image as a bitmap. Although this method is less ideal at lower magnification due to the compression of spots and resulting loss of contrast, it allows interactive use to remain extremely fast. The zoom panel also allows display of individual intensity values superposed on the pixels (Figure 1), similar to features in *adxv* and *XDisplayF* (Otwinowski & Minor, 1997).

The viewer can easily be embedded in other applications, either as part of a larger interface or at the end of a command-line process. The "paint" method of the display panel is designed to allow extension with custom draw commands using the wxPython API. For instance, the following simple code overlays yellow circles around integrated spots:

```
def draw_spot_predictions (window, dc, image) :
  import wx
  scale = window.get_scale()
  predictions = image.get_drawable_predictions()
  dc.SetPen(wx.Pen((255,255,0), 1))
  dc.SetBrush(wx.TRANSPARENT_BRUSH)
  for (x, y) in predictions :
    dc.DrawCircle(x, y, 8*scale)
```

A slightly more complex real-world example is shown in Figure 2. The main limitation to these overlays is the number of function calls that can be performed before the interface slows noticeably. In practice, the programmer must incorporate information about the current scale and drawable area of the image, and simplify or clip the image annotations where appropriate. The Python class used to manage and convert image data provides a convenience method, get_drawable_points, that accepts as input a list of

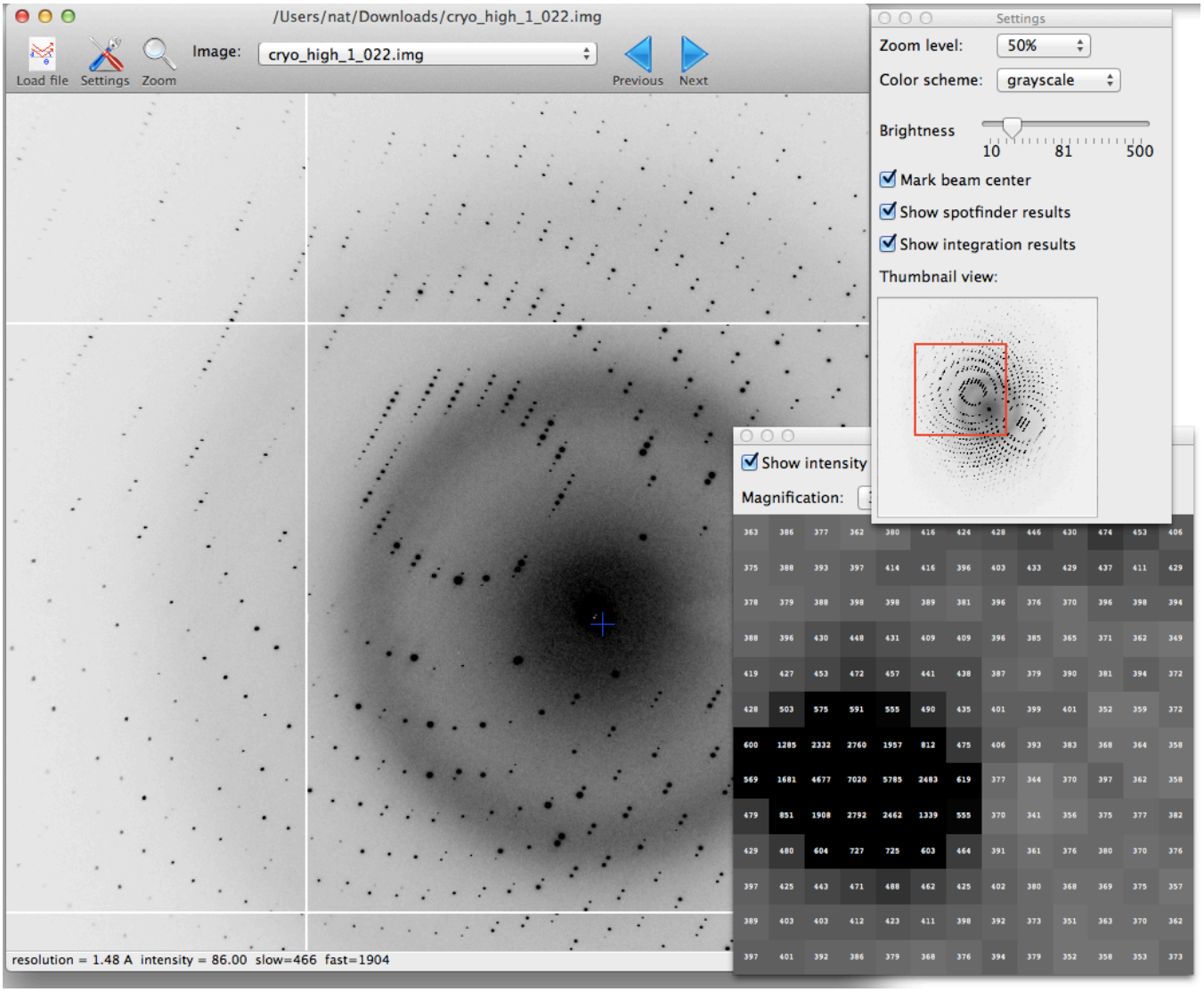


Figure 1: The main image viewer interface, showing a diffraction pattern from *E. coli* dihydrofolate reductase (provided by James Fraser, UCSF). The zoom window is visible at lower right.

x,y coordinates (*i.e.* detector pixels) and filters out those which occur outside the current view area.

The program *distl.image_viewer* (Figure 3) provides a simple example of how the module may be quickly re-used as a driver for additional computation. The settings panel is modified to control the applicaton *distl.signal_strength*, a spot-finding tool designed for rapid evaluation of crystal diffraction (Sauter, 2010). Spot detection is triggered by wxPython events, such as changing the "minimum spot area" field, and the interface is redrawn with the new predictions. The subclassed interface consists of less than 200 lines of Python code (rstbx/viewer/spotfinder_frame.py), most of which implements the additional controls and

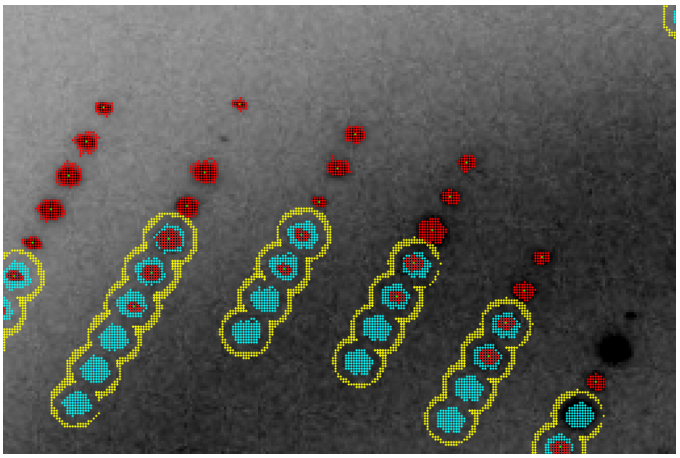


Figure 2: Closeup of the diffraction pattern shown in Figure 1 with the spotfinder results (red), integration masks (cyan), and background masks (yellow) overlaid.

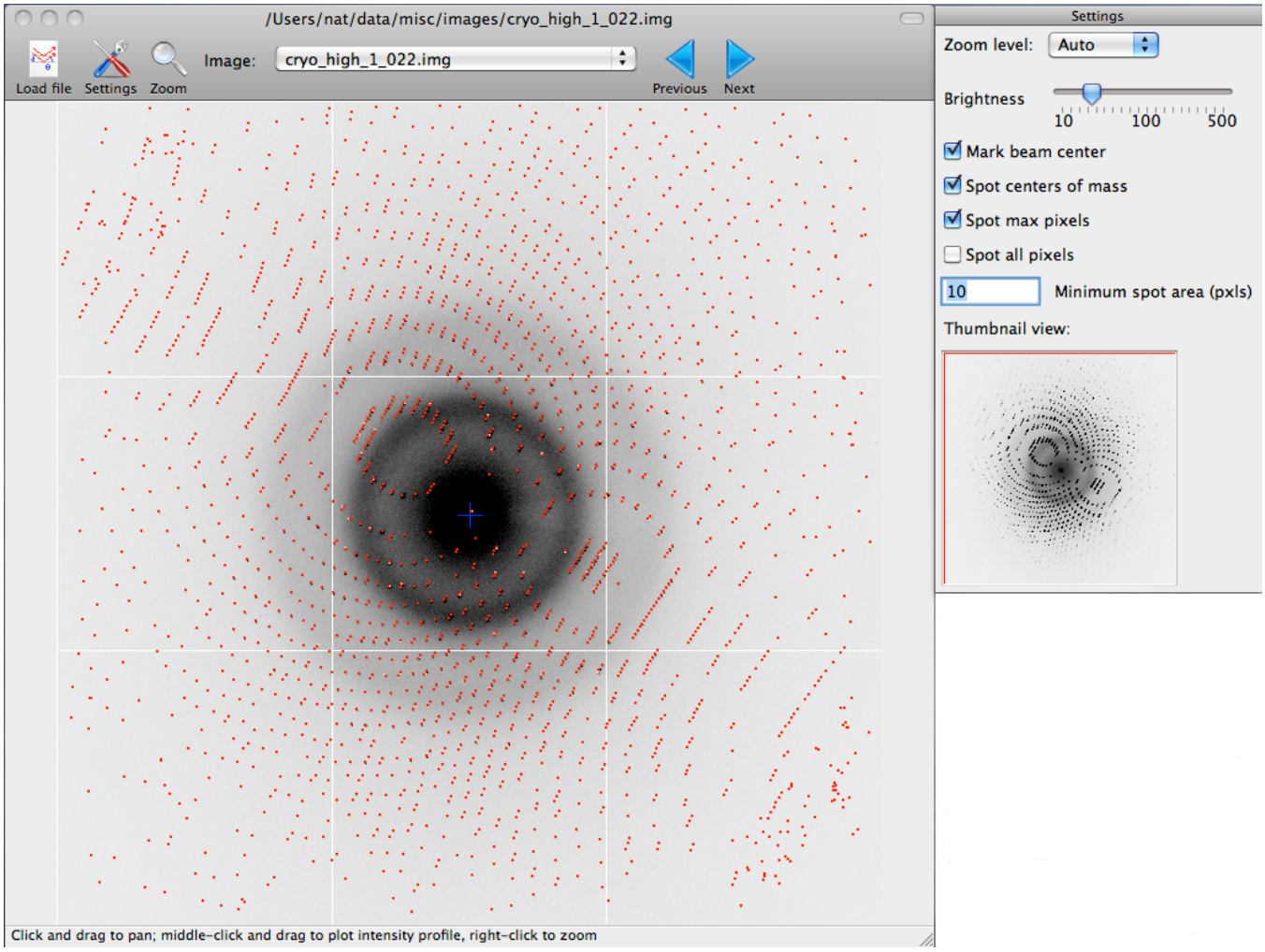


Figure 3: The spotfinder front end, *distl.image_viewer*.

event handling.

Future directions

We extensively use this module as a development aid. An NIH-funded project, "Realizing New Horizons in X-ray Crystallography Data Processing", is focused on problems in diffraction image processing that can not be handled automatically (or at all) by current software, such as split or multiple lattices (Sauter & Poon, 2010), severe anisotropy, and other common pathologies. Although the eventual goal is a suite of programs that can be run unattended (either graphically or from the command line), interactive, real-time visualization of the results will be continue to be essential, especially for the early stages of development. As the program framework is relatively simple and easily extended, we anticipate that it could also be adapted to monitoring live beamline data with minimal effort.

In the long term, we hope to integrate this tool with others developed by the CCI group, including alternative methods for displaying raw diffraction data (Sauter, 2011), visualization of reciprocal space (Echols & Adams, 2011), and analysis of processed data quality (Zwart *et al.*, 2005).

Availability

All code described here is open-source under the CCTBX license, and may be downloaded from http://cctbx.sf.net. The CCTBX builds do not include the required graphical libraries, so additional installation of wxPython (version 2.8.12.1 or newer) and Matplotlib (1.0.1 or newer) is necessary. (These are included in the *PHENIX* installers.)

Acknowledgments

We thank the NIH (grant numbers R01-GM095887 and P01-GM063210) and the *PHENIX* Industrial Consortium for financial support.

References

Echols, N. & Adams, P.D. (2011) Computational Crystallography Newsletter 2, 88-92.

Grosse-Kunstleve, R. W., Sauter, N.K., Moriarty N.W. & Adams, P.D. (2002). J Appl Cryst. 35, 126-136.

Otwinowski, Z. & Minor, W. (1997). Methods in Enzymology 276, 307-326.

Sauter, N.K. & Poon, B.K. (2010). J Appl Cryst. 43, 611-616.

Sauter, N.K. (2010). Computational Crystallography Newsletter 1, 18-23.

Sauter N.K. (2011). Computational Crystallography Newsletter 2, 15-24.

Zwart, P.H., Grosse-Kunsteleve, R.W., & Adams, P.D. (2005). CCP4 Newsletter Winter, Contribution 7.

On the contribution of hydrogen atoms to X-ray scattering

Afonine, P.V. [1,#] and Adams, P.D. [1,2]

[1]*Lawrence Berkeley National Laboratory, One Cyclotron Road, MS64R0121, Berkeley, CA 94720 USA*
[2]*Department of Bioengineering, University of California Berkeley, Berkeley, CA, 94720, USA.*

[#]Correspondence email: PAfonine@lbl.gov

Approximately half of all atoms in bio-macromolecular structures are hydrogens. While X-ray diffraction data rarely allows direct determination of their positions, with a few exceptions the geometry of hydrogen atoms can be inferred from the positions of other atoms. Even though a hydrogen atom is a weak X-ray scatterer its contribution to the total scattering is not negligible (for a review see Afonine *et al.*, 2010). Until recently it was customary to ignore hydrogen atoms throughout the process of crystallographic X-ray structure determination. However, it has been demonstrated (Chen *et al.*, 2010; Headd *et al.*, 2009; Davis *et al.*, 2007; Word *et al.*, 1999) that using hydrogens in structure determination typically improves model geometry and highlights problems otherwise difficult to detect. In this article we illustrate the contribution of hydrogen atoms to calculated X-ray structure factors and *R*-factors.

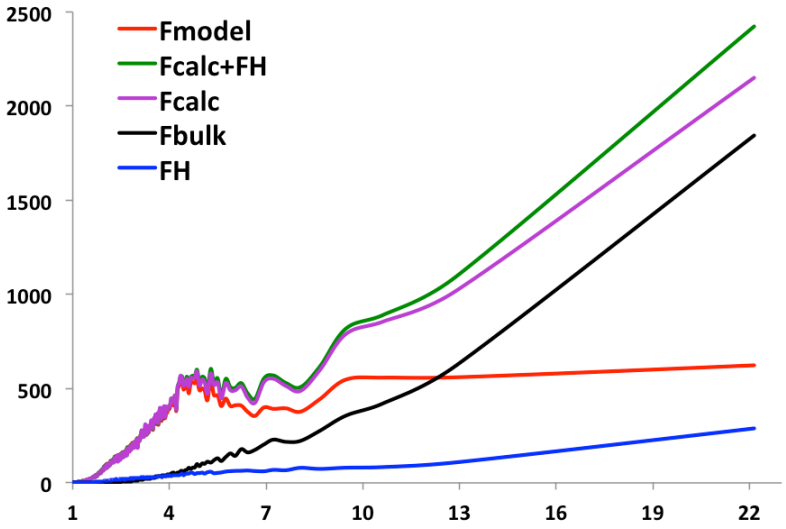


Figure 1. Resolution bin averaged values of each term in formula 1 calculated for a structure with code 1F8T using all theoretically possible reflections up to 1Å resolution.

Defining the total model structure factor (F_{model}) as the scaled sum of structure factors calculated from non-hydrogen atoms (F_{calc}), hydrogen atoms (F_H) and bulk-solvent (F_{bulk}) (Afonine *et al.*, 2005; Cooper *et al.*, 2010)

$$\mathbf{F}_{\text{model}} = k(\mathbf{F}_{\text{calc}} + \mathbf{F}_{\text{H}} + \mathbf{F}_{\text{bulk}}) \qquad (1)$$

allows us to illustrate the individual contributions in (1). Figure 1 shows resolution bin averaged values of each term of (1) calculated for a structure taken from the Protein Data Bank (PDB; code 1F8T) using all theoretically possible reflections to 1Å resolution. The bulk-solvent contribution was calculated as $F_{bulk} = k_{sol}\exp(-B_{sol}s^2/4)F_{mask}$ with $k_{sol} = 0.35 e/Å^3$ and $B_{sol} = 50 Å^2$ (for details see Afonine *et al.*, 2005), and k=1 since all the calculated terms in (1) are in absolute scale. As expected, the contribution from the hydrogen atoms (F_H; blue line) it not negligible compared to the total structure factor (F_{model}; red line). At low resolution F_{model} is significantly smaller than the components ($F_{calc} + F_H$) or F_{bulk} because even though the bulk-solvent contribution is large at this resolution, the bulk scatterers are out of phase with the protein scatterers and therefore the bulk-solvent contribution is out of phase with the other terms (Podjarny & Urzhumtsev, 1997). Our observation is that the plots in figure 1 are characteristic and do not vary significantly from structure to structure.

To illustrate the impact of hydrogen atoms on the crystallographic *R*-factor and how this depends on data resolution we selected approximately 250 structures from PDB. The structures were selected such that each of six resolution ranges (bins): 0-1, 1-1.5, 1.5-2, 2-2.5, 2.5-3 and 3-3.5Å contained approximately the same number of structures. Additional selection criteria aimed to select the best available structures and included 99% complete data across the whole resolution range, no twinning, *R*-factors lower than average, and minimal geometry violations (clashscore, C_β

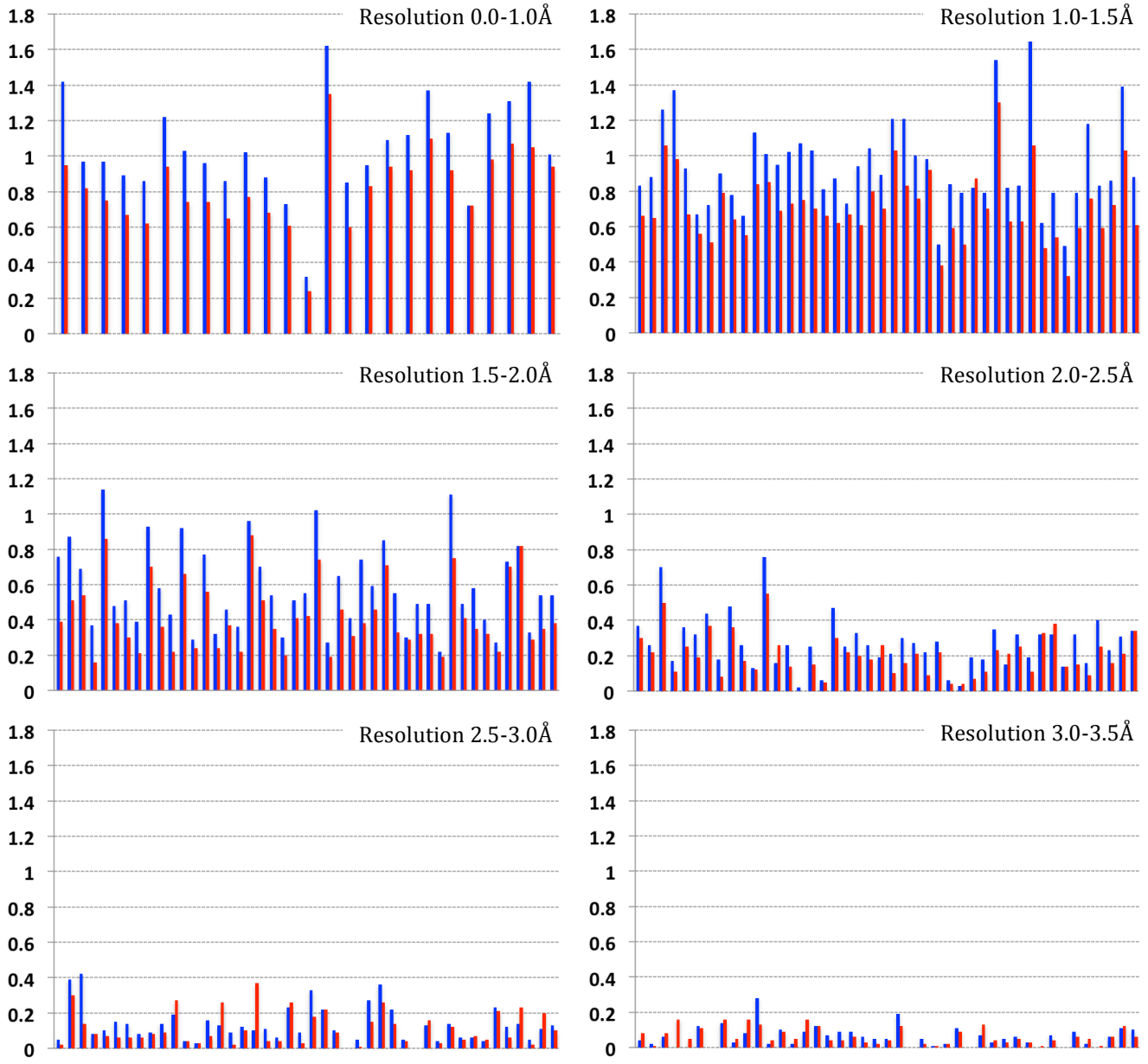


Figure 2. See text for details. R_{noH}-R_H(blue) and R_{noH}-R_{H_unique} (red). PDB codes of structures used in calculations in the order of their appearance in the plots; (0,1): *1mnz, 1nwz, 1r2m, 2ddx, 2e4t, 2fvy, 2gg2, 2gud, 2h5c, 2h5d, 2ov0, 2pne, 2ppp, 2rh2, 2zq7, 3agn, 3ago, 3cnj, 3f7l, 3gyi, 3gyj, 3l8w, 3mi4, 3noq, 3pyp*; (1,1.5): *1uww, 2eht, 2ivj, 2j23, 2j3p, 2jhq, 2jin, 2jju, 2o7i, 2qfe, 2r8o, 2rb2, 2rbo, 2rbp, 2rby, 2rc2, 2uxw, 2w47, 2xeu, 2z3h, 2zw0, 3a3v, 3ahs, 3b36, 3b3a, 3bc9, 3bne, 3ce1, 3deo, 3dxl, 3f9b, 3fm6, 3fwk, 3i3f, 3ioh, 3ivv, 3jsc, 3kwu, 3laa, 3m1z, 3m6b, 3mea, 3mnb, 3moy*; (1.5,2): *1lka, 1so3, 1t7t, 1vh2, 1xkg, 1y57, 1yb1, 1ze3, 2bik, 2d20, 2ei9, 2fp2, 2fxs, 2h9b, 2p5q, 2qvm, 2qwm, 2uya, 2ves, 2vun, 2wgb, 2wgp, 2wm3, 2wn2, 2x9w, 3a23, 3c67, 3cpg, 3d3i, 3ddw, 3dms, 3f3s, 3fj4, 3flu, 3fuy, 3g28, 3hbn, 3hbu, 3hfk, 3hgm, 3krr, 3kvc, 3l3v, 3m5v, 3m8u*; (2,2.5): *1o51, 1t73, 1w6w, 1x31, 2dbi, 2dso, 2fb0, 2hha, 2q36, 2qtb, 2uv2, 2vx0, 2vz6, 2w5o, 2wmr, 2wo3, 2wul, 3a7r, 3bbd, 3bbe, 3bkq, 3byi, 3dd3, 3dv5, 3e2k, 3e32, 3e87, 3g2f, 3gg3, 3haz, 3hd5, 3hk8, 3hy6, 3hzu, 3i3d, 3ib3, 3ic5, 3igu, 3iqa, 3l48, 3lnt, 3ls8, 3mtc, 3n51*; (2.5,3): *1e3h, 1h2n, 1o8c, 1yah, 1yc0, 1zkf, 2c44, 2ecf, 2f89, 2iun, 2izv, 2jas, 2qkx, 2rd5, 2vd5, 2vqa, 2vwk, 2w2c, 2wb7, 2zjy, 2zs9, 3a29, 3a2j, 3bic, 3c9l, 3csn, 3d2z, 3d8a, 3dd1, 3ffb, 3gn4, 3gxe, 3i4e, 3ibg, 3ihl, 3ir6, 3k5d, 3khj, 3llm, 3ly2, 3m4p, 3mg2, 3mle, 3n3k*; (3,3.5): *1b9x, 1l8h, 1n21, 1q9c, 1sjp, 1t7z, 1vg2, 1vg7, 1wdl, 1x03, 2a81, 2dgl, 2ffl, 2fqq, 2jjd, 2o0i, 2qqv, 2uy9, 2v8a, 2wdr, 2wdv, 2wfn, 2wuy, 2wy6, 2x8c, 2zrc, 2zrk, 3a2i, 3bbp, 3br1, 3dbc, 3dbd, 3dbe, 3dbf, 3ef7, 3ej1, 3fbn, 3gzp, 3hd7, 3hy5, 3ibp, 3krx, 3l2j*.

deviations, Ramachandran plot and rotamer outliers). For each structure we then computed three *R*-factors corresponding to a structure without hydrogen atoms (R_{noH}), a structure with all hydrogens added to expected positions (R_H) using the Reduce program (Word *et al.*, 1999) as

implemented in *phenix.reduce*, and a structure with hydrogens added to uniquely defined positions only (no hydrogens with rotational degrees of freedom; R_{H_unique}). Since *phenix.reduce* adds hydrogens to nuclear positions we re-optimized the X-H bond lengths (X is the heavy atom the hydrogen, H, is bonded to) such that the new hydrogen positions correspond to the electron cloud distance (for details see Afonine *et al.*, 2010). Since hydrogen atoms were added to already well refined structures we had to scale their contribution F_H to account for the fact that the refined ADPs of non-hydrogen atoms may have been inflated to account for absent hydrogens. This effect has been observed previously when anisotropic ADPs can model deformation density at ultra-high resolution (Afonine *et al.*, 2004). The scaling of F_H consisted of multiplying it by a resolution dependent factor $k_h \exp(-B_h s^2/4)$ with two refinable parameters k_h and B_h. Also, this scaling of F_H is intended to account for the effect of hydrogen atom abstraction (when applicable) described by Meents *et al.* (2009).

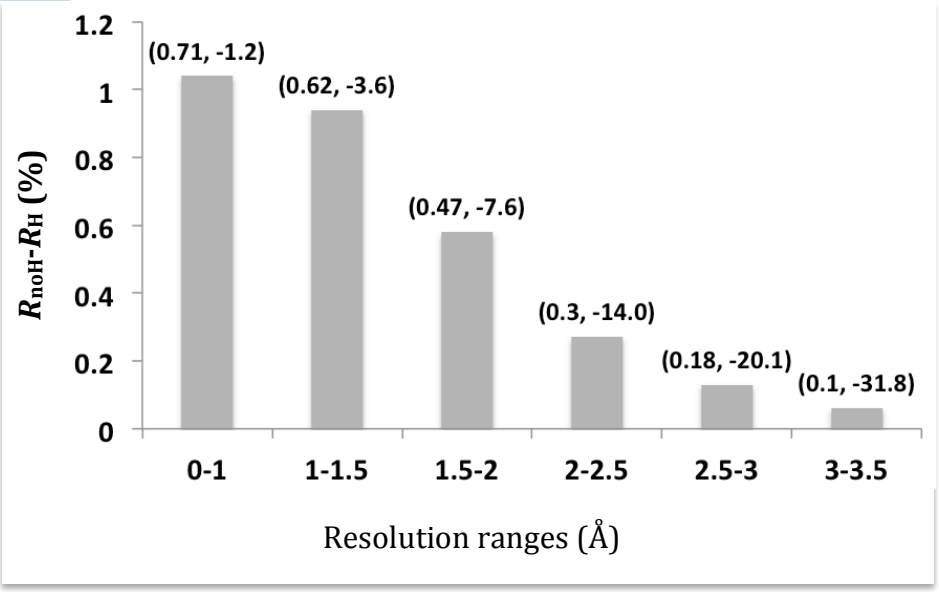


Figure 3. Averaged R_{noH}-R_H values shown for six resolution ranges. Each bar caption shows corresponding bin-averaged pairs of (k_h , B_h).

Figure 2 shows six plots corresponding to six selected resolution bins. Each plot presents two series of bars representing the *R*-factor differences R_{noH}-R_H (blue) and R_{noH}- R_{H_unique} (red) for each structure. It is clear that contribution of hydrogen atoms is non-zero across all six selected resolution ranges, and it ranges from an average of approximately 1% at highest resolution to about 0.04% at lowest resolution. In all cases adding hydrogen atoms improved the *R*-factors. Not including hydrogens with rotational degree of freedom almost always diminishes the *R*-factor improvement at resolutions up to 2.5Å and has a mixed effect at resolutions worse than 2.5Å. The decrease of *R*-factor improvement (R_{noH}-R_H) at lower resolution may have at least two explanations: 1) the positional error of non-hydrogen atoms is higher at lower resolution which in turn means a higher positional error for the hydrogen atoms and therefore less improvement in *R*-factor, and 2) as mentioned above the inflated ADPs of non-hydrogen atoms may already have partly compensated for the absence of hydrogens.

Figure 3 shows averaged R_{noH}-R_H values shown for six resolution ranges along with corresponding bin-averaged values of k_h and B_h. These values may be different for models where hydrogen atoms were used throughout the process of structure determination and refinement, as this may make the contribution of hydrogen atoms more distinct.

In summary, the contribution of hydrogen atoms to X-ray scattering is not negligible, hydrogens do contribute to the total model structure factor and we have illustrated how this affects the *R*-factors. The effect on the *R*-factor diminishes with resolution and could be the result of using well refined structures in our tests or/and at lower resolution the predicted positions of hydrogen atoms are less accurate. It is therefore possible that the effect on *R*-factor may be more significant if hydrogens were used throughout the process of structure determination and refinement. Finally, good quality structures at high resolution permit the inclusion of hydrogens possessing rotational degree of freedom. However this is not the case for lower resolution structures, and is likely also not the case for partially refined models.

References

Afonine, P.V., Mustyakimov, M., Grosse-Kunstleve, R.W., Moriarty, N.W., Langan, P. & Adams, P.D. (2010). *Acta Cryst*. D**66**, 1153-1163.

Afonine, P. V., Grosse-Kunstleve, R. W. & Adams, P. D. (2005). *Acta Cryst.* D**61**, 850-855.

Chen, V.B., Arendall III, W.B., Headd, J.J., Keedy, D.A., Immormino, R.M., Kapral, G.J., Murray, L.W., Richardson, J.S. & Richardson, D.C. (2010). *Acta Cryst*. D**66**, 12-21.

Cooper, R.I., Thompson, A.L. and Watkin, D.J. (2010). *J. Appl. Cryst.* **43**, 1100-1107.

Davis, I.W., Leaver-Fay, A., Chen, V.B., Block, J.N., Kapral, G.J., Wang, X., Murray, L.W., Arendall III, W.B., Snoeyink, J., Richardson, J.S. and Richardson, D.C. (2007). *Nucleic Acids Res.* **35**, W375-W383.

Headd, J.J., Immormino, R.M., Keedy, D.A., Emsley, P., Richardson, D.C. and Richardson, J.S. (2009). *Journal of Structural and Functional Genomics*. **10**, 83-93.

Meents, A., Dittrich, B. and Gutmann S. (2009). *J. Synchrotron Rad.* **16**, 183-190.

Podjarny, A. D. & Urzhumtsev, A.G. (1997). *Methods Enzymol*. **276**, 641-658.

Word, J.M., Lovell, S.C., Richardson, J.S. and Richardson, D.C. (1999). *JMB*. **285**, 1735-47.

CCTBX tools for derivative-free optimization

Haiguang Liu[1,2], Billy Poon[1], Gang Chen[1], Ralf Grosse-Kunstleve[3] & Peter Zwart[1*]

1. Berkeley Center for Structural Biology, Lawrence Berkeley National Laboratories, Physical Biosciences Division

2. Currently at Arizona State University, Department of Physics

3. Computational Crystallographic Initiative, Lawrence Berkeley National Laboratories, Physical Biosciences Division

Email: PHZwart@lbl.gov

Introduction

Many applications developed in crystallography, small angle scattering and general scattering sciences require the determination of model parameters given experimental data (Afonine *et al.*, 2005; Liu *et al.*, 2012). In crystallography, gradient-based optimizers such as L-BFGS (Liu & Nocedal, 1989) have proven to be quite successful for handling challenges associated with large-scale optimization problems such as structure refinement. Whereas the L-BFGS and other Newton-based optimizers available in CCTBX have proven their utility (Bourhis *et al.*, 2007), the drawback of these methods is the requirement that the starting solution is close to the global minimum of the target function, and that the path from the starting location towards the global minimum does not lead through local minima in which gradient-based methods may get trapped. From a software development point of view, an additional drawback is the need to implement numerically stable and well-tested first (and second) derivatives. Deriving, implementing and testing gradients and second derivatives can often be time consuming relative to the problem one aims to solve, especially when the target functions are involved or when one has to deal with domain restrictions for the parameters. In cases where the optimization of the parameters is not a (perceived) time-limiting step in the total, the use of derivative-free optimizers can result in straightforward and robust code that gets the job done. In this article, we highlight the derivative-free optimization methods available in the CCTBX and provide a brief comparison of their performance.

Available Methods

In the following sections, five different derivative-free optimization methods are outlined. For completeness, we provide a brief overview of the L-BFGS gradient-based optimizer as well. The code snippets are taken from `scitbx/examples/minimizer_examples.py`, as distributed with CCTBX from January 19, 2012 onwards.

Gradient based methods

Available gradient-based minimisation methods in CCTBX are

1. L-BFGS (`scitbx.lbfgs`)
2. Damped Newton (`scitbx.minimizers.damped_newton`)
3. Newton with More-Thuente line search (`scitbx.newton_more_thuente_1994`)

The L-BFGS minimizer uses first derivatives only, but builds up an estimate of the inverse of the Hessian based on the BFGS formula (Nocedal & Wright, 2006) using information from previous steps. For large-scale optimization problems for which first derivatives are available, this minimizer is typically a good choice. The Newton-type minimizers (with and without line search) require second derivative information. These minimizers are well suited for problems in which the variables have a large spectrum of scales, or problems with strongly correlated parameters. By design, these gradient-based minimizers perform a local search only.

Simplex Search

A standard simplex algorithm is available for local optimization purposes. The simplex optimization algorithm was first proposed by Nelder and Mead (Nelder & Mead, 1965). A modified version is adopted in the current CCTBX implementation. A pure-Python implementation of this algorithm is available in `scitbx.simplex`. A prototype use of this optimizer is shown in Scheme 1.

Direct-Search Simulated Annealing (DSSA)

Combining a simplex search with simulated annealing results in a versatile and robust optimizer that is able to tackle a wide variety of optimization problems (Hedar & Fukushima, 2002). Due to the introduction of the annealing

```
class test_simplex():
  def __init__(self, name):
    self.n = 2; self.x = flex.double(self.n, 2)
    self.target = Function(name)(dim).eval
    self.starting_simplex=[]
    for ii in range(self.n+1):
        self.starting_simplex.append(
                 (flex.random_double(self.n)/2-1)*0.1+ self.x)
    self.optimizer = simplex.simplex_opt( dimension=self.n,
          matrix  = self.starting_simplex,evaluator=self,tolerance=1e-10)
    self.x = self.optimizer.get_solution()
```

Schema 1: Simplex method

```
class test_dssa():
  def __init__(self,name):
    self.n = 2
    self.x = flex.double(self.n, 2)
    self.target = Function(name)(dim).eval
    self.starting_simplex=[]
    for ii in range(self.n+1):
      self.starting_simplex.append(
          0.01*(flex.random_double(self.n)/2-1) + self.x)
    self.optimizer = direct_search_simulated_annealing.dssa(
                          dimension=self.n,
                          matrix = self.starting_simplex,
                          evaluator = self,
                          tolerance=1e-8,
                          further_opt=True,n_candidate=None,
                          coolfactor=0.5, simplex_scale=1
                                         )
    self.x = self.optimizer.get_solution()
```

Schema 2: DSSA method

procedure, the algorithm is less likely to get stuck in local minima. Furthermore, the rate of convergence is enhanced by the use of a local search. The interface to the DSSA algorithm is similar to that of the simplex method and is found in `scitbx.direct_search_simulated_annealing` (See Schema 2). Although the algorithm is mostly self-steering, important parameters are the cooling factor and the choice of the initial simplex size.

Cross Entropy (CE)

The cross-entropy method for optimization (Rubenstein, 1997) is a Monte Carlo based optimization method. In the CE method, a prior probability distribution of the refinable parameters is defined from which candidate solutions are drawn. Each candidate solution is subsequently used to compute calculated data. Using only a small fraction of candidate solutions (the elite set) that provide a best fit to the data, the prior distribution is modified using the elite candidates. The interface to this optimizer requires the definition of a prior distribution by providing estimates of the mean and standard deviation of the proposed parameters.
The updates to the sampling distribution are performed on the mean and standard deviations only.

```
class test_cross_entropy():
  def __init__(self,name):
      self.n = 2
      self.x = flex.double(self.n, 2)
      self.target = Function(name)(dim).eval
      self.means = flex.double( self.n, 4.0 )
      self.sigmas = flex.double( self.n, 2.0 )
      self.optimizer =  cross_entropy.cross_entropy_optimizer(self,
                                                mean=self.means,
                                                sigma=self.sigmas,
                                                alpha=0.75,
                                                beta=0.75,
                                                q=8.5,
                                                elite_size=10,
                                                sample_size=100)
```

Schema 3: Cross Entropy method

```
class test_cma_es():
  def __init__(self,name):
      self.m = flex.double( [2,2] )
      self.s = flex.double( [1,1] )
      self.name = name
      self.minimizer = cma_es_interface.cma_es_driver(
                                          2, self.m, self.s, self.my_function )
      self.best_solution = self.minimizer.x_final

  def my_function(self,vector):
      tmp = Function(self.name)(dim)
      return tmp.eval(list(vector))
```

Schema 4: Covariance Matrix Adaption – Evolution Strategy

Estimates of the correlation between parameters are not taken into account. Important parameters to experiment with are the elite and sample sizes, as well as the learning rates, alpha and beta. The learning rates, which range between 0 and 1, affect the steps sizes when updating the means (alpha) and standard deviations (beta).

Covariance Matrix Adaptation - Evolution Strategy (CMA-ES)

CMA-ES is another Monte Carlo based optimization method (Hansen, 2003). In essence, it is very similar to the CE method outlined above, but has two major advantages. First of all, contrary to the CE method, CMA-ES actively explores correlations between parameters. A second advantage is that it provides adaptive step-size control in order to prevent preliminary convergence. The CMA-ES method starts with a trial Gaussian distribution assuming no dependence between input parameters. In each optimization cycle, a set of candidate solutions is drawn from this sampling distribution. The candidate solutions are sorted according to their correspondence to the data. At this point, a weighted mean is computed from these candidates, with weights proportional to their ranks only. This updated mean is an improved estimate over the previous estimated mean. In a similar manner, updates to the estimated covariance matrix are performed. Integral to the distribution updates is the determination of the step size. The adaptive step-size control protocol prevents premature convergence of standard deviations to very small values. This prevents the algorithm from premature convergence as is sometimes seen in the CE method. An example

```
class test_differential_evolution():
  def __init__(self,name):
      self.n = 2
      self.x = None #flex.double(self.n, 2)
      self.target = Function(name)(dim).eval
      self.domain = [(start[0]-2,start[0]+2),(start[1]-2, start[1]+2)]
      self.optimizer=differential_evolution.differential_evolution_optimizer(
            self,population_size=20,cr=0.9,n_cross=2)
```

Schema 5: Differential Evolution

interface is found below:
The algorithm is virtually free of parameters that needs to be tuned, in contrast to the DSSA or CE algorithms.

Differential Evolution (DE)

Differential evolution is yet another population-based, Monte Carlo type optimization algorithm (Storn & Price, 1997). In the DE algorithm, new candidate solutions are constructed around an existing solution by perturbations generated from the difference between two other candidate solutions. Over time, an initial population with large spread contracts around the (hopefully) global minimum. The interface is similar to most other optimizers discussed in this article:
As in the case for CMA-ES, DE can be used virtually as a black box. The only important parameter is the definition of the domain in which the solution is likely to be found. This definition does not necessarily need to contain the minimum.

Results

To test the performance of these algorithms, four test functions (Easom, Rosenbrock, Ackley, Rastrigin) were optimized with the derivative free-optimization methods. For comparison, L-BFGS with finite difference derivatives (FD-L-BFGS) trials were performed as well. Apart from the Rosenbrock function, all test functions have multiple local minima. The test functions are depicted in Figure 1.

The (mean) starting point for all optimizations is located at (4,4), except in the tests with the Easom function, in which an initial solution around (0,0) was used. Table 1 lists the average and standard deviation of the number of function calls before convergence is reached for 1000 independently seeded runs of each minimizer.
As is clear from Table 1 (and common wisdom), the availability of derivative information (FD-L-BFGS) improves the rate of convergence, at the price of getting trapped in local minima. The same holds for the simplex algorithm. Furthermore, it is also clear that the particular simplistic implementation of the CE algorithm in `scitbx` is suboptimal because it requires many iterations before convergence and the number of function calls is highly dependent on how the random number generator is initially seeded.
The difference between CMA-ES and DE in the above tests is striking, since both algorithms can handle global optimization problems. The influence of the sample size on the success rate of the algorithm is demonstrated in Table 2.
From the above table it is clear that for this particular test problem, CMA-ES outperforms DE at low sample sizes. For the 2D Rastrigin test function, DE outperforms CMA-ES for sample sizes above 15. This behavior is however very specific to the test function. For a 20 dimensional Rosenbrock function for instance, DE (population size 40) has a success rate below 20% to converge to the right solution within 27000 function

Table 1: Average and standard deviation for the number of function calls before convergence for 1000 trials. Italicised numbers indicate that the global minimum was not found. All minimizers were run with typical settings.

Optimizer / Function	Easom	Rosenbrock	Ackley	Rastrigin
FD-L-BFGS	36	120	*100*	*20*
Simplex	195 ± 20	403 ± 84	*124 ± 96*	*126 ± 16*
CE	17087 ± 4605	13633 ± 9722	18745 ± 4894	14115 ± 4894
DSSA	379 ± 96	884 ± 37	561 ± 128	*476 ± 21*
DE	792 ± 80	1525 ± 150	2596 ± 181	2140 ± 349
CMA-ES	551 ± 128	820 ± 118	976 ± 69	*745 ± 100*

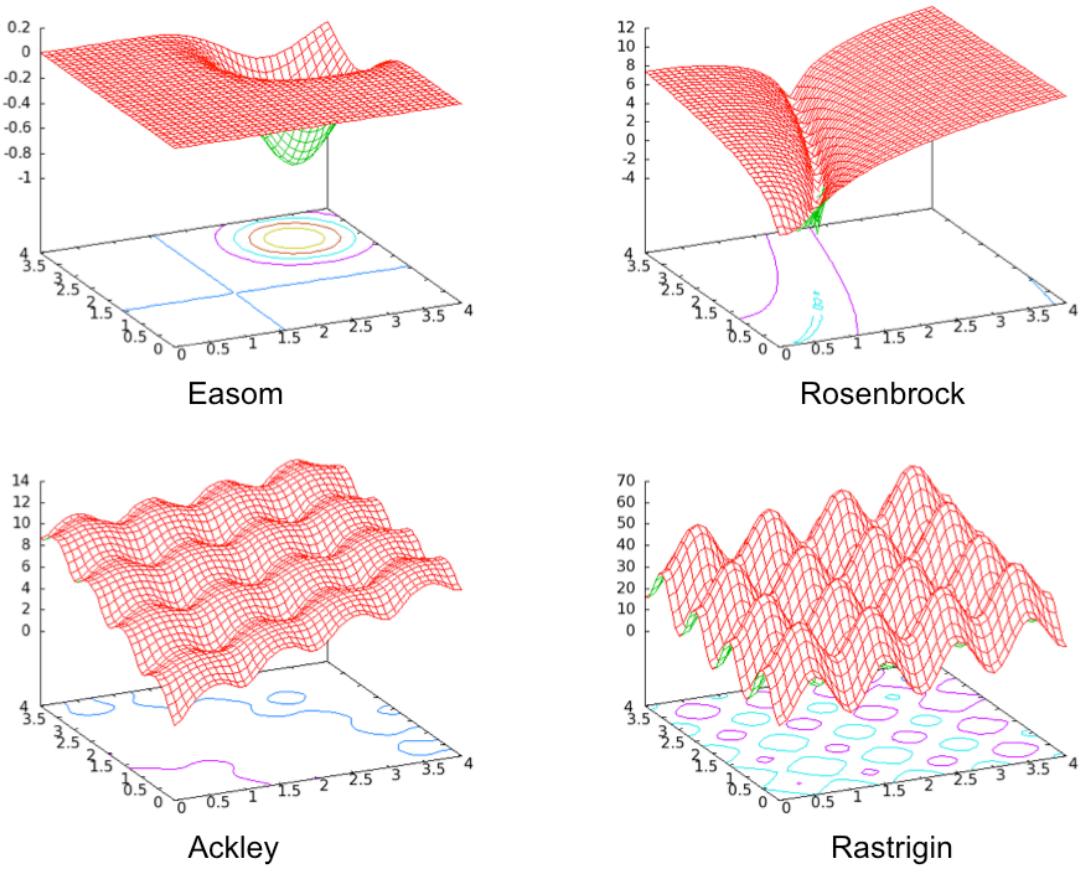


Figure 1: Test functions for optimization routines.

evaluations, while CMA-ES (with a population size of 20) converges to the correct answer within 27000 function evaluations. Since the Rosenbrock function is convex, the FD-L-BFGS algorithm outperforms CMA-ES by using only 1100 function evaluations.

Discussion and Conclusions

Optimization, very often involving non-quadratic target functions, plays an important role in many scientific procedures. When confronted with an optimization problem, typically two choices emerge. Either one has to estimate a starting solution, sufficiently close to the global minimum, that is within the reach of gradient-based local minimization methods. Or alternatively, one has to treat the optimization problem as a (semi-)global problem and use non-local optimizers to find the solution. The `scitbx` module of CCTBX includes functionality for both choices as illustrated in this article. The derivative-free optimizers are typically outperformed by the L-BFGS minimizer on convex functions. For (semi-)global optimization problems, both DE and CMA-ES seem to be suitable choices as they require little tuning and both have good convergence properties. The availability of these optimizers in `scitbx` removes important bottlenecks in scientific software development.

Table 2: Success rate of DE and CMA-ES for 2D-Rastrigin global optimization problem.

Population size	DE	CMA-ES
5	0.01	0.24
10	0.27	0.45
15	0.67	0.58
20	0.88	0.71
40	1.00	0.91

References

Afonine, P.V., Grosse-Kunstleve, R.W. & Adams, P.D. (2005). *CCP4 Newsletter on Protein Crystallography*, **42** (8).

Bourhis, L.J., Grosse-Kunstleve, R.W. & Adams, P.D. (2007). *Newsletter of the IUCr Commission on Crystallographic Computing*, **8**, 74-80.

Hansen, N., Müller, S.D. & Koumoutsakos, P. (2003). *Evolutionary Computation*, **11**, 1–18.

Hedar, A. & Fukushima, M. (2002). *Optimization Methods and Software*, **17**, 891-912.

Liu, D.C. & Nocedal, J. (1989). *Mathematical Programming B* **45** (3), 503–528.

Liu, H., Morris, R. J., Hexemer, A., Grandison, S. & Zwart, P.H. (2012). *Acta Crystallographica* A (in press).

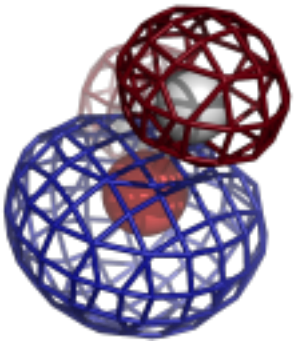
Nelder, J.A. & Mead, R. (1965). *The Computer Journal*, **7**, 308-313.

Nocedal, J. & Wright, S.J. (2006). *Numerical Optimization, 2nd Edition*, Springer Verlag.

Rubinstein, R.Y. (1997). *European Journal of Operations Research*, **99**, 89-112.

Storn, R. & Price, K. (1997). *Journal of Global Optimization*, **11**, 341-359.

VOLUME THREE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXII

QUEUING, ALTERNATIVE LOCATIONS, AM1/RM1

Table of Contents

- PHENIX News 28
- Crystallographic meetings 29
- Expert Advice
  - Fitting tips - Rotamer correction with backrub 29
- FAQ 30
- Short Communications
  - DETAC: tools to detect alternative conformations by unrestrained refinement 32
  - ERRASER, a powerful new system for correcting RNA models 35
- Articles
  - *cctbx* tools for transparent job execution on clusters 37
  - On the analysis of residual density distributions on an absolute scale 43

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New releases

Model-building

New beta-testing releases in model-building include phenix.combine_models and a trace_chain option for phenix.fit_loops. The combine_models method will take two overlapping models and a map and try to identify the best parts of each model based on the map. It then creates a single new model. The method can be used for models that are assigned to sequence as well as those that are not. The trace_chain option for loop fitting uses a fast chain-tracing algorithm to find a path between two ends of a loop and then builds a loop based on that chain.

New features

New autosol features

Classical density modification is available in the autosol program using the command mask_type="classic". Autosol (versions 1078 and higher) now has a "derscale=xxx" keyword that allows you to fix the scale factor for derivative datasets (useful for rip phasing where the scale factor can be critical for structure solution).

Anisotropy corrections and sharpening

phenix.autosol and phenix.autobuild now automatically apply an anisotropy correction and sharpening for all density-modified map calculations. At the end of these methods a final model is produced that is refined against the original (uncorrected) data. This model is

written out as overall_best.pdb. The uncorrected data for refinement is written out as overall_best_refine_data.mtz. You can adjust how this works by specifying a target overall B-value if you want (e.g., b_iso=25) , and you can also turn it off if you like (remove_aniso=False).

Ensemble output options in phaser.MRage.solutions

Ensemble models, which contain structural information from a series of homologous templates, can be very powerful in molecular replacement. However, after molecular replacement has succeeded, information content of the solution is reduced, since due to limitations of the PDB format, the ensemble model is normally replaced by the best model in the collection. Novel output options have been added to phaser.MRage.solutions that allow the preservation of the previously discarded information content in part or in full. (It is important to note that the output map is not affected in any way, and is always calculated with the full information content of the selected solution.) The list of output options are as follows (available via the --output switch): - single (default). This just select the first model from the ensemble. - best-single-model. This calculates the LLG of each model and selects the one that gives the highest LLG. The best scoring model of one ensemble is selected independently for each copies of the ensemble, and can therefore vary if there are significant differences between the NCS-related molecules. - combined. This option uses phenix.combine_models to combine all components of the ensemble model into a "chimera" model, based on their fit to the electron density. As for the previous option, this is also done independently for each copies of the ensemble, and the procedure can yield slightly different results for NCS-related molecules. - ensemble. Outputs all components of the ensemble model in one file. Altloc identifiers are assigned to each component of the ensemble.

Crystallographic meetings and workshops

Gordon Research Conference on Diffraction Methods in Structural Biology, Bates College, Lewiston, ME, July 15-20, 2012

Jeff Headd will be presenting at the Gordon Conference on Diffraction Methods in the "Getting the Best Out of Your Data: Data Analysis" section.

The 27th Meeting of the European Crystallographic Association, Bergen, Norway, August 6-11, 2012.

Pavel Afonine will give the plenary and receive the Fifth Erwin Felix Lewy Bertaut Prize of the European Crystallographic Association (ECA) and European Neutron Scattering Association (ENSA).

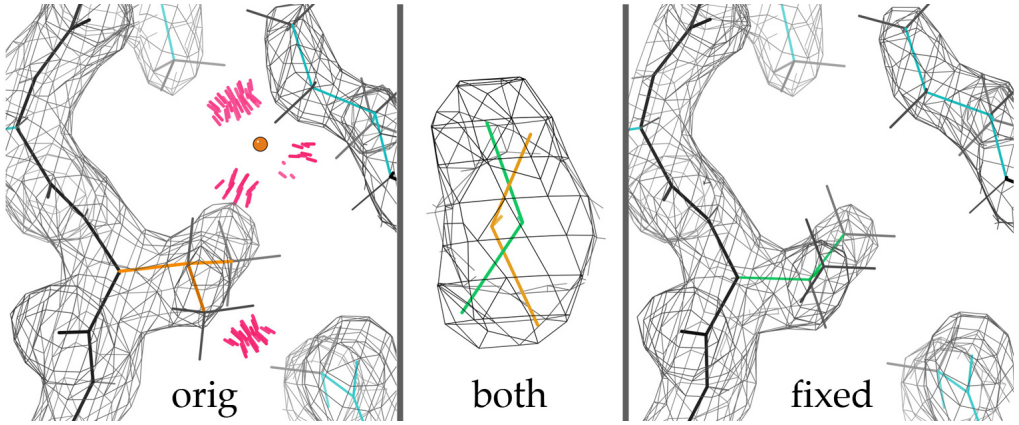
Expert advice

Fitting Tip - Rotamer correction, with backrubs

Jane Richardson, Duke University; Jeff Headd, LBL

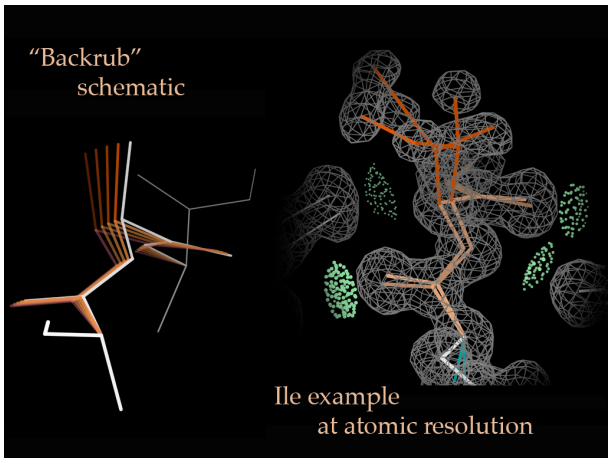
Backward-fit `Asn/Gln/His` aminoacid side-chains (including His protonation - see Tip CCN, January, 2012) can be corrected essentially without affecting the match of model to data. However, sidechains with tetrahedral geometry (`Thr`, `Val`, `Ile`, `Leu`, and even `Arg`) may also be modeled backwards into elongated or flat density, and those corrections must be done along with a refinement process, either integrated or as rebuilding (e.g. in Coot or in KiNG) alternated with refinement.

The figure on the next page shows an example (`Val` 218 in `1LPL`) where the density does not clearly indicate directionality of the tetrahedral branch, and the wrong choice was modeled (in gold). A problem is diagnosed by the doubly-eclipsed rotamer outlier and the serious all-atom clashes. The imaginary water is also suspicious, and such cases distort bond



angles, often enough to be 4σ outliers although not here. If you know that such misfits are a common systematic error, you will see immediately how to make the correction, which is easy and satisfying to do. The Cβ position and sidechain geometry should first be idealized; then the χ1 value changed about 180° to align with the density from the other direction (for a backward `Leu` or `Arg`, 2 χ angles must be changed). This will usually place the sidechain parallel to but somewhat outside the density, so it must be shifted in by a manual fit or by real-space refinement. The result (green), as replaced in `1TOV` (Arendall 2005) occupies more-or-less the same space, fits the density a bit better, and corrects all of the validation outliers.

The most effective motion for achieving the shift of sidechain sidewise into density is the "backrub" (Davis 2006), a simple rotation around an axis between the i-1 and i+1 Cα's that has a good lever-arm for shifting the Cβ in a direction perpendicular to the backbone. The second figure (below) shows a schematic of the backrub motion and an example of how this same motion is used dynamically to relate two clear alternate conformations seen at high resolution. You can get a good feeling for how the backrub motion works by rebuilding backward-fit sidechains in KiNG, where it is implemented as a separately controllable feature under the "fit rotamer" tool. Try `Thr 77 in 1BKR`, for instance, or `Ile A 120 in 1MO0`. Then you will understand better how backrubs are helping the real-space rotamer rebuilding in Coot or in other automated procedures, as well as being able to do them yourself when that's needed. In Phenix, backrubs have now been added to enable even better rotamer correction in the torsion-NCS procedure, and in future they will also be added to the real-space rotamer fit option within phenix.refine.



References

Davis IW, Arendall WB III, Richardson JS, Richardson DC (2006). "The Backrub Motion: How Protein Backbone Shrugs When a Sidechain Dances." *Structure* **14**: 265-274.

Arendall WB III, Tempel W, Richardson JS et al. (2005). "A test of enhancing model accuracy in high-throughput crystallography." *Journal of Structural & Functional Genomics* 6, 1-11.

Contributors

P. D. Adams, P. V. Afonine, G. Bunkóczi, F. C. Chou, R. Das, N. Echols, J. J. Headd, N. W. Moriarty, J. S. Richardson, N. K. Sauter, O. V. Sobolev, T. C. Terwilliger, A. Urzhumtsev

FAQ

How can I use semi-empirical quantum chemical methods to optimise the geometry of my ligand?

There is an option in the ligand restraints building module in PHENIX known as eLBOW (Moriarty, Grosse-Kunstleve & Adams, 2009) the can do this for you. The option is `--opt` and will minimise the energy of geometry for the ligand. The minimised geometry is reflected in the ideal restraints written to disk.

It was originally decided to use the Austin Model 1 (AM1) with the implementation being released in 2007. It soon became clear that the reparameterisation for a subset of elements (H, C, N, O, P, S, F, Cl, Br, and I) known as Recife Model 1 (RM1) was easily implemented on top of the AM1 method and performed better for the smaller set of elements. For more information on the implementation of RM1 in eLBOW see the reference and for more information of the RM1 method see Rocha et al., 2006.

In addition to the semi-empirical methods implemented in eLBOW, external quantum chemistry packages can be used to optimise the ligand geometry. For example, with the GAMESS package installed, adding the `--gamess` option will use the AM1 method in GAMESS. Furthermore, the other higher-level quantum mechanical methods are available via the `--method` and `--basis` options.

References

Moriarty, N. W., R. W. Grosse-Kunstleve, et al. (2009). "electronic Ligand Builder and Optimization Workbench (eLBOW): a tool for ligand coordinate and restraint generation." Acta Crystallographica Section D **65**: 1074-1080.

Rocha, G. B., R. O. Freire, et al. (2006). "RM1: A Reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br and I." J. Comput. Chem. **27**: 1101-1111.

DETAC: tools to detect alternative conformations by unrestrained refinement.

Oleg V. Sobolev

Institute of Mathematical Problems of Biology, Russian Academy of Sciences, Pushchino, Moscow region, 142290, Russia

Correspondence email: sobolev@impb.psn.ru

Introduction

Unrestrained refinement is stable for the vast majority of atoms when working at atomic resolution. Nevertheless geometrical restraints should be used in refinement for residues that are present in several (alternative) conformations in the crystal used for the X-ray experiment, otherwise residue geometries deteriorate significantly. We believe that large distortions of the residue geometries in unrestrained refinement may hint at the presence of alternative conformations for this residue.

To get these hints in a routine way we suggest two methods to analyze shifts of atomic centers resulted from several cycles of unrestrained refinement. A simple diagram plotting the values of atomic shifts against the residue number may give an idea of the crystallographic order of different parts of the structure at qualitative level (Sobolev & Lunin, 2008). To put the analysis on a more quantitative basis several decision-making procedures were developed and tested which compose a list of residues that are likely to be present in alternative conformations or disordered. These residues should be checked thoroughly with Fourier syntheses and included into the model with alternative conformations, when necessary (Sobolev & Lunin, 2012).

To study the mobility of atoms in unrestrained refinement and derive parameters for decision-making procedures we studied 203 structures from PDB with resolution of experimental data better than 1.2Å and R-work better than 0.13. We took care in setting the limits for R-factor values of the models to select high-quality structures in which the alternative conformations were assigned reliably. The following steps were applied sequentially to each selected model:

1. The alternative conformation possessing the largest occupancy was left in the model with the occupancy set to 1.
2. Hydrogen atoms were included.
3. The standard restrained refinement was performed for the model with `phenix.refine` using anisotropic ADP and riding hydrogens.
4. Three macro-cycles of unrestrained refinement were performed with `phenix.refine`.
5. Calculation of atomic shifts between model after step 3 and 4.

The performed analysis of atomic shifts values revealed significant difference between the atoms in alternative and single conformations. Moreover, side chain atoms usually get larger shifts than main chain atoms, and atoms at the surface of the molecule get larger shifts than atoms inside the globule. At the average, shifts are increasing with decrease of data resolution and R-factor growth. These differences provide the possibility of distinguishing atom types by the study of their atomic shifts in unrestrained refinement.

TUR-routine

To analyze a particular model, we suggest the application of the Trial Unrestrained Refinement (TUR). The TUR should consist of three macro-cycles of unrestrained refinement by `phenix.refine` with anisotropic ADP and riding hydrogens. The crystal should diffract to approximately 1.2Å or better and the model should be refined rather well (to the R-factor around 0.15 or better). The following two programs (`shift_plot` and `ac_prediction`) may be

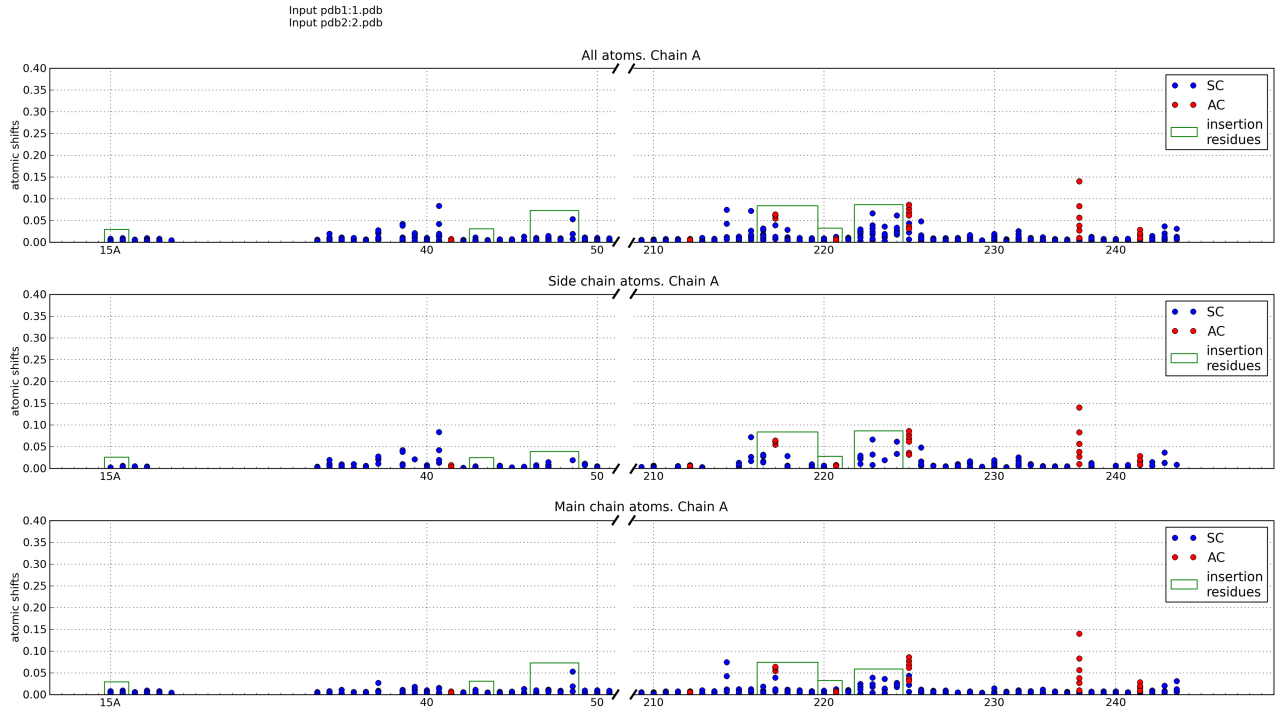


Figure 1: Fragment of diagram of atomic shifts produced by shift_plot for 1SSX structure (Fuhrmann *et.al.*, 2004).

used to analyze the results of the TUR.

Shift_plot

`Shift_plot` displays the results of TUR as several diagrams presenting atomic shifts vs. residue number (Fig.1). In these diagrams each dot corresponds to a non-hydrogen atom and the dots corresponding to the same residue are grouped in a column. High columns reveal residues that had significant atomic shifts in unrestrained refinement and thus are unstable and suspected to be present in alternative conformations. `Shift_plot` produces three diagrams: (i) for all atoms, (ii) for side chain atoms and (iii) for main chain atoms. The diagrams hint at which residues may have alternative conformations, but the proper choice of cut-off level to select high columns is not always obvious. These diagrams may be used at any resolution and do not depend on the refinement program.

Atoms that were in alternative conformations in input PDB files are plotted as red circles. Some PDB files contain residues with equal residue number but different insertion code (column 27 of PDB files). Such residues will be plotted as separate columns in the diagrams but these columns will be marked by green rectangle that shows that residues in it have the same residue number.

ac_prediction

`ac_prediction` implements automatic decision-making procedure that classifies every residue as 'single conformation' or 'alternative conformations' (SC and AC below). The current version of `ac_prediction` works with atomic-resolution data and unrestrained refinement conducted by `phenix.refine`. The procedure is based on the observation that SC-residues and AC-residues have different mobility in unrestrained refinement. The decision is made either by comparing the observed atomic shifts with a predetermined threshold, or comparing the probabilities of

such shifts for SC and AC-residues. The thresholds and probability distributions of atomic shifts were calculated during analysis of unrestrained refinement of 203 atomic-resolution structures from PDB.

To run the program, two PDB files with models before TUR and after TUR should be provided along with resolution and R-factor values of the model before TUR. `ac_prediction` produces a list of residues that should be checked first with electron-density maps as main candidates for introducing alternative conformations.

It should be noted that, `ac_prediction` does not guarantee the correctness of the decision. It provides a user with sorted list of residues for which the presence of alternative conformations is the most probable. The testing of `ac_prediction` proved that prediction based on atomic shifts is more accurate than prediction based on ADP or density values in atomic centers in '2mFo-DFc' syntheses (Sobolev & Lunin, 2012).

Availability

The programs are available for free download from the following address:
http://www.impb.ru/lmc/programs/ac_prediction/

Questions, comments and bug reports are welcome.

Acknowledgments

The author is grateful to Prof. Vladimir Lunin for useful discussions. This work was partially supported by grant 10-04-00254a of Russian Foundation for Basic Research.

References

Fuhrmann C.N., Kelch B.A., Ota N., Agard D.A. (2004). *The 0.83 Å resolution crystal structure of alpha-lytic protease reveals the detailed structure of the active site and identifies a source of conformational strain.* J. Mol. Biol. **338**. 999-1013.

Sobolev O. V. & Lunin V. Y. (2008). *Unrestrained reciprocal space refinement of macromolecular structures as a tool to indicate alternative conformations*. Mathematical Biology and Bioinformatics, **3**(2), 50-59 (in Russian). URL: http://www.matbio.org/downloads/Sobolev2008(3_50).pdf

Sobolev O.V. & Lunin V.Y. (2012). *Detection of alternative conformations by unrestrained refinement*. Acta Cryst., D**68** (in press).

ERRASER, a powerful new system for correcting RNA models

Fang-Chieh Chou[a], Jane Richardson[b] and Rhiju Das[a]

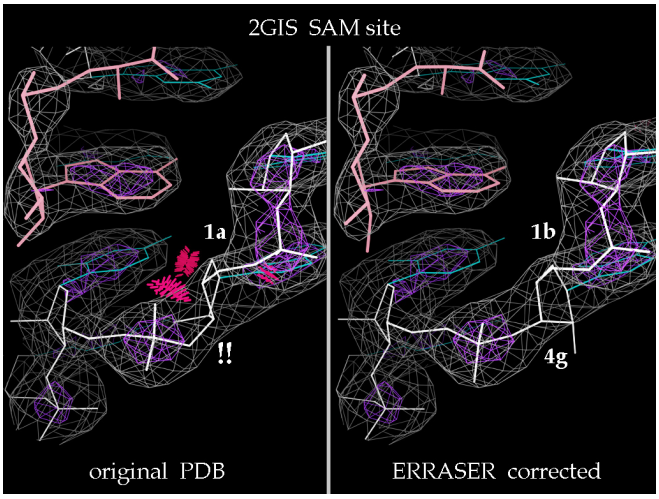
[a]*Stanford University, Stanford, CA*

[b]*Duke University, Durham, NC*

Correspondence email: rhiju@stanford.edu

PHENIX collaborators Rhiju Das and Fang Chou at Stanford have combined their full-enumeration local search called Step-Wise Assembly, or SWA, (Sripakdeevong 2010) with the Rosetta RNA energy function (Das 2010), Rosetta electron-density score (DiMaio 2009), MolProbity diagnosis, and Phenix refinement into an automated procedure called Erraser. The Duke team has evaluated this as being by far the most effective method yet for correcting fitting errors in RNA structures, especially aimed at the difficult backbone conformation and ribose pucker. ERASSER starts with refinement in PHENIX (`phenix.refine`), including backbone-specific RNA target parameters (Adams 2010), followed by Rosetta "relax" minimization (Leaver-Fay 2011). It then runs the `phenix.rna_validate` MolProbity-style validation (Chen 2010) to identify nucleotides with problems in ribose pucker or suite conformers (Richardson 2008), all-atom clashes, or covalent geometry. Those residues are then put through the exhaustive SWA sampling protocol to look for better conformations that match the density at least as well. SWA and Rosetta relax are cycled three times, and those results considered clear improvements are given a last round of refinement.

As reported in the on-line preprint (arXiv:1110.0276v2) of their paper (Chou 2012), for the 24-structure test set *R-free* improves and all-atom clashes, dubious ribose puckers, and poor backbone conformers are all reduced on average by factors of 2 to 8, while bond lengths and angles can then remain outlier-free in refinement. This works even at lower resolution -- for instance, the clashscores at 3.2-2.7Å resolution drop to values typical in PDB RNAs at 1.8Å. Some of the independently validated improvements are at critical binding or active sites. An example of coupled local correction of ribose pucker, conformers, and clashes is shown in the figure, at the `SAM-I` binding site of the `2GIS` riboswitch (Montange 2006).



ERRASER's success depends on its exhaustive enumeration of the possibilities, so that it is somewhat compute-intensive -- but not unreasonably so because the sampling is purely local. The user needs to install Rosetta as well as PHENIX. The scripts for running both the PHENIX refinements and the ERRASER cycles in Rosetta are included in the supplementary material for the in-press paper. ERRASER has also been integrated into PHENIX, and the users can run it under command line through the "`phenix.erraser`" application.

References

Adams PD, *et al.* (2010) PHENIX: a comprehensive Python-based system for macromolecular structure solution, *Acta Crystallogr* **D66**: 213-221.

Chen VB, *et al.* (2010) MolProbity: all-atom structure validation for macromolecular crystallography, *Acta Crystallogr* **D66**: 12-21.

Chou F-C, Sripakdeevong P, Dibrov SM, Hermann T, Das R (2012) Correcting pervaseive errors in RNA crystallography through enumerative structure prediction, *Nature Methods* (under revision).

Das, R., Karanicolas, J., Baker., D (2010) Atomic accuracy in predicting and designing noncanonical RNA structure, *Nature Methods* **7**:291-294.

DiMaio F, Tyka MD, Baker ML, Chiu W, Baker D (2009) Refinement of Protein Structures into Low-Resolution Density Maps Using Rosetta, *J Mol Biol* **392**: 181-190.

Leaver-Fay A, *et al*. (2011) ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules, *Methods Enzymol* **487**: 545-574.

Montange RK, Batey RT (2006) Structure of the S-adenosylmethionine riboswitch regulatory mRNA element, Nature 441: 1172-1175.

Richardson JS, *et al*. (2008) RNA backbone: Consensus all-angle conformers and modular string nomenclature (an RNA Ontology Consortium contribution). *RNA* **14**: 465-481.

Sripakdeevong P, Kladwang W, Das R (2010) An enumerative stepwise ansatz enables atomic-accuracy RNA loop modeling, *Proc Nat Acad Sci USA* **108**: 20573-20578.

cctbx tools for transparent job execution on clusters

Gábor Bunkóczi[a] and Nathaniel Echols[b]

[a]*Department of Haematology, University of Cambridge, Cambridge, UK*
[b]*Lawrence Berkeley National Laboratory, Berkeley, CA*

Correspondence email: gb360@cam.ac.uk

Introduction

With the availability of multi-core CPUs, parallelization is becoming increasingly common in software development for speeding up execution. Specialized supporting tools and libraries have been made available for various programming languages to help developers in writing parallel code. The Python programming language (http://www.python.org) provides the `threading` and `multiprocessing` modules in the Python Standard Library (http://docs.python.org/library/index.html), which allow parallel execution of calculations on a separate thread or process, respectively. These modules have proved very useful in shared-memory-based parallelization, but cannot harness the computational power provided by clusters without specialized code. For certain tasks in scientific computing, access to CPUs in clusters would boost productivity considerably, but due to the inherent heterogeneity in mechanisms of access and job control, it is not possible (or not practical) to maintain specialized code that would allow universal deployment. The current work describes a component that allows transparent access to an underlying submission queue, and enables execution in a unified way. It is available with the `libtbx` module of the Computational Crystallography Toolbox (`cctbx`, http://cctbx.sourceforge.net; Grosse-Kunstleve *et al.*, 2002), the open-source component of the *PHENIX* project (http://www.phenix-online.org; Adams *et al.*, 2010).

Goals

The `threading` and `multiprocessing` modules in the Python Standard Library provide the `Thread` and `Process` classes, respectively, to execute code concurrently with the main thread. Both of these classes share a common interface, and therefore code designed to use one of them would run with the other correctly. Providing a `Job` class that shares the same interface with the `Thread` and `Process` classes, and acts as a drop-in replacement, but submits the calculation to a queuing system, would enable a suitably written program to run seamlessly on clusters.

In addition, the need for another component arises from the fact that both `Thread` and `Process` rely on data channels to send the result of the computation back to the main thread (Figure 1). Such channels are provided by e.g. the `Queue.Queue` and `multiprocessing.Queue` classes. `Queue.Queue` is memory-based and is primarily useful in conjunction with `Thread`, because the data cannot cross a process boundary, while `multiprocessing.Queue` uses a (file or network) socket that allows inter-process data exchange, and could also be used with `Process`. Unfortunately, even the latter would not be suitable for the proposed `Job` class, because execution in general would take place on a different host, and makes the development of a novel `Queue` necessary.

It is clear at this stage that a fair amount of temporary data has to be written to the current directory. This needs to be done in a way that concurrent instances of the same program running in the same directory does not result in a race condition. As a non-functional goal, the number and size of temporary files should be kept at the necessary minimum.

Constraints

A completely general implementation of such functionality may be possible, but beyond available resources. For practical reasons, the only function calls supported are those that can be serialized with the `pickle` module. This includes all pure Python objects, Boost.Python (Abrahams & Grosse-Kunstleve, 2003) exported objects with serialization support, and named functions

```
from multiprocessing import Process, Queue

def f(q):
    q.put([42, None, 'hello'])

if __name__ == '__main__':
    q = Queue()
    p = Process(target=f, args=(q,))
    p.start()
    print q.get()    # prints "[42, None, 'hello']"
    p.join()
```

Figure 1 Example demonstrating typical use of the `multiprocessing` module (taken from http://docs.python.org/library/multiprocessing.html). Note that the result is returned via a `multiprocessing.Queue` instance that is passed to the call as an argument.

(http://docs.python.org/library/pickle.html#what-can-be-pickled-and-unpickled). Notable exceptions are `lambda`-functions and functions defined in the interactive interpreter.

Design

Analyzing the format of job submission commands and available options on queuing system we have access to, i.e. Sun Grid Engine (SGE, http://en.wikipedia.org/wiki/Oracle_Grid_Engine), Load Sharing Facility (LSF, http://en.wikipedia.org/wiki/Load_Sharing_Facility) and Portable Batch System (PBS, http://en.wikipedia.org/wiki/Portable_Batch_System), allowed us to formulate common and varying properties that can be exploited in the design:

- Command-line options are rarely conserved, but control similar behavior. These could be encapsulated so that the actual switches are hidden, but the behavior is exposed.
- All systems capture `stdout` and `stderr` from the process in files. Default naming convention varies, but it is possible to assign a name via command-line options.
- All systems provide a command-line option to change the default directory to the current one.
- The studied systems accept commands from `stdin`, i.e. rendering an initial (temporary) command file unnecessary. Unfortunately, this is not universally valid, e.g. when using Condor (http://en.wikipedia.org/wiki/Condor_High-Throughput_Computing_System), a command file and a Condor control file are required, and the documentation suggests that the command file has to be an actual file on the file system. This restriction, however, is specific to Condor, and it would not be productive to not take advantage of the read from `stdin` feature when it is available. Therefore, this part should be factored out in a component that has to be replaced for systems that do not support it, e.g. Condor.
- Passing input data to the job can be done by either writing out a temporary file, or by incorporating this data in the command file. The most efficient method depends on the size of input.
- SGE and LSF support synchronized mode (in addition to the more common asynchronous one) that has certain benefits when checking whether a job has completed or not (polling). This could be made available for queuing systems that support it by suitable encapsulation.

Since queuing systems all differ, actual details have to be passed to the `Job` constructor at instantiation. However, this changes the interface from that of `Thread` and `Process`, and makes it unsuitable for drop-in replacement. This can be solved by storing details of the queuing system in a handler class that has a method with an interface identical to `Thread` or `Process`. Varying execution details are then passed onto the `Job` constructor by the handler. The final design of the `processing` class is shown in Figure 2.

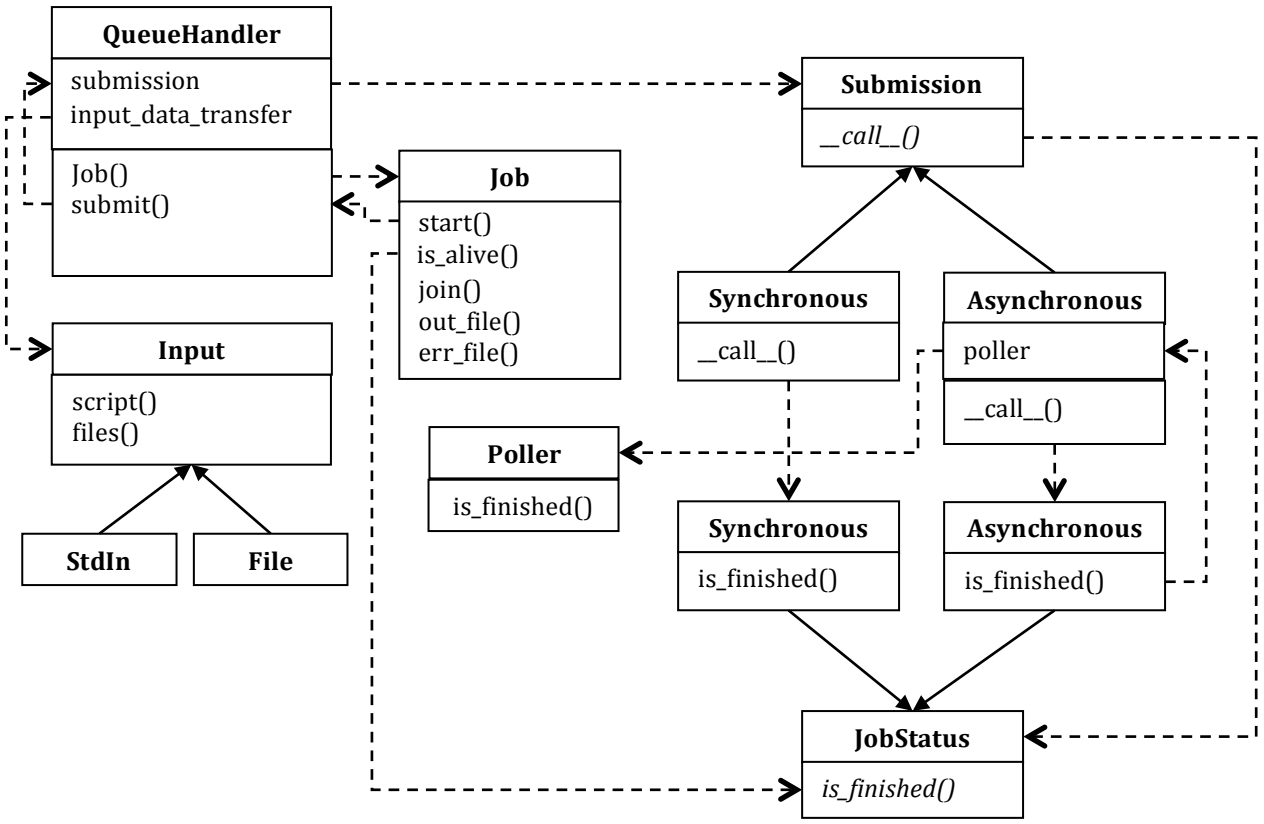


Figure 2 UML-style class diagram of components provided by the `processing` module.

Factoring the `Poller` out to a separate entity deserves an explanation. Polling for asynchronous jobs can be very slow, because it requires a system call to the appropriate queuing system command. This design allows a global `Poller` shared by all jobs, which requires less frequent updates than individual jobs would poll.

The data channel `processing.Queue` is implemented using temporary files that are placed in the current directory (that is assumed to be on a file system shared between the hosts). It mimics the interface of `Queue.Queue` and `multiprocessing.Queue`. Each instance is assigned a unique label, consisting of a root name, which is provided as a convenience for the user to know which program has created the temporary file, the process ID, and the object identifier (as returned by the Python `id` function); the latter two ensures the label is unique. This is used as the root of the file names. Temporary files are then named using the unique label and a sequence number, so that the channel could return multiple data, although this is not normally necessary. While this simple implementation suffices for most practical uses, this class is not functionally equivalent to `Queue.Queue` or `multiprocessing.Queue`, since those are multi-producer, multi-consumer channels, while `processing.Queue` is strictly single-producer and single-consumer. In addition, due to latency associated with network file systems, a generous timeout has to be allowed. Therefore, for functional and performance reasons, `processing.Queue` is not a generic replacement for `Queue.Queue` or `multiprocessing.Queue`, but rather should be employed when necessary, i.e. with the `processing.Job` class. In certain cases, a higher performance replacement is possible. MPI (message-passing interface) provides network-based inter-host communication that could be exploited. In addition, network-based channels between hosts can also be used in favorable cases, if there are no firewalls blocking access to incoming connections.

```
from libtbx.queuing_system_utils import processing

def f(q):
    q.put([42, None, 'hello'])

if __name__ == '__main__':
    q = processing.Queue( identifier = "tmp" )
    sge = processing.SGE()
    p = sge.Job(target=f, args=(q,))
    p.start()
    print q.get()    # prints "[42, None, 'hello']"
    p.join()
```

Figure 3 Example code from Figure 1 rewritten to run on SGE.

Coding

The details of the implementation in Figure 2 may appear daunting. However, only few applications will need to exploit the full flexibility provided by the design. For regular use, convenience constructors are provided for the queuing systems that are supported. These constructors also allow incorporation of user configuration, e.g. using a dedicated command for submitting with certain options, or extra parameters to direct the job to a certain queue. A simple example is shown in Figure 3.

Scheduling

Queuing systems manage the submitted jobs and balance the load over the allocated hosts, and make sure those are not overloaded. This is a very convenient feature that would often be useful for parallel programs. The `multiprocessing.Pool` class offers very similar functionality, but it is limited to handle `multiprocessing.Process` instances, and also provides no way for checking how many of the assigned CPUs are idle. Furthermore, in some workflows incoming results can make certain calculations unnecessary, and in this case it is important not to commit a calculation when there are no resources for execution. Therefore, we have implemented such functionality as a new module, `libtbx.scheduling`, so that it can be used independently from the `processing` module.

The class diagram is shown in Figure 4. `ExecutionUnit` encapsulates the actual execution mode. It holds a factory function that creates an execution class instance (`Thread`, `Process` or `QueueHandler.Job`). `Manager` objects hold and distribute the jobs onto `ExecutionUnits`. `Manager` provides a simple interface to submit jobs, checks jobs in various phases of execution (waiting, running, finished), and iterate through the results as they are produced or any particular order. A separate `Adaptor` class is provided to give access to the same `Manager` from various parts of the program, but keeping the jobs "private", so that a job submitted through an `Adaptor` is not accessible from other `Adaptor` instances, even if they use the same `Manager`. This allows more efficient design than by simply allocating resources to various parts of the program, because a single global resource can distribute load until all CPUs are busy, while with isolated `Managers` free CPUs from one instance cannot be reallocated to another.

The class diagram in Figure 4 suggests that the `Manager` allows `ExecutionUnits` with various types of execution classes to be used. However, when a job is submitted, the system will not know how the calculation will get executed, i.e. which execution class will be instantiated, because it is not predictable which `ExecutionUnit` will first complete the calculation it is currently running. Since a data channel has to be passed along with the calculation to return the result, and the type of the data channel depends on the execution class, this presents a problem. One solution would be to use the most general data

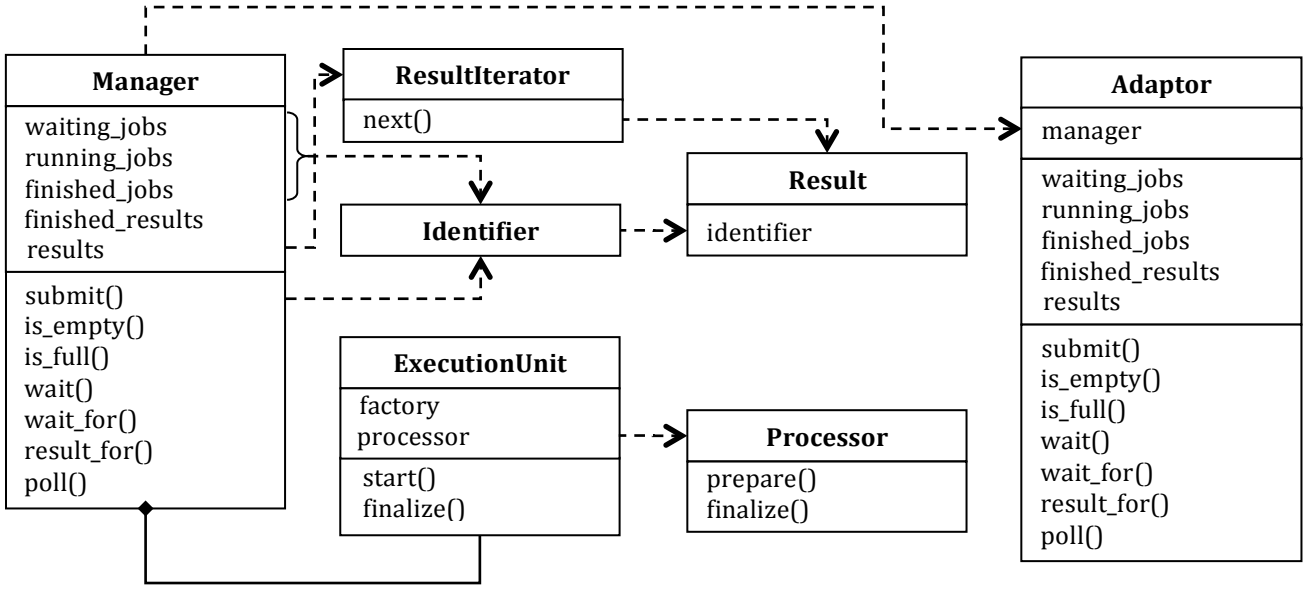


Figure 4 Class diagram of the `scheduling` module.

channel, i.e. the file-based `processing.Queue` that is implemented for `processing.Job`. However, this would represent a performance sink when `processing.Job` is not used (which may not be possible to know until run time). A better solution is to assign the appropriate data channel to the `ExecutionUnit`. This is possible using a suitable `Processor` class that bundles the call with the data channel, and extracts the results after the calculation has finished. Such functionality is available in the `RetrieveProcessor` class, which is provided with the `scheduling` module. Using a `RetrieveProcessor` may be slightly more efficient even when only uniform execution classes are used, because it allows the data channels to be reused. An example is shown in Figure 5.

By using a `Manager`, it is possible to detach execution from program logic entirely. This is desirable, because the client code need not be aware, how the result is obtained. After instantiation, the `Manager` object can be used uniformly, without reference to the number and type of execution classes. In addition, it is also possible to use main-thread-only (non-parallel) execution, with little performance overhead. In principle, using a single `Thread` or `Process` class would be an acceptable solution, but a slightly more efficient way is to define an "execution factory function" that performs the function call, and avoids job startup overhead.

Current uses

The molecular replacement pipeline `phaser.MRage` is based on a `scheduling.Manager` for handling parallel calculations. Depending on the molecular replacement search, `phaser.MRage` can use a large number of CPUs, and therefore accessing clusters can boost performance substantially. Encapsulating the execution mode was a major factor that allowed enabling additional execution modes incrementally. Currently, besides the Python standard library modules `multiprocessing` and `threading`, SGE, LSF and PBS are supported, but additional queuing systems can be added on request.

Acknowledgements

The first version of this module was based on `libtbx.queuing_system_utils.lsf`, written by Joel Bard. We thank Robert Nolte and Herbert Klei for suggestions.

```
from libtbx.queuing_system_utils import scheduling
import multiprocessing
import threading
import Queue

def f(value):
    return value

manager = scheduling.Manager(
    units = (
        [ scheduling.ExecutionUnit(
            factory = multiprocessing.Process,
            processor = scheduling.RetrieveProcessor(
                queue = multiprocessing.Queue(),
                ),
            )
        for i in range(2) ]
        + [ scheduling.ExecutionUnit(
            factory = threading.Thread,
            processor = scheduling.RetrieveProcessor(
                queue = Queue.Queue(),
                ),
            )
        for i in range(3) ]
        )
    )
adapter1 = scheduling.Adapter( manager = manager )
adapter2 = scheduling.Adapter( manager = manager )

for i in range(10):
  adapter1.submit( target = f, args = (i,) )

for i in range(10, 20):
  adapter2.submit( target = f, args = (i,) )

# after a short delay: print numbers from 10 to 19 in some order
# (adapter1 jobs are processed first)
print [ i.result for i in adapter2.results ]

# immediately: print numbers from 0 to 9 in some order
# (these jobs have already been completed)
print [ i.result for i in adapter1.results ]
```

Figure 5 Example code demonstrating the `scheduling` module.

References

Abrahams, D. & Grosse-Kunstleve, R. W. (2003). *C/C++ Users Journal* **21**, 29-36.

Adams, P. D., Afonine, P. V., Bunkoczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L. W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). *Acta Crystallogr D* **66**, 213-221.

Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *Journal of Applied Crystallography* **35**, 126-136.

On the analysis of residual density distributions on an absolute scale

Pavel V. Afonine[1], Alexandre Urzhumtsev[2,3] and Paul D. Adams[1,4]

[1]*Lawrence Berkeley National Laboratory, One Cyclotron Road, BLDG 64R0121, Berkeley, CA 94720 USA*
[2]*IGBMC, CNRS-INSERM-UdS, 1 rue Laurent Fries, B.P.10142, 67404 Illkirch, France*
[3]*Université de Lorraine, Physics Department, Vandoeuvre-lès-Nancy, 54506 France*
[4]*Department of Bioengineering, University of California Berkeley, Berkeley, CA, 94720, USA*

Correspondence e-mail: pafonine@lbl.gov, sacha@igbmc.fr

Preamble

This newsletter article was inspired by figure 1 in the B.W. Matthews' paper (Matthews, 2009). The figure shows averaged residual density distributions as a function of distance from an atom. Unexpectedly there is substantial negative density in the vicinity of the atomic centers and substantial positive density in the bulk-solvent region. This became a subject of our analysis described below.

Calibrating expectations

Analysis of electron density on an absolute scale (i.e. in units of e/Å^3) requires the value of F000 reflection and the normalization (division) of Fourier transform of the corresponding map coefficients by the unit cell volume. While F000 is never measured directly in the diffraction experiment, it can be estimated. One way is to measure the crystal average density $\rho_{average}$ and then estimate

$$F000 \approx \rho_{average} V_{cell} \qquad (1)$$

where V_{cell} is the unit cell volume. If the macromolecular structure is already known, then another alternative is to calculate it as

$$F000 = F000_{calc} + F000_{bulk} \qquad (2)$$

where $F000_{calc}$ is the sum of the scattering factors at zero diffraction angle calculated over all atoms in the unit cell (which includes the molecule itself and all the symmetry mates) and $F000_{bulk}$ is the bulk-solvent contribution $k_{sol}V_{sol} = k_{sol}V_{cell}x$, where k_{sol} and x are the average density of the bulk-solvent and its volume fraction, correspondingly. This obviously requires knowing k_{sol}, which can be estimated as described in Fokine & Urzhumtsev (2002) and Afonine *et al.* (2005), though may not be adequate (Afonine *et al.*, 2012, in preparation). Yet another approach relies on the assumption that residual electron density around reliably placed atoms should be zero, which in turn defines

$$F000 \approx -\rho_{average,\ selected\ atoms} V_{cell} + F000_{calc} \qquad (3)$$

where $\rho_{average,\ selected\ atoms}$ is the average density around reliably placed atoms obtained using a Fourier synthesis calculated without F000.

To test the applicability of formula (3) we performed the following numerical experiment. For four selected structures from the PDB (Berman *et al.*, 2000), `2x8h`, `3ahs`, `3m8u` and `3krr`, we simulated experimental data as $F_{obs} = |\mathbf{F}_{model}| = k_{total}|\mathbf{F}_{calc} + \mathbf{F}_{bulk}|$, where $\mathbf{F}_{bulk}$ was computed using a flat bulk-solvent model with an average density of 0.35 e/Å^3. These calculations were performed using the `phenix.fmodel` program (Afonine, unpublished). For each test case two sets of data were simulated: one is a 100% complete set up to the highest measured resolution (F^f_{obs}) and the second set (F^i_{obs}) corresponds to the set of Miller indices of actually measured reflections, which may be incomplete. Next we calculated four Fourier syntheses on an absolute scale with ($\Delta\rho_1$, $\Delta\rho_2$, $\Delta\rho_4$), and without ($\Delta\rho_3$), accounting for F000:

$$\Delta\rho_1 = \{F^f_{obs}/k_{total}, \varphi_{model} - F_{calc}, \varphi_{calc}\ ;\ \Delta F000\}$$

$$\Delta\rho_2 = \{F^f_{obs}/k_{total} - F_{calc}, \varphi_{calc}\ ;\ \Delta F000\}$$

$$\Delta\rho_3 = \{F^f_{obs}/k_{total} - F_{calc}, \varphi_{calc}\}$$

$$\Delta\rho_4 = \{F^i_{obs}/k_{total} - F_{calc}, \varphi_{calc}\ ;\ \Delta F000\}$$

and plotted averaged density values calculated as described by Matthews (2009) (figure 1). Here

$$\Delta F000 = F000_{obs} - F000_{model} = F000_{bulk} \approx -\rho_{average,\ selected\ atoms} V_{cell}.$$

The first synthesis ($\Delta\rho_1$) is obviously expected to show the distribution of bulk-solvent: nearly zero density levels around atomic centers, constant density in the bulk solvent region with a level of 0.35 e/Å^3, and a smooth range of values at the solvent-macromolecule boundary. Figure 1 confirms this expectation for all four structures.

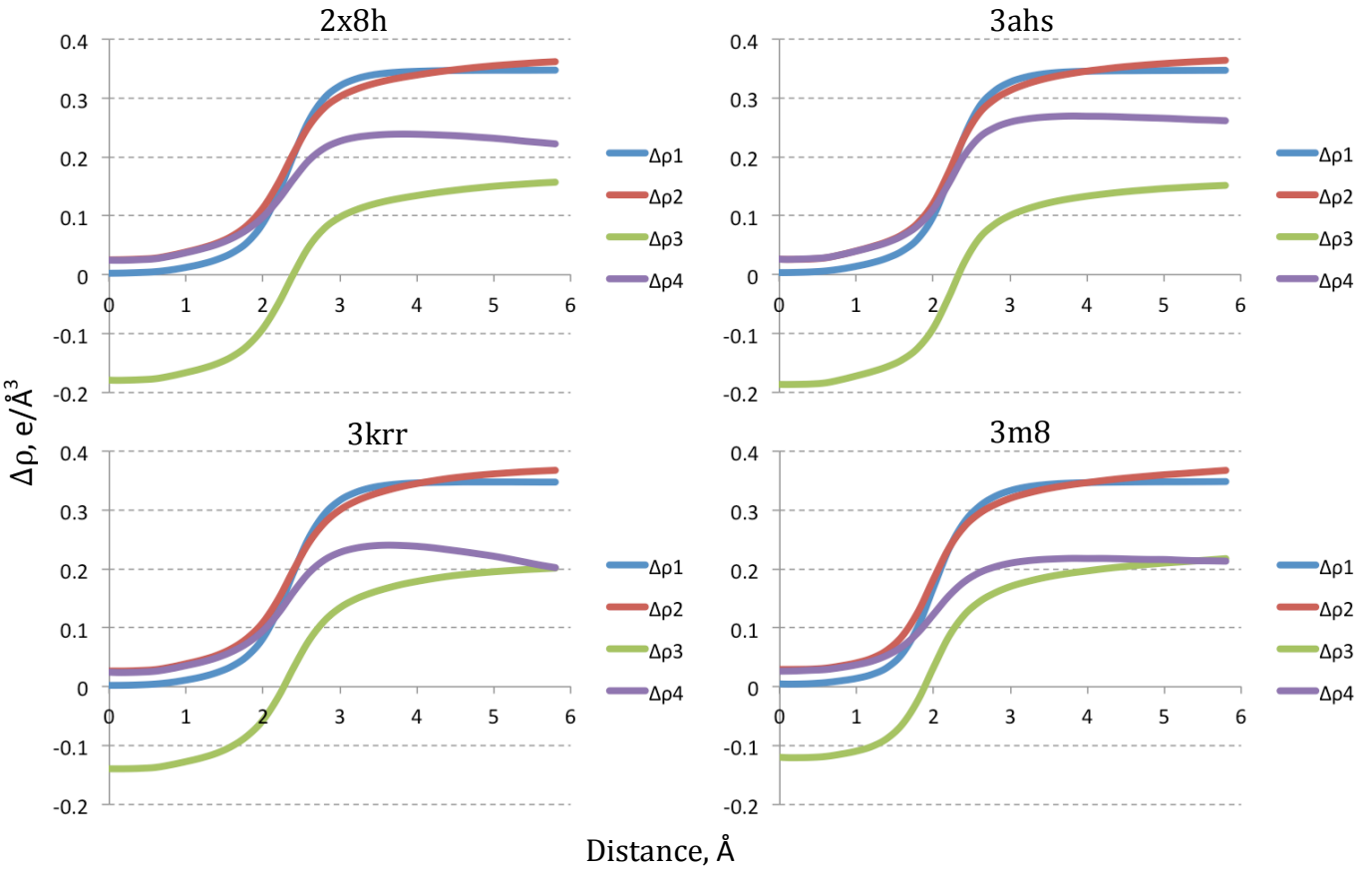


Figure 1. Average residual electron density as a function of the distance from the nearest atom (for details, see Matthews, 2009). Simulated data; see text for notations.

The second synthesis ($\Delta\rho_2$) differs from $\Delta\rho_1$ in that it is calculated with less accurate phases, corresponding to the atomic model only. This is expected to disturb the synthesis in some way but perhaps not significantly. Indeed figure 1 shows some positive density values in the vicinity of atoms and a not perfectly constant (0.35 e/Å^3) level of density in the solvent region away from the macromolecule.

The third synthesis ($\Delta\rho_3$) is equivalent to $\Delta\rho_2$ but calculated without ΔF000 term. The shape of this synthesis is very similar to that of $\Delta\rho_2$ but the values are shifted by a constant, resulting in substantially negative density around atoms and reduced density values in the bulk-solvent region (figure 1).

Finally the fourth synthesis ($\Delta\rho_4$) is equivalent to $\Delta\rho_2$ but calculated using the incomplete set F^i_{obs}. The density distribution is expected to be further distorted compared to $\Delta\rho_1$ and $\Delta\rho_2$ (figure 1), and the degree of distortion is a function of the number and type of missing reflections. The analysis of how missing reflections affect this distribution is beyond the scope of this article.

Results

As we now know how the density distribution in question appears in ideal or near-to-ideal settings we can meaningfully analyze the analogous syntheses calculated using real measured data, F_{obs} (as opposed to simulated data used above). For this we calculated two syntheses for each of four models:

$$\Delta\rho_5 = \{F_{obs}/k_{total} - F_{calc}, \varphi_{calc} ; \Delta F000\}$$

$$\Delta\rho_6 = \{F_{obs}/k_{total} - F_{model}, \varphi_{model} ; \Delta F000\}$$

and plotted averaged density values calculated as described by Matthews (2009) (figure 2). Here $\Delta F000 = -\rho_{average,\ selected\ atoms} V_{cell}$ and is not necessarily equal to $F000_{bulk}$ but also accounts for other missing scattering in the model.

The bulk-solvent-omit synthesis ($\Delta\rho_5$) overall

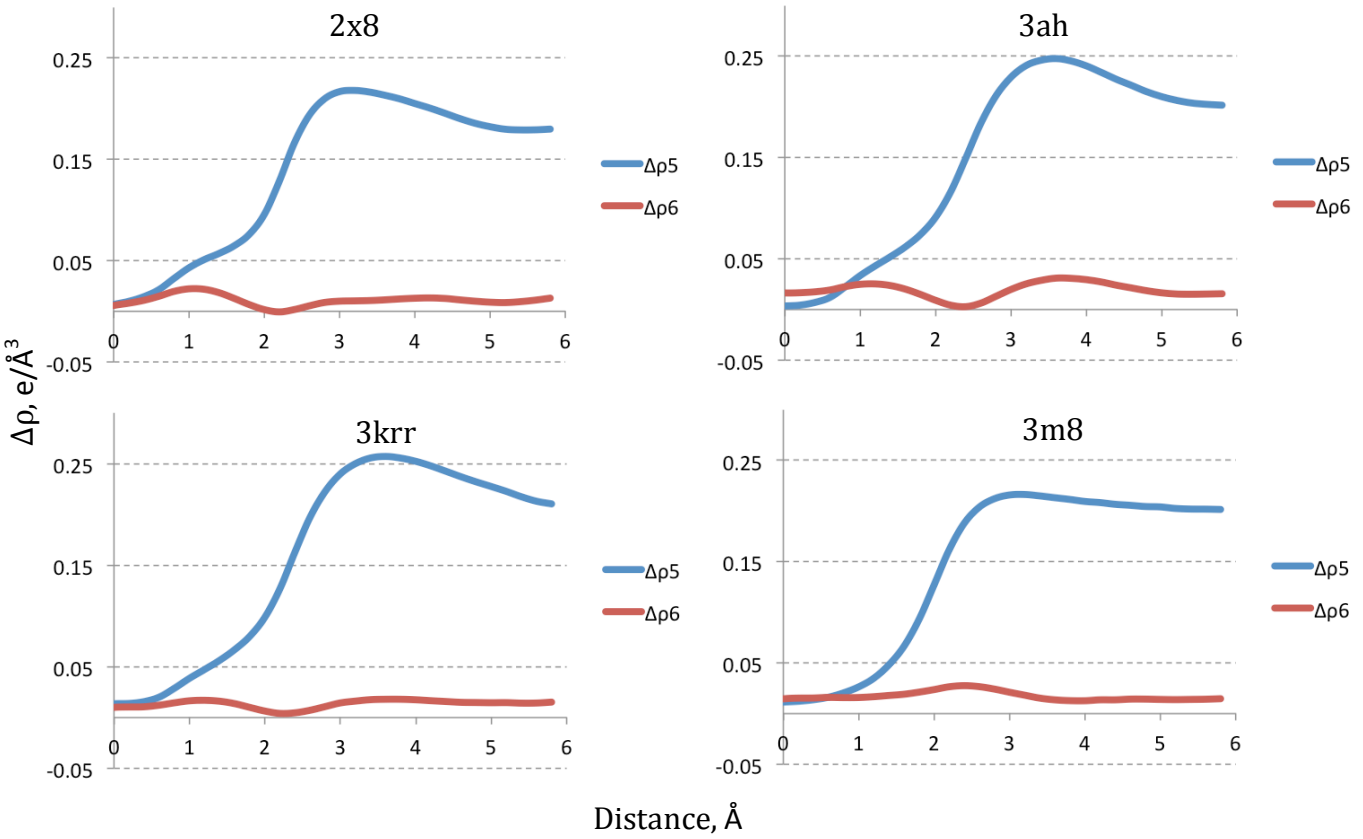


Figure 2. Average residual electron density as a function of the distance from the nearest atom (for details, see Matthews, 2009). See text for notations.

resembles the expected curve (e.g. compare with $\Delta\rho_4$). The differences may be attributed to the errors in data and model parameters, model incompleteness (missing atoms), and deviation of the flat bulk-solvent model used to simulate F^i_{obs} from the real bulk-solvent distribution. One can postulate that a peak at approximately 3.5Å in the $\Delta\rho_5$ synthesis could arise from partially structured weakly bound and therefore unmodeled solvent. However, one can notice the same behavior (though less pronounced) in the $\Delta\rho_4$ syntheses for all four models, where no such partially structured solvent was included in the calculation of the data. Indeed, the peak positions in $\Delta\rho_5$ (3.6, 3.2, 3.2 and 3.6 Å) are quite similar to those in $\Delta\rho_4$ (3.8, 3.8, 3.8 and 3.6 Å) for 2x8h, 3ahs, 3m8u and 3krr correspondingly. This may be attributed to Fourier truncation artifacts, though a conclusive answer would require further analysis.

The nearly flat residual synthesis, $\Delta\rho_6$, with a density level significantly lower than the bulk-solvent value suggests that on average there is no systematically missing features left to model in the bulk-solvent region. However, since these density distributions are the average over all atoms and grid points of the map in the unit cell, they may not capture local features that are yet to be accounted for.

The residual density distribution shown in figure 1 in Matthews (2009) (red bars) closely resembles the $\Delta\rho_3$ distribution shown above, which suggests that the ΔF000 reflection was not fully accounted for, as would be required for map calculations on an absolute scale.

References

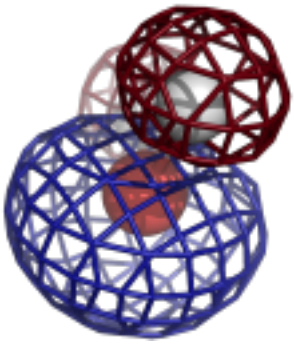
Afonine, P., Grosse-Kunstleve, R.W., & Adams, P.D. (2005). *Acta Cryst.*, **D61**, 850-855.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. & Bourne, P.E. (2000). *Nucleic Acids Research*. **28**, 235-242.

Fokine, A & Urzhumtsev, A. (2002). *Acta Cryst.*, **D58**, 1387-1392.

Matthews, B.W. (2009). *Protein Sci.*, **18**, 1135-1138.

VOLUME FOUR



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXIII

HYDROGENS, PARALLELISM, SBKB TECH PORTAL

Table of Contents

- PHENIX News 1
- Crystallographic meetings 2
- Expert Advice
  - Fitting tips #5 - What's with water 2
- FAQ
  - I'm seeing a lot of CIF files. Are they all restraints? 6
- Short Communications
  - Phenix / MolProbity Hydrogen Parameter Update 9
  - PSI SBKB Technology Portal: A Web Resource for Structural Biologists 11
- Articles
  - *cctbx* tools for transparent parallel job execution in Python. I. Foundations 16
  - *cctbx* tools for transparent parallel job execution in Python. II. Convenience functions for the impatient. 23

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New features

New AutoBuild features

AutoBuild now has a set of parameters for building from very accurate but very small parts of a model. You can now use the keyword `rebuild_from_fragments=True` to start rebuilding from fragments of a model. You might want to use this if you look for ideal helices using Phaser and then rebuild the resulting partial model, as in the Arcimboldo procedure. The special feature of finding helices is that they can be very accurately placed in some cases. This really helps the subsequent rebuilding. If you have enough computer time, then run it several or even many times with different values of `i_ran_seed`. Each time you'll get a slightly different result.

New MR_Rosetta features

You can now give commands for Rosetta in `mr_rosetta`, including a command to specify where disulfide bonds are located (provided you have a version of Rosetta that can handle these commands!). Also the output Rosetta models are now identified by an ID number so that they have unique names. To avoid running too many jobs, the default number of homology models to download is now 1. Also the default number of NCS copies (if `ncs_copies=Auto`) is now the number leading to solvent content closest to 50%; if `ncs_copies=None` then all plausible values of `ncs_copies` are still tested.

Base-pairing is now implemented in RNA building.

`phenix.autobuild` will now try to guess which bases in a model are base-paired, and if there is no positive sequence match to the model, the bases that are base-paired will be chosen to be complementary. You can set the cutoff for base pairing with the keyword `dist_cut_base`.

Morphing

A new feature in `phenix.morph_model` and `phenix.autobuild` morphing allows you to specify that only main-chain and c-beta atoms are to be used in calculating the shifts for morphing. The keyword to do this is "`morph_main=True`".

Output models

In `phenix.autobuild` and `phenix.autosol` waters are now automatically named with the chain of the closest macromolecule if you set `sort_hetatms=True`. This is for the final model only. You can also supply a target position for your model with `map_to_object=my_target.pdb`. Then at the very end of the run, your molecule will be placed as close to this as possible. The center of mass of the `autobuild` model will be superimposed on the center of mass of `my_target.pdb` using space group symmetry, taking any match closer than 15 A within 3 unit cells of the original position. The new file will be `overall_best_mapped.pdb`

Crystallographic meetings and workshops

The West Coast Protein Crystallography Workshop, Monterey, CA, March 17- 20, 2013

The web site is wcpcw.org and a number of PHENIX developers will be in attendance.

43rd Mid-Atlantic Macromolecular Crystallography Meeting, Duke University, Durham NC, May 30-June 1, 2013

The website is: http://www.mid-atlantic.org/ and information is available from Jeffrey Headd (jeffrey.headd@duke.edu).

International Conference on Structural Genomics 2013 "Structural Life Science" Sapporo, Japan, July 29-August 1, 2013

The contact information is web site:
http://www.c-linkage.co.jp/ICSG2013
Contact: Katsumi Maenaka
email: maenaka@pharm.hokudai.ac.jp

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Seminar, Bates College, Lewiston, ME, July 26-27, 2014

A seminar series preceding the GRC meeting.

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Conference, Bates College, Lewiston, ME, July 27-August 1, 2014

A very interesting and important meeting for protein crystallographers.

Expert advice

Fitting Tip #5 – What's with water

Jeff Headd and Jane Richardson, Duke University

In X-ray crystallography maps, small, disconnected peaks of electron density are routinely filled with water molecules, either through automated water-picking or hand fitting, either of which may champion lower R factors over chemistry-supported atom placement. Many of these density regions are not, in fact, waters, and through careful analysis of the local environment of the peak, a more reasonable model can be built. Local sterics and electrostatics are often key indicators of incorrectly fit waters, and can provide clues as to what a more likely candidate atom or atoms could be. Here are a few examples of density peaks that have been incorrectly fit with a water molecule, and ways to arrive at a more plausible model.

Alternate conformations:

Density peaks for un-modeled alternate sidechain conformations are a common map

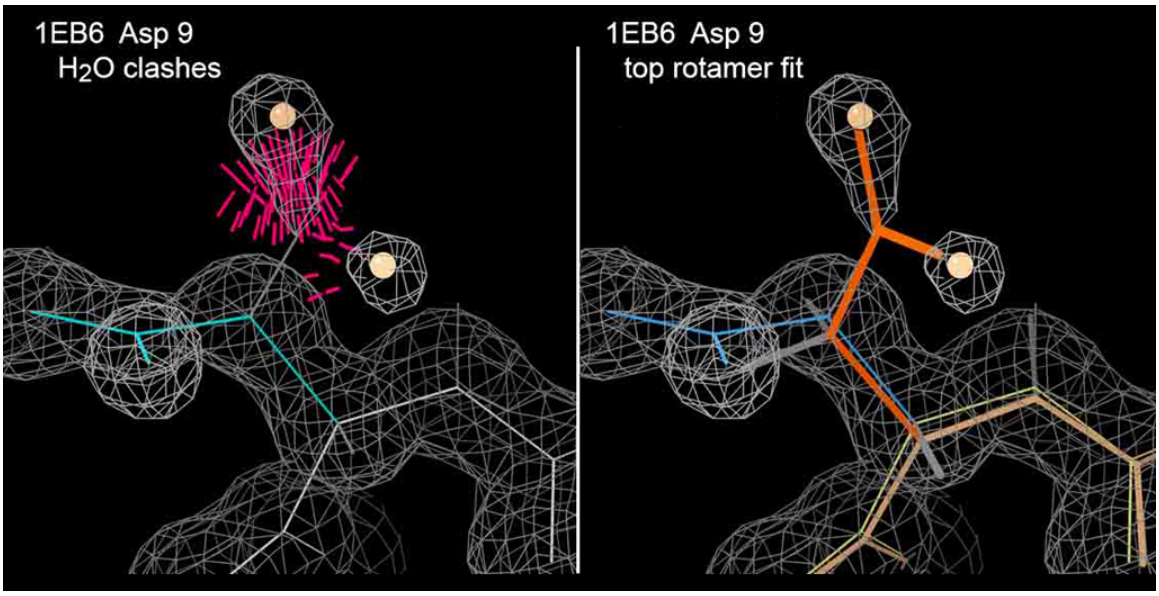


Figure 1

feature that may be erroneously filled with water molecules. In Fig. 1 (left panel), Asp 9 of 1EB6 is fit as a **t70** rotamer. The two nearby waters are reasonable fits to their respective density peaks, but both have significant steric clashes with the hydrophobic C-beta hydrogen. Rather than two waters, an alternate conformation for Asp A 9 in an **m-20** rotamer fits the two density peaks very well (right panel), and is sterically viable. Note that the backbone must be altered slightly to accommodate the **m-20** alternate through a "backrub" motion (Davis 2006), as discussed in Fitting Tip #4 in CCN, July, 2012.

Unidentified ions:

Another common situation where waters are placed incorrectly is a density peak that is better explained by an ion. In Fig. 2A, taken from the SECSG target Pfu-1218608 (Arendall 2005), the depicted water molecule is an excellent fit to the density peak, but analysis of the local steric environment reveals significant steric overlaps with the three

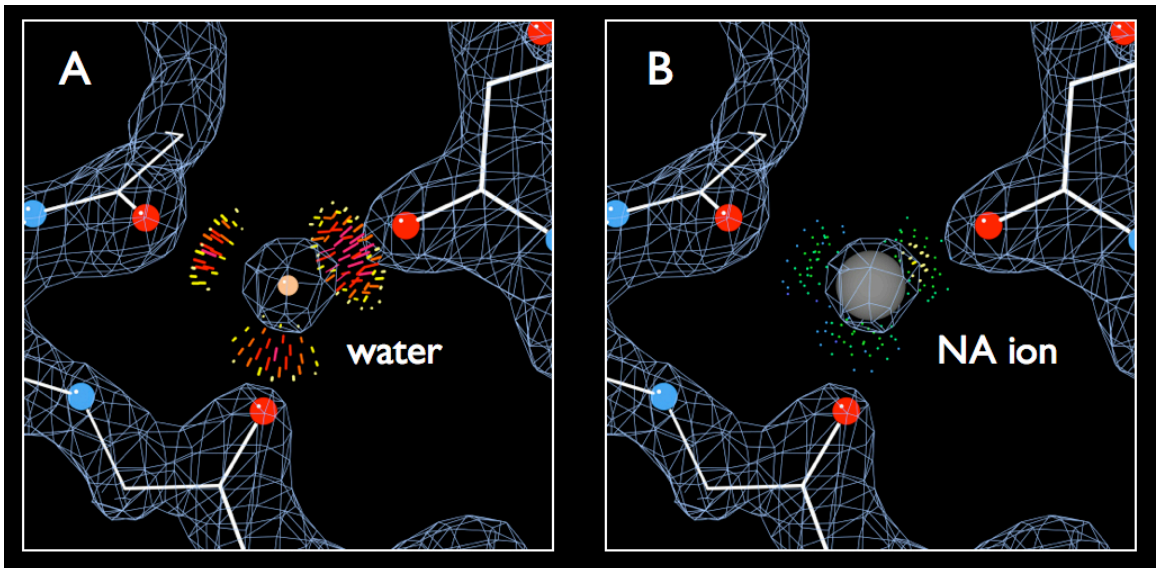


Figure 2

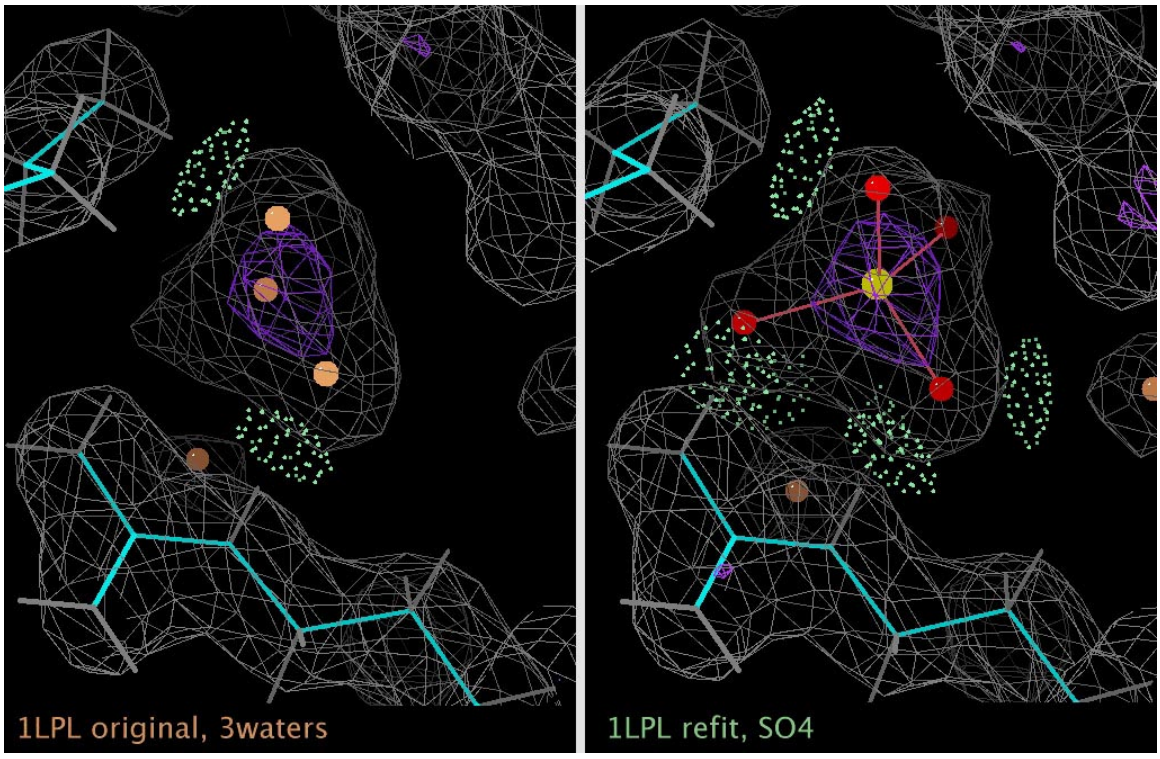


Figure 3

neighboring carbonyl oxygen atoms. The negative charge of this local area is an excellent candidate for a positively charged ion, however, and excellent packing is achieved with the placement of a sodium ion (Fig. 2B; 1NNW). It's also important to keep in mind that if anomalous data were collected, the presence of an anomalous peak at the same location is a strong indicator that the peak is not a water, and a search for a possible anomalous scatterer should be considered.

Unidentified ligands:

Ligand density is another example of density that is often erroneously fit with water molecules. Regions of continuous electron density that cannot be attributed to the primary crystallographic molecules are often filled with one or more water molecules. Such arrangements may slightly improve R-factors, but do not make much chemical or steric sense. For example, Fig. 3 (left panel) shows three water molecules placed in a tetrahedral-shaped density peak from PDBID: 1LPL. The local hydrogen-bonding environment does not offer much support for this arrangement of water molecules, but an SO4 molecule fits quite well, and is much more consistent with the local hydrogen-bonding network (now redeposited as 1TOV). Careful consideration of a given crystal's crystallization conditions can offer candidate ligands that may fit unidentified density regions.

Noise peak:

In some cases, waters are fit to density peaks that are simply noise. Due to errors in data collection, phasing, Fourier artifacts, and other sources of error, some small density peaks are simply just noise. Automated water-picking methods may place water atoms in these peaks, but such peaks are often too small, amorphous, and lack the expected electrostatic environment to support an ordered water molecule.

Real water:

Finally, it's also possible that a water peak is actually a water molecule! If the water forms hydrogen bonds and fits sterically in the region, it's very likely to be a correct ordered water.

Future development:

To supplement the water picking method in phenix.refine, automatic identification and building of other structural features (alternate conformations, ligands, etc.) is an active focus of Phenix development. Ion placement will be available in the near future.

References:

Davis IW, Arendall WB III, Richardson JS, Richardson DC (2006). "The backrub motion: How protein backbone shrugs when a sidechain dances" *Structure* **14**:265-274.

Arendall WB III, Tempel W, Richardson JS, Zhou W, Wang S, Davis IW, Liu Z-J, Rose JP, Carson WM, Luo M, Richardson DC, Wang B-C (2005). "A test of enhancing model accuracy in high-throughput crystallography." *Journal of Structural and Functional Genomics* **6**:1-11.

Contributors

P. D. Adams, H. M. Berman, G. Bunkóczi, V. B. Chen, L. N. Deis, N. Echols, M. J. Gabanyi, L. K. Gifford, R. J. Gildea, R. W. Grosse-Kunstleve, J. J. Headd, N. W. Moriarty, M. H. Prisant, D. C. Richardson, J. S. Richardson, J. Snoevink, T. C. Terwilliger, V. Verma, L. L. Videau

FAQ

This issue's FAQ grow to such an extent that it merits its own section. See next page.

I'm seeing a lot of CIF files. Are they all restraints?

Richard J. Gildea[a], Nigel W. Moriarty[a] and Paul D. Adams[a,b]
[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: pdadams@lbl.gov

Introduction

The acronym CIF is used both to refer to the *Crystallographic Information File*[1], and the *Crystallographic Information Framework*. CIF is designed to be a human and machine-readable text format consisting of pairs of data names and data items (with a looping facility that allows repeated items). The IUCr maintains a central repository of application-specific machine-readable CIF dictionaries, containing a collection of standard data names along with definitions and permissible values. The formal specification of the CIF format and data name dictionaries can be found at the IUCr website[2]. The macromolecular CIF dictionary (mmCIF) contains those data names that are necessary to describe the results of a macromolecular crystallographic experiment.

As a flexible and extensible format, CIF is used to record a variety of information in different contexts. Phenix[3] programs are capable of reading and writing several different types of CIF files. Here we present a summary of the various types of CIF file that the typical user of a macromolecular refinement program may encounter.

Restraint files

Restraint CIF files contain the minimum information necessary for a refinement program to restrain the internal coordinates of a given monomer to chemically meaningful values. The CCP4 monomer library[4] and the GeoStd[5] contain a complete description of the internal chemical structure for the most common monomers, including ideal target values for bonds, angles and torsion angles, and also definitions of planarity and chirality if required.

Both libraries also contain other pieces of useful information including common residue synonyms and common links such as the bonds for inter-residue restraints. In addition, the GeoStd contains the rotamer information for the common amino acids.

Generation of restraint files in Phenix is handled by eLBOW[6], which is called by a number of other programs including ReadySet! and the ligand pipeline.

Chemical components files

The PDB chemical components dictionary[7] describes all monomers and small molecule components found in PDB entries. Each chemical component entry is referenced by the 3-character alphanumeric code assigned to it by the PDB, and contains detailed chemical information and idealized coordinates. Chemical components files can be used as input to `phenix.elbow` in order to generate restraints (in CIF format) for refinement. A chemical components file may look superficially similar to a restraints file, however it lacks the necessary information on target values for restraints that is needed by the refinement program.

Coordinate files

Currently, the PDB format[8] is the most widely used file format for storing the atom site coordinates of macromolecular structures. Alternatively, it is possible to record the atom site coordinates (and much more) in CIF format files, with both formats accepted for PDB deposition. The atomic coordinates in both PDB and CIF files can be obtained from the PDB database using the command

```
phenix.fetch_pdb 1xxx --all
```

This will download the following files:

```
1xxx.pdb (atomic coordinates in PDB format)
1xxx.cif (atomic coordinates in CIF format)
1xxx-sf.cif (reflection data in CIF format)
1xxx.fa (sequence file in FASTA format)
```

Coordinate CIF files can be identified by the presence of the `atom_site` loop containing information about the atom sites, similar to that stored in the PDB ATOM record: element type, Cartesian or fractional site coordinates, atomic B-factor, occupancy, atom name, residue name, alternate location identifier, residue sequence number, insertion code, chain ID, etc. Look for the presence of a section that looks something like this (the order and exact data items present may vary depending on the source of the file):

```
loop_
  _atom_site.group_PDB
  _atom_site.id
  _atom_site.label_atom_id
  _atom_site.label_alt_id
  _atom_site.label_comp_id
  _atom_site.auth_asym_id
  _atom_site.auth_seq_id
  _atom_site.pdbx_PDB_ins_code
  _atom_site.Cartn_x
  _atom_site.Cartn_y
  _atom_site.Cartn_z
  _atom_site.occupancy
  _atom_site.B_iso_or_equiv
  _atom_site.type_symbol
  _atom_site.pdbx_formal_charge
  _atom_site.label_asym_id
  _atom_site.label_entity_id
  _atom_site.label_seq_id
  _atom_site.pdbx_PDB_model_num
```

Coordinate CIF files typically also contain a description of the unit cell and crystallographic symmetry. Frequently, they also describe experimental details, similar to those present in the PDB REMARK records, but presented in a much more computer-readable format. Coordinate mmCIF files can now be used in place of PDB-format coordinate files in most Phenix programs (e.g. `phenix.refine`[9], `phenix.model_vs_data`[10]), and `phenix.refine` will output the coordinates in CIF format given the option `write_model_cif_file=True`.

Reflection data files

Users of reflection data from the PDB will be familiar with the mmCIF format reflection files, most probably as an additional step where they need to first convert the reflection data to MTZ format, e.g. using phenix.cif_as_mtz, before being able to use the reflection data with their favorite programs. For some time, Phenix has supported the direct use of mmCIF format reflection files in place of MTZ files in many programs, although it may still be preferable to use an MTZ file for normal workflow, as the binary MTZ file format is inherently faster for file reading, particularly for large datasets. `phenix.refine` will output the reflections in CIF format given the option `write_reflection_cif_file=True`. An mmCIF format reflection file will contain a section similar to this:

```
loop_
  _refln.index_h
  _refln.index_k
  _refln.index_l
  _refln.F_meas_au
  _refln.F_meas_sigma_au
  _refln.pdbx_FWT
  _refln.pdbx_PHWT
  _refln.pdbx_DELFWT
  _refln.pdbx_DELPHWT
  _refln.status
```

Depending on the source of the file and the type of reflection data contained, the order and exact items present may vary, however the columns containing the miller indices, `_refln.index_h`, etc., must be present.

Small molecule CIF files

Coordinate and reflection data files obtained from the Cambridge Structural Database (CSD) or the Crystallography Open Database (COD) may look similar to the files described above, however these files use a slightly different set of data names based on an older version of the dictionary definition language (DDL) that describes the relationships between data names. Small molecule CIF files are not currently utilized by any Phenix program.

References

[1]S. R. Hall, F. H. Allen and I. D. Brown, *Acta Cryst.* (1991). A**47**, 655-685

[2]http://www.iucr.org/resources/cif

[3]P. D. Adams, P. V. Afonine, G. Bunkóczi, V. B. Chen, I. W. Davis, N. Echols, J. J. Headd, L.-W. Hung, G. J. Kapral, R. W. Grosse-Kunstleve, A. J. McCoy, N. W. Moriarty, R. Oeffner, R. J. Read, D. C. Richardson, J. S. Richardson, T. C. Terwilliger and P. H. Zwart (2010). *Acta Cryst.* D**66**, 213-221.

[4]A. A. Vagin, R. A. Steiner, A. A. Lebedev, L. Potterton, S. McNicholas, F. Long and G. N. Murshudov, (2004). *Acta Cryst.* D**60**, 2184-2195.

[5]http://geostd.sourceforge.net/

[6]N. W. Moriarty, R. W. Grosse-Kunstleve and P. D. Adams, *Acta Cryst.* (2009). D**65**, 1074-1080.

[7]http://www.wwpdb.org/ccd.html

[8]http://www.wwpdb.org/documentation/format33/v3.3.html

[9]P. V. Afonine, R. W. Grosse-Kunstleve, N. Echols, J. J. Headd, N. W. Moriarty, M. Mustyakimov, T. C. Terwilliger, A. Urzhumtsev, P. H. Zwart and P. D. Adams. *Acta Cryst.* 2012,**D68**:352-367

[10]P. V. Afonine, R. W. Grosse-Kunstleve, V. B. Chen, J. J. Headd, N. W. Moriarty, J. S. Richardson, D. C. Richardson, A. Urzhumtsev, P. H. Zwart and P. D. Adams. *J. Appl. Cryst.* (2010). **43**, 669-676.

Phenix / MolProbity Hydrogen Parameter Update

Lindsay N. Deis[a], Vishal Verma[b], Lizbeth L. Videau[a], Michael G. Prisant[a], Nigel W. Moriarty[c], Jeffrey J. Headd[a], Vincent B. Chen[a], Paul D. Adams[c,d], Jack Snoeyink[b], Jane S. Richardson[a], and David C. Richardson[a]

[a]*Duke University, Durham, NC*

[b]*University of North Carolina, Durham, NC*

[c]*Lawrence Berkeley National Laboratory, Berkeley, CA*

[d]*Bioengineering Department, University of California Berkeley, Berkeley, CA*

Correspondence email: lindsay.deis@duke.edu

The new distribution of PHENIX incorporates major updates in the parameters and procedures for hydrogen atoms, providing consistency between *phenix.refine* and MOLPROBITY and more correct treatment in each system across the range of usage needs.

Most distances between bonded atoms were settled long ago to high accuracy, but, in the case of hydrogens, the values in common use often differ by as much as 20%. This is primarily because X-ray diffraction sees the electron cloud, rather than the nucleus, meaning that the hydrogen's center is systematically displaced toward the bonded atom from the hydrogen's nuclear position by more than 0.1Å (Stewart 1965; Iijima 1987; Coppens 1997). In addition, a hydrogen's electron cloud is sometimes shifted by local non-covalent interactions such as H-bonding or tight packing; both systematic and local shifts can be seen in the figure. The current effort optimizes allowance for the systematic effects, but does not treat environment-dependent distortions.

MOLPROBITY and REDUCE have positioned H atoms at the better-determined nuclear distances (Word 1999), while PHENIX has used the X-ray suitable electron-cloud distances. This difference in parameters affects user clashscores. In addition, we do believe, along with Pauling (1960), that all-atom contacts would more appropriately be calculated at van der Waals radii centered on the hydrogen's electron cloud. MOLPROBITY needs more subcategories of H atom types, while all crystallographic software with libraries for each atom in each monomer type need correction of the typos and internal inconsistencies endemic to such systems. The largest change needed for the PHENIX electron-cloud positions inherited from SHELX through CCP4 is 0.03Å (for O-H) and for the MOLPROBITY nuclear positions is 0.04Å (for tetrahedral N-H). However, each system has

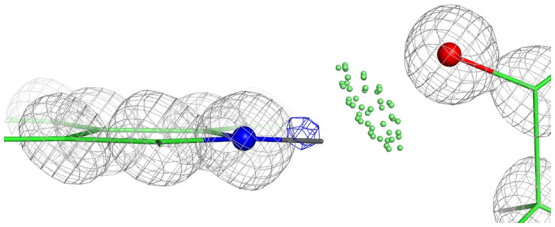


Figure 1: The difference peak for Hε1 (blue contours) is both shifted left along the bond to its parent N atom (the systematic effect) and also upward toward the line of the H-bond (an environmental effect). Note the H nuclear position, illustrated here by a grey stick. Taken from PDB:1YK4, Trp 37, 0.69Å resolution.

changes of up to ~0.17Å for cases where it was applying what we now consider the wrong type of value.

Packing analysis and validation both depend on the total system of hydrogen bondlengths, van der Waals radii, and the 0.4Å threshold defined for clashes. Several factors have convinced us that the current system in MOLPROBITY is slightly too strict. We have therefore re-examined the existing sources for x-H distance values and have undertaken new computational, database, and manual analyses to settle on a confirmed, best set of electron-cloud x-H distances for implementation in both MOLPROBITY and PHENIX. This has involved sphere-fitting to electron densities calculated using quantum mechanics, examining high-resolution H difference-density peaks, and analyzing database distributions of nearest-neighbor atom-atom distances to re-optimize the associated van der Waals radii. In addition, we have compiled bondlengths from X-ray diffraction and neutron-diffraction small-molecule structures from the literature and from the Cambridge Structural Database. The various sources of experimental and theoretical data for x-H distances unfortunately are consistent within

0.01Å only for the nuclear aliphatic C-H case, so that future research would still be desirable. However, we judge that the values presented here are correct within 0.02-0.03Å, nearly an order of magnitude better than the previous situation. Happily, we find that the new parameters produce clashscores that have a better trend towards zero for the best structures at mid to high resolutions and do a slightly better job determining sidechain NQH flips.

We are currently implementing this change in both MOLPROBITY and PHENIX so that the two services will add hydrogens identically and in an appropriately application-specific fashion. The electron-cloud values will be the default in both systems due to the predominant use for X-ray crystallography. However, each will also include an option to use updated nuclear positions for neutron-diffraction refinement, for NMR structures, or by user choice, and MOLPROBITY will use van der Waals radii tuned for each case when calculating all-atom contacts.

References

Coppens P (1997) *X-ray Charge Densities and Chemical Bonding*, Oxford University Press, NY, ISBN 0-19-509823-4.

Iijima H, Dunbar JBJ, Marshall GR (1987) Calibration of effective van der Waals atomic contact radii for proteins and peptides. *Proteins: Struct. Funct. Genet.* **2**: 330-339.

Pauling L (1960) *The Nature of the Chemical Bond*, 3rd ed, Cornell University Press, Ithaca, ISBN 0-8014-0333-2.

Stewart RF, Davidson ER, Simpson WT (1965) Coherent x-ray scattering for the hydrogen atom in the hydrogen molecule, *J. Chem. Phys.* **42**: 3175-87.

Word JM, Lovell SC, LaBean TH, et al. (1999) Visualizing and quantifying molecular goodness-of-fit: Small-probe contact dots with explicit hydrogen atoms, *J. Mol. Biol.* **285**: 1711-33.

PSI SBKB Technology Portal: A Web Resource for Structural Biologists

Lida K. Gifford[a], Margaret J. Gabanyi[b], Helen M. Berman[b], and Paul D. Adams[a]

[a]*Physical Biosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720.*

[b]*Department of Chemistry & Chemical Biology, Rutgers – The State University of New Jersey, Piscataway, NJ 08854.*

Correspondence email: LKGifford@LBL.gov

Introduction

The Protein Structure Initiative (PSI) was first funded in 2000 and remains committed to facilitating the process of solving three-dimensional atomic-level protein structures [1]. In 2008, a web-based resource, now the Structural Biology Knowledgebase (SBKB, sbkb.org), was established to capture and disseminate structural data and results from PSI research [2]. The Knowledgebase has a modular architecture providing access to individual websites for experimental data tracking, protocols, materials, protein annotations, modeling, technologies, and publications. It also makes connections to related data found in GO [3], PDB [4], UniProt [5], and 100+ other publicly available gene and protein resources.

The Technology Portal is one module of the SBKB and is designed to present methods and technologies catalyzed by the PSI high-throughput protein production and structure determination efforts [6]. In addition, it functions as a repository for tools developed by the wider scientific community for structural biologists to take and use in their research. An overarching goal of the Technology Portal is to encourage collaboration among scientists, not only within the PSI, but also between PSI researchers and the broader biological community.

This web resource is comprised of 350 tools and technology summaries representing each step of the protein structure determination pipeline, the majority of which have been developed by the PSI and are in use at PSI Centers. The Technology Portal home page presents all items in one easy-to-navigate page. While several components of the Portal have been described in-depth elsewhere [6], an overview of the search functionality and accessible tools will be presented, with particular emphasis placed on new features.

The Technology Portal Website

The Technology Portal can be reached directly (technology.sbkb.org/portal/) or via the Methods Hub of the SBKB (sbkb.org/kb/methodshub.html). In addition, performing a keyword search on the SBKB can access Technology Portal information. Once at the Portal homepage, users can access all of its technology pages and online resources. The Technology Portal is hosted using Apache HTTP software [7] and constructed using the Django 1.2 web framework [8], which accesses data from a SQLite 3 database [9] and presents them as technology pages. A screen shot of the current homepage shows all of the content and functionalities (Figure 1).

Searching the Technology Portal

There are two ways to search technologies from the Technology Portal homepage: plain text and by experimental stage (Figure 2a). A text search box will return a list of technology pages with the most recently edited pages shown first, prioritizing technology pages containing the newest information. Each technology is indexed by step in the structure determination process, so a more general search can be performed by choosing one of the following topics from a drop-down menu: Target Selection, Reagents, Cloning, Protein Expression, Purification, Crystallography, NMR, Electron Microscopy, SAXS, Annotation/Function, Modeling, or Dissemination Tools. Users can browse search results, which are presented in an alphabetical listing of all records tagged for that particular process (Figure 2b). Technology pages contain all pertinent information, such as a description of the technology, figures, and information regarding publication, whom to contact, web links, and availability, if applicable (Figure 2c).

Technology Toolbox

The first items listed in the Technology Toolbox are the Featured Technology article, Profile, and Technical Note (Figure 1a). All of these features are

Figure 1

The PSI SBKB Technology Portal homepage. a) Users can access information by keyword or experimental step in the search area. b) The Technology Toolbox contains three items that are updated on a periodic basis: Featured Technology, Center or Resource Profile, and Technical Note. In addition, links to web-based resources, Center technology web pages, Tech Videos, and the Tech Forum can be found in the bottom portion of the Toolbox. There are two menus on the homepage: the Main menu on the left and the SBKB banner menu at the top of the page.

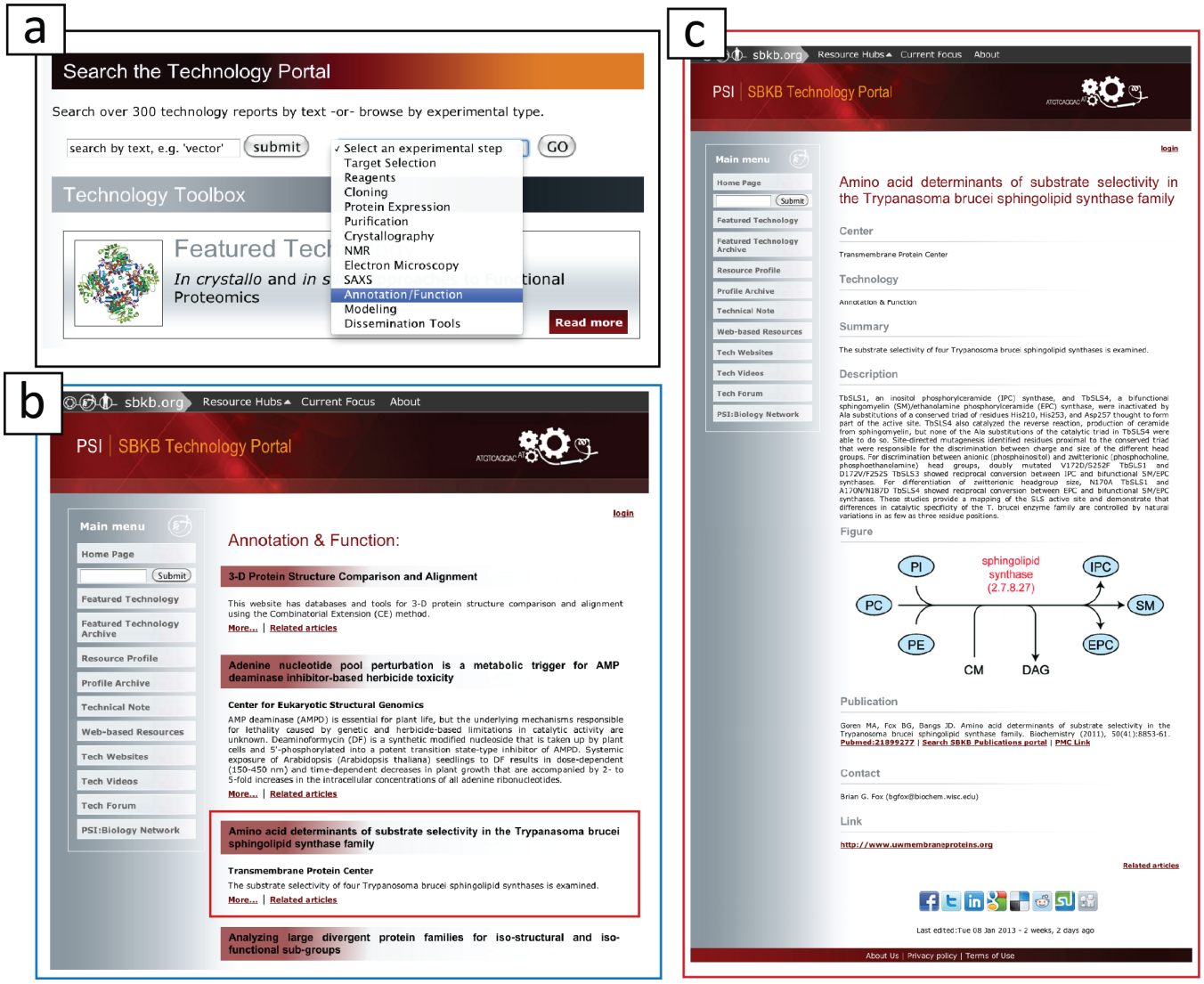


Figure 2

Example search by experimental step. a) Close-up view of the search box area from the home page with Annotation/Function highlighted on the drop-down menu. b) The first page of the search results list obtained by clicking "Go" on the main page. c) A typical technology page containing Title, PSI Center, Summary, Description, Figure and legend, Publications, Contact Information, a Link, and social networking and bookmarking buttons for sharing and saving a link to the page.

periodically updated, with the content changing every one to four months. Featured Technology highlights technologies and resources of interest to structural biologists. Recent articles have covered new advances in functional proteomics [10], the Membrane Proteins of Known 3D Structure Database [11], and the Biosync website [12]. All previous articles are displayed in the Featured Technology Archive, accessed by the Main menu (Figure 1b, [13]). The Profile section, started in June 2012, details the research and outreach projects performed at different PSI Centers. Most recently, the Profile focused on the PSI:Biology-Materials Repository (PSI:Biology-MR; [14, 15]), which stores, maintains, and distributes PSI-created protein expression plasmids and vectors. This article introduces users to the types of information that are available on the PSI:Biology-MR website and offers a glimpse into the structure, function, and reach of this valuable resource [16]. A link to the Profile Archive, currently displaying three previous Center Profiles, can be found in the Main menu (Figure 1b, [17]).

The most recent addition to the Technology Toolbox is the Technical Note. This feature was established in December 2012 and the inaugural entry describes the standard operating procedure for making insect cell media in-house [18], contributed by James D. Love, PhD, Head of Technology, New York Structural Genomics Research Consortium (nysgrc.org). This Note will change periodically and be used to disseminate detailed protocols that would be beneficial to a wide audience of structural biologists.

The bottom portion of the Technology Toolbox contains links to Web-based Resources, Tech Websites, Tech Videos, and the Tech Forum. The Web-Related Resources section contains a list of over 70 technology pages describing web servers or web-based tools that can be employed by structural biologists to design, predict, and model results (technology.sbkb.org/portal/useful_servers). Clicking on Tech Websites takes users to a page listing the technology websites maintained by PSI Centers. All four of the PSI:Biology High-Throughput Centers for Protein Structure Determination, over half of the Centers for Membrane Protein Structure Determination, and the Mitochondrial Protein Partnership, a PSI:Biology Consortium for High-Throughput Enabled Structural Biology Partnership, maintain web pages describing the software, tools, and other technologies they have developed to relieve bottlenecks in protein structure determination (technology.sbkb.org/portal/technology_links). The Tech Videos page displays sbkbtech YouTube channel videos (www.youtube.com/user/sbkbtech). Currently, there are eight technologies in action (technology.sbkb.org/portal/video). Tech Forum is a direct conduit to the Technology Portal Forum group (network.nature.com/groups/psikb_tech/) hosted on the Nature Network. This forum provides a virtual place for scientists to connect on a professional level and exchange information about structural biology technology.

Outreach

The Technology Portal maintains a Facebook page at facebook.com/techportal. Posting of new content is announced on Facebook, as well as interesting items from the SBKB [19] and PSI:Biology-MR [20] Facebook pages, meeting announcements, and notifications from sources such as Nature [21], the Protein Data Bank [4], and NCBI [22].

The Technology Portal works closely with two other PSI:Biology web resources by using link-outs to connect useful information between sites. For example, the PSI Publications Portal (olenka.med.virginia.edu/psi/) contains all publication information and statistics for the more than 1800 peer-reviewed articles that have been published by the PSI over the past 12 years. Users can find links to article summaries on most technology pages. In addition, if a publication is referenced on a technology page, the Publications Portal entry contains a link to the appropriate page on the Technology Portal.

Crosslinks have also been established with the PSI:Biology-Materials Repository (psimr.asu.edu/; [14, 15]). All technology pages describing a vector designed and used by PSI Centers contain availability information and a link to the PSI:Biology-MR. Reciprocally, on appropriate vector pages, the PSI:Biology-MR refers users to the Technology Portal for more information.

Conclusion

The Technology Portal is dedicated to capturing and highlighting technological advances that are instrumental to enabling structural biological research. This is accomplished by collaborating with PSI Centers and members of the wider scientific community to gather methods and tools that have been developed that would be widely beneficial to scientists. Tech Portal feature articles and technology pages are updated frequently so that scientists can take and use the knowledge and tools provided in their research. We welcome feedback from the community at psi-tech@lbl.gov.

Acknowledgements

The authors would like to thank the Protein Structure Initiative PIs, researchers, and Structural Biology Knowledgebase members for their collaboration and support of the Technology Portal. We thank Nicholas Sauter for server maintenance and technical support. The Technology Portal is a resource center within the Protein Structure Initiative and is supported by grant U01GM093324 from the National Institute of General Medical Sciences. This work was supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231.

References

1. Gabanyi MJ, *et al.* (2011) The Structural Biology Knowledgebase: a portal to protein structures, sequences, functions, and methods. *J Struct Funct Genomics* 12(2):45-54.
2. Berman HM, *et al.* (2009) The Protein Structure Initiative Structural Genomics Knowledgebase. *Nucleic Acids Res* 37(Database issue):D365-368.
3. The Gene Ontology Consortium. (2000) Gene Ontology: tool for the unification of biology. *Nature Genet.* 25:25-29.
4. Berman HM, *et al.* (2000) The Protein Data Bank. *Nucleic Acids Res* 28:235-242.
5. The UniProt Consortium. (2012) Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Res* 40:D71-D75.
6. Gifford LK, Carter LG, Gabanyi MJ, Berman HM, & Adams PD. (2012) The Protein Structure Initiative Structural Biology Knowledgebase Technology Portal: a structural biology web resource. *J Struct Funct Genomics* 13(2):57-62.
7. Apache HTTP Server Project. (2011) Available from: http://httpd.apache.org/
8. Django: A Python Web Framework. (2005) Available from: http://www.djangoproject.com
9. SQLite 3: SQL database engine. (2004) Available from: http://www.sqlite.org/
10. PSI SBKB Technology Portal. (2012) *In crystallo* and *in silico* Approaches to Functional Proteomics. Available from: http://technology.sbkb.org/portal/fall_2012_feat_tech
11. White, S. (1998) Membrane Proteins of Known 3D Structure Database. Available from: http://blanco.biomol.uci.edu/mpstruc/
12. Kuller A, *et al.* (2002) A biologist's guide to synchrotron facilities: the BioSync web resource. *Trends Biochem Sci* 27(4):213-215.
13. PSI SBKB Technology Portal. (2011) PSI Tech Portal Featured Technology Archive. Available from: http://technology.sbkb.org/portal/archive
14. Cormier CY, *et al.* (2010) Protein Structure Initiative Material Repository: an open shared public resource of structural genomics plasmids for the biological community. *Nucleic Acids Res* 38(Database issue):D743-749.
15. Cormier CY, *et al.* (2011) PSI:Biology-materials repository: a biologist's resource for protein expression plasmids. *J Struct Funct Genomics* 12(2):55-62.
16. PSI SBKB Technology Portal. (2012) PSI:Biology-Materials Repository: Keeping the Wheels of Structure Determination in Motion. Available from: http://technology.sbkb.org/portal/PSIMR_profile
17. PSI SBKB Technology Portal. (2012) PSI Tech Portal Profile Archive. Available from: http://technology.sbkb.org/portal/profile_archive
18. PSI SBKB Technology Portal. (2012) Hi5 & Sf9 Media: Standard Operating Procedure for In-House Production. Available from: http://technology.sbkb.org/portal/insect_tech_note
19. PSI Structural Biology Knowledgebase. (2010) PSI Structural Biology Knowledgebase Facebook Page. Available from: https://www.facebook.com/psisbkb
20. PSI:Biology-Materials Repository. (2010) PSI:Biology-Materials Repository (PSI:Biology-MR) Facebook Page. Available from: https://www.facebook.com/pages/PSIBiology-Materials-Repository-PSIBiology-MR/108495809177357
21. Nature Publishing Group. (2007) Nature Facebook Page. Available from: https://www.facebook.com/nature
22. National Center for Biotechnology Information. (2010) NCBI - National Center for Biotechnology Information Facebook Page. Available from: https://www.facebook.com/ncbi.nlm

cctbx tools for transparent parallel job execution in Python. I. Foundations

Gábor Bunkóczi[a] and Nathaniel Echols[b]

[a]*Department of Haematology, University of Cambridge, Cambridge, UK*
[b]*Lawrence Berkeley National Laboratory, Berkeley, CA*

Correspondence email: gb360@cam.ac.uk

Introduction

Parallelization can distribute independent calculations over multiple processing units and can decrease the run time of a series of slow calculations. Unfortunately, parallel code is also more complicated, therefore error-prone and difficult to maintain. While there are several reasons for increased complexity, notably the need for synchronization between parallel processes, maintenance of parallel-enabled program sections is also a (perhaps underrated) problem that deserves attention.

While most languages do not provide direct support for parallelization and rely on low-level operating system support, there are successful initiatives for enabling parallel processing either by compiler technology (e.g. OpenMP, http://openmp.org) or an external library component (e.g. Intel Threading Building Blocks, http://threadingbuldingblocks.org). Similarly, the Python-specific `scheduling` module (Bunkóczi & Echols, 2012) provides high-level constructs for transparent parallel execution that aim to be minimally intrusive on code layout. It is available with the `libtbx` module of the Computational Crystallography Toolbox (`cctbx`, http://cctbx.sourceforge.net; Grosse-Kunstleve *et al.*, 2002), the open-source component of the *PHENIX* project (http://www.phenix-online.org; Adams *et al.*, 2010).

Problem setting

For the purposes of this article, *parallelization* is defined as multiple simultaneous job execution managed at Python code level. This includes the Python standard library modules `threading` and `multiprocessing`, as well as any module that supports job execution on batch queue systems (e.g. `libtbx.queuing_system_utils.processing`), as long as they provide a factory function that conforms to a standard interface defined by `threading.Thread` or `multiprocessing.Process`. Parallelization technologies that affect code at lower levels (e.g. an OpenMP-parallelized function exported to Python) are excluded from the discussion.

A typical maintenance problem is illustrated with the (slightly exaggerated) code example on Figure 1. Apart from not being aesthetically pleasing, there are several serious problems with this pattern:

1. It is not feasible to come up with a (sufficiently) comprehensive list of all possible parallelization techniques, especially batch queue managers. New systems may appear in the future, or may become non-backwards compatible after a major release. In this case, the existing codebase has to be searched and amended as necessary.
2. There exist almost certainly instances of severe code duplication among the branches of the conditional expression. If the algorithm changes, all branches need to be updated. Moreover, the algorithm logic is potentially interleaved with calculation details and it is more difficult to follow the program flow.
3. The program will typically contain several parallel sections and parameters of parallel execution to be propagated to all of these. In addition to distributing knowledge throughout the program (discussed in 1), this makes it very difficult or impossible to reuse existing parts, because parallelization normally lies outside the computational domain of a given calculation. This requires the introduction of numerous thin wrappers and further code duplication, because details of parallelization are not mandated to be universal among different programs and are therefore potentially incompatible.

```
if num_cpus == 1:
  ... do something ...

else if parallelisation_method = "shared_memory":
  ... do something else, potentially using num_cpus ...

else if parallelisation_method == "SGE":
  ... do something else, potentially using num_cpus ...

else:
  raise ValueError, "Unknown parallelization method"
```

Figure 1

Code segment from a hypothetical naively constructed parallel-enabled function.

The `scheduling` module

The `libtbx.queuing_system_utils.scheduling` module (Bunkóczi & Echols, 2012) helps with encapsulating execution details within a `Manager` object and could provide a solution for some of the aforementioned problems:

1. The `Manager` object is independent of the computation. It is possible to instantiate it at the very beginning of the application using a standardized user interface and use it throughout the program. This allows encapsulation of all details in a user interface definition and a module-level function that creates a `Manager` object from the standard interface. Since this could be stored at a central location, code changes are automatically propagated to all applications. This in turn enables incremental delivery and support for new queuing systems can be added at any time without necessary modifications to application code.
2. Since execution mode is encapsulated, there is no need for code duplication. This also results in a less complex program flow.
3. The `Manager` object allows parallel execution of Python callable objects. There is no other requirement imposed by the module. However, the `Manager` object still needs to be propagated to all parallel enabled code. In addition, there may be further requirements imposed by the execution technology (e.g. certain objects need to be pickleable). These, however, can be satisfied by external code (e.g. adding pickling support) and do not require customizations in the `scheduling` module.

However, there are some additional tradeoffs to consider:

1. Unlike OpenMP, the code needs to be written using the module explicitly.
2. There is a slight overhead (compared to specialized non-parallel code) when parallel-enabled code written using the module is executed on a single CPU. This overhead roughly equals to several additional function calls and array manipulations per iteration. However, if the parallel section takes significantly longer than typical Python array manipulations, this overhead should be negligible.

On the other hand, the module is sufficiently flexible so that program behavior does not need to change. It is possible to re-structure non-parallel code so that the version rewritten using the `scheduling` module behaves identically from the user's perspective.

Parallel architecture

The scheduling module supports a master-slave type pattern. The main thread (the master) assigns calculations to workers and these can proceed independently without blocking the master. The workers only communicate with the master and therefore there is no need for synchronization between them. Scheduling is performed by the master to ensure that available computational resources are fully exploited, but not overcommitted. Tests show that scheduling overhead is negligible and scaling is only dependent on the fraction of parallel code.

The scheduler component is provided by implementations of the `Manager` interface. Calculations that can be delegated to workers take the form of Python function calls, with a callable (either function or method) and (positional and keyword) arguments. These are stored inside the `Manager` object and the calculation is started when a worker becomes idle. Calculation status queries and access to results are made available through method calls.

`Manager` models

The `Manager` interface is an abstraction for controlling program behavior from the users' perspective. It is defined in the `scheduling` module and allows multiple implementations with varying properties so that a good match for the program's requirements could either be found or implemented. There are currently three basic implementations available:

- `Manager`. This has possibly the most widespread applicability. It manages not only the execution of calculations (which is pre-configured, but allowed to be a mixture of modes), but also a job queue. It assumes, however, that executing a calculation does not block the main thread of control.
- `MainthreadManager`. This is a dummy `Manager` that executes calculations on the main thread. Execution is delayed so that output can keep scrolling as calculations complete. In addition, this has the lowest job startup overhead equivalent to about a single function call and is therefore the right choice if there is only one processing unit available.
- `Adapter`. This allows sharing of a single `Manager` instance in order to provide scalable computing power to multiple parts of the same application. It behaves as a `Manager` with a variable number of processing units, since depending on resource utilization of other components, computational resources available to a given component can vary.

There are additional classes that may be necessary for certain tasks and whose implementations are pending:

- `QueueManager`. This implements access to a batch queuing system and delegates queue management to the same. Only a single execution mode is allowed, but the number of concurrent processes is unlimited.
- `PoolManager`. Instead of creating a new thread of control for each calculation, this re-uses a pre-created pool of workers. If running through a batch queue system, this could lower job startup overhead considerably as it would effectively bypass the job queue. An efficient implementation could also provide for automatic resource allocation/de-allocation according to the load.

Calculation results and error handling

The `Manager` is unaware of any backwards communication between the actual workers and their proxies (`ExecutionUnit`s) held by the `Manager`. This responsibility is handled by the individual `ExecutionUnit`s. After termination of the underlying calculation, the `Manager` provides an opportunity for the `ExecutionUnit` to "post-process" the results and become ready for accepting the next calculation. In this time window it is possible to obtain results from the worker through a suitable data queue. In case a data queue becomes stuck (e.g. through an NFS latency), the `Manager` temporarily suspends the `ExecutionUnit` to avoid stalling the job queue and then periodically retries. Eventually, after a preset timeout, the `ExecutionUnit` is reset and the calculation is handled as failed.

Signaling error conditions is delayed until the calculation result is accessed. This is to ensure that in case an error occurs the internal state of the `Manager` is not corrupted, so that processing can be resumed (in case it is allowed by the nature of the computation, e.g. a suitable default value can be substituted for the missing result). The returned job handle is a type with a polymorphic `__call__` method that yields the value returned by the call if the job terminated normally or raises a `RuntimeError` exception if failed. The module currently takes the view that any exception handling code for exceptions that are expected should be included within the call. When executing a call on a batch queue, it is the exit status of the process that is monitored and an error is raised if a process terminates with a non-zero exit status. Since execution occurs independently, errors that would normally cause the parent process to terminate if executed on a

```
from mymodule import mycalc

identifier1 = manager.submit( target = mycalc, args = ( 1, ) )
identifier2 = manager.submit( target = mycalc, args = ( 2, ) )

# fetch result for designated job
handle2 = manager.result_for( identifier = identifier2 )

try:
  result2 = handle2()

except RuntimeError, e:
  print "Second calculation failed"

# fetch result for next job
handle1 = manager.results.next()

try:
  result1 = handle1()

except RuntimeError, e:
  print "First calculation failed"
```

Figure 2

Code example: using methods of a `Manager` to submit jobs and wait for results. `Manager` can return result handles for requested calculations or for the one that finishes next. The `__call__` method of a result handle returns immediately and only necessary to signal an error condition. `mymodule` is a hypothetical module containing the function `mycalc`. See Bunkóczi & Echols (2012) on how to instantiate a `Manager`.

separate thread or process (e.g. segmentation fault) can be handled more gracefully, either by logging the error and resuming execution, or by terminating gracefully. A typical error-handling scenario is shown in Figure 2.

Parallel constructs

Using a `Manager` object directly allows full flexibility. This may be necessary if each job is a different type of calculation or if subsequent jobs are dependent on previously returned results (e.g. from molecular replacement, parallel translation functions for rotation function peaks). The `Manager` provides an API similar to that of a queuing system. When calls are submitted, the `Manager` returns a job identifier that can later be used to check whether the job has finished. The `Manager` also provides various methods to access results of finished calculations. A very simple example is shown in Figure 2.

While having full flexibility is sometimes advantageous, it is not always necessary. In the majority of cases, parallelization is employed to execute a loop in parallel (cf. `#pragma omp parallel for`). While this can be emulated by an initial `for` loop that submits all calculations to the `Manager`, and a subsequent `for` loop that fetches results for each identifier, this solution suffers from several drawbacks. First, the number of loop iterations is normally larger than the number of available processing units. Input arguments for all calculations have to be generated upfront, unnecessarily filling up memory. Second, there may be a long delay while calculations are being submitted before the result for the first calculation is accessed. This can cause not only user inconvenience, but also accumulation of results that may also lead to out-of-memory errors. Third, if the sequence of calculations is infinite, the program will never reach the second loop (this may sound rather

esoteric, but it is in fact a valid technique to use an infinite sequence and is employed by several widely used programs in crystallography).

The `scheduling` module provides the `ParallelForIterator` class for these situations. This requires a `Manager` instance and an iterable returning a tuple `(target, args, kwargs)` that defines each calculation. The `ParallelForIterator` maps these calculations onto the `Manager` in a dynamic fashion. Since the iterable can be a generator, the arguments are generated as needed and not upfront. The delay is also much less, because only as many calculations are submitted as there are idle processing units. In addition, processing infinite sequences is possible with finite memory, since calculations are submitted and results are accessed continuously, and at any given time point there is a finite number of active calculations.

In contrast to what is suggested by its name, `ParallelForIterator` is not a Python iterable. However, an iterable can be constructed by passing the object to the ordering classes `FinishingOrder` and `SubmissionOrder`. These control the order in which results are returned. If there is no particular need, `FinishingOrder` should always be preferred, because it does not incur any memory overhead. `SubmissionOrder` returns results in the order they appear in the iterable that defines the calculations. However, it can lead to accumulation of results if a calculation finishes much slower then subsequent ones, and may result in an out-of-memory error (it should be noted that this overhead is unavoidable and not an implementation artifact of `SubmissionOrder`). On the other hand, a very slow calculation does not represent a bottleneck for execution because subsequent calculations will be started as soon as `ExecutionUnit`s become idle. Further ordering classes may be developed as needed. A typical example is shown on Figure 3.

Infinite sequences

While it is clear that an infinite sequence can neither be processed in finite time nor would it fit into finite memory, lazily evaluated infinite sequences are valuable tools for computing certain quantities. A very practical example from crystallography is the implementation of *ab initio* direct methods in the program SHELXD (Sheldrick, 2010). The default setting for SHELXD is an infinite run with unlimited tries. Each try is independent, and depends on a random number from an infinite random number series. Termination of the job is the responsibility of the user. In `phenix.hyss` (Grosse-Kunstleve & Adams, 2003), manual termination has been replaced by suitable decision criteria. When either a suitably good solution has been found or the user decides to give up, processing the sequence of Patterson peaks is terminated; the best result is then selected and refined. Both the infinite sequence processing from SHELXD and the graceful termination from `phenix.hyss` can be implemented using a `ParallelForIterator`. A suitably initialized random sequence and a function that performs dual space recycling can be mapped onto a `Manager`. Graceful termination can be implemented with the `suspend` method of `ParallelForIterator`, which simulates that the end-of-sequence has been reached, and no more calculations are submitted to the queue (`ParallelForIterator` also provides a `resume` method to continue processing). Iteration continues until all currently running calculations finish. A simple infinite sequence example is shown in Figure 4.

```
from mymodule import mycalc

# non-parallel code
values = []

for i in range( 50 ):
  values.append( mycalc( i ) )

# identical parallel code
from libtbx.queuing_system_utils import scheduling

parallel_for = scheduling.ParallelForIterator(
  calculations = ( ( mycalc, ( i, ), {} ) for i in range( 50 ) ),
  manager = manager,
  )

values = []

for ( params, handle ) in scheduling.SubmissionOrder( parallel_for ):
  try:
    result = handle()

  except RuntimeError, e:
    ( function, args, kwargs ) = params
    print "Error (mycalc with arguments %s): %s" % ( str( args ), e )
    continue

  values.append( result )

# parallel list comprehension, no error handling
parallel_for = scheduling.ParallelForIterator(
  calculations = ( ( mycalc, ( i, ), {} ) for i in range( 50 ) ),
  manager = manager,
  )

values = [ handle() for ( params, handle )
  in scheduling.SubmissionOrder( parallel_for = parallel_for ) ]
```

Figure 3

Code example: parallelizing a loop using a `Manager` and `ParallelForIterator`. Although more complex constructs are used, the basic structure (i.e. a single loop) does not change. In addition, `ParallelForIterator` can be used in any Python list expressions and will be evaluated in parallel (error handling is still possible, but not shown for simplicity).

```
from mymodule import is_prime
from libtbx.queuing_system_utils import scheduling
import itertools

parallel_for = scheduling.ParallelForIterator(
  calculations = (
    ( is_prime, ( i, ), {} ) for i in itertools.count( 10000 )
    ),
  manager = manager,
  )
primes = []

for ( params, handle ) in scheduling.SubmissionOrder( parallel_for ):
  try:
    result = handle()

  except RuntimeError, e:
    ( function, args, kwargs ) = params
    print "Error (is_prime with arguments %s): %s" % ( str( args ), e )
    continue

  if result: # number is prime
    ( function, args, kwargs ) = params
    primes.append( args[0] )
    parallel_for.suspend()

# there may be multiple primes found, but we are only interested in the
# lowest
print "The lowest prime number higher than 10000 is %s" % primes[0]
```

Figure 4

Code example: finding the first prime number above a preset value using parallel code and an infinite sequence. `is_prime` is a hypothetical function returning a Boolean value.

References

Adams, P. D., Afonine, P. V., Bunkoczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L. W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). *Acta Cryst.* **D66**, 213-221.

Bunkóczi, G. & Echols, N. (2012). *Computational Crystallography Newsletter* **3**, 37-42.

Grosse-Kunstleve, R. W. & Adams, P. D. (2003). *Acta Cryst.* **D59**, 1966-1973.

Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J. Appl. Cryst.* **35**, 126-136.

Sheldrick, G. M. (2010). *Acta Cryst.* **D66**, 479-485.

cctbx tools for transparent parallel job execution in Python. II. Convenience functions for the impatient.

Nathaniel Echols[a], Gábor Bunkóczi[b], and Ralf W. Grosse-Kunstleve[a,c]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA*
[b]*Department of Haematology, University of Cambridge, Cambridge, UK*
[c]*Present address: Google Inc. (San Bruno, CA)*

Correspondence email: nechols@lbl.gov

Introduction

We have described (Bunkóczi & Echols, 2012, 2013) a set of Python classes for handling parallel execution of tasks on a multiprocessor system or managed cluster. This API has the advantage of being suitable for any situation where parallelism is desired, and is not necessary limited either to "embarrassingly parallel" methods where the same single function is called repeatedly. However, this latter case occurs frequently enough in large applications that it merits its own treatment, and specialized parallelization functions that minimize the programming overhead.

Examples of embarrassingly parallel methods currently used in Phenix (Adams *et al.*, 2010) include:

- Multiple MR searches (phaser.MRage), which were the original motivation for the tools described in the accompanying article
- Multiple Rosetta rebuilding jobs in MR-Rosetta (DiMaio *et al.*, 2011)
- Exploration of alternate model-building strategies (Terwilliger *et al.*, 2007; Terwilliger, Grosse-Kunstleve, Afonine, Moriarty, Zwart, *et al.*, 2008)
- Ligand search trials with varying parameters (Terwilliger *et al.*, 2006)
- Omit map calculation (Terwilliger, Grosse-Kunstleve, Afonine, Moriarty, Adams, *et al.*, 2008)
- Optimization of the X-ray/restraint weights by grid search in phenix.refine (Afonine *et al.*, 2011; Afonine *et al.*, 2012)
- Validation of multiple models in an ensemble (mmtbx.validation_summary)

Many other crystallographic software packages and automation pipelines employ similar parallelism, including Mr-BUMP (Keegan & Winn, 2008), ARCIMBOLDO (Rodriguez *et al.*, 2009; Rodriguez *et al.*, 2012), qFit (van den Bedem *et al.*, 2009), MR-GRID (Schmidberger *et al.*, 2010), WS-MR (Stokes-Rees & Sliz, 2010), Xsolve (van den Bedem *et al.*, 2011), and weight optimization for DEN refinement (Schroder *et al.*, 2010; O'Donovan *et al.*, 2012). Targeted system resources range from multi-core laptop or desktop computers up to supercomputers or grid platforms comprising thousands of CPUs (which are beyond the scope of this article). Increasingly, a target goal of automation efforts is to obtain an initial structure solution at the synchrotron beamline while the experiment is ongoing (Panjikar *et al.*, 2005), which also typically requires a high level of parallelism.

More generally, experienced crystallographers often operate parallel workflows like these manually, sometimes with the aid of shell scripts. In many of these situations, the majority of the input data remains the same for the individual jobs and the runtime of each job is expected to be similar. This essentially reduces the problem to calling `map(function, iterable)`. For this purpose we have introduced two parallel `map()` implementations in the module `libtbx.easy_mp`, one based on the tools described in part I, and another extending the built-in `multiprocessing` module.

General-purpose, platform-independent parallelism

Many of the examples listed above have runtimes for individual jobs in the range of minutes to hours; this makes them suitable for execution either on a queuing system or a shared-memory multiprocessor machine. For such applications, the `parallel_map` function provides access to both types of resource. Consider the following simplified example[1]:

[1] This is essentially an abstraction of the code used in the experimental "Parallel Phaser" GUI, which simply provides a frontend for running MR searches on a large number of models in parallel.

```
class phaser_manager (object) :
  def __init__ (self, data_file) :
    self.data_file = data_file

  def __call__ (self, model) :
    # the actual implementation is elsewhere
    return run_phaser(self.data_file, model)

def run_all (data_file, models, method="multiprocessing",
    processes=1, qsub_command=None, callback=None) :
  phaser = phaser_manager(data_file)
  from libtbx.easy_mp import parallel_map
  return parallel_map(
    func=phaser,
    iterable=models,
    method=method,
    processes=processes,
    callback=callback,
    qsub_command=qsub_command)
```

In this pseudo-code, we are simply running Phaser MR searches on a list of models, and returning the list results (whatever they may be) to the parent function. On Linux systems, the `method` argument could instead have been "sge", "lsf", "pbs", or "condor"; otherwise, the execution and retrieval of results is completely transparent. Internally, the function creates as many `scheduling.ExecutionUnit` objects as requested (the `processes` argument), then creates a `scheduling.Manager` object with the `ExecutionUnits`, and submits each individual function call as a separate task. The results are returned in the same order as the inputs. Although execution of the main thread blocks until all tasks have finished, the optional `callback` argument specifies a function (or callable object) to be run each time a result is retrieved. This is used, for instance, to display the results of MR searches in the Phenix GUI as they become available, instead of waiting until the end of the run.

A few important details must be taken into account when using the parallel mapping function `libtbx.easy_mp.parallel_map`. First, both the target function and the iterable must be pickle-able: this includes most of the essential low-level C++ objects used in cctbx, including all of the `scitbx.array_family` classes, the Miller array (`cctbx.miller.array`), and the PDB hierarchy (`iotbx.pdb.hierarchy.root`). In the example the `data_file` argument could be either a file name or a collection of Miller arrays already extracted from the file. The models argument could be either a list of PDB file names, or a list of PDB hierarchy objects. Similarly, the result objects must also be pickle-able. Note that since the passing of data to and from processes on queuing systems requires the use of intermediate pickle files, the I/O overhead is non-negligible whether or not the data and model files must be read in by each process. As noted above, the runtime for each model should be similar; although this is not strictly a requirement, it will result in the maximum speedup with more processes. Finally, although `parallel_map` can be used for relatively short tasks (on the order of seconds) in multiprocessing mode, this is not recommended for queuing systems.

Except for these limitations, the design of the target function has no special constraints; it may be any combination of Python or C++ code and system calls, and does not necessarily need to be an object method as shown in the example. It may even employ additional parallelism, although some care is required to ensure even allocation of system resources. It should also be noted that not all parallelization methods are compatible with each other. For user-oriented programs where both multiprocessing and queuing system support is desired, a `libtbx.phil` scope (Grosse Kunstleve *et al.*, 2005) containing execution options (method, processes, commands, etc.) can be embedded in the application parameters, and the extracted Python objects passed as a block to `parallel_map`.

Extending multiprocessing.Pool for greater flexibility

The `parallel_map` function is optimal for new applications where the data to be passed between the parent and the child processes are pickle-able. (It is also the only method fully supported on Windows, which has built-in limitations that make process-level Python parallelism more difficult.) However, this is not always desirable or technically feasible, especially where previously written code must be significantly refactored. A representative example is the optimization of X-ray/stereochemistry restraint weights, described in (Afonine *et al.*, 2011), here greatly simplified to the following code:

```
from libtbx import easy_mp
import sys

class refine_xyz (object) :
  def __init__ (self, model, fmodel, params,
      out=sys.stdout, nproc=1) :
    self.model = model
    self.fmodel = fmodel
    self.params = params
    target_weights = [0.1, 0.25, 0.5, 1.0, 2.0, 5.0]
    trial_results = easy_mp.pool_map(
      fixed_func=self.try_weight,
      args=target_weights,
      processes=nproc)
    self.opt_r_free, self.opt_weight, sites_best = \
      min(trial_results) # pick best R-free
    self.fmodel.xray_structure.set_sites_cart(sites_best)
    self.fmodel.update_xray_structure(update_f_calc=True)

  def try_weight (self, weight) :
    # function defined elsewhere; modifies objects in place
    out = StringIO()
    minimize_coordinates(
      model=self.model,
      fmodel=self.fmodel,
      weight=weight,
      log=out)
    sites_cart = self.fmodel.xray_structure.sites_cart()
    return (self.fmodel.r_free(), weight, sites_cart)
```

The minimization function requires a complex collection of objects, including the `mmtbx.f_model.manager` that handles the structure factor calculations (and is pickle-able) and the `mmtbx.model.manager` that contains not only the PDB hierarchy but all geometry restraints (which currently cannot be pickled). To use the `parallel_map` method, either pickling support would have to be added to all of the classes involved or the code would need to be refactored to regenerate the restraints for the current model (e.g. as a string of `ATOM` records) each time `minimize_coordinates` is called. Either change involves a significant loss of efficiency for either the developer or the end user.

The critical advantage of the `pool_map` method used in the second example is that it does not have the pickling requirement for the target function (although the arguments and results must still be pickle-able). This exploits a detail of the implementation of `multiprocessing.Process` on Linux and other Unix systems such as MacOS: new processes are started by calling `os.fork()`, which creates a clone process with copy-on-write memory pages shared by the parent and child processes. By subclassing the `multiprocessing.Pool` object and passing the `fixed_func` argument to the initializer, all attributes are inherited by the child process when forked, after which the process pool's existing

`map()` method is called. In many practical situations the bulk of the memory pages are never modified by the child processes, and therefore never duplicated. Therefore fork's copy-on-write approach not only avoids the pickling overhead for the target function, but also makes it possible to have many child processes even if the parent process already consumes a large fraction of the available memory.

Using this method, any existing embarrassingly parallel `for` loop can be parallelized with the addition of a few lines of code. This is used in several places in *phenix.refine* and *phenix.den_refine*, where near-linear speedups are achieved for the parallelized loops. The copy-on-write memory management of the Unix `fork()` system call keeps the memory overhead to a minimum, and since all other objects are shared in memory, no file I/O is required. The overhead for forking a process is minimal, and if the function arguments (and return values) can be kept small and simple, pickling these will not have a significant impact on runtime. For these reasons, `pool_map` is also suitable for cases where the individual tasks have runtimes on the order of seconds (e.g. for the validation of ensemble models).

Practical considerations

Because `libtbx.easy_mp` is intended to make parallelization of existing code as easy and non-invasive as possible, neither of the methods described here place many restrictions on the implementation of the mapped function. However, following several general guidelines should make the transition easier:

- Keep file I/O to a minimum - in particular, avoid any writes to paths or filehandles that may be in use by other processes. Use local "scratch" disks whenever possible if running on a queuing system.
- It is also helpful to keep printed log output to a minimum; any important information that the user needs to see should be saved for display until the result is retrieved in the main process.
- Avoid uncaught exceptions, except where programmer error is likely to be at fault. Errors such as an inappropriate input or a failed calculation should be included in the return value and handled in the main process.
- If the runtime of the target function on each of the arguments is expected to vary and can be guessed or approximated, the arguments that are expected to take the longest should be first in the list.
- For maximum convenience and compatibility, both methods will simply execute a serial for loop over all arguments if number of processes equals 1 or the parallelization method is not supported (such as `easy_mp` under Windows). This means that no special application-level code is required to decide whether or not to use parallelism.

An additional limitation is that unlike the `ParallelForIterator` described previously (Bunkóczi & Echols, 2013), the parallel `map()` implementations do not have the option of exiting early if an acceptable result is found. Therefore they are only suitable for tasks where all results need to be collected and analyzed at once.

Finally, we emphasize that these libraries are designed for end-user applications such as Phenix that must be as portable as possible. They target the computing resources available to the typical crystallographer, i.e. multi-core personal computers and workstations ranging from 2 to 64 CPU cores, up to lab- or department-scale shared clusters, where the maximum available resources are expected to be, at most, several hundred processor cores. Deployment of the application is intended to require no additional setup or dedicated resources beyond the optional availability of one of the supported queuing systems. It therefore makes several compromises that limit the scalability, in particular the use of the file system for execution on a cluster, and it is not fault-tolerant with regards to outright crashes or hardware failures. For applications intended to run on a specific system, or where scalability, stability, and efficiency across thousands of processors are essential, a more specialized (but less portable) platform or API such as MPI or MapReduce[1] (Dean & Ghemawat, 2008) should provide significantly better performance.

Availability and platform support

Any CCTBX version released after January 2013 has the described functions (this is also included

[1] A popular open-source implementation, Apache Hadoop (http://hadoop.apache.org), includes Python support, and there does not appear to be any obstacle to using Hadoop as an additional backend in the framework we describe.

in Phenix build 1282 or newer). Support for queuing systems is effectively limited to Linux; the `pool_map` function is limited to Linux and MacOS (it may still be used on Windows, but will run serially). Additional examples of usage are available by request from the authors.

References

Adams, P. D., Afonine, P. V., Bunkoczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L. W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). *Acta Crystallogr D* **66**, 213-221.

Afonine, P. V., Echols, N., Grosse Kunstleve, R. W., Moriarty, N. W. & Adams, P. D. (2011). *Computational Crystallography Newsletter* **2**, 99-103.

Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M. W., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H. & Adams, P. D. (2012). *Acta Crystallogr D* **68**, in press.

Bunkóczi, G. & Echols, N. (2012). *Computational Crystallography Newsletter* **3**, 37-42.

Bunkóczi, G. & Echols, N. (2013). *Computational Crystallography Newsletter* **4**, (in press).

Dean, J. & Ghemawat, S. (2008). *Communications of the ACM* **51**, 107-113.

DiMaio, F., Terwilliger, T. C., Read, R. J., Wlodawer, A., Oberdorfer, G., Wagner, U., Valkov, E., Alon, A., Fass, D., Axelrod, H. L., Das, D., Vorobiev, S. M., Iwai, H., Pokkuluri, P. R. & Baker, D. (2011). *Nature* **473**, 540-543.

Grosse Kunstleve, R. W., Afonine, P. V., Sauter, N. K. & Adams, P. D. (2005). *Newsletter of the IUCr Commission on Crystallographic Computing* **5**, 69-91.

Keegan, R. M. & Winn, M. D. (2008). *Acta Crystallogr D Biol Crystallogr* **64**, 119-124.

O'Donovan, D. J., Stokes-Rees, I., Nam, Y., Blacklow, S. C., Schroder, G. F., Brunger, A. T. & Sliz, P. (2012). *Acta Crystallogr D Biol Crystallogr* **68**, 261-267.

Panjikar, S., Parthasarathy, V., Lamzin, V. S., Weiss, M. S. & Tucker, P. A. (2005). *Acta Crystallogr D Biol Crystallogr* **61**, 449-457.

Rodriguez, D., Sammito, M., Meindl, K., de Ilarduya, I. M., Potratz, M., Sheldrick, G. M. & Uson, I. (2012). *Acta Crystallogr D Biol Crystallogr* **68**, 336-343.

Rodriguez, D. D., Grosse, C., Himmel, S., Gonzalez, C., de Ilarduya, I. M., Becker, S., Sheldrick, G. M. & Uson, I. (2009). *Nat Methods* **6**, 651-653.

Schmidberger, J. W., Bate, M. A., Reboul, C. F., Androulakis, S. G., Phan, J. M., Whisstock, J. C., Goscinski, W. J., Abramson, D. & Buckle, A. M. (2010). *Plos One* **5**, e10049.

Schroder, G. F., Levitt, M. & Brunger, A. T. (2010). *Nature* **464**, 1218-U1146.

Stokes-Rees, I. & Sliz, P. (2010). *Proc Natl Acad Sci U S A* **107**, 21476-21481.

Terwilliger, T. C., Grosse-Kunstleve, R. W., Afonine, P. V., Adams, P. D., Moriarty, N. W., Zwart, P., Read, R. J., Turk, D. & Hung, L. W. (2007). *Acta Crystallogr D* **63**, 597-610.

Terwilliger, T. C., Grosse-Kunstleve, R. W., Afonine, P. V., Moriarty, N. W., Adams, P. D., Read, R. J., Zwart, P. H. & Hung, L. W. (2008). *Acta Crystallogr D* **64**, 515-524.

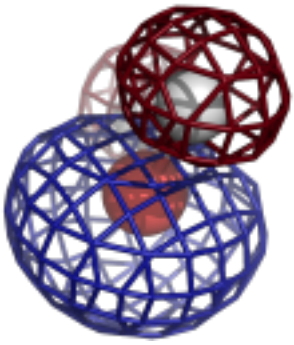
Terwilliger, T. C., Grosse-Kunstleve, R. W., Afonine, P. V., Moriarty, N. W., Zwart, P. H., Hung, L. W., Read, R. J. & Adams, P. D. (2008). *Acta Crystallogr D* **64**, 61-69.

Terwilliger, T. C., Klei, H., Adams, P. D., Moriarty, N. W. & Cohn, J. D. (2006). *Acta Crystallogr D Biol Crystallogr* **62**, 915-922.

van den Bedem, H., Dhanik, A., Latombe, J. C. & Deacon, A. M. (2009). *Acta Crystallogr D Biol Crystallogr* **65**, 1107-1117.

van den Bedem, H., Wolf, G., Xu, Q. & Deacon, A. M. (2011). *Acta Crystallogr D Biol Crystallogr* **67**, 368-375.

VOLUME FOUR



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXIII

ENSEMBLE REFINEMENT, CABLAM

Table of Contents

- PHENIX News 28
- Crystallographic meetings 29
- Expert Advice
  - Fitting tips #6 - Potential misfitting by a switch of sidechain vs mainchain 30
- FAQ 32
- Short Communications
  - *CaBLAM*: Identification and scoring of disguised secondary structure at low resolution 33
  - Structural Classification of Allergen IgE Epitopes by Hierarchical Clustering 36
  - New Tool: *phenix.real_space_refine* 43
- Articles
  - *cctbx* tools for transparent parallel job execution in Python. III. Remote access 45
  - *phenix.ensemble_refinement*: a test study of apo and holo BACE1 51

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

FEM: Feature Enhanced Maps (Pavel V. Afonine)

Interpretation of a crystallographic map is a means of obtaining an atomic representation of a crystal structure or the map itself may serve as the crystal model. There are number of factors that affect quality of crystallographic maps that in turn affect difficulty (or even feasibility) of their interpretation and quality of resulting model of crystal structure, and include:

- finite resolution of measured reflections;
- incompleteness of data (missing reflections within the resolution range of the measured data);
- experimental errors in measured reflections;
- errors in atomic model parameters.

These factors a) result in artificial peaks in the map that may be confused with the signal and therefore erroneously interpreted in terms of atomic model, b) introduce noise that may obscure the signal and c) may distort the signal in various ways.

Another fundamentally different contributor to the difficulty of map interpretation is that not all the signal has the same strength. For example, a strong signal arising from a heavy atom derivative may easily obscure a very weak signal (that may be at or below the noise level) arising from a partially occupied very mobile ligand or residue side chain alternative conformation or even hydrogen atoms.

The combination of the map artifacts arising from data and model errors with the signal variation problems makes map interpretation for macromolecules never unambiguous, trivial or unique.

A procedure is being developed that produces a map with less noise and stronger weak signal. In a nutshell it consists of two steps of map and Fourier map coefficients modification:

- *Eliminate noise as much as possible*. This is done by perturbing map just enough to shuffle the noise but leave the signal unchanged or changed insignificantly. Repeating such perturbation many times and combing resulting maps (by averaging or intersection, or both) is expected to reduce the artifacts but retain the signal. This is based on the Dusan Turk's idea of kick maps calculation, but takes it further by not only "kicking" atomic coordinates but also perturbing all possible variables that pertain to map calculation, such as data completeness, ways of filling missing F_{obs}, calculation of m and D for $2mF_{obs}$-DF_{model} map, varying components that go into Fmodel calculation, many trials of random elimination of largest F_{obs}-F_{model} outliers and more.
- *Equalize weak and strong signal*: equalize signal in the map such that strong and weak signal become of a similar strength. This is a known procedure of local map scaling used in some crystallographic applications (density modification and model building), and routinely used in digital image processing for local contrast improvement. There are multiple ways of doing this: from scaling map by standard deviation locally to local map histogram equalization. We are still investigating which of possible options performs best in this context.

More details, including usage instructions, can be found at www.phenix-online.org -> Presentations -> Feature Enhanced Map

The final algorithm and implementation details will be published elsewhere.

New tool: *phenix.real_space_refine*

An announcement of a new tool is included in the short communications section.

New features

New REEL features

Editing molecules and generating restraints has been simplified in REEL. The ability to add and delete atoms is now available on the toolbar. Naturally it is best to load a restraints CIF file (not a Chemical Components data CIF file) as the starting point and run *eLBOW* on the resulting molecule to generate the best possible restraints.

The ability to view the restraints actually used in a refinement has been added. By default, the bonding of a molecule loaded from a PDB file is performed using a distance-based approach. Calling REEL with both a PDB file and a geometry (`.geo`) file output from `phenix.refine` (or other restraints based program) will use the actual restraints defined and allow viewing of actual and ideal values.

Crystallographic meetings and workshops

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Seminar, Bates College, Lewiston, ME, July 26-27, 2014

A seminar series preceding the GRC meeting.

Biophysical Society 58th Annual Meeting, February 15-19, 2014

An IYCr2014 symposium entitled "Celebrating 100 Years of Crystallography: X-Rays Are Photons Too" will be of interest to all crystallographers. This year the annual Biophysics101 session will be on tips for biophysicists who want to use the crystallographic technique. Check the website for details as the date approaches.

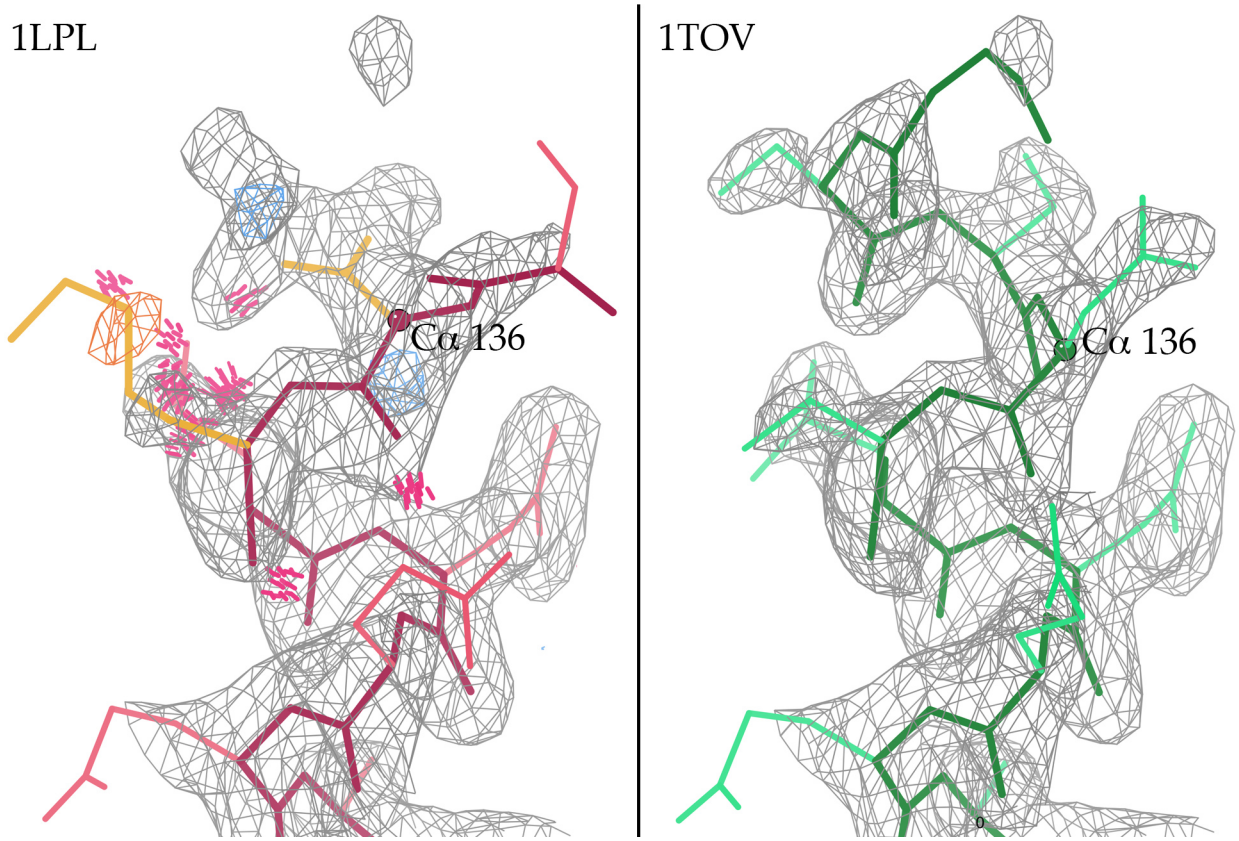


Figure 1. Example of CAP-Gly domain at 1.77Å resolution from 1LPL.

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Conference, Bates College, Lewiston, ME, July 27-August 1, 2014

A very interesting and important meeting for protein crystallographers.

Expert advice

Fitting Tip #6 – Potential misfitting by a switch of sidechain vs mainchain

Jane Richardson and Bryan Arendall, Duke University

One thing that can sometimes go wrong in fitting a protein model is to put a sidechain into what should be backbone density and vice-versa. This usually happens either near a chain terminus or when coming in or out of a disordered loop. This is incorrect, of course, and will usually produce poor geometry even if that residue is really the chain terminus. But it is even more serious if there should have been more residues, since they will either be omitted or wrong because there is not connected density to fit more backbone out from the end of what should really have been the sidechain.

At a chain terminus: Figure 1 shows an example from the 1LPL CAP-Gly domain at 1.77Å resolution (Li 2002), where residues are missing at the N-terminus because of such a sidechain-mainchain switch at the start of an α-helix. The sidechain of Asp 136 was fit into electron density that is really helical backbone, and the mainchain can therefore only continue as far as the end of the Asp sidechain. The Cα of 136 is labeled and marked with a ball in both panels. The left panel shows model and map from the original 1LPL structure, where the fact that something is wrong can be diagnosed by all-atom steric clashes, difference-density peaks, and poor rotamers. The sidechain and mainchain were switched, producing a map that showed density for another residue; after 3 cycles of

rebuilding all validation outliers were gone and nearly a full extra turn of helix was fit at the N-terminus (right panel). This change was the largest of several that gave a 4% drop in Rfree, redeposited as 1TOV (Arendall 2005).

Next to a disordered loop: Any transition between well-ordered structure and a disordered region is equivalent to a chain end, so sidechain-mainchain switches also occur at the ends of loops that are partially or fully disordered. An example is shown in Figure 2, from the 1INP inositol polyphosphate 1-phosphatase at 2.3Å resolution (York 1994). The top panel shows the 97-102 loop in 1INP as deposited, in its discontinuous and ambiguous electron density. That loop also has 4 steric clashes, 4 bond-angle outliers, a Ramachandran outlier, and a poor rotamer (not flagged in the figure). The center panel shows the switch of sidechain vs mainchain around the Cα of Glu 102, and the bottom panel shows the rebuilt loop in its improved density, with validation outliers corrected.

Such problems can occur even at fairly high resolution, because both people and software tend to treat backbone and sidechains as separate fitting tasks. If you want to try rebuilding a sidechain-mainchain switch, practice on 1LPL Asp136, 1INP Glu102, or 1BKR Met109 (harder - probably has two conformations).

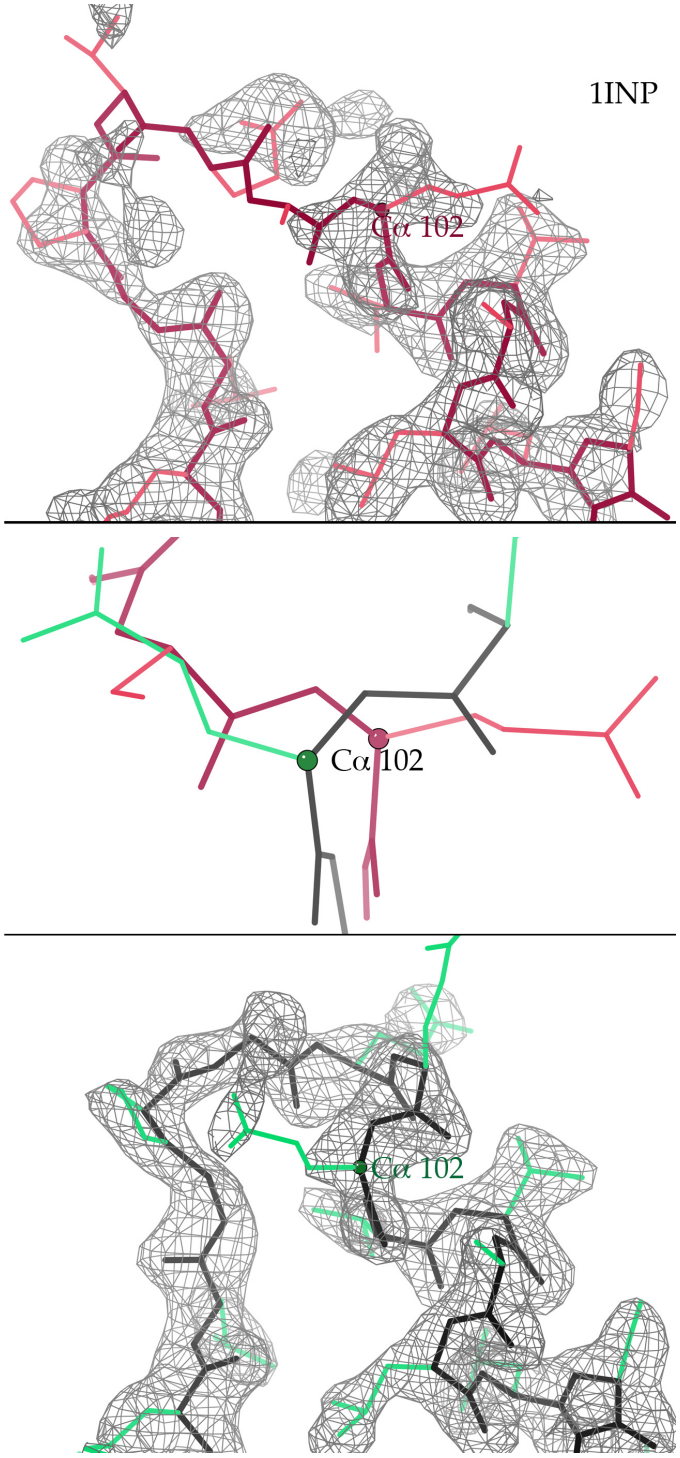


Figure 2. Example from 1INP.

References

Arendall WB III, Tempel W, Richardson JS, Zhou W, Wang S, Davis IW, Liu Z-J, Rose

JP, Carson WM, Luo M, Richardson DC, Wang B-C (2005). "A test of enhancing model accuracy in high-throughput crystallography." Journal of Structural and Functional Genomics 6:1-11.

Li S, Finley J, Liu Z-J, Qiu SH, Luan CH, Carson M, Tsao J, Johnson D, Lin G, Zhao J, Thomas W, Nagy A, Sha B, DeLucas LJ, Wang B-C, Luo M (2002). "Crystal structure of the cytoskeleton-associated protein glycine-rich (CAP-Gly) domain". Journal of Biological Chemistry 277:48596.

York JD, Ponder JW, Chen ZW, Mathews FS, Majerus PW (1994). "Crystal structure of inositol polyphosphate 1-phosphatase at 2.3Å resolution". Biochemistry 33: 13164.

FAQ

After I refine my model, why isn't atom X bound to atom Y when I look at it in a molecular viewer?

Most molecular viewers use the distance between atoms and knowledge of standard bond lengths to draw a line in the viewer pane that symbolises the bond. One reason for this is that the current PDB format does not enforce the specifying of bonds. This means that the lack of bond in a viewer does not mean that the refinement did not have a bond restraint between the specific atoms.

Running *phenix.refine* causes a file with an extension .geo to be written to disk (at the beginning of the refinement by default and optionally using the final model at the end of the refinement). This is the master file for restraints. All restraints used by the refinement engine are written to this file. The restraints are grouped by type and ordered with the restraints with largest residues first. It is human readable text with PDB strings for easy searching.

There are some simple tools for interrogating the information in this file but the most useful could be a new feature of REEL. The bonds can be read and displayed from the .geo using:

```
phenix.reel model.pdb model.geo
```

Other tools can be used at the command line and will be made available upon request.

Contributors

P. D. Adams, P. V. Afonine, B. Arendall, G. Bunkóczi, B. T. Burnley, N. Chakicherla, P. Gros, B. J. Hintze, N. W. Moriarty, D. C. Richardson, J. S. Richardson, C. J. Williams

CaBLAM: Identification and scoring of disguised secondary structure at low resolution

Christopher J. Williams, Bradley J. Hintze, David C. Richardson, and Jane S. Richardson
Department of Biochemistry, Duke University Medical Center, Durham, NC 27710

Correspondence email: christopher.j.williams@duke.edu

Low-resolution electron density presents challenges to fitting the details of the protein backbone. Density between 3 and 4Å generally lacks the details that would allow reliable placement of the backbone carbonyl oxygens. Low-resolution structures are therefore vulnerable to errors in peptide plane orientation. However, both humans and initial fitting programs tend to perform well in placing the general Cα trace, even into otherwise ambiguous density. CaBLAM identifies errors in peptide plane orientation and uses the relatively reliable Cα trace information to identify probable secondary structure disguised by these errors.

CaBLAM uses three parameters to identify modeling errors. μ_{in} and μ_{out} are two pseudodihedrals defined by Cα atom positions (Figure 1a). These parameters capture the Cα trace conformation around each residue. ν is a pseudodihedral that captures the orientation of a residue's peptide plane relative to its preceding neighbor (Figure 1b). ν is defined in a Cα-centric fashion to eliminate dependence on the positions of atoms that may not be reliably modeled. This parameter captures the majority of the outlier behaviors that CaBLAM targets.

These parameters were calculated for all residues in our Top8000 quality-filtered database that passed a B-factor filter of B<30 for all main chain atoms. Three-dimensional contours similar to those used in `phenix.rotalyze` and `phenix.ramalyze` (Chen *et al.*, 2010) were constructed from these high-quality reference data (Figure 2a). Glycine and proline residues showed significantly different behavior from the general case, and so received separate contours (not shown).

In analysis of a low-resolution structure, residues are compared against these contours. Those that fall outside the 95th percentile are considered outliers, and those that fall outside the 99th percentile are considered severe outliers. We

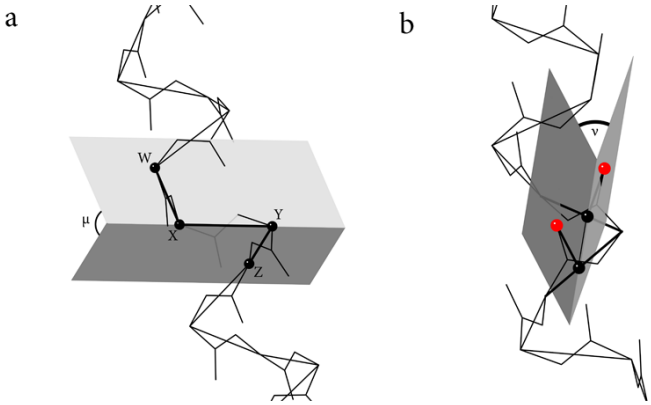


Figure 1: (a) Definition of the μ dihedral from four successive Cα positions. The μ_{in} and μ_{out} of adjacent residues overlap. The dihedral shown is both the μ_{out} of residue X and the μ_{in} of residue Y. (b) Defininition of the ν dihedral. To calculate ν, pseudoatom positions are constructed at the point on each Cα-Cα line closest to the carbonyl oxygen.

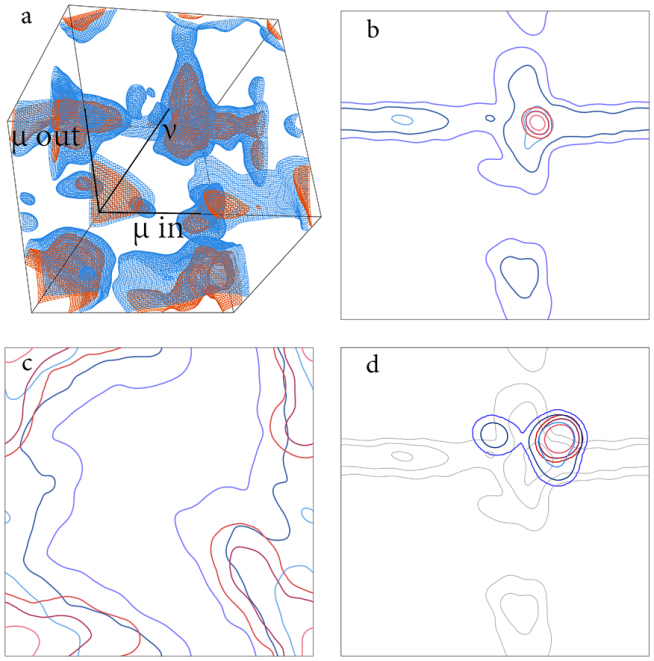


Figure 2: (a) Validation contours for general case residues in the CaBLAM parameter space. Contour levels shown are 99th and 99.5th percentiles. (b) Identification contours for α helix, (c) for β sheet, and (d) for 3_{10} helix, with α in the background for comparison. In b-d, the red contours represent a strict secondary structure definition, the blue a more permissive definition. The contour levels are 99th, 99.9th, and 99.99th percentiles.

found these cutoffs to provide a good balance between coverage of probable errors and acceptance of good structure.

A second set of contours is used to identify probable secondary structure. These contours were derived from the μ_{in} and μ_{out} parameters of residues in the Top8000 that matched hydrogen bonding patterns typical of each secondary structure type. Separate contours for α helix (Figure 2b), β sheet (2c), and 3_{10} helix (2d) were constructed.

Residues found to be CaBLAM parameter space outliers are compared against the 2-D secondary structure contours. Residues that fall within the 99.9th percentile are considered candidates for secondary structure. (In the region of overlap between α and 3_{10}, whichever receives the higher score is considered the more likely structure type.) Individual residues are assembled into complete secondary structure elements if at least three residues in a row match the secondary structure contours and continue for as many residues as the match persists.

Because the accuracy of the Cα trace through a residue is crucial to determining that residue's secondary structure type in CaBLAM, we have constructed an additional set of contours to check the reliability of local Cα geometry. These contours (not shown) are three-dimensional, using μ_{in}, μ_{out}, and the Cα virtual angle. Residues falling outside the 99.5th percentile are considered Cα geometry outliers and are excluded from further secondary structure assessments, as they cannot be guaranteed to have interpretable geometry. This cutoff was chosen as a breakpoint in contour behavior and for good coverage of the conformational space of the secondary structure contours.

CaBLAM is implemented inside PHENIX (Adams *et al*. 2010), making use of the `cctbx` utilities. The most comprehensive access to CaBLAM's validation is through the command line call:

```
phenix.cablam_validate \
  give_full_kin=True filename.pdb
```

This will return a multi-criterion-style kinemage with comprehensive CaBLAM validation and annotation. CaBLAM outliers are marked in purple, and severe CaBLAM outliers are marked in

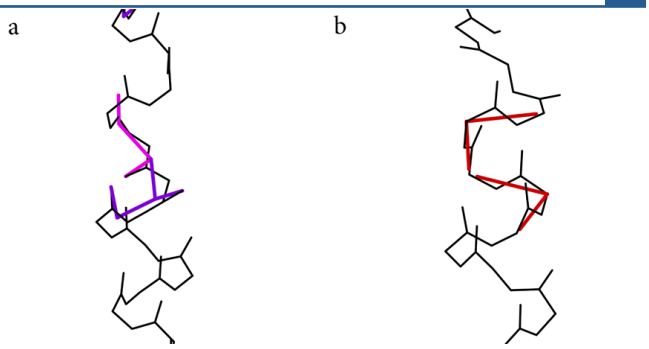


Figure 3: (a) Validation markup for CaBLAM outliers on a helix segment from 2o01. Purple denotes mild outliers, outside the 95th percentile contour. Pink denotes severe outliers, outside the 99th percentile contour. CaBLAM outlier markup follows the trace of the ν dihedral. (b) Validation markup for Cα geometry outliers, outside the 99.9th percentile contour.

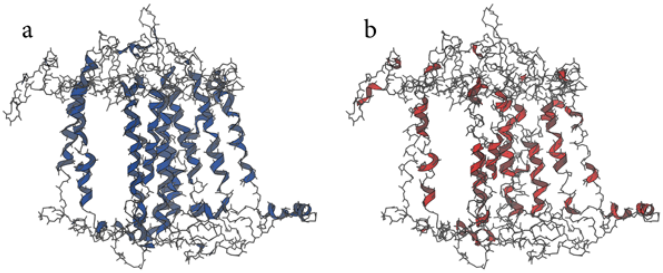


Figure 4: (a) Ribbons as drawn by `phenix.king` for secondary structure elements identified by CaBLAM in 2o01, a structure with many peptide plane orientation outliers. CaBLAM provides good coverage of the transmembrane helices. (b) Ribbons for the helices identified by `phenix.ksdssp`.

pink (Figure 3a). The markup follows the trace of the ν dihedral, since this is the most likely source of the error. Cα geometry outliers are marked in red (Figure 3b). That markup follows the Cα virtual angle of the outlier residue. Selecting a specific markup annotation in the KiNG window will display statistics on that residue's outlier level and its likelihood for secondary structure; for instance, a β-sheet outlier might show "general=0.7% (outlier), beta=85%, alpha=0%".

Rather than forcing users to interpret those statistics for themselves, CaBLAM also writes out pdb-style HELIX and SHEET records for the secondary structure elements it identifies. KiNG (Chen *et al*., 2009) interprets these records as ribbons for easy interpretation (Figure 4a). When `phenix.ksdssp` is used to return HELIX and SHEET records for the challenging Photosystem I structure `2o01`, the hydrogen-bonding-dependent analysis of `ksdssp` results in much less thorough identification of the structure's transmembrane

helices (Figure 4b). Like many of the usual tools of structural biology, `ksdssp` was not designed to address the unusual challenges of low-resolution structures. By focusing on the relatively reliable Cα trace, CaBLAM goes some distance toward filling this gap in our validation capabilities.

References

Adams P.D., Afonine P.V., Bunkóczi G., Chen V.B., Davis I.W., Echols N., Headd J.J., Hung L.-W., Kapral G.J., Grosse-Kunstleve R.W., McCoy A.J., Moriarty N.W., Oeffner R., Read R.J., Richardson D.C., Richardson J.S., Terwilliger T.C. and Zwart P.H.. (2010) PHENIX: a comprehensive Python-based system for macromolecular structure solution. Acta Cryst. D66:213-221.

Chen V.B., Davis I.W. and Richardson D.C. (2009) KiNG (Kinemage, Next Generation): A versatile interactive molecular and scientific visualization program. Protein Science 18:2403-2409.

Chen V.B., Arendall W.B., Headd J.J., Keedy D.A., Immormino R.M., Kapral G.J., Murray L.W., Richardson J.S. & Richardson D.C. (2010) MolProbity: all-atom structure validation for macromolecular crystallography Acta Cryst D66:12-21

Structural Classification of Allergen IgE Epitopes by Hierarchical Clustering

Naveen Chakicherla

Dougherty Valley High School, San Ramon, CA 94582; Intern, Lawrence Berkeley National Laboratory, Berkeley, CA 94720.

Correspondence email: naveenchaki@gmail.com

Abstract

Allergen epitopes derived from the Structural Database of Allergenic Proteins (SDAP) were modeled and classified structurally using single linkage hierarchical clustering. The largest single cluster in this study contained 231 epitopes representing 12 of the 16 species studied. Cluster analysis of 3D models generated for these epitopes revealed that, while sequence identity between the allergen epitopes was generally very low, epitopes in this cluster shared a common tertiary structure in the form of the W-shaped motif with a consensus sequence CR+AKA--SK+SG. The putative shared tertiary structural motif in the epitopes in the largest cluster could also contribute to the significant amount of cross-reactivity clinically observed in several allergens.

Introduction

Over the past decade, there has been a dramatic increase in both incidence and knowledge gain in the area of allergy medicine [1, 2]. The sequences of over 100 food allergens have been published and have been compiled into various public databases [2, 3]. Yet effective treatment of allergic response is still not established, prevention (avoidance) being the most touted cure. Surprisingly, analyses of plant food allergens show that most belong to only a small number of protein superfamilies. Most plant allergens belong to one of only four protein superfamilies, namely prolamins, cupins, profilins, and the Bet V1 family [4, 5]. The World Health Organization/Food and Agriculture Organization (WHO/FAO) guidelines define a protein as potentially allergenic if it either has an identity of at least six contiguous amino acids or a minimum of 35% sequence similarity over a window of 80 amino acids when compared with known allergens [6]. The epitopes, or antigenic determinant of protein antigens, are categorized as linear or conformational epitopes [7], which interact with the paratope based on linear sequence or 3-dimensional structure respectively. Conformational epitopes are stretches of amino acids that interact with an antigen and they are recognized on the basis of their 3D structure. Linear epitopes are recognized on the basis of their amino acid sequence. Although most epitopes are of the conformational kind, the allergen classification has been largely based on sequence similarity, and cross-reactivity predictions based on the WHO/FAO rules result in large numbers of false positives. Furthermore, sequence-based detection could miss several true positives. This study proposes the application of a clustering method to the 3-dimensional structures of allergenic proteins in order to better define the characteristics of the IgE epitopes of allergens in general.

Methods

Sequences of allergenic proteins with known and defined epitopes were obtained from the Structural Database of Allergenic Proteins (SDAP) [3]. In total, 41 full-length sequences of 29 different proteins were submitted to the homology modeling server AS2TS [8], resulting in 35 successful model building runs yielding five putative models for each sequence. Structures of epitopes, as listed in the SDAP, were extracted from these models using JMol [9].

In all, 1655 epitope models were generated, of which 170 models were of t-cell epitopes, 80 models of IgG epitopes and the remaining 1405 were of IgE epitopes. Of the total 1655 models, 446 had a file size of zero, since the segment of the allergen containing that epitope had not been modeled in AS2TS. A structural comparison between all remaining 1209 3D models was performed with SUBCOMB [10] with normalized spatial discrepancy (NSD). The 728424 structure superpositioning jobs were carried out using 30 2.4 GHz processors simultaneously, resulting in 1.3 CPU weeks in total, with a wall clock time of 7.5 hours. The resulting distance matrix was used to perform a

Table 1: Allergen types with known epitopes that were downloaded from SDAP and their representation in the largest cluster in the study.

Allergen	Species - Scientific Name	Species - Common Name	In Cluster 282
Ara h 1; Ara h 2.0101; Ara h 3	*Arachis hypogaea*	Peanut	Y
Asp f 1; Asp f 13; Asp f 2; Asp f 3	*Aspergillus fumigatus*	Fungi	Y
Bet v 1	*Betula verrucosa (Betula pendula)*	White birch	Y
Cha o 1	*Chamaecyparis obtusa*	Japanese Cypress	N
Cry j 1; Cry j 2;	*Cryptomeria japonica*	Japanese Cedar, sugi	Y
Jun a 1.010101; Jun a 3	*Juniperus ashei*	Mountain Cedar	N
Fag e 1;	*Fagopyrum esculentum*	Common Buckwheat	Y
Gly m glycinin G1; Gly m glycinin G2	*Glycine max*	Soybean	Y
Hev b 1; Hev b 3; Hev b 5	*Hevea brasiliensis*	Rubber (latex)	Y
Jug r 1	*Juglans regia*	English walnut	Y
Len c 1.0101	*Lens culinaris*	Lentil	N
Par j 1; Par j 2	*Parietaria judaica*	Pellitory-of-the-Wall	Y
Pen a 1	Penaeus aztecus	Shrimp	Y
Ves v 5	*Vespula vulgaris*	Yellow Jacket	N
Gal d 1	*Gallus domesticus*	Chicken	Y
Pen ch 18	*Penicillium chrysogenum (formerly P. notatum)*	Fungi	Y

cluster analysis with single linkage hierarchical clustering as available in the CCTBX [11].

The 19 clusters with ten or more members were further analyzed for composition, sequence identity, and other characteristics. The epitopes included in the largest cluster, cluster 282 were further analyzed. In order to compare their primary structures the PDB structures of the epitopes needed to be converted back to a primary sequence format. A custom application written by Sebastian Raschka, Michigan State University was used in this study to convert pdb files to fasta sequence format [12]. This custom tool takes a folder of multiple PDB files and writes the fasta sequences for all PDBs into one file that is placed in the same folder. Multiple alignment of the representative members of the largest cluster 282 was conducted using Clustal Omega [13]. In order to compare them structurally, the tool MAPSCI [17] was used.

Results

In this study, we analyzed the 29 allergens with epitopes that are to be found in the SDAP allergen database. These 29 allergens with epitopes originate from 16 distinct plant and animal species (table 1). Three-dimensional structural models were generated for the epitopes of these 29 allergens, resulting in 1209 3D models. Single linkage hierarchical clustering of the epitope structures resulted in a distribution of clusters as depicted in table 2. Clusters varied widely in size,

Table 2: Cluster distribution of the 270 clusters resulting from single linkage hierarchical clustering of structural models of all epitopes of the modeled allergens from SDAP.

Cluster Size	Occurrence
231	1
38	1
20	1
15	3
12	1
11	1
10	11
9	1
8	4
7	1
6	2
5	45
4	37
3	25
2	79
1	77

with the largest cluster having 231 members, while the smallest had one. There were 77 clusters with only one member, and these were eliminated. Clusters with ten or more epitope members were further analyzed. Of the 11 clusters with ten members each, nine clusters contained epitopes of the allergen Aspf1 from the fungi *Aspergillus fumigatus*, while the other two contain epitopes of the allergen, lipid transfer protein; P2 from the weed *Parietaria judaica*. The single cluster with 11 members had epitopes from the two allergens Pench18 and Hev b5 from the fungus *Penicillium chrysogenum* and the kiwi and potato homolog from rubber, *Hevea brasiliensis* respectively. The single cluster with 12 members included epitopes Ara h3 and Gly m glycinin G2 from peanut *Arachis hypogea*, and soybean *Glycine max* respectively. The three clusters with 15 members each all contained epitopes from the fungus *Aspergillus fumigatus.* The single cluster with 20 members had epitopes from the allergen Ara h 2 from peanut *Arachis hypogea* and from the allergen Fag e 1, from the common buckwheat *Fagopyrum esculentum*. The largest cluster containing 231 members is described further below.

Discussion

The majority of the clusters in this structural classification study were composed of epitopes from within the same species, some from different parts of the same protein model. This result suggests that epitopes within allergens are repeated at separate locations in the tertiary structure. However, there were some interesting clusters that showed membership from more than one species. The largest cluster, 282, with 231 members, was analyzed further. Of the 16 allergen species in the entire data set studied, epitopes of allergens from 12 species were included in this largest cluster. They included epitopes from *Arachis hypogaea (*peanut*), Aspergillus fumigatus (*fungus*), Betula verrucosa or Betula pendula (*white birch*), Cryptomeria japonica* (Japanese cedar, sugi), *Fagopyrum esculentum* (common buckwheat), *Glycine max* (soybean), *Hevea brasiliensis* (rubber (latex)), *Juglans regia* (English walnut),*Parietaria judaica* (Pellitory-of-the-Wall), *Penaeus aztecus* (shrimp), *Gallus domesticus* (chicken) and *Penicillium chrysogenum* formerly *P. notatum* (fungus). The four species not represented in this cluster included *Chamaecyparis obtuse* (Japanese Cypress), *Juniperus ashei* (mountain cedar), *Lens culinaris* (lentil) and *Vespula vulgaris* (yellow jacket).

The absence of the mountain cedar epitopes and the yellow jacket epitopes in cluster 282 is easily explained, because AS2TS failed to identify structural models in RCSB for the allergens from these two species. Their epitopes were therefore not represented in our final cluster analysis. The Japanese cypress allergen epitopes had 50 models generated in our analysis. However, it should be noted that the epitope in all cases of this allergen were classified by SDAP as t-cell epitopes that presumably has a different structure than the IgE type epitope that is prevalent in all other epitopes in the cluster 282 (the largest cluster). Hong *et. al.* have shown that pepsin digestion eliminates IgE reactivity but maintains T-cell reactivity suggesting that the two epitopes have different structure and or location in the allergen [16]. The lentil allergen did in fact have 12 models generated by AS2TS, one set modeled using the A chain of the adzuki bean 7S globulin 1 protein structure 2ea7-a and the second set modeled using the PDB structure of a hypothetical Coenzyme A-binding protein from *Thermus thermophiles* bacterium, 1iuk-a.

Sequence analysis of all epitopes in cluster 282 using Clustal Omega gave an alignment length of 30 and an average identity of 7.27% (for default settings) and an alignment length of 25 and an alignment length of 24 and average identity of 8.53% for two HMM iteration and one guide-tree iteration. When all FASTA sequences with exact sequence matches to other FASTA from the same allergen and epitope were removed, as were 1-mer epitopes, this resulted in a smaller cluster of 53 unique sequences. This set gave similar results as the entire set with average length of 6.9 residues, an alignment length of 23 and an average identity of 6.17%.

However, multiple structural analysis of essentially the same set of epitopes from cluster 282 using MAPSCI [17] (epitope PDBs with fewer than four C-alphas were removed, resulting in 45 PDBs) showed a consensus motif in several members of this cluster (See figure 1 and table 3). This motif consisted of two parts with a consensus sequence of `CR+AKA- -SK+SG`. The 3D structural visualization of these epitopes reveals a

Table 3: Summary of cluster 282 epitopes analyzed using MAPSCI.

MAPSCI code	Allergen	SDAP Source Number	AS2TS model number	PDB template	PDB template chain	Epitope type	Epitope start residue	Epitope stop residue
uMS01	arah1	p43237	1	1iuk	a	IgE	311	320
uMS02	arah1	p43237	1	1iuk	a	IgE	559	568
uMS04	arah1	p43237	3	3s7i	a	IgE	578	587
uMS05	arah1	p43237	5	1dgw	y	IgE	578	587
uMS06	arah2.0101	9186485	1	1w2q	a	IgE	39	48
uMS07	arah2.0101	9186485	1	1w2q	a	IgE	127	132
uMS09	arah2.0101	9186485	2	3ob4	a	IgE	49	56
uMS10	arah2.0101	9186485	5	1pnb	b	IgE	143	150
uMS11	Gald1	P01005	1	2p6a	d	IgE	64	74
uMS12	Gald1	P01005	1	2p6a	d	IgE	95	99
uMS13	Gald1	P01005	1	2p6a	d	IgG	55	59
uMS14	Gald1	P01005	1	2p6a	d	IgG	70	74
uMS15	Gald1	P01005	1	2p6a	d	IgG	189	198
uMS19	Parj1	P43217	3	1t12	a	IgE	72	79
uMS20	Parj1	O04404	1	1mid	a	IgE	41	47
uMS22	Parj1	Q40905	1	1t12	a	IgE	41	47
uMS23	Parj1	Q40905	1	1t12	a	IgE	72	79
uMS24	Parj2	P55958	1	2alq	a	IgE	45	50
uMS25	Parj2	P55958	1	2alq	a	IgE	61	70
uMS26	Parj2	P55958	1	2alq	a	IgE	77	86
uMS27	Parj2	P55958	1	2alq	a	IgE	83	91
uMS28	Parj2	P55958	1	2alq	a	IgE	122	129
uMS29	Parj2	O04403	1	fk5	a	IgE	45	50
uMS30	Parj2	O04403	1	fk5	a	IgE	61	70
uMS31	Parj2	O04403	1	fk5	a	IgE	77	86
uMS32	Parj2	O04403	1	fk5	a	IgE	83	91
uMS33	Parj2	O04403	1	fk5	a	IgE	122	129
uMS34	Pena1	11893851	3	2b9c	b	IgE	85	99
uMS35	Pench18	7963902	5	1sh7	a	IgE	401	410
uMS37	arah2.0101	15418705	1	1w2q	a	IgE	127	132
uMS38	arah2.0101	15418705	2	3obq4	a	IgE	49	56
uMS39	arah2.0101	15418705	2	3obq4	a	IgE	117	122
uMS40	Aspf1	166486	1	1jbs	a	IgE	51	60
uMS42	aspf1	p04389	1	1jbs	a	IgE	51	60
uMS43	Aspf2	P79017	1	1eb6	a	IgE	58	64
uMS45	Aspf2	P79017	1	1eb6	a	IgE	181	185
uMS46	Aspf2	P79017	1	1eb6	a	IgE	189	192
uMS47	Aspf2	P79017	1	1eb6	a	IgE	200	204
uMS49	Aspf3	664852	1	1eb6	a	IgE	115	125
uMS52	Cryj2	P43212	5	2x3h	a	T-cell	400	414
uMS53	Fage1	2317670	1	3fz3	b	IgE	360	367
uMS54	Fage1	2317670	1	3fz3	b	IgE	411	416
uMS55	Fage1	2317674	1	2e9q	a	IgE	460	465
uMS56	Fage1	2983941	1	3qac	a	IgE	460	465
uMS57	Fage1	2983941	1	3qac	a	IgE	506	511

W-shaped "basket" motif (figure 2). The first part of the motif showed better quality in the multiple alignment, but with lower consensus than the second part. The second half of the motif showed greater conservation and was more consistently present in all the epitopes in the cluster 282, although the quality was overall lower.

Conclusion

The decision tree approach to evaluating the potential allergenicity of genetically engineered food products [14, 15] proposed in 1996 has since

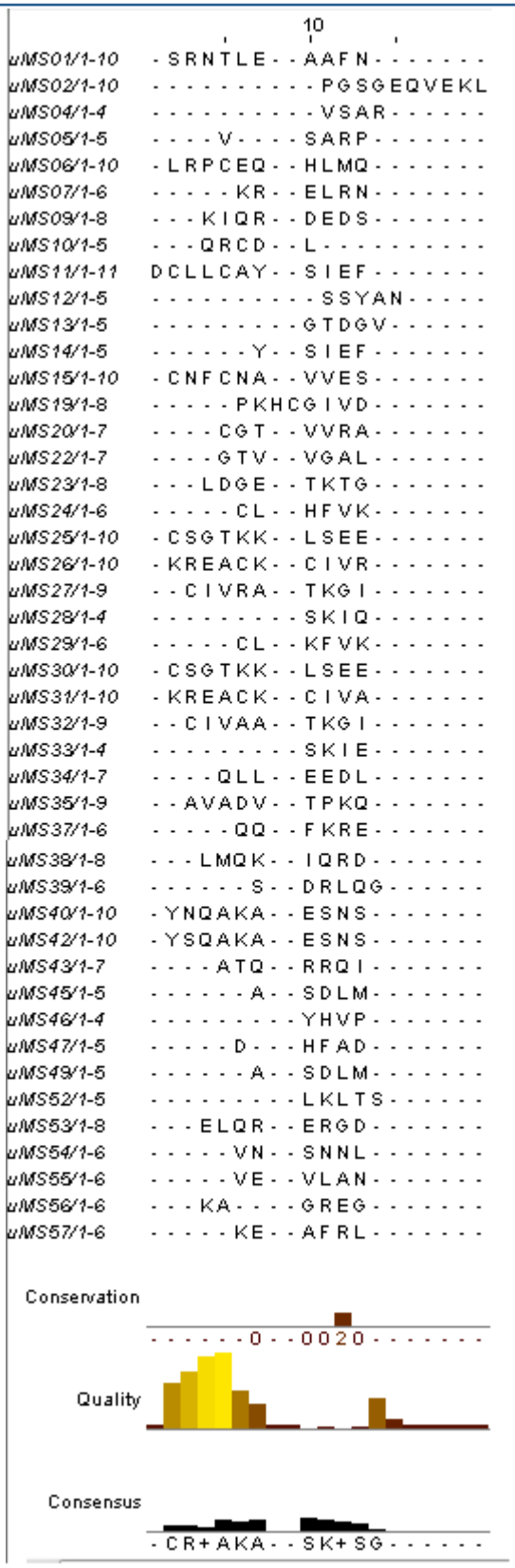


Figure 1. Structure based sequence alignment of members of Cluster 282 using MAPSCI shows a consensus sequence of CR+AKA..SK+SG.

been practiced widely to determine allergenic nature of novel protein products. This strategy uses primary sequence analysis and immunochemistry along with gene source to predict the allergenicity of the novel protein. We have presented here a structural classification of the epitopes from all allergens having epitopes in the Structural Database of Allergenic Proteins, SDAP. The largest single cluster in this study contained 231 epitopes representing 12 of the 16 species studied. Cluster analysis of 3D models generated for these allergens revealed that, while sequence identity between the allergen epitopes was generally very low, epitopes in this cluster shared common tertiary structure in the form of the W-shaped motif with consensus sequence CR+AKA--SK+SG. This study lends support to the recommendation that the decision tree approach to evaluating the potential allergenicity of genetically engineered food products, from the FAO/WHO, should include a stronger component of structural identity, in addition to the current focus on sequence identity. The putative shared tertiary structure motif in the epitopes in the largest cluster in the analysis could also contribute to the significant amount of cross-reactivity clinically observed in several allergens.

Acknowledgements

This study was funded in part by a Student Assistantship in the Physical Biosciences Division at Lawrence Berkeley National Laboratory (Berkeley Lab) in the lab of Dr. Peter H. Zwart. I would like to especially thank my advisor and mentor, Peter H. Zwart, for his guidance and encouragement. I would also like to thank the Berkeley Center of Structural Biology for allowing me the use of their computational facilities for this research. I would like to acknowledge the support of Sebastian Raschka, Ph.D. student at Michigan State University for a tool that converts pdb files to fasta files.

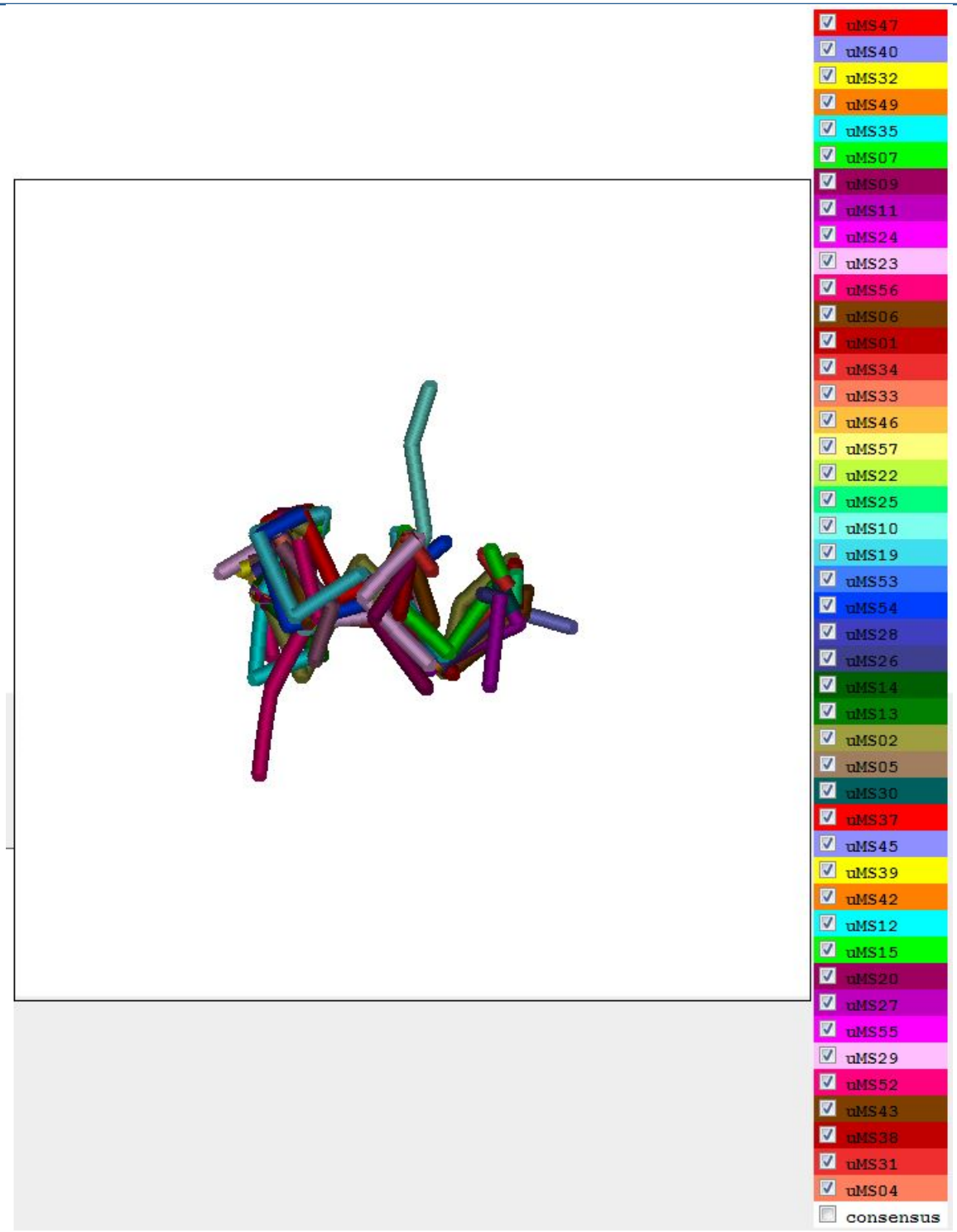


Figure 2. 3D visualization of the members of cluster 282 showing the W-shaped "basket" motif of the cluster. The members of this alignment are available in table 3.

References

1. Ovidiu Ivanciuc, Tzintzuni Garcia, Miguel Torres, Catherine H. Schein, Werner Braun (2009). *Molecular Immunology* 46, 559–568.
2. Christian Radauer, Merima Bublin, Stefan Wagner, Adriano Mari, Heimo Breiteneder (2008) *J Allergy Clin Immunol* 2008;121:847-52.
3. Ivanciuc, O., Schein. C. H., and Braun, W. (2003). *Nucleic Acids Res.* 31(1). 359-362.
4. Heimo Breiteneder, E.N. Clare Mills (2005). *Biotechnology Advances* 23, 395–399.
5. Christian Radauer, and Heimo Breiteneder (2007). *J Allergy Clin Immunol*. 120:518-25.
6. FAO/WHO, 2001, http://www.fao.org/es/ESN/food/pdf/allergygm.pdf; 2003, http://www.codexalimentarius.net/download/report/46/Al0334ae.pdf
7. Ovidiu Ivanciuc, Catherine H. Schein, Tzintzuni Garcia, Numan Oezguen, Surendra S. Negi, Werner Braun. (2009). *Regulatory Toxicology and Pharmacology* 54, S11–S19.
8. Zemla A. (2003). *Nucleic Acids Research*, Vol. 31, No. 13, pp. 3370-3374.
9. Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/
10. Kozin, M. B., and Svergun, D. I. (2001) *J. Appl. Crystallogr.* 34, 33.
11. http://cctbx.sourceforge.net/
12. http://sebastianraschka.com/webapps.html
13. Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson, and Desmond G Higgins (2011). *Mol Syst Biol.* 7: 539.
14. Dean D. Metcalfe, James D. Astwood, Rod Townsend, Hugh A. Sampson, Steve L. Taylor & Roy L. Fuchs (1996). *Critical Reviews in Food Science and Nutrition* 36, Suppl. 001.
15. http://www.fao.org/docrep/007/y0820e/y0820e05.htm
16. Hong SJ, Michael JG, Fehringer A, Leung DY. (1999). *J Allergy Clin Immunol*. Aug;104(2 Pt 1):473-8.
17. Ivaylo Ilinkin, Jieping Ye, Ravi Janardan (2010). *BMC Bioinformatics*, 11:71.

New tool: phenix.real_space_refine

Pavel V. Afonine[a], Jeffrey J. Headd[b], Thomas C. Terwilliger[c] and Paul D. Adams[a,d]

[a]*Physical Biosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720-8235*
[b]*Department of Biochemistry, Duke University Medical Center, Durham, NC 27710*
[c]*Los Alamos National Laboratory, Los Alamos, NM 87545-0001*
[d]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720-1762*

Correspondence email:e-mail: PAfonine@lbl.gov

The aim of this article is to announce and briefly document *phenix.real_space_refine* – a new command line tool for refinement of a model against a map.
The program is being developed with the following goals:

- fit a model as well as possible into a map that can vary in quality: from low to high resolution;
- produce models without poor validation metrics, such as clashes or rotamer outliers;
- correct obvious model completeness problems, such as incomplete residues or simple alternative conformations (that are unambiguously defined by the map);
- fast execution;
- have a large convergence radius.

Not all the above goals are implemented at present and are therefore the subject of future work. Also, the program is still under active development, which implies that some protocols or/and functionality may change or improve in future versions.

The current workflow of *phenix.real_space_refine* includes:

- reading a PDB file with an initial model and a map that can be given as an actual map in CCP4 or X-plor format or as Fourier coefficients (MTZ file). There are no assumptions made about the technique used to obtain the map (X-ray, neutron, electron microscopy, etc);
- refinement macro-cycle (repeated execution of various refinement techniques listed below);
- the results of running *phenix.real_space_refine* are two PDB files: one contains the final refined model and the other one is a multiple model file containing all intermediate models that were generated during refinement.

The refinement macro-cycle consists of five refinement steps:

1. *Model morphing* (Terwilliger *et al.*, 2012) is performed first and repeated until it shifts the model by no more than 0.3Å or the number of iterations exceeds a specified threshold.
2. *Local real-space fitting* of individual residues that the program identifies as outliers. Residues that need local real-space refinement (outliers) are chosen based on map correlation, rotameric state or clashes with neighboring atoms. First, the procedure fits main-chain atoms and then performs local side-chain rotamer searches to find a valid rotameric state that best fits the map. For each newly fit residue torsion angle restrains are added to maintain its current conformation in subsequent global refinement (so the residue doesn't become an outlier again). These restraints are discarded or updated at the next macro-cycle.
3. *Global gradient-driven* (lbfgs) refinement using the target function, $T = T_{MAP} + w * T_{RESTRAINTS}$, where the restraints target, $T_{RESTRAINTS}$, consists of a set of standard restraints (Grosse-Kunstleve *et al.*, 2004; Afonine *et al.*, 2012) as well as optional extra restraints such as secondary structure, C_β deviation and Ramachandran plot restraints (Headd *et al.*, 2012). The weight 'w' is determined automatically best testing of several values: this array of values is tried until a value is found that yields a refined model with bond and angle deviations from ideal values lower than predefined limits (which are user-controlled parameters). T_{MAP} is the negated sum of map values calculated at atomic centers.
4. *Simulated Annealing* (SA) is optional using the same implementation from *phenix.refine* (Afonine *et al.*, 2012).
5. *Rigid-body* refinement is also optional and currently can only be performed on the entire

model (as opposed to smaller rigid groups) limiting its usefulness somewhat.

Current known limitations, inefficiencies and planned future functionalities include:

- model may not contain hydrogen atoms;
- morphing is done only on the whole model and not on individual parts;
- the procedure could be stabilized by allowing single atoms to be harmonically restrained to their current positions;
- there is not yet an GUI (command line only);
- there are some missing restraint features: (no reference model or NCS);
- there is no model completion;
- non amino-acid residues are not handled in morphing and local real-space fitting.
- There is no ADP (individual, TLS) and occupancy refinement.

A detailed description of the program and underlying tools will be published elsewhere.

Usage examples:

```
phenix.real_space_refine model.pdb ccp4_formatted_map.map

phenix.real_space_refine model.pdb map_coefficients.mtz

phenix.real_space_refine model.pdb map_coefficients.mtz label='FOFCWT,PHFOFCWT'
```

To see all parameters: `phenix.real_space_refine -h`

References

1. Afonine, P.V., Grosse-Kunstleve, R.W., Echols, N., Headd, J.J., Moriarty, N.W., Mustyakimov, M., Terwilliger, T.C., Urzhumtsev, A., Zwart, P.H. & Adams, P.D. (2012). "*Towards automated crystallographic structure refinement with phenix.refine*". Acta Cryst. D68, 352-367.
2. Grosse-Kunstleve, R.W., Afonine, P.V. and Adams, P.D. (2004). "cctbx news: Geometry restraints and other new features". Newsletter of the IUCr Commission on Crystallographic Computing, 4, 19-36.
3. Headd, J.J., Echols, N., Afonine, P.V., Grosse-Kunstleve, R.W., Chen, V.B., Moriarty, N.W., Richardson, D.C., Richardson, J.S. & Adams, P.D. (2012). "*Use of knowledge-based restraints in phenix.refine to improve macromolecular refinement at low resolution*". Acta Cryst. D68, 381-390.
4. Terwilliger, T.C., Read, R.J., Adams, P.D., Brunger, A.T., Afonine, P.V., Grosse-Kunstleve, R.W. & Hung, L.-W. (2012). "*Improved crystallographic models through iterated local density-guided model deformation and reciprocal-space refinement*". Acta Cryst. D68, 861-870.

cctbx tools for transparent parallel job execution in Python. III. Remote access

Gábor Bunkóczi

Department of Haematology, University of Cambridge, Cambridge, UK

Correspondence email: gb360@cam.ac.uk

Introduction

CPU development in the past decade, especially the introduction of multi-core CPUs, has made an increasing number of structural biology problems computationally feasible, given the availability of a suitably large computing cluster. Although such a cluster is not always available, the necessary computing power is often present, but distributed over a number of workstations. Harnessing the collective power of this workstation pool can be a sufficient substitute to a computing cluster for certain calculations. However, a number of these processing nodes must be accessed remotely.

On the other hand, even when a cluster is available, access is often complicated by network security rules and usage restrictions that can explicitly forbid users running specific processes (e.g. graphical user interfaces) on the cluster's head node (although it is more frequent that system administrators simulate direct access to the cluster with remote command execution). While such a solution is transparent for most uses, it can interfere with the `processing` (Bunkóczi & Echols, 2012) module (for brevity, the `libtbx.queuing_system_utils` package will be implicit in all unqualified module names, except stated otherwise), because of potentially different base directories used when translating relative file paths. In addition, the overhead of job polling can increase significantly, since each request must be authenticated by the selected network protocol (normally `ssh`). Finally, this mode of access implicitly assumes a shared file system, and network latency can become a problem.

The `libtbx.queuing_system_utils.remote` module offers a convenient and transparent solution to these problems.

Remote execution

The core architecture resembles a common client-server setup, although in this case the server process is started up by the client after establishing a network connection with the remote machine, and can only use a single communication channel, namely the one that is open towards the client that initiated the connection. The server process has two responsibilities: (1) job execution and (2) data exchange with the client. Management of job execution is done using the `scheduling` module (Bunkóczi & Echols, 2012). Scheduling is, however, delegated to the client, which is responsible for tracking how many jobs are being executed. The server behaves as an "infinite resource", and every time a new job execution request is made, it is fulfilled. Data exchange with the client is managed by the main thread. This task is very much I/O bound, and it spends most of its time waiting for requests. To achieve efficient and regular job lifecycle management, job execution is handled by a separate thread.

On the client side, a proxy object is created that holds communication streams to the server, and acts as the only point of contact. Similarly to the `processing.QueueHandler` object, it exports a `Job` factory function that can be used to create a remote job on the server machine. This creates an object that shares the same interface as `multiprocessing.Process` or `threading.Thread` from the Python Standard Library. Job execution data is sent over serialized (using the Python Standard Library `pickle` module) to the server, where it gets reconstructed and executed. Results are sent back using a similar mechanism. Therefore, input data (including the function to execute) and results have to be pickleable. It is important to note that the client object does not have its own thread: submission, job status checks and result downloads are performed on the main thread, whenever requested. This may add a certain overhead to the whole program, but this is normally not serious.

For the network communication to work, the `remote` module on both ends of the connection

should be the same version. In addition, for serialization to work, input and output objects (including the called function) should also be the same version.

The `remote` module tries to resemble direct execution as closely as possible. For example, anything that is printed by the executing process at the remote machine is transferred and printed onto the local computer's terminal.

Management of the network connection has been kept entirely independent in the design. The client only requires the input and output file handles of the server to maintain communication, and would therefore work with any network protocols (although the server code may need adjustment).

Fetching job results

It is assumed that each remote job returns a value that needs to be sent back to the client (note that the design of the `scheduling` module allows the execution of jobs without capturing their return value). This is normally done by passing one end of a data queue to the underlying execution process. However, for a remote job this is done using a two-stage approach.

1. The server process configures each `ExecutionUnit` with a `RetrieveProcessor`. The appropriate data queue is selected at time of creation. This is used to return results from jobs to the server's `Manager` object.
2. The server sends this result back to the client. The client attaches it to the corresponding `Job` object, and the parent `ExecutionUnit` can extract it using a `FetchProcessor`, which is a new processor class developed for this task.

An alternative option would be to pass the details of a data queue to the remote machine, and make the executing processes return their result via a network socket. This would transform the whole return procedure into a single-step process, and would also make the `FetchProcessor` superfluous. Unfortunately, this approach makes the assumption that the local machine can be contacted from the remote (which is often invalid), and would also require development of network server process that would handle incoming connections and channel results to the main program.

Configuration

Remote execution is not tied to any particular execution mode: in fact, it would be possible (even if less useful) for a remote machine to delegate job execution even further to another remote machine. Execution details are controlled by the programmer, and are communicated to the server process via command-line arguments. The current server implementation provides two customizations: a factory function for the executing process and a factory function for the `RetrieveProcessor`'s data queue. Naturally, these factory functions have to be serialized to be able to act as command line arguments, but the `remote` module provides convenience functions to do this (Figure 1).

Changes in the `scheduling` module

For the `remote` module, it is important that job lifecycle management and resource allocation are separated. This saves a tiny amount of unnecessary scheduling overhead, but more importantly, eliminates a possible mismatch between the allocated number of CPUs at the local and the remote machine. To this end, the resource allocation responsibility of the `scheduling.Manager` object is separated out into a series of new components. These encapsulate various strategies in resource management:

- `Allocated`: maps jobs onto a number of predefined `ExecutionUnit` instances. This is the behaviour of the `scheduling.Manager` object.
- `Unlimited`: each new request results in the creation of a new `ExecutionUnit` instance (the exact type is described with a factory function). There is no limit to the number of concurrently active `ExecutionUnit`s. This can work well with common queuing systems, which also possess built-in scheduling functionality. After the job has finished, the `ExecutionUnit` is destroyed, and recreated when necessary.
- `UnlimitedCached`: similar to `Unlimited`, but after jobs finish, the idle `ExecutionUnit` is stored, and preferentially re-used. This is designed to save time on the data queue

```
from libtbx.queuing_system_utils import remote

import subprocess

server = subprocess.Popen(
  (
    "ssh",
    "mymachine",
    remote.server_process_command_line(
      job_factory = multiprocessing.Process,
      queue_factory = multiprocessing.Queue,
      ),
    ),
  stdin = subprocess.PIPE,
  stdout = subprocess.PIPE,
  )
client = remote.SchedulerClient(
  instream = server.stdout,
  outstream = server.stdin,
  )

...use client.Job as multiprocessing.Process...

client.close()
```

Figure 1
Creating a remote job class. Execution on the remote system is made using `multiprocessing.Process` (`job_factory` argument), and these processes use a `multiprocessing.Queue` data queue to return a result to the server process (`queue_factory` argument). In this case, `ssh` is used to establish a network connection between the two hosts, but alternatives are available (e.g. `paramiko`, https://github.com/paramiko/paramiko). Input and output file handles are opened and passed onto the client).

creation, since apart from an initial growth period, no new `ExecutionUnit` instances are created in a stationary execution flow.

The previous `scheduling.Manager` class is replaced with a function that creates the new `scheduling.Scheduler` class with the `Allocated` strategy for backward compatibility.

In addition, a new processor class (`FetchProcessor`) is provided to be used with primarily remote jobs. However, it can also be employed with other job types, although the use a data queue may not be possible to avoid (Figure 2).

Network optimizations

Network data exchange can be slow, since there is a delay for a request to reach the remote host and a response to arrive. For this reason, the module packages submission and status requests, and only executes them when a certain number of requests has been exceeded, or a certain time limit has been reached. Optimal parameters seem to depend on the speed of the network (fast/slow) and the number of CPUs allocated.

File operations

The module can send input data along through the network connection, and therefore does not depend on a shared file system. However, when multiple calculations require the same input data, it could be more efficient to transfer this as a file. While there is no generic support built into the module (since file transfer is entirely a network operation), the basic case when the file system is shared between the two systems is supported. This is achieved by changing the working

```
...create client...

from libtbx.queuing_system_utils import scheduling

manager = scheduling.Scheduler(
  handler = scheduling.Allocated(
    units = [
      scheduling.ExecutionUnit(
        factory = client.Job,
        processor = scheduling.FetchProcessor.Unpack(),
        )
      for i in range( 8 )
      ]
      + [
      scheduling.ExecutionUnit(
        factory = multiprocessing.Process,
        processor = scheduling.RetrieveProcessor(
          queue = multiprocessing.Queue(),
          ),
        priority = 1,
        )
      for i in range( 4 )
      ]
    )
  )

...use manager as before...
```

Figure 2

Using remote jobs with the `scheduling` module. The example allocates 12 CPUs, 4 on the local and 8 on a remote machine. It is also possible to use multiple remote machines. The `priority` argument controls the preference of selecting nodes. In the example, the CPUs in the local machine are used first.

directory of the server to correspond to that of the client, so that relative paths are equivalent.

Job and queue factories

The Python Standard Library `pickle` module does not support serialization for object methods. However, factory functions are sometimes objects methods, and it is important for efficiency reasons that the object instance is not recreated every time a factory is called. For example, when using the `Queue` method of a `multiprocessing.Manager` object to create a data queue, it is important that the `Manager` object is created only once, because it spawns a new process to manage the connections, and it would be very inefficient for each data queue to have its own managing process. The remote `module` provides the `RemoteFactory` class to help with this issue. It requires a deferred call to the object constructor, and the name of the object method that will act as the factory function. The object is created on first access, and is reused for subsequent calls. A full example is shown on Figure 3.

```
from libtbx.queuing_system_utils import scheduling
from libtbx.queuing_system_utils import remote
import subprocess
import multiprocessing

arg = remote.server_process_command_line(
  job_factory = multiprocessing.Process,
  queue_factory = remote.RemoteFactory(
    calculation = multiprocessing.Manager,
    method = "Queue",
    ),
  )

server = subprocess.Popen(
  ( "ssh", "mymachine", arg ),
  stdin = subprocess.PIPE,
  stdout = subprocess.PIPE,
  )
client = remote.SchedulerClient( server.stdout, server.stdin )

manager = scheduling.Scheduler(
  handler = scheduling.Allocated(
    units = [
      scheduling.ExecutionUnit(
        factory = client.Job,
        processor = scheduling.FetchProcessor.Unpack(),
        )
      for i in range( 4 )
      ]
    )
  )
import math
pfor = scheduling.ParallelForIterator(
  calculations = ( math.sin, ( i, ), {} ) for i in range( 100 ) ),
  manager = manager,
  )
for ( parms, res ) in scheduling.SubmissionOrder( parallel_for = pfor ):
  try:
    result = res()

  except Exception, e:
    print "Job: %s : exception '%s'" % ( parms, e )

  else:
    print result

client.close()
```

Figure 3
Parallel execution of an arbitrary calculation (in this case, `math.sin`) on a remote machine. Data queues are created using an object method, which is serialized using the `RemoteFactory` helper class.

Table 1
Benchmark results of executing 100 fast (100 ms) jobs on 4 CPUs using `multiprocessing.Process` as execution technology via a `scheduling.Manager` in various setups. The remote machine is about 5000 miles away from the local.

Description	Timing
Direct (local execution)	5.43 s
Same machine, remote execution via `subprocess`	10.15 s
Same machine, remote execution via SSH	10.58 s
Remote machine via remote execution (SSH)	15.36 s

Table 2
Benchmark of executing 10 jobs that take about 1 s each, using 10 CPUs on a Sun Grid Engine cluster. Sun Grid Engine is accessed using the `processing` module. The remote machine is the same as in the previous benchmark.

Description	Timing
Direct (local execution)	14.98 s
Remote machine via remote execution (SSH)	15.83 s

Overheads

Analysis of benchmarks (Table 1) shows that there seems to be a per-job overhead that consists of two parts:

1. Serialization overhead, which includes pickling and unpickling the requests and the results. This is about 50 ms/job, and calculated by comparing the timings of direct execution and the no-network remote execution.
2. Network overhead, which scales with the speed of the connection. For a very fast network (and physically close machines), this is negligible, but starts to increase with the distance. For the test case, it contributes an additional 50 ms overhead for machines that are about 5000 miles apart and connected by a fast network.

Although this looks quite significant (effectively tripling the execution time), it is only because the jobs are very fast, and also the job startup overhead is very small. When switching to longer calculations (Table 2), and execution technology that involves a larger overhead, the difference between the two are negligible.

References

Bunkóczi, G. & Echols, N. (2012). *Computational Crystallography Newsletter* **3**, 37-42.

phenix.ensemble_refinement: a test study of apo and holo BACE1

B. Tom Burnley and Piet Gros
Crystal and Structural Chemistry, Bijvoet Center for Biomolecular Research, Utrecht University
Correspondence email: B.T.Burnley@uu.nl & P.Gros@uu.nl

Introduction

`phenix.ensemble_refinement` (Burnley et al. 2012) combines molecular dynamic (MD) simulations with X-ray structure refinement to generate ensemble models fitted to diffraction data. It is an evolution of the 'time-averaging' method first proposed by Gros et al. in 1990 (Gros, van Gunsteren, and Hol 1990) and now utilizes a maximum-likelihood target function, a dual explicit-bulk solvent model and introduces a TLS fitting procedure that absorbs rigid-body motions such that short MD simulations can be used to sample local disorder. The resulting ensemble model represents, as a Boltzmann-weighted population of structures, the simulation trajectory and provides implicit modeling of anisotropic and anharmonic motions. This family of structures provides two main advantages: 1) a reduction in R_{free} compared with traditional single structure models and 2) quantification and visualization of protein dynamics that have been demonstrated to correlate with biological function. Methodological implementation and testing is described in detail in Burnley et al. 2012; here we focus on a practical usage of the method.

For this test case we examine β-Secretase (β -site amyloid precursor protein-cleaving enzyme1; BACE1), in the apo and holo state. BACE1 is a transmembrane aspartic protease that cleaves β-amyloid precursor protein and is the putative rate-limiting enzyme in the amyloid β-peptide (Aβ) pathway (Sinha 1999). Aβ is likely responsible for the Alzheimer's disease cascade and therefore BACE1 is a primary target for the development of inhibitors to treat/prevent Alzheimer's disease (Xu et al. 2011). The importance of this protein has resulted in multiple entries in the PDB (Berman et al. 2000). We selected two 1.6 Å resolution datasets deposited by Xu and co-workers (Xu et al. 2011) of the protease ectodomain in the apo state and holo state complexed with a nonpeptide inhibitor (*N0*-[(2*S*,3*R*)-4-(cyclopropylamino)-3-hydroxy-1-phenylbutan-2-yl]-5-(methylsulfonylamino)-*N*-[(1*R*)-1-phenylethyl]benzene-1,3-dicarboxamide; BSIIV) for analysis by ensemble refinement (PDB codes 3TPJ and 3TPP respectively). Here we detail the phenix.ensemble_refinement process, from preparing input data through parameter optimization to structure analysis.

Preparing input data

`phenix.ensemble_refinement` uses a set of input files similar to `phenix.refine (Adams et al. 2010)`, i.e. structure (e.g. .pdb format) and reflection files (e.g. .mtz) are required and specified parameter definitions (e.g. .eff) and ligand restraints (e.g. .cif) should be supplied as needed. Ensemble refinement is dependent on the quality of the input structure therefore the input structure should be post-traditional refinement and thus ready for deposition, or as deposited, in the PDB. It should be noted that if TLS was used in the refinement protocol the TLS contributions must be present in the atom records (see `phenix.tls`).

For this test case the PDB structures and structure factors were downloaded from the PDB_REDO server (Joosten et al. 2010). One of the advantages of the PDB_REDO server is that it automatically builds and refines missing side-chains, and the usage of complete side-chains is recommended for ensemble refinement. However, we do not recommend building long sequences (>~4 residues) of highly disordered regions such as loops or termini if there is not sufficient electron density to support placement when using standard model building procedures. In the BACE1 structures loop 153-173 and loop 308-319 have missing residues, as do both the N and C termini and after examining in these regions. It was not possible to build any additional residues. `phenix.ready_set` was then used to generate ligand restraints and to add explicit hydrogen atoms to the PDB file. The structures were further refined using `phenix.refine` and TLS groups were determined using `phenix.find_tls`. Table 1 shows the refinement statistics. Prior to use with `phenix.ensemble_refinement` any alternative conformations should be removed and

Table 1. BACE1 apo and holo dataset and refinement statistics as deposited in the PDB and following re-refinement using `phenix.refine` and `phenix.ensemble_refinement`.

	Apo	Holo
PDB code	3tpj	3tpp
Space group	$C222_1$	$C222_1$
Unit-cell parameters		
a (Å)	104.4	104.5
b (Å)	128.7	128.2
c (Å)	76.7	76.5
Resolution range (Å)	35.8 - 1.6	49.1 - 1.6
PDB structure		
R_{work} (%)	17.8	15.5
R_{free} (%)	19.6	18.0
phenix.refine		
R_{work} (%)	14.4	14.0
R_{free} (%)	15.8	16.1
Geometric rmsd		
Bond lengths (Å)	0.0	0.0
Bond angles (°)	1.3	1.4
Dihedral angles (°)	12.5	13.1
Ramachandran		
Disallowed residues (%)	0.0	0.0
Allowed residues (%)	2.3	1.3
Favored residues (%)	97.7	98.7
phenix.ensemble_refinement		
R_{work} (%)	12.1	12.1
R_{free} (%)	14.7	14.6
Geometric rmsd (centroid)		
Bond lengths (Å)	0.008	0.007
Bond angles (°)	1.113	1.151
Dihedral angles (°)	8.38	8.51
Geometric rmsd (per structure)		
Bond lengths (Å)	0.014	0.013
Bond angles (°)	1.713	1.728
Dihedral angles (°)	17.12	17.22
Ramachandran (centroid)		
Disallowed residues (%)	1.6	0.5
Allowed residues (%)	1.3	1.9
Favored residues (%)	97.0	97.6
Ramachandran (per structure)		
Disallowed residues (%)	2.5	2.6
Allowed residues (%)	4.4	5.1
Favored residues (%)	93.1	92.3

occupancies re-adjusted. Alternative conformations will be sampled automatically during the simulation.

Both 3TPJ and 3TPP datasets were collected at 100 K and exhibit the $C222_1$ space group, also both have similar unit cell parameters and resolution ranges (Table 1). When possible it is advantageous to use analogous datasets such as these as it allows for more direct structural comparison. Care should be taken if, for example, datasets have different space groups or cell parameters as differences in crystal packing may influence atomic fluctuations and must be considered when comparing the resultant ensembles.

Running ensemble refinement

`phenix.ensemble_refinement` can be run from the command line or from the *PHENIX* GUI. There are several parameters that should be defined and/or optimized by the user. Parameter optimization is covered in the next section; here we examine the parameters defined for the BACE1 simulations that remain consist across all runs.

TLS groups used in the TLS fitting procedure should be defined by the user and do not correlate with those used in standard refinement (i.e. it is not recommended to use `phenix.find_tls` for ensemble refinement). Each defined TLS group will fit the rigid-body motion of that group such that they are not required to be sampled during the simulation. This reduces the conformational space that needs to be sampled in the simulation and simplifies the output ensemble. In practice it is best to use complete chains or domains for the defined groups (e.g. tls_group_selections = 'chain a') unless there is good reason to do otherwise. In the case of BACE1 it has a defined 2-lobe architecture so each lobe is assigned to a separate TLS group. The N terminal lobe occurs between residues -4 to 180 and the C-terminal lobe from residue 181 to 386. Both the apo and holo structures contain a number of non-solvent ligands and these must be assigned to the correct TLS group. Visual inspection in a molecular graphics program, e.g. PyMOL (Schrödinger 2010) is used to determine which lobe each ligand is closest to. For the apo dataset the following TLS selections were used to define two groups: tls_group_selections = 'resseq -4:180 or resseq 394 or resseq 395 or resseq 398:400 or resseq 402:406', tls_group_selections = 'resseq 181:386 or resseq 396:397 or resseq 401 or resseq 999'. The selection strings can be defined using the standard nomenclature. It is not necessary to include water atoms as these are automatically assigned to the nearest TLS groups during the simulation.

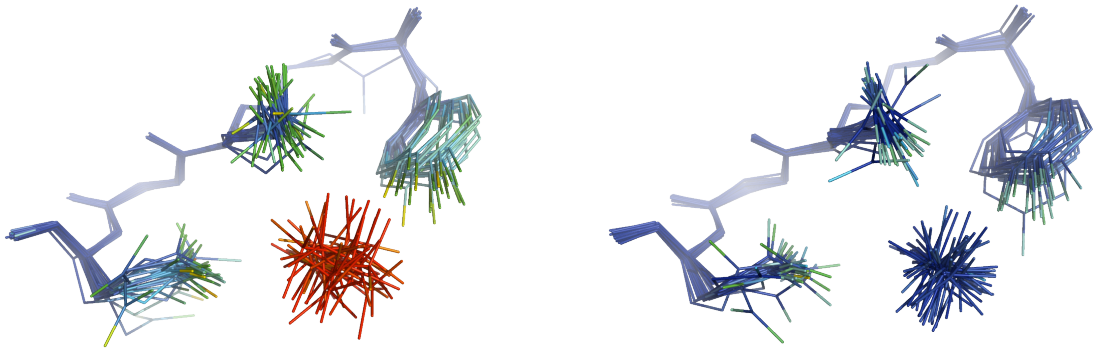


Figure 1. Effect of atom occupancy on kinetic energy. Two simulations of holo BACE1 were performed with difference occupancies for the SO_4^{2-} hetgroup, 1.0 on the left and 0.4 on the right. Atoms are colored by kinetic energy ranging from 0.0 to 2.0 kcal/mol (blue to red).

For weakly bound non-specific ligands it may be appropriate to add harmonic restraints to prevent stochastic displacement during the simulation. For apo BACE1 harmonic restraints we defined as: harmonic_restraints.selections = 'resname so4 or resname cl or resname ure'. This restrains all selected atoms to the x,y,z co-ordinate positions given in the input structure. To allow for any uncertainty in position the harmonic restraints have a 'flat bottom' so deviations less than a given slack value (distance in Å) will generate no force (e.g. harmonic_restraints.slack = 2.0). For the holo structure it is not necessary to add harmonic restraints for the high-affinity, well-ordered, inhibitor.

At present non-solvent ligands with non-unity occupancies cannot be replaced/reinserted in the simulation *a la* the explicit solvent atoms (see Burnley et al. 2012 for details). Atom occupancy is defined from the input structure and if applicable this should be refined, when possible, using single-structure methods prior to ensemble refinement. Errors in occupancy can be detected during ensemble refinement: an atom with too high occupancy will sample a greater amplitude and frequency of disorder than neighboring atoms and exhibit excessive kinetic energy. Atomic kinetic energies (kcal/mol x 10^{-1}) can be outputted in the final PDB ensemble in the occupancy column by setting the parameter: `output_running_kinetic_energy_in_occupancy_column=True`. Figure 1 shows a sulphate group in holo BACE1, where two simulations were performed with occupancies of 1.0 and 0.4 (0.4 was calculated using combined occupancy and ADP refinement in `phenix.refine`). An occupancy of 1.0 results in the sulphate group sampling excessively high kinetic energies whereas at the refined occupancy of 0.4 these atoms exhibit kinetic energies similar to that of the neighboring environment. For datasets where stable refinement of occupancies is not possible, e.g. at low resolution, we recommend running multiple simulations with different occupancy values (e.g. 0.3,0.4,0.5…) .

Finally the water picking parameters can be adjusted depending on the dataset resolution. As standard the water positions are assessed using cutoffs of 3.0 and 1.0 σ for mFo-DFm and 2mFo-DFm maps respectively. For structures with resolution better than ~1.7 Å, better results maybe obtained by lowering the mFo-DFm threshold (e.g. `ensemble_ordered_solvent.primary_map_cutoff = 2.5`). For both the apo and holo BACE1 structures better quality mFo-DFm difference maps and lower R_{free} values where found when using a cutoff of 2.5 σ was used (values of 2.5 and 3.0 σ were tested).

The simulation can be started from the GUI or from the command line using `phenix.ensemble_refinment`. The simulation length depends on several factors including: the number of atoms in the ASU, dataset resolution and hardware configuration. In our experience runtime is typically between several hours and a few days. Using a 1.9 GHz processor for BACE1 the CPU time was 3.6 days. While the simulations are running no user input is required and, due to the length of CPU time, it is highly recommended to test alternative parameters in parallel.

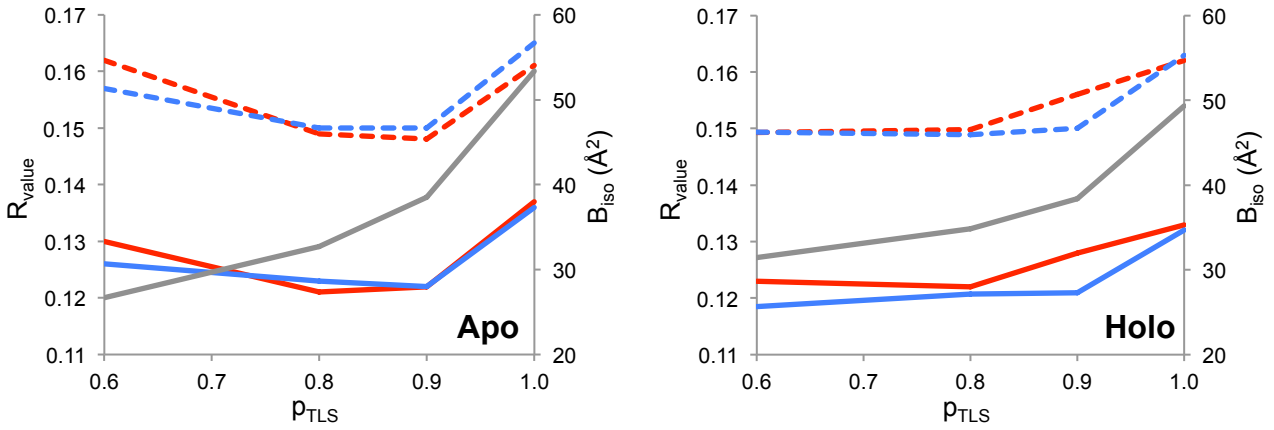


Figure 2. Ensemble refinement parameter optimization for apo and holo BACE1 datasets. An array of p_{TLS} values (0.6 – 1.0) and T_{bath} (2 and 10 K shown in red and blue respectively) was performed and the resulting R_{work} and R_{free} values are shown in solid and dashed lines respectively. The isotropic equivalent of the fitted TLS B-factors are shown in grey, averaged over all non-solvent, non-hydrogen atoms.

Optimizing simulation parameters

There are two main empirical parameters (p_{TLS}, T_{bath}) that affect the running and the outcome of ensemble refinement, and it is critical to optimize both on a dataset by dataset basis.

p_{TLS} (e.g. `ptls=0.9`) defines the fraction of atoms included in the TLS fitting procedure (for details see Burnley et al. 2012). The optimum value for this parameter cannot be determined *a priori*. Setting up a parallel array of simulations with different values (e.g. 1.0, 0.9, 0.8, 0.6) is recommended. The TLS fitting process requires that the number of non-hydrogen, non-solvent atoms per TLS group multiplied by the p_{TLS} fraction to be greater than 63 (enforcing a data:parameter ratio greater than 3 for TLS fitting). If this is not the case, p_{TLS} will be automatically increased. In the case that a TLS group has less than 63 non-solvent, non-hydrogen atoms then an isotropic model, based on the Wilson B-factor, is automatically applied.

T_{bath} (e.g. `wxray_coupled_tbath_offset=5`) set the temperature bath offset (K) and, counter intuitively, T_{bath} controls the X-ray weight. The X-ray weight is modulated in situ such that the simulation runs at the target temperature (e.g. `cartesian_dynamics.temperature=300`). The simulation uses velocity scaling to maintain the target temperature. This is coupled to a temperature thermostat which is set as: thermostat = cartesian_dynamics.temperature - wxray_coupled_tbath_offset. The non-conservative time-averaged X-ray force generates heat (Gros, van Gunsteren, and Hol 1990), thus larger offsets increase the X-ray weight. The default value is 5 K and, for example, values of 2 and 10 K may also be tested.

For both the apo and holo BACE1 datasets a parallel array of eight separate simulations was performed with a grid of p_{TLS} (1.0, 0.9, 0.8 and 0.6) and T_{bath} (2 and 10 K), as shown in Figure 2. A shallow optimum in R_{values} is observed and with optimum values of 0.9 and 2 K for p_{TLS} and T_{bath} respectively for the apo dataset and 0.6 and 2 K for the holo dataset.

Ensemble refinement is based on MD simulations that are stochastic by nature. Therefore to aid analysis it is useful to run a number of random seed repeats (e.g. `random_seed=26556257`). This seed is used in the initial velocity assignment and the alternative trajectories that arise can be used as a guide to the reproducibility of the simulation as a whole and also the individual atom fluctuations (see Burnley et al., 2012, Figures 4, 5 & 10). In total five random number seed repeats where run for both BACE1 datasets producing consistent global R_{values}, see Table 2, the best of which (as judged by R_{work} *and* R_{free} *values*) were then selected for the subsequent structural analysis and the refinement statistics for these are reported in Table 1. In this case we found that the R_{values} improve for both the apo and holo datasets after ensemble refinement.

Table 2. R_{values} and number of structures in output PDB ensemble model for five random number seed repeats of `phenix.ensemble_refinement` for apo and holo BACE1.

	Apo			Holo		
Repeat	R_{work} (%)	R_{free} (%)	*Structures*	R_{work} (%)	R_{free} (%)	*Structures*
1	12.1	14.7	134	12.1	14.6	107
2	12.1	14.7	178	11.9	14.7	134
3	12.2	14.8	107	12.1	14.8	134
4	12.1	14.8	134	12.3	14.9	89
5	12.2	14.8	134	12.4	15.0	134
Mean	*12.1*	*14.8*	*137*	*12.2*	*14.8*	*120*

Two sets of geometry statistics are reported in Table 1 for the ensemble structures: 'centroid' and 'per structure' (see Burnley et al. 2012 for complete definitions). This is because the ensemble as a whole is fitted to the data. Therefore the structures represent a distribution that reflects fluctuations around ideal geometric values, i.e. each individual restraint in each individual structure does not necessarily have to fit the ideal geometric value and a degree of variation is expected. However, considering the model as a whole ensemble, the restraints deviations can be expected to oscillate around an ideal value therefore 'centroid' statistics are provided. The geometric deviations are also calculated for each structure separately and then averaged across the ensemble giving the 'per structure' values. While large deviations can be observed when evaluated per structure, the centroids of the distribution show fluctuations around the correct value for the BACE1 datasets, see Table 1.

Structural analysis

By default phenix.ensemble_refinement outputs the following files: log (.log), map coefficients (.mtz), geometry of input structure (.geo) and the ensemble structure (.pdb.gz). As the ensemble PDB structure can contain hundreds of models it is zipped to reduce size and should be unzipped (using gunzip for instance) before use. During the simulation thousands of sets of coordinates are sampled, so to simplify the final model the minimum number of models is found that reproduces the R_{free} value of the whole trajectory within a 0.25 % tolerance. To generate these reduced ensembles every 1st,2nd,3rd...nth model of the total number of coordinates sets stored during the acquisition phase of the simulation are selected and the R_{values} of these concerted ensembles are recalculated. The relationship between the number of structures and the R_{values} is shown for apo and holo BACE1 in Figure 3. There is some fluctuation in the recalculations and as such the number of models in the final PDB can vary as is shown in Table 2 for the five number seed repeats.

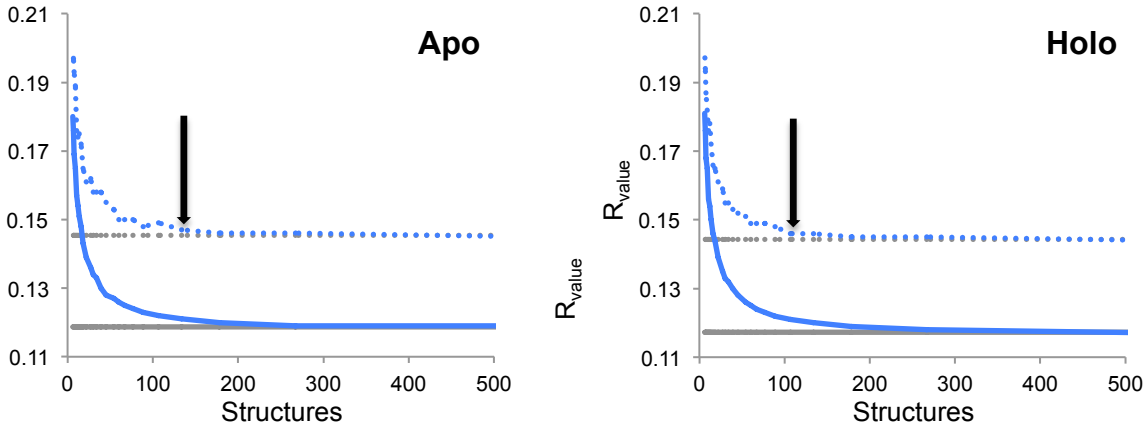


Figure 3. Determination of the number of models in the output ensemble PDB. For the best apo and holo simulation the charts shows the recalculation of R_{work} and R_{free} (solid and dashed line respectively) with different numbers of structures in the ensemble models (blue), the minimum number of models to reproduce complete trajectory R_{values} (shown in grey) within a 0.25% tolerance is selected for output (highlighted with arrow). In this case, apo and holo BACE1 contained 134 and 107 models.

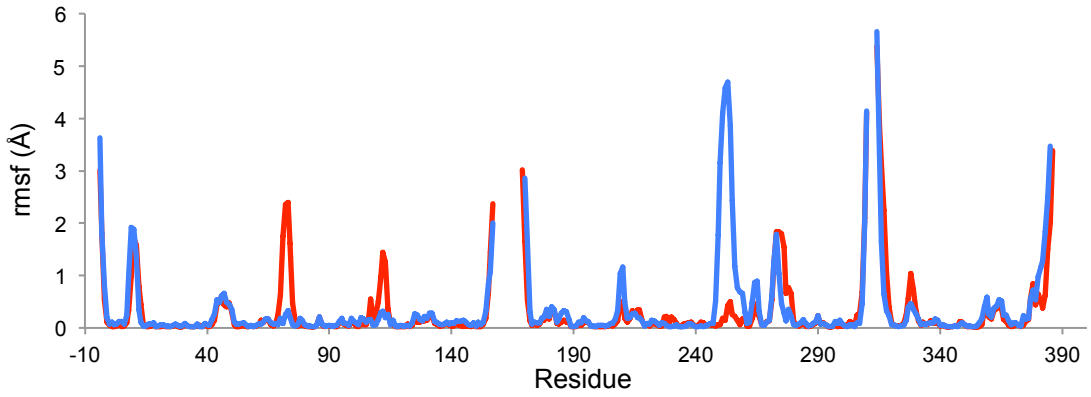


Figure 4. Backbone rmsf of apo (red) and holo (blue) BACE1 ensembles. Human aspartic protease residue numbering is used. Values were calculated used ens_tools.py script for PyMOL.

Visual analysis of the structures and maps can be done using standard programs like PyMOL and COOT (Emsley et al. 2010). When using PyMOL the models can be viewed sequentially as a movie using the play button or superimposed together via the show all states option (via Movie header or the command 'set all_states, 1'). If the user wants to view atomistic atom B-factors or kinetic energies (see above) for each individual model in the ensemble the additional option 'discrete=1' should be specified with the load command. Viewing in COOT may be improved by reducing the number of models in the ensemble to between 5 to 50 and this can be done using command line program `phenix.ensemble_reducer`, which allows the further truncation of the ensemble. Once the ensemble models have been produced no manual model building is possible. If, on inspection, an error has occurred, for instance a ligand has become displaced from its binding site than the simulation parameters need to altered, e.g. harmonic restraints maybe added or adjusted, and ensemble refinement should be re-run.

The structural fluctuation in the ensembles can be quantified by calculating the rmsf and this is shown per residue for apo and holo BACE1 in Figure 4. Many programs can process ensemble structures to calculate rmsf and other metrics, and we have provided a script for PyMOL that can also do this. 'ens_tool.py' is available from $PHENIX/cctbx_project/mmtbx/refinement/ensemble_refinement/ and the command 'ens_rmsf, selection' will return the rmsf values for selected atoms. Figure 4 shows the highest deviations for both apo and holo BACE1 datasets occur in the N and C terminal regions as well loop 153-173 and loop 308-319. These regions contain missing residues so it is unsurprising to observed high proximal mobility located here. A high rmsf value is returned for tyr71 in the apo structure. This residue is located in the 'flap' region, a hairpin loop found in the N-terminal lobe and spanning residues 67-75. Other structures deposited in the PDB show that this loop can adopt multiple conformations. Figure 5 shows the flap region in the apo and holo forms of BACE1. When the active site is empty the flap region is mobile, however, upon addition of the inhibitor BSIIV the flap folds over the binding pocket such that tyr71 can make weak hydrophobic contacts with the inhibitor and thus trapping the flap in a single conformation. Figure 6 shows the remainder of the binding pocket and here a general tightening of the residues surrounding the ligand is observed upon binding. In particular, the conserved catalytic dyad formed by asp32 and asp228 becomes more ordered in presence of the inhibitor, see Figure 7. Interesting, the holo structure is highly mobile between residues 248-256, a surface turn is located in the C-terminal lobe, whereas the apo structure is more rigid. This pattern is repeated for all five random number seed repeats for both apo and holo datasets, and could have functional role for instance in offsetting the loss of entropy upon binding and/or as a method of signaling.

In conclusion, `phenix.ensemble_refinement` is suitable for standard protein datasets and does not require specialist experimental techniques or hardware. For both apo and holo BACE1 it provides multi-structure models that fit well to the diffraction data. The TLS fitting procedure absorbs global motions and this gives the

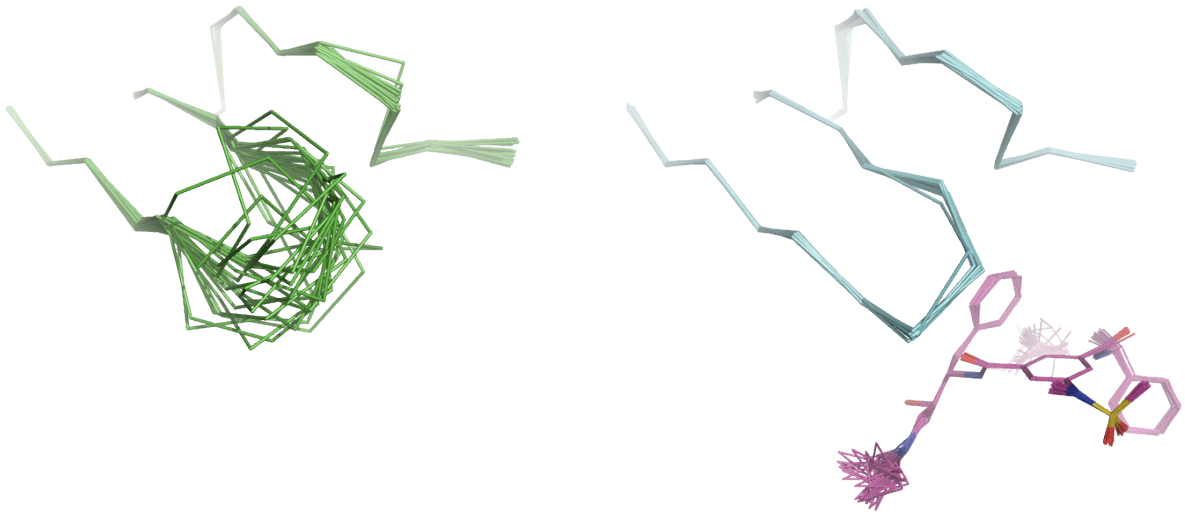


Figure 5. Comparison of the 'flap' region in the ensemble structures of apo (left, green ribbon) and holo (right, blue ribbon) BACE1. The hairpin loop (residues 67-75) becomes significantly more ordered in the presence of the inhibitor. 25 structures are shown for each ensemble.

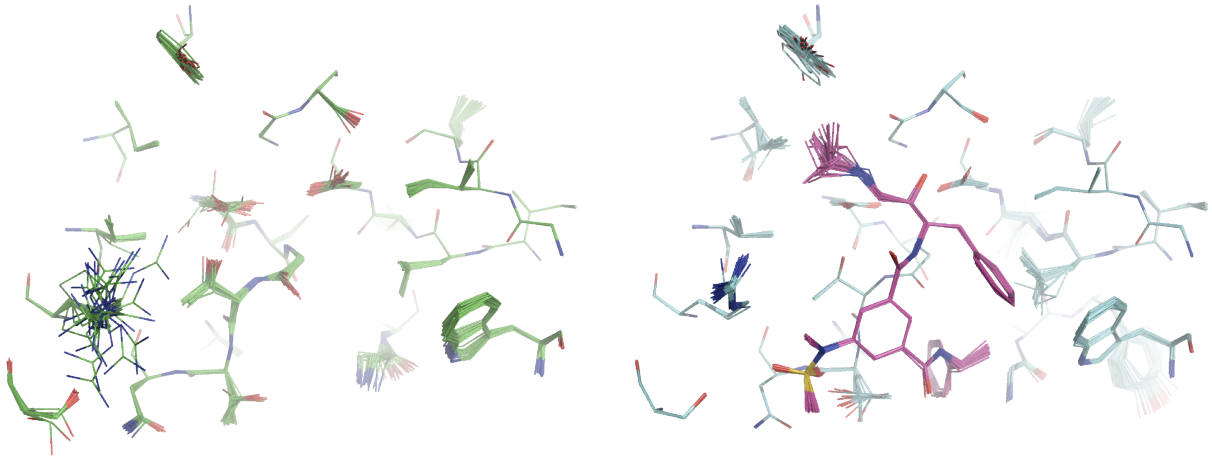


Figure 6. The binding pocket of apo (left, green carbon atoms) and holo (right, blue carbon atoms) BACE1 rigidifies in the presence of the inhibitor (right, pink carbon atoms). 25 structures are shown for each ensemble.

opportunity to visualize and quantify the intramolecular dynamics of the molecule. As such, it provides a tool to probe structure, function and dynamics from diffraction data.

Caveat Emptor

There are some important caveats that should be noted to prevent accidental misinterpretation of the models produced using ensemble refinement. 1) Ensemble refinement is recommended for use with datasets with a resolution of 3 Å or better. 2) The number of structures in the output ensemble is variable and should not be over interpreted. 3) The ensembles are a Boltzmann-weighted population and therefore contain high-energy conformations. 4) Dynamic rates cannot be calculated from the simulations. The X-ray force is non-conservative and accelerates atomic sampling. This allows movements that occur at timescales several orders of magnitude greater than a 100ps simulation, e.g. side chain sampling or loop motions, to be modeled in the (relatively) short simulation time but therefore the rate of conformation exchange cannot be derived. 5) No information regarding correlated motion is provided by the time- and spatially-averaged X-ray restraints although local correlations may be observed via short-range steric interactions.

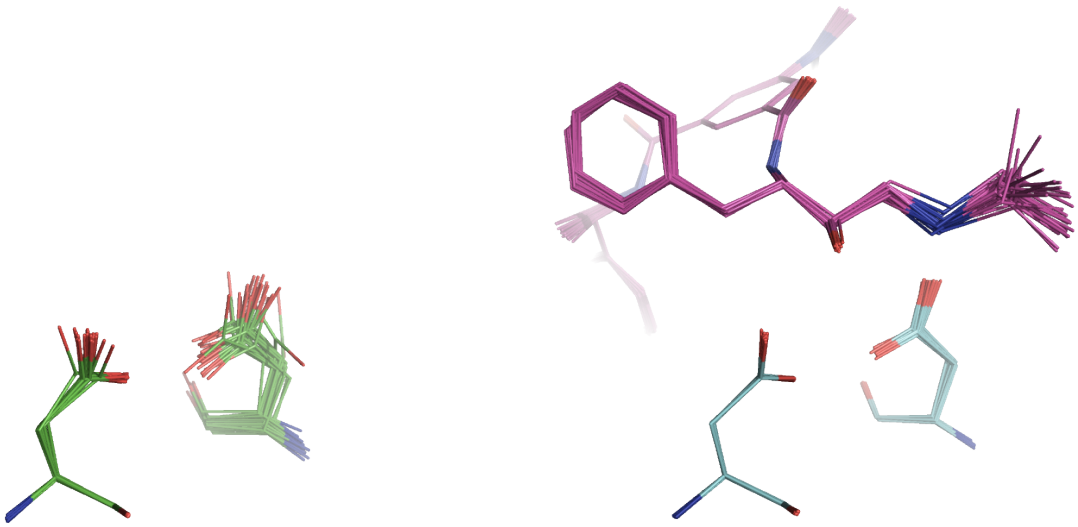


Figure 7. Both asp32 and asp228 residues that form the conserved catalytic dyad exhibit higher mobility in the absence of the inhibitor. Apo BACE1 ensemble is shown left (green carbon atoms) and holo ensemble is shown right (blue and pink for protein and inhibitor carbon atoms respectively). 25 structures are shown for each ensemble.

References

Adams, P D, Pavel V Afonine, Gábor Bunkóczi, Vincent B Chen, Ian W Davis, Nathaniel Echols, Jeffrey J Headd, et al. 2010. "PHENIX: a Comprehensive Python-based System for Macromolecular Structure Solution." *Acta Crystallographica Section D* 66 (2) (February): 213–221. doi:10.1107/S0907444909052925.

Berman, Helen M., John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. 2000. "The Protein Data Bank." *Nucleic Acids Research* 28 (1): 235–242. doi:10.1093/nar/28.1.235.

Burnley, B. T., P. V. Afonine, P. D. Adams, and P. Gros. 2012. "Modelling Dynamics in Protein Crystal Structures by Ensemble Refinement." *eLife* 1 (0) (January 1): e00311–e00311. doi:10.7554/eLife.00311.

Emsley, P., B. Lohkamp, W. G. Scott, and K. Cowtan. 2010. "Features and Development of *Coot*." *Acta Crystallographica Section D Biological Crystallography* 66 (4) (March 24): 486–501. doi:10.1107/S0907444910007493.

Gros, P, W F van Gunsteren, and W G Hol. 1990. "Inclusion of Thermal Motion in Crystallographic Structures by Restrained Molecular Dynamics." *Science (New York, N.Y.)* 249 (4973) (September 7): 1149–1152.

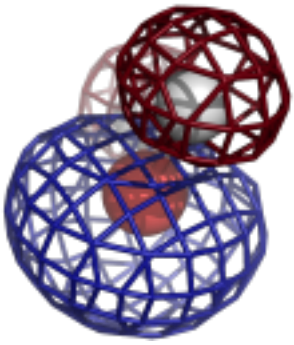
Joosten, R. P., T. A. H. te Beek, E. Krieger, M. L. Hekkelman, R. W. W. Hooft, R. Schneider, C. Sander, and G. Vriend. 2010. "A Series of PDB Related Databases for Everyday Needs." *Nucleic Acids Research* 39 (Database) (November): D411–D419. doi:10.1093/nar/gkq1105.

Schrödinger, LLC. 2010. "The PyMOL Molecular Graphics System, Version 1.3r1."

Sinha, S. 1999. "Cellular Mechanisms of Beta -amyloid Production and Secretion." *Proceedings of the National Academy of Sciences* 96 (20) (September 28): 11049–11053. doi:10.1073/pnas.96.20.11049.

Xu, Yechun, Min-jun Li, Harry Greenblatt, Wuyan Chen, Aviv Paz, Orly Dym, Yoav Peleg, et al. 2011. "Flexibility of the Flap in the Active Site of BACE1 as Revealed by Crystal Structures and Molecular Dynamics Simulations." *Acta Crystallographica Section D Biological Crystallography* 68 (1) (December 9): 13–25.

VOLUME FIVE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXIV

BIOMT, XFEL, IMAGECIF, BIG DATA, DIVCON

Table of Contents

- PHENIX News 1
- Crystallographic meetings 3
- Expert Advice
  - Fitting tips #7 – Getting the Pucker Right in RNA Structures 4
- Short Communications
  - Phenix tools for interpretation of BIOMT and MTRIX records of PDB files 8
  - Coping with BIG DATA image formats: integration of CBF, NeXus and HDF5 12
- Articles
  - XFEL Detectors and ImageCIF 19
  - Quantum Mechanics-based Refinement in Phenix/DivCon 26

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

phenix.composite_omit_map

Nat Echols & Pavel Afonine

A new program for omit map calculation (Bhat, 1988) is now available. The implementation features two alternatives for de-biasing the phases:

- An iterative procedure that generates an F(model) map, and repeatedly computes a composite omit map which then provides starting phases for the next cycle. This is done over boxes encompassing the entire unit cell. Because no refinement is required, the procedure is extremely fast and is the default mode of operation.

- A more conventional refinement-based procedure, similar to the composite omit map implementation in CNS (Hodel et al. 1992, Brunger et al. 1997). Simulated annealing is available as an option; the omitted atoms will be harmonically restrained to prevent the structure from collapsing into the omit regions. This method is significantly slower but can be parallelized over multi-core computers or supported queuing systems.

Although the program is primarily intended for composite omit maps covering the entire unit cell, the refinement procedure can also be used to generate a simple omit map, for instance showing de-biased density for a ligand. This functionality is essentially identical to running phenix.refine with custom parameters, but via a simplified interface.

These features mostly supersede the equivalent modes in the AutoBuild wizard,

with the exception of the "iterative build" omit maps (Terwilliger et al. 2009).

References

Bhat, T. N. (1988) J. Appl. Cryst., 21, 279-281.

Brunger, A. T., Adams, P. D., Clore, G. M., DeLano, W. L., Gros, P., Grosse-Kunstleve, R. W., Jiang, J.-S., Kuszewski, J., Nilges, M., & Pannu, N. S. (1998). Acta Crystallographica D. 54, 905–921.

Hodel, A., Kim, S.-H., & Brunger, A. T. (1992). Acta Crystallographica A. 48, 851–858.

Terwilliger, T. C., Grosse-Kunstleve, R. W., Afonine, P. V., Moriarty, N. W., Adams, P. D., Read R. J., Zwart P. H., & Hung L.-W. (2009). Acta Crystallographica D. 64, 515–524.

New features

Optimizing phenix.autosol and phenix.hyss for weak SAD data

Tom Terwilliger, Randy Read, Gábor Bunkóczi and Nat Echols

You might notice some big differences in HySS and AutoSol in the new 2014 versions of Phenix (starting with dev-1610 and the soon-to-be-released Phenix-1.8.5). The big changes are parallelization, the use of Phaser completion in HySS, iterative Phaser completion using maps and models in AutoSol, and on-the-fly optimization of parameters in AutoSol for cases with weak anomalous signal. Now HySS and AutoSol can solve structures with very weak SAD phase information that previous versions could not solve.

A new powerful and parallel Hybrid Substructure Search (HySS)

The first thing you might notice in HySS is that it can use as many processors as you have on your computer. This can make for a really quick direct methods search for your anomalously-scattering substructure.

You might notice next that HySS now automatically tries Phaser completion to find a solution if the direct methods approach does not give a clear solution right away. Phaser completion uses the likelihood function to create an LLG map that is used to find additional sites. This is really great because Phaser completion in HySS can be much more powerful than direct methods in HySS. Phaser completion takes a lot longer than direct methods completion but it is now quite feasible, particularly if you have several processors on your computer.

The next thing you might notice in HySS is that it automatically tries several resolution cutoffs for the searches if the first try does not give a convincing solution. Also HySS will start out with a few Patterson seeds and then try more if that doesn't give a clear solution.

HySS now considers a solution convincing if it finds the same solution several times, starting with different initial Patterson peaks as seeds. The more sites in the solution, the fewer duplicates need to be found to have a convincing solution.

Putting all these together, the new HySS is much faster than the old HySS and it can solve substructures that the old HySS could not touch.

An AutoSol optimized for weak SAD data

The new AutoSol is specifically engineered to be able to solve structures at low or high resolution with a very weak anomalous signal.

One feature you may notice right away is that the new AutoSol will try to optimize several choices on the fly. AutoSol will use the Bayesian estimates of map quality and the R-value in density modification to decide which choices lead to the best phasing. AutoSol will try using sharpened data for substructure identification as well as unscaled data as input to AutoSol and pick the one leading to the best map. AutoSol will also try several smoothing radii for identification of the solvent boundary and pick the one that gives the best density modification R-value.

You'll also notice that AutoSol uses the new parallel HySS and that it can find substructures with SAD data that are very weak or that only have signal to low resolution. You can use any number of processors on your machine in the HySS step (so far the parallelization is only for HySS, not

the other steps in AutoSol, but those are planned as well).

The biggest change in AutoSol is that it now uses iterative Phaser LLG completion to improve the anomalously-scattering substructure for SAD phasing. The key idea is to use the density-modified map (and later, the model built by AutoSol) to iterate the identification of the substructure. This feature is amazingly powerful in cases where only some of the sites can be identified at the start by HySS and by initial Phaser completion. Phaser LLG completion is more powerful if an estimate of part of the structure (from the map or from a model) is available.

The new AutoSol may take a little longer than the old one due to the heavy-atom iteration, but you may find that it gives a much improved map and model. Give it a try!

Crystallographic meetings and workshops

Biophysical Society 58th Annual Meeting, February 15-19, 2014

An IYCr2014 symposium entitled "Celebrating 100 Years of Crystallography: X-Rays Are Photons Too" will be of interest to all crystallographers. This year the annual Biophysics101 session will be on tips for biophysicists who want to use the crystallographic technique. Check the website for details as the date approaches.

Keystone Symposium, "Frontiers in Structural Biology," March 30-April 4, 2014

This meeting will be held in Snowbird, Utah.

44th Mid-Atlantic Macromolecular Crystallography Meeting and 11th Annual SER-CAT Symposium, April 23-26, 2014

Keynote speakers include Bi-Cheng Wang and Wayne Hendrickson.

American Crystallographic Association (ACA) Annual Meeting, May 24-28, 2014

This year the meeting is in Albuquerque, New Mexico.

WK.01 Joint Neutron and X-ray Structure Refinement using Joint Refine in PHENIX

A workshop in association with the ACA Annual Meeting. Go to the ACA website http://www.amercrystalassn.org/2014-wk.01 for details.

Macromolecular Crystallography School - MCS2014, May 26-31, 2014

A school for students and researchers. Go to http://www.xtal.iqfr.csic.es/MCS2014/index.html for details.

2014 (Pacific) Northwest Crystallography Workshop, June 20-22, 2014

The conference organizer is P. Andrew Karplus, Department of Biochemistry and Biophysics, School of Life Sciences, Oregon State University, Corvallis, OR 97331, NWCW2014@science.oregonstate.edu.

Andrew is pleased to announce that the 2014 (Pacific) Northwest Crystallography Workshop will be held this summer at the Linus Pauling Science Center, Oregon State University. To avoid conflicts with other meetings we are restoring the meeting to its traditional month of June. In particular, it will be held Friday evening through Sunday noon the weekend of the June 20th through the 22nd.

Registration and abstract submission will begin on February 1st. Detailed information can be found at http://oregonstate.edu/conferences/event/NWCW2014/. Talks will be selected from the abstracts submitted for posters. To encourage an informal atmosphere, we will give preference to students and post-docs.

Corvallis is located between the mountains and the sea with each in easy driving distance. It is surrounded by wine country and other recreational opportunities. It is just off the I-5 corridor and can be accessed from either the Portland or Eugene airports.

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Seminar, Bates College, Lewiston, ME, July 26-27, 2014

A seminar series preceding the GRC meeting.

Gordon Research Conference on Diffraction Methods in Structural Biology: Towards Integrative Structural Biology, Gordon Research Conference, Bates College, Lewiston, ME, July 27-August 1, 2014

A very interesting and important meeting for protein crystallographers.

23rd International Union of Crystallography (IUCr) Congress, August 5-12, 2014

This year the congress will be in Montreal, Canada.

5th Murnau conference on Structural Biology – Focus Topic: Signal Transduction Sept 10-13, 2014

Location: Murnau am Staffelsee, Germany www.murnauconference.de/2014/index.html

Expert advice

Fitting Tip #7 – Getting the Pucker Right in RNA Structures

Swati Jain, Gary Kapral, David Richardson and Jane Richardson, *Duke University*

Building good RNA backbone models into electron density is quite a challenge, but there is now a very effective trick to get one of the difficult parts right – the ribose ring pucker. Ribose rings in RNA are known, from small-molecule or high-resolution larger RNA structures, to adopt just two main conformations: C3'-endo and C2'-endo, each with a tightly defined range for the δ dihedral angle (figure 1). However, to distinguish between these two conformations from the electron density alone at more typical 2.5-3.5 Å resolution is virtually impossible. As a result, most RNA structures contain pucker errors, usually accompanied by steric clashes and geometry outliers. Therefore, it is very desirable to model the pucker correctly as early in the structure building process as feasible.

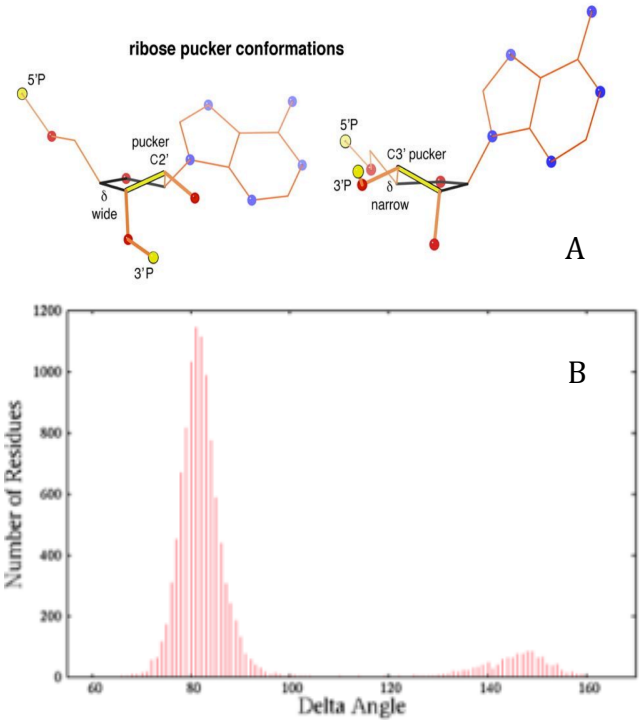


Figure 1: (A) The two main ribose pucker conformations found in RNA structures, C2'-endo (left) and C3'-endo (right). (B) Occurrence frequency plotted against the δ dihedral for the RNA11 dataset after applying clash and B-factor < 60 filters on the RNA backbone. The two occupied δ ranges are: C3'-endo: 60°-105°, C2'-endo: 125°-165°.

The Pperp Test

The trick to identify the correct ribose pucker is the 3' Pperp test, which has the great advantage of working from the two most clearly seen RNA structural features in the electron density: the phosphate and the base. This test involves judging the relative position of the 3'P (the phosphate P atom in the 3' direction) to the plane of the base, or better, to the vector extension of the glycosidic bond (C1'-N1/N9). This test is similar to the zp distance used in the program 3DNA (Lu 2003) to distinguish between A form and B form DNA helices, but is more general and can be applied to any RNA residue, even in loops, bulges, or junctions. When the ribose ring has a C3'-endo pucker, the perpendicular distance

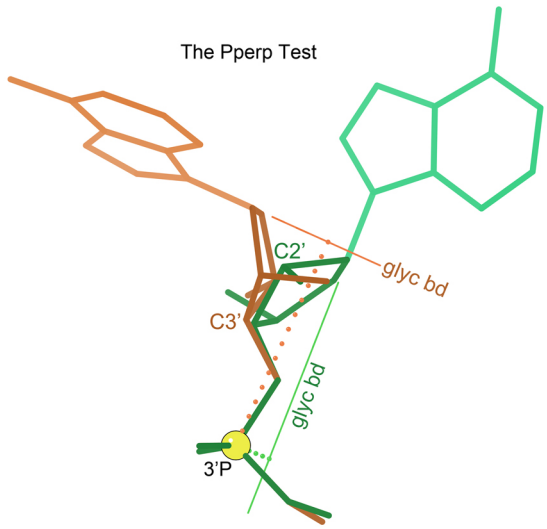


Figure 2: Judging the length of the perpendicular (dotted lines) dropped from the 3'P to the vector extension of the glycosidic bond (C1'-N9/N1). That distance is long for C3'-endo pucker (orange) and short for C2'-endo pucker (green): 4.6Å vs 1.1Å for this pair of examples.

is longer than when the ribose ring has a C2'-endo pucker (figure 2). The specific protocol to apply the Pperp test is to drop a perpendicular from the 3'P to the vector extension of the glycosidic bond (C1'-N1/N9), and measure whether its length is > 2.9Å (for C3'-endo pucker) or < 2.9Å (for C2'-endo pucker). However, the long vs. short perpendicular can be judged quite well by eye. This test is based on the strong empirical correlation between the length of the perpendicular and the pucker of the ribose ring, which becomes almost entirely clean as additional residue-level quality filters are applied (figure 3). This allows one to use a distance of 2.9Å as the cutoff distance to distinguish between the two puckers and then make a correction if the indicated pucker does not match the modeled δ value. In addition, the values of some bond angles and of the other backbone dihedral angles (besides δ) are pucker-specific, and the ability to apply the Pperp diagnosis in Phenix and then use pucker-specific values in the target function significantly improves RNA refinement behavior as well as validation statistics for the final structure.

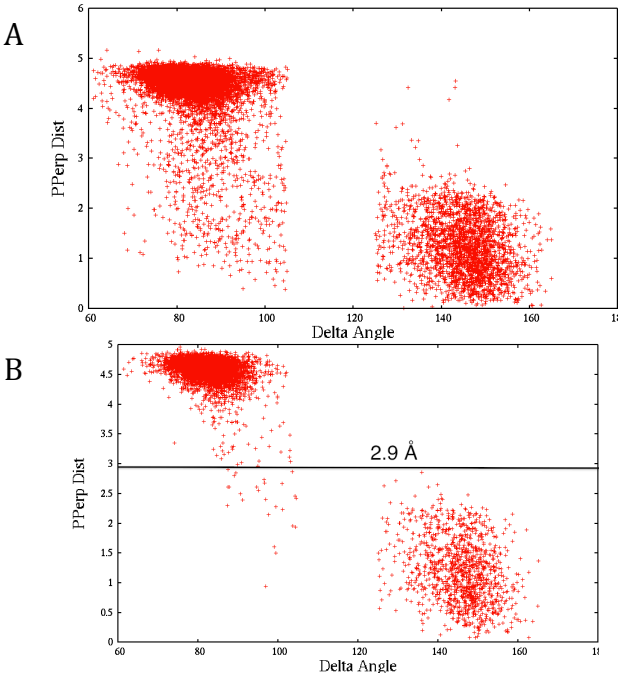


Figure 3: Pperp distance vs δ angle, plotted in the valid δ ranges for C3'-endo and C2'-endo pucker. Data is from the RNA11 dataset: (A) no residue-level filters; (B) with clash, B-factor < 60, and ε filters.

Correcting Pucker Outliers

If a residue fails the Pperp test, i.e. the Pperp distance and the δ angle indicate different puckers, the residue is flagged as a pucker outlier. Such problems can be corrected using several available tools. RNA Rotator in KiNG (Chen 2009) allows the user to change the backbone dihedral angles manually and choose a different backbone conformation from a list of known conformers. RNABC (Wang 2008) and the RCrane plugin in Coot (Keating 2012) rebuild the ribose atoms keeping the base and phosphate fixed. The most powerful option, however, is a new automated tool called ERRASER (Chou 2012a) that locally rebuilds the full RNA backbone, using Phenix refinement and a step-wise assembly procedure in Rosetta that accounts for fit to the electron density in its scoring function. It is scriptable in Phenix if Rosetta is also installed (Chou 2012b).

Common Mistakes to Avoid

The most common problem with RNA sugar pucker is to model a C2'-endo pucker as a C3'-endo, presumably because more than 80% of

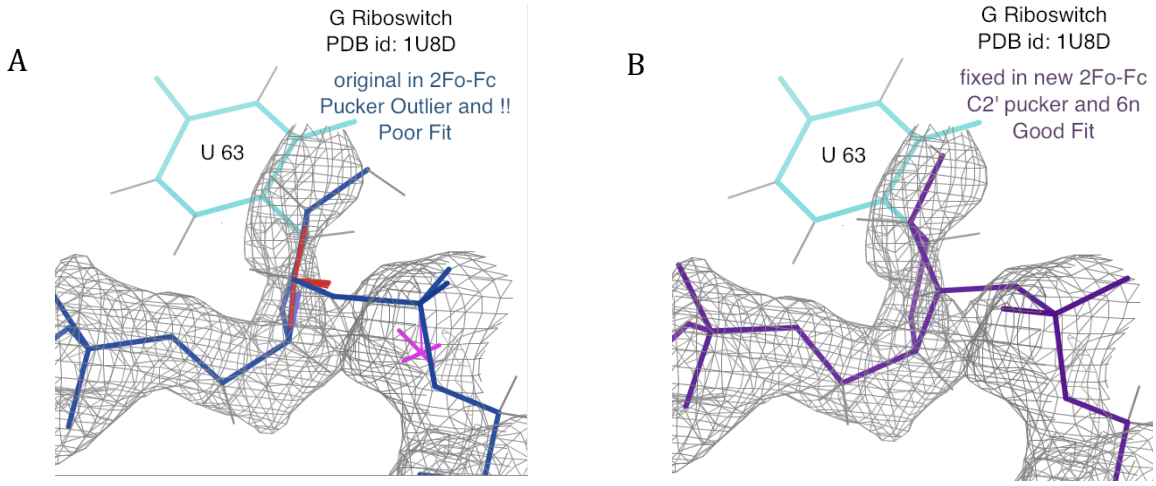


Figure 4: Residue U 63 of the G Riboswitch (PDB id: 1U8D) in the electron density. (A) Original model, with outlier flags for pucker (magenta cross), backbone conformer (**!!**), and bond angles (red and blue fans). (B) Fixed residue, with C2'-endo pucker, a **6n** backbone conformer, and better fit to the ribose density. (For both, the base and the C5' at lower right are in good density, but lie behind the clipping plane.)

the residues in RNA are C3'-endo pucker, and is thus the default expectation. An example of such an error is in the earliest G riboswitch structure (PDB id: 1U8D; Batey 2004). The resolution is 1.95Å, and it can actually be seen that the modeled pucker for U 63 does not fit the density well (figure 4A) - but pucker correction was difficult back then. Residue 63 is also an outlier for bond angles, ε, and RNA backbone conformer (Richardson 2008). Figure 4B shows our rebuild of the structure using C2'-endo pucker, which also fixes the bond-angle and ε outliers, moves the backbone to a recognized **6n** conformer, and fits the density better. A later dataset at 1.32Å (PDB id: 4FE5) confirms this and other changes.

A less common source of pucker error can result from trying too hard to force corresponding residues in different structures/chains into the same conformation. An example of such a pucker outlier occurs in the structure of a ternary complex of human exo-ribonuclease protein, histone mRNA stem loop, and stem-loop binding protein (PDB id: 4HXH (Tan 2013)). Of the two RNA chains in the asymmetric subunit, chain A is bound to both the proteins but chain D is bound only to the exo-nuclease. Residue 12 in chain A is at the protein-RNA interface of the stem-loop with the stem-loop binding protein, and it has C2'-endo pucker. As a result, residue 12 in chain D was also modeled with C2'-endo pucker. This residue is flagged as an outlier by the Pperp test (figure 5A). We rebuilt it as a C3'-endo pucker, which gets rid of the backbone clash as well as the pucker outlier (figure 5B). The corrected structure is deposited in the PDB as 4L8R.

Conclusion

It is healthy to keep in mind that a pucker outlier could possibly be real and the structure strained, but that will be extremely uncommon and there should be very strong evidence for it. Getting the ribose pucker right is highly beneficial, because the difference between the C3'-endo and C2'-endo flings around the base and the backbone dramatically (figures 1A and 2). If the pucker is fit incorrectly, then refinement is forced into large local distortions. Now the Pperp test has made pucker diagnosis quite easy for manual as well as automated use.

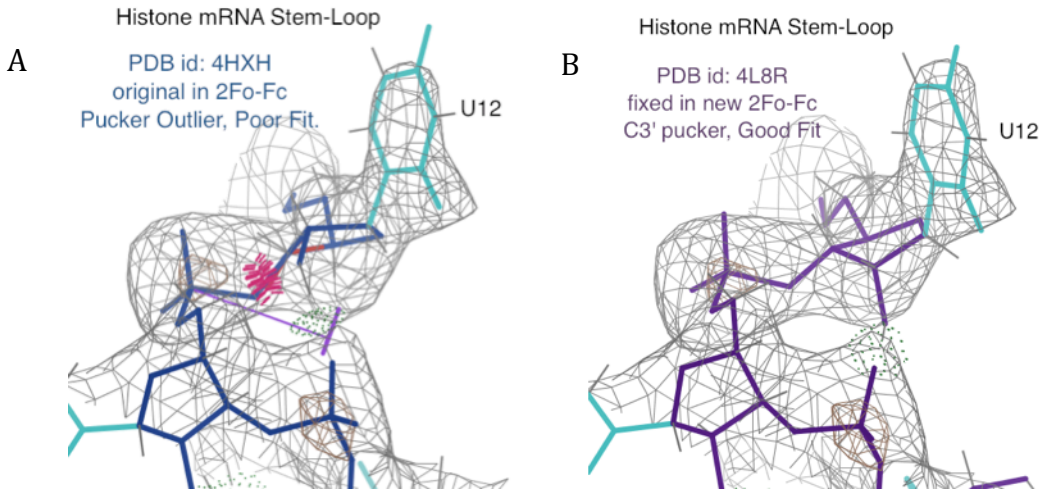


Figure 5: Residue 12 in Chain D of the histone mRNA stem-loop (1HXH) in its electron density. (A) The original residue is an outlier for pucker (purple cross), bond angle (red fan), and steric clashes (pink spikes). (B) The rebuilt model keeps the H-bond, has C3'-endo ribose pucker, no outliers, and fits the density better.

References

Batey RT, Gilbert SD, Montange RK (2004) Structure of a natural guanine-responsive riboswitch complexed with the metabolite hypoxanthine, *Nature* **432**: 411-415

Chen VB, Davis IW, Richardson DC (2009) KiNG (Kinemage, Next Generation): a versatile interactive molecular and scientific visualization program, *Protein Sci* **18**: 2403-2409

Chou F-C, Sripakdeevong P, Dibrov SM, Hermann T, Das R (2012a) Correcting pervasive errors in RNA crystallography through enumerative structure prediction, *Nature Meth* **10**: 74-76

Chou F-C, Richardson J, and Das R (2012b) ERRASER, a powerful new system for correcting RNA models, *Comp Cryst Newsletter* **3**: 35-36

Keating KS, Pyle AM (2012) RCrane: semi-automated RNA model building, *Acta Crystallogr* **D 68**: 985-995

Lu XJ, Olson WK (2003) 3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures, *Nucleic Acids Res* **31**: 5108-5121

Richardson JS, Schneider B, Murray LW, Kapral GJ, Immormino RM, *et al.* (2008) RNA backbone: Consensus all-angle conformers and modular string nomenclature (an RNA Ontology Consortium contribution), *RNA* **14**: 465-481

Tan D, Marzluff WF, Dominski Z, Tong L (2013) Structure of histone mRNA stem-loop, human stem-loop binding protein, and 3'hExo ternary complex, *Science* **339**: 318-321

Wang X, Kapral G, Murray L, Richardson D, Richardson J, Snoeyink J (2008) RNABC: Forward kinematics to reduce all-atom steric clashes in RNA backbone, *J Math Biol* **56**:253-278.

Phenix tools for interpretation of BIOMT and MTRIX records of PDB files

Youval Dar[a], Pavel V. Afonine[a], Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: ydar@lbl.gov

Introduction

Currently, *Phenix* (Adams et al 2010) tools that require the atomic model as input expect the model to contain the atoms corresponding to the entire asymmetric unit. If non-crystallographic symmetry (NCS) is present and all NCS copies are assumed to be identical (strict NCS) then one NCS copy can be considered as independent. Furthermore, the entire contents of the asymmetric unit (ASU) can be generated by applying appropriate NCS transformations to the independent copy. These transformations are rotation matrices and translation vectors. Protein Data Bank (PDB) (Berman et al 2000, Bernstein et al 1977) model files store these transformations in MTRIX records in REMARK 350 section of the file header. As a first step towards handling strict NCS in *Phenix*, particularly as constraints for structure refinement in *phenix.refine* (Afonine et al 2012) we have implemented tools that can read PDB file with single NCS copy and corresponding MTRIX records, generate the entire ASU and output it as expanded PDB file or use corresponding objects internally. The corresponding command line tool is *phenix.pdb.mtrix_reconstruction* and the underlying implementation is located in *multimer_reconstruction.py* file of *iotbx.pdb* module.

When the biologically functional macromolecular assembly (biological unit) is known, the information that provides the generation of the biological unit is recorded in BIOMT records in REMARK 350 section of PDB file header. Similarly to MTRIX, these are rotation matrices and translation vectors that are to be applied to all chains recorded in the PDB file. Note that the biological unit may be a copy or multiple copies of the ASU or portions of the ASU. Handling of these records is coded in multimer_reconstruction.py as well. The command line tool *phenix.pdb.biomt_reconstruction* is designed to convert the content of PDB file into an expanded set of atoms representing the biological unit.

To exercise the new tools we surveyed the entire PDB and applied them to entries that contain non-trivial (non-unit) MTRIX and BIOMT records. As part of the PDB survey we checked that matrices in MTRIX and BIOMT records are proper rotation matrices (Rotation matrix R is a proper rotation when *Transpose(R) = Inverse(R), Determinant(R) = 1*). We also checked for general formatting issues of these records and more. A summary is presented in table 1.

BIOMT and MTRIX records in PDB (as of Oct. 2013)

Records processing issues include:

- The number of BIOMT matrices is different than the serial number of matrices or have missing records. In many cases this can happen if the identity matrix is listed several times with the

Table 1: PDB survey summary

Number of records surveyed:	93,252
MTRIX	
- Files with no MTRIX records	88,522
- Files with non-trivial matrix MTRIX records	2,202
- Files with non-trivial matrix which also have structure factor (SF) records	1,736
- Non-trivial files with SF records that contains only one NCS independent part	157
- Non-trivial files with SF records that have processing issues	19
BIOMT	
- Files with trivial BIOMT records (only identity matrix)	47,262
- BIOMT records with more than the identity matrix	11,960
- Files with BIOMT records processing issues (see details below)	26,207

same serial number. For example, the PDB file *4dzi* has two biomolecules and the instruction in the file, REMARKS 300 and 350, call for applying the identity matrix on two different chains, for the different biomolecules.

- Missing information in CIF file, such as files where the string "******" replaces some sigma F_meas_sigma values, causing *phenix.cif_as_mtz* to fail (*3m8l*, *2btv*, *2w0c*, *3dpr*).
- Unknown chemical element type in present in PDB file, "HETATM xxx UNK UNX ... " (*2zah, 2wtl, 4fp4*)
- Empty miller_array because R-free-flags are zero in mtz file (*2uu7, 2uwa, 2bnl*)
- "CifBuilderError: Miller arrays _refln.F_calc_1 and _refln.phase_calc are of different sizes " (*1wbi, 2xts, 3nap*)
- "CifBuilderError: Space group is incompatible with unit cell parameters" (*1x33, 2bny*)
- Analysis could not be continued because more than half of the data have values below 1e-16" (*1qez*)
- Symmetry issues, can't convert cif to mtz (*2xvt*)
- CifParserError: lexer error 1 : Unexpected character (*3zll*)
- CifParserError: error 4 : Unexpected token, at offset 11 near release, : unexpected input (*4bl4*)
- Improper transformation, tested using *Transpose(R) approximately equal Inverse(R)* and *Determinant(R)=1* with *eps*. In table 1 we used *eps=0.01*.

Only about 2.4% of all PDB files contain MTRIX information where the rotation matrices are not the identity matrix. Only in 157 files a single NCS copy is present (see Table 2) along with MTRIX records necessary to generate the entire ASU contents. Out of the 157 files with a single NCS copy 8 had processing issues.

There are about 13% of all PDB files for which valid BIOMT records are available. For those files the biological assembly reconstruction function can be used to obtain the complete set of biological assembly atoms coordinates.

If a PDB file contains only one NCS copy, then non-trivial MTRIX records must be provided so the complete ASU content can be generated. R-factors calculated from such expanded file are expected to match the published values (in REMARK 3 record) within some reasonably small tolerance (usually a few percentage points of relative difference).

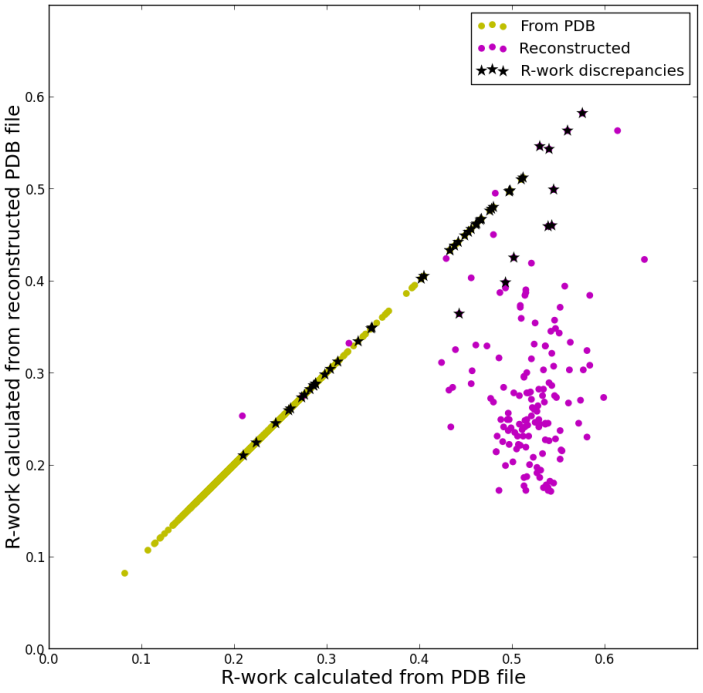


Figure 1: Scatter plot of the R-work calculated using the expanded PDB file (using MTRIX records) (blue) vs. R-work calculated using original PDB file (yellow). In black are 48 files for which the calculated R-work, either from the PDB file (38) or from the reconstructed file (10), differs in more than 50% than the R-work value published in the PDB file records. 1717 files were used for this out of 1736 for which the structure factors are available (19 files were excluded due to processing issues).

If a PDB file contains the entire ASU of atoms, then a single NCS copy is not readily available. R-factors calculated from such PDB file taken as is are expected to closely match published values.

To assert the above statements true and exercise our new tools we selected all PDB entries that have non-trivial MTRIX records and have experimental structure factors available (1736 total). The MTRIX records have a flag indicating whether a file contains only one NCS copy or the entire ASU. For each entry, with only one NCS copy, we calculated two R-factors: using the PDB file as is and after applying MTRIX records. The reconstruction function does not apply the MTRIX records when a file contains the entire ASU.

Table 2: 157 PDB file containing a single NCS (non crystallographic symmetry) copy. PDB columns contain the file name and reported R-work value. ASU is the R-work calculated from the PDB file, and NCS is the value calculated from the expanded NCS. Entries in the table are sorted (from large to small) by R-factor discrepancy between published and calculated from reconstructed ASU content values. PDB files with processing errors are listed at the end.

PDB ID	PDB	ASU	NCS	PDB ID	PDB	ASU	NCS	PDB ID	PDB	ASU	NCS	PDB ID	PDB	ASU	NCS
3p0s	0.209	0.560	0.563	2gtl	0.288	0.521	0.315	3zfe	0.240	0.518	0.231	2xbo	0.247	0.508	0.244
2wws	0.230	0.530	0.546	2g33	0.360	0.563	0.333	4fts	0.259	0.561	0.267	1h8t	0.246	0.497	0.249
2xpj	0.308	0.576	0.582	2zzq	0.242	0.535	0.268	4fsj	0.262	0.574	0.270	5msf	0.188	0.530	0.186
2bfu	0.230	0.545	0.499	2ws9	0.275	0.516	0.249	3raa	0.294	0.543	0.286	4jgz	0.286	0.456	0.288
3n7x	0.278	0.540	0.543	1c8n	0.253	0.502	0.278	2c4q	0.190	0.541	0.182	3vbo	0.227	0.490	0.225
2iz9	0.200	0.543	0.460	2w4z	0.296	0.521	0.271	1qjy	0.233	0.491	0.241	2ztn	0.305	0.577	0.303
1dwn	0.288	0.539	0.459	1x9t	0.306	0.529	0.282	4hl8	0.352	0.510	0.359	1b35	0.228	0.540	0.226
1ei7	0.195	0.443	0.364	3r0r	0.225	0.488	0.249	3tn9	0.265	0.527	0.258	6msf	0.195	0.531	0.194
2wzr	0.230	0.493	0.398	4gb3	0.330	0.525	0.354	3es5	0.248	0.514	0.241	4aed	0.279	0.518	0.278
2vf1	0.273	0.502	0.425	1vsz	0.380	0.456	0.403	1js9	0.238	0.484	0.231	3vbs	0.273	0.477	0.272
1m1c	0.266	0.493	0.392	2g34	0.365	0.551	0.343	4gbt	0.256	0.522	0.262	3vbr	0.257	0.496	0.256
1llc	0.374	0.482	0.495	2c4y	0.192	0.542	0.171	4fte	0.239	0.539	0.245	3vbf	0.236	0.496	0.237
1dzl	0.280	0.557	0.394	1z7s	0.224	0.501	0.203	2x5i	0.288	0.534	0.282	3ra2	0.244	0.517	0.243
2vf9	0.311	0.521	0.419	1ddl	0.151	0.515	0.172	2izn	0.197	0.527	0.191	3kz4	0.328	0.536	0.329
3ntt	0.252	0.547	0.348	2bq5	0.293	0.599	0.273	1zba	0.183	0.513	0.177	1za7	0.245	0.536	0.244
4ar2	0.280	0.509	0.371	2c4z	0.191	0.539	0.172	1r2j	0.247	0.434	0.241	1vb2	0.261	0.524	0.260
4gmp	0.269	0.546	0.357	3fbm	0.294	0.546	0.275	1k5m	0.216	0.507	0.222	1rhq	0.273	0.274	0.274
3vdd	0.230	0.486	0.316	2gh8	0.249	0.512	0.231	2w4y	0.278	0.436	0.284	4iv3	0.235	0.503	0.235
3bcc	0.289	0.509	0.373	1qjx	0.231	0.495	0.249	2qqp	0.285	0.520	0.279	3vbh	0.217	0.505	0.217
3s4g	0.380	0.516	0.300	1f8v	0.219	0.552	0.237	4jgy	0.227	0.497	0.222	3nou	0.390	0.515	0.390
3lob	0.347	0.643	0.423	2iz8	0.192	0.539	0.175	4ftb	0.240	0.572	0.245	3not	0.387	0.515	0.387
3qpr	0.461	0.487	0.387	2bs1	0.245	0.547	0.228	4ang	0.308	0.534	0.303	3nop	0.384	0.514	0.384
1tdi	0.228	0.513	0.296	4aqq	0.291	0.533	0.275	3zfg	0.249	0.530	0.244	2xgk	0.303	0.562	0.303
1ohg	0.373	0.545	0.307	1qju	0.206	0.509	0.221	3zff	0.243	0.511	0.238	2wbh	0.275	0.508	0.275
3e8k	0.365	0.424	0.311	1x36	0.245	0.581	0.230	3cji	0.258	0.521	0.253	2qij	0.331	0.524	0.331
2qzv	0.615	0.614	0.563	1w39	0.245	0.506	0.231	2c51	0.185	0.545	0.180	2c50	0.219	0.515	0.219
2e0z	0.268	0.553	0.216	1x35	0.268	0.432	0.281	1vak	0.211	0.552	0.206	2buk	0.273	0.548	0.273
1wcd	0.219	0.480	0.268	1pgw	0.212	0.493	0.199	1uf2	0.303	0.584	0.308	1wce	0.371	0.552	0.371
4bcu	0.155	0.519	0.200	2wff	0.462	0.480	0.450	1ohf	0.219	0.483	0.214	1tnv	0.384	0.584	0.384
1vcr	0.379	0.429	0.424	2vq0	0.257	0.536	0.245	3ux1	0.283	0.516	0.278	3m8l	--	--	--
1ng0	0.281	0.581	0.324	2fz2	0.276	0.528	0.264	4g0r	0.216	0.533	0.212	2btv	--	--	--
3dar	0.212	0.209	0.253	2fz1	0.309	0.543	0.321	3s6p	0.237	0.529	0.241	2zah	--	--	--
4f5x	0.293	0.536	0.329	1x9p	0.307	0.513	0.295	3ra4	0.212	0.523	0.208	3dpr	--	--	--
4g93	0.379	0.542	0.345	3vbu	0.272	0.491	0.284	2bu1	0.219	0.554	0.215	2w0c	--	--	--
1bcc	0.270	0.457	0.302	3hag	0.277	0.540	0.289	1laj	0.218	0.483	0.214	2bny	--	--	--
2izw	0.294	0.439	0.325	4gh4	0.167	0.537	0.178	1a34	0.179	0.534	0.175	1x33	--	--	--
1f2n	0.218	0.529	0.249	3chx	0.342	0.324	0.332	7msf	0.200	0.527	0.197	3nap	--	--	--
1ny7	0.202	0.486	0.172	1pgl	0.196	0.513	0.186	4iv1	0.190	0.516	0.187				
3oah	0.275	0.525	0.246	1a37	0.320	0.461	0.330	3ra9	0.243	0.499	0.240				
1vb4	0.255	0.536	0.227	1lp3	0.338	0.473	0.329	3ra8	0.245	0.514	0.248				

Figure 1 shows the effect of the reconstruction function of the ASU content from MTRIX records on R-work. As expected, for Protein Databank files containing the complete asymmetric unit, *phenix.pdb.mtrix_reconstruction* does not change the input model or the R-work (points on the diagonal). For most of the 136 entries where the PDB files contain only one NCS copy (Out of the 157, 8 had processing errors), R factors drastically drop, as expected. Table 2 provides a summary of R values for all 157 files with a single NCS copy. It includes reported value, calculated from PDB file as is, and calculated after applying the MTRIX transformation.

R-work values reported in the PDB file and for the model calculated using *mmtbx.f_model.manager* are not exactly the same. In figure 2 we can see that the values for the expanded files are at least as good as those calculated for files with complete ASU. We can also see that there can be significant difference between the reported and the calculated values.

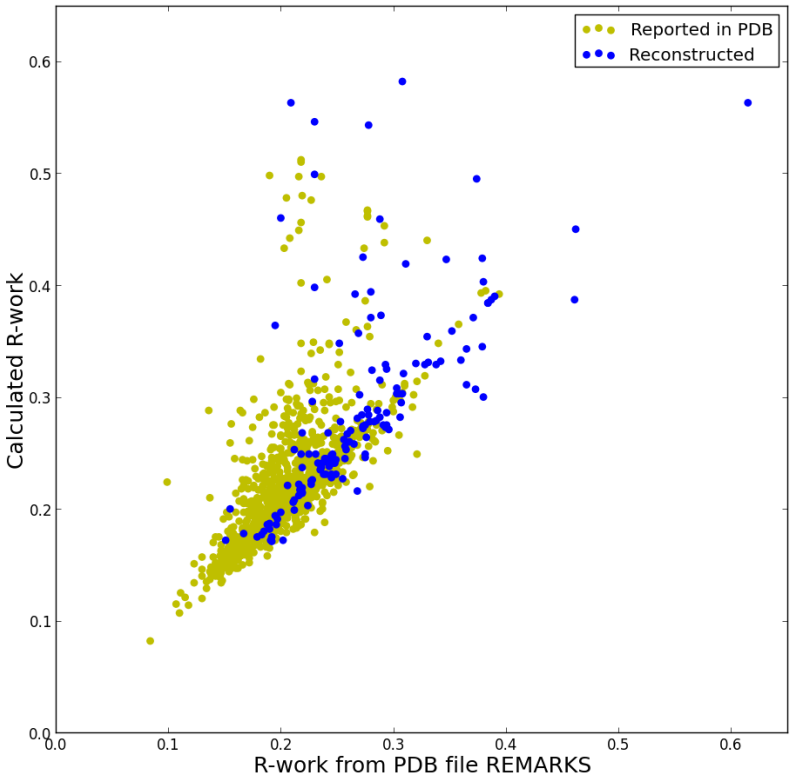


Figure 2: R-work reported in PDB file vs R-work calculated. In yellow are points were the complete ASU were in the PDB file. In blue are the values for the expanded files.

References

Adams PD, Afonine PV, Bunkoczi G, Chen VB, Davis IW, et al. 2010. PHENIX: a comprehensive Python-based system for macromolecular structure solution. *Acta Crystallogr D* 66: 213-21

Afonine PV, Grosse-Kunstleve RW, Echols N, Headd JJ, Moriarty NW, et al. 2012. Towards automated crystallographic structure refinement with phenix.refine. *Acta crystallographica. Section D, Biological crystallography* 68: 352-67

Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, et al. 2000. The Protein Data Bank. *Nucleic Acids Res* 28: 235-42

Bernstein FC, Koetzle TF, Williams GJ, Meyer EF, Jr., Brice MD, et al. 1977. The Protein Data Bank: a computer-based archival file for macromolecular structures. *J Mol Biol* 112: 535-42

Kaufmann B, Bowman, V.D., Li, Y., Szelei, J., Waddell, P.J., Tijssen, P., Rossmann, M.G. 2010. PDB ID: 3n7x. pp. Crystal structure of Penaeus stylirostris densovirus capsid. http://www.rcsb.org/pdb/explore/explore.do?structureId=3n7x: J.Virol.

Kaufmann B, El-Far, M., Plevka, P., Bowman, V.D., Li, Y., Tijssen, P., Rossmann, M.G. 2011. PDB ID: 3p0s. pp. rystal structure of Bombyx mori densovirus 1 capsid. http://www.rcsb.org/pdb/explore/explore.do?structureId=3p0s: J.Virol.

Kleywegt GJ, Jones TA. 1997. Model building and refinement practice. *Method Enzymol* 277: 208-30

Sagurthi SR, Rajaram, V., Savithri, H.S., Murthy, M.R.N. PDB ID: 2wws. pp. PHYSALIS MOTTLE VIRUS: NATURAL EMPTY CAPSID. http://www.rcsb.org/pdb/explore/explore.do?structureId=2wws: To be Published

Sagurthi SR, Rajaram, V., Savithri, H.S., Murthy, M.R.N. PDB ID: 2xpj pp. Crystal structure of Physalis Mottle Virus with intact ordered RNA. http://www.rcsb.org/pdb/explore/explore.do?structureId=2xpj: To be Published

Coping with BIG DATA image formats: integration of CBF, NeXus and HDF5

Herbert J. Bernstein[a], Jonathan M. Sloan[b], Graeme Winter[b], Tobias S. Richter[b], NeXus International Advisory Committee[c], Committee on the Maintenance of the CIF Standard[d]

[a]*Department of Mathematics and Computer Science, Dowling College, Oakdale, NY 11769*
[b]*Diamond Light Source, Harwell Science and Innovation Campus, OX11 0DE (UK)*
[c]*http://wiki.NeXusformat.org/NIAC* *
[d]*http://www.iucr.org/resources/cif/comcifs* †

Correspondence email: yayahjb@gmail.com

Introduction

This is an update to a July 2013 report of the same title (Bernstein, Sloan *et al.* 2013).

The BIG DATA demands of the new generation of X-ray pixel array detectors necessitate the use of new storage technologies as we meet the limitations of existing file systems. In addition, the modular nature of these detectors provides the possibility of more complex detector arrays, which in turn requires a complex description of the detector geometry (for example, see the companion article "XFEL Detectors and ImageCIF", this issue). Taken together these give an opportunity to combine the best of CBF/imgCIF (the Crystallographic Binary File), NeXus (a common data framework for neutron, X-ray and muon science) and HDF5 (Hierarchical Data Format, version 5, the high-performance data format used by NeXus) for the management of such data at synchrotrons. Discussions are in progress between COMCIFS (the IUCr Committee for the Maintenance of the CIF Standard) and NIAC (the NeXus International Advisory Committee) on an integrated ontology. A proof-of-concept API based on CBFlib and the HDF5 API is being developed in a collaboration among Dowling College, Brookhaven National Laboratory and Diamond Light Source. A preliminary mapping and a combined API are under development. Releases of CBFlib since CBFlib 0.9.2.12 can store arbitrary CBF files in HDF5 and recover them, support use of all CBFlib compressions in HDF5 files and can convert sets of miniCBF files to a single NeXus file.

Data Rates, Formats and High Performance X-ray Detectors

CCD X-ray detectors provide images at a moderate data rate of one every few to several seconds (see http://www.adsc-xray.com/Q4techspecs.html). Current higher performance X-ray detectors, such as the DECTRIS Pilatus, are capable of collecting six-megapixel images at 10 - 25 frames per second (Trueb *et al.* 2012), while the newest Pilatus3 6M instruments can operate at 100 frames per second (https://www.dectris.com/pilatus3_specifications.html). The coming next generation of high performance X-ray detectors for MX such as the DECTRIS Eiger will be capable of collecting 16+ megapixel images at more than 125 frames per second (Willmott 2011) (Johnson *et al.* 2012). The ADSC DMPAD is also expected to produce 900 fine-sliced images in steps of two-tenths of a degree at 125 frames per second (Hamlin, Hontz and Nielsen 2012). The Cornell-SLAC pixel array detector (CSPAD) for XFELs produces 120 2.3 megapixel frames per second using 2 bytes per pixel (Hart, Boutet *et al.* 2012). Note that gain-corrected CSPAD images use 8 bytes per pixel. Table 1 shows typical sustained data rates for detectors used for MX at NSLS, DLS, etc. compared to uncompressed XFEL rates (likely to decrease

* The members of NIAC are: Mark Könnecke, Paul Scherrer Institut, Switzerland (Chair), Frederick Akeroyd, Rutherford Appleton Laboratory, UK (ISIS Representative, Technical Committee Chair), Herbert J. Bernstein, ImgCIF, Bjorn Clausen, Los Alamos National Laboratory, USA, Stephen Cottrell, Rutherford Appleton Laboratory, UK (Muon Representative), Jens-Uwe Hoffmann, Helmholtz Zentrum Berlin, Germany, Pete Jemian, Advanced Photon Source, USA (Documentation Release Manager), David Männicke, Australian Nuclear Science and Technology Organisation, Australia, Raymond Osborn, Argonne National Laboratory, USA, Peter Peterson, Spallation Neutron Source, USA, Tobias Richter, Diamond Light Source, UK (Executive Secretary), Armando Sole, European Synchrotron Radiation Facility, France, Jiro Suzuki, KEK, Japan, Benjamin Watts, Swiss Light Source, Switzerland, Eugen Wintersberger, DESY, Germany, Joachim Wuttke, FRM II and JCNS, Germany.

† The members of COMCIFS are: James R. Hester (Chair), Herbert J. Bernstein, John C. Bollinger, Brian McMahon (Coordinating Secretary), John Westbrook.

Table 1. Typical Sustained Data Rates

Detector	Raw Image Size (MB)	Frame Rate (Hz)	Compressed Rate (Gb/sec)	USB Disk Data Rate (%)
ADSC Q315 (2x2 binned)	18	0.37	.013	7
Pilatus 2 6M	24	10	.48	240
Pilatus 2 Fast 6M	24	25	1.2	600
CSPAD	4.6	120	4.4	2208
Pilatus 3 6M	24	100	4.8	2400
Eiger 16M	72	125	18	9000

with suitable compression) and expected rates from Eiger, expressed in terms of the typical data rate for an inexpensive USB disk of 25 MB/sec = 200 Mb/sec. A data management system designed for very large numbers of files as well as for very large data volumes and data rates is needed (Fig. 1). Efficient recording of metadata coordinated with the data is also needed and database access to information about images and experimental runs is needed. For MX, these data rates, data volumes, numbers of distinct images and numbers of distinct experiments argue for a very organized, high performance infrastructure. HDF5 and NeXus provide the necessary organization of the raw data and CBF provides the necessary organization of the associated metadata for subsequent processing as well as contributing useful compression algorithms.

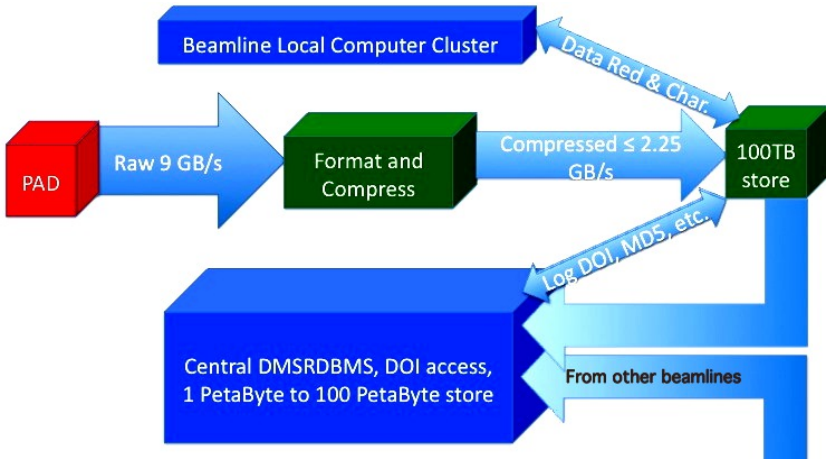


Figure 1. Major data flows from the beamline advanced pixel array detectors (PADs) to the data management system relational database (DMSRDBMS). In order to be manageable, the raw 9 gigabyte per second data flow from each PAD needs to be compressed locally by at least 4:1, before going into a beamline 100 terabyte store for beamline local computer cluster access for up to a week for data reduction and characterization. The required bandwidth of the pipes from the beamlines to the DMSRDBMS depends on the compression used. If no further compression is used, 2.25 gigabyte (18 gigabit) per second per beamline network connections are required. If a combined lossless/lossy compression is used, then 225 - 450 megabyte (1.8 to 3.6 gigabit) per second per beamline network connections will suffice. The flows for transfers to user home institutions are not shown.

Today for MX alone Diamond Light Source employs one Pilatus 2M, three Pilatus 6M fast and one Pilatus 3 6M, giving a combined data rate of over 1 GB/sec and over 200 files/sec, creating the need to manage hundreds of thousands of images each day. For the Advanced Beamlines for Biological Investigations with X-rays (ABBIX) that are being built for NSLS-II (Hendrickson 2012), just two of the beam lines, the Frontier Macromolecular Crystallography (FMX) beamline and the Automated Macromolecular Crystallography (AMX) beamline (Schneider *et al*. 2012), are expected to produce an aggregate of more than 94 terabytes per operational half day, 660 terabytes per week or 38 petabytes per year. The anticipated beamline flux is 10^{13} photons per second for FMX and $2x10^{13}$ photons per second for AMX, approximately 50 times the NSLS X25 and X29 fluxes. One subtle effect of these high fluxes is that there will be more photons per pixel in images, making them more difficult to compress.

A final issue, in addition to the actual recording of data, is that of automated processing. At Diamond Light Source and elsewhere there has been a push towards the automated analysis of diffraction data, as interactively processing diffraction data at the current rate of typically 20 data sets per hour

per beamline is unsustainable. This however places an increased strain on the file system, as typically the same data are read as many as six times in order to be processed, resulting in over 1000 file access operations per second. With the storage of many frames per file, as planned for NeXus, the rate of file operations would decrease substantially.

HDF5 and NeXus

The Hierarchical Data Format Version 5 (HDF5) is a self-describing file format with a robust, well-documented API routinely handling multi-gigabyte files of data. It has a diverse user community covering a wide range of disciplines and is fully supported (Dougherty *et al.* 2009). HDF5 is particularly well suited to the management of very large volumes of complex scientific data and has been adopted as the primary data format in a wide range of disciplines (http://www.hdfgroup.org/HDF5/users5.html) and provides the "inner workings" of important frameworks, such as NetCDF (Rew *et al.* 2004) and NeXus. To avoid confusion we use the term *format* to describe the logical organization of data on a storage medium. An *ontology* is a dictionary of terms that may include descriptions of the relationships between terms. An ontology can be realized in one or more formats. We are therefore dealing with the HDF5 *format*, the NeXus *ontology*, a CBF *format* and an imgCIF *ontology*. The HDF5 *format*, XML *format* and NeXus *ontology* together form the NeXus data transfer *framework*. The CBF *format*, CIF *format* and imgCIF *ontology* form the imgCIF data transfer *framework*.

HDF5 is tree-oriented, which is a very powerful and useful characteristic allowing file-system-like nesting of groups of data within groups of data, in order for information to be easily, reliably and efficiently searched. However, tables are more useful for loading information into a relational database management system (Codd 1970).

NeXus (Filges 2001) (Könnecke 2006) is a tree-oriented ontology for use wth HDF5 (and XML and HDF4) of importance in managing neutron and X-ray data. NeXus adds rules for storing data in files and a dictionary of documented names to HDF-5 in order to make HDF-5 applicable to the problem domain of synchrotron, neutron and muon scattering. NeXus is a convenient thin layer over HDF5 that is widely used at many physics research centers, including at synchrotrons. Together NeXus and HDF5 provide a portable, extensible and efficient framework for the storage and management of data.

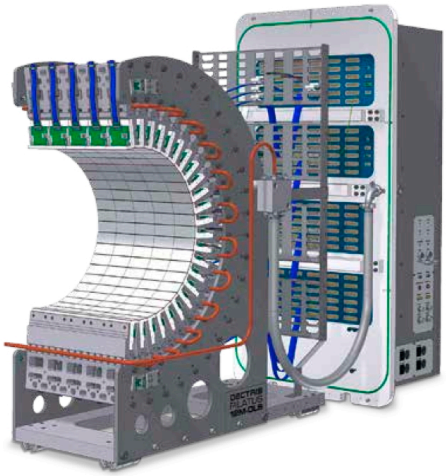


Figure 2. Curved DECTRIS detector for DLS beamline I23, an example of a detector with a complex geometry best described using the imgCIF/CBF ontology. Prior to this work, NeXus could not support a detector such as this. Now it will be possible.

Jan 2013 DECTRIS Eiger Workshop and Followup

The attendees at the January 2013 DECTRIS Workshop agreed on the use of an HDF5-based NeXus framework for the DECTRIS Eiger pixel array detector. The workshop charged Herbert J. Bernstein with following up on mapping additional terms to the new format. Tobias Richter, Jonathan Sloan and Herbert J. Bernstein have worked on a CBF-NeXus concordance and supporting software based on CBFlib and HDF5 with the cooperation of Bob Sweet, Graeme Winter and Mark Koennecke. Discussions with NIAC were held and then discussions with COMCIFS were held prior to ECM 28 in August 2013. There was general agreement that it was a good idea to have CIF and NeXus interoperate. COMCIFS and NIAC have agreed to start on a single crystal monochromatic macromolecular crystallography experiment NeXus application definition. An application definition in NeXus is a specification of the required metadata and data for that application. Significant progress has been made on the application definition and a draft will be available in Spring 2014.

Mapping from NeXus to CBF

All NeXus base classes now have proposed slots in CIF categories. Handling of the DECTRIS-proposed Eiger HDF5 format is in the concordance. This concordance will require some relaxation of current NeXus name practices. "CBF_" prefixes are being used as an interim solution. Most of these prefixes are expected to be removed in the final version.

For this project, organizing data and metadata according to the conventions of the IUCr Crystallographic Information File (Hall *et al.* 1991) using imgCIF (Bernstein, Hammersley, 2005) and its open source supporting software CBFlib (Ellis, Bernstein 2005) provides a database-friendly tabular structure. The imgCIF ontology provides the metadata needed for the analysis of diffraction images and is supported by all the major detector manufacturers. This aspect is particularly important for instruments with complex geometries, e.g., the Pilatus 12M being constructed by DECTRIS for the long wavelength beamline I23 at Diamond Light Source (Fig. 2).

The embedding of CIF tables in HDF5 files was demonstrated at the "HDF5 as hyperspectral data analysis format" workshop in January 2010 (Götz *et al.* 2010). The workshop recommendation was, in part, "Adopt as much as possible from imgCIF and sasCIF".

Tables are easily embedded into trees. Going in the other direction is more difficult. There is serious effort required to make general trees into tables suitable for use in a relational database management system, involving a process known as "normalization"(Codd 1972). See Fig. 3.

One of the tasks of this project is to extend the imgCIF ontology to ensure workable database access to metadata in the HDF5 tree that has not already been normalized into CIF categories. For example, Digital Object Identifiers (DOIs) and SHA2 or SHA3 checksums from multiple experiments will need to be brought forward into a common table for post-experiment forensic validation.

CBF and Database Access

The Crystallographic Binary File (CBF) format is a complementary format to the Crystallographic Information File (CIF), supporting efficient storage of large quantities of experimental data in

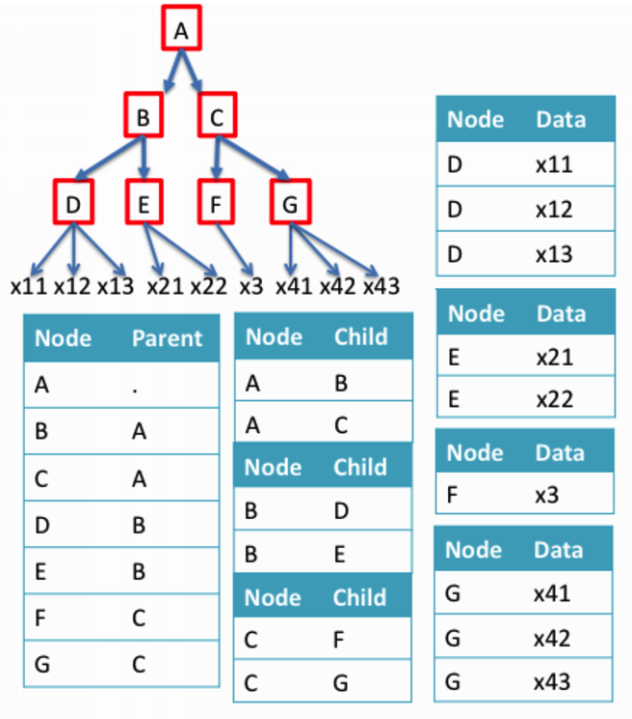


Figure 3. Example of a tree mapped into tables for database access identifying the links between parents and children in both directions as well as data. This is about the simplest example we can provide of a tree that demonstrates the need for "normalization" in the conversion from trees to tables.

a self-describing binary format with a sophisticated description of the experimental geometry. For large PAD images, the raw binary CBF format is heavily used both within laboratories and for interchange among collaborating groups. When dealing with large numbers of independent experiments producing large numbers of CBF/imgCIF image files, HDF5 provides the virtual file system needed to manage the massive data flow. While it is feasible to simply encapsulate the CBF/imgCIF image files as opaque objects within an HDF5-based data-management system, active management of the data can be done more efficiently when the imgCIF tags are made visible in the HDF5 tree, a capability demonstrated in 2010.

With the anticipated throughput of NSLS II beamlines and the current capabilities of MX beamlines equipped with pixel array detectors, the management of data and the possibility of interrogating the data files for experimental information becomes critical.

Figure 3. Tests on fifty image files from DLS with low to moderate pixel density. Total size 1.2 gigabytes. Relative times are shown. Multiply by 3 for the number of cores needed to keep up with the compression workload. The fifty files are from a set of 900 images recorded by Graeme Winter at Diamond Light Source beamline I02 as part of routine development work, and come from a crystal of DNA (TCGGCGCCGA) bound to a large ligand (lambda-[Ru(TAP)2(dppz)]2+). The aggregate of fifty was chosen to produce an uncompressed file small enough (1.2 GB) to be acceptable at user sites. Larger aggregates could be used for sites able to accommodate larger files.

Compression Method	Compression Ratio	Relative Time
external bzip2 compression	20.4:1	5.6
HDF5 CBFlib canonical compression	15.7:1	3.9
HDF5 CBFlib nibble offset compression	11.5:1	2.9
HDF5 CBFlib packed V2 compression	11.0:1	2.8
HDF5 zip compression	9.7:1	2.4
external LZ4 compression (C1 one pass)	8.7:1	2.2
HDF5 CBFlib packed compression	8.6:1	2.2
external LZ4 compression (C0 two passes)	5.2:1	1.3
HDF5 CBFlib byte offset compression	4.0:1	1.0

Compression

There are long-standing issues about compression in crystallography. High-speed, high-compression-ratio compression is a critical issue for the next generation of detectors. Some compressions raise license issues. Some popular ones are slow or inefficient or both. Some can be handled in processing programs such as XDS if license and programming language issues can be addressed. Low pixel density fine-slicing with clean backgrounds makes some compressions more effective.

CBFlib provides useful compressions. See Table 2. A plugin module has been written to allow HDF5 to read and write CBFlib compressions. Starting with CBFlib release 0.9.2.11, that module is included. HDF5 1.8.11 and later is required. For general documentation on HDF5 dynamically loaded filters, see

http://www.hdfgroup.org/HDF5/doc/Advanced/DynamicallyLoadedFilters/HDF5DynamicallyLoadedFilters.pdf

The filter has been registered with the HDF5 group as 32006and cbf.h includes the symbolic name for the filter CBF_H5Z_FILTER_CBF. The source and header of the CBFlib filter plugin are cbf_hdf5_filter.c and cbf_hdf5_filter.h, respectively, in the CBFlib kit. To use the filter in C applications, you will need to include cbf_hdf5_filter.h in the application and have the cbflib.so library in the search path used by HDF5 1.8.11. Each compressed image in the HDF5 file is in the same format as the MIME-headed compressed images in the corresponding CBF, so the Fortran image search logic used in XDS can be used directly on these files.

Where to Find Software and Documentation

Draft imgCIF/CBF version 1.7 dictionary that now includes information on going from CBF to NeXus: https://www.sites.google.com/site/nexuscbf/home/cbf-dictionary

PDF summary of the concordance: https://www.sites.google.com/site/nexuscbf/mapping-draft

CBFlib kit: http://downloads.sf.net/cbflib/CBFlib-0.9.3.3.tar.gz

that includes both Jonathan Sloan's utilities to convert sets of minicbfs into a single NeXus file and a plugin filter that supports the full set of CBFlib compressions in HDF5.

Conclusion

The essential first steps in the integration of CBF, NeXus and HDF5 have been taken. There is work still to be done in applying this work at beam lines and in data processing software. Collaborators are most welcome.

Acknowledgements

Our thanks to James Hester, Chair of COMCIFSand Mark Koennecke, Chair of NIAC, for supporting and encouraging this effort and to all participants in COMCIFS and NIAC for helpful commentsand also to James Hester for contributing the section on ontologies and formats, a critical issue in this

work. Our thanks go to Nick Sauter and Aaron Brewster for extending this work to CSPAD. Our thanks for years of supporting efforts, at the BNL PXRR Group: Robert M. Sweet, Dieter Schneider, Howard Robinson, John Skinner, Matt Cowan, Leonid Flaks, Richard Buono; at DLS: Alun Ashton, Bill Pulford; and at the Dowling College ARCiB Lab Group: Mojgan Asadi, Kostandina Bardhi, Maria Karakasheva, Ming Li, Limone Rosa

Our thanks to DECTRIS, BIOIHDFand the HDF Group

Our thanks to Frances C. Bernstein

This work was funded in part by NIGMS, DOE, NSF, PaNdata ODI (EU 7th Framework Programme)

References

Bernstein HJ and Hammersley AP (2005). “Specification of the Crystallographic Binary File (CBF/imgCIF)". In: S. R. Hall and B. McMahon, Eds., International Tables For Crystallography, Chap. 2.3, pp. 37 - 43, International Union of Crystallography, Springer, Dordrecht, NL.

Bernstein HJ, Sloan JM, Winter G, Richter TS, NeXus International Advisory Committee and Committee on the Maintenance of the CIF Standard (2013). "Coping with BIG DATA Image Formats: Integration of CBF, NeXus and HDF5." poster, American Crystallographic Association, 2013 Annual Meeting. Honolulu, HI.

Codd EF (1970). “A relational model of data for large shared data banks”. Communications of the ACM, Vol. 13, No. 6, pp. 377 - 387.

Codd EF (1972). Courant Computer Science Symposium 6, Chap. “Further Normalization of the Data Base Relational Model”, pp. 33 - 64. Prentice-Hall.

Dougherty MT, Folk MJ, Bernstein HJ, Bernstein FC, Eliceiri KW, Benger W, Zadok E and Best C (2009). “Unifying Biological Image Formats with HDF5”. Communications of the ACM, Vol. 52, No. 10, pp. 42 – 47.

Ellis PJ and Bernstein HJ (2005). Definition and Exchange of Crystallographic Data, International Tables For Crystallography, Chap. “CBFlib: an ANSI C library for manipulating image data”, pp. 544 -- 556. International Union of Crystallography, Springer, Dordrecht, NL.

Filges U (2001). “The new NeXus API based on HDF5". In: VITESS Workshop Berlin, 25 - 27 June 2001.

Götz A, Solé V, Madonna C and Maydew AF (2010). “ELISA VEDAC Workshop Report, Workshop Title: HDF5 as hyperspectral data exchange and analysis format, Grenoble, January 11th - January 13th, 2010”. http://vedac.esrf.eu/public-discussions/hdf5-workshop/workshop-report.

Hall SR, Allen FH and Brown ID (1991). “The Crystallographic Information File (CIF): a new standard archive file for crystallography”. Acta Crystallographica Section A: Foundations of Crystallography, Vol. 47, No. 6, pp. 655 – 685.

Hamlin RC, Hontz T and Nielsen C (2012). “The New Dual Mode Pixel Array Detector" in: Meeting of the American Crystallographic Association, Boston, MA, 28 July - 1 August 2012, American Crystallographic Association, Abstract 11.01.1151.

Hart P, Boutet S, Carini G, Dubrovin M, Duda B, Fritz D, Haller G, Herbst R, Herrmann S, Kenney C, Kurita N, Lemke H, Messerschmidt M, Nordby M, Pines J, Schafer D, Swift M, Weaver M, Williams G, Zhu D, Van Bakel N and Morse J (2012). "The CSPAD megapixel x-ray camera at LCLS." Proceedings of SPIE 8504: 85040C-85040C-85011.

Hendrickson WA (2012). “NSLS-II - Status of the Life Sciences Program”. Tech. Rep., Brookhaven National Laboratory, X6A Science Advisory Committee, February 2012. http://protein.nsls.bnl.gov/mediawiki/images/e/e3/Hendrickson_2012.pdf.

Johnson I, Bergamaschi A, Buitenhuis J, Dinapoli R, Greiffenberg D, Henrich B, Ikonen T, Meier G, Menzel A, Mozzanica A, Radicci V, Satapathy DK, Schmitt B and Shi X (2012). “Capturing dynamics with Eiger, a fast-framing X-ray detector”. Journal of Synchrotron Radiation, Vol. 19, No. 6, pp. 19, 1001 -1005.

Könnecke, M (2006). “The State of the NeXus data Format”, Physica B: Condensed Matter Vol. 385–386, Part 2, 15 November 2006, 1343–1345, Proceedings of the Eighth International Conference on Neutron Scattering.

Rew R, Ucar B and Hartnett E (2004). “Merging netCDF and HDF5”. In: 20th Int. Conf. on Interactive Information and Processing Systems.

Schneider DK, Sweet RM and Skinner J (2012). “Projection of MX ABBIX needs at AMX and FMX for Data Acquisition, Data Processing, Software, Data Archiving and Networking". February 2012. Private Communication.

Trueb P, Sobott BA, Schnyder R, Loeliger T, Schneebeli M, Kobas M, Rassool RP, Peake DL and Broennimann C (2012). “Improved count rate corrections for highest data quality with PILATUS detectors”. Journal of Synchrotron Radiation, Vol. 19, No. 3, pp. 347 - 351.

Willmott P (2011). An Introduction to Synchrotron Radiation: Techniques and Applications. John Wiley and Sons, Chichester, UK. page 6.

XFEL Detectors and ImageCIF

Aaron S. Brewster[a], Johan Hattne[a], James M. Parkhurst[b], David G. Waterman[c], Herbert J. Bernstein[d], Graeme Winter[b], and Nicholas K. Sauter[a]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
[b]*Diamond Light Source, Harwell Science and Innovation Campus, OX11 0DE (UK)*
[c]*CCP4, Research Complex at Harwell, STFC Rutherford Appleton Laboratory, OX11 0FA (UK)*
[d]*Department of Mathematics and Computer Science, Dowling College, Oakdale, NY 11769*

Correspondence email: asbrewster@lbl.gov

Introduction

Serial femtosecond crystallography performed using X-ray free electrons lasers (XFELs) creates a challenging task for modern detectors (Chapman *et al.* 2011, Boutet *et al.* 2012, Kern *et al.* 2013). Pulses containing 10^{12} photons, 40-50 femtoseconds (fs) long are generated using a linear accelerator and impact a liquid jet of protein microcrystals at a rate of 120 pulses per second. Still image diffraction patterns from thousands of crystals can be collected in a matter of minutes. The Cornell-SLAC pixel array detector (CSPAD) is a unique detector designed to operate at these rates and record data from exposures on the fs time scale (Hart *et al.* 2012). The CSPAD is modularized into 32 sensors arranged in a roughly square pattern. This creates unique challenges for representing the data in such a way that the geometric layout of the experiment is accurately recorded. To this end, we have adopted and extended the ImageCIF/CBF file specification (Bernstein & Hammersley 2005) to record CSPAD diffraction data, adding sufficient parameters to the ImageCIF dictionary to lay out the XFEL experiment at SLAC, including fully specifying the detector geometry. The new ImageCIF parameters described below help us handle the detector geometry by expressing it easily refineable terms. This extensibility and the ability to parameterize the entire experiment explicitly made ImageCIF/CBF the best option for representing this data. The CSPAD CBF format is natively understood by *cbflib* (Ellis & Bernstein 2001), a software package developed specifically for reading and writing CBF files. We have also incorporated the format into *cctbx* (Grosse-Kunstleve *et al.* 2002, Sauter *et al.* 2013), using a new multi-tile detector model defined by the module *dxtbx* (the diffraction experiment toolbox)(Waterman *et al.* 2013, Parkhurst *et al.* in preparation). These software packages allow us to refine the experimental geometry against measured data, leading to better indexing rates and more accurate integration of the reflection signal (Hattne *et al.* submitted).

CSPAD Detector Geometry

Three full-size CSPAD detectors are in service at the present, two at the Linac Coherent Light Source (LCLS) Coherent X-ray Imaging (CXI) instrument, and one at the LCLS X-ray Pump Probe (XPP) instrument. Each detector is comprised of 32 sensors and each sensor houses two application-specific integrated circuits (ASICs), 194x185 pixels in dimension, with a pixel size of 110 microns and a three-pixel gap between them (Figure 1). 8 sensors comprising 16 ASICs form a quadrant. The four quadrants surround a central hole, through which the undiffracted beam passes. The CXI detector quadrants are adjustable on diagonal rails, allowing the central size to grow and shrink. This allows the second detector, typically positioned 2.5 meters behind the first, to receive signal.

The 32 sensors are not orthogonalized, meaning edges between sensors are not parallel; each sensor is tilted slightly off of 90°. Further, the sensors are not co-planar with the detector, having small angles off of the planar normal. LCLS provides optical measurements to position the sensors in three-dimensional space, and these measurements have been enormously useful in specifying the detector geometry. At the CXI beamline, quadrant positions are not provided, as they are variable. Their location needs to be experimentally determined, initially by aligning the quadrants to rings from powder diffraction, and subsequently refined against single crystal diffraction. For both CXI and XPP, detector tilt and position need to be refined as well. For example, the beam itself is not always perfectly parallel to the detector rail, leading to small changes in beam center at different detector distances. All of this geometric information needs to be recorded for each still in a way understandable by developers working on indexing and integration while still

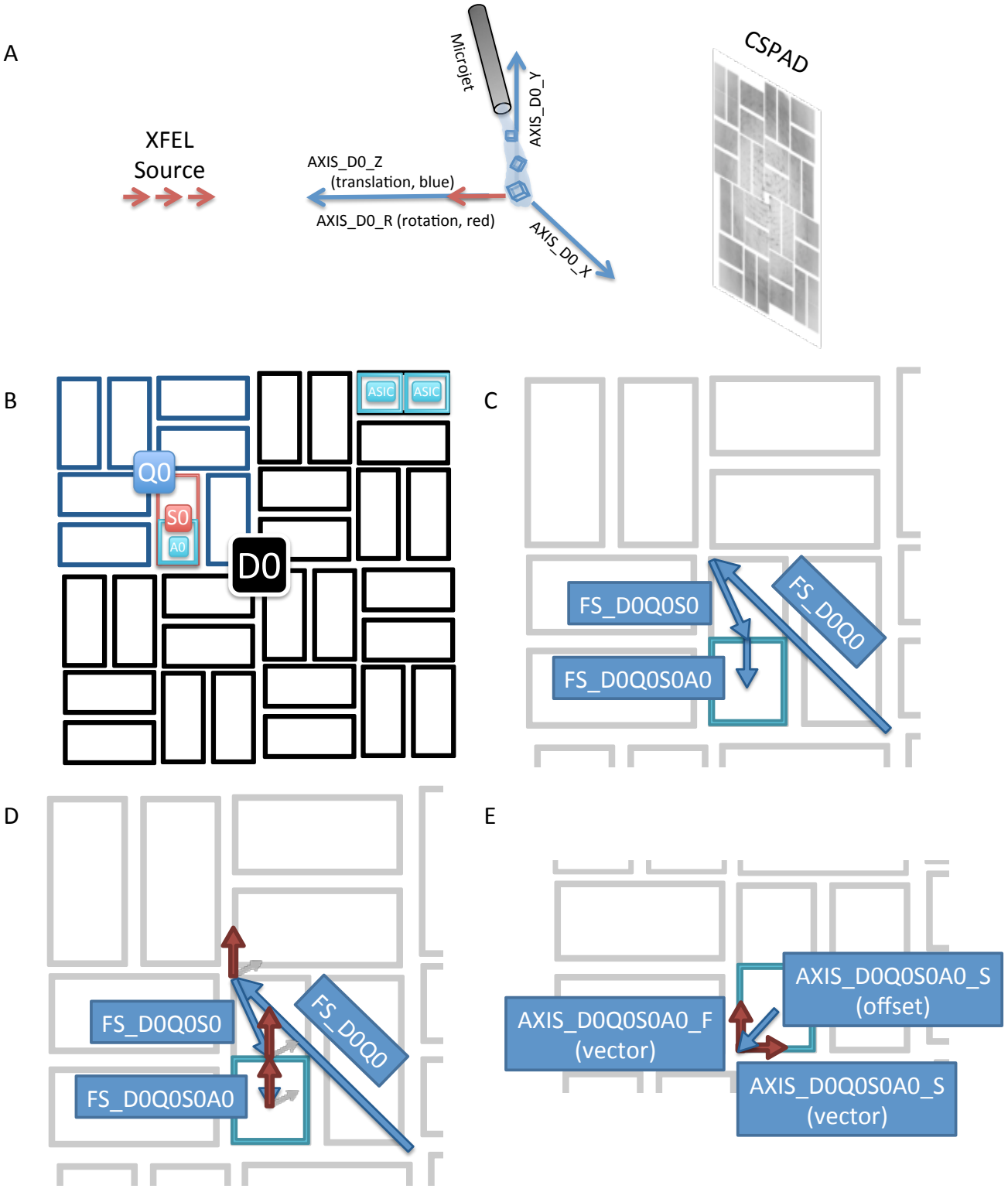


Figure 2: ImageCIF axes describing the CSPAD detector. A) XFEL experiment overview. Crystals are injected into the XFEL stream via a micro-injection system. The root ImageCIF axes for the CSPAD detector as a whole are shown. Axes AXIS_D0_X, _Y and _Z are translation axes along which the detector can be moved. Detector distance is specified as a translation along AXIS_D0_Z. A fourth axis, AXIS_D0_R, defines a rotation axis around which the detector can be rotated. B) Overview of the detector. Rectangles are 32

Caption continues on the following page.

sensors, each comprising 2 ASIC pixel array chips, as shown in the upper right hand corner. D0, Q0, S0 and A0 are highlighted, corresponding to detector zero (*i.e.*, the detector as a whole), quadrant zero in the upper left hand corner, sensor zero and ASIC zero. C) Frame shift vector offsets for the three rotation axes that position the quadrant, sensor, and ASIC centers. CBF rotation axes have three components: an offset from the base of the parent axis to its base, a vector describing the direction around which to rotate, and a rotation angle. Three axis offsets are shown (blue) that together describe the position of an ASIC relative to the center of the detector. D) Vector components (red) of the three rotation axes shown in B. Arrows are normal to the respective surface planes. Around these axes the various elements are rotated into position by specifying appropriate angles. E) Vectors (red) and offsets (blue) for the 2 fast and slow axes of an ASIC element. Here, the red vectors are translation axes that are co-planar with the ASIC chip. They relate how the pixel array is laid out in space to its in-memory arrangement. The blue arrow is the slow axis' offset from the center of the ASIC. The fast axis depends on the slow axis so its offset is zero.

being easily parameterized for refinement.

ImageCIF/CBF

ImageCIF is a specific CIF dictionary for representing diffraction data. Binary encoding of the pixel array data together with ImageCIF metadata comprises the Crystallographic Binary File format (CBF). In use by a variety of companies to record diffraction frame data, ImageCIF and CBF are internationally agreed upon standards maintained by the International Union of Crystallography. ImageCIF allows complete description of the geometry of the crystallographic experiment. For these reasons, we found it applicable to our needs.

In ImageCIF, one describes frame data in the form of a blueprint for the detector. First, the individual pixel-array elements are defined. In the case of the CSPAD, 64 elements are specified:

```
loop_
_diffrn_detector_element.id
_diffrn_detector_element.detector_id
 ELE_D0Q0S0A0 CSPAD_FRONT
 ELE_D0Q0S0A1 CSPAD_FRONT
 ELE_D0Q0S1A0 CSPAD_FRONT
 ELE_D0Q0S1A1 CSPAD_FRONT
...
 ELE_D0Q3S7A0 CSPAD_FRONT
 ELE_D0Q3S7A1 CSPAD_FRONT
```

Here, a new CIF table is defined with the 'loop_' keyword, named diffrn_detector_element. The table links elements by their IDs to a detector ID (CSPAD_FRONT). Multiple detectors can be defined in the same file; if the second detector at CXI, known as the back detector and positioned up to 2.5 meters behind the front detector, is in use, its data and metrology could be recorded in the same file. The convention we use for naming the CSPAD elements includes IDs for the detector (D), quadrant (Q), sensor (S) and ASIC (A). Thus ELE_D0Q0S1A0 is the array of pixels that represents detector 0, quadrant 0, sensor 1, ASIC 0. Later in the file, each of these elements has a separate binary encoding of their pixel data. Other tables specify gain, array dimensions and further physical properties of each element.

Once elements are defined, the geometry of the detector is laid out using two tables: axis and diffrn_scan_frame_axis. The axis table specifies lines of motion and axes of rotation for the experiment, while the diffrn_scan_frame_axis table specifies physical settings for detector components along the axes specified in the axis table. The ImageCIF axis convention specifies the origin to be the sample position, with the X-axis pointing along the axis of right-handed goniometer rotation, the Z-axis pointing to the beam source, and the Y-axis completing a right handed system (Figure 1A). In the case of many XFEL experiments, no goniometer is present, so the X-axis is simply orthogonal to the beam and gravity. Thus the first few lines of the CSPAD axis table are shown in scheme 1.

Each line defines an axis by its type: general, translation or rotation. Equipment refers to kinds of devices that move along the given axis. Other examples include goniometer axes, which XFEL experiments generally do not include. The vector specifies either the direction of translation or the axis about which rotation is performed, and the offset positions the base of the axis in space relative to the parent axis specified in the depends_on field. Finally, equipment_component is a new field we have added to the ImageCIF dictionary in collaboration with its principal maintainer, Herbert Bernstein. This field allows us to group axes together, which will be important to distinguish hierarchy level later when we construct a detector model using *dxtbx* software.

Axis positions are specified in the diffrn_scan_frame_axis table:

```
loop_
_axis.id
_axis.type
_axis.equipment
_axis.depends_on
_axis.vector[1]
_axis.vector[2]
_axis.vector[3]
_axis.offset[1]
_axis.offset[2]
_axis.offset[3]
_axis.equipment_component
 AXIS_SOURCE     general     source    .              0  0 1   .   . . .
 AXIS_GRAVITY    general     gravity   .              0 -1 0   .   . . .
 AXIS_D0_Z       translation detector  .              0  0 1   .   . . detector_arm
 AXIS_D0_Y       translation detector AXIS_D0_Z       0  1 0   .   . . detector_arm
 AXIS_D0_X       translation detector AXIS_D0_Y       1  0 0   .   . . detector_arm
 AXIS_D0_R       rotation    detector AXIS_D0_X       0  0 1   0   0 0 detector_arm
```

Scheme 1: ImageCIF 'loop' table. The first 12 lines comprise a header in which the table and each of its 11 columns are named. The first 6 axes are also shown, describing the detector as a whole and its orientation in the laboratory.

```
 FS_D0Q0         rotation    detector AXIS_D0_R       0  0 1 -50  42 0 detector_quadrant
 FS_D0Q0S0       rotation    detector FS_D0Q0         0  0 1  11 -23 0 detector_sensor
 FS_D0Q0S0A0     rotation    detector FS_D0Q0S0       0  0 1 -11   0 0 detector_asic
```

Scheme 2: These three axis entries in the loop table correspond to frameshifts describing the transition from the detector as a whole to quadrant 0, from quadrant 0 to sensor 0, and from sensor 0 to ASIC 0.

```
 AXIS_D0Q0S0A0_S translation detector FS_D0Q0S0A0     0 -1 0 -11  10 0 detector_asic
 AXIS_D0Q0S0A0_F translation detector AXIS_D0Q0S0A0_S 1  0 0   0   0 0 detector_asic
```

Scheme 3: The fast and slow axes of an ASIC. The slow axis is offset (-10, 11, 0) mm from the ASIC center and points in the -Y direction (in relation to its parent). The fast axis depends on the slow axis and points in the X direction (in relation to its parent).

```
loop_
_diffrn_scan_frame_axis.axis_id
_diffrn_scan_frame_axis.frame_id
_diffrn_scan_frame_axis.angle
_diffrn_scan_frame_axis.displacement
 AXIS_SOURCE  FRAME1 0    0
 AXIS_GRAVITY FRAME1 0    0
 AXIS_D0_X    FRAME1 0    0
 AXIS_D0_Y    FRAME1 0    0
 AXIS_D0_Z    FRAME1 0 -171
 AXIS_D0_R    FRAME1 0    0
```

While both angle and displacement can be specified, only the one or the other is meaningful, depending on if the axis is a rotation or translation axis. The detector distance is specified by translating 171 mm along AXIS_D0_Z (in the negative Z direction since Z points to the source).

Once the detector position is specified, subcomponents are laid out in the axis table (Figures 1B, 1C and 1D) as shown in scheme 2. Here, the frame shifts needed to position quadrant 0, detector 0 and asic 0 are specified. Because these are not mechanical axes, we adopt a convention of naming them FS_ for frame shift instead of AXIS_. These are rotation axes to allow sensor rotations to be specified in the diffrn_scan_frame_axis table:

```
 FS_D0Q0     FRAME1    0 0
 FS_D0Q0S0   FRAME1 89.7 0
 FS_D0Q0S0A0 FRAME1    0 0
```

We can see that sensor 0 is rotated 89.7 degrees around its rotation axis specified in the axis table, the (0, 0, 1) axis *i.e.* the Z-axis. In reality, the sensor is tilted slightly from normal. Another CBF file we have generated records the sensor 0 axis vector to be (-0.000974376302058 0.00044773585801 0.999999425062), indicating a very slight tilt from the normal (about 0.6°). ImageCIF allows us to record even this small error, improving the accuracy of the detector description.

Finally, the fast and slow axes are specified for each asic tile in the axis table (Figure 1E, scheme 3). Note that the slow axis is offset from the center

```
# read the file
image = reader(filename)

# iterate through the quadrants of the detector
detector = reader.get_detector()

quadrants = detector.hierarchy()
for quadrant in quadrants:
       # vector pointing to the center of the quadrant relative to the
       # center of the detector
       origin = quadrant.get_origin()

       # unit vectors pointing in the fast and slow directions of the
       # quadrant plane
       fast = quadrant.get_fast_axis()
       slow = quadrant.get_slow_axis()

       # these three vectors form a 3D basis for this quadrant
       <optimize 9 parameters against a set of measured data>

       quadrant.set_frame(refined_fast,
                          refined_slow,
                          refined_origin)

# apply the detector object changes to the image's internal cbf handle
image.sync_detector_to_cbf

#write the new file
image._cbf_handle.write_file(new_filename)
```

Scheme 4: Pseudo-Python code describing a possible optimization of the four quadrant positions.

of the ASIC, positioning it at the (0, 0) pixel. The fast and slow axes are unit vectors that specify the readout directions for the data stored in the CBF binary sections. These entries, together with the above information, completely describe the detector geometry.

dxtbx and CSPAD ImageCIF

Recently, we have collaborated with researchers at the Diamond Light Source in the UK to develop a new *cctbx* component, the diffraction experiment toolbox *dxtbx*. This toolbox provides Python and C++ based interfaces for generically reading crystallographic data regardless of file format. Importantly, the toolbox exposes models of the diffraction experiment through a set of four interfaces, the detector, the scan, the goniometer and the beam. The developer can sub-class from more general file reader classes and expose the detector geometry through these interfaces. For the purpose of XFEL data (still data), only the detector and beam models are useful.

We have written an appropriate generic reader for multi-tile detector data in CBF format, and ensured its compatibility with this CSPAD CBF format. The reader reads the axis list and creates a hierarchy of components using the equipment_component tag in the axis table to group axes together. Scheme 4 is an example of Python code that uses this reader to read a CSPAD CBF file and show how the hierarchy can be used to refine quadrant positions.

The hierarchical model provides powerful tools for interacting with detector geometry to accomplish tasks of importance to XFEL data collection in a straightforward manner.

Finally, XFEL sources can produce hundreds of thousands of individual diffraction patterns. Representation of each pattern as a single CBF file in hard disk storage can be detrimental to file system performance, a problem exacerbated when handling large numbers of experimental runs, each with many files. Use of HDF5 reduces the file system burden for large numbers of runs by grouping multiple images into large HDF5 files, reducing the burden for each run. Therefore, optional conversion of CBF/ImageCIF files to HDF5/NeXus in *CBFlib* is under development (Bernstein *et al.* 2013). The hierarchical geometries presented here will be preserved, with

the added benefit that metadata only needs to be recorded once per complete dataset in an single HDF5 master file, as opposed to being repeated in thousands of separate CBF image files, each containing a full header description (see also the *Computational Crystallography Newsletter* companion article in this issue, "Coping with BIG DATA image formats: integration of CBF, NeXus and HDF5").

Conclusion

Integration of XFEL intensity data requires precise knowledge of where individual pixels are in physical space. Spot centroids are used for indexing, followed by crystal unit cell and orientation refinement. Correct refinement will predict spot locations such that integration masks will capture true signal while avoiding background. The ImageCIF/CBF representation we are implementing in *cctbx.xfel* for the CSPAD detector allows for simpler refinement of detector geometry, at the detector, quadrant, sensor and ASIC levels to sub-pixel accuracy. Incorporation into *dxtbx* enables straightforward access to detector and beam models, facilitating this refinement.

Acknowledgements

This work was supported by NIH grants GM095887 and GM102520 and Director, Office of Science, Department of Energy (DOE) under contract DE-AC02-05CH11231 for data-processing methods (N.K.S.).

References

Bernstein HJ and Hammersley AP (2005). Specification of the Crystallographic Binary File (CBF/imgCIF). *International Tables For Crystallography*. H. S. R. and M. B. Dordrecht, NL, Springer. **G:** 37-43.

Bernstein HJ, Sloan JM, Winter G, Richter TS, NeXus International Advisory Committee and Committee on the Maintenance of the CIF Standard (2013). "Coping with BIG DATA Image Formats: Integration of CBF, NeXus and HDF5." *poster, American Crystallographic Association, 2013 Annual Meeting*. Honolulu, HI.

Boutet S, Lomb L, Williams GJ, Barends TR, Aquila A, Doak RB, Weierstall U, DePonte DP, Steinbrener J, Shoeman RL, Messerschmidt M, Barty A, White TA, Kassemeyer S, Kirian RA, Seibert MM, Montanez PA, Kenney C, Herbst R, Hart P, Pines J, Haller G, Gruner SM, Philipp HT, Tate MW, Hromalik M, Koerner LJ, van Bakel N, Morse J, Ghonsalves W, Arnlund D, Bogan MJ, Caleman C, Fromme R, Hampton CY, Hunter MS, Johansson LC, Katona G, Kupitz C, Liang M, Martin AV, Nass K, Redecke L, Stellato F, Timneanu N, Wang D, Zatsepin NA, Schafer D, Defever J, Neutze R, Fromme P, Spence JC, Chapman HN and Schlichting I (2012). "High-resolution protein structure determination by serial femtosecond crystallography." *Science* **337**: 362-364.

Chapman HN, Fromme P, Barty A, White TA, Kirian RA, Aquila A, Hunter MS, Schulz J, DePonte DP, Weierstall U, Doak RB, Maia FR, Martin AV, Schlichting I, Lomb L, Coppola N, Shoeman RL, Epp SW, Hartmann R, Rolles D, Rudenko A, Foucar L, Kimmel N, Weidenspointner G, Holl P, Liang M, Barthelmess M, Caleman C, Boutet S, Bogan MJ, Krzywinski J, Bostedt C, Bajt S, Gumprecht L, Rudek B, Erk B, Schmidt C, Homke A, Reich C, Pietschner D, Struder L, Hauser G, Gorke H, Ullrich J, Herrmann S, Schaller G, Schopper F, Soltau H, Kuhnel KU, Messerschmidt M, Bozek JD, Hau-Riege SP, Frank M, Hampton CY, Sierra RG, Starodub D, Williams GJ, Hajdu J, Timneanu N, Seibert MM, Andreasson J, Rocker A, Jonsson O, Svenda M, Stern S, Nass K, Andritschke R, Schroter CD, Krasniqi F, Bott M, Schmidt KE, Wang X, Grotjohann I, Holton JM, Barends TR, Neutze R, Marchesini S, Fromme R, Schorb S, Rupp D, Adolph M, Gorkhover T, Andersson I, Hirsemann H, Potdevin G, Graafsma H, Nilsson B and Spence JC (2011). "Femtosecond X-ray protein nanocrystallography." *Nature* **470**: 73-77.

Ellis P and Bernstein H (2001). "CBFlib: An API for CBF/imgCIF Crystallographic Binary Files with ASCII Support.

Grosse-Kunstleve RW, Sauter NK, Moriarty NW and Adams PD (2002). "The Computational Crystallography Toolbox: crystallographic algorithms in a reusable software framework." *Journal of applied crystallography* **35**: 126-136.

Hart P, Boutet S, Carini G, Dubrovin M, Duda B, Fritz D, Haller G, Herbst R, Herrmann S, Kenney C, Kurita N, Lemke H, Messerschmidt M, Nordby M, Pines J, Schafer D, Swift M, Weaver M, Williams G, Zhu D, Van Bakel N and Morse J (2012). "The CSPAD megapixel x-ray camera at LCLS." *Proceedings of SPIE* **8504**: 85040C-85040C-85011.

Hattne J, Echols N, Tran R, Kern J, Gildea R, Brewster A, Alonso-Mori R, Glöckner C, Hellmich J, Laksmono H, Sierra R, Lassalle-Kaiser B, Lampe A, Han G, Gul S, DiFiore D, Milathianaki D, Fry A, Miahnahri A, White W, Schafer D, Seibert M, Koglin J, Sokaras D, Weng T, Sellberg J, Latimer M, Glatzel P, Zwart P, Grosse-Kunstleve R, Bogan M, Messerschmidt M, Williams G, Boutet S, Messinger J, Zouni A, Yano J, Bergmann U, Yachandra V, Adams P and Sauter N (submitted). "The accurate processing of diffraction data from X-ray free-electron lasers."

Kern J, Alonso-Mori R, Tran R, Hattne J, Gildea RJ, Echols N, Glockner C, Hellmich J, Laksmono H, Sierra RG, Lassalle-Kaiser B, Koroidov S, Lampe A, Han G, Gul S, Difiore D, Milathianaki D, Fry AR, Miahnahri A, Schafer DW, Messerschmidt M, Seibert MM, Koglin JE, Sokaras D, Weng TC, Sellberg J, Latimer MJ, Grosse-Kunstleve RW, Zwart PH, White WE, Glatzel P, Adams PD, Bogan MJ, Williams GJ, Boutet S, Messinger J, Zouni A, Sauter NK, Yachandra VK, Bergmann U and Yano J (2013). "Simultaneous femtosecond X-ray spectroscopy and diffraction of photosystem II at room temperature." *Science* **340**: 491-495.

Parkhurst J, Brewster A, Fuentes-Montero F, Waterman D, Hattne J, Ashton A, Echols N, Evans G, Sauter N and Winter G (in preparation). "dxtbx: the diraction experiment toolbox."

Sauter NK, Hattne J, Grosse-Kunstleve RW and Echols N (2013). "New Python-based methods for data processing." *Acta crystallographica. Section D, Biological crystallography* **69**: 1274-1282.

Waterman DG, Winter G, Parkhurst JM, Fuentes-Montero L, Hattne J, Brewster A, Sauter NK and Evans G (2013). "The DIALS framework for integration software." *CCP4 Newsletter on Protein Crystallography* **49**: 13-15.

Quantum Mechanics-based Refinement in Phenix/DivCon

Oleg Y. Borbulevych[a], Nigel W. Moriarty[b], Paul D. Adams[b,c] and Lance M. Westerhoff[a]

[a] *QuantumBio Inc, 2790 West College Ave, State College, PA, 16801, USA*

[b] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[c] *Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: lance@quantumbioinc.com

Introduction

Conventional macromolecular crystallographic refinement relies on *a priori* determined stereochemistry restraints and a simple restraint function to ensure the correct model geometry of the macromolecule along with any bound ligands, cofactors and metal coordination spheres. The benefit of this method is in its speed: entire structures can be refined and re-refined many times in a cycle in order to arrive at a structure the practitioner expects or trusts. However, unfortunately, as revealed in a recent survey of ligand structures deposited to PDB, on the order of 60% of all ligands within the PDB have questionable or wrong geometry (Gore *et al.*, 2011). This problem can often be traced to deficiencies or inaccuracies both in the ligand restraints and in the energy functional. In terms of the restraints, correct creation of an accurate file requires an *a priori* understanding of the ligand geometry within the active site. Traditionally, this is an error prone process, and tools such as *eLBOW* (Moriarty *et al.*, 2009) have helped immensely. However, even perfectly created restraints may not capture the influences of intermolecular covalent and non-bonded interactions, metal coordination and/or solvation. These influences are left to the energy functional to capture and compared with modern molecular and quantum mechanics methods, this energy functional is very simple in nature as it is missing electrostatics, polarization, hydrogen bonds, dispersion, *etc*.

In order to help address these deficiencies, we have integrated the semiempirical quantum mechanics (SE-QM) engine DivCon (Borbulevych et al., 2014) with the *Phenix* crystallographic package (Adams *et al.*, 2010). DivCon uses a fast, all-atom, linear-scaling, semiempirical quantum mechanics (SE-QM) method (Dixon & Merz, 1996, 1997, QuantumBio & Inc, 2011) to routinely characterize structures with thousands or even 10's (or 100's) of thousands of atoms. This *Phenix*/DivCon method - invoked using the `phenix.refine` (Afonine *et al.*, 2012) command line argument, `qblib=True` - does not rely on *a priori*-determined stereochemical restraints. Instead, it uses the same SE-QM Hamiltonians (AM1(Dewar *et al.*, 1985), PM3(Stewart, 1989)and PM6(Stewart, 2009, Hostas *et al.*, 2013)) available in advanced computational chemistry tools to calculate the gradients required to drive structure optimization and refinement. With this method, SE-QM gradients are calculated in "real-time" at each cycle of the LBFGS minimization for the entire structure or just the region(s) of interest. Since the region includes not only the ligand but the surrounding active site as well, the QM protocol also replaces link restraints and will model the geometry surrounding intermolecular covalent bonds, metal coordination spheres and so on.

Region QM Refinement

The DivCon package utilizes the linear scaling QM methodology that decreases the computational time as compared with conventional QM calculations. To optionally further speed-up the calculation, the command line argument `qblib_region_selection` has been added to carry out a region-specific QM-based refinement on the area(s) of interest within the protein/ligand complex (figure 1). All residues within a certain distance of any atom of the ligand are defined as a part of the main or core region (argument: `qblib_region_radius`). This core region is the region for which the atomic SE-QM gradients are determined and used at each step of the refinement. The second SE-QM region, referred to as a buffer region, includes any residues surrounding the core region (argument: `qblib_buffer_radius`). By using a buffer region to chemically insulate the core region, we gain a significant speed-up versus a full QM treatment, and at the same time, we limit any errors that may occur in the gradients due to capping or the artificial chemical environment surrounding the core region. In this case, the QM-gradients generated from the atoms within the buffer region

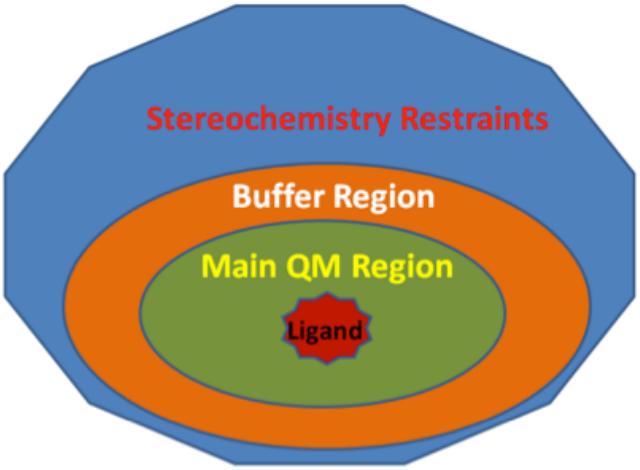


Figure 1. The schematic representation of the region QM refinement concept. The ligand and surrounding protein residues consists of the main QM region that is treated at the QM level as well as the buffer region. The rest of the protein is treated with the conventional stereochemistry restraints.

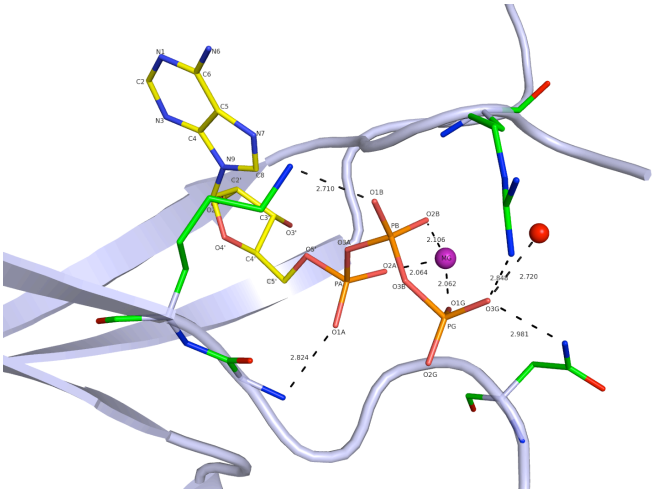


Figure 2. Intermolecular interactions of ATP (*yellow*) and in the protein structure 4AFF.

are not used in the refinement and the standard stereochemical restraints are used instead. This regional QM method can also be applied concurrently to more than one region within the complex.

For the examples presented here, the published atom placement for the ligand and for the protein is adopted at the beginning of the refinement. It should be noted that the QM method is not a "silver bullet," and the successful investigator will still need to protonate the structure and perform any initial atomic placement (by hand or through the use of experimental density-aware docking techniques). Also, while the restraints are provided to `phenix.refine` to satisfy its built-in error-trapping features, unless the `macro_cycle_to_skip` command line parameter is used to run conventional refinement prior to QM refinement, the restraints are not actually used in QM refinement. Therefore, any ligand(s) will need to be chemically correct upon initial placement. Here, just as in anywhere else, the concept of "garbage-in/garbage-out" is important.

Example #1: ATP geometry - QM refinement of PDB:4AFF

For the first example, adenosine triphosphate (ATP) is a cofactor found in numerous macromolecular structures and is therefore of significant interest to the community. PDBid:4AFF contains the ATP coordinated to the Mg^{2++} (figure 2). The default PM6 Hamiltonian (Stewart, 2009, Hostas *et al.*, 2013), which has published support for 70 elements, is used in this refinement.

In order to trigger the addition of both *Phenix* and DivCon to your `$PATH`, both the `phenix_env` and `qbenv.csh` file (`phenix_env.sh` & `qbenv.sh` for bash) will need to be sourced as per scheme 1 (assuming the csh shell is used).

Once this initial atom placement and chemical connectivity has been adopted, QuantumBio provides a `qbphenix` Perl script that can be used to prepare the PDB file and run `phenix.refine` as required. This script is configured to run either the *Phenix ReadySet!* protonation or the Protonate3D protonation found in MOE from the Chemical Computing Group, Inc. (CCG) depending upon availability. For the example in scheme 2, the *ReadySet!* tool will be used.

4AFF Re-refinement Results

The region QM refinement of 4AFF centered around ATP in *Phenix* yielded R_{work}=0.175 and R_{free}=0.181. Despite the good 2Fo-Fc density in the region of ATP we find that there are several noticeable differences in the QM refined geometry

```
% source /path/to/phenix-dev-1555-or-newer/phenix_env
% source /path/to/DivConDiscoverySuite-b####/etc/qbenv.csh
```

Scheme 1: Environment setup commands for Phenix/DivCon

```
% qbphenix --pdbFile 4AFF.pdb --sfFile 4AFF.mtz --chain A          \
     --resname ATP --resid 1117 --region-radius 4.0                \
     --buffer-radius 3.0 --protonation ReadySet
% phenix.refine 4AFF.pdb 4AFF.mtz 4AFF.cif qblib=True              \
     qblib_region_radius=4.0 qblib_buffer_radius=3.0               \
     qblib_region_selection="chain A and resname ATP and resid 1117"
```

Scheme 2. Commands to run DivCon refinement in Phenix using the 4AFF example.

Table 1. Selected geometry parameters in ATP.

Parameter	QM refined	Phenix Restraint Value	Small Molecule Crystal Structure[±]
PA-O3A-PB	126	120.5	125(1)-129.3(2)
O3A-PB-O3B	105	108.2	101.9(2)-104(1)
PB-O3B-PG	128	120.5	127(1)-130.5(2)
PA-PB-PG	86	-	84.6(3)-85.4(7)

compared with the Monomer Library (v4.3) values for ATP (table 1).

While it is expected that the SE-QM method and the simplified force field in *Phenix* would provide different results, initially this difference was flagged as significant. Tamasi *et al.* (Tamasi *et al.*, 2010) performed crystallographic studies on a number of small molecule systems with ATP and demonstrated that crystalline water molecules as well as counterions affect the molecule geometry of ATP. In 4AFF, the 3-phosphate moiety of ATP forms numerous H-bonds with protein residues as well as is involved in the coordination with magnesium and one water molecule (figure 2). When comparing these results with those observed in the QM-based refinement, we find that the geometry parameters in question are very close to the corresponding experimental data (table 1). Conventional restraint targets on the other hand deviate significantly from the experimental values listed in table 1 suggesting that in this case, *a priori* prediction of the molecular geometry can be a challenging task especially when non-bonded interactions significantly influence the ligand/co-factor geometry. In order to capture this sort of influence in conventional refinement, the practitioner would need to know enough about the bound state and manipulate the restraints in order to mimic the chemistry that is automatically captured using SE-QM.

Example #2: Covalently Bound Ligand - QM refinement of PDB:2V6N

Suicide inhibitors represent a traditional challenge for conventional refinement (Kleywegt, 2007). Such ligands are covalently bound to certain amino acids (*e.g.* SER or LYS) thus becoming a part of the polypeptide chain. This bond makes it difficult to choose restraints for the chemically modified amino acid *and* the ligand. Furthermore, structural changes of the ligand (and amino acid) due to this bond can "trickle down" through the molecule affecting adjacent bond angles/lengths/etc.

The standard approach to a bound ligand in *Phenix* is to use the `phenix.ligand_linking` program to generate the two files required by `phenix.refine` to add the bonds and angles to the restraints model. The bond values used are from QM calculations, however, the angles are not as precisely determined.

For this refinement, perhaps surprisingly, there is no significant difference between `phenix.refine` command on the non-covalently bound ligand and the covalently bound ligand (see scheme 3). In QM, there is no concept of "explicitly defined" covalent bonds. As before, *ReadySe*t! was chosen for this protonation method in order to simplify the use of the *Phenix* suite. MOE could have been used as well (`--protonation` **`MOE`**).

```
% qbphenix --pdbFile 2V6N.pdb --sfFile 2V6N.mtz --chain A          \
     --resname XP1 --resid 2307 --region-radius 3.0                \
     --buffer-radius 2.5 --protonation ReadySet
% phenix.refine 2V6N.pdb 2V6N.mtz 2V6N.cif                         \
     refinement.input.xray_data.labels=FP,SIGFP qblib=True         \
     qblib_region_radius=3.0 qblib_buffer_radius=2.5               \
     qblib_region_selection="chain A and resname XP1 and resid 2307"
```

Scheme 3. Commands to run DivCon refinement in Phenix using the 2V6N example.

Table 2. Selected bond lengths (Å) and bond angles (°) in the structure 2V6N.

Parameters	QM Refinement	*Phenix* Refinement	Original PDB
CAC XP1- CAD XP1	1.46	1.48	1.52
SG Cys145-CAC XP1	1.74	1.81	1.83
CB Cys145-SG Cys145-CAC XP1	101	101	91
SG Cys145-CAC XP1- OAH XP1	123	118	127
SG Cys145-CAC XP1- CAD XP1	115	116	111

2V6N Re-refinement Results

The structure 2V6N determined at 1.98 Å resolution has revealed the SARS coronavirus main proteinase inactivated by the covalently bound ligand 4-dimethylaminobenzoic acid (XP1). The covalent bond is made between the carbonyl carbon of the ligand and the sulfur atom of CYS 145 (table 2).

The deposited structure exhibits several geometry distortions in the region of the linkage bond between the ligand and Cys145. In particular, the linkage bond S-C (1.83 Å) is longer than the normal C_{sp2}-S bond (1.75 Å) (Allen *et al.*, 1987). Notably, the C-S-C bond angle is 91°, which is much smaller than the expected value between 99-109° depending on the chemistry (Shigeru & Joyce, 1991).

The re-refined structure using the standard *Phenix* procedure described earlier produces a linkage bond of 1.81 Å despite the ideal value being 1.78 Å. There is also a marked deviation from the ideal restraint value for the C-S-C angle. The model value is 101° while the ideal value was tetrahedral. This demonstrates that the density influences the final structure strongly.

The key advantage of the QM refinement that *no geometry restraints* for the ligand and the surrounding active site, including the linkage C-S bond, are needed. QM refinement leads to correct geometry of the ligand and the ester bond and the structure is completely fixed relative to the original PDB model. Notably, the C-S-C angle became 101° that is within the acceptable range. Figure 3 depicts the refined structure vs. the originally deposited structure.

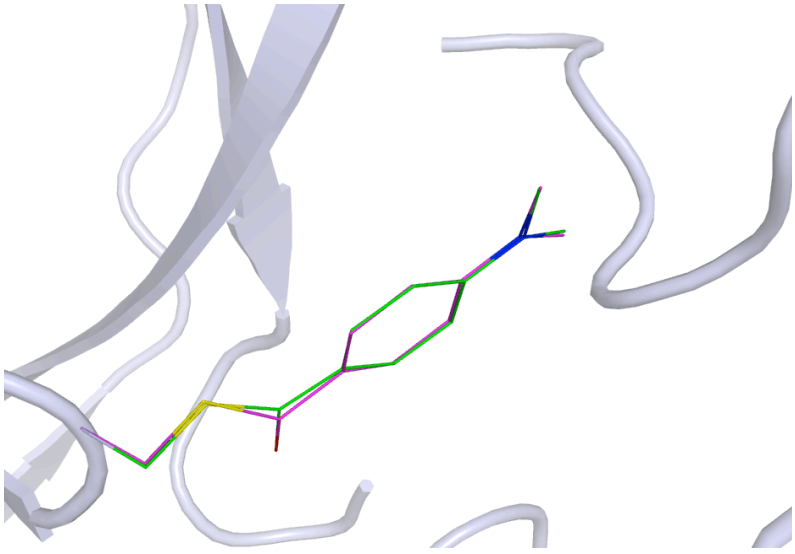


Figure 2. Superimposition of the QM re-refined PDB structure 2V6N (green) with the original PDB (magenta).

Discussion & Further information

X-ray crystallography is a crucial tool in the modern, industrial and academic structure based drug discovery toolbox, and refined crystal structures often form the basis of core discovery research projects. Unfortunately, with conventional methods based on *a priori* restraints and simple functions, it is not unusual for investigators to download a crystal structure of interest, add protons, and re-optimize the once carefully refined heavy atoms positions using any number of different molecular mechanics force fields (e.g. AMBERFF, MMFF, etc). With the *Phenix*/DivCon integration, it is hoped that this practice can become less prevalent as resulting models adopt structures that are chemically correct at the outset since these technologies are the same methods that are used in some of the most advanced computational chemistry and molecular modeling approaches available. Further, since *Phenix*/DivCon actually uses QM to treat the target:ligand complex together during the refinement, this method will be able to capture the interactions between the various species and ultimately provide greater insight into the inner workings of the active site. The following is a list of websites that provide additional information on the use and access of these technologies:

- Usage Tutorial: http://www.quantumbioinc.com/support/manual-phenixdc/tutorial
- Performance Guidelines: http://www.quantumbioinc.com/support/manual-phenixdc/guidelines
- FAQ: http://www.quantumbioinc.com/support/manual-phenixdc/phenix_divcon_faq
- Publications: http://www.quantumbioinc.com/publications?tag=xray
- Licensing Information: http://www.quantumbioinc.com/products/software_licensing

References

Adams, P. D., Afonine, P. V., Bunkoczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L. W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). Acta Crystallographica Section D-Biological Crystallography 66, 213-221.

Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H. & Adams, P. D. (2012). Acta Crystallographica Section D 68, 352-367.

Allen, F. H., Kennard, O., Watson, D. G., Brammer, L., Orpen, A. G. & Taylor, R. (1987). Journal of the Chemical Society-Perkin Transactions 2 S1-S19.

Borbulevych, O. Y., Plumley, J. A., Martin, R. M., Merz, K. M. & Westerhoff, L. M. (2014). Acta Crystallographica Section D-Biological Crystallography, in press.

Dewar, M. J. S., Zoebisch, E. G., Healy, E. F. & Stewart, J. J. P. (1985). Journal of the American Chemical Society 107, 3902-3909.

Dixon, S. L. & Merz, K. M. (1996). Journal of Chemical Physics 104, 6643-6649.

Dixon, S. L. & Merz, K. M. (1997). Journal of Chemical Physics 107, 879-893.

Gore, S., Tjelvar, S., Olsson, G. & Zhuravleva, M. (2011). Acta Crystallographica Section A: Foundations of Crystallography A67, C104.

Hostas, J., Rezac, J. & Hobza, P. (2013). Chem. Phys. Lett. 568-569, 161-166.

Kleywegt, G. J. (2007). Acta Crystallographica Section D-Biological Crystallography 63, 94-100.

Moriarty, N. W., Grosse-Kunstleve, R. W. & Adams, P. D. (2009). Acta Crystallographica Section D-Biological Crystallography 65, 1074-1080.

QuantumBio & Inc (2011). LibQB. Version 5.0, www.quantumbioinc.com.

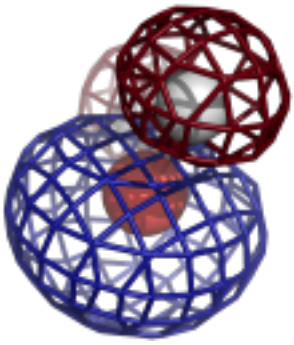
Shigeru, O. & Joyce, D. (1991). Organic Sulfur Chemistry.

Stewart, J. J. P. (1989). Journal of Computational Chemistry 10, 209-220.

Stewart, J. J. P. (2009). Journal of Molecular Modeling 15, 765-805.

Tamasi, G., Berrettini, F., Hursthouse, M. B. & Cini, R. (2010). The Open Crystallography Journal 3, 1-13.

VOLUME FIVE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXIV

FAB ELBOW, MAP CONNECTIVITY, CDL

Table of Contents

- PHENIX News 31
- Crystallographic meetings 32
- Expert Advice
  - Fitting tips #8 – Acetyl Groups are like Peptides: Planar and *trans* 32
- Short Communications
  - Connectivity analysis tools in *CCTBX* 35
  - phenix.fab_elbow_angle: Fragment Antigen-Binding elbow angle calculation tool 39
- Articles
  - Details of the Conformation-Dependent Library 43

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

Guided Ligand Replacement

Adding ligands to an *apo* structure can now be performed using a similar ligand fit into a similar protein. The guiding protein-ligand complex can be sourced from the RCSB via web services, from a local file directory or supplied by the user.

New features

Conformation Dependent Library

The CDL has been added to *Phenix* and tested extensively. High resolution structures were used to determine restraints of the protein backbone based on the φ,ψ torsion angles. The parameter `cdl=True` activates its use. More details can found in the article on page 42 of this issue.

Automatic linking

All programs based on model geometry in *Phenix* (e.g. `phenix.real_space_refine`, `phenix.geometry_minimization` and `phenix.refine`) have moved to automatic determination of metal coordination and covalent bonding. Currently not the default, the parameter `link_all=True` will activate the linking based on residue type and distance cutoff. Finer control can be achieved by using the linking type switches (`link_metals`, `link_rna_dna`, `link_residues` and `link_carbohydrates`) and the corresponding distance cutoff parameters. The procedure covers covalently linked ligands and does simple validation of the carbohydrate polymers.

Crystallographic meetings and workshops

5th Murnau conference on Structural Biology – Focus Topic: Signal Transduction
Sept 10-13, 2014

Location: Murnau am Staffelsee, Germany
www.murnauconference.de/2014/index.html

Expert advice

Fitting Tip #8 – Acetyl Groups are like Peptides: Planar and *trans*

Jane Richardson, Jeff Headd, and Nigel W. Moriarty,
Duke University, J&J & LBNL

Acetylation is quite a common protein modification. Disturbingly, however, a recent study (Genshaft 2013) found that even for crystal structures involving the histone acetyl-lysine (3-letter code "ALY") so important in control of gene expression over 25% were modeled with extremely implausible high-energy conformations. (If it makes you feel better, over 50% were implausible in NMR models.) Clearly our automated tools have some catching up. But in the meantime it is very easy for you to apply user expertise: as we will show, acetyl groups have the same chemistry and conformational limitations as the familiar peptide bond, so just make sure they are planar and *trans* in your deposited structure. Figure 1 shows front and side views of an acetyl group at 1.2Å resolution, where the density clearly distinguishes between the carbon and the oxygen. The *trans* dihedral between the Cα-analog atoms is marked by a green line. Note that the N–H and C–O bonds in the peptide plane are also *trans* to each other and the planarity is quite exact.

Acetyl-lysine

For 109 acetyl examples in the Cambridge Structural Database (CSD; Allen 2002) of high-resolution small molecule crystal structures, every one was planar and *trans* (ω = 180° +/- 4°; Genshaft 2013). This agrees with energy calculations for *cis* versus *trans* acetyl and the high barrier between them, and the same regularity is seen for well-ordered high-resolution examples in protein crystals, such as the figures here. Genshaft et al. saw no significant correlation of acetyl-lysine error rate with refinement software, or even with resolution (since high-B examples are not adequately constrained by density even at high resolution). However in our informal survey, most multi-acetyl protein structures were either nearly random or all correct. Presumably this means that so far none of the automated software gets acetyls right reliably, but many crystallographers do understand acetyl geometry.

Acetyl N-termini

Acetyl N-termini ("ACE") are not usually as functionally important as acetyl-lysines, but they are an order of magnitude more common. Figure 2 shows map and model for a 0.8-Å resolution acetyl N-terminus, where the analogy with a standard peptide linkage is even more obvious than in figure 1. For crystal structures with a modeled ACE group, Genshaft et al. found 17% with highly strained conformations.

Confirmation by all-atom contacts

When there are other atoms nearby, then looking at the all-atom contacts for H-bonds and clashes is another very helpful diagnostic, or confirmation, of the correct orientation. Within the validation section of the Phenix GUI (Echols 2012), or on the MolProbity web site (Chen 2010), you can do this

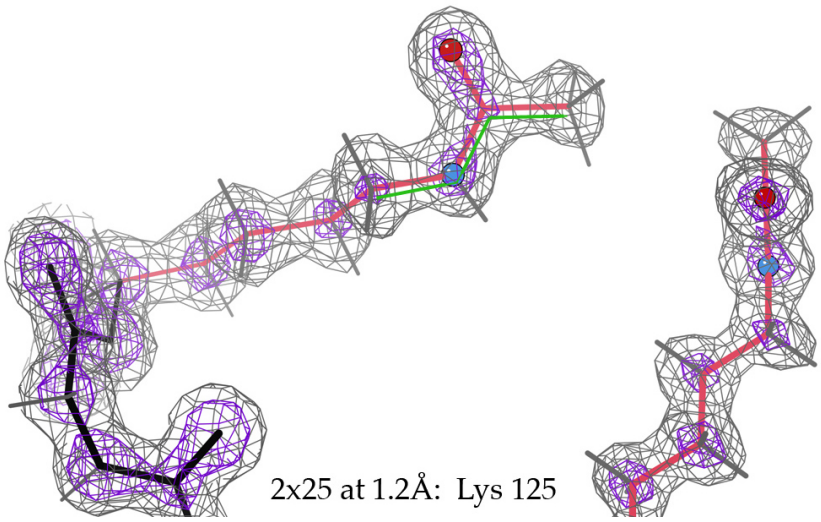


Figure 1: An N-acetyl-lysine at high resolution: (A) front view, showing that the C–N–C-C dihedral (green line) is trans, confirmed by higher density at the O than at the methyl; (B) side view, to show the tight planarity (2x25; Lammers 2010)

easily by launching the multi-criterion kinemage in KiNG (Chen 2009). For the somewhat-related case of non-covalent acetate ligands ("ACT"), all-atom contacts are the best way of correctly fitting the triangular electron density at mid resolutions (see figure 3).

Issues in refinement

In response to this problem, we have re-examined the handling of acetyl restraints in Phenix, which historically has produced almost no highly twisted cases (defined in Genshaft 2013 as >30° from planarity, and occurring from other programs), but a fair number of *cis* (0° +/- 30°). This is because Phenix has always had a restraint to planarity. However, since early 2013 it's had a strong term preferring *trans* for both ALY and the link to ACE, but no way to fix an initial fit of *cis*. We are working on a system that would check for highly strained torsions, including acetyls, and try flipping them.

Conclusion

Acetyl groups on lysine side-chains or on N-termini are fairly often modeled wrong, either by people or by software. But all you need to remember to get them right is that they have the same chemistry and conformational properties as a peptide -- so, as you know, they are very close to planar and essentially always *trans*.

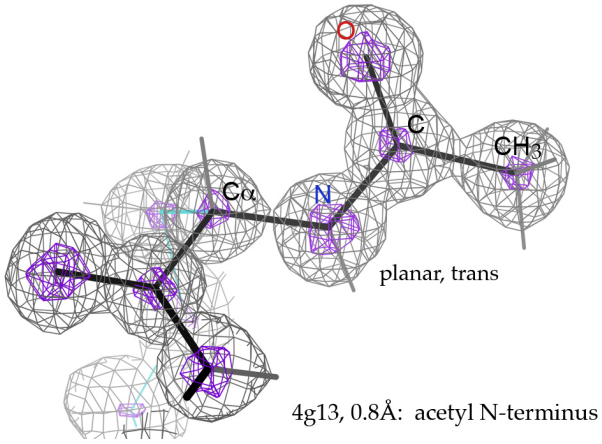


Figure 2: An acetyl N-terminus at high resolution, model and electron density (4g13; Gessmann 2012).

References

Allen FH (2002) "The Cambridge Structural Database: a quarter of a million crystal structures and rising", Acta Crystallogr. B58: 380-388

Chen VB, Davis IW, Richardson DC (2009) "KiNG (Kinemage, Next Generation: A versatile interactive molecular and scientific visualization program", Protein Sci 18:2403-240

Chen VB, Arendall WB III, Headd JJ, Keedy DA, Immormino RM, Kapral GJ, Murray LW, Richardson JS, Richardson DC (2010) "MolProbity: all-atom structure validation for macromolecular crystallography", Acta Crystallogr D66:12-21

Echols N, Grosse-Kunstleve RW, Afonine PV, Bunkoczi G, Chen VB, Headd JJ, McCoy AJ, Moriarty NW, Read RJ, Richardson DC, Richardson JS, Terwilliger TC, Adams PD (2012) "Graphical tools for macromolecular crystallography in Phenix", *J Applied Crystallogr* **45**: 581-586

Genshaft A, Moser JA, D'Antonio EL, Bowman CM, Christiansen DW (2013) "Energetically unfavorable amide conformations for N6-acetyllysine side chains in refined protein structures", *Proteins: Struct Funct Bioinf* **81**: 1051-1057

Gessmann R, Axford D, Evans G, Bruckner H, Petratos K (2012) "The crystal structure of samarosporin at atomic resolution", J Pept Sci 18: 678-684

Lammers M, Neumann H, Chin JW, James LC (2010) "Acetylation regulates cyclophilin A catalysis, immunosuppression, and HIV isomerization", *Nat Chem Biol* **6**: 331-337

Symersky J, Li S, Carson M, Luo M (2003) "Structural genomics of Caenorhabditis elegans: triosephosphate isomerase", *Proteins: Struct Funct Bioinf* **51**: 484-486

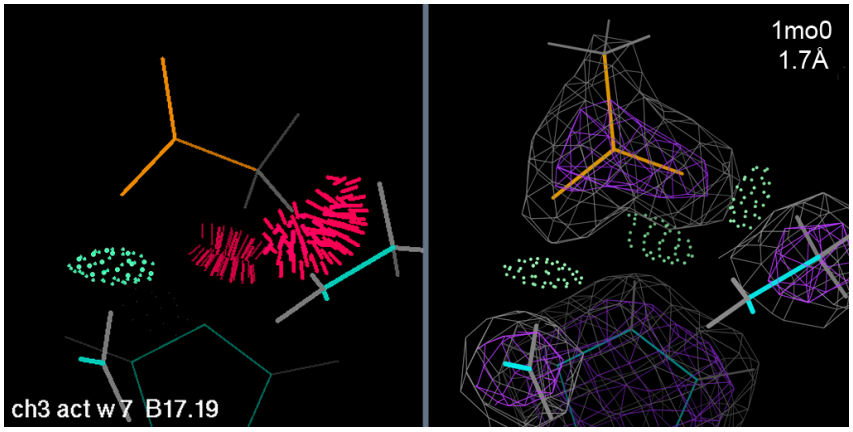


Figure 3: A solvent acetate, all-atom contacts before and after refitting (1mo0; Symersky 2003).

Connectivity analysis tools in *CCTBX*

Oleg V. Sobolev[a], Pavel V. Afonine[a] and Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: OSobolev@lbl.gov

Introduction

Connected regions in crystallographic maps defined at a given contouring level are an important feature of the map and map analysis. Studies of this feature may have various uses in crystallography: from *ab initio* phasing procedures (Lunin *et. al.,* 2000), various methods of map improvement and interpretation to analysis of connectivity of reflection data in reciprocal space (Urzhumtseva & Urzhumtsev, 2011). Robust implementation of connectivity analysis algorithm (Lunina *et al.,* 2003) in *CCTBX* (Grosse-Kunstleve *et al.* 2002) is presented.

A crystallographic map is a three-dimensional array of numbers representing the contents of a unit cell or a smaller region (asymmetric unit or a box). These, for instance, may be a distribution of electron or nuclear density, Fourier maps, solvent or other masks. One can contour the map at a particular level and study the regions that are above this level. This is the usual way for Fourier map interpretation during manual model building procedures. One can think of connected regions as peaks (also sometimes referred to as 'blobs') in a map. In the present work we report the implementation of an established connectivity analysis algorithm and extend it to determine the volume of all connected regions, coordinates, and the value of the maximum point for each region.

Implementation

The connectivity algorithm is implemented as a C++ module, `maptbx/connectivity.h`, in *CCTBX* as a separate class. Corresponding functionality is available from Python via the boost.python bindings (Abrahams & Grosse-Kunstleve, 2003) in `maptbx/boost_python/maptbx_ext.cpp`.

The runtime of the algorithm linearly scales with the number of grid points of the map. Typical runtimes are 1.85 seconds for a large map of size 400*400*400 points and 0.03 seconds for medium map of size 100*100*100 points. Calculation of the region volume, values and coordinates of maximum point in each region during the analysis can be performed at no significant additional computational cost.

The definition of connected region is based on the definition of neighbors for a particular point. In the present implementation we use six neighbors for a point with `(x,y,z)` coordinates: `(x-1,y,z)`, `(x+1,y,z)`, `(x,y-1,z)`, `(x,y+1,z)`, `(x,y,z-1)`, `(x,y,z+1)`. Alternative definitions of neighbor points may use 18 neighbor points (varying 2 of 3 coordinates) or 26 neighbor points (varying all 3 coordinates). We believe that implementation of 18- or 26-point neighbor scheme is unnecessary at this point.

Examples

1. Basics

The following code snippet illustrates the basic manipulations with the connectivity object (`co` in what follows): analysis of connected regions on the map and obtaining supplementary data (volumes, coordinates of maxima and values of maxima). To instantiate a connectivity object one needs to pass the map data and the threshold level to its constructor (see schema 1). In this example we obtained four arrays from connectivity object:

- `result_of_connectivity_analysis` – a 3D integer array of the same dimension as `map_data` filled with numbers 0, 1, 2, … N, that enumerate the N connected regions. Each grid point contains 0 if the input map values

```
>>> co = maptbx.connectivity(map_data=map_data, threshold=100)
>>> result_of_connectivity_analysis = co.result()
>>> region_volumes = list(co.regions())
>>> print region_volumes
[975696, 12152, 12152]
>>> coordinates_of_maximum_points = list(co.maximum_coors())
>>> print coordinates_of_maximum_points
[(74, 62, 62), (20, 20, 20), (60, 60, 60)]
>>> values_of_maximum_points = list(co.maximum_values())
>>> print values_of_maximum_points
[99.8855747054916, 1569.9055625594979, 1569.9055625594979]
```

Schema 1: Initialization and basic commands of connectivity analysis object.

```
>>> co = maptbx.connectivity(map_data=cmap, threshold=5)
>>> resulting_mask = co.volume_cutoff_mask(volume_cutoff=10)
```

Schema 2: Commands to obtain mask for filtering out peaks with volume smaller than 10 on a threshold equals 5.

are less than the threshold value and a non-zero value for connected regions. Each connected region is assigned its own unique integer number.

- `region_volumes` – 1D integer array of length N+1. Each element, *i*, contains the volume of *i*-th region. The zeroth element contains the volume of the under-threshold region.
- `coordinates_of_maximum_points` – 1D array of length N+1 of tuples. Similarly to volumes, contains the coordinates of maximum point.
- `values_of_maximum_points` – 1D float array of length N+1, contains the values of corresponding maximum points.

As noted previously the input data to the connectivity analysis procedure is a three-dimensional array with numbers. Therefore coordinates of the maximum points are provided as array indexes of these grid points. The volume of the region is the number of grid points with values greater than the specified threshold. The `result_of_connectivity_analysis` may be used further to obtain various kinds of masks.

2. Volume cutoff

An example of useful application of connectivity analysis procedure is to eliminate small map peaks at a particular threshold level (Afonine *et al.*, 2014, in preparation) is illustrated in schema 2. Here we obtain the `co` object the same way as in the first example, and then call the `volume_cutoff_mask` method with the `volume_cutoff` parameter to provide the smallest size of peaks that should be kept in the `resulting_mask`. A binary 3D mask with 0 where the value in `cmap` are less than the threshold and where the volume of connected regions are less or equal to `volume_cutoff` and 1 everywhere else. It should be stressed that in this case we obtain a binary mask of 0 and 1 although the connectivity analysis results are still available via `co.result()`. Finally to apply the filtration to the data one has to multiply map data `cmap` by `resulting_mask`.

3. Noise elimination based on analysis of two cutoff levels.

A more elaborate noise elimination procedure is used in the "Feature enhanced map" tool implemented in *Phenix* package (Adams *et al.*, 2010) as the *phenix.fem* program (Afonine et al, 2014, in preparation) that analyzes connectivity regions at two cutoff levels simultaneously.

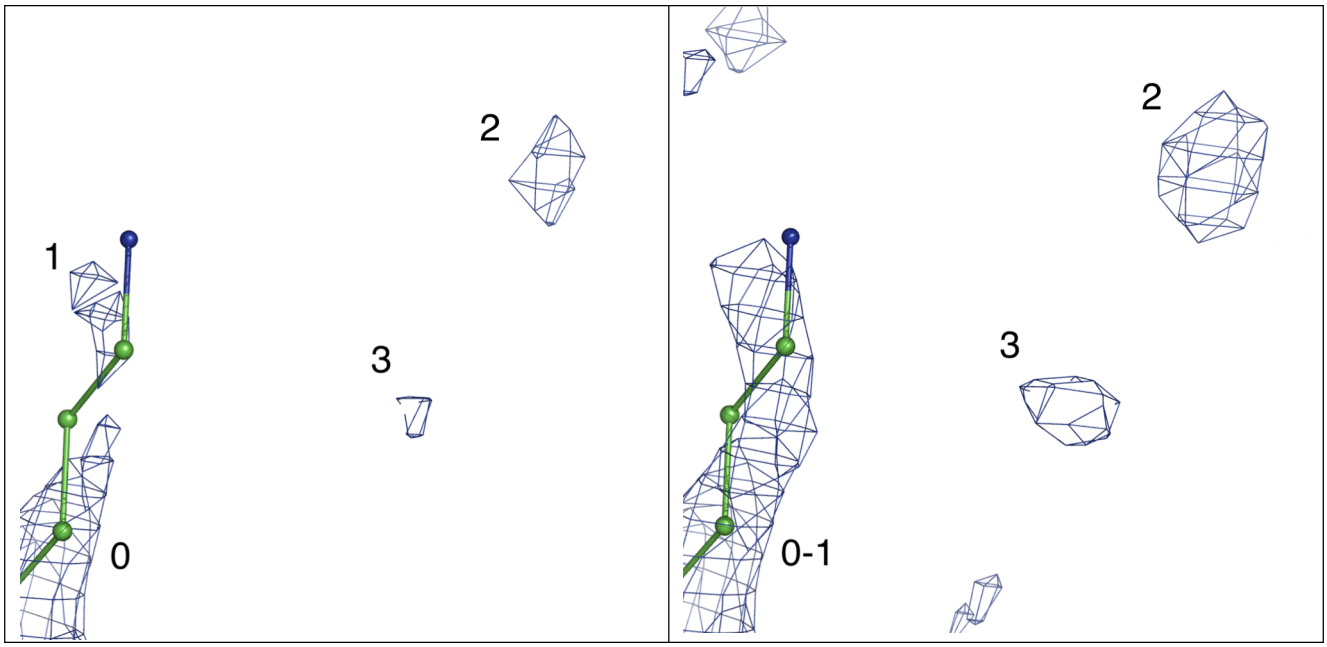


Figure 1: Schematic illustration of noise peaks elimination. Small-volume peaks at threshold *t* (left; peaks 1,2,3) and *t*-δ (right; peaks 2 and 3). Peaks 2 and 3 will be eliminated.

```
>>> co1 = maptbx.connectivity(map_data=cmap, threshold=25)
>>> co2 = maptbx.connectivity(map_data=cmap, threshold=22)
>>> resulting_mask = co2.noise_elimination_two_cutoffs(
      connectivity_object_at_t1=co1,
      elimination_volume_threshold_at_t1=12,
      zero_all_interblob_region=True)
```

Schema 3: Commands to obtain mask for noise filtering using two cutoffs and volume estimation of good peaks on the first cutoff.

Map peaks that have smaller volume than the typical volume of a reliably placed atom (e.g. water) at some threshold level *t* can be considered as noise peaks and ideally should be eliminated. Moreover, removing them at a lower level, *t*-δ, could be even better because more noise peaks would be removed. Nevertheless at the *t*-δ cutoff level some peaks previously scheduled for deletion could merge with good peaks. Such peaks should be retained. This procedure is schematically illustrated in figure 1. Figure 1(a) shows peaks 0,1,2,3 at the *t* cutoff level. Naïvely, peaks 1,2,3 could be deleted based on their small volume. However, figure 1(b) shows the same map on *t*-δ cutoff level where peak 1 has merged with molecule-related (not scheduled for deletion) peak 0 and should therefore be retained. Since peaks 2 and 3 did not merge with any region not scheduled for deletion, they will be zeroed using the shape as they appear at *t*-δ level. Particular threshold levels can be determined based on a trial-and-error approach to achieve best performance of the procedure. Information between peaks below the *t*-δ threshold level may be zeroed or preserved at the values in the original map. The code to do this filtration is shown in schema 3.

Here we created two connectivity objects with the same map data but with different threshold levels for determining connected regions. Then we call the `noise_elimination_two_cutoffs` function of the connectivity object with the lower threshold, provide the first connectivity object (with the higher threshold), volume of blobs that should be considered as good on

threshold t and a boolean parameter to specify whether to keep (`zero_all_interblob_region =False`) or mask out (`zero_all_interblob_region=True`) data between peaks at the t-δ threshold. The result is a 3D mask with the same dimensions as the original map. It is filled with 0 for map points that should be deleted and 1 for map points that should be retained. To apply the filtration one need to multiply the initial map data `cmap` by `resulting_mask`.

Conclusion

We have implemented a fast and reliable search of connected regions along with determination of their volume, values and position of maximum point at an arbitrary cutoff level. This module could provide a strong foundation for the development of various map-processing algorithms. The tools described here are available as a part of *CCTBX*.

References

Abrahams D. and Grosse-Kunstleve R.W., C/C++ Users Journal, July, 2003.

Adams, P. D., Afonine, P. V., Bunkóczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L.-W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. and Zwart, P. H. (2010). *Acta Cryst.* D**66**, 213-221.

Grosse-Kunstleve, R.W., Sauter, N.K., Moriarty, N.W. and Adams, P.D. (2002). *J. Appl. Cryst.* **35**, 126-136.

Lunin, V. Yu., Lunina, N. L. & Urzhumtsev, A. G. (2000). Acta Cryst. A**56**, 375-382.

Lunina, N.L., Lunin, V.Y. & Urzhumtsev, A. (2003). Acta Cryst. D**59**. 1702-1755.

Urzhumtseva, L. & Urzhumtsev, A. (2011). *J. Appl. Cryst.* **44**. 865-872.

phenix.fab_elbow_angle: Fragment Antigen-Binding elbow angle calculation tool

Youval Dar[a], Pavel V. Afonine[a], Nigel W. Moriarty[a] and Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: ydar@lbl.gov

Introduction

Antibody (Ab) used by white blood cell and the immune system to identify and fight antigens (Antibody generators) such as viruses and bacteria. They are large Y-shape proteins (figure 1). Fragment Antigen-Binding (Fab) are the tips of the Y-shape Ab allowing only a specific antigen to bind. The Fab is cleaved from the complete Ab to allow studying of the active portion without interference from other portions of the molecule. The Fab are composed of heavy and light chains, each with a constant and a variable regions. In a schematic illustration (figure 1.A) the constant and variable portions of the Fab are aligned at angle 180°. Figure 1.B shows a common situation, where the angle is not 180°. The angle between the variable and constant regions is called the Fab elbow angle. To find that angle, the light chain of each region is aligned onto the heavy chain by a rotation. The corresponding rotation axes are pseudo-dyad axes (imperfect dyad symmetry axes) and the Fab elbow angle is defined as the angle between these two axes. The elbow angle is an important characteristics of this class of protein structures and is typically reported in corresponding structural reports (Stanfield *et al.*, 2006). We have added a function to *CCTBX* (Grosse-Kunstleve *et al.*, 2002) that calculates Fab elbow angle.

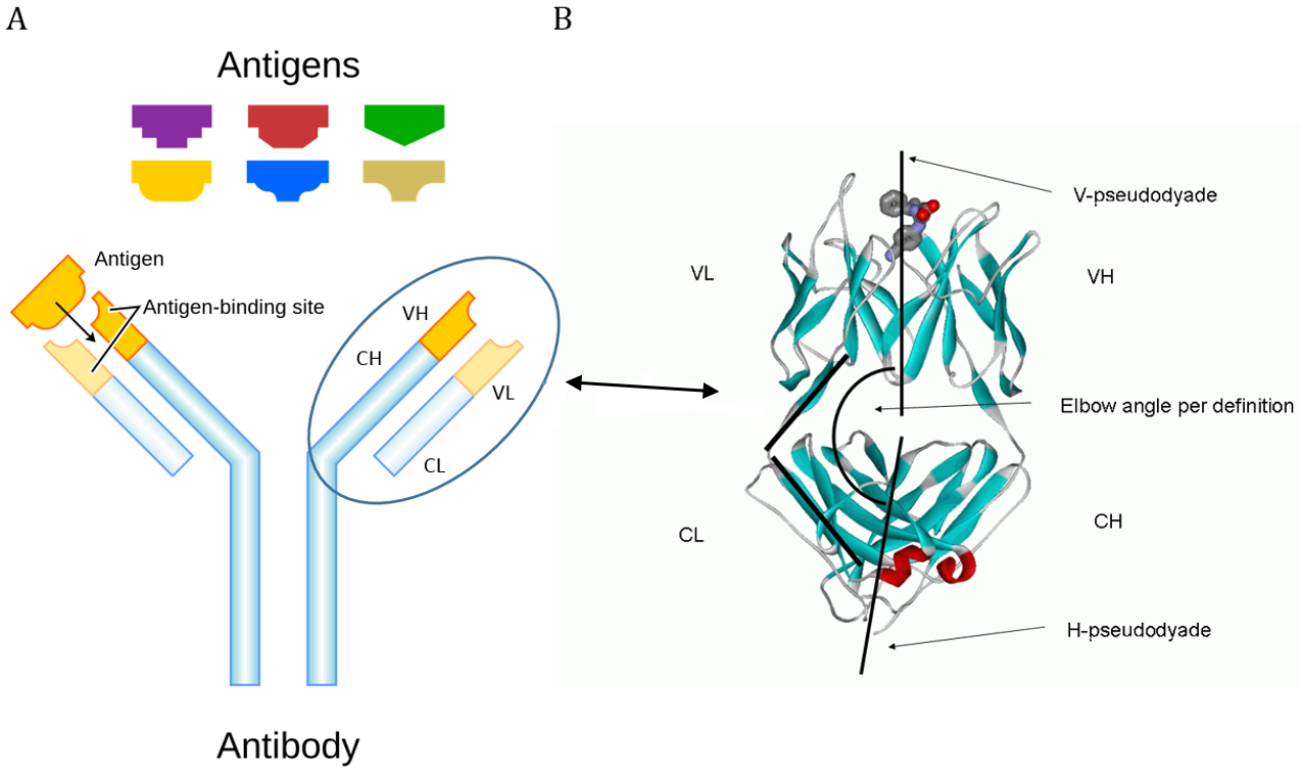


Figure 1: (A) Antigen, Wikipedia, Antigen, http://upload.wikimedia.org/wikipedia/commons/2/2d/Antibody.svg. By Fvasconcellos 19:03, 6 May 2007 (UTC) [Public domain], via Wikimedia Commons. (B) Fab angle from (Zemla A. 2005) (Web server addition). VL – Variable Light, VH – Variable Heavy, CL – Constant Light, CH – Constant Heavy.

Since the Fab is not the complete Ab its orientation with respect to the rest of the Ab is unknown. As a result, the angle between the two pseudo-dyad axes does not define a unique orientation between the variable portion of the Fab and rest of the Ab. This causes an ambiguity in the definition and interpretation of the Fab elbow angle. Factors affecting the Fab elbow angle determination are:

- Choice of limit residues (residues separating the variable and constant domains of the heavy and light chains).
- Selection of residues to align.

We used two methods to resolve the angle ambiguity in a way that matches published work (Stanfield *et al.*, 2006) and a similar function available in PyMol (Schrodinger, 2010; DeLano, 2002). The first method uses a known Fab structure as a reference while the second uses only the pseudo-dyad axes to determine the angle. Preferring not to rely on a particular structure as a reference, we have incorporated only the second method to *CCTBX*.

Method

Following (Stanfield *et al.*, 2006) we consider only the two dimensional ambiguity in elbow angle calculation, namely checking if the resulting angle is α or $(360-\alpha)$.

The program steps for finding the pseudo-dyad axes and the angle between them are as follows:

- Using *CCTBX* tools we get the rotation matrices that superpose light segments onto heavy ones.
- The atoms selected to be superposed are $C\alpha, C$ and N that do not have alternative locations. The selection strings for each are listed:
  - For the variable segment:
    chain *chain_ID* and resseq *1:Limit* and pepnames and (name ca or name n or name c) and altloc " "
  - For the constant segment:
    chain *chain_ID* and resseq *Limit+1:end* and pepnames and (name ca or name n or name c) and altloc " "
- The normalized eigenvectors, corresponding to eigenvalues of unity of those rotation matrices, represent the pseudo-dyad axes of the variable and the constant domains, $\hat{V}$ and $\hat{C}$ (Figure .B).
- $\cos(\alpha) = \hat{V} \cdot \hat{C}$ (α can be either between 90° and 180°, or between 180° and 270°).

Known protein as reference

Following the implementation described in (Stanfield *et al.*, 2006), we used PDB (Berman *et al.*, 2000; Bernstein *et al.*, 1977) protein *1bbd* as a reference using this procedure:

- Align Fab constant-heavy segments of a *1bbd* with the tested protein.
- Use the method described above and find the angle β between the two variable heavy parts.
- If $\alpha + \beta > 180°$ then choose the angle α or $(360-\alpha)$, whichever is larger than 180°, otherwise choose the smaller.

Geometrical method

Define $\hat{Z} = \hat{V} \times \hat{C}$ and $\hat{X} = \hat{C} \times \hat{Z}$ as described in figure 2, where $\hat{V}, \hat{C}$ are the pseudo-dyad axes of the Fab variable and constant portion: $\hat{X}, \hat{C}, \hat{Z}$ form a right-hand coordinate system. We can decide whether the Fab angle is α or $(360-\alpha)$ by comparing $\hat{V}$ with $\hat{X}$ or $\hat{Z}$. Because we are turning a three-dimensional problem to a two-dimensional problem, the choice is somewhat arbitrary without further refinement of the Fab angle definition. Figure 2.A suggests that comparing $\hat{V}$ with $\hat{X}$ may be a reasonable choice for resolving the angle ambiguity, while figure 2.B implies that comparing $\hat{V}$ with $\hat{Z}$ could be a better choice. The choice of reference axis $(\hat{X}, \hat{Z})$ can affect whether the angle we choose is larger or smaller than 180°. In *CCTBX* we used $\hat{Z}$ as a reference, to keep in line with PyMol and (Stanfield *et al.*, 2006) test cases results.

Results

Table 1 compares the elbow angle calculation between *CCTBX*, PyMol elbow angle calculation functionality and values reported in (Stanfield *et al.*, 2006). Table 2 shows how a

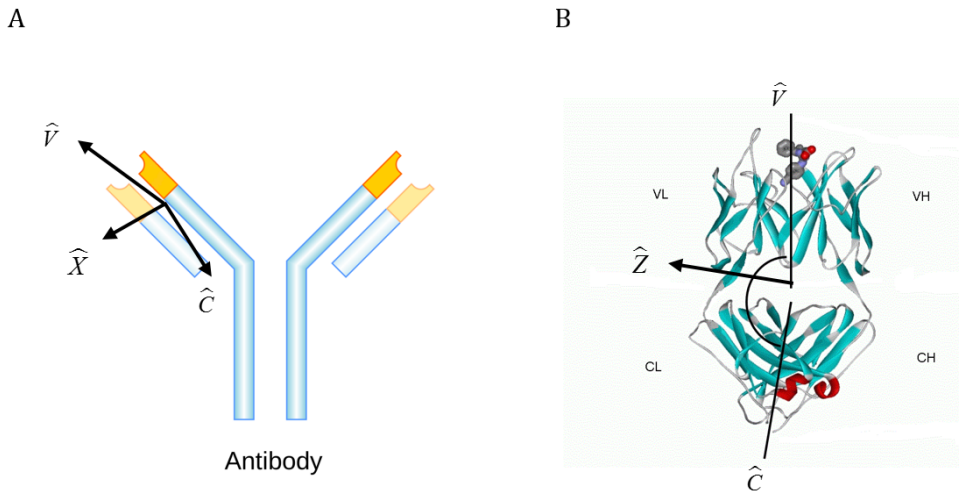


Figure 2: Geometric method. Define coordinate system to describe Antibody. $\hat{V}$ and $\hat{C}$ are the pseudo-dyad axes of the variable and constant segments respectively. In (A) $\hat{V}$ and $\hat{C}$ are drawn in a slight angle to demonstrate the geometrical approach.

Table 1: Compare the Fab angle (degrees) for five structures as reported by (Stanfield et al 2006) in blue, PyMol elbow angle in white and is CCTBX Fab angle in yellow. L and L.H are the limit light and limit heavy residues, the residue number separating the constant and variable portions of the chains.

1bbd			7fab			1dba			1plg			1nl0		
L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle
114	118	127	104	117	132	107	113	183	112	117	190	107	113	220
114	118	125	104	117	126	107	113	176	112	117	189	107	113	221
114	118	133	104	117	123	107	113	172	112	117	187	107	113	214

small difference in the choice of the limit residues. For example, ±2 residues can affect the angle by up to 4° for *1nl0* when Limit-Heavy (L.H.) changes from 113 to 114. In other cases the angle can be insensitive to the change of four residues in L.H., see *1dba*. When the angle is close to 180°, the choice of limit residue can affect the decision between α or $(360 - \alpha)$ see *1dba* when L.L. is 109, changing L.H from 112 to 113 causes the angle to change from 193° to 172°

Additional source of differences between the elbow angle calculation results might stem from different superposition tools used to overlay the Variable-Light (VL) onto the Variable-Heavy (VH) and the Constant-Light (CL) onto the Constant-Heavy (CH) rotation matrices.

Implementation in *CCTBX*

The Fab elbow angle calculation routine assumes labels for the heavy and light chains to be "*H*" and "*L*", and the limits (residue numbers separating the constant and variable segments) to be *113* and *107* based on Kabat and Chothia numbering (Wu & Kabat, 1970; Chothia & Lesk, 1987).

Command line implementation

phenix.fab_elbow_angle pdb_file_name [light=L] [heavy=H] [limit_l=107] [limit_h=113]

CCTBX

from mmtbx.utils.fab_elbow_angle import fab_elbow_angle
fab = fab_elbow_angle(pdb_hierarchy= pdb_hierarchy,limit_light=107,limit_heavy=113)
fab_angle = fab.fab_elbow_angle

Conclusion

Functionality to calculate the Fab elbow angle was added to *CCTBX*. For a set of *1bbd,7fab, 1dba, 1plg* and *1nl0* structures it yields results consistently similar to those produced by PyMol and (Stanfield *et al.*, 2006).

Table 2: Exploring the effect of different limits-heavy and limit-light choices relative to the limits in Table 1

		1bbd	7fab	1dba	1plg	1nl0
ΔL.L	ΔL.H	angle	angle	angle	angle	angle
-2	-1	126	126	172	187	214
-2	0	126	126	172	187	214
-2	1	126	126	172	187	214
-2	2	126	126	172	187	214
-1	-2	133	123	172	187	214
-1	0	126	126	172	187	214
-1	1	126	126	190	187	214
-1	2	126	126	172	187	214
0	-2	133	123	172	195	210
0	-1	133	123	172	187	214
0	1	126	126	172	187	214
0	2	126	126	172	187	214
1	-2	133	123	197	195	210
1	-1	133	123	172	195	210
1	0	133	123	180	187	214
1	2	126	126	172	187	214
2	-2	123	119	185	195	210
2	-1	133	123	193	195	210
2	0	133	123	172	195	210
2	1	131	123	172	187	214

References

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res* **28**, 235-242.

Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Jr., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *J Mol Biol* **112**, 535-542.

Chothia, C. & Lesk, A. M. (1987). *Journal of Molecular Biology* **196**, 901-917.

DeLano, W. L. (2002). *The PyMOL Molecular Graphics System,* http://www.pymol.org.

Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J Appl Crystallogr* **35**, 126-136.

Schrodinger, LLC (2010).

Stanfield, R. L., Zemla, A., Wilson, I. A. & Rupp, B. (2006). *J Mol Biol* **357**, 1566-1574.

Wu, T. T. & Kabat, E. A. (1970). *J Exp Med* **132**, 211-&.

Details of the Conformation-Dependent Library

Nigel W. Moriarty[a], Paul D. Adams[a,b] and P. Andrew Karplus[c]

[a]*Physical Biosciences Department, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA*
[b]*Department of Bioengineering, University of California Berkeley, Berkeley, CA, 94720 USA*
[c]*Department of Biochemistry and Biophysics, Oregon State University, Corvallis, OR 97331 USA*

Correspondence email: NWMoriarty@lbl.gov

Introduction

Crystallographic refinement of protein models is generally performed using a geometry restraints library based on the Engh and Huber work (Engh and Huber 1991; Engh and Huber 2001). This is a single value library (SVL) in the sense that the internal coordinates for each type (based on atom names) are restrained to a single value regardless of the molecular environment. Clearly, not all bonds between atom name pairs are given the same restraint values for every amino acid residue as shown in table 1. Indeed, alanine, for example, has different values for some of its restraints because its side chain terminates at Cβ. The only bond that is unchanged throughout the SVL is a C=O double bond which is set to 1.231 Å, while the only angle unchanged throughout is N–C=O which is set to 123.0 degrees. Certain features of the peptide linkage have special values for residues that precede a proline, as for instance, the angle C–N–Cα is adjusted by 0.9 degrees and the C–N peptide bond is elongated by 0.012 Å. An analysis of the estimated standard deviations (ESD) reveals similar characteristics.

A number of studies have shown that the SVL does not adequately represent the reality of the geometry of the polypeptide chains, but that the geometric parameters should instead be described by 'ideal geometry functions" explicitly allowing the ideal bond lengths and angles to vary with the conformation of the peptide. This was predicted over 30 years ago based on quantum mechanics calculations of small peptides (Schäfer et al. 1984; Klimkowski et al. 1985) and then verified in empirical observations of crystal structures of small peptides (Schäfer and Cao 1995)) and high-resolution crystal structures of proteins (Karplus 1996; Jiang et al. 1997). In 2008, it was proposed (Karplus et al. 2008) that the inadequacy of the SVL paradigm was responsible for the perplexing observation

Table 1: Backbone bond and angle restraints based on Engh & Huber (1991) specifying the residues associated with each restraint value and, in the case of the some of the peptide linking restraints, the class (preceding PRO or not preceding PRO) restraint values. Values are in Å and degrees.

Restraint	Value	Residues
Cα–N	1.466	PRO
	1.451	GLY
	1.458	ALA ARG ASN ASP CYS GLN GLU HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
C=O	1.231	all
C–Cα	1.516	GLY
	1.525	ALA ARG ASN ASP CYS GLN GLU HIS ILE LEU LYS MET PHE PRO SER THR TRP TYR VAL
C–N	1.341	preceding PRO
	1.329	not preceding PRO
Cα–Cβ	1.521	ALA
	1.540	ILE THR VAL
	1.530	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE PRO SER TRP TYR
C–N–Cα	122.6	preceding PRO
	121.7	not preceding PRO
Cα–C=O	119.0	PRO
	120.8	ALA ARG ASN ASP CYS GLN GLU GLY HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
N–Cα–C	112.2	ASN
	112.5	GLY
	111.8	PRO
	111.2	ALA ARG ASP CYS GLN GLU HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
N–C=O	123.0	all
Cα–C–N	116.9	preceding PRO
	116.2	not preceding PRO
C–Cα–Cβ	110.5	ALA
	109.1	ILE THR VAL
	110.1	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE PRO SER TRP TYR
N–Cα–Cβ	110.4	ALA
	103.0	PRO
	111.5	ILE THR VAL
	110.5	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE SER TRP TYR

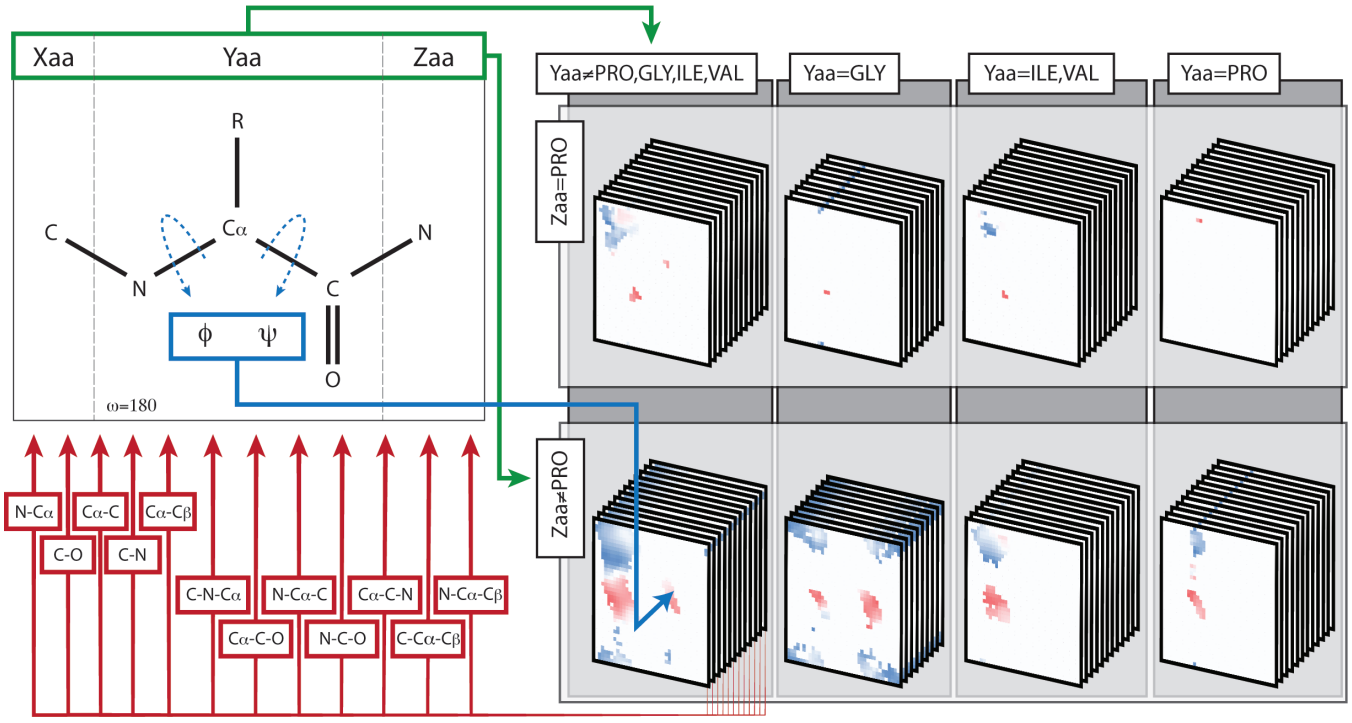


Figure 1: Diagrammatic representation of information content of the backbone CDL showing how a central residue (Yaa) and its C-terminal neighbour (Zaa) define one of eight residues classes (green lines), and the f,y-angles of the residue specify which restraint values to obtain from that class of residue (blue line) for each of the up to seven backbone bond angles and five backbone bond lengths (red lines). The colouring scheme for the CDL plots of each residue type has a common background color for simplicity.

that ultra-high resolution crystal structures consistently showed had much larger discrepancies with the restraint libraries than was expected (Jaskolski et al. 2007; Stec 2007).

The Conformation-Dependent Library (CDL) for the protein backbone (Berkholz et al. 2009; Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) is an approach that begins to address these complexities by allowing changes to the target backbone bond and angle values based on the φ,ψ torsion angles of a residue, and it has recently been implemented by Moriarty et al (Moriarty et al. 2014) in *Phenix* (Adams et al. 2010; Adams et al. 2011). The conclusions include that the CDL generates more accurate protein crystal structures and that the dramatic improvement in the geometric residuals that derives from use of the CDL comes with no draw-backs; this reinforces the conclusion that conformation-dependent ideal geometry functions truly do constitute a more accurate representation of reality than the conventional single-value ideal geometry targets.

A graphical representation of the procedure for loading the appropriate restraint values is shown in figure 1. The procedure requires three protein residues in order to determine the φ,ψ angles of the central residue. The current version of the CDL (v1.2) is limited to the largest class of residue triplets: those with both the ω torsions in the trans configuration. Each residue triplet can be further divided into those with/without the last residue (Zaa in figure 1) being a proline. This is conceptually equivalent to preceding or not preceding PRO in table 1. Figure 1 displays the lookups for Zaa=PRO in the top row and Zaa≠PRO in the bottom row. Each of these groupings is divided into four categories based on the identity of the central residue, Yaa. Explicitly, these groups are GLY; PRO; ILE

```
>> mmtbx.cdl_lookup residue_names="ALA,ALA,ALA" phi_psi_angles="90,90"

CDL values returned for specific tripeptide of sequence aa1-aa2-aa3 with the
central residue having (phi,psi) angles of (xxx°,xxx°)

Tripeptide class: NonPGIV_nonxpro

CDL values
  statistical type, number : B 218
  C(-1) - N(0)  - Ca(0)    :    122.38      1.81
  N(0)  - Ca(0) - Cb(0)    :    110.44      1.53
  N(0)  - Ca(0) - C(0)     :    113.16      1.24
  Cb(0) - Ca(0) - C(0)     :    110.01      1.80
  Ca(0) - C(0)  - O(0)     :    119.28      1.21
  Ca(0) - C(0)  - N(+1)    :    118.32      1.35
  O(0)  - C(0)  - N(+1)    :    122.37      1.38
  C(-1) - N(0)             :      1.3306    0.0148
  N(0)  - Ca(0)            :      1.4560    0.0132
  Ca(0) - Cb(0)            :      1.5319    0.0178
  Ca(0) - C(0)             :      1.5226    0.0141
  C(0)  - O(0)             :      1.2354    0.0133
```

Scheme 1: CDL lookup command-line example and output.

or VAL; and all of the rest. Once the residue triplet class has been determined, the lookup has been conceptually narrowed to the 12 restraint/Standard Deviation (SD) value pairs (nine for GLY). Each of the restraint and SD pairs is selected from a function consisting of a 36 by 36 array of values corresponding to the φ,ψ torsion angle ranges of -180 to 180 degrees binned into 10x10 degree boxes. The stacks of pages in figure 1 represent the restraint value functions for the set of 12 (or 9 for GLY) backbone bonds and angles. The values of φ,ψ (determined by the geometry of the triplet) is used to lookup the grid point for all restraints and SD. Each grid is colour-coded to represent the deviation of the value from the global "average" value which is represented as white in these images and provides the value returned for φ,ψ-bins that do not have sufficient observations to give a reliable CDL value .

Tools for accessing CDL

The CDL database of restraint values contains sets of 12 restraints and SD value pairs (nine value pairs for GLY). Accessing them in *Phenix* is done using a dedicated and thoroughly tested python module. This allows quick and easy modifications (including future CDL versions). A simple command-line tool has been added to *Phenix* that returns the restraint values. See schema 1 for an example of the command and the output. A user can easily view some restraint value pairs, and those interested in programming can view the code in the open-source *CCTBX* (Grosse-Kunstleve et al. 2002) and use it as an example to create their own scripts.

Difficulties with the hyper-dimensional optimisation space

A macromolecular refinement involves a multitude of parameters that are optimized simultaneously. This leads to a very complex potential surface that hinders the goal of finding the global minima. The starting model can have an impact on the final results. The differences can be numerical (e.g. rounding errors) or "real" (local minima with a high barrier potential). An additional complication is the weight between the experimental data and the geometry terms. The range of usable values can be quite large resulting in various differences in the final result.

Many algorithmic changes to protein refinement have been implemented in the current packages or are being tested at any

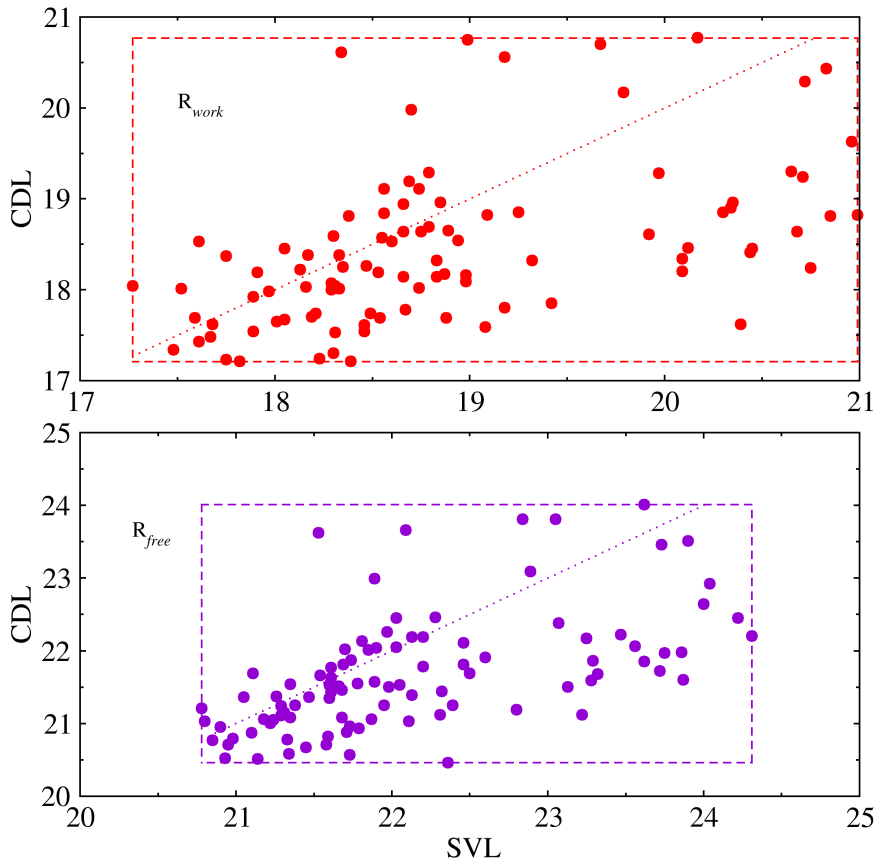


Figure 2: Comparisons of 100 randomly shaken model refinement R-factors.

point in time by the software developers. Typically, each change is tested on a certain protein model and data to determine if it is any improvement. This approach can also be used by the crystallographer to determine the best set of procedures for their particular system. Developers extend the tests to include a large number of systems to understand the parameters that control the improvements and to get statistics that validate the method as a real improvement and not a coincidence. This approach was taken by us (Moriarty et al. 2014) when investigating the effect of the CDL in *Phenix* (Adams et al. 2010; Adams et al. 2011).

An alternative approach to testing involves more extensive explorations of a single system. For instance, the effect of the starting geometry on the results (both with and without the algorithmic change) can be examined by running a number of refinements using a different random seed for each and "shaking" the geometry randomly by a reasonable amount. The values of the R-factors and the geometry rmsd will not be identical even after many refinement steps to a converged result. The spread of the values can be viewed as providing an estimate of the uncertainty in these values. This simulates the noise that the model may contain from many sources and also tests the convergence radius of the algorithmic change. The variance in the results requires a more precise approach to verifying that a change in the refinement algorithm really does improve the statistics. This technique was modified slightly for the tests.

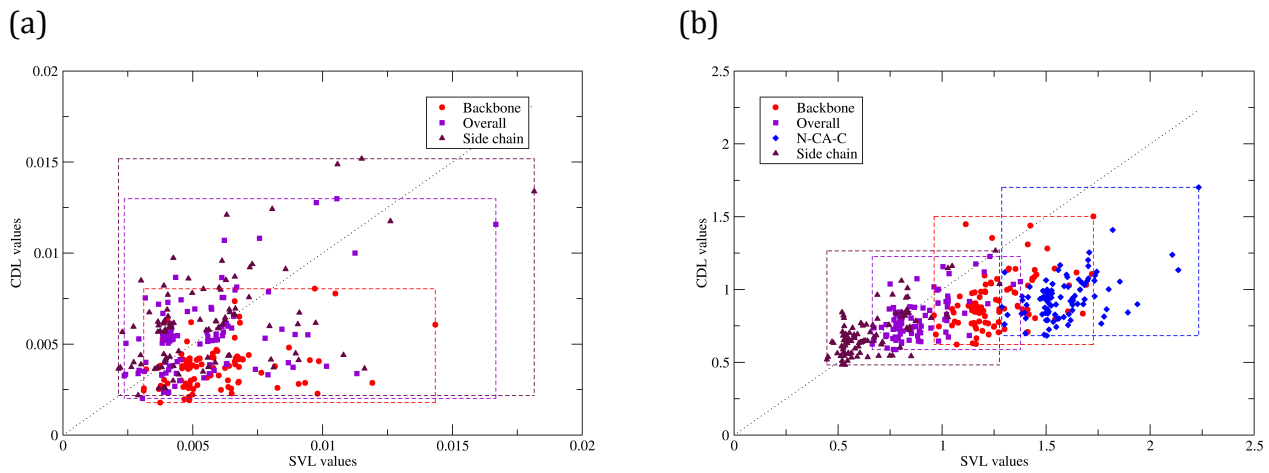


Figure 3: Comparisons of 100 randomly shaken model refinement rmsd values for bond (left) and angles (right). The plots are further divided into backbone, side chain and overall restraints.

Tests

Several PDB codes taken from the earlier proof-of-principle CDL implementation papers (Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) were chosen as representative. For each model, a set of 100 random structures were produced using the `phenix.refine` (Afonine et al. 2012) parameter, `modify_start_model.sites.shake`, set to 0.3 Å. These initial 100 models were saved so that parallel refinement could be performed that differed only by the parameter controlling the use of the CDL being set to False or True. Matching the starting geometries allows the direct comparison of what changes by a certain parameter choice. Note that the random number seed is the same for each refinement to restrict the input differences to only the geometry and the algorithmic choices.

The final values that are presented include the R-factors and the overall rmsd values for bonds and angles. Because the CDL operates on the main chain atoms via changing the bond and angle ideal values and estimated standard error of each geometry restraint, the rmsd for the main chain atoms is also presented. In addition, the Molprobity clashscore and overall score are plotted. In all plots, the values for the CDL refinements are plotted on the vertical axis versus the SVL on the horizontal axis. Each point is the result of using the same random starting geometry with and without the benefit of the CDL. Consequently, if the point is below the y=x line (included in all figures) then the CDL value is less than the SVL value. In the case of rmsd and validation scores, this is an improvement. In general, there are usually points on both sides of the y=x line. To assist in visualising the trends, a dashed box surrounds each group of points.

Figure 2 displays the R-factors for the refinements of the perturbed models of `3eln`. All following figures relate to the `3eln` example. The top portion is the R_{work} values plotted with the CDL values plotted on the vertical axis and the SVL on the horizontal. The limits of the CDL are approximately 17.2–20.8% and SVL is 17.5–21.0%. This means that the spreads are 3.6 and 2.5%, respectively. The R_{free} values has ranges of 20.5–24.0% and 20.8–24.3% meaning the CDL has generally slightly lower R factors.

Similar variance plots can be produced for the bond and angle rmsd. The bond data is presented in figure 3a. The most striking feature is that box of spreads for the backbone rmsd values is mostly below the y=x trend

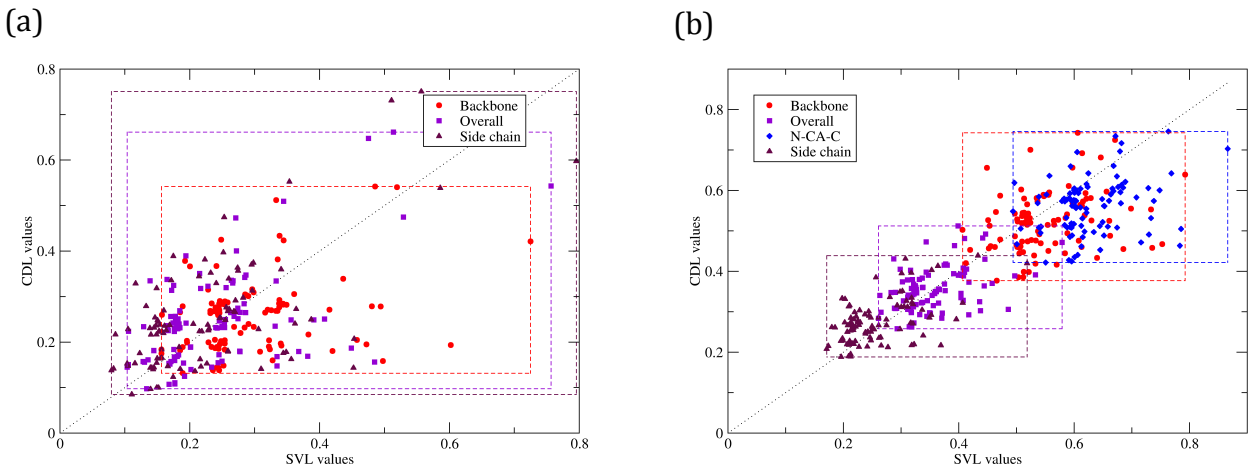


Figure 4: Comparisons of 100 randomly shaken model refinement rmsZ values for bond (left) and angles (right). The plots are further divided into backbone, side chain and overall restraints.

indicating that, in general, the bond rmsd values are reduced by the use of CDL. The rmsd value box is slightly below the trend. Interestingly, the backbone and overall rmsd have a similar distribution pattern within the box.

The angle rmsd data is plotted in figure 3b. In addition to the overall and backbone rmsd values, the rmsd of the N-C_α-C angle is also plotted. The spread boxes have a higher fraction of the values below the y=x trend than the corresponding bond boxes. In fact, the majority of the backbone and all of the N-C_α-C angle rmsd data are below the line. The angle rmsd values for the side chain are also plotted and show a relatively even spread.

In the CDL library formalism, not only are the ideal target values changed based on φ,ψ but the force constant of the restraint is also adjusted. The CDL SD value is always smaller than the corresponding SVL value. Taken alone this could be the sole reason for the decrease in rmsd results. However, as shown in figure 2, the R-factors are not unduly affected.

A common statistical measure of deviation from a mean by a population is the *Z* score. The *Z* value of an observation is defined as the

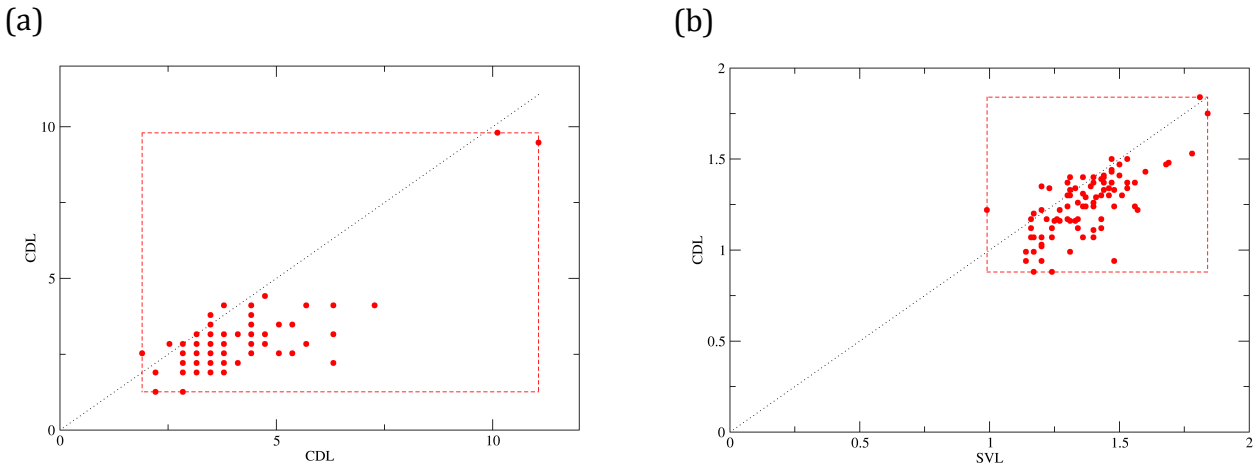


Figure 5: Comparisons of 100 randomly shaken model refinement validation scores – clashscore (left) and Molprobity (right).

number of standard deviations of a population of observations from the mean of the population. The formula is

$$Z = \frac{x - \bar{x}}{\sigma} \quad (1)$$

where x is the observation and σ is standard deviation. The *Z* score gives a dimensionless measure of the deviations based on the spread of the population. In a similar fashion to rmsd values, a *Z* value close to zero means a small deviation.

Applying the *Z* score formalism to geometry deviation can be performed as follows. Each geometry restraint has an ideal value and an SD. This provides the information needed to calculate the *Z* scores. The root mean square of the *Z* score (rms*Z*) can be calculated using

$$\mathrm{rms}Z = \sqrt{\frac{\sum_{j=1}^{N} Z_j^2}{N}} \quad (2)$$

In the context of bond and angle restraints, the smaller an rms*Z* value the closer the observations are to the ideal values. However, if the rms*Z* is close to 1.0, the distribution of the observed values has a similar (Gaussian) distribution to the distribution that provided the mean and standard deviation used in calculating the *Z* scores.

Figure 4 shows the bond and angle rms*Z* values, respectively. The *Z* scores for the SVL are calculated using the ideal values and ESD values from the standard restraints provided by the GeoStd (Moriarty and Adams) based on the Monomer Library (Vagin et al. 2004). For the CDL, the appropriated ideal and SD values are taken from the database for the backbone restraints and from the SVL for side chain.

The distributions for the bond rms*Z* values are generally less then 0.5 with a slight preference for the CDL. Interestingly, the angle rms*Z* values have the side chain scores close to 0.2 while the backbone scores are in the 0.4–0.7 range indicating that the backbone has a distribution that more closely matches the database distribution in both the CDL and SVL cases than the side chain distributions.

Validation of CDL structures is an ongoing area of research. Figure 5 shows the Molprobity clashscore (a) and overall score plots (b). In general, the spread of values is not unreasonable giving the variation in the input geometry with the scores generally better for the CDL refined models.

Conclusion

As was concluded in the *Phenix* CDL implementation paper based on parallel refinements of 27,587 models (Moriarty et al. 2014) and with the earlier papers based on refinements of only a few representative models (Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) the CDL improves the geometry of the models without adversely effecting the R-factors.

Acknowledgements

This work was supported in part by National Institutes of Health (NIH) grant R01-GM083136 (to PAK), by the NIH Project 1P01 GM063210 (to PDA), and the Phenix Industrial Consortium. This work was further supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231.

References

Adams, Paul D., Pavel V. Afonine, Gabor Bunkoczi, Vincent B. Chen, Ian W. Davis, Nathaniel Echols, Jeffrey J. Headd, et al. 2010. "PHENIX: A Comprehensive Python-Based System for Macromolecular Structure Solution." *Acta Crystallographica Section D-Biological Crystallography* 66 (February): 213–21. doi:10.1107/S0907444909052925.

Adams, Paul D., Pavel V. Afonine, Gabor Bunkoczi, Vincent B. Chen, Nathaniel Echols, Jeffrey J. Headd, Li-Wei Hung, et al. 2011. "The Phenix Software for Automated Determination of Macromolecular Structures." *Methods* 55 (1): 94–106. doi:10.1016/j.ymeth.2011.07.005.

Afonine, Pavel V., Ralf W. Grosse-Kunstleve, Nathaniel Echols, Jeffrey J. Headd, Nigel W. Moriarty, Marat Mustyakimov, Thomas C. Terwilliger, Alexandre Urzhumtsev, Peter H. Zwart, and Paul D. Adams. 2012. "Towards Automated Crystallographic

Structure Refinement with Phenix.refine." *Acta Crystallographica Section D-Biological Crystallography* 68 (April): 352–67. doi:10.1107/S0907444912001308.

Berkholz, Donald S., Maxim V. Shapovalov, Roland L. Dunbrack, Jr., and P. Andrew Karplus. 2009. "Conformation Dependence of Backbone Geometry in Proteins." *Structure* 17 (10): 1316–25. doi:10.1016/j.str.2009.08.012.

Engh, RA, and R Huber. 2001. "Structure Quality and Target Parameters." In *International Tables for Crystallo-graphy*, edited by MG Rossmann and E Arnold, F:382–92. Dordrecht: Kluwer Academic Publishers.

Engh, RA, and R. Huber. 1991. "Accurate Bond and Angle Parameters for X-Ray Protein-Structure Refinement." *Acta Crystallographica Section A* 47 (July): 392–400. doi:10.1107/S0108767391001071.

Grosse-Kunstleve, R. W., N. K. Sauter, N. W. Moriarty, and P. D. Adams. 2002. "The Computational Crystallography Toolbox: Crystallographic Algorithms in a Reusable Software Framework." *Journal of Applied Crystallography* 35 (February): 126–36. doi:10.1107/S0021889801017824.

Jaskolski, Mariusz, Miroslaw Gilski, Zbigniew Dauter, and Alexander Wlodawer. 2007. "Stereochemical Restraints Revisited: How Accurate Are Refinement Targets and How Much Should Protein Structures Be Allowed to Deviate from Them?" *Acta Crystallographica Section D-Biological Crystallography* 63 (May): 611–20. doi:10.1107/S090744490700978X.

Jiang, Xiaoqin, Ching-Hsing Yu, Ming Cao, Susan Q. Newton, Erich F. Paulus, and Lothar Schäfer. 1997. "Ø/ψ-Torsional Dependence of Peptide Backbone Bond-Lengths and Bond-Angles: Comparison of Crystallographic and Calculated Parameters." *Journal of Molecular Structure* 403 (1–2): 83 – 93. doi:http://dx.doi.org/10.1016/S0022-2860(96)09390-8.

Karplus, P. A. 1996. "Experimentally Observed Conformation-Dependent Geometry and Hidden Strain in Proteins." *Protein Science* 5 (7): 1406–20.

Karplus, P. A., M. V. Shapovalov, R. L. Dunbrack, Jr., and D. S. Berkholz. 2008. "A Forward-Looking Suggestion for Resolving the Stereochemical Restraints Debate: Ideal Geometry Functions." *Acta Crystallographica Section D-Biological Crystallography* 64 (3): 335–36. doi:10.1107/S0907444908002333.

Klimkowski, V.J., L Schäfer, F.A. Momany, and C. Van Alsenoy. 1985. "Local Geometry Maps and Conformational Transitions between Low Energy Con-Formers of N-Acetyl-N'-Methyl Glycine Amide: An Ab Initio Study at the 4–21G Level with Gradient Relaxed Geometries." *Journal of Molecular Structure* 124: 143–53.

Moriarty, Nigel W., and Paul D. Adams. *GeoStd*. http://sourceforge.net/projects/geostd.

Moriarty, Nigel W., Dale E. Tronrud, Paul D. Adams, and P. Andrew Karplus. 2014. "Conformation-Dependent Backbone Geometry Restraints Set a New Standard for Protein Crystallographic Refinement." *FEBS Journal*, n/a–n/a. doi:10.1111/febs.12860.

Schäfer, L., and M. Cao. 1995. "Predictions of Protein Backbone Bond Distances and Angles from First Principles." *Theochem-Journal of Molecular Structure* 333 (3): 201–8.

Schäfer, L., V. J. Klimkowski, Frank A. Momany, H. Chuman, and C. Van Alsenoy. 1984. "Conformational Transitions and Geometry Differences between Low-Energy Conformers of N-Acetyl-N′-Methyl Alanineamide: An Ab Initio Study at the 4-21G Level with Gradient Relaxed Geometries." *Biopolymers* 23 (11): 2335–47. doi:10.1002/bip.360231115.

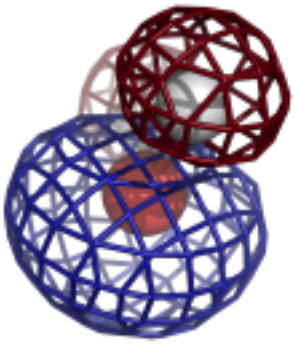
Stec, Boguslaw. 2007. "Comment on - Stereochemical Restraints Revisited: How Accurate Are Refinement Targets and How Much Should Protein Structures Be Allowed to Deviate from Them? By Jaskolski, Gilski, Dauter & Wlodawer (2007)." *Acta Crystallographica Section D-Biological Crystallography* 63 (October): 1113–14. doi:10.1107/S0907444907041406.

Tronrud, Dale E., Donald S. Berkholz, and P. Andrew Karplus. 2010. "Using a Conformation-Dependent Stereochemical Library Improves Crystallographic Refinement of Proteins." *Acta Crystallographica Section D-Biological Crystallography* 66 (7): 834–42. doi:10.1107/S0907444910019207.

Tronrud, Dale E., and P. Andrew Karplus. 2011. "A Conformation-Dependent Stereochemical Library Improves Crystallographic Refinement Even at Atomic Resolution." *Acta Crystallographica Section D-Biological Crystallography* 67 (August): 699–706. doi:10.1107/S090744491102292X.

Vagin, A. A., R. A. Steiner, A. A. Lebedev, L. Potterton, S. McNicholas, F. Long, and G. N. Murshudov. 2004. "REFMAC5 Dictionary: Organization of Prior Chemical Knowledge and Guidelines for Its Use." *Acta Crystallographica Section D-Biological Crystallography* 60 (December): 2184–95. doi:10.1107/S0907444904023510.

VOLUME FIVE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXIV

FAB ELBOW, MAP CONNECTIVITY, CDL

Table of Contents

- PHENIX News 31
- Crystallographic meetings 32
- Expert Advice
  - Fitting tips #8 – Acetyl Groups are like Peptides: Planar and *trans* 32
- Short Communications
  - Connectivity analysis tools in *CCTBX* 35
  - phenix.fab_elbow_angle: Fragment Antigen-Binding elbow angle calculation tool 39
- Articles
  - Details of the Conformation-Dependent Library 43

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

Guided Ligand Replacement

Adding ligands to an *apo* structure can now be performed using a similar ligand fit into a similar protein. The guiding protein-ligand complex can be sourced from the RCSB via web services, from a local file directory or supplied by the user.

New features

Conformation Dependent Library

The CDL has been added to *Phenix* and tested extensively. High resolution structures were used to determine restraints of the protein backbone based on the φ,ψ torsion angles. The parameter `cdl=True` activates its use. More details can found in the article on page 42 of this issue.

Automatic linking

All programs based on model geometry in *Phenix* (e.g. `phenix.real_space_refine`, `phenix.geometry_minimization` and `phenix.refine`) have moved to automatic determination of metal coordination and covalent bonding. Currently not the default, the parameter `link_all=True` will activate the linking based on residue type and distance cutoff. Finer control can be achieved by using the linking type switches (`link_metals`, `link_rna_dna`, `link_residues` and `link_carbohydrates`) and the corresponding distance cutoff parameters. The procedure covers covalently linked ligands and does simple validation of the carbohydrate polymers.

Crystallographic meetings and workshops

5th Murnau conference on Structural Biology – Focus Topic: Signal Transduction
Sept 10-13, 2014

Location: Murnau am Staffelsee, Germany
www.murnauconference.de/2014/index.html

Expert advice

Fitting Tip #8 – Acetyl Groups are like Peptides: Planar and *trans*

Jane Richardson, Jeff Headd, and Nigel W. Moriarty,
Duke University, J&J & LBNL

Acetylation is quite a common protein modification. Disturbingly, however, a recent study (Genshaft 2013) found that even for crystal structures involving the histone acetyl-lysine (3-letter code "ALY") so important in control of gene expression over 25% were modeled with extremely implausible high-energy conformations. (If it makes you feel better, over 50% were implausible in NMR models.) Clearly our automated tools have some catching up. But in the meantime it is very easy for you to apply user expertise: as we will show, acetyl groups have the same chemistry and conformational limitations as the familiar peptide bond, so just make sure they are planar and *trans* in your deposited structure. Figure 1 shows front and side views of an acetyl group at 1.2Å resolution, where the density clearly distinguishes between the carbon and the oxygen. The *trans* dihedral between the Cα-analog atoms is marked by a green line. Note that the N–H and C–O bonds in the peptide plane are also *trans* to each other and the planarity is quite exact.

Acetyl-lysine

For 109 acetyl examples in the Cambridge Structural Database (CSD; Allen 2002) of high-resolution small molecule crystal structures, every one was planar and *trans* (ω = 180° +/- 4°; Genshaft 2013). This agrees with energy calculations for *cis* versus *trans* acetyl and the high barrier between them, and the same regularity is seen for well-ordered high-resolution examples in protein crystals, such as the figures here. Genshaft et al. saw no significant correlation of acetyl-lysine error rate with refinement software, or even with resolution (since high-B examples are not adequately constrained by density even at high resolution). However in our informal survey, most multi-acetyl protein structures were either nearly random or all correct. Presumably this means that so far none of the automated software gets acetyls right reliably, but many crystallographers do understand acetyl geometry.

Acetyl N-termini

Acetyl N-termini ("ACE") are not usually as functionally important as acetyl-lysines, but they are an order of magnitude more common. Figure 2 shows map and model for a 0.8-Å resolution acetyl N-terminus, where the analogy with a standard peptide linkage is even more obvious than in figure 1. For crystal structures with a modeled ACE group, Genshaft et al. found 17% with highly strained conformations.

Confirmation by all-atom contacts

When there are other atoms nearby, then looking at the all-atom contacts for H-bonds and clashes is another very helpful diagnostic, or confirmation, of the correct orientation. Within the validation section of the Phenix GUI (Echols 2012), or on the MolProbity web site (Chen 2010), you can do this

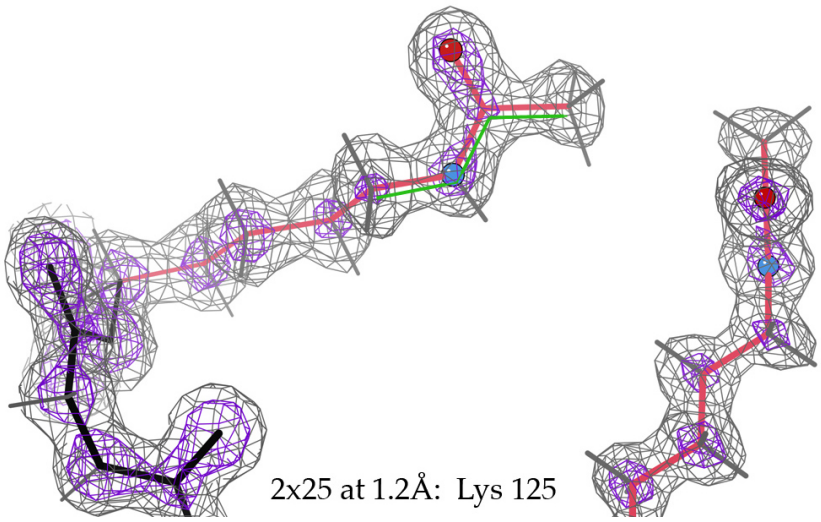


Figure 1: An N-acetyl-lysine at high resolution: (A) front view, showing that the C–N–C-C dihedral (green line) is trans, confirmed by higher density at the O than at the methyl; (B) side view, to show the tight planarity (2x25; Lammers 2010)

easily by launching the multi-criterion kinemage in KiNG (Chen 2009). For the somewhat-related case of non-covalent acetate ligands ("ACT"), all-atom contacts are the best way of correctly fitting the triangular electron density at mid resolutions (see figure 3).

Issues in refinement

In response to this problem, we have re-examined the handling of acetyl restraints in Phenix, which historically has produced almost no highly twisted cases (defined in Genshaft 2013 as >30° from planarity, and occurring from other programs), but a fair number of *cis* (0° +/- 30°). This is because Phenix has always had a restraint to planarity. However, since early 2013 it's had a strong term preferring *trans* for both ALY and the link to ACE, but no way to fix an initial fit of *cis*. We are working on a system that would check for highly strained torsions, including acetyls, and try flipping them.

Conclusion

Acetyl groups on lysine side-chains or on N-termini are fairly often modeled wrong, either by people or by software. But all you need to remember to get them right is that they have the same chemistry and conformational properties as a peptide -- so, as you know, they are very close to planar and essentially always *trans*.

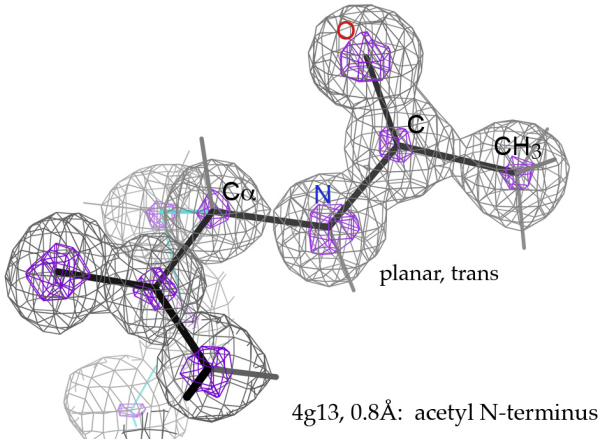


Figure 2: An acetyl N-terminus at high resolution, model and electron density (4g13; Gessmann 2012).

References

Allen FH (2002) "The Cambridge Structural Database: a quarter of a million crystal structures and rising", Acta Crystallogr. B58: 380-388

Chen VB, Davis IW, Richardson DC (2009) "KiNG (Kinemage, Next Generation: A versatile interactive molecular and scientific visualization program", Protein Sci 18:2403-240

Chen VB, Arendall WB III, Headd JJ, Keedy DA, Immormino RM, Kapral GJ, Murray LW, Richardson JS, Richardson DC (2010) "MolProbity: all-atom structure validation for macromolecular crystallography", Acta Crystallogr D66:12-21

Echols N, Grosse-Kunstleve RW, Afonine PV, Bunkoczi G, Chen VB, Headd JJ, McCoy AJ, Moriarty NW, Read RJ, Richardson DC, Richardson JS, Terwilliger TC, Adams PD (2012) "Graphical tools for macromolecular crystallography in Phenix", *J Applied Crystallogr* **45**: 581-586

Genshaft A, Moser JA, D'Antonio EL, Bowman CM, Christiansen DW (2013) "Energetically unfavorable amide conformations for N6-acetyllysine side chains in refined protein structures", *Proteins: Struct Funct Bioinf* **81**: 1051-1057

Gessmann R, Axford D, Evans G, Bruckner H, Petratos K (2012) "The crystal structure of samarosporin at atomic resolution", J Pept Sci 18: 678-684

Lammers M, Neumann H, Chin JW, James LC (2010) "Acetylation regulates cyclophilin A catalysis, immunosuppression, and HIV isomerization", *Nat Chem Biol* **6**: 331-337

Symersky J, Li S, Carson M, Luo M (2003) "Structural genomics of Caenorhabditis elegans: triosephosphate isomerase", *Proteins: Struct Funct Bioinf* **51**: 484-486

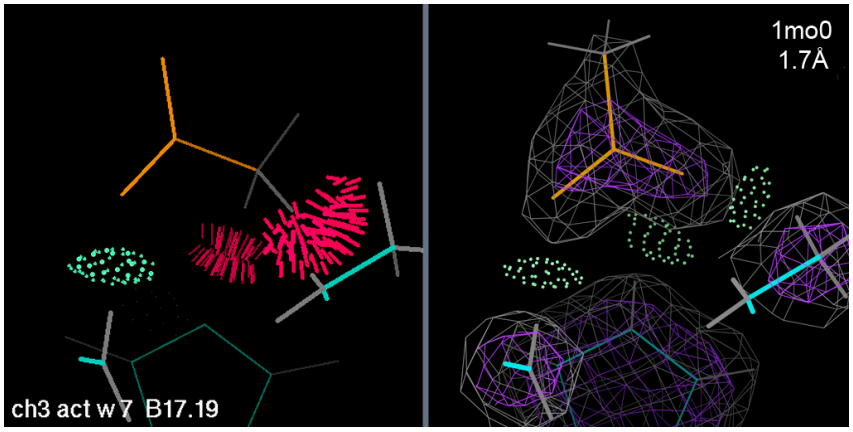


Figure 3: A solvent acetate, all-atom contacts before and after refitting (1mo0; Symersky 2003).

Connectivity analysis tools in *CCTBX*

Oleg V. Sobolev[a], Pavel V. Afonine[a] and Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: OSobolev@lbl.gov

Introduction

Connected regions in crystallographic maps defined at a given contouring level are an important feature of the map and map analysis. Studies of this feature may have various uses in crystallography: from *ab initio* phasing procedures (Lunin *et. al.,* 2000), various methods of map improvement and interpretation to analysis of connectivity of reflection data in reciprocal space (Urzhumtseva & Urzhumtsev, 2011). Robust implementation of connectivity analysis algorithm (Lunina *et al.,* 2003) in *CCTBX* (Grosse-Kunstleve *et al.* 2002) is presented.

A crystallographic map is a three-dimensional array of numbers representing the contents of a unit cell or a smaller region (asymmetric unit or a box). These, for instance, may be a distribution of electron or nuclear density, Fourier maps, solvent or other masks. One can contour the map at a particular level and study the regions that are above this level. This is the usual way for Fourier map interpretation during manual model building procedures. One can think of connected regions as peaks (also sometimes referred to as 'blobs') in a map. In the present work we report the implementation of an established connectivity analysis algorithm and extend it to determine the volume of all connected regions, coordinates, and the value of the maximum point for each region.

Implementation

The connectivity algorithm is implemented as a C++ module, `maptbx/connectivity.h`, in *CCTBX* as a separate class. Corresponding functionality is available from Python via the boost.python bindings (Abrahams & Grosse-Kunstleve, 2003) in `maptbx/boost_python/maptbx_ext.cpp`.

The runtime of the algorithm linearly scales with the number of grid points of the map. Typical runtimes are 1.85 seconds for a large map of size 400*400*400 points and 0.03 seconds for medium map of size 100*100*100 points. Calculation of the region volume, values and coordinates of maximum point in each region during the analysis can be performed at no significant additional computational cost.

The definition of connected region is based on the definition of neighbors for a particular point. In the present implementation we use six neighbors for a point with `(x,y,z)` coordinates: `(x-1,y,z)`, `(x+1,y,z)`, `(x,y-1,z)`, `(x,y+1,z)`, `(x,y,z-1)`, `(x,y,z+1)`. Alternative definitions of neighbor points may use 18 neighbor points (varying 2 of 3 coordinates) or 26 neighbor points (varying all 3 coordinates). We believe that implementation of 18- or 26-point neighbor scheme is unnecessary at this point.

Examples

1. Basics

The following code snippet illustrates the basic manipulations with the connectivity object (`co` in what follows): analysis of connected regions on the map and obtaining supplementary data (volumes, coordinates of maxima and values of maxima). To instantiate a connectivity object one needs to pass the map data and the threshold level to its constructor (see schema 1). In this example we obtained four arrays from connectivity object:

- `result_of_connectivity_analysis` – a 3D integer array of the same dimension as `map_data` filled with numbers 0, 1, 2, … N, that enumerate the N connected regions. Each grid point contains 0 if the input map values

```
>>> co = maptbx.connectivity(map_data=map_data, threshold=100)
>>> result_of_connectivity_analysis = co.result()
>>> region_volumes = list(co.regions())
>>> print region_volumes
[975696, 12152, 12152]
>>> coordinates_of_maximum_points = list(co.maximum_coors())
>>> print coordinates_of_maximum_points
[(74, 62, 62), (20, 20, 20), (60, 60, 60)]
>>> values_of_maximum_points = list(co.maximum_values())
>>> print values_of_maximum_points
[99.8855747054916, 1569.9055625594979, 1569.9055625594979]
```

Schema 1: Initialization and basic commands of connectivity analysis object.

```
>>> co = maptbx.connectivity(map_data=cmap, threshold=5)
>>> resulting_mask = co.volume_cutoff_mask(volume_cutoff=10)
```

Schema 2: Commands to obtain mask for filtering out peaks with volume smaller than 10 on a threshold equals 5.

are less than the threshold value and a non-zero value for connected regions. Each connected region is assigned its own unique integer number.

- `region_volumes` – 1D integer array of length N+1. Each element, *i*, contains the volume of *i*-th region. The zeroth element contains the volume of the under-threshold region.
- `coordinates_of_maximum_points` – 1D array of length N+1 of tuples. Similarly to volumes, contains the coordinates of maximum point.
- `values_of_maximum_points` – 1D float array of length N+1, contains the values of corresponding maximum points.

As noted previously the input data to the connectivity analysis procedure is a three-dimensional array with numbers. Therefore coordinates of the maximum points are provided as array indexes of these grid points. The volume of the region is the number of grid points with values greater than the specified threshold. The `result_of_connectivity_analysis` may be used further to obtain various kinds of masks.

2. Volume cutoff

An example of useful application of connectivity analysis procedure is to eliminate small map peaks at a particular threshold level (Afonine *et al.*, 2014, in preparation) is illustrated in schema 2. Here we obtain the `co` object the same way as in the first example, and then call the `volume_cutoff_mask` method with the `volume_cutoff` parameter to provide the smallest size of peaks that should be kept in the `resulting_mask`. A binary 3D mask with 0 where the value in `cmap` are less than the threshold and where the volume of connected regions are less or equal to `volume_cutoff` and 1 everywhere else. It should be stressed that in this case we obtain a binary mask of 0 and 1 although the connectivity analysis results are still available via `co.result()`. Finally to apply the filtration to the data one has to multiply map data `cmap` by `resulting_mask`.

3. Noise elimination based on analysis of two cutoff levels.

A more elaborate noise elimination procedure is used in the "Feature enhanced map" tool implemented in *Phenix* package (Adams *et al.*, 2010) as the *phenix.fem* program (Afonine et al, 2014, in preparation) that analyzes connectivity regions at two cutoff levels simultaneously.

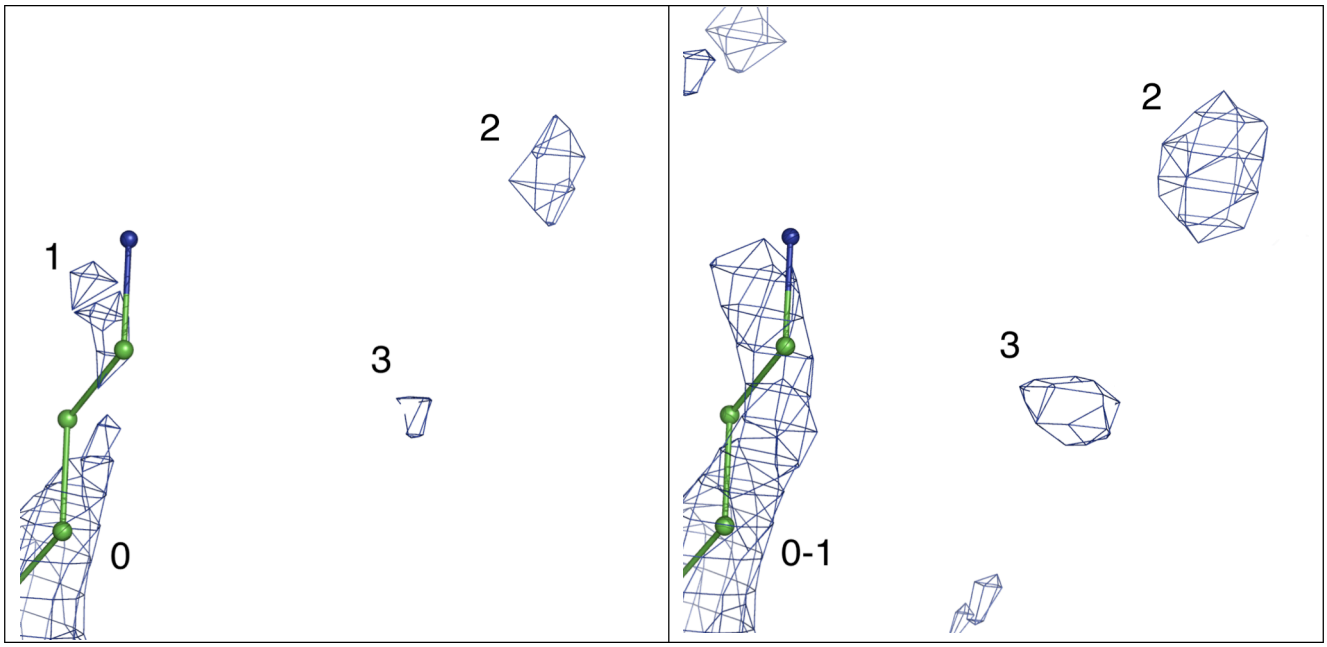


Figure 1: Schematic illustration of noise peaks elimination. Small-volume peaks at threshold *t* (left; peaks 1,2,3) and *t*-δ (right; peaks 2 and 3). Peaks 2 and 3 will be eliminated.

```
>>> co1 = maptbx.connectivity(map_data=cmap, threshold=25)
>>> co2 = maptbx.connectivity(map_data=cmap, threshold=22)
>>> resulting_mask = co2.noise_elimination_two_cutoffs(
      connectivity_object_at_t1=co1,
      elimination_volume_threshold_at_t1=12,
      zero_all_interblob_region=True)
```

Schema 3: Commands to obtain mask for noise filtering using two cutoffs and volume estimation of good peaks on the first cutoff.

Map peaks that have smaller volume than the typical volume of a reliably placed atom (e.g. water) at some threshold level *t* can be considered as noise peaks and ideally should be eliminated. Moreover, removing them at a lower level, *t*-δ, could be even better because more noise peaks would be removed. Nevertheless at the *t*-δ cutoff level some peaks previously scheduled for deletion could merge with good peaks. Such peaks should be retained. This procedure is schematically illustrated in figure 1. Figure 1(a) shows peaks 0,1,2,3 at the *t* cutoff level. Naïvely, peaks 1,2,3 could be deleted based on their small volume. However, figure 1(b) shows the same map on *t*-δ cutoff level where peak 1 has merged with molecule-related (not scheduled for deletion) peak 0 and should therefore be retained. Since peaks 2 and 3 did not merge with any region not scheduled for deletion, they will be zeroed using the shape as they appear at *t*-δ level. Particular threshold levels can be determined based on a trial-and-error approach to achieve best performance of the procedure. Information between peaks below the *t*-δ threshold level may be zeroed or preserved at the values in the original map. The code to do this filtration is shown in schema 3.

Here we created two connectivity objects with the same map data but with different threshold levels for determining connected regions. Then we call the `noise_elimination_two_cutoffs` function of the connectivity object with the lower threshold, provide the first connectivity object (with the higher threshold), volume of blobs that should be considered as good on

threshold t and a boolean parameter to specify whether to keep (`zero_all_interblob_region =False`) or mask out (`zero_all_interblob_region=True`) data between peaks at the t-δ threshold. The result is a 3D mask with the same dimensions as the original map. It is filled with 0 for map points that should be deleted and 1 for map points that should be retained. To apply the filtration one need to multiply the initial map data `cmap` by `resulting_mask`.

Conclusion

We have implemented a fast and reliable search of connected regions along with determination of their volume, values and position of maximum point at an arbitrary cutoff level. This module could provide a strong foundation for the development of various map-processing algorithms. The tools described here are available as a part of *CCTBX*.

References

Abrahams D. and Grosse-Kunstleve R.W., C/C++ Users Journal, July, 2003.

Adams, P. D., Afonine, P. V., Bunkóczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L.-W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. and Zwart, P. H. (2010). *Acta Cryst.* D**66**, 213-221.

Grosse-Kunstleve, R.W., Sauter, N.K., Moriarty, N.W. and Adams, P.D. (2002). *J. Appl. Cryst.* **35**, 126-136.

Lunin, V. Yu., Lunina, N. L. & Urzhumtsev, A. G. (2000). Acta Cryst. A**56**, 375-382.

Lunina, N.L., Lunin, V.Y. & Urzhumtsev, A. (2003). Acta Cryst. D**59**. 1702-1755.

Urzhumtseva, L. & Urzhumtsev, A. (2011). *J. Appl. Cryst.* **44**. 865-872.

phenix.fab_elbow_angle: Fragment Antigen-Binding elbow angle calculation tool

Youval Dar[a], Pavel V. Afonine[a], Nigel W. Moriarty[a] and Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: ydar@lbl.gov

Introduction

Antibody (Ab) used by white blood cell and the immune system to identify and fight antigens (Antibody generators) such as viruses and bacteria. They are large Y-shape proteins (figure 1). Fragment Antigen-Binding (Fab) are the tips of the Y-shape Ab allowing only a specific antigen to bind. The Fab is cleaved from the complete Ab to allow studying of the active portion without interference from other portions of the molecule. The Fab are composed of heavy and light chains, each with a constant and a variable regions. In a schematic illustration (figure 1.A) the constant and variable portions of the Fab are aligned at angle 180°. Figure 1.B shows a common situation, where the angle is not 180°. The angle between the variable and constant regions is called the Fab elbow angle. To find that angle, the light chain of each region is aligned onto the heavy chain by a rotation. The corresponding rotation axes are pseudo-dyad axes (imperfect dyad symmetry axes) and the Fab elbow angle is defined as the angle between these two axes. The elbow angle is an important characteristics of this class of protein structures and is typically reported in corresponding structural reports (Stanfield *et al.*, 2006). We have added a function to *CCTBX* (Grosse-Kunstleve *et al.*, 2002) that calculates Fab elbow angle.

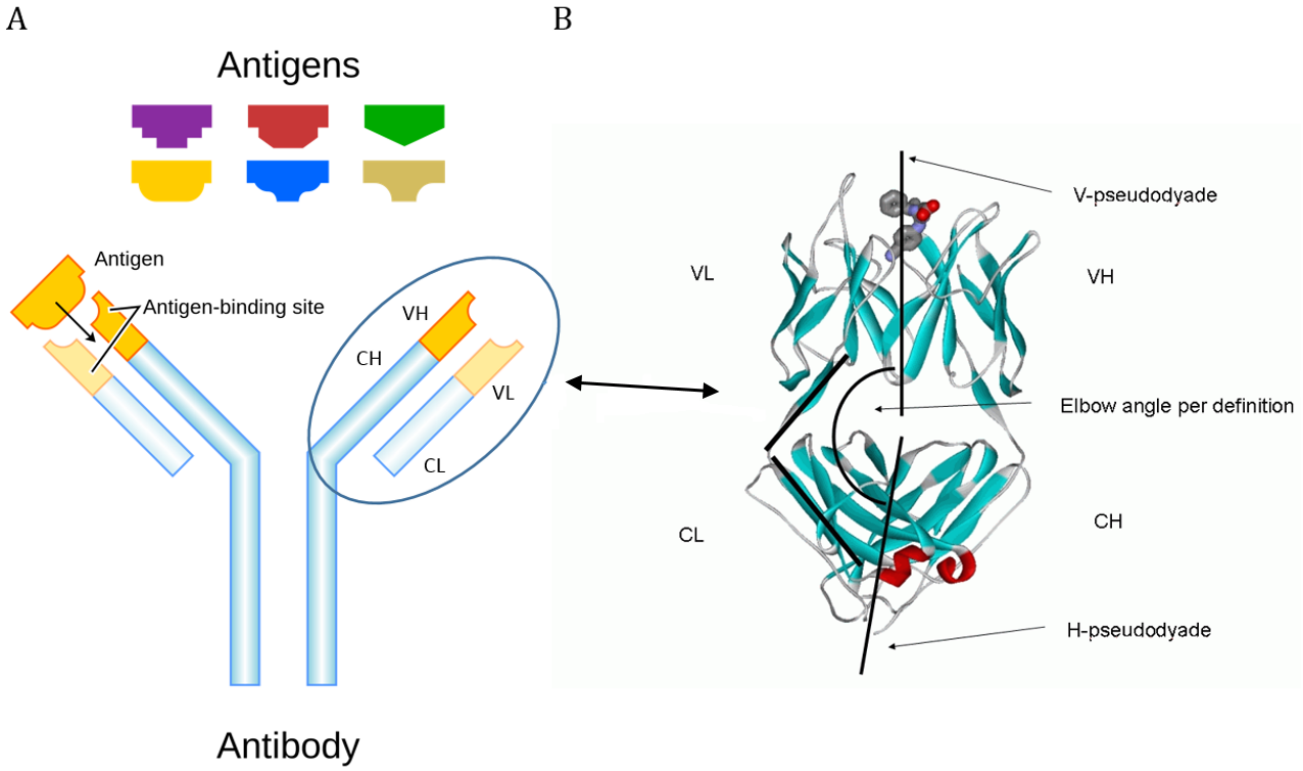


Figure 1: (A) Antigen, Wikipedia, Antigen, http://upload.wikimedia.org/wikipedia/commons/2/2d/Antibody.svg. By Fvasconcellos 19:03, 6 May 2007 (UTC) [Public domain], via Wikimedia Commons. (B) Fab angle from (Zemla A. 2005) (Web server addition). VL – Variable Light, VH – Variable Heavy, CL – Constant Light, CH – Constant Heavy.

Since the Fab is not the complete Ab its orientation with respect to the rest of the Ab is unknown. As a result, the angle between the two pseudo-dyad axes does not define a unique orientation between the variable portion of the Fab and rest of the Ab. This causes an ambiguity in the definition and interpretation of the Fab elbow angle. Factors affecting the Fab elbow angle determination are:

- Choice of limit residues (residues separating the variable and constant domains of the heavy and light chains).
- Selection of residues to align.

We used two methods to resolve the angle ambiguity in a way that matches published work (Stanfield *et al.*, 2006) and a similar function available in PyMol (Schrodinger, 2010; DeLano, 2002). The first method uses a known Fab structure as a reference while the second uses only the pseudo-dyad axes to determine the angle. Preferring not to rely on a particular structure as a reference, we have incorporated only the second method to *CCTBX*.

Method

Following (Stanfield *et al.*, 2006) we consider only the two dimensional ambiguity in elbow angle calculation, namely checking if the resulting angle is α or $(360-\alpha)$.

The program steps for finding the pseudo-dyad axes and the angle between them are as follows:

- Using *CCTBX* tools we get the rotation matrices that superpose light segments onto heavy ones.
- The atoms selected to be superposed are $C\alpha, C$ and N that do not have alternative locations. The selection strings for each are listed:
  - For the variable segment:
    chain *chain_ID* and resseq *1:Limit* and pepnames and (name ca or name n or name c) and altloc " "
  - For the constant segment:
    chain *chain_ID* and resseq *Limit+1:end* and pepnames and (name ca or name n or name c) and altloc " "
- The normalized eigenvectors, corresponding to eigenvalues of unity of those rotation matrices, represent the pseudo-dyad axes of the variable and the constant domains, $\hat{V}$ and $\hat{C}$ (Figure .B).
- $\cos(\alpha) = \hat{V} \cdot \hat{C}$ (α can be either between 90° and 180°, or between 180° and 270°).

Known protein as reference

Following the implementation described in (Stanfield *et al.*, 2006), we used PDB (Berman *et al.*, 2000; Bernstein *et al.*, 1977) protein *1bbd* as a reference using this procedure:

- Align Fab constant-heavy segments of a *1bbd* with the tested protein.
- Use the method described above and find the angle β between the two variable heavy parts.
- If $\alpha + \beta > 180°$ then choose the angle α or $(360-\alpha)$, whichever is larger than 180°, otherwise choose the smaller.

Geometrical method

Define $\hat{Z} = \hat{V} \times \hat{C}$ and $\hat{X} = \hat{C} \times \hat{Z}$ as described in figure 2, where $\hat{V}, \hat{C}$ are the pseudo-dyad axes of the Fab variable and constant portion: $\hat{X}, \hat{C}, \hat{Z}$ form a right-hand coordinate system. We can decide whether the Fab angle is α or $(360-\alpha)$ by comparing $\hat{V}$ with $\hat{X}$ or $\hat{Z}$. Because we are turning a three-dimensional problem to a two-dimensional problem, the choice is somewhat arbitrary without further refinement of the Fab angle definition. Figure 2.A suggests that comparing $\hat{V}$ with $\hat{X}$ may be a reasonable choice for resolving the angle ambiguity, while figure 2.B implies that comparing $\hat{V}$ with $\hat{Z}$ could be a better choice. The choice of reference axis $(\hat{X}, \hat{Z})$ can affect whether the angle we choose is larger or smaller than 180°. In *CCTBX* we used $\hat{Z}$ as a reference, to keep in line with PyMol and (Stanfield *et al.*, 2006) test cases results.

Results

Table 1 compares the elbow angle calculation between *CCTBX*, PyMol elbow angle calculation functionality and values reported in (Stanfield *et al.*, 2006). Table 2 shows how a

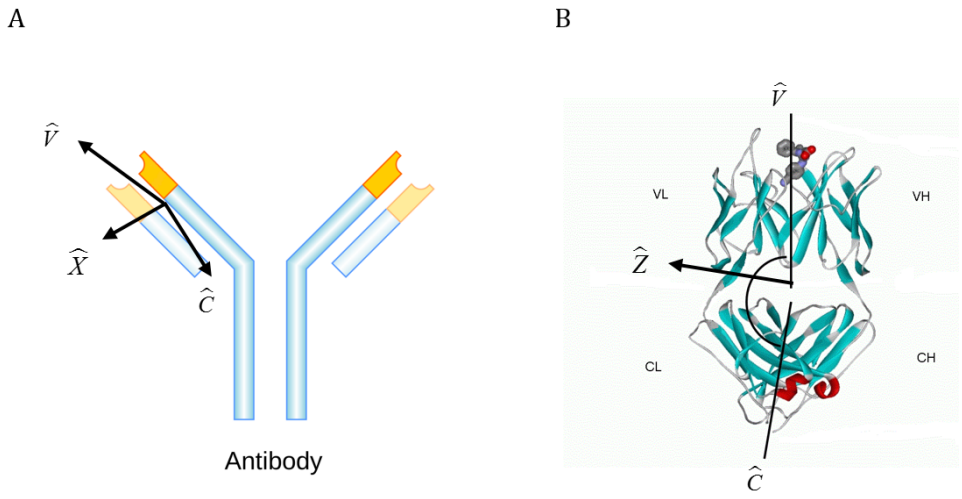


Figure 2: Geometric method. Define coordinate system to describe Antibody. $\hat{V}$ and $\hat{C}$ are the pseudo-dyad axes of the variable and constant segments respectively. In (A) $\hat{V}$ and $\hat{C}$ are drawn in a slight angle to demonstrate the geometrical approach.

Table 1: Compare the Fab angle (degrees) for five structures as reported by (Stanfield et al 2006) in blue, PyMol elbow angle in white and is CCTBX Fab angle in yellow. L and L.H are the limit light and limit heavy residues, the residue number separating the constant and variable portions of the chains.

1bbd			7fab			1dba			1plg			1nl0		
L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle	L.L	L.H	angle
114	118	127	104	117	132	107	113	183	112	117	190	107	113	220
114	118	125	104	117	126	107	113	176	112	117	189	107	113	221
114	118	133	104	117	123	107	113	172	112	117	187	107	113	214

small difference in the choice of the limit residues. For example, ±2 residues can affect the angle by up to 4° for *1nl0* when Limit-Heavy (L.H.) changes from 113 to 114. In other cases the angle can be insensitive to the change of four residues in L.H., see *1dba*. When the angle is close to 180°, the choice of limit residue can affect the decision between α or $(360 - \alpha)$ see *1dba* when L.L. is 109, changing L.H from 112 to 113 causes the angle to change from 193° to 172°

Additional source of differences between the elbow angle calculation results might stem from different superposition tools used to overlay the Variable-Light (VL) onto the Variable-Heavy (VH) and the Constant-Light (CL) onto the Constant-Heavy (CH) rotation matrices.

Implementation in *CCTBX*

The Fab elbow angle calculation routine assumes labels for the heavy and light chains to be "*H*" and "*L*", and the limits (residue numbers separating the constant and variable segments) to be *113* and *107* based on Kabat and Chothia numbering (Wu & Kabat, 1970; Chothia & Lesk, 1987).

Command line implementation

phenix.fab_elbow_angle pdb_file_name [light=L] [heavy=H] [limit_l=107] [limit_h=113]

CCTBX

from mmtbx.utils.fab_elbow_angle import fab_elbow_angle
fab = fab_elbow_angle(pdb_hierarchy= pdb_hierarchy,limit_light=107,limit_heavy=113)
fab_angle = fab.fab_elbow_angle

Conclusion

Functionality to calculate the Fab elbow angle was added to *CCTBX*. For a set of *1bbd,7fab, 1dba, 1plg* and *1nl0* structures it yields results consistently similar to those produced by PyMol and (Stanfield *et al.*, 2006).

Table 2: Exploring the effect of different limits-heavy and limit-light choices relative to the limits in Table 1

		1bbd	7fab	1dba	1plg	1nl0
ΔL.L	ΔL.H	angle	angle	angle	angle	angle
-2	-1	126	126	172	187	214
-2	0	126	126	172	187	214
-2	1	126	126	172	187	214
-2	2	126	126	172	187	214
-1	-2	133	123	172	187	214
-1	0	126	126	172	187	214
-1	1	126	126	190	187	214
-1	2	126	126	172	187	214
0	-2	133	123	172	195	210
0	-1	133	123	172	187	214
0	1	126	126	172	187	214
0	2	126	126	172	187	214
1	-2	133	123	197	195	210
1	-1	133	123	172	195	210
1	0	133	123	180	187	214
1	2	126	126	172	187	214
2	-2	123	119	185	195	210
2	-1	133	123	193	195	210
2	0	133	123	172	195	210
2	1	131	123	172	187	214

References

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res* **28**, 235-242.

Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Jr., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *J Mol Biol* **112**, 535-542.

Chothia, C. & Lesk, A. M. (1987). *Journal of Molecular Biology* **196**, 901-917.

DeLano, W. L. (2002). *The PyMOL Molecular Graphics System,* http://www.pymol.org.

Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J Appl Crystallogr* **35**, 126-136.

Schrodinger, LLC (2010).

Stanfield, R. L., Zemla, A., Wilson, I. A. & Rupp, B. (2006). *J Mol Biol* **357**, 1566-1574.

Wu, T. T. & Kabat, E. A. (1970). *J Exp Med* **132**, 211-&.

Details of the Conformation-Dependent Library

Nigel W. Moriarty[a], Paul D. Adams[a,b] and P. Andrew Karplus[c]

[a]*Physical Biosciences Department, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA*
[b]*Department of Bioengineering, University of California Berkeley, Berkeley, CA, 94720 USA*
[c]*Department of Biochemistry and Biophysics, Oregon State University, Corvallis, OR 97331 USA*

Correspondence email: NWMoriarty@lbl.gov

Introduction

Crystallographic refinement of protein models is generally performed using a geometry restraints library based on the Engh and Huber work (Engh and Huber 1991; Engh and Huber 2001). This is a single value library (SVL) in the sense that the internal coordinates for each type (based on atom names) are restrained to a single value regardless of the molecular environment. Clearly, not all bonds between atom name pairs are given the same restraint values for every amino acid residue as shown in table 1. Indeed, alanine, for example, has different values for some of its restraints because its side chain terminates at Cβ. The only bond that is unchanged throughout the SVL is a C=O double bond which is set to 1.231 Å, while the only angle unchanged throughout is N–C=O which is set to 123.0 degrees. Certain features of the peptide linkage have special values for residues that precede a proline, as for instance, the angle C–N–Cα is adjusted by 0.9 degrees and the C–N peptide bond is elongated by 0.012 Å. An analysis of the estimated standard deviations (ESD) reveals similar characteristics.

A number of studies have shown that the SVL does not adequately represent the reality of the geometry of the polypeptide chains, but that the geometric parameters should instead be described by 'ideal geometry functions" explicitly allowing the ideal bond lengths and angles to vary with the conformation of the peptide. This was predicted over 30 years ago based on quantum mechanics calculations of small peptides (Schäfer et al. 1984; Klimkowski et al. 1985) and then verified in empirical observations of crystal structures of small peptides (Schäfer and Cao 1995)) and high-resolution crystal structures of proteins (Karplus 1996; Jiang et al. 1997). In 2008, it was proposed (Karplus et al. 2008) that the inadequacy of the SVL paradigm was responsible for the perplexing observation

Table 1: Backbone bond and angle restraints based on Engh & Huber (1991) specifying the residues associated with each restraint value and, in the case of the some of the peptide linking restraints, the class (preceding PRO or not preceding PRO) restraint values. Values are in Å and degrees.

Restraint	Value	Residues
Cα–N	1.466	PRO
	1.451	GLY
	1.458	ALA ARG ASN ASP CYS GLN GLU HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
C=O	1.231	all
C–Cα	1.516	GLY
	1.525	ALA ARG ASN ASP CYS GLN GLU HIS ILE LEU LYS MET PHE PRO SER THR TRP TYR VAL
C–N	1.341	preceding PRO
	1.329	not preceding PRO
Cα–Cβ	1.521	ALA
	1.540	ILE THR VAL
	1.530	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE PRO SER TRP TYR
C–N–Cα	122.6	preceding PRO
	121.7	not preceding PRO
Cα–C=O	119.0	PRO
	120.8	ALA ARG ASN ASP CYS GLN GLU GLY HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
N–Cα–C	112.2	ASN
	112.5	GLY
	111.8	PRO
	111.2	ALA ARG ASP CYS GLN GLU HIS ILE LEU LYS MET PHE SER THR TRP TYR VAL
N–C=O	123.0	all
Cα–C–N	116.9	preceding PRO
	116.2	not preceding PRO
C–Cα–Cβ	110.5	ALA
	109.1	ILE THR VAL
	110.1	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE PRO SER TRP TYR
N–Cα–Cβ	110.4	ALA
	103.0	PRO
	111.5	ILE THR VAL
	110.5	ARG ASN ASP CYS GLN GLU HIS LEU LYS MET PHE SER TRP TYR

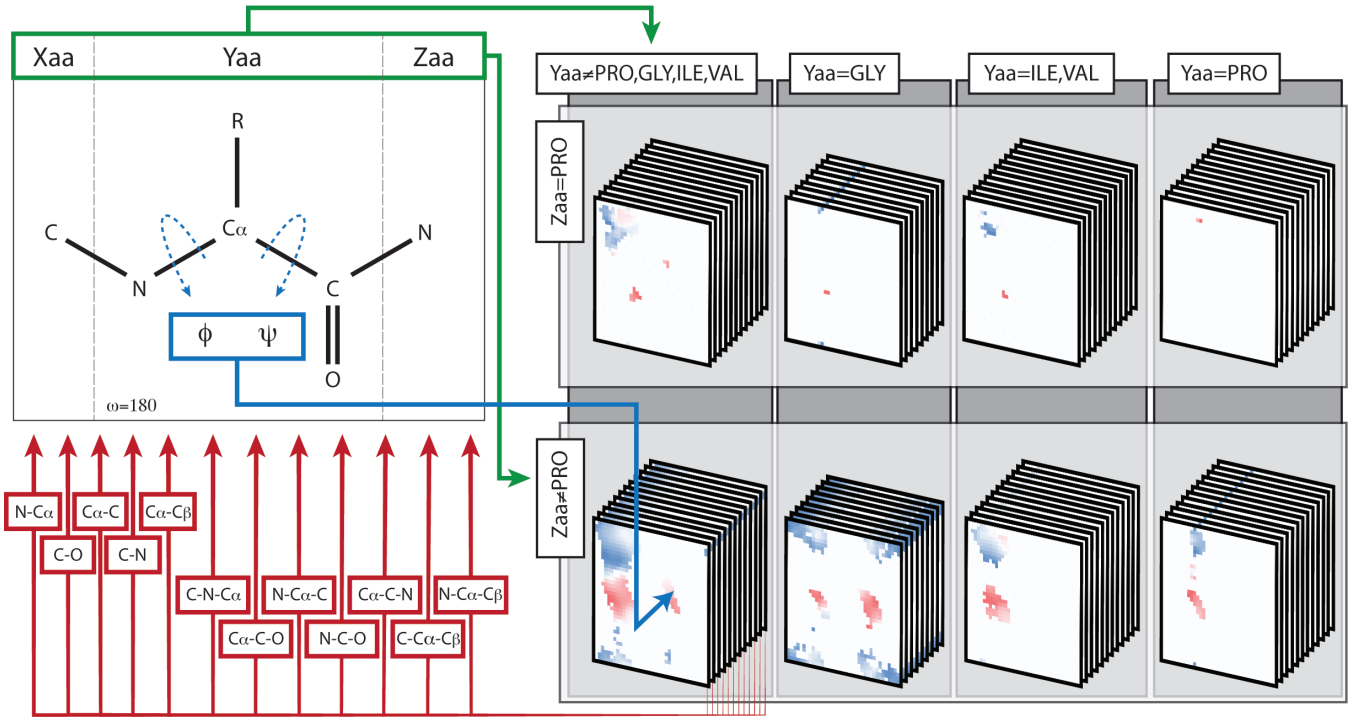


Figure 1: Diagrammatic representation of information content of the backbone CDL showing how a central residue (Yaa) and its C-terminal neighbour (Zaa) define one of eight residues classes (green lines), and the f,y-angles of the residue specify which restraint values to obtain from that class of residue (blue line) for each of the up to seven backbone bond angles and five backbone bond lengths (red lines). The colouring scheme for the CDL plots of each residue type has a common background color for simplicity.

that ultra-high resolution crystal structures consistently showed had much larger discrepancies with the restraint libraries than was expected (Jaskolski et al. 2007; Stec 2007).

The Conformation-Dependent Library (CDL) for the protein backbone (Berkholz et al. 2009; Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) is an approach that begins to address these complexities by allowing changes to the target backbone bond and angle values based on the φ,ψ torsion angles of a residue, and it has recently been implemented by Moriarty et al (Moriarty et al. 2014) in *Phenix* (Adams et al. 2010; Adams et al. 2011). The conclusions include that the CDL generates more accurate protein crystal structures and that the dramatic improvement in the geometric residuals that derives from use of the CDL comes with no draw-backs; this reinforces the conclusion that conformation-dependent ideal geometry functions truly do constitute a more accurate representation of reality than the conventional single-value ideal geometry targets.

A graphical representation of the procedure for loading the appropriate restraint values is shown in figure 1. The procedure requires three protein residues in order to determine the φ,ψ angles of the central residue. The current version of the CDL (v1.2) is limited to the largest class of residue triplets: those with both the ω torsions in the trans configuration. Each residue triplet can be further divided into those with/without the last residue (Zaa in figure 1) being a proline. This is conceptually equivalent to preceding or not preceding PRO in table 1. Figure 1 displays the lookups for Zaa=PRO in the top row and Zaa≠PRO in the bottom row. Each of these groupings is divided into four categories based on the identity of the central residue, Yaa. Explicitly, these groups are GLY; PRO; ILE

```
>> mmtbx.cdl_lookup residue_names="ALA,ALA,ALA" phi_psi_angles="90,90"

CDL values returned for specific tripeptide of sequence aa1-aa2-aa3 with the
central residue having (phi,psi) angles of (xxx°,xxx°)

Tripeptide class: NonPGIV_nonxpro

CDL values
  statistical type, number : B 218
  C(-1) - N(0)  - Ca(0)    :    122.38      1.81
  N(0)  - Ca(0) - Cb(0)    :    110.44      1.53
  N(0)  - Ca(0) - C(0)     :    113.16      1.24
  Cb(0) - Ca(0) - C(0)     :    110.01      1.80
  Ca(0) - C(0)  - O(0)     :    119.28      1.21
  Ca(0) - C(0)  - N(+1)    :    118.32      1.35
  O(0)  - C(0)  - N(+1)    :    122.37      1.38
  C(-1) - N(0)             :      1.3306    0.0148
  N(0)  - Ca(0)            :      1.4560    0.0132
  Ca(0) - Cb(0)            :      1.5319    0.0178
  Ca(0) - C(0)             :      1.5226    0.0141
  C(0)  - O(0)             :      1.2354    0.0133
```

Scheme 1: CDL lookup command-line example and output.

or VAL; and all of the rest. Once the residue triplet class has been determined, the lookup has been conceptually narrowed to the 12 restraint/Standard Deviation (SD) value pairs (nine for GLY). Each of the restraint and SD pairs is selected from a function consisting of a 36 by 36 array of values corresponding to the φ,ψ torsion angle ranges of -180 to 180 degrees binned into 10x10 degree boxes. The stacks of pages in figure 1 represent the restraint value functions for the set of 12 (or 9 for GLY) backbone bonds and angles. The values of φ,ψ (determined by the geometry of the triplet) is used to lookup the grid point for all restraints and SD. Each grid is colour-coded to represent the deviation of the value from the global "average" value which is represented as white in these images and provides the value returned for φ,ψ-bins that do not have sufficient observations to give a reliable CDL value .

Tools for accessing CDL

The CDL database of restraint values contains sets of 12 restraints and SD value pairs (nine value pairs for GLY). Accessing them in *Phenix* is done using a dedicated and thoroughly tested python module. This allows quick and easy modifications (including future CDL versions). A simple command-line tool has been added to *Phenix* that returns the restraint values. See schema 1 for an example of the command and the output. A user can easily view some restraint value pairs, and those interested in programming can view the code in the open-source *CCTBX* (Grosse-Kunstleve et al. 2002) and use it as an example to create their own scripts.

Difficulties with the hyper-dimensional optimisation space

A macromolecular refinement involves a multitude of parameters that are optimized simultaneously. This leads to a very complex potential surface that hinders the goal of finding the global minima. The starting model can have an impact on the final results. The differences can be numerical (e.g. rounding errors) or "real" (local minima with a high barrier potential). An additional complication is the weight between the experimental data and the geometry terms. The range of usable values can be quite large resulting in various differences in the final result.

Many algorithmic changes to protein refinement have been implemented in the current packages or are being tested at any

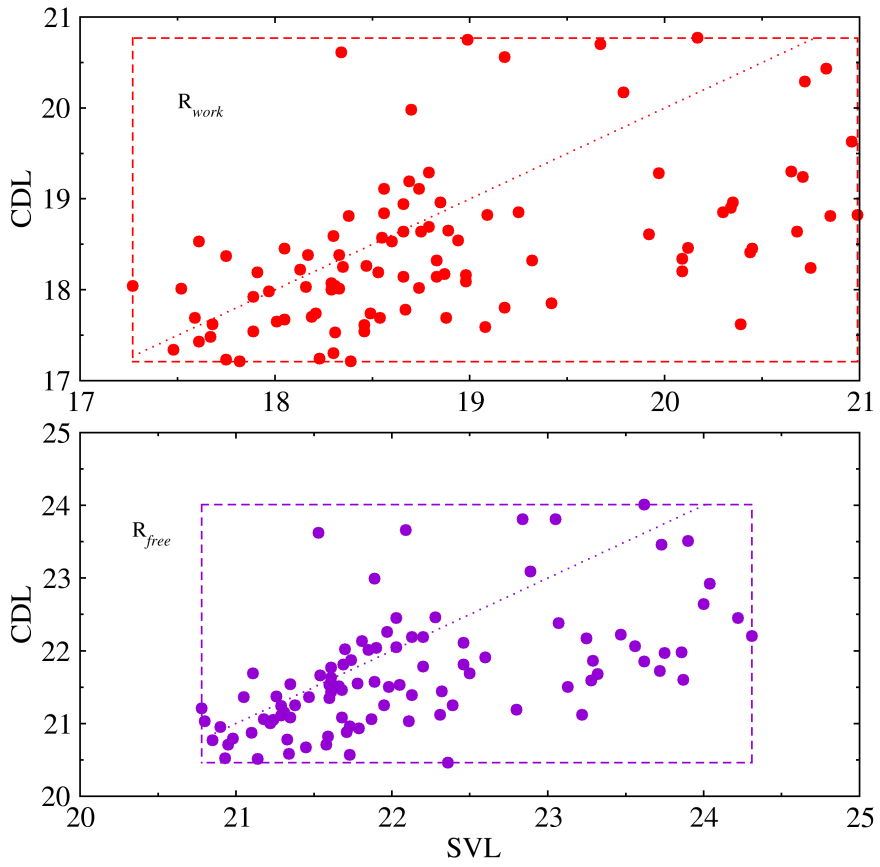


Figure 2: Comparisons of 100 randomly shaken model refinement R-factors.

point in time by the software developers. Typically, each change is tested on a certain protein model and data to determine if it is any improvement. This approach can also be used by the crystallographer to determine the best set of procedures for their particular system. Developers extend the tests to include a large number of systems to understand the parameters that control the improvements and to get statistics that validate the method as a real improvement and not a coincidence. This approach was taken by us (Moriarty et al. 2014) when investigating the effect of the CDL in *Phenix* (Adams et al. 2010; Adams et al. 2011).

An alternative approach to testing involves more extensive explorations of a single system. For instance, the effect of the starting geometry on the results (both with and without the algorithmic change) can be examined by running a number of refinements using a different random seed for each and "shaking" the geometry randomly by a reasonable amount. The values of the R-factors and the geometry rmsd will not be identical even after many refinement steps to a converged result. The spread of the values can be viewed as providing an estimate of the uncertainty in these values. This simulates the noise that the model may contain from many sources and also tests the convergence radius of the algorithmic change. The variance in the results requires a more precise approach to verifying that a change in the refinement algorithm really does improve the statistics. This technique was modified slightly for the tests.

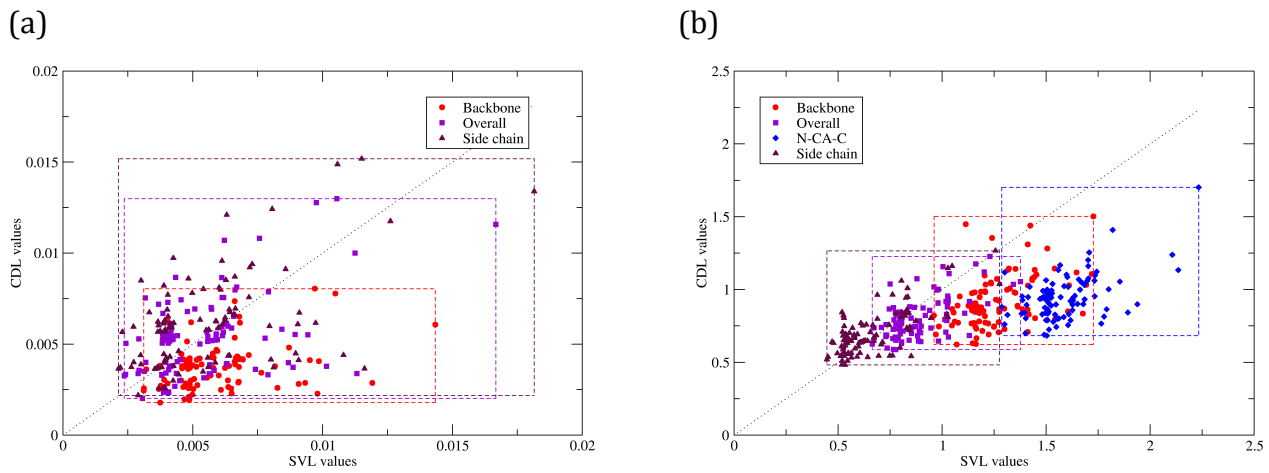


Figure 3: Comparisons of 100 randomly shaken model refinement rmsd values for bond (left) and angles (right). The plots are further divided into backbone, side chain and overall restraints.

Tests

Several PDB codes taken from the earlier proof-of-principle CDL implementation papers (Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) were chosen as representative. For each model, a set of 100 random structures were produced using the `phenix.refine` (Afonine et al. 2012) parameter, `modify_start_model.sites.shake`, set to 0.3 Å. These initial 100 models were saved so that parallel refinement could be performed that differed only by the parameter controlling the use of the CDL being set to False or True. Matching the starting geometries allows the direct comparison of what changes by a certain parameter choice. Note that the random number seed is the same for each refinement to restrict the input differences to only the geometry and the algorithmic choices.

The final values that are presented include the R-factors and the overall rmsd values for bonds and angles. Because the CDL operates on the main chain atoms via changing the bond and angle ideal values and estimated standard error of each geometry restraint, the rmsd for the main chain atoms is also presented. In addition, the Molprobity clashscore and overall score are plotted. In all plots, the values for the CDL refinements are plotted on the vertical axis versus the SVL on the horizontal axis. Each point is the result of using the same random starting geometry with and without the benefit of the CDL. Consequently, if the point is below the y=x line (included in all figures) then the CDL value is less than the SVL value. In the case of rmsd and validation scores, this is an improvement. In general, there are usually points on both sides of the y=x line. To assist in visualising the trends, a dashed box surrounds each group of points.

Figure 2 displays the R-factors for the refinements of the perturbed models of `3eln`. All following figures relate to the `3eln` example. The top portion is the R_{work} values plotted with the CDL values plotted on the vertical axis and the SVL on the horizontal. The limits of the CDL are approximately 17.2–20.8% and SVL is 17.5–21.0%. This means that the spreads are 3.6 and 2.5%, respectively. The R_{free} values has ranges of 20.5–24.0% and 20.8–24.3% meaning the CDL has generally slightly lower R factors.

Similar variance plots can be produced for the bond and angle rmsd. The bond data is presented in figure 3a. The most striking feature is that box of spreads for the backbone rmsd values is mostly below the y=x trend

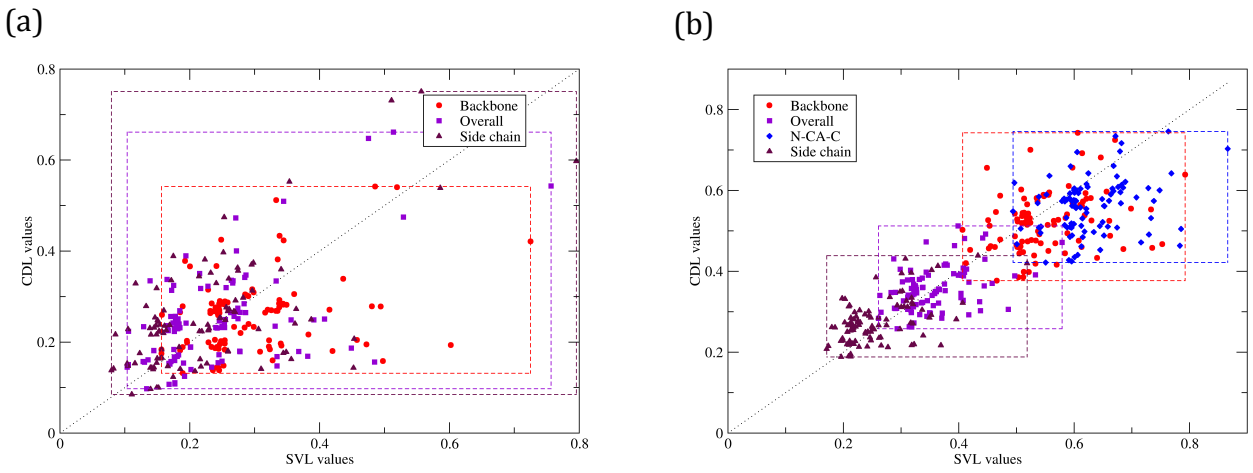


Figure 4: Comparisons of 100 randomly shaken model refinement rmsZ values for bond (left) and angles (right). The plots are further divided into backbone, side chain and overall restraints.

indicating that, in general, the bond rmsd values are reduced by the use of CDL. The rmsd value box is slightly below the trend. Interestingly, the backbone and overall rmsd have a similar distribution pattern within the box.

The angle rmsd data is plotted in figure 3b. In addition to the overall and backbone rmsd values, the rmsd of the N-C_α-C angle is also plotted. The spread boxes have a higher fraction of the values below the y=x trend than the corresponding bond boxes. In fact, the majority of the backbone and all of the N-C_α-C angle rmsd data are below the line. The angle rmsd values for the side chain are also plotted and show a relatively even spread.

In the CDL library formalism, not only are the ideal target values changed based on φ,ψ but the force constant of the restraint is also adjusted. The CDL SD value is always smaller than the corresponding SVL value. Taken alone this could be the sole reason for the decrease in rmsd results. However, as shown in figure 2, the R-factors are not unduly affected.

A common statistical measure of deviation from a mean by a population is the *Z* score. The *Z* value of an observation is defined as the

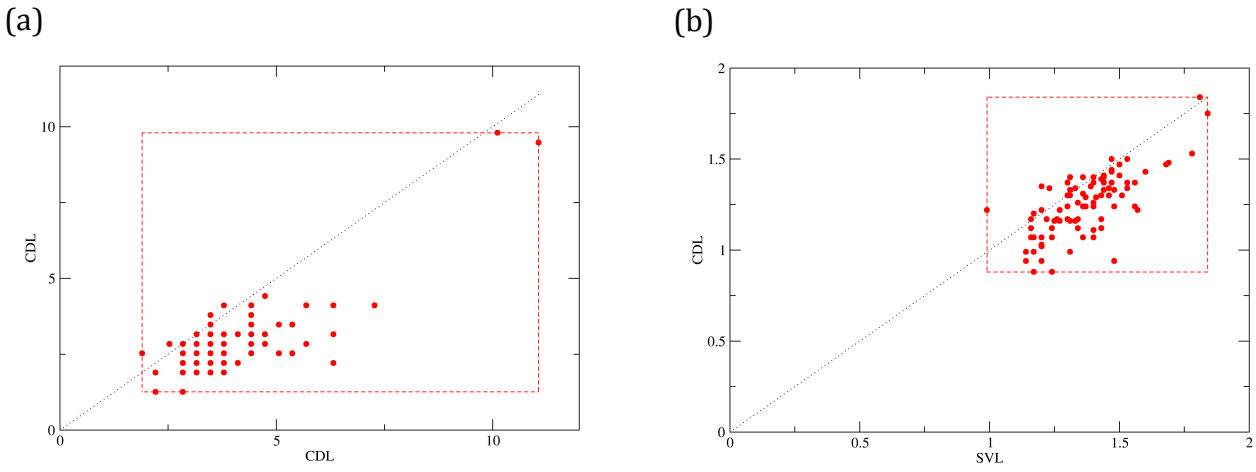


Figure 5: Comparisons of 100 randomly shaken model refinement validation scores – clashscore (left) and Molprobity (right).

number of standard deviations of a population of observations from the mean of the population. The formula is

$$Z = \frac{x - \bar{x}}{\sigma} \quad (1)$$

where x is the observation and σ is standard deviation. The *Z* score gives a dimensionless measure of the deviations based on the spread of the population. In a similar fashion to rmsd values, a *Z* value close to zero means a small deviation.

Applying the *Z* score formalism to geometry deviation can be performed as follows. Each geometry restraint has an ideal value and an SD. This provides the information needed to calculate the *Z* scores. The root mean square of the *Z* score (rms*Z*) can be calculated using

$$\mathrm{rms}Z = \sqrt{\frac{\sum_{j=1}^{N} Z_j^2}{N}} \quad (2)$$

In the context of bond and angle restraints, the smaller an rms*Z* value the closer the observations are to the ideal values. However, if the rms*Z* is close to 1.0, the distribution of the observed values has a similar (Gaussian) distribution to the distribution that provided the mean and standard deviation used in calculating the *Z* scores.

Figure 4 shows the bond and angle rms*Z* values, respectively. The *Z* scores for the SVL are calculated using the ideal values and ESD values from the standard restraints provided by the GeoStd (Moriarty and Adams) based on the Monomer Library (Vagin et al. 2004). For the CDL, the appropriated ideal and SD values are taken from the database for the backbone restraints and from the SVL for side chain.

The distributions for the bond rms*Z* values are generally less then 0.5 with a slight preference for the CDL. Interestingly, the angle rms*Z* values have the side chain scores close to 0.2 while the backbone scores are in the 0.4–0.7 range indicating that the backbone has a distribution that more closely matches the database distribution in both the CDL and SVL cases than the side chain distributions.

Validation of CDL structures is an ongoing area of research. Figure 5 shows the Molprobity clashscore (a) and overall score plots (b). In general, the spread of values is not unreasonable giving the variation in the input geometry with the scores generally better for the CDL refined models.

Conclusion

As was concluded in the *Phenix* CDL implementation paper based on parallel refinements of 27,587 models (Moriarty et al. 2014) and with the earlier papers based on refinements of only a few representative models (Tronrud, Berkholz, and Karplus 2010; Tronrud and Karplus 2011) the CDL improves the geometry of the models without adversely effecting the R-factors.

Acknowledgements

This work was supported in part by National Institutes of Health (NIH) grant R01-GM083136 (to PAK), by the NIH Project 1P01 GM063210 (to PDA), and the Phenix Industrial Consortium. This work was further supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231.

References

Adams, Paul D., Pavel V. Afonine, Gabor Bunkoczi, Vincent B. Chen, Ian W. Davis, Nathaniel Echols, Jeffrey J. Headd, et al. 2010. "PHENIX: A Comprehensive Python-Based System for Macromolecular Structure Solution." *Acta Crystallographica Section D-Biological Crystallography* 66 (February): 213–21. doi:10.1107/S0907444909052925.

Adams, Paul D., Pavel V. Afonine, Gabor Bunkoczi, Vincent B. Chen, Nathaniel Echols, Jeffrey J. Headd, Li-Wei Hung, et al. 2011. "The Phenix Software for Automated Determination of Macromolecular Structures." *Methods* 55 (1): 94–106. doi:10.1016/j.ymeth.2011.07.005.

Afonine, Pavel V., Ralf W. Grosse-Kunstleve, Nathaniel Echols, Jeffrey J. Headd, Nigel W. Moriarty, Marat Mustyakimov, Thomas C. Terwilliger, Alexandre Urzhumtsev, Peter H. Zwart, and Paul D. Adams. 2012. "Towards Automated Crystallographic

Structure Refinement with Phenix.refine." *Acta Crystallographica Section D-Biological Crystallography* 68 (April): 352–67. doi:10.1107/S0907444912001308.

Berkholz, Donald S., Maxim V. Shapovalov, Roland L. Dunbrack, Jr., and P. Andrew Karplus. 2009. "Conformation Dependence of Backbone Geometry in Proteins." *Structure* 17 (10): 1316–25. doi:10.1016/j.str.2009.08.012.

Engh, RA, and R Huber. 2001. "Structure Quality and Target Parameters." In *International Tables for Crystallo-graphy*, edited by MG Rossmann and E Arnold, F:382–92. Dordrecht: Kluwer Academic Publishers.

Engh, RA, and R. Huber. 1991. "Accurate Bond and Angle Parameters for X-Ray Protein-Structure Refinement." *Acta Crystallographica Section A* 47 (July): 392–400. doi:10.1107/S0108767391001071.

Grosse-Kunstleve, R. W., N. K. Sauter, N. W. Moriarty, and P. D. Adams. 2002. "The Computational Crystallography Toolbox: Crystallographic Algorithms in a Reusable Software Framework." *Journal of Applied Crystallography* 35 (February): 126–36. doi:10.1107/S0021889801017824.

Jaskolski, Mariusz, Miroslaw Gilski, Zbigniew Dauter, and Alexander Wlodawer. 2007. "Stereochemical Restraints Revisited: How Accurate Are Refinement Targets and How Much Should Protein Structures Be Allowed to Deviate from Them?" *Acta Crystallographica Section D-Biological Crystallography* 63 (May): 611–20. doi:10.1107/S090744490700978X.

Jiang, Xiaoqin, Ching-Hsing Yu, Ming Cao, Susan Q. Newton, Erich F. Paulus, and Lothar Schäfer. 1997. "Ø/ψ-Torsional Dependence of Peptide Backbone Bond-Lengths and Bond-Angles: Comparison of Crystallographic and Calculated Parameters." *Journal of Molecular Structure* 403 (1–2): 83 – 93. doi:http://dx.doi.org/10.1016/S0022-2860(96)09390-8.

Karplus, P. A. 1996. "Experimentally Observed Conformation-Dependent Geometry and Hidden Strain in Proteins." *Protein Science* 5 (7): 1406–20.

Karplus, P. A., M. V. Shapovalov, R. L. Dunbrack, Jr., and D. S. Berkholz. 2008. "A Forward-Looking Suggestion for Resolving the Stereochemical Restraints Debate: Ideal Geometry Functions." *Acta Crystallographica Section D-Biological Crystallography* 64 (3): 335–36. doi:10.1107/S0907444908002333.

Klimkowski, V.J., L Schäfer, F.A. Momany, and C. Van Alsenoy. 1985. "Local Geometry Maps and Conformational Transitions between Low Energy Con-Formers of N-Acetyl-N'-Methyl Glycine Amide: An Ab Initio Study at the 4–21G Level with Gradient Relaxed Geometries." *Journal of Molecular Structure* 124: 143–53.

Moriarty, Nigel W., and Paul D. Adams. *GeoStd*. http://sourceforge.net/projects/geostd.

Moriarty, Nigel W., Dale E. Tronrud, Paul D. Adams, and P. Andrew Karplus. 2014. "Conformation-Dependent Backbone Geometry Restraints Set a New Standard for Protein Crystallographic Refinement." *FEBS Journal*, n/a–n/a. doi:10.1111/febs.12860.

Schäfer, L., and M. Cao. 1995. "Predictions of Protein Backbone Bond Distances and Angles from First Principles." *Theochem-Journal of Molecular Structure* 333 (3): 201–8.

Schäfer, L., V. J. Klimkowski, Frank A. Momany, H. Chuman, and C. Van Alsenoy. 1984. "Conformational Transitions and Geometry Differences between Low-Energy Conformers of N-Acetyl-N′-Methyl Alanineamide: An Ab Initio Study at the 4-21G Level with Gradient Relaxed Geometries." *Biopolymers* 23 (11): 2335–47. doi:10.1002/bip.360231115.

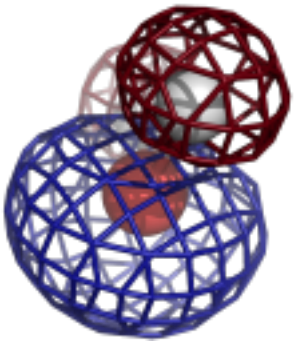
Stec, Boguslaw. 2007. "Comment on - Stereochemical Restraints Revisited: How Accurate Are Refinement Targets and How Much Should Protein Structures Be Allowed to Deviate from Them? By Jaskolski, Gilski, Dauter & Wlodawer (2007)." *Acta Crystallographica Section D-Biological Crystallography* 63 (October): 1113–14. doi:10.1107/S0907444907041406.

Tronrud, Dale E., Donald S. Berkholz, and P. Andrew Karplus. 2010. "Using a Conformation-Dependent Stereochemical Library Improves Crystallographic Refinement of Proteins." *Acta Crystallographica Section D-Biological Crystallography* 66 (7): 834–42. doi:10.1107/S0907444910019207.

Tronrud, Dale E., and P. Andrew Karplus. 2011. "A Conformation-Dependent Stereochemical Library Improves Crystallographic Refinement Even at Atomic Resolution." *Acta Crystallographica Section D-Biological Crystallography* 67 (August): 699–706. doi:10.1107/S090744491102292X.

Vagin, A. A., R. A. Steiner, A. A. Lebedev, L. Potterton, S. McNicholas, F. Long, and G. N. Murshudov. 2004. "REFMAC5 Dictionary: Organization of Prior Chemical Knowledge and Guidelines for Its Use." *Acta Crystallographica Section D-Biological Crystallography* 60 (December): 2184–95. doi:10.1107/S0907444904023510.

VOLUME SIX



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXV

CIS PEPTIDES, SAD, CARBOHYDRATES, MATTPROB, NBO

Table of Contents

- PHENIX News 1
- Crystallographic meetings 2
- Expert Advice
  - Fitting Tip #9: Avoiding excess *cis* peptides at low resolution or high B 2
- FAQ
- Short Communications
  - Plan a SAD experiment, scale SAD data, and analyze your anomalous signal 7
  - Validation of carbohydrate structures in CCP4 6.5 10
  - Disulfide bond restraints 13
- Articles
  - Improved Probabilistic Estimates of Biocrystal Solvent Content 14
  - Rapid evaluation of non-bonded overlaps in atomic modes 20

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New programs

Omegalyze *cis*-peptide validation

To help users avoid modeling unwarranted non-*trans* peptides in regions of poor data, automated identification of *cis*-peptides and peptides non-planar by >30° is now available. `phenix.omegalyze` provides text feedback and a listing of omega dihedral values. Running the command:

```
phenix.omegalyze nontrans_only=False
```

includes *trans* residues in the output. Running the command:

```
phenix.omegalyze kinemage=True
```

provides multicriterion kinemage markup for *cis*-peptides. For more details see this issue's "Avoiding excess *cis* peptides at low resolution or high B" fitting tip.

Real resolution of a dataset

Traditionally, the resolution of a diffraction dataset, d_{min} or d_{high}, is defined as the resolution of the highest-resolution reflection that belongs to this set. This value is a measure of the details that can be distinguished in the corresponding Fourier maps. Defined this way, it is meaningful if and only if the dataset is near 100% complete in the Ewald sphere $d \geq d_{high}$. If the dataset is incomplete, i.e. if there are unmeasured (missing) reflections in this sphere, then the actual resolution of the dataset may be different from the resolution of the highest-resolution reflection. Moreover, the resolution may vary in space and may be

different along different directions, with the minimum and maximum values along the directions not necessarily coinciding with the coordinate axes. There were previous efforts to give a better estimation of an 'effective' resolution taking into account the overall completeness of the data set (Weiss, 2001). Recently Urzhumtseva *et al*. (2013) proposed a mathematically strict definition of the data resolution and suggested a practical algorithm to calculate it. *Phenix* versions starting from dev-1935 have this algorithm implemented and available as the `phenix.resolution` command. The command takes a reflection data file in any of commonly used formats. It outputs three numbers: data resolution d_{high} calculated in the traditional way, and lowest and highest 'effective' resolutions of the data set calculated along all possible directions. For example, application of the `phenix.resolution` command to the PDB data set 4b44 results in values d_{high} = 2.30Å, d_{eff_min} = 2.23Å, d_{eff_max} = 3.07Å, showing a very large anisotropy of the measured set of reflections.

References

Urzhumtseva, L., Klaholz, B.P., Urzhumtsev, A. (2013) "On effective and optical resolution of diffraction data sets". *Acta Cryst.,* **D69**, 1921-1934.

Weiss, M.S. (2001). "Global indicators of X-ray data quality". *J.Appl.Cryst*., **34**, 130-135.

New features

New rotamer distributions

Use of sidechain rotamers derived from the new Top8000 dataset will now be the default system used both in refinement and validation. They are tuned so that the average number of rotamer outliers should be about the same; some individual outliers will differ, but are more accurately identified.

Carbohydrate linking

After a extensive testing of the Protein DataBank entries, automatic linking of carbohydrates is now the default for all Phenix programs using the PDB interpretation module. This includes N-linked sugars and glycosidic bonds.

Crystallographic meetings and workshops

Phenix Spring Workshop, March 2-5, 2015

Location: Berkeley, California. Presentations will be webcast and a User's Meeting for local students and postdocs will be held on Thursday.

West Coast Protein Crystallography Workshop XXII, March 15-18, 2015

Location: Monterey, California. A number of Phenix developers will be in attendance.

Expert advice

Fitting Tip #9: Avoiding excess *cis* peptides at low resolution or high B

Christopher Williams & Jane Richardson, *Duke University*

Even truly excellent pieces of software can do you in if their assumptions do not match your situation, so you should remain on the lookout for outstanding oddities. As a perhaps unexpected case, overall distribution patterns can go badly wrong when each fitting choice is made one at a time, independent of the rest.

The case in point here is *cis* peptides, both the classic x-Pro cases and especially the real but extremely rare *cis*-nonPro peptides (e.g. figure 1a). Tristan Croll, author of a paper now in press (Croll 2015) documents that the occurrence of *cis*-nonPro peptides has increased dramatically at ≥2.5Å resolution in recent years (such as the example in figure 1b). This includes otherwise well-done structures with >100 *cis*-nonPro, almost certainly unrealized by their depositors, in spite of both the warning message in Coot when fitting changes a peptide to *cis* and also the wwPDB's list of all *cis* peptides in the file header at deposition. The wwPDB is not strident about *cis* peptides and the warning in Coot is temporary, perhaps leading users to believe it has been reverted back to *trans*.

Tristan Croll contacted us, assigning some of the blame to MolProbity for not flagging *cis*-

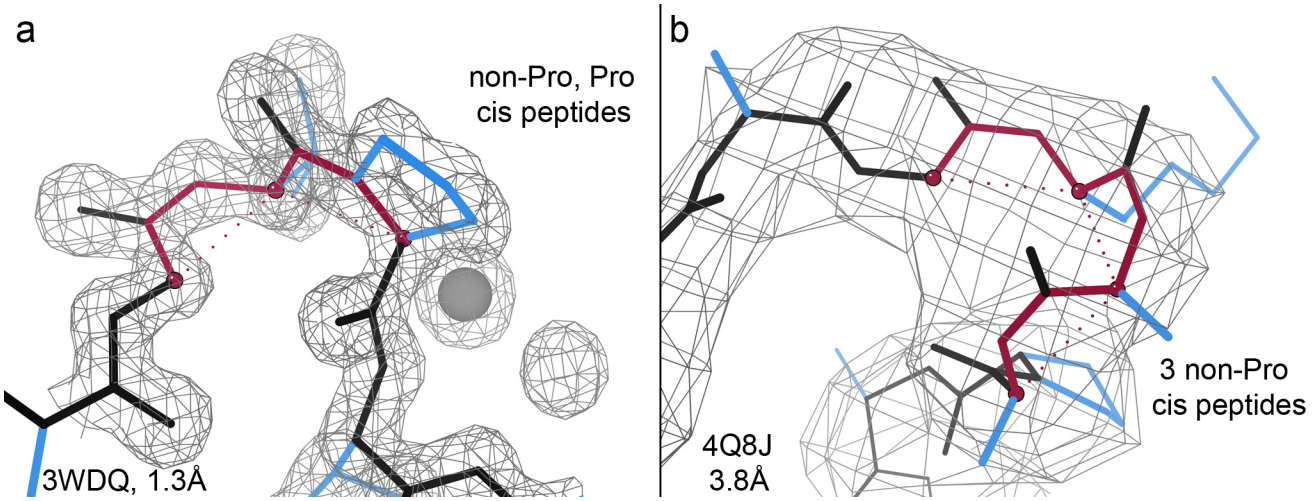


Figure 1: a) Genuine cis peptides in clear density; b) 3 unjustified cis-nonPro at low resolution.

nonPro, since people now trust that service too much for identifying all their model problems. In response, we will now report on peptide geometry. Determining the conformational category is of course simple, but devising an interface suitable at all resolution ranges was quite tricky. When we revisited the crystallographic evidence, we found that there are also an increasing number of clearly unjustified *cis* peptides at very high resolution, essentially all on loops with very poor electron density (e.g., figure 2).

Peptides preceding Pro are *cis* a bit less than 5% of the time in folded proteins, where the *cis*-*trans* barrier is substantial and packing of the Pro ring can stabilize or require a *cis* conformation. In contrast, genuine *cis*-nonPro peptides occur at a frequency of only about 0.03-0.05%; their reproducible stabilization is much harder to achieve and they are typically found at functional sites such as the classic *cis*-Gly in dihydrofolate reductase (Kraut 1982). This means some recent low-resolution structures have too many *cis*-nonPro peptides by as much as two orders of magnitude.

Even at 2Å resolution it is very difficult to be confident in the fit of a *cis*-nonPro and there is no way to justify such an assignment at >2.5Å — unless perhaps its occurrence is clear in a homologous protein at high resolution, given that conservation is strong for functionally important cases (Lorenzen 2005). We feel that eventually model-building routines should never fit *cis* or twisted nonPro peptides at low resolution or, in general, into less than very high-quality electron density. Avoidance would be very much easier to achieve than later correction. Even the much commoner *cis*-Pro are increasingly over-represented and should be subject to some limitation. As already noted (Croll 2015), unpenalized fit of *cis* peptides provides extra degrees of freedom that can allow an incorrect fit to be apparently outlier-free, potentially hiding serious problems such as

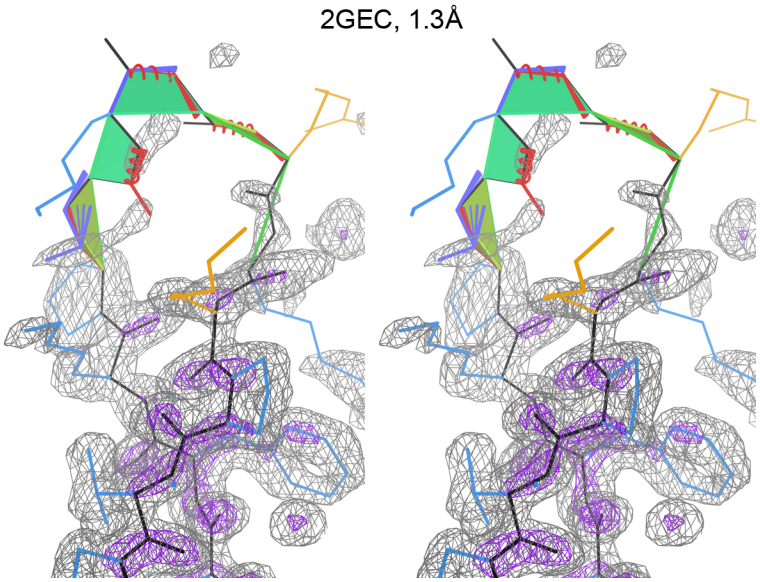


Figure 2: Cis & twisted peptides in poor density at 1.3Å.

shifted sequence register. In addition, the extra compactness of *cis* residues means they will always be systematically over-used at low resolution, since they can keep more atoms inside the contracted electron density.

The punchline for practicing crystallographers is, specifically, to check on *cis*-peptide frequencies reported in MolProbity, Coot, PDB, or elsewhere, and not to fit a *cis*-nonPro into unclear electron density. More generally, any very rare (so probably high energy) conformation should be considered suspect in high B-factor surface locations. And, finally, there will always be some new exception not yet dealt with by automated software, so you still need to actually look at your structure!

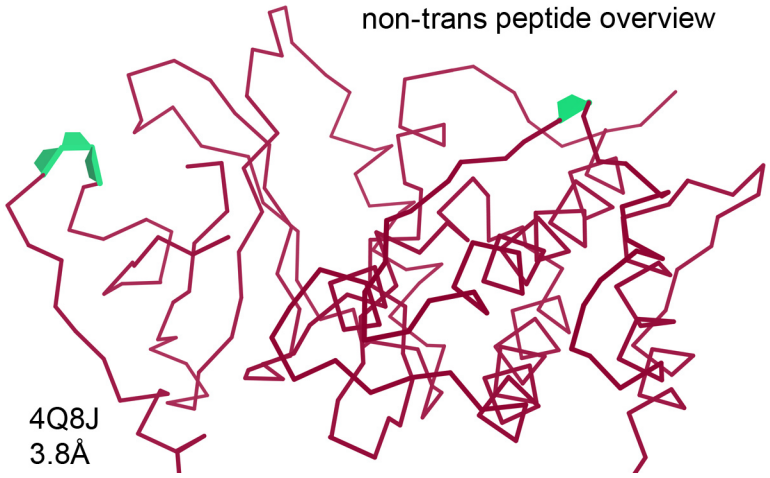


Figure 3: LoRx cis-peptide markup in an overview.

Technical notes for Phenix: Omegalyze methods

Our goals for a *cis*-peptide validation report were to provide an overall count, a list, a visualization of *cis*-peptides and to draw attention to their location in a protein structure (figure 3), without pre-judging the correctness of each individual case. Genuine *cis*-nonPro peptides are just as rare as genuine Ramachandran outliers and each case merits examination, before either acceptance as valid and interesting, or correction, or reluctant acceptance as likely wrong but uncorrectable.

An additional category of peptide conformation was needed to capture peptides with omega dihedrals far from either of the plausible planar conformations. Following usage in the PDB header, we designated peptides more than 30° away from either planar *trans* or planar *cis* as "Twisted". Although a few very convincing cases >30° are seen at high resolution (Berkholz 2012), twisted peptides are presumed to be modeling errors, requiring very strong experimental and biological evidence for justification.

Each omega peptide dihedral spans two residues, so a decision must be made for which residue number to report it. In the omegalyze assessment, as in ramalyze validation, each omega is associated with the residue immediately following, in order to preserve the unique importance of *cis*-prolines. The `phenix.omegalyze` functionality in `cctbx` was built with the same inheritance and many of the same methods as `phenix.ramalyze` to simplify its interface and maintenance.

The *cis*-peptide validation is so far available in Phenix only through the command line:

```
phenix.omegalyze file.pdb
```

Integration with the Phenix GUI is planned for implementation soon.

The default output is text, printed to `stdout` in the form as shown in schema 1. The output shown is from chain B of 2CN3.pdb, a xyloglucanase at 1.95Å, which happens to include all 3 main categories of peptide conformation, some clearly correct in the density (e.g. *cis*-Pro 353, 406) and some clearly incorrect (e.g. *cis*-Gly299, twisted His 275, Glu378); an example of each is illustrated in figure 4 a & b. Omegalyze output is in four colon-delimited columns, with additional summary lines at the end. The columns are:

1) residue identifier
2) residue type, either Pro or general

```
residue:type:omega:conformation
 B 162  GLU:General:-1.10:Cis
 B 270  GLY:General:-19.62:Cis
 B 275  HIS:General:132.01:Twisted
 B 294  PRO:Pro:5.66:Cis
 B 299  GLY:General:18.68:Cis
 B 349  ASN:General:-5.01:Cis
 B 353  PRO:Pro:-1.70:Cis
 B 377  PRO:Pro:7.46:Cis
 B 378  GLU:General:-78.58:Twisted
 B 402  PRO:Pro:9.87:Cis
 B 406  PRO:Pro:-2.29:Cis
SUMMARY: 5 cis prolines out of 45 PRO
SUMMARY: 0 twisted prolines out of 45 PRO
SUMMARY: 4 other cis residues out of 682 nonPRO
SUMMARY: 2 other twisted residues out of 682 nonPRO
```

Schema 1: Output example from omegalyse

3) omega value
4) category of peptide conformation: either *Cis*, *Trans*, or Twisted.

By default, *trans* conformation residues are not displayed. The summary lines that follow the residue-by-residue output provide whole-model counts for *cis* and twisted peptides.

Omegalyze also produces 3D validation markup for *cis*-peptides, available in the multi-criterion kinemage generated through the command line:

```
phenix.kinemage file.pdb
```

As seen in the figures here, *cis* and twisted peptides are marked with green planes that fill the space between the C_α trace and the full mainchain trace at the site of the peptide of interest. For *cis*-peptides, this results in a green trapezoid shape. For twisted peptides, the trapezoid likewise becomes twisted, indicating the severity of the twist by the angle

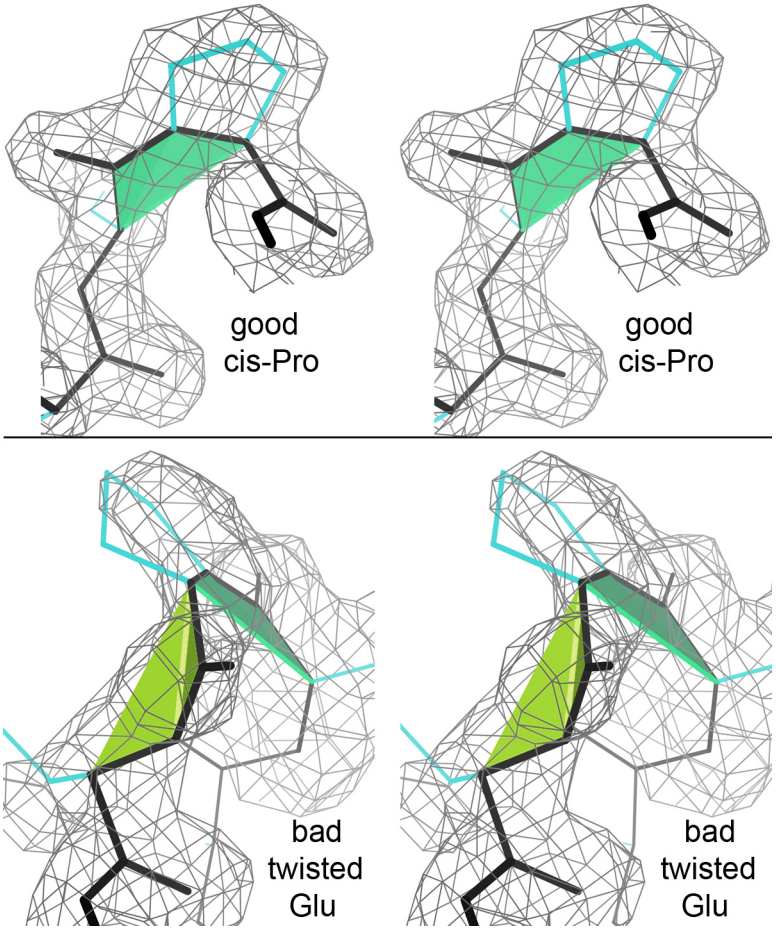


Figure 4: 2CN3: correct cis-Pro 406, incorrect twisted Glu 378.

between its two component triangles. *Cis*-peptides are colored in sea green: while the probable-outlier twisted peptides are colored in a more obnoxious lime green. These colors are intentionally similar to, but distinct from, the green used to mark Ramachandran outliers, our main method of backbone assessment. Selecting a vertex of the markup will display the calculated omega and the assigned category of the peptide.

Outside of Phenix proper, the omegalyze assessments will appear on the Richardson lab's MolProbity webserver, starting in version 4.2. They are part of the new LoRx mode for validation at low resolution along with CaBLAM (Williams 2014), but since *cis* peptides can be an issue at any resolution their diagnosis will always be done.

These various forms of omega validation will help alert users to the presence of excessive *cis* peptides in their models.

References

Berkholz DS, Driggers CD, Shapovalov MV, Dunbrack RL Jr, Karplus, A (2012) Non-planarity of peptide bonds: a common and conformation-dependent feature of proteins, *Proc Natl Acad Sci USA* **109**: 449-453.

Croll TI (2015) The rate of *cis-trans* conformation errors is increasing in low-resolution crystal structures, *Acta Cryst* **D**, in press.

Filman DJ, Matthews DA, Bolin JT, Kraut J (1982) Crystal structures of *Escherichia coli* and *Lactobacillus casei* dihydrofolate reductase refined at 1.7 Å resolution. I. General features and binding of methotrexate, *J Biol Chem* **257**: 13650-13662.

Lorenzen S, Peters B, Goede A, Preissner R, Frommel C (2005) Conservation of *cis* prolyl bonds in proteins during evolution, *Proteins* **58**: 589–595

Williams CJ, Hintze BJ, Richardson DC, Richardson JS (2013) CaBLAM identification and scoring of disguised secondary structure at low resolution, *Comp Cryst Newsletter* **4**: 9-10

FAQ

Tips for coordinated metal refinement

It is not uncommon to observe pronounced residual (difference) map features around metal ions. These features may originate from a number of possible reasons, such as: a) under-refined metal parameters, b) non-optimal metal parametrization, c) Fourier map artifacts, d) partial or/and shared occupancy, e) incorrect metal identity.

Provided that the metal identity is correctly assigned, refinement hints below may be helpful:

1. Ensure charge is in the model input file. In case of PDB file it is defined in rightmost part of ATOM record, for example:

```
HETATM 3241  SN   SN C   3      5.000   5.000   5.000  0.25 41.55          SN4+
```

2. Refine occupancy of the metal.
3. If it is a heavy metal (has substantially more electrons than typical macromolecular atoms C, O and N), refine anisotropic ADP of metal.
4. If it is anomalous scatterer refine f' and f''.
5. Run refinement until convergence. Usually it takes more than default 3 macro-cycles, about 5-10 macro-cycles.

If the residual map features are Fourier map artifacts then there isn't much one can do about it.

More details at

www.phenix-online.org/presentations/faq.pdf

Plan a SAD experiment, scale SAD data, and analyze your anomalous signal

Thomas C Terwilliger

Los Alamos National Laboratory, Los Alamos, NM

Correspondence email: terwilliger@lanl.gov

There are three new tools available to you in Phenix for planning and analyzing your SAD experiment. These tools are designed to help you decide how accurately you need to measure your data in order to solve the substructure, to scale your unmerged SAD data and to evaluate the signal you actually obtained in that SAD data.

Planning a SAD experiment with phenix.plan_sad_experiment

`phenix.plan_sad_experiment` is a tool for estimating the anomalous signal that you might get from your SAD experiment and for predicting whether this signal would be sufficient to solve the structure.

You supply `phenix.plan_sad_experiment` with a sequence file, the anomalously scattering atom you plan to use for the experiment and the wavelength for data collection. `phenix.plan_sad_experiment` will estimate the necessary I/σ_I of your dataset to provide enough anomalous signal to solve the structure.

`phenix.plan_sad_experiment` will try various values of I/σ_I for your dataset at each of several resolutions. For each I/σ_I it will estimate the half-dataset anomalous correlation that would result along with the likely true correlation between your anomalous differences and those that would be calculated from a final model of your structure (cc^*_{ano}). From this anomalous correlation (cc^*_{ano}), `phenix.plan_sad_experiment` will estimate the anomalous signal (related to cc^*_{ano} by the square root of the number of reflections divided by the square root of the number of sites). Then `phenix.plan_sad_experiment` will choose a value of I/σ_I that gives an anomalous signal of about 15 (if achievable with the maximum I/σ_I you specify).

The way that `phenix.plan_sad_experiment` and `phenix.anomalous_signal` estimate the probability that you can solve your dataset is to compare the anomalous signal in this dataset with the anomalous signal in other datasets at the same resolution. Then the fraction of similar datasets that can be solved by HySS is used as the probability that the anomalous substructure for your dataset will also be found.

Similarly, the mean figure of merit for datasets with an estimated anomalous correlation (cc^*_{ano}) similar to that for your data is used as an estimate of the figure of merit that you would obtain if the substructure is found for your crystal.

Scale unmerged anomalous data or multiple datasets with phenix.scale_and_merge

`phenix.scale_and_merge` is a tool for scaling unmerged anomalous data or multiple data files and creating a scaled dataset and two scaled half-datasets.

You supply `phenix.scale_and_merge` with a directory containing data files or the name of a single unmerged data file. You can optionally also specify a pair of labels that identifies datasets that are to be kept together. For example if you collected your data as pairs of data files with inverse beam geometry, you might have called the members of a pair `data_1_0_w1.HKL` and `data_1_0_w2.HKL`, related by w1 and w2.

`phenix.scale_and_merge` will first check the cell dimensions of all the datasets. Normally it will choose the largest set of similar crystals (you can have it keep all datasets with `only_similar_datasets=False`). It will also check the anisotropy in all the data files and calculate the average anisotropy (to be applied by default to all data files before scaling).

`phenix.scale_and_merge` will then scale all the data together to create an overall scaled dataset. This will be done in several steps. First `phenix.scale_and_merge` will split your data files into smaller files if your data files contain duplicate measurements of the same indices. The intensities in each data file will be adjusted for anisotropy to match the average anisotropy of all the data files. In this way all the data files are matched but the overall character of the data is not changed. Next, `phenix.scale_and_merge` will scale each individual file with local scaling.

Once all the individual data files have been scaled with local scaling, `phenix.scale_and_merge` will merge all the scaled files together. Merging of the individual datasets is done twice and then optionally two additional times to optimize anomalous differences.

In the first merging the individual datasets are simply averaged with weights based on the sigma for each reflection. Then the merged dataset is used as a reference and each individual dataset is compared to it. This allows an estimation of dataset variances (estimates of systematic differences between datasets and the mean). The dataset variance plus the individual variances are then to be used as an estimate of the total variance for each reflection. The second merging uses these total variances in weighting rather than the original sigma values.

If anomalous differences are optimized (default with `optimize_anomalous=True`), merging is carried out another time in order to optimize the weighting of anomalous differences in the merging step. For each unique reflection in the asymmetric unit of each dataset, the *I+* and *I-* are used to calculate anomalous differences. The anomalous differences from each individual dataset are then compared with the anomalous differences from the merged dataset to estimate individual dataset anomalous difference variances. Then the anomalous differences from each individual dataset are averaged, with weights based on the original sigmas and the dataset variances. These merged anomalous differences are then used to replace the anomalous differences in the merged dataset above (for example, a reflection in the merged dataset above that has *I+*, σ_{I+}, *I-* and σ_{I-} would get new values of *I+* and *I-* that have a difference equal to the appropriate merged anomalous difference, but the same mean as before.) The correlation of the anomalous differences in the original merged dataset and after optimization is printed (this should be high, for example 0.80).

The original datasets are then split into two parts for creation of two half-datasets. These half-datasets are useful for estimating the quality of the data and are used in `phenix.anomalous_signal` for this purpose.

The splitting into half-datasets is done in one of four ways with the method chosen based on the number of anomalous differences available for comparison using each method. With each method the data in each half-dataset are scaled just as the entire dataset was scaled. The preferred method is to split by files. Half of the data files are used to create each half-dataset. The next preferred method is to split with the first half of each dataset in one half-dataset and the second half in the other. The third preferred method is splitting alternate reflections with each unique index (after mapping to the asymmetric unit) into the two half-datasets. The last

method is to randomly assign reflections to the two half-datasets. The reason for this hierarchical approach is that reflections measured close in time, within the same dataset are better matched than those measured further in time or within different datasets. These approaches for splitting the data files attempt to pair anomalous differences measured close in time and in the same dataset.

Analyzing the anomalous signal in a SAD dataset with phenix.anomalous_signal

Once you have scaled your data with `phenix.scale_and_merge`, you can use `phenix.anomalous_signal` to analysis the anomalous signal in your data and to predict whether this signal is sufficient to solve the structure.

You supply `phenix.anomalous_signal` with scaled anomalous data, two half-dataset files with scaled anomalous data, the number of sites or a sequence file and name of the anomalously scattering atom. `phenix.anomalous_signal` will calculate the anomalous signal in your dataset from (1) the half-dataset anomalous correlation, (2) the skew of the anomalous difference Patterson map and (3) the estimated measurement error in your data.

`phenix.anomalous_signal` will then estimate the probability that you can solve this dataset using likelihood-based HySS (standard run) and will estimate the figure of merit of phasing that you should obtain.

The way that `phenix.plan_sad_experiment` and `phenix.anomalous_signal` estimate the probability that you can solve this dataset is to compare the anomalous signal in this dataset with the anomalous signal in other datasets at the same resolution. Then the fraction of similar datasets that can be solved by HySS is used as the probability that your dataset will also be solved.

With these new tools you should now be able to plan and carry out your SAD experiment even more effectively than before!

Validation of carbohydrate structures in CCP4 6.5

Jon Agirre and Kevin Cowtan
Department of Chemistry, The University of York, YO10 5DD (UK)
Correspondence email: kevin.cowtan@york.ac.uk

Introduction

Pyranose and furanose sugars, as most other cyclic compounds, have strong conformational preferences that are dictated by a minimization of angle, torsional and steric strains. For most of the biologically relevant pyranoses, the preferred conformation is either a $^4C_1$ or a $^1C_4$ chair, and any transitions to higher-energy conformations (*e.g.* half-chair or envelope) are usually a consequence of external factors such as the neighboring presence of catalytic residues from a carbohydrate-active enzyme.

While the set of geometric restraints that crystallographic refinement software impose is usually descriptive enough to reproduce a realistic geometry for amino acids modeled at medium to low resolution, cyclic sugars may end up in a high-energy conformation that, in the absence of clear density supporting it, should be treated as an outlier. Even with the addition of harmonic torsion restraints, any subtle mistakes in the specification of bonding distances – linkages between sugars need to be explicitly declared – or a wrong three-letter code selection (*e.g.* using 'GLC' for β-D-glucopyranose, together with the restraints designated for it) can result in distortion.

Conformational analysis

The method proposed by Cremer and Pople (1975) has been chosen as primary conformational analysis tool. The algorithm, which is applicable to rings of any cardinality, calculates a minimal set of puckering coordinates that describe each conformation. A total puckering amplitude term (*Q*) is also calculated

$$Q = \sqrt{\sum_{j=1}^{N} (\vec{R}_j \cdot \vec{n})^2} = \sqrt{\sum_{j=1}^{N} Z_j^2}$$

for *N* atoms, with $\vec{R}_j$ being the positional vector of atom *j* in a coordinate system with the origin in the ring's geometrical center, and $\vec{n}$ being the unit vector normal to the ring's mean plane. Therefore, Z_j accounts for the vertical displacement of atom *j*

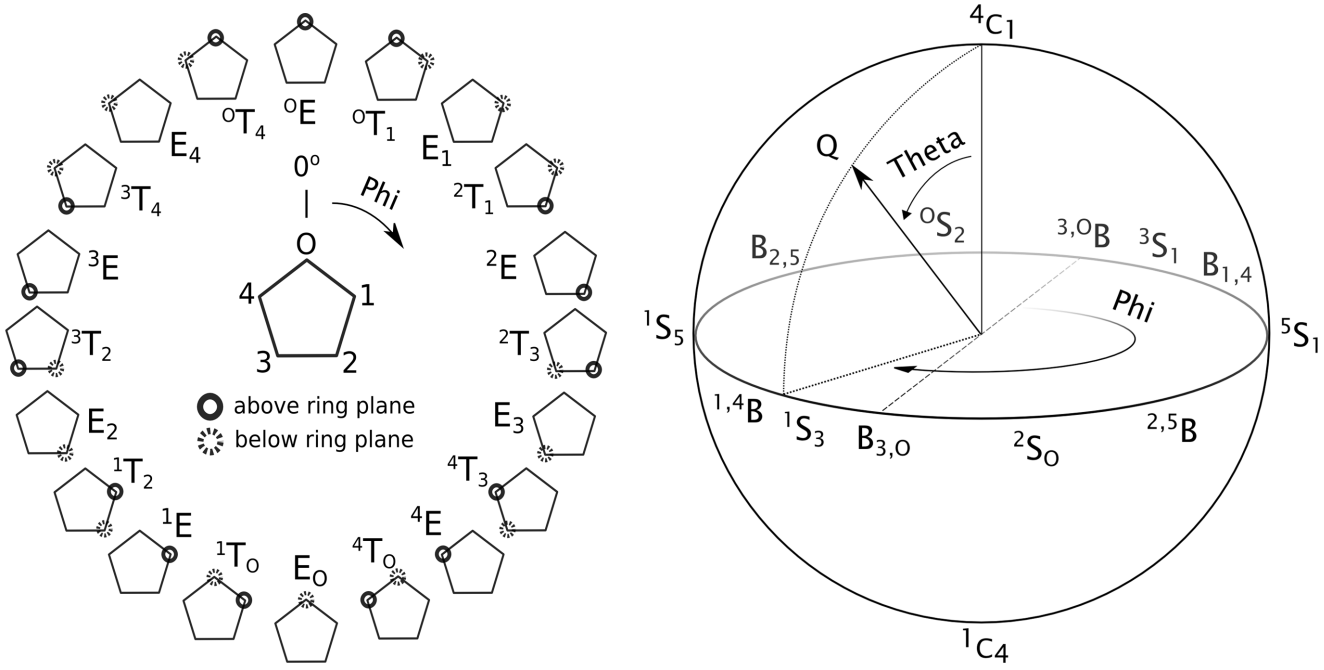


Figure 1: Correspondence between the Cremer-Pople angles for furanoses (Φ, image on the left) and pyranoses (Φ, Θ) to the conformation codes defined by IUPAC.

from the mean ring plane. In the case of pyranoside rings, the puckering coordinates are most conveniently expressed in angular form ($\Phi_{[0,2\Pi]}, \Theta_{[0,\Pi]}$) by solving the following set of equations

$$Q \sin\Theta \cos\Phi = \sqrt{\frac{1}{3}} \sum_{j=1}^{6} Z_j \cos\left[\frac{4\Pi(j-1)}{6}\right]$$

$$Q \sin\Theta \sin\Phi = \sqrt{\frac{1}{3}} \sum_{j=1}^{6} Z_j \sin\left[\frac{4\Pi(j-1)}{6}\right]$$

$$Q \cos\Theta = \sqrt{\frac{1}{6}} \sum_{j=1}^{6} (-1)^{j-1} Z_j$$

so that they can be graphically represented on the surface of a sphere of radius Q. This sphere, with lowest energy $^4C_1$ and $^1C_4$ ($\Theta = 0$ and $\Theta = \Pi$) chair conformations on the North and South pole respectively, is able to depict every conformational itinerary followed by pyranose sugars in their transition from their low-energy chair conformation to a more distorted boat or skew-boat intermediate ($\Theta = {}^{\Pi}/_{4}$) during catalysis (Davies *et al.*, 2011). For convenience, additional vertical displacements akin to the Z_j ones are calculated for those atoms implicated in the anomer and handedness detection.

A similar calculation is performed for furanose rings, but producing just Q and Φ .

Characteristics

Privateer-validate relies on a small database of three-letter codes for which the anomer, handedness and lowest-energy conformation have been calculated. By comparison to these values, the program is able to determine, for instance, if a modeled carbohydrate has been distorted from its initial conformation. When run within CCP4i2 (currently in alpha test phase), an HTML report is displayed with the IUPAC-compliant conformation code, the Cremer-Pople parameters and diagnostics for each sugar. The equivalence between Cremer-Pople angles and conformations can be visualized in Figure 1.

In addition to chemical correctness checks, a real space correlation coefficient is calculated for each sugar against an mFo-DFc map computed omitting all sugar models from the phase calculation. The resulting map coefficients can also be output to an MTZ file for later use.

Whenever glycosylation is present in the input structure, the program will produce linear descriptions of the detected trees. Here is an example of the nomenclature used:

$$\text{Man}\leftarrow\alpha_2\text{–Man}\leftarrow\alpha_6\text{–Man}\leftarrow\alpha_6\text{–(Man}\leftarrow\alpha_3\text{–)Man}\leftarrow\beta_4\text{–GlcNac}\leftarrow\beta_4\text{–GlcNac} \leftarrow \beta\text{–ASN}_{523}$$

Coot script files (Emsley *et al.*, 2010) are also produced with a guided tour of the detected issues. These scripts can be used manually outside CCP4i2 or by simply selecting 'Manual model rebuilding' within the aforementioned graphical interface. The omit mFo-DFc map is presented in pink color while 2mFo-DFc density is displayed in blue. Each button contains a description of the issue or issues detected by privateer-validate, as it can be seen in Figure 2.

The produced startup scripts also activate torsion angle restraints by default; using them in combination with the 'sphere refinement' function (hotkey: 'R') makes most of the issues exposed by privateer-validate easily fixable in Coot. Torsion angle restraints may be subsequently required by refinement software in order to avoid further distortions.

Availability

The Privateer software package can be obtained as part of the CCP4 distribution (http://www.ccp4.ac.uk). The validation software presented here serves as the prelude to a sugar detection and modeling tool that will be distributed in the forthcoming weeks as an update to CCP4 6.5. Privateer uses the Clipper libraries (Cowtan, 2003) and is distributed under the terms of the GNU Lesser General Public License.

Acknowledgements

The authors would like to thank Professors Eleanor J Dodson, Keith S Wilson and especially Gideon J Davies for many stimulating discussions.

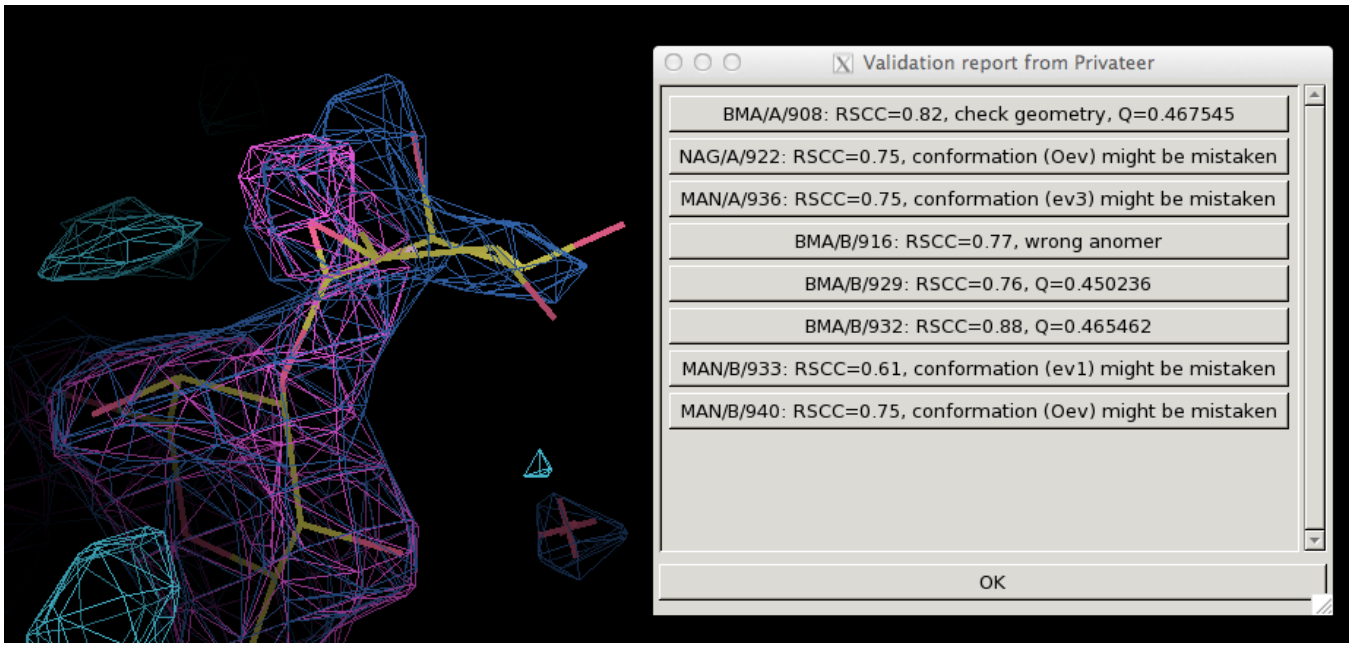


Figure 2: Validation of a glycoprotein (PDB code: 4IID). A number of terminal sugars display a high-energy conformation (envelope) as a consequence of the weak density they have been modeled in and the absence of torsion restraints in the original refinement.

References

Cowtan, K. (2003). The Clipper C++ libraries for x-ray crystallography. IUCr Computing Commission Newsletter, 2:4–9.

Cremer, D. t. and Pople, J. (1975). General definition of ring puckering coordinates. Journal of the American Chemical Society, 97(6):1354–1358.

Davies, G. J., Planas, A., and Rovira, C. (2011). Conformational analyses of the reaction coordinate of glycosidases. Accounts of chemical research, 45(2):308– 316.

Emsley, P., Lohkamp, B., Scott, W., and Cowtan, K. (2010). Features and development of coot. Acta Crystallographica Section D: Biological Crystallography, 66(4):486–501.

Disulfide bond restraints

Oleg V. Sobolev,[a] Nigel W. Moriarty,[a] Pavel V. Afonine,[a] Bradley J. Hintze,[c] David C. Richardson,[c] Jane S. Richardson[c] and Paul D. Adams[a,b]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
[b]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*
[c]*Department of Biochemistry, Duke University, Durham, NC 27710 USA*

Correspondence email: NWMoriarty@LBL.Gov

Starting from *Phenix* dev-1810 version, restraints across disulfide links have been implemented on one angle and two dihedrals, including handedness-dependent values for χ3 around the central SS bond. The angle is for C_β-S_γ-S_γ atoms on either side across the link, and the central dihedral restraint applies to C_β-S_γ-S_γ-C_β atoms. Current values in the Monomer Library were target 103.8° and esd 1.8° for the bond angle, and target 90.0° and esd 10.0° for the SS χ3 dihedral, but were not in use as restraints.

Small-molecule crystal structures documented both left-handed (Peterson *et al.* 1960) and right-handed (Oughton & Harrison 1959) conformations of the bridge, and there is an early protein survey in Richardson (1981). More recently, high-resolution protein structures show multiple rotamers and handedness-dependent SS dihedral values that systematically deviate from 90°. A survey of 1677 quality-filtered disulfides in the Top8000 dataset (Richardson *et al.* 2013) gives mean SS χ3 dihedral values of +93° ±11° and -86° ±9°, as shown in figure 1a. The χ2 values (which also span the link) have a broader but still useful distribution (figure 1b), with means +79° ±17°, +183° ±29°, and -73 ±17°. The C_β-S_γ-S_γ angle mean is 104.2° ±2.1°.

The restraint values have been updated accordingly, using the alternative-value syntax available in *Phenix* for the individual, non-periodic peaks of χ2 and χ3. The esd for χ3 was set at 10° and for χ2 at 20°.

A re-refinement of the `1ejg` crambin with these new targets reduced the χ3 deviations of the refined model from [15°, 9°, 3°] for its three disulfides to [12°, 5°, 1°]. These reductions are due to the new target values being closer to the high-res structure values. The bond and angle rmsds as well as the R factors are not significantly different.

These restraint value changes were introduced in dev-1950.

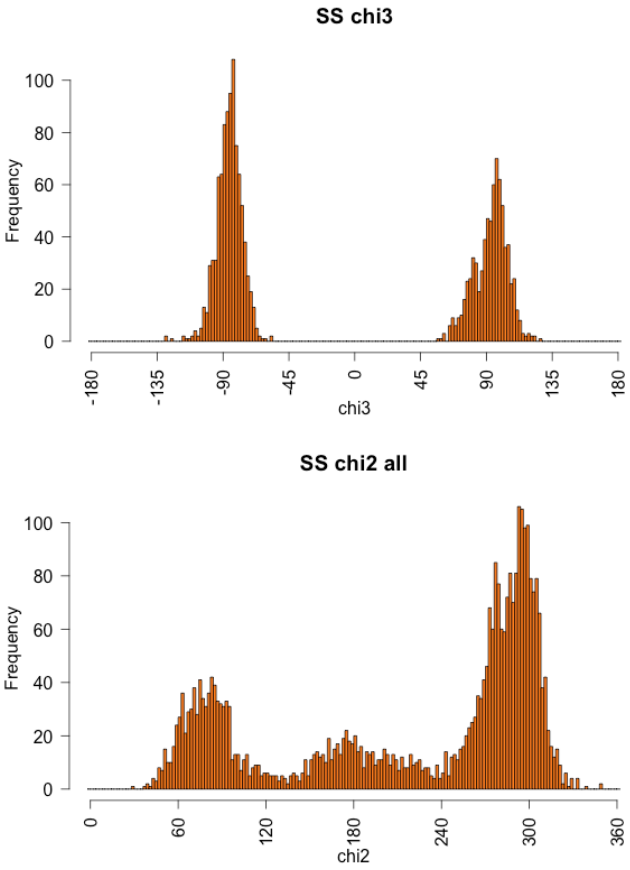


Figure 1: Histograms of the χ2 and χ3 values taken from the 1677 quality-filtered disulfide bridges.

References

Oughton BM, Harrison PM. *The crystal structure of hexagonal L-cystine*, Acta Cryst (1959) **12**: 396-404

Peterson J, Steinrauf LK, Jensen LH. *Direct determination of the structure of L-cystine dihydrobromide.* Acta Cryst (1960) **13**:104-109

Richardson JS. *Anatomy and Taxonomy of Protein Structures*, Adv Prot Chem (1981) **34**: 167-339

Richardson JS, Keedy DA, Richardson DC. *The Plot thickens: more data, more dimensions, more uses* (2013), pp. 46-61 in Biomolecular Forms and Functions: A Celebration of 50 Years of the Ramachandran Map, ed. Bansal M, Srinivasan N, World Scientific Publishing, Singapore, ISBN 978-981-4449-13-27

Improved Probabilistic Estimates of Biocrystal Solvent Content

Christian X. Weichenberger[a] and Bernhard Rupp[b,c]

[a]*Center for Biomedicine, European Academy of Bozen/Bolzano (EURAC), Viale Druso 1, I-39100 Bozen/Bolzano, Italy*
[b]*Department of Forensic Crystallography, k.-k. Hofkristallamt, 991 Audrey Place, Vista, CA 92084, USA*
[c]*Department of Genetic Epidemiology, Medical University Innsbruck, Schöpfstrasse 41, A-6020 Innsbruck, Austria*

Correspondence email: br@hofkristallamt.org, Christian.Weichenberger@eurac.edu

Introduction

When copies of biomolecular chains assemble into a crystal lattice, the molecules occupy a certain fraction of the available space, and the remainder is filled with solvent. As early as in 1968 Matthews (Matthews, 1968) has addressed the question concerning the distribution of fractional solvent in the crystallographic asymmetric unit. In his work, he defined the quantity V_M, nowadays known as the Matthews coefficient, as the fraction of the asymmetric unit volume V_A and the molecular weight M, $V_M = V_A/M$ and derived the equation for the solvent content $V_S = 1 - 1.230/V_M$. Analysis of 116 protein crystal structures has shown that solvent content ranged from 27% (almost spherical packing density) to 65%, with the most common value of approximately 43%. In the very early stages of structure determination, prior knowledge of the solvent content distribution allows estimating the number of molecules that can be present in the asymmetric unit. As suspected by Matthews (Matthews, 1976) and as demonstrated by Kantardjieff and Rupp in 2003 (Kantardjieff & Rupp, 2003), a distinct correlation of higher experimental resolution with lower solvent content exists. This dependency is accounted for in the Matthews probability (MP) calculator, *MATTPROB*, a web applet publicly available at www.ruppweb.org/mattprob to compute the oligomerization probabilities given the experimental resolution, unit cell parameters, crystal space group, and the macromolecule's weight. In this short communication we summarize the results from an update (Weichenberger & Rupp, 2014) ten years after the initial publication of the MP calculator (Kantardjieff & Rupp, 2003), describe the web interface, and present an alternative, non-parametric approach to compute the probabilities, which has become the default mode in the web interface.

Ten years of probabilistic solvent content estimates: an update

Motivated by Matthews' early studies on protein solvent content (Matthews, 1976), about ten years ago Kantardjieff and Rupp (Kantardjieff & Rupp, 2003) carried out a systematic analysis on more than 15,000 Protein Data Bank (PDB) (Bernstein *et al.*, 1977, Berman, 2008) entries determined by X-ray crystallography available at that time. The amount of protein structures was sufficient to statistically investigate the correlation between solvent content and molecular weight and experimental resolution. A weak tendency was recognized that V_M correlates to molecular weight, but experimental resolution was a much clearer discriminator of V_M: Protein structures that are packed more tightly tend to diffract better. This insight gave rise to the computation of Matthews probabilities based on the distribution of V_M conditional on experimental resolution. The approach relies arranging the resolution range of 1.2 Å to 3.5 Å for protein crystals in 13 bins, and separate, non-binned treatment of nucleic acid chains and protein/nucleic acid complexes. A bin is here defined as the resolution range from 0 Å (highest possible resolution) down to any of the 13 points for proteins or the full range of resolutions in the two non-binned cases. For each such bin, the distribution of V_M is parameterized by a modified extreme value function, which at the heart is a Gumbel (Gumbel, 1941) probability density function with additional scaling parameters. This empirical fit function serves as the analytical probability density function for probing the possible oligomerization number m of a query molecule with molecular weight M by reporting the probabilities of V_M as a function of $m \times M$, which by definition of V_M corresponds to values of V_M/m. The advantage of introducing resolution as a parameter in probability calculation has been demonstrated through examples where a resolution-agnostic

Table 1: Number of PDB homo-oligomers (column labeled "Nr.") by oligomerization number *n*, furnished with probability of occurrence P(*n*). From our dataset of 50,190 homo-oligomers we identified 30 distinct oligomerization numbers by grouping identical SEQRES records. Less than half of the entries are monomers but more than three quarters of the oligomers are monomers and dimers. These numbers may give a guide when estimating the oligomerization number from solvent content. Unsurprisingly, odd homo-oligomerization numbers are less frequent when compared to their even neighbors. For example, there are only four 11-mers, compared to hundreds of 10-mers and 12-mers.

n	Nr.	P(*n*)	*n*	Nr.	P(*n*)	*n*	Nr.	P(*n*)	*n*	Nr.	P(*n*)	*n*	Nr.	P(*n*)
1	21984	0.44	7	69	1.37×10^{-3}	13	6	1.20×10^{-4}	22	4	7.97×10^{-5}	44	2	3.98×10^{-5}
2	17143	0.34	8	783	1.56×10^{-2}	14	41	8.17×10^{-4}	24	44	8.77×10^{-4}	45	1	1.99×10^{-5}
3	2169	4.32×10^{-2}	9	51	1.02×10^{-3}	15	17	3.39×10^{-4}	28	11	2.19×10^{-4}	48	4	7.97×10^{-5}
4	5440	1.08×10^{-1}	10	199	3.96×10^{-3}	16	62	1.24×10^{-3}	30	6	1.20×10^{-4}	54	1	1.99×10^{-5}
5	373	7.43×10^{-3}	11	4	7.97×10^{-5}	18	14	2.79×10^{-4}	32	2	3.98×10^{-5}	56	1	1.99×10^{-5}
6	1390	2.77×10^{-2}	12	321	6.40×10^{-3}	20	45	8.97×10^{-4}	36	1	1.99×10^{-5}	60	2	3.98×10^{-5}

computation would have lead to a different favored oligomerization number.

It is important to note that the calculation of MPs assumes the Bayesian argument that the observed resolution represents an empirical lower limit of the true diffraction potential of the crystal: it has diffracted to at least the reported resolution, but in theory could have diffracted better. This is reflected in the definition of the resolution bins described above as they collect data from all structures with at least the resolution specified by the bin. Furthermore, the calculator reports probabilities, thus chances to find a different oligomerization state other than that associated with the highest probability are real. This happens if the crystal's V_M is different from the mode of the distribution.

In our follow-up work we have reexamined the statements from the 2003 publication and addressed several other questions that arose in the literature during the past decade. We followed the same data mining protocol as presented in (Kantardjieff & Rupp, 2003): from the initial set of 77,481 crystal structures we removed highly redundant entries and found 60,218 protein structures, 998 nucleic acid structures and 2,414 structures of protein/nucleic acid complexes. Of the 50,190 protein structures consisting of homo-oligomers we did not find any dependency of solvent content on oligomerization number, confirming previous findings reported by (Chruszcz *et al.*, 2008). A comprehensive list of oligomerization numbers and occurrences in PDB is given in Table 1.

We emphasize the importance of an accurate estimate of molecular weight *M*, as this becomes a sensitive parameter when investigating structures with an expected high number of oligomers. For this reason, in the web interface we have removed the potentially misleading species-dependent option to compute the molecular weight from sequence length (i.e., number of residues). Instead, links to compute the actual molecular weight from sequence are provided.

In principle the prior probability P(*n*) (*cf.* Table 1) could be used to always bias the resolution dependent prediction of oligomerization states *m* as P(*m*|n,res). However, very often a strong biological prior exists based on experimental knowledge that will lead to a corrected posterior estimate of the solvent content. Therefore, our calculator explicitly allows including such a strong biological prior by selecting a probable oligomerization state. We believe that an informed decision by the user to include a strong biological prior should override the automatic imposition of a generic P(*n*) prior[1].

[1] The effect of simply weighting the MP by P(*n*) can be tested by selecting the 2013 parameter set and sending the GET string to the server with *matt_prob_linux_pn* instead of default *matt_prob_linux*.

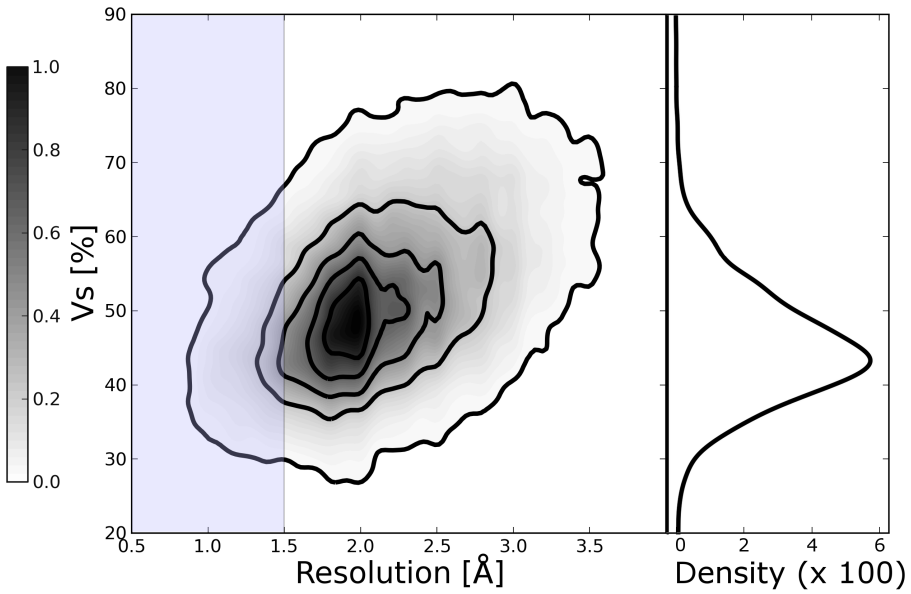


Figure 1: Parameter-free kernel density estimate of 60,218 pairs of solvent content V_S and resolution retrieved from PDB. In the middle part of the figure we show the two-dimensional kernel density estimate with an axis-aligned bivariate normal kernel of the full data set, normalized to have 1.0 as the maximum. A clear tendency towards lower values of V_S is visible for higher (better) resolution. The right hand side of the figure displays the one-dimensional probability density function of the highlighted region in the central figure that consists of approximately 6,200 protein structures resolved at a resolution of 1.5Å or better. We notice that the mode of this distribution is shifted towards higher crystal packing compared to similar distributions that include structures with lower resolution. This figure was generated with matplotlib (Hunter, 2007) and R (R Core Team, 2012).

We applied principal component analysis on the set of 50,190 homo-oligomeric protein structures using the observed variables resolution, molecular weight, and V_M. The analysis corroborates use of resolution as the most important and single variable to describe V_M, and minor, insignificant changes were observed when investigating the full data set of 60,218 protein structures. We repeated the parameter fitting process with the updated data sets and published the parameter set and its use in the web interface.

When comparing the distributions of V_M and V_S, we observed that V_S is a much better behaved distribution function in the sense that it is more symmetric and less tailed, indicated by a great reduction of sample skewness. This motivated us to favor V_S over V_M when constructing a parameter-free version of MP in the R programming language (R_Core_Team, 2012) by reconstructing the density function for pairs of resolution and V_S observed in the PDB with a two-dimensional kernel density estimator with an axis-aligned bivariate normal kernel (Wand, 1994, Wand & Jones, 1995). We arrive at the function $P_r(V_S) = P(V_S \mid \text{resolution} \leq r)$, i.e. the probability density function under the Bayesian assumption that the crystal has diffracted to at least resolution r, where resolution r has previously been observed in the PDB. Figure 1 presents the two-dimensional kernel estimate of P(V_S, resolution) with an example of its one-dimensional projection $P_r(V_S)$ for a fixed resolution r = 1.5Å.

The *MATTPROB* web interface has been updated with the new parameter set and the parameter-free version of the MP calculator. In Figure 2 we give a real world example taken from PDB entry 3vto, a tail-forming metal binding protein from bacteriophage Mu (Harada *et al.*, 2013) used to penetrate the host membrane during the infection process, which crystallized as a hexamer at resolution 1.44Å. Without using resolution as prior information, a pentamer would be the most likely oligomer. Using the kernel-estimated probability density function $P_{1,44}(V_S)$ with data arriving from 4,042 proteins determined at a resolution of 1.44Å or better, a hexamer is predicted as the most probable oligomer. Entering additional information that the tail protein

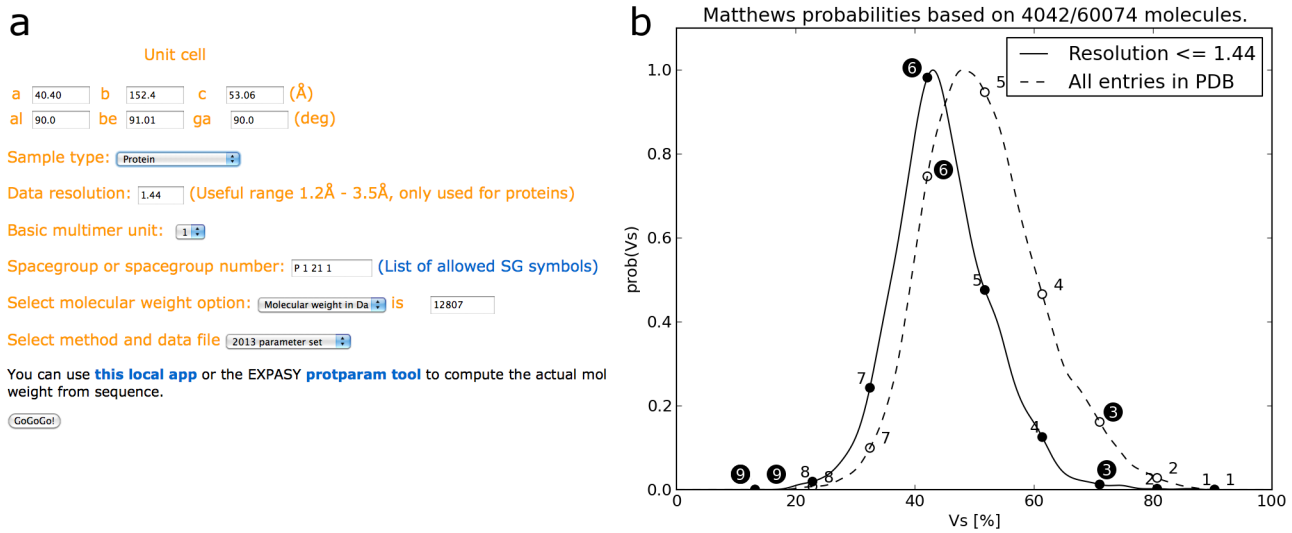


Figure 2: **Usage of the *MATTPROB* web applet. (a)** Web interface of *MATTPROB*, **www.ruppweb.org/mattprob**, filled with the unit cell parameters of PDB entry 3vto, where we have computed the molecular weight of the protein chain with 115 residues as M = 12,807 Dalton. The first row of values represents the unit cell axes lengths, and in the second row the respective angles alpha, beta, and gamma are input. The *Sample type* drop down menu offers three types of macromolecules: proteins, nucleic acids, and complexes of protein and nucleic acids. The input field *Data resolution* expects the experimental resolution in units of Ångströms for the query protein crystal. In the other two cases involving nucleic acids, resolution-dependent probability calculation is only available with the parameter-free kernel-based estimator and should be critically examined due to sparse data in PDB. In case there exists prior knowledge of functional oligomerization, the *Basic multimer unit* drop down menu allows restricting computation of probabilities to multiples of the specified chain(s). In the case of 3vto, this number could be set to three (see panel (b), black circled numbers, for the result). The Hermann-Mauguin space group symbol is input in the *Spacegroup or spacegroup number* field, followed to the right by a supporting link to a list of all allowed symbols and settings. The molecular weight M is supplied in the following line, and ultimately the MP calculation method is chosen by *Select method and data file*, where either the 2003 or 2013 parameterized version or the new 2013 kernel-based estimator can be chosen, the latter is set as the default method. Computation of MPs is initiated pressing the *GoGoGo!* button and after a few seconds the results are presented on a separated page. The underlying *MATTPROB* program using the kernel density estimator has been implemented in the Python programming language and utilizes *matplotlib* (Hunter, 2007) as its plotting backend. The parameterized versions of *MATTPROB* have been implemented using the original 2003 FORTRAN code. **(b)** MP graph. The summary graph shows possible values of V_S and associated probabilities for the resolution-dependent density function as well as for the resolution-agnostic version. The title of the plot informs about the number of PDB entries used for computing the probability density function, and it should be underlined that the higher the resolution of the query crystal, the lower the number of PDB entries found, i.e. fewer data points have been available for density estimation. Only the numbers in black are output when selecting a trimer as basic biological unit in the *Basic multimer unit* drop down menu. In the graphs, the probability density functions are always normalized to have a maximum value of 1.

assembles as a trimer then points to a hexameric structure in the ASU, excluding the possibility of a single trimer and the pentamer (Figure 2b, dark circles around predicted oligomerization number).

Example: Implementation of the *MATTPROB* kernel estimator in R

Many programming languages come with scientific libraries that support kernel density estimations. For the sake of brevity and simplicity, we demonstrate computation of a resolution dependent kernel density estimate in the statistical programming language R. From our website, www.ruppweb.org/mattprob, the three data files for protein, nucleic acids, and protein/nucleic acids complexes, respectively, can be downloaded. The R code below reads in the comma separated data file for protein crystal structures, and filters for PDB entries with resolution better or equal to 1.44 Å (variable `maxRes`). We construct a kernel density estimate

```
library("KernSmooth")
max.res   = 1.44
vs        = 51.77
dat       = read.csv("pdb_02_06_2013_pro_sorted_flagged_highest_cs.csv")
trunc.dat = dat[ dat$reso<=max.res, ]
dens      = bkde(trunc.dat$vs, gridsize=500)
i         = max(which(dens$x<=vs))
vs.x      = dens$x[i]
dens.y    = dens$y[i]
xlim      = c(0, 100)
ylim      = c(0, 0.06)
plot(dens, xlab="Vs [%]", ylab="Density", type="l", xlim=xlim, ylim=ylim)
par(new=TRUE)
plot(vs.x, dens.y, xlim=xlim, ylim=ylim, xlab="", ylab="", axes=FALSE)
```

Schema 1: Code fragment for implementing a fully functional *MATTPROB* calculator.

of the probability density of the solvent content from these truncated data using the `bkde` call from the previously imported `KernSmooth` library. The result is stored in the list `dens`, with abscissa and ordinate values in `dens$x` and `dens$y`, respectively. We then find the abscissa index `i`, which corresponds to the solvent content closest to the query variable `vs`. (The value `vs`=51.77 is retrieved via Matthews' formula given in the Introduction for a pentamer for the above example PDB entry 3vto, see also Figure 2b.) Finally, we plot both the density estimate curve and the calculated point.

With the code fragment in schema 1, the interested reader can easily implement a fully functional *MATTPROB* calculator. We would like to point out that data files for nucleic acids and protein/nucleic acids complexes have a very limited number of entries, and therefore high quality resolution-dependent predictions cannot be expected for these types of macromolecules.

Conclusions

We have reinvestigated possible variables for predicting the most probable oligomer given the crystallographic unit cell dimensions, space group, and molecular weight. We found that experimental resolution is still the most powerful discriminatory variable for predicting solvent content from known protein structures. We have updated the parameter set to reflect the state of PDB in 2013. An MP calculator based on a parameter-free kernel density estimate of the solvent content probability density has been implemented and is the default mode of Matthews probability computation in the web interface at www.ruppweb.org/mattprob. A major advantage of this method is that it is free of any previously used binning approach and can be queried with any resolution currently found in the PDB. Finally, we have sketched the central *MATTPROB* calculation function in the R programming language giving a starting point for independent implementations. Supporting data files are available for download at www.ruppweb.org/mattprob/kernel_data_tables_2013.zip.

Acknowledgements

BR acknowledges support from the European Union under a FP7 Marie Curie People Action, grant PIIF-GA-2011–300025 (SAXCESS). The web site www.ruppweb.org is supported by the k-k. Hofkristallamt, Vista, CA 92084.

References

Berman, H. (2008). *Acta Crystallogr.* **A64**, 88-95.

Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Jr., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *J Mol Biol* **112**, 535-542.

Chruszcz, M., Potrzebowski, W., Zimmerman, M. D., Grabowski, M., Zheng, H., Lasota, P. & Minor, W. (2008). *Protein Sci* **17**, 623-632.

Gumbel, E. J. (1941). *Ann. Math. Statist.* **12**, 163-190.
Harada, K., Yamashita, E., Nakagawa, A., Miyafusa, T., Tsumoto, K., Ueno, T., Toyama, Y. & Takeda, S. (2013). *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics* **1834**, 284-291.
Hunter, J. D. (2007). *Computing In Science and Engineering* **9**, 90-95.
Kantardjieff, K. A. & Rupp, B. (2003). *Protein Sci* **12**, 1865-1871.
Matthews, B. W. (1968). *J. Mol. Biol.* **33**, 491-497.
Matthews, B. W. (1976). *Ann Rev Phys Chem* **27**, 493-523.
R Core Team (2012). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.
Wand, M. P. (1994). *J. Computational Graphical Statistics* **3**, 433-445.
Wand, M. P. & Jones, M. C. (1995). *Kernel Smoothing*. Boca Raton, Florida: Chapman and Hall/CRC Press.
Weichenberger, C. X. & Rupp, B. (2014). *Acta Crystallogr.* **D70**, 1579-1588.

Rapid Evaluation of Non-Bonded Overlaps in Atomic Models

Youval Dar,[a] Nigel W. Moriarty,[a] Jeffrey J. Headd[b], Jane S. Richardson[c], David Richardson[c], and Paul D. Adams[a,d]

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
[b]*Janssen Research & Development, LLC, 1400 McKean Road, Spring House, PA 19477*
[c]*Biochemistry Department, Duke University Medical Center, Durham, NC 27710*
[d]*Department of Bioengineering, University of California at Berkeley, Berkeley, CA 94720*

Correspondence email: ydar@lbl.gov

Introduction

Detecting and evaluating steric overlaps between non-bonded atoms is an important tool in the process of improving protein and nucleic acid structure models. Overlaps may arise between atoms of a single monomer or protomer, between solvent molecules or due to interactions between symmetry related atoms. *MolProbity* (Davis *et al.*, 2004), a suite of tools for structure validation, produces a *clashscore* – the number of steric overlaps of the electron cloud per 1000 atoms. In *MolProbity,* the overlaps of atoms electron density or clashes are evaluated by a rolling-probe algorithm (Word *et al.*, 1999). This is achieved by rolling a 0.5Å diameter sphere on the Van Der Waals (VdW) surface and recording a clash when VdW surface of non-bonded atoms overlap is ≥ 0.4Å. An overlap typically indicates local model stereochemistry issues. *MolProbity* performs a very comprehensive all-atom contact analysis (Chen *et al.*, 2010) and is an integral part of the model validation in the *Phenix* suite of programs (Adams *et al.*, 2010). However, there is a need to rapidly and repeatedly calculate non-bonded overlaps during structure refinement with phenix.refine (Afonine *et al.*, 2012), to report the current quality of the model at each refinement step.

Therefore, a tool has been developed in the *CCTBX* toolbox (Grosse-Kunstleve et al., 2002) that makes it possible to perform a rapid, but simplified analysis of overlapping atoms during the model refinement process. This algorithm makes some approximations compared to the more sophisticated approach used in *MolProbity*, but reports the overlaps due to crystal symmetry which are not detected currently by the rolling-probe algorithm as implemented in *MolProbity*. This CCTBX non-bonded overlaps (NBO) analysis is used to report on atomic overlaps during structure refinement, while the comprehensive structure refinement performed after refinement makes use of the more detailed *MolProbity* analysis.

In the process of model refinement *Phenix* analyzes all non-bonded interactions, including interactions between symmetry related copies. This process results in information on non-bonded interacting atoms pairs that is readily available for calculating the non-bonded overlaps. This method allows easy filtering of the overlaps into groups such as the total number of overlaps, overlaps due to symmetry operations and overlaps in the macromolecule (protein, DNA or RNA). Additional breakdowns can be easily added if the need arises using the selection mechanism available in *Phenix*.

In contrast to *MolProbity* (the rolling-probe algorithm), the *CCTBX* overlaps are evaluated from the difference between the model non-bonded atomic distance and the calculated VdW distance. The non-bonded overlaps (NBO) is therefore a count of the number of overlaps. A normalized NBO can be calculated by dividing by the number of atoms. Multiplying by 1000 places the normalized NBO on a similar scale to the *MolProbity clashscore.* However, since the methods for finding overlaps and clashes are different, the NBO count cannot be directly compared to the

MolProbity clashscore. The *CCTBX* NBO does not provide any graphical visualization and it is not intended to replace the *MolProbity clashscore* but to provide additional indicator for model quality during structure refinement.

Method and usage

Evaluation of the *CCTBX* non-bonded *clashscore* is performed as follows:

1. Create a *Geometry Restraints Manager* object (Grosse-Kunstleve *et al.*, 2002) and extract the non-bonded proxies (non-bonded atoms pairs list). While no explicit secondary structure or other parameters related to Hydrogen bonds are provided, the *Geometry Restraints Manager* identifies most of H-Bonds donor-acceptor pairs and adjust the Van der Waals distance to reflect hydrogen bonding.
2. Collect all overlapping non-bonded atoms where an overlap is defined as:

   $$R_{nonbonded} - R_{VdW} \leq 0.4\text{Å}$$

   where $R_{nonbonded}$ is the distance between the atoms and is the sum of the Van der Waals radii.
3. *Heavy Atoms – Hydrogen* overlaps are omitted when they are less than 5 covalent bonds apart.
4. Inline overlaps (eclipsing): In the situation where two bonded atoms are overlapping with the same non-bonded atom, there are two different situations. If all atoms are inline, an overlap is be considered as a single overlap even if the there is unacceptable overlap between all three atoms. If the non-bonded atom perpendicular to the bond of the bonded atoms and overlaps with both, it is considered two overlaps. This situation is illustrated in figure 1. Atoms are considered to be inline when the *abs(α)<45°*, where α is defined (see figure 1) as the angle between the X-H bond and the line from the center of the X-H bond and the second atom, Y. The value of *α* was chosen to be the inline limit based on manual inspection.
5. Eliminate double counting of clashes related to symmetry operation, since each symmetry overlap appears twice in the non-bonded overlapping atoms pairs list.

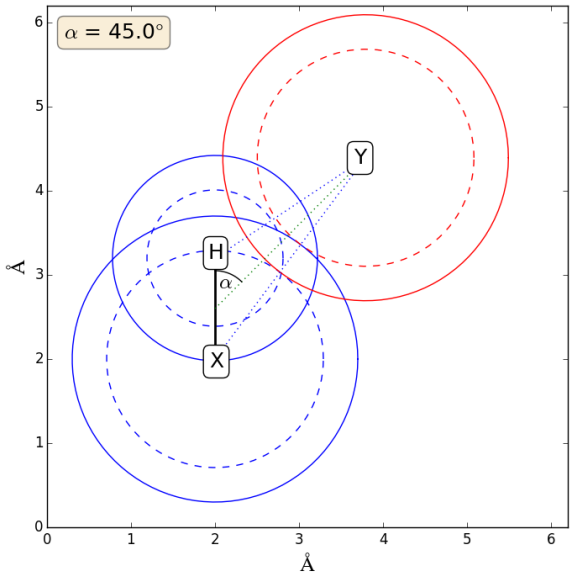


Figure 1: Collision between X-H and Y atoms. The solid circles are the VDW radii. The dash circles are the radius of allowed overlap. When the overlap exceeds the dashed circle it considered to be a clash. The angle α is the limit of inline, ecplipsed, overlap.

Currently three NBO values are provided:

1. All NBO: *number of ALL unique overlaps*.
2. Symmetry related NBO: *number of all overlaps due to symmetry*.
3. Macro molecule (protein, RNA or DNA) NBO: *number of overlaps in macro molecule, excluding symmetry related NBO*.

For comparison purposes, normalized NBO values were calculated for this work. One can obtain the normalized NBO as follows: *1000 x (number of unique clashes) / (number of atoms in the model)*. Note that this is the number of atoms in model, not necessarily the complete asymmetry unit of the structure. Note, also, that both the NBO value and the number of atoms are generally different from the number of clashes and the number of atoms used to calculate *MolProbity clashscore*. Differences include atoms with partial occupancy, solvent – solvent clashes, definition of bonded and non-bonded, and the inclusion of solvent atoms in model atom count.

When evaluating the normalized NBO, the number of atoms used to normalize is as follows:

1. All NBO in model:
   1000 x (number of ALL unique overlaps) / (number of ALL atoms in the model)
2. Symmetry related NBO:
   1000 x (number of overlaps due to symmetry) / (number of ALL atoms in the model)
3. Macromolecule NBO:
   The model considered for this include only protein, RNA and DNA
   1000 x (number of overlaps, excluding symmetry related) / (number of atoms in the macro molecule)

Command line usage

The non-bonded overlaps (NBO) can be calculated using the command-line script

> mmtbx.nonbonded_overlaps xxxx.pdb [options]

Because the NBO algorithm relies on both the VDW distance and bonding information, the presence of nonstandard residues requires a restraints CIF be supplied:

> mmtbx.nonbonded_overlaps xxxx.pdb xxxx.ligands.cif

A restraints *CIF* file can be obtain using *ReadySet!* or *eLBOW (Moriarty et al., 2009)*. A code implementation example is given in the appendix.

When used from the command line, if the model contain no Hydrogen atoms, Hydrogen are added using *phenix.reduce* with the following options:

1. Add hydrogens on OH and SH group (-oh)
2. Create NH hydrogens on HIS rings (-his)
3. Add H and rotate and flip NQH groups (-flip)
4. Fraction of std. bias towards original orientation (-pen9999)
5. Keep bond lengths as found (-keep)
6. Process adjustments for all conformations (-allalt)

PDB NBO Survey

To test the NBO and in particular analyze the symmetry-related NBO in the Protein DataBank (Berman *et al.*, 2000; Bernstein *et al.*, 1977), the NBO program was run on PDB files (based on the index of PBD structures at the Lawrence Berkeley Lab PDB mirror on Jan. 21 2015) filtered to contain a single model, no unknown residues and valid CRYST1 crystal symmetry records. This resulted in 50,646 models for investigation.

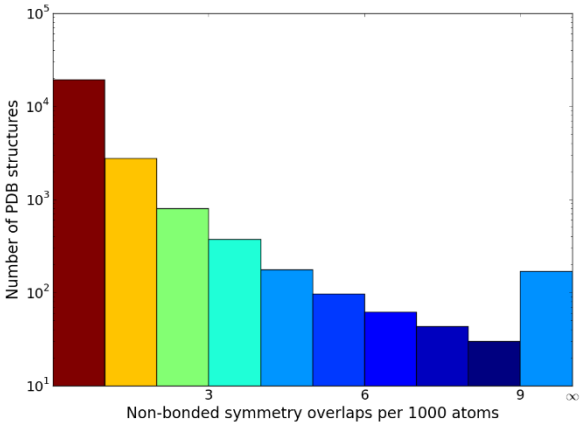


Figure 2: Macro molecule (models include only protein, RNA or DNA) normalized symmetry-related overlaps for 23,685 structures were MolProbity clashscore is less or equal to three and PDB structures with good crystal symmetry, a single model and no unknown atom pairs.

Recent work (Moriarty *et al.*, 2014) that investigated the relationship of the rmsd of the N-C_{α}-C angle in 25,976 refined structures and the starting *MolProbity clashscore* showed that structures with a *clashscore* greater than six failed to exhibit the expected reduced rmsd values for low resolution structures. In fact, if structures with *clashscore* values greater than six are included, the average rmsd for the N-C_{α}-C angle (and to a lesser extent all angles) was larger for structures in the 3.0-3.5Å range than at 2.0Å. This is likely due to regions of the model that are outside the radius of convergence of the refinement method and contribute to the higher *clashscore* value. A more conservative filtering choice for the *clashscore* of less than three removes the majority of models that have any poor quality regions. Filtering for *clashscore* less than three resulted in 23,685 entries to calculate NBO statistics (figure 2), the

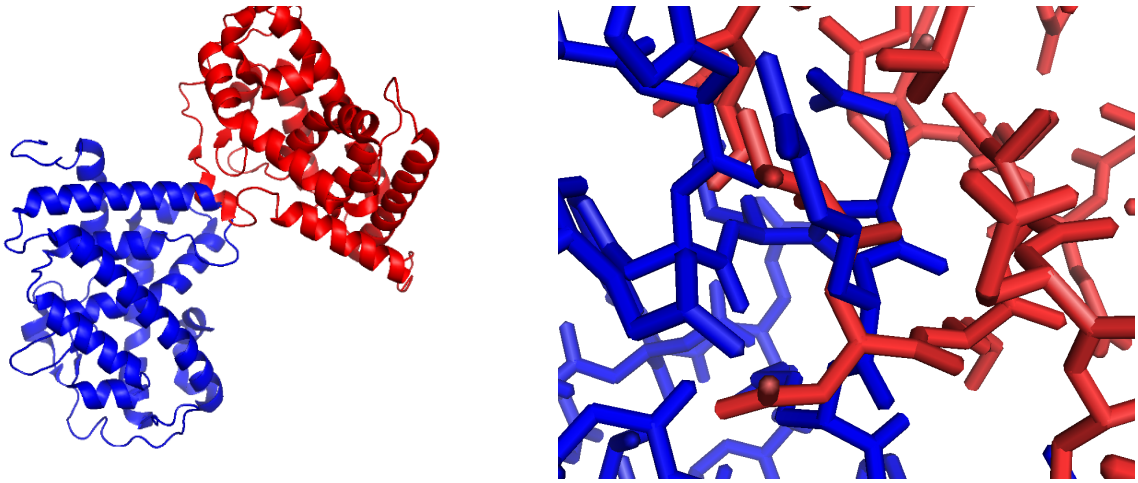


Figure 3: Overlap due to symmetry operation in 2H77 (DOI:10.2210/pdb2h77/pdb). On the right, a close up on the overlapping region.

histogram of normalized symmetry-related NBO. The symmetry-related NBO of the macromolecule counts overlaps that are not included in the *MolProbity clashscore*. If the models were uniform in quality, then the number symmetry-related overlaps should be less than the overlaps internal to the model. This is partly because the quality should be the same throughout the model and partly because there are fewer opportunities for overlaps on the model surface. Two reasons for the latter are that there are fewer atoms on a surface compared to the internal volume and the fact that less than the entire surface interacts with a symmetry surface. Therefore, it can be expected that the normalized symmetry-related NBO should be smaller than the upper limit of three chosen for the *clashscore*. However, the results of the NBO analysis (figure 2) show that there are a significant number of structures with elevated normalized symmetry-related NBO.

A striking example of an extreme symmetry-related overlap is shown in figure 3. This example, 2H77 (DOI:10.2210/pdb2h77/pdb), has a resolution of 2.33Å and was deposited in 2006.

Figure 4 explores symmetry-related overlaps over time — it shows the percent of the filtered structures deposited each year in the PDB that have a normalized symmetry NBO below the thresholds listed in the legend. The total number of deposited models is displayed in the bottom portion. It is clear that among the 50,646 suitable structures tested, there is

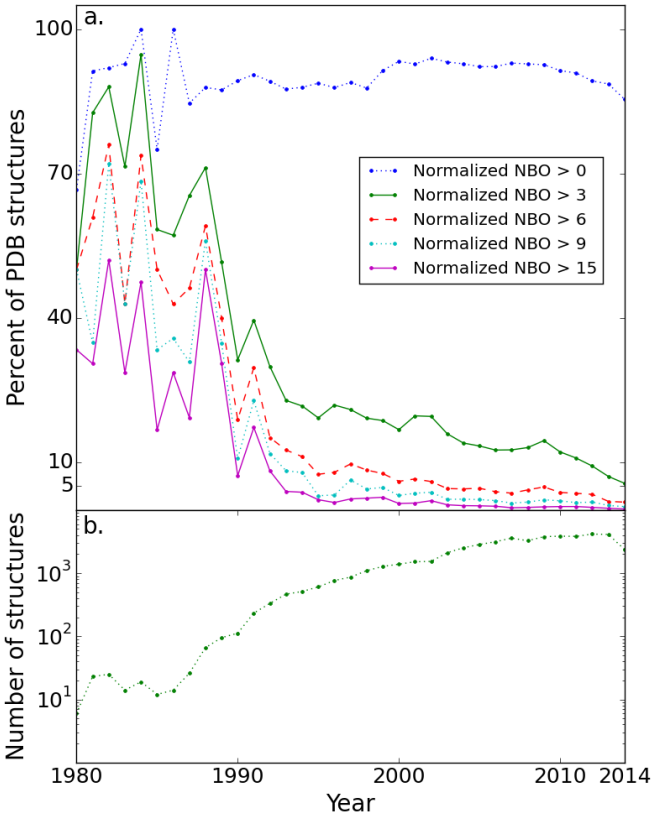


Figure 4: (a) Percent from the total structure number of structures with symmetry related NBO per 1000 atom larger than 0,3,6,9,15. (b) The number of structures with a single model, good CRYST1 records and no unknown atoms types vs. PDB deposition year. Total of 50,646 structures data points.

a notable improvement in the number of symmetry overlaps. This can be attributed to improved model validation during the PDB deposition process, improved structure refinement algorithms, and the use of advanced validation tools such as *MolProbity*. However, in 2014 85% of tested structures still had non-zero symmetry NBO values. Furthermore, considering a desirable value for a normalized total NBO to be less than three, in 2014 132 structures (5.5%) were worse than this.

Summary

A tool has been developed to rapidly evaluate non-bonded overlaps (NBO) in protein and nucleic acid structure models during structure refinement with phenix.refine. These NBO include symmetry related overlaps, and are calculated using VdW bond lengths and model distances. An analysis of the PDB revealed a significant number of structures with an elevated symmetry-related NBO, emphasizing the need to account for these kinds of interactions during structure refinement and ultimately validation. The combination of MolProbity and the *CCTBX* NBO provides Phenix with a comprehensive and complementary set of tools for efficient refinement and validation of models.

The NBO information is not currently available in the Phenix GUI but will be made available in the near future.

References

Adams, P. D., Afonine, P. V., Bunkoczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L. W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). *Acta Crystallogr D* **66**, 213-221.

Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H. & Adams, P. D. (2012). *Acta crystallographica. Section D, Biological crystallography* **68**, 352-367.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res* **28**, 235-242.

Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Jr., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *J Mol Biol* **112**, 535-542.

Chen, V. B., Arendall, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S. & Richardson, D. C. (2010). *Acta Crystallogr D* **66**, 12-21.

Davis, I. W., Murray, L. W., Richardson, J. S. & Richardson, D. C. (2004). *Nucleic Acids Research* **32**, W615-W619.

Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J Appl Crystallogr* **35**, 126-136.

Moriarty, N. W., Grosse-Kunstleve, R. W. & Adams, P. D. (2009). *Acta Crystallogr D* **65**, 1074-1080.

Moriarty, N. W., Tronrud, D. E., Adams, P. D. & Karplus, P. A. (2014). *Febs Journal* **281**, 4061-4071.

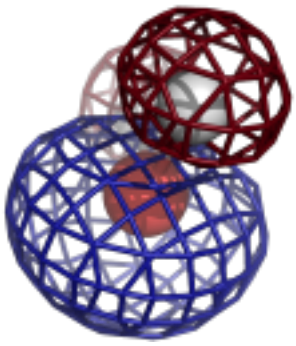
Word, J. M., Lovell, S. C., LaBean, T. H., Taylor, H. C., Zalis, M. E., Presley, B. K., Richardson, J. S. & Richardson, D. C. (1999). *Journal of Molecular Biology* **285**, 1711-1733.

Appendix

Code implementation example

```
import mmtbx.monomer_library.pdb_interpretation as pdb_inter
import cctbx.geometry_restraints.nonbonded_overlaps as nbo
from libtbx.utils import null_out
from libtbx.utils import Sorry
#
pdb_processed_file = pdb_inter.run(
   args=files,
   assume_hydrogens_all_missing=False,
   hard_minimum_nonbonded_distance=0.0,
   nonbonded_distance_threshold=None,
   substitute_non_crystallographic_unit_cell_if_necessary=False,
   log=null_out())
# test that CRYST1 records are ok
 sps = pdb_processed_file.all_chain_proxies.special_position_settings
 if not sps: raise Sorry('Bad CRYST1 records')
#
grm = pdb_processed_file.geometry_restraints_manager()
xrs = pdb_processed_file.xray_structure()
macro_mol_sel = nbo.get_macro_mol_sel(pdb_processed_file)
#
nb_overlaps = nbo.info(
   geometry_restraints_manager=grm,
   macro_molecule_selection=macro_mol_sel,
   sites_cart= xrs.sites_cart(),
   site_labels= xrs.scatterers().extract_labels(),
   hd_sel= xrs.hd_selection())
nb_overlaps.show()
```

VOLUME SIX



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JULY MMXV

PHASER GUI, ALT. LOCS, BASE-PAIR STACKING

Table of Contents

- Editor's Note 26
- Phenix News 27
- Crystallographic meetings 28
- Expert Advice
  - Fitting tips #10 – How do your base pairs touch and twist? 28
- FAQ
- Short Communications
  - New Phaser-MR search panel in Phenix 32
  - Quantum chemical techniques for minimising ligand geometries in the active site 35
  - 13 typical occupancy refinement scenarios and available options in *phenix.refine* 37
- Articles
  - A context-sensitive guide to RNA & DNA base-pair & base-stack geometry 47

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

Editor's Note

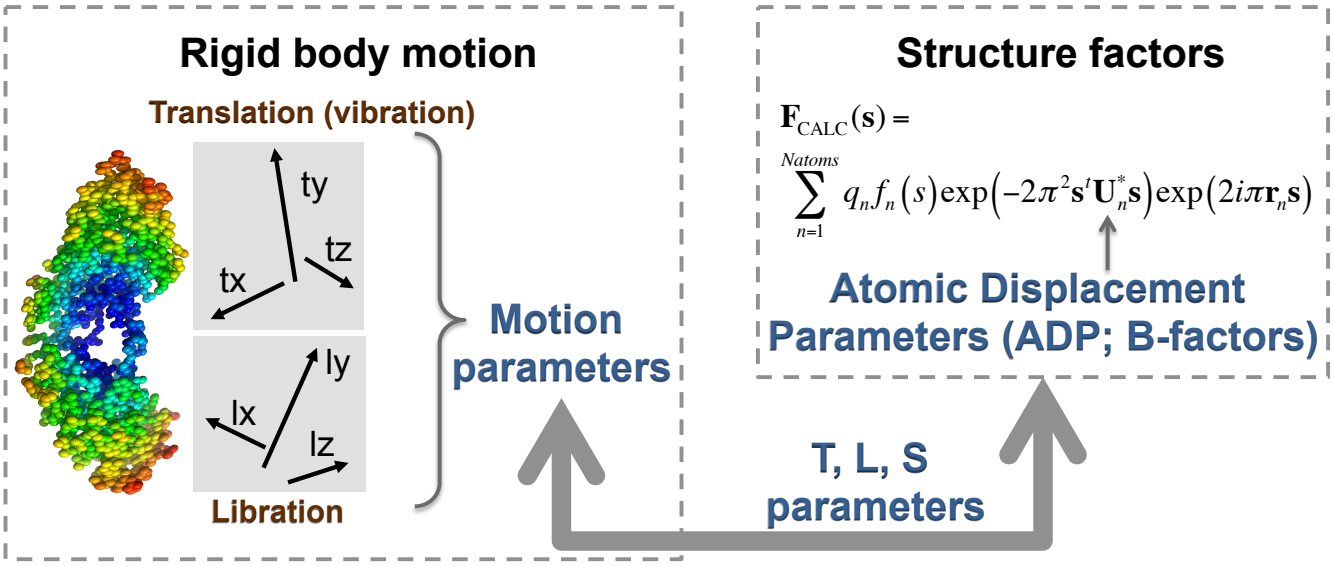
Crystallography makes use of identifiers to access a database of information. The most ubiquitous is the four-digit code used to identify an entry in the Protein Databank known as the PDB id. The first digit is numeric but the remaining three digits are alphanumeric. Alphanumeric digits are also used for all digits of the chemical component identifiers use to identify the protein residues, RNA/DNA bases, ligands and more.

The use of alphanumeric digits can lead to confusion when reading these identification codes stemming from the similarity between the numeral 0 (zero) and the letter O (15th letter of the alphabet) in most typefaces used in scientific literature. The same applies to the numeral 1 (one), the letter I (ninth letter of the alphabet) and the letter l (lowercase of the twelfth letter). This is particularly true in small fonts and sans serif typefaces.

This confusion has led this publication to implement a policy to make these codes more human readable. In all cases, the letters mentioned are to be represented in the appropriate case (o,i,L). This can be used in an otherwise uppercase or lowercase mode with the knowledge that there will be no confusion between similar digits. Examples of codes from the PDB are shown in table A.

Table A: Examples of human readable PDB codes compared with standard representations.

Standard		Human readable	
Uppercase	Lowercase	Uppercase	Lowercase
1OI0	1oi0	1oi0	1oi0
1IJJ	1ijj	1iJJ	1ijj
4OCL	4ocl	4oCL	4ocL



Rigid body motion is described by three vibration axes and three vibration amplitudes along these axes, three libration (oscillation) axes and three libration amplitudes about these axes. TLS model can be used to describe rigid-body motion in terms of individual Atomic Displacement Parameters (ADP; B-factors) thus making possible to include this information into structure factor calculation.

Phenix News

Announcement

All programs that rely on the restraints library default to using the Conformation-Dependent Library proposed by Karplus and others (Tronrud et al. 2010; Moriarty et al. 2014). This includes all types of refinements and, to be consistent, validation automatically detects the appropriate restraints library to use. The CDL is the default in Phenix version 1.10.

References

Moriarty, N.W., D.E. Tronrud, P.D. Adams, and P.A. Karplus. 2014. FEBS J. 281 (18): 4061–71.

Tronrud, D.E., D.S. Berkholz, and P.A. Karplus. 2010. Acta Cryst. D 66 (7): 834–42.

Parameter organisation

Starting with Phenix 1.10 several parameter scopes were changed. This should not affect new projects. However, when restoring jobs from previous versions, some parameters may not be restored properly. In this case one needs to select them again in the GUI. Command-line users should check the updated documentation and correct their scripts accordingly. Affected functionality includes secondary structure restraints, NCS restraints, reference model restraints and C-beta restraints.

New programs

TLS analysis and validation

`phenix.tls_analysis` is a new program for validation of refined TLS parameters based on the recent article by Urzhumtsev *at al.* (2015) entitled "From deep TLS validation to ensembles of atomic models built from elemental motions."

In a nutshell, TLS parameters are a way to pack descriptors of rigid-body motion of a group of atoms into a form suitable to calculate structure factors (see figure above). While refined elements of ***T***, ***L*** and ***S*** matrices contain information about concerted motion of corresponding atoms, the rigid-body motion descriptors themselves, such as the vibration and libration axes and amplitudes, are not readily available from TLS matrices. This information needs to be extracted from TLS matrices in order to be interpreted.

We developed a mathematical procedure that interprets TLS matrices in terms of elemental motions that these matrices fundamentally describe. Decomposing TLS matrices into underlying motions is only possible if these matrices satisfy a set of certain criteria that we formulated. To illustrate the impact of this work, we applied these criteria to all PDB structures that have TLS matrices available. To our surprise, the TLS parameters in about 80% of these structures cannot be interpreted in terms of atomic motions and therefore do not make physical sense within the TLS paradigm.

The result of this work is tri-fold. First, we developed a tool that provides a simple interpretation of TLS matrices in terms of molecular motions. Second, this tool performs a comprehensive validation of TLS matrices. Third, we suggest a reformulation of TLS refinement methods and the corresponding programs to avoid these problems in the future as well as to correct existing problems.

References

Urzhumtsev, A., P. V. Afonine, A. H. Van Benschoten, J. S. Fraser and P. D. Adams. 2015. Acta Cryst. D 71, 1668-1683.

More TLS

`phenix.tls_as_xyz` is another software outcome of the Urzhumtsev *et al.* article (2015) from the previous note. This new program provides an explicit interpretation of refined TLS parameters by decomposing them into elemental rigid-body motions and generating structural ensembles that are consistent with these motions.

Map comparisons

`phenix.map_comparison` is a program that implements some of the ideas described in Urzhumtsev *et al.* (2014). Specifically, for the two input maps it calculates Peak Correlation (CC_{peak}) and Discrepancy Function (D-function) for varying map contouring thresholds. These tools offer at least complimentary and at most better map comparison instruments compared to traditional map correlation coefficient. Also, the program reports cumulative distribution function for both maps that allows choosing meaningful map contour thresholds for comparison of two maps as described in section 2.9 and figure 16 of Afonine *et al.* (2015), the feature enhanced maps (FEM) paper.

References

Afonine, P.V., N.W. Moriarty, M. Mustyakimov, O.V. Sobolev, T.C. Terwilliger, D. Turk, A. Urzhumtsev and P.D. Adams. 2015. Acta Cryst. D 71, 646-666.

Urzhumtsev, A., P.V. Afonine, V.Y. Lunin, T.C. Terwilliger and P.D. Adams. 2014. Acta Cryst. D 70, 2593-2606.

Crystallographic meetings and workshops

The 29th European Crystallographic Meeting, August 23-28, 2015

Location: Rovinj, Croatia. Representatives from Phenix and collaborators will be presenting.

CCP4 Study Weekend, 9-10 January 2016.

Location: East Midlands Conference Centre, Nottingham. The topic of "Protein-Ligand Complexes: Understanding Biological Chemistry".

Expert advice

Fitting Tip #10 – How do your base pairs touch and twist?

Jane Richardson, *Duke University*

Nucleic-acid base pairs, either Watson-Crick or non-canonical, are hydrogen bonded and stack with one another or with other aromatics. The textbook view shows them as coplanar, but that is only approximately true. As a crystallographer, you can fit RNA and DNA more easily if you know what to expect from some of their further subtleties that are summarized and illustrated here and in the accompanying longer article (Richardson, 2015). Each of the various helical forms has its own typical pattern of base relationships,

moderately but significantly distinctive. Even stronger variation occurs for bases or base pairs in the junction, bulge, loop, and interaction regions critical to forming complex tertiary structures and to catalytic function or binding specificity.

The strongest restraint, unsurprisingly, is on planarity of the individual bases. Even modified bases (e.g., methylated, Y base, etc.) are still mostly aromatic and planar, except for a few saturated, puckered rings such as the dihydro-U that is the hallmark of the D loop in tRNAs. Next strongest is base stacking: whenever possible, bases or base pairs that are near to parallel will in fact be quite parallel and at quite ideal vdW distance across their contact, as seen for a regular A-RNA stem in figure 1A.

Next in line is the effect of the base-pair H-bonds, which generally keep quite good H-to-acceptor distances (wide "pillows" of overlap, as seen in figure 1B) but have enough flexibility in angle to accommodate the average propeller twists of -11° to -15° (left-handed) seen in A-RNA, B-DNA, and A-DNA double helices. Figure 1B looks along the base planes of the strand on the right, so the relative twist of the left-strand bases is easily visible even without the stereo. This twist is a result of the constraints of the handed backbone conformation that produces this favorable, repeating arrangement. Other helices produce different average twists: Z-DNA has near-zero propeller twist and parallel poly-A RNA duplex has about +11° (right-handed) twist; both are illustrated in the accompanying article. As seen in figure 2, the 4-fold helix of a G-quadruplex has fairly flat layers on average, with no consistent direction of the low twist.

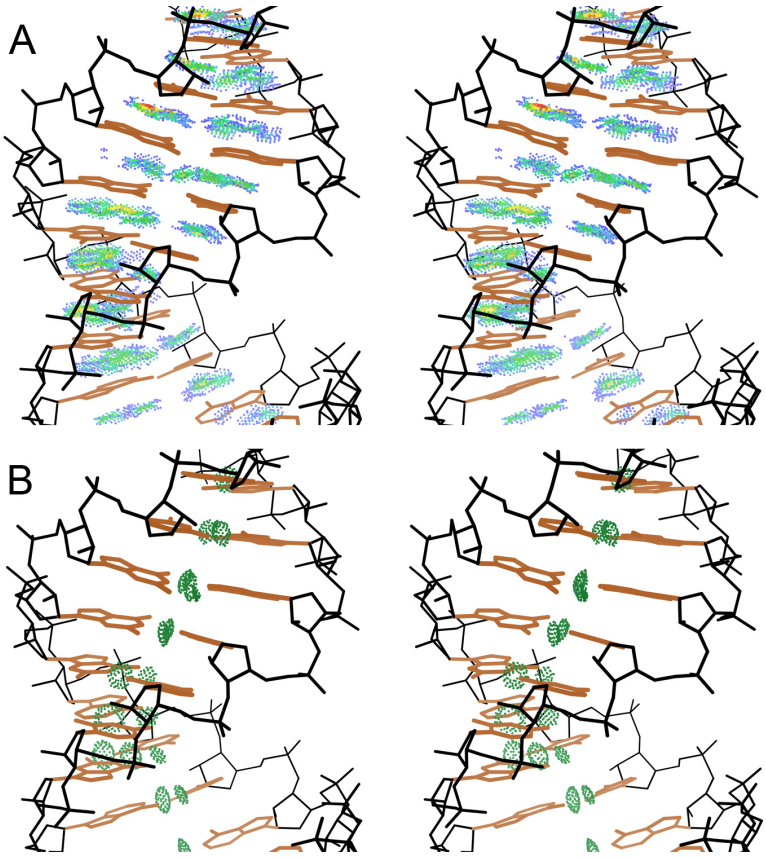


Figure 1: Base-pair stack and twist of regular A-form RNA in a lysozyme aptamer. (A) All-atom contact dots show the large, flat areas of good van der Waals stacking between base pairs, which also provides good π–π interaction. (B) Dot pillows show the base-pair H-bonds, well formed despite an average propeller twist of -14°. From 4M4o at 2.0Å resolution (Malashkevich 2013).

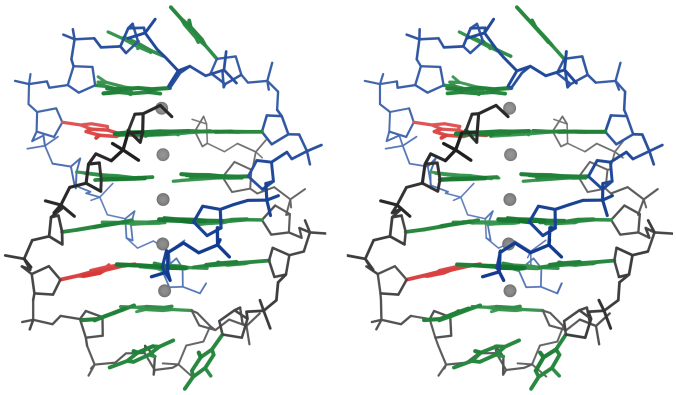


Figure 2: The 2HBN telomeric DNA G-quadruplex at 1.55Å (Gill 2006), with four layers of four H-bonded G bases and ions (Thallium here) between the layers. The two central layers are fairly flat, but in the two outer layers, the base (red) that leads into the capping loop twists about -15° relative to one neighbor base and buckles about 15° relative to the other neighbor.

In contrast to the significant but moderate twist found in repeating helical nucleic-acid forms, at the end or outside of such helices base pairs or adjacent bases can depart much more from parallel. For example, in the G-quadruplex of figure 2, the base (in red) on the strand that leaves one of the outer layers to start the loop is strongly angled. It still doubly H-bonds with both neighbor bases, but twists about -15° relative to one neighbor and buckles (bends across the line of the H-bonds) about 15° relative to the other. Along the top and bottom loops, successive bases do not stack well, but instead angle nearly 30° and touch only at one edge.

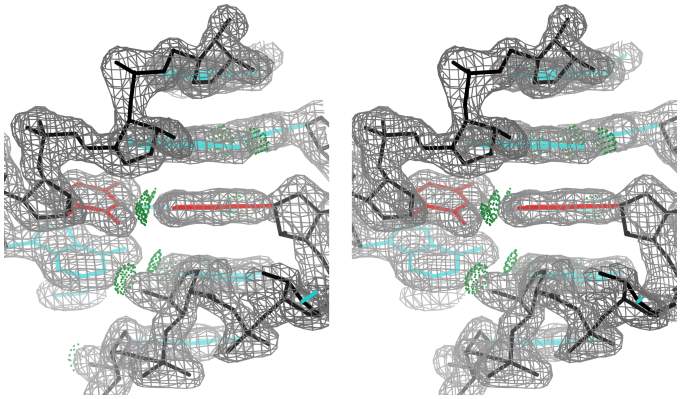


Figure 3: A 25° propeller-twist G-C base-pair (red) with its 1.6Å electron density, from the 437D viral RNA pseudo-knot (Su 1999). The stacks change direction at this base pair, next to where the pseudo-knot switches strand pairing.

Propeller twist is primarily determined by the relative stacking direction of the two paired bases, but it is also restricted by the base-pair H-bonds -- more so if there are more H-bonds to satisfy. G-C pairs, with 3 H-bonds, can twist up to about 25°, as corroborated by the clear electron density seen in figure 3 for the G-C base pair shown in red. Those two bases form their stacks in competing directions where the pseudo-knot switches strand pairing, and their resulting twist is an integral 3D feature of this structure. Base pairs with 2 H-bonds can twist up to about 40° in either direction (see examples in the accompanying article), while those with only one H-bond are not restricted at all by that factor. However *i* to *i+1* base pairs, which can form only one H-bond, happen to be closely coplanar in their most frequent contexts of either a dinucleotide platform or an S-motif.

As mentioned above, these larger departures from parallel stacking or from coplanar base pairs are needed to form the non-repetitive, distinctly recognizable local conformations that enable complex tertiary structure, binding specificity for other molecules, and RNA catalysis. It is important to fit them correctly, because most of them are functionally important. Figure 4 A and B show the location of clusters of such bases

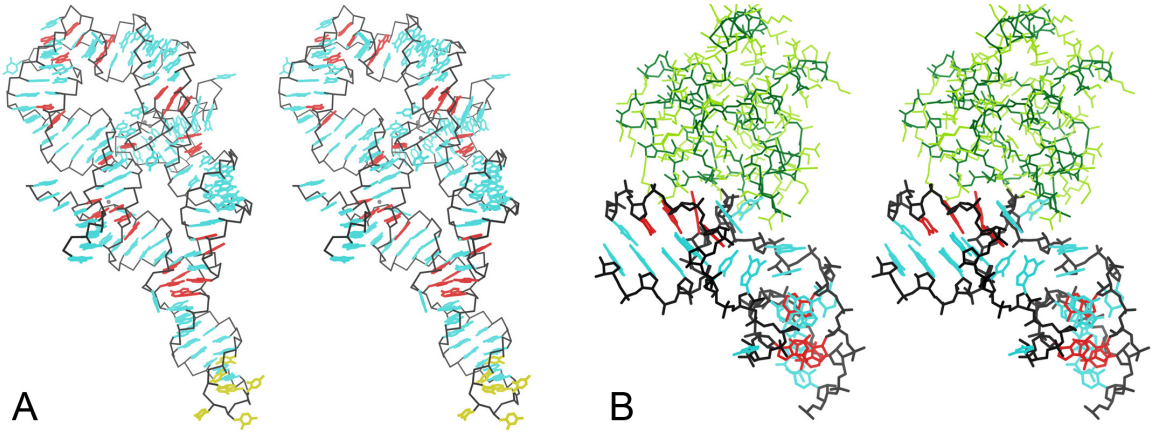


Figure 4: Location of strongly non-parallel bases (red) with high twist or buckle, or with angled, one-spot stacking. They occur at catalytic sites, tertiary contacts, helix junctions, and sites of specific binding, in: (A) the 2R8S self-cleaving P4P6 intron at 1.95Å (Ye 2008) and (B) the 1SDS kink-turn/L7Ae complex at 1.8Å (Hamma 2004). Protein is in green and poorly ordered bases in yellow.

(red) in a ribozyme and in an RNA/protein complex.

Conclusion

At high resolution, the electron density in the well-ordered regions will show unambiguously how to fit the bases, as true for the examples shown here.

At mid resolution, the expectations described here can aid in fitting, and refinement will behave better if base-pair H-bonds are restrained. In Phenix, base-pair H-bonds in RNA or DNA can be identified, classified, and outlier-filtered automatically if they are already in position (Headd, 2012), using a process similar to that for helix and sheet H-bonds in protein; however, it is also a good idea to check manually for marginal but suggestive interactions. For RNA, Phenix can diagnose the correct ribose pucker, but that pucker will be better achieved and kept if you apply the same "P perpendicular" rule (Jain, 2014) in initial fitting.

At the 3 to 4Å resolution range typical for large, important nucleic-acid molecules and complexes, even the adjacent base pairs tend to merge together in the density, and increasingly the model must be built as a unit into the continuous spiral shape of the particular duplex. Maintaining that structural integrity in refinement is improved by restraining stacking as well as H-bonding. Note that in this regime, a crystallographer can benefit greatly from the outside information of secondary and tertiary base-pair prediction if based on alignment of many related sequences, which can achieve more reliable sequence-specificity than unaided low-resolution map fitting.

References

Gill ML, Strobel SA, Loria JP (2006) Crystallization and characterization of the Thallium form of Oxytricha nova G-quadruplex, *Nucleic Acids Res* **34**: 4506-4514 (2HBN)

Jain S, Kapral G, Richardson D, Richardson J (2014) "Fitting Tips #7: Getting the pucker right in RNA structures", *Comput Crystallogr Newsletter* **5**: 4-7

Hamma T, Ferre-D'Amare A (2004) Structure of protein L7Ae bound to a K-turn derived from an archaeal box H/ACA sRNA at 1.8 Å resolution, *Structure* **12**: 893-903 (1SDS)

Headd JJ, Echols N, Afonine PV, Grosse-Kunstleve RW, Chen VB, Moriarty NW, Richardson JS, Richardson DC, Adams PD (2012) Use of knowledge-based restraints in *phenix.refine* to improve macromolecular refinement at low resolution, *Acta Crystallogr* **D68**: 381-390

Malashkevich VN, Padlan FC, Toro R, Girvin M, Almo SC (2013) Crystal structure of the aptamer minE-lysozyme complex, to be published (4M4o)

Richardson JS (2015) A context-sensitive guide to RNA & DNA base-pair & base-stack geometry, *Comp Cryst Newsletter* **6**: 47-53

Su N, Chen L, Egli M, Berger JM, Rich A (1999) Minor groove RNA triplex in the crystal structure of a ribosomal frameshifting viral pseudoknot, *Nat Struct Biol* **6**: 285-292 (437D)

Ye JD, Tereshko V, Frederiksen JK, Koide A, Fellouse FA, Sidhu SS, Koide S, Kossiakoff AA, Piccirilli JA (2008) Synthetic antibodies for specific recognition and crystallization of structured RNA, *Proc Natl Acad Sci USA* **105**:82-87 (2R8S)

New Phaser-MR search panel in Phenix

Robert D. Oeffner, Airlie J. McCoy and Randy J. Read
Department of Haematology, University of Cambridge, Cambridge CB2 0XY, UK

The Phaser-MR graphical user interface (GUI) has undergone relatively few changes since the inception of the modern version of Phenix in 2010. Although fully functional this GUI has sometimes baffled users in the way user input is entered. In particular, users have found the "Search procedure" panel confusing, commonly leading to mistakes when entering input. With the release of Phenix 1.10 these shortcomings have been addressed. Below we discuss the new and the old GUI with a focus on these issues.

The old Phenix Phaser-MR GUI

In figure 1, the old Phaser-MR GUI is shown with the search panel active for the beta-blip tutorial included in Phenix.

This GUI embodies the notion of components defined as ensembles, each of which consists of one or more superimposed atomic coordinate sets or, alternatively, an electron density map. Components are the entities that Phaser MR attempts to place as rigid bodies during an MR search. Anything expected to behave as a rigid object can be used as a search component, which can thus range in size from a structural domain to a whole protein or even an enormous assembly such as the large subunit of the ribosome.

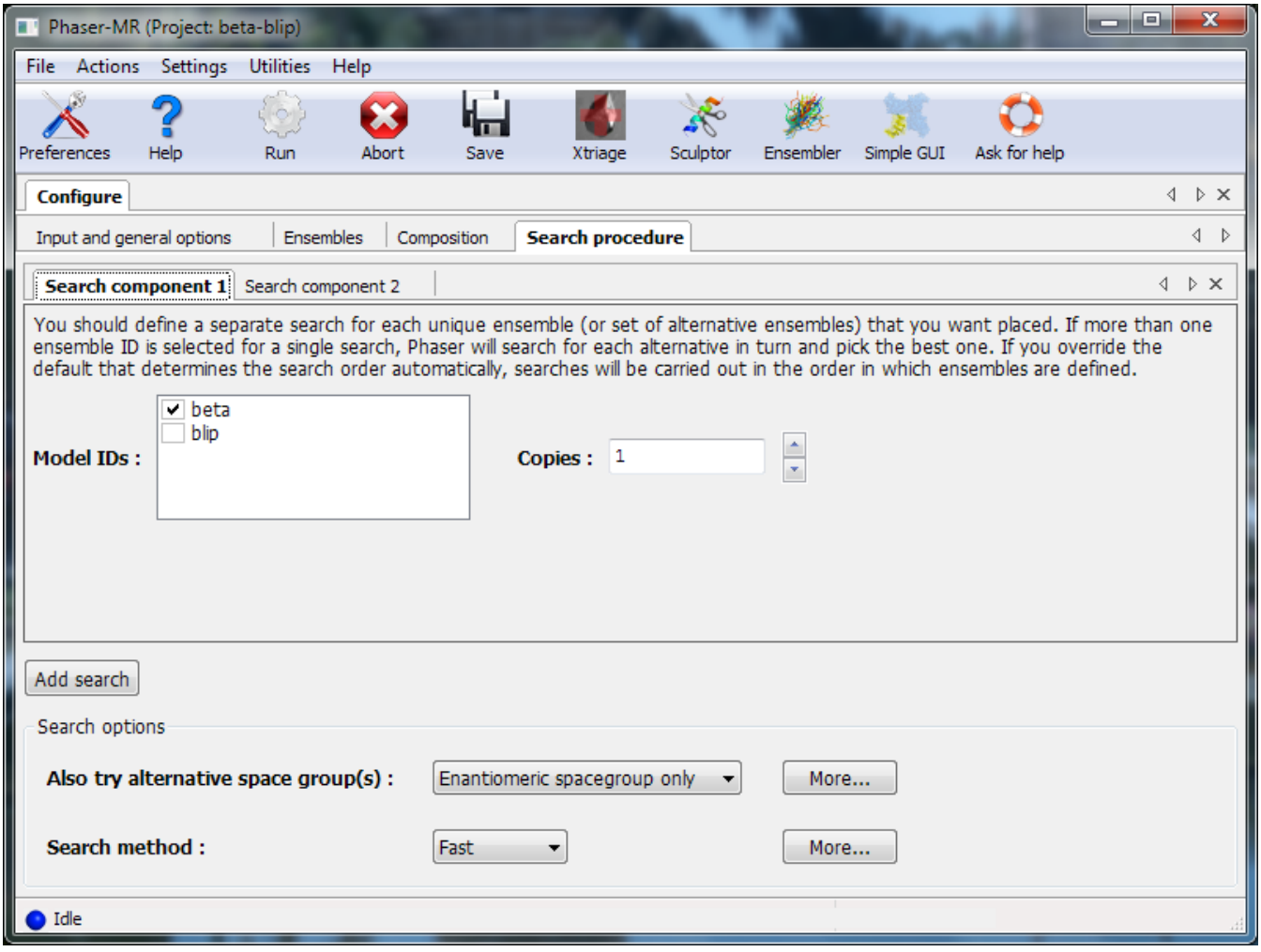


Figure 1: Old style "Search procedure" panel for Phaser MR

Although the representation of the molecular replacement search in the old GUI is technically correct, users were often confused about what to do when setting up a search to find multiple components. The problem was that there are two very different things you can do with multiple ensembles – you can search for multiple different components or you can use them as alternative models for the same component – and the distinction was not intuitively obvious in the interface. Initially there is only one "Search component" page. It features a checklist-box with the available ensembles that can be included in the definition of the particular component. To specify another "Search component" page one has to press the "Add search" button. This button is not particularly prominent. Moreover, because only one "Search component" page is visible at a time the old GUI conveys less clearly what has been specified on other "Search component" pages.

In the case of the beta-blip example it is not uncommon for users to erroneously tick both the beta and the blip ensembles in the checklist-box on the "Search component" page. Although they are expecting Phaser-MR to search for each of the ensembles, they are actually telling it to use these ensembles as alternative models for the same component. As a result, Phaser-MR will do MR searches with each of the ensembles and eventually select the ensemble giving the clearest peak to represent a single component of the crystal. The unintended consequence is that Phaser-MR would consider the MR job done and not attempt to place a second component.

The correct procedure would be for the user to add another "Search component" page and tick complementary ensembles on the checklist-box on each of them so that the total of all components adds up to what the user intends to place if the MR search succeeds. In the case of the beta-blip tutorial this means ticking the blip ensemble on one "Search component" page and ticking the beta ensemble on the other "Search component" page.

The new Phenix Phaser-MR GUI

To overcome these deficiencies a new GUI has been developed, depicted in figure 2. It presents the Search procedure as a tree-list with the root branch of the tree named "Searches". Next to the "Searches" branch is an "Add Search" button that enables the user to add components to search for. These become sub-branches of the "Searches" branch. Next to each "Component" branch is a button allowing the user to add one or more ensembles to the component in question (as alternative choices of model, if more than one is specified), or remove the component and any ensembles altogether. Added ensembles feature as sub-branches on the component branches with names as defined on the "Ensembles" panel. Each of these can be expanded with the mouse to show the names of the constituent coordinate or map files of the ensembles.

To aid the eye icons have been used for all of the branches. Next to the "Searches" branch is an icon of a small protein molecule with a few domains in different colors. Each of the component branches features icons highlighting one of those domains with the remaining domains dimmed. The ensemble branches feature an icon of superposed residues as to illustrate the ensembles of sets of atom coordinates that generally comprises the search model. Expanding an ensemble branch reveals an icon of a single strand and a helix for each of the coordinate files in the ensemble. If a map file is used for an ensemble the icon depicts part of the electron density of a phenylalanine residue. All of these icons only serve as visual cues and any resemblance with targets or models used for a Phaser MR calculation is entirely coincidental.

Figure 2 demonstrates how one would specify the components in the beta-blip tutorial.

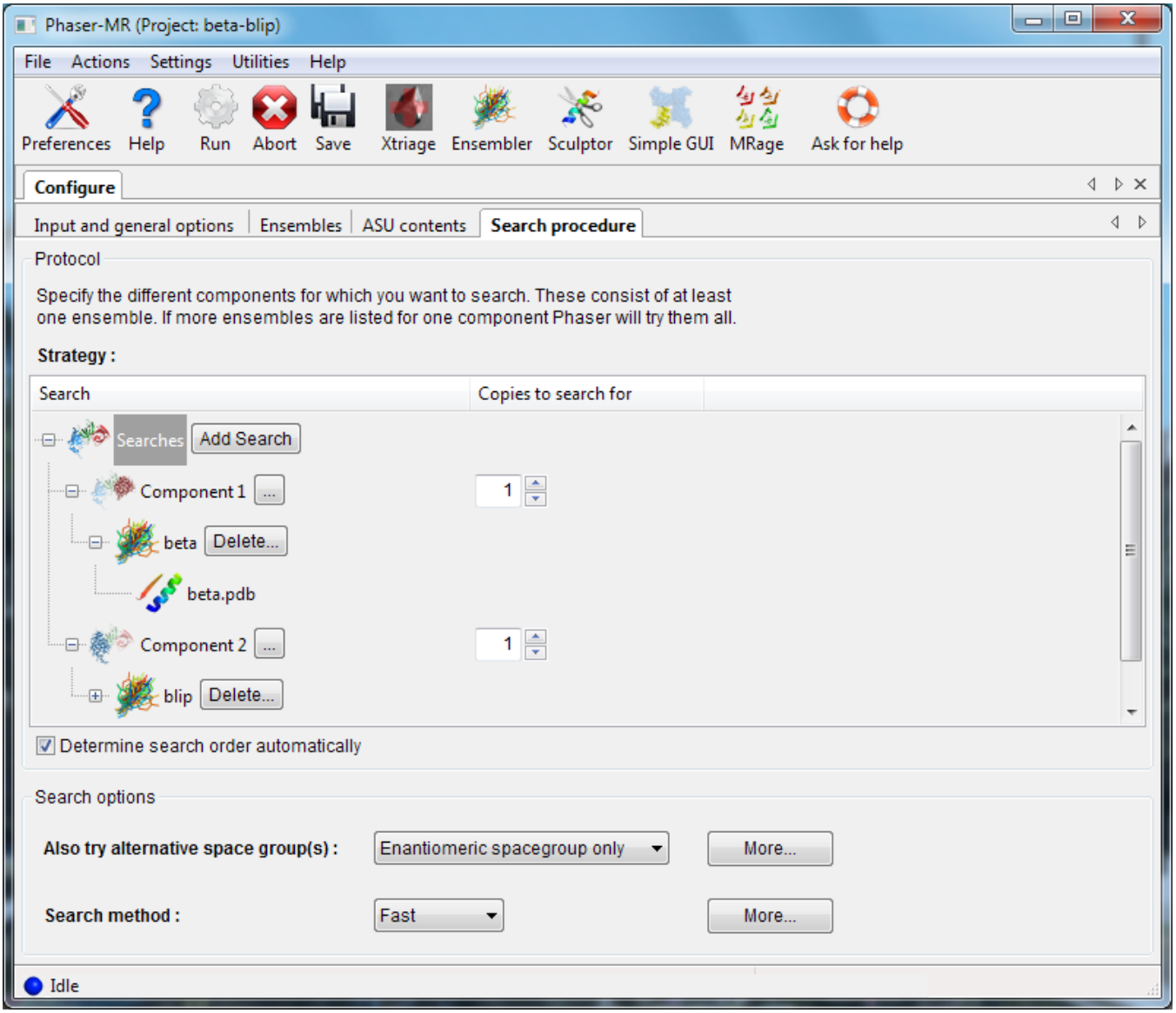


Figure 2: New "Search procedure" panel for Phaser MR

Conclusion

The new GUI addresses the issues that the old GUI suffered from: it presents what has been selected for all components without obscuring selections for one or the other. This makes it clearer to the user which different ensembles are assigned as search models to the various components.

Quantum chemical techniques for minimising ligand geometries in the active site

Nigel W. Moriarty,[a] & Janet E. Deane[b]
[a]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
[b]*Department of Haematology, Cambridge Institute for Medical Research, University of Cambridge, Cambridge CB2 0XY, UK*

Correspondence email: NWMoriarty@LBL.Gov

The lysosomal enzyme β-galactocerebrosidase is essential for the normal catabolism and recycling of galactosphingolipids. GALC catalyzes the removal of the terminal galactose moiety from substrates including galactosylceramide, the principal lipid component of myelin, and psychosine, a cytotoxic metabolite. Krabbe disease (also known as globoid cell leukodystrophy) is a rare, inherited autosomal recessive disorder caused by loss of GALC function, leading to devastating and ultimately fatal neurodegeneration.

Loss of GALC function can be due to mutations that result in misfolding of the enzyme such that it is degraded by cellular quality control mechanisms. Treatment options for Krabbe Disease are very limited but recent work into pharmacological chaperone therapy (PCT) has shown promise in related diseases. PCT involves the use of small molecules that bind and stabilise partially misfolded proteins to aid in their passage from their site of folding in the endoplasmic reticulum to their site of action elsewhere in the cell.

We recently identified a series of azasugar molecules that specifically bind and stabilize the GALC enzyme and may be excellent PCT candidate molecules. Six new structures were determined of these molecules bound in the GALC active site (Hill et al. 2015). The conformation details of one of these (`dgj` from PDB entry `4ufm`) was not completely clear from the density. The `dgj` molecule is shown in Figure 1 and depicts the charged nitrogen that is believed to hydrogen bond to a protein side-chain. The minimum energy conformation of the isolated ligands is the chair conformation by approximately 23 kJ/mol. However, in the active site, in order to maximise the important interactions the ligand would need to adopt the $^1S_3$ twisted-boat conformation.

The resolution of the x-ray information was 2.4Å and the data was not conclusive concerning the conformation of the ligand. The first technique employed to elucidate the conformation was the feature-enhanced map (FEM) (Afonine et al. 2015). Using both the chair and twisted boat conformations, a FEM was calculated for each. To compare the maps on an equal scale the contour level of each map was determined such that each contour level was set to enclose the same amount of electron density. This was preformed using the Phenix command-line tool, `phenix.map_comparison` (Urzhumtsev et al. 2014). The density was unchanged from the standard $2F_o$-F_c map for the chair conformation but there was a bulge in the density blob near the nitrogen that would indicate the twisted-boat was in fact the conformation in the crystal.

Wanting further proof, a second technique was applied to the problem. It appeared that the ligand was adopting a higher energy conformation in the active state compared to the isolated minima because of an energetic advantage from the surrounding amino acids.

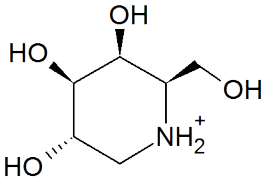


Figure 1: Representation of 1-deoxy-*galacto*-nojirimycin — `dgj` from PDB entry `4ufm`.

Quantum chemistry can provide energy differences sufficiently accurate to determine conformational energy minima of this type. Hydrogen bonding can be approximated with moderately high levels of electron correlation and basis set. In order to keep the size of the calculation tractable a reduced model of the active set was needed.

The steps involved were as follows:

1. Remove all amino acid residues that did not have an atom with 5Å of the selected ligand.
2. Remove backbone atoms expect for C_α. An exception should be made for backbone atoms and termini that interact with the ligand.
3. Add hydrogens to the resulting molecules to produce a neutral moiety except for amino acid chain termini and ligands that are charged or radial.
4. Optimise just the hydrogen positions using the quantum mechanical method of choice. It is recommended that a solvent model be included in the calculation.
5. Check that the hydrogens are still located in chemically sensible positions. Migration of hydrogens can highlight structural problems.
6. Optmise the positions of all ligand atoms and hydrogens of the protein. This allows for optimisation of the hydrogen bonding atoms.

These steps were repeated for both the chair and the twisted-boat models. The chair conformation maintained its starting conformation because of the energy barrier to convert to chair. This is approximately 45 kJ/mol in isolation. The energy difference between the two constrained minimisation gives an indication of the strain energy of the ligand in the active site.

In the case of `dgj`, all quantum mechanical methods indicated that the twisted-boat was the lower energy conformation in the presence of the amino acid side chain atoms in the active site. The best method, B3LYP/6-31G(d,p) with the PCM solvent, favoured the twisted-boat by 59 kJ/mol.

All calculations were performed using GAMESS (Schmidt et al. 2010). A python script was developed to speed the creation of suitable models of the active site and input files for the GAMESS.

References

Afonine, Pavel V, Nigel W. Moriarty, Marat Mustyakimov, Oleg V. Sobolev, Thomas C. Terwilliger, Dusan Turk, Alexandre Urzhumtsev, and Paul D. Adams. 2015. "FEM: Feature-Enhanced Map." *Acta Crystallographica Section D-Biological Crystallography* 71. doi:10.1107/S1399004714028132.

Hill, Chris H., Agnete H. Viuff, Samantha J. Spratley, Stéphane Salamone, Stig H. Christensen, Randy J. Read, Nigel W. Moriarty, Henrik H. Jensen, and Janet E. Deane. 2015. "Azasugar Inhibitors as Pharmacological Chaperones for Krabbe Disease." *Chemical Science*, March. doi:10.1039/C5SC00754B.

Schmidt, M.W., K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, et al. 2010. *GAMESS* (version 1 OCT 2010 (R1)). 64 bit Intel. Iowa State University.

Urzhumtsev, Alexandre, Pavel V. Afonine, Vladimir Y. Lunin, Thomas C. Terwilliger, and Paul D. Adams. 2014. "Metrics for Comparison of Crystallographic Maps." *Acta Crystallographica Section D-Biological Crystallography* 70: 2593–2606.

13 typical occupancy refinement scenarios and available options in *phenix.refine*

Pavel V. Afonine
Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Introduction

By definition, occupancy is atomic model parameter (figure 1) that describes the fraction of corresponding atoms in the crystal that occupy the position defined in the crystal structure model. If all molecules in the crystal are identical occupancies for all atoms are unit. Refining occupancy of an atom against experimental data provides an estimate of the frequency to find this atom at given position. Similarly to Atomic Displacement Parameter (ADP or B-factor), occupancy describes the disorder. Unlike ADP that describes small disorder (within harmonic approximation), occupancy describes large-scale discrete disorder.

In practice there are several scenarios when occupancy may differ from unity and therefore its refinement may be desirable.

1. Residue side chain adopts several conformations (figure 2)

This situation is recognized automatically by `phenix.refine` as long as PDB file contains both conformers labeled with unique alternative location identifiers (`altloc id`; figure 2b). Note,

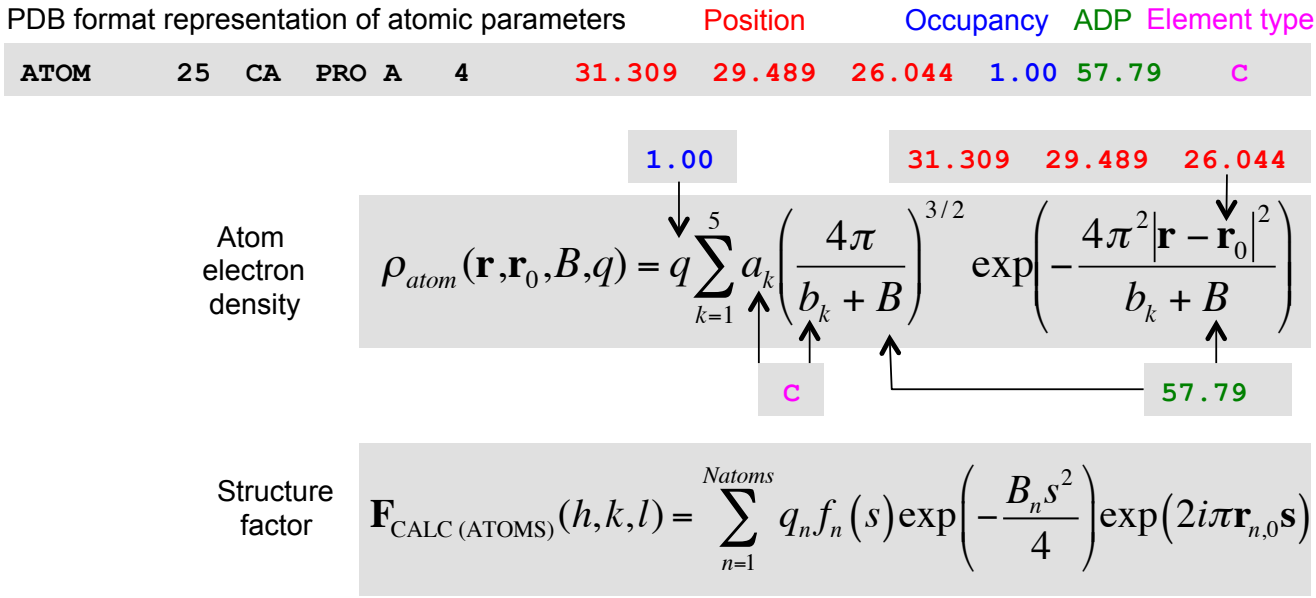


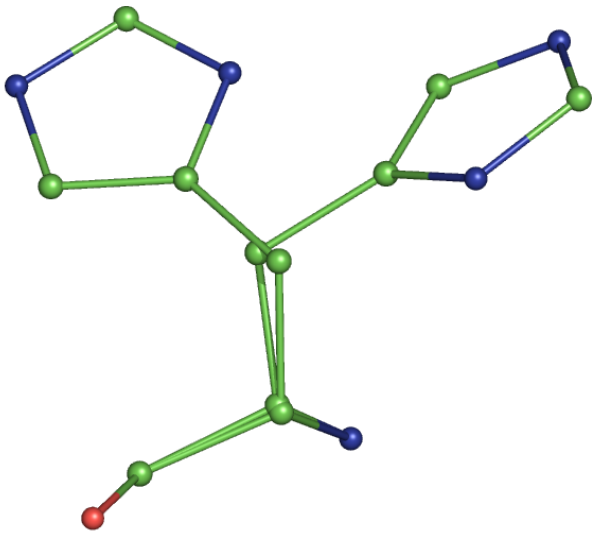
Figure 1: Illustration of atomic model parameters as represented in PDB and their relationship with atomic electron density and structure factors. Position is a triplet of Cartesian coordinates of the atom, $\mathbf{r}_0$. ADP is atomic displacement parameter, B (Å), that describes harmonic atomic vibrations around its mean position $\mathbf{r}_0$. Chemical element type defines form-factor table values to be used. Occupancy q defines probability to find the atom at its mean position $\mathbf{r}_0$.

it is essential that chain and residue `id` be identical for both rotamers. A TER record between the conformers (explicitly atom 1494 and 1495) will stop *`phenix.refine`* from grouping the occupancies of the two rotamers such that their sum adds up to 1. Initial occupancy values set in input PDB file that is subject to occupancy refinement are technically not important as *`phenix.refine`* will make sure that occupancies of all atoms within each conformer will be equal to each other and the sum of occupancies of each conformer will be 1, as shown in figure 2a. However, setting reasonable initial values may improve refinement convergence and thus provide a better refinement result.

2. Continues range of residues adopts several conformations (figure 3a)

This is very similar to the previous case (1) with the following nuance. By default *`phenix.refine`* will account for the fact that adjacent residues have alternative conformations and thus will group occupancies such that identical `altlocs` of consecutive residues will form one group. In this example, there will be two refinable

(a)



(b)

```
ATOM   1485  N   HIS A 501      -2.738  -8.187 -81.483  1.00 46.52           N
ATOM   1486  C   HIS A 501      -4.954  -8.042 -80.397  1.00 44.56           C
ATOM   1487  O   HIS A 501      -5.556  -8.218 -79.338  1.00 46.62           O
ATOM   1488  CA AHIS A 501      -3.511  -7.547 -80.420  0.48 47.00           C
ATOM   1489  CB AHIS A 501      -3.515  -6.028 -80.607  0.48 48.03           C
ATOM   1490  CG AHIS A 501      -2.151  -5.411 -80.606  0.48 46.50           C
ATOM   1491  ND1AHIS A 501      -1.359  -5.360 -79.480  0.48 47.66           N
ATOM   1492  CD2AHIS A 501      -1.447  -4.802 -81.590  0.48 49.62           C
ATOM   1493  CE1AHIS A 501      -0.221  -4.756 -79.772  0.48 50.19           C
ATOM   1494  NE2AHIS A 501      -0.249  -4.407 -81.046  0.48 51.14           N
ATOM   1495  CA BHIS A 501      -3.486  -7.611 -80.375  0.52 46.62           C
ATOM   1496  CB BHIS A 501      -3.337  -6.084 -80.311  0.52 47.45           C
ATOM   1497  CG BHIS A 501      -4.075  -5.347 -81.386  0.52 49.27           C
ATOM   1498  ND1BHIS A 501      -3.442  -4.521 -82.289  0.52 47.63           N
ATOM   1499  CD2BHIS A 501      -5.393  -5.300 -81.692  0.52 52.32           C
ATOM   1500  CE1BHIS A 501      -4.338  -4.001 -83.109  0.52 46.71           C
ATOM   1501  NE2BHIS A 501      -5.529  -4.459 -82.769  0.52 36.75           N
```

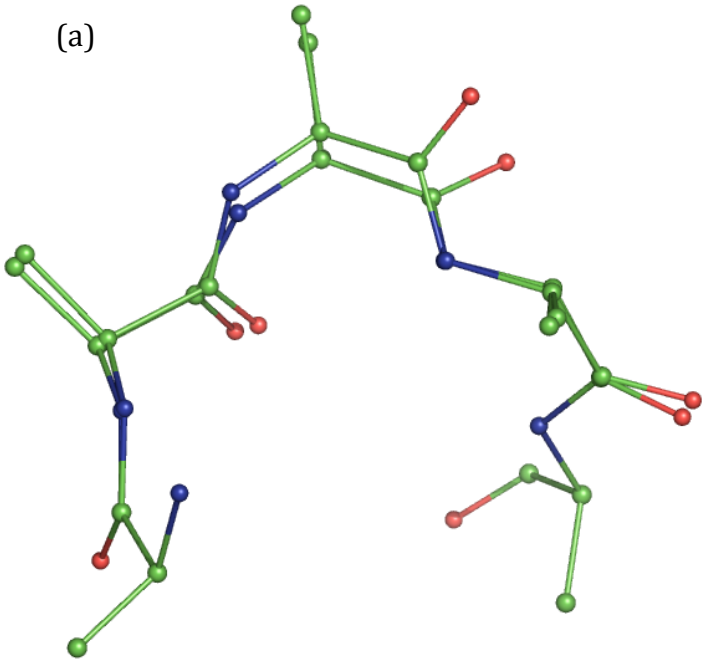
Figure 2: (a) Simple alternative location example above and (b) PDB format input for simple alternative location example below.

occupancies: one for `altloc` A of residues 2-4 and one for `altloc` B of residues 2-4 (figure 3b). As in example 1 both occupancies will add up to 1 and all atoms with the same `altloc id` will have identical occupancy value.

3. Ligand adopts several conformations.

Similarly to case 1 above, a ligand may be modeled with several instances, in this case three copies (figures 4a,b). In case only one copy of partially occupied ligand can be modeled there are two possibilities (figures 4c,d). The differences between the two are following. In

(a)



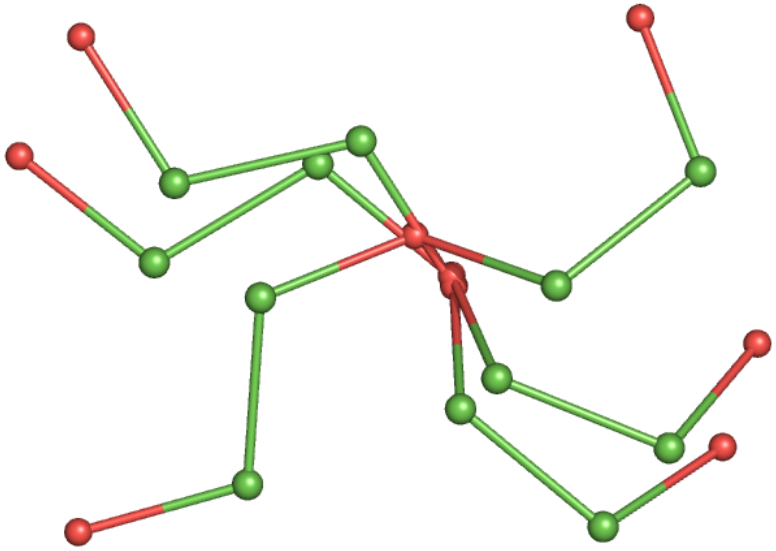
(b)

```
ATOM      1  N    ALA E   1      11.457  32.103  12.052  1.00 40.00           N
ATOM      2  CA   ALA E   1      10.493  31.763  13.123  1.00 40.00           C
ATOM      3  C    ALA E   1       9.291  31.107  12.416  1.00 40.00           C
ATOM      4  O    ALA E   1       8.325  30.669  13.094  1.00 40.00           O
ATOM      5  CB   ALA E   1      10.068  32.938  13.894  1.00 40.00           C
ATOM      6  N  AALA E   2       9.408  30.972  11.116  0.70 40.00           N
ATOM      7  CA AALA E   2       8.377  30.266  10.365  0.70 40.00           C
ATOM      8  C  AALA E   2       8.828  28.858   9.947  0.70 40.00           C
ATOM      9  O  AALA E   2       8.439  27.876  10.572  0.70 40.00           O
ATOM     10  CB AALA E   2       7.896  31.075   9.196  0.70 40.00           C
ATOM      6  N  BALA E   2       9.469  30.978  11.090  0.30 40.00           N
ATOM      7  CA BALA E   2       8.487  30.183  10.293  0.30 40.00           C
ATOM      8  C  BALA E   2       9.057  28.820   9.846  0.30 40.00           C
ATOM      9  O  BALA E   2       8.805  27.806  10.515  0.30 40.00           O
ATOM     10  CB BALA E   2       8.007  30.974   9.055  0.30 40.00           C
ATOM     11  N  AALA E   3       9.688  28.736   8.922  0.70 40.00           N
ATOM     12  CA AALA E   3       9.890  27.338   8.454  0.70 40.00           C
ATOM     13  C  AALA E   3      11.094  26.643   9.096  0.70 40.00           C
ATOM     14  O  AALA E   3      11.317  25.461   8.839  0.70 40.00           O
ATOM     15  CB AALA E   3       9.966  27.369   6.939  0.70 40.00           C
ATOM     11  N  BALA E   3       9.654  28.778   8.644  0.30 40.00           N
ATOM     12  CA BALA E   3      10.138  27.531   8.077  0.30 40.00           C
ATOM     13  C  BALA E   3      11.523  27.206   8.558  0.30 40.00           C
ATOM     14  O  BALA E   3      12.344  26.906   7.766  0.30 40.00           O
ATOM     15  CB BALA E   3      10.156  27.607   6.544  0.30 40.00           C
ATOM     16  N  AALA E   4      11.953  27.354   9.822  0.70 40.00           N
ATOM     17  CA AALA E   4      13.153  26.709  10.368  0.70 40.00           C
ATOM     18  C  AALA E   4      12.894  25.784  11.604  0.70 40.00           C
ATOM     19  O  AALA E   4      13.791  25.098  12.048  0.70 40.00           O
ATOM     20  CB AALA E   4      14.168  27.800  10.647  0.70 40.00           C
ATOM     16  N  BALA E   4      11.803  27.234   9.837  0.30 40.00           N
ATOM     17  CA BALA E   4      13.013  26.554  10.334  0.30 40.00           C
ATOM     18  C  BALA E   4      12.849  25.762  11.621  0.30 40.00           C
ATOM     19  O  BALA E   4      13.844  25.374  12.223  0.30 40.00           O
ATOM     20  CB BALA E   4      14.100  27.587  10.574  0.30 40.00           C
ATOM     21  N    ALA E   5      11.670  25.842  12.211  1.00 40.00           N
ATOM     22  CA   ALA E   5      11.279  24.881  13.256  1.00 40.00           C
ATOM     23  C    ALA E   5       9.827  24.379  13.029  1.00 40.00           C
ATOM     24  O    ALA E   5       8.913  24.936  13.515  1.00 40.00           O
ATOM     25  CB   ALA E   5      11.420  25.549  14.553  1.00 40.00           C
```

Figure 3: (a) Multi-residue alternative location example and (b) PDB format input for mutli-resiude alternative location example.

case of figure 4d a single occupancy per whole ligand will be always refined regardless of starting values constraining it to be between 0 and 1. Also, the ligand will not "see" other atoms that have `altloc` different from A. In the other case (figure 4c) two facts trigger the group occupancy refinement (one occupancy per whole ligand): occupancy is less then 1 and occupancies of all atoms are equal to each other. In this case occupancies that have zero as starting value will not be refined. If occupancies of all were not equal to each other, then occupancies $0<q<1$ will be refined individually each constrained to be in [0,1] range.

(a)



(b)

```
HETATM    1  O4 APEG M   1      -0.091   8.619  56.188  1.00 97.26           O
HETATM    2  C4 APEG M   1       0.240   9.310  57.377  1.00 93.43           C
HETATM    3  C3 APEG M   1       0.193  10.813  57.129  1.00 97.43           C
HETATM    4  O2 APEG M   1      -0.687  11.080  56.017  1.00102.78           O
HETATM    5  C2 APEG M   1      -0.510  12.375  55.423  1.00 89.67           C
HETATM    6  C1 APEG M   1      -1.088  13.457  56.368  1.00 88.16           C
HETATM    7  O1 APEG M   1      -1.445  14.612  55.612  1.00 78.51           O
HETATM    8  O4 BPEG M   1       0.133  10.917  53.508  1.00 97.26           O
HETATM    9  C4 BPEG M   1       0.605  10.017  54.492  1.00 93.43           C
HETATM   10  C3 BPEG M   1       0.387  10.610  55.879  1.00 97.43           C
HETATM   11  O2 BPEG M   1      -0.693  11.565  55.819  1.00102.78           O
HETATM   12  C2 BPEG M   1      -0.733  12.479  56.926  1.00 89.67           C
HETATM   13  C1 BPEG M   1      -1.229  11.734  58.189  1.00 88.16           C
HETATM   14  O1 BPEG M   1      -1.810  12.666  59.098  1.00 78.51           O
HETATM   15  O4 CPEG M   1       0.133   8.810  55.355  1.00 97.26           O
HETATM   16  C4 CPEG M   1       0.605   9.212  56.627  1.00 93.43           C
HETATM   17  C3 CPEG M   1       0.387  10.709  56.807  1.00 97.43           C
HETATM   18  O2 CPEG M   1      -0.693  11.135  55.949  1.00102.78           O
HETATM   19  C2 CPEG M   1      -0.733  12.550  55.711  1.00 89.67           C
HETATM   20  C1 CPEG M   1      -1.229  13.272  56.989  1.00 88.16           C
HETATM   21  O1 CPEG M   1      -1.810  14.525  56.636  1.00 78.51           O
```

(c)

```
HETATM    1  O4  PEG M   1      -0.091   8.619  56.188  0.70 97.26           O
HETATM    2  C4  PEG M   1       0.240   9.310  57.377  0.70 93.43           C
HETATM    3  C3  PEG M   1       0.193  10.813  57.129  0.70 97.43           C
HETATM    4  O2  PEG M   1      -0.687  11.080  56.017  0.70 92.78           O
HETATM    5  C2  PEG M   1      -0.510  12.375  55.423  0.70 89.67           C
HETATM    6  C1  PEG M   1      -1.088  13.457  56.368  0.70 88.16           C
HETATM    7  O1  PEG M   1      -1.445  14.612  55.612  0.70 78.51           O
```

(d)

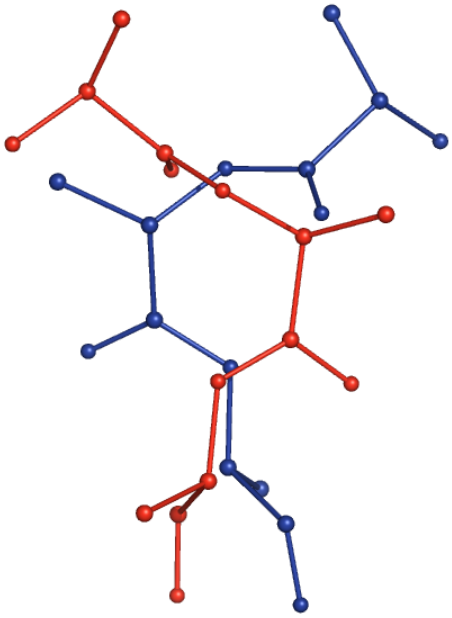
```
HETATM    1  O4 APEG M   1      -0.091   8.619  56.188  0.70 97.26           O
HETATM    2  C4 APEG M   1       0.240   9.310  57.377  0.70 93.43           C
HETATM    3  C3 APEG M   1       0.193  10.813  57.129  0.70 97.43           C
HETATM    4  O2 APEG M   1      -0.687  11.080  56.017  0.70 62.78           O
HETATM    5  C2 APEG M   1      -0.510  12.375  55.423  0.70 89.67           C
HETATM    6  C1 APEG M   1      -1.088  13.457  56.368  0.70 88.16           C
HETATM    7  O1 APEG M   1      -1.445  14.612  55.612  0.70 78.51           O
```

Figure 4: Example of ligands adopting one partial or several alternative conformation (a), and their representation in PDB file (b,c,d).

4. Overlapping chains

It may happen that stretches of two or more chains overlap in space such that when one is present the other one isn't (figure 5a). This situation is not recognized automatically by *phenix.refine* but can be dealt with successfully. This requires two actions. Firstly, overlapping atoms of both chains need to be assigned different `altloc` (figure 5b); this will ensure that these atoms will not be pushed apart by non-bonded repulsion. Secondly, one needs to compose a parameter file telling *phenix.refine* what occupancies need to be coupled (figure 5c); this will override any default behavior applicable to these groups of atoms.

(a)



(b)

```
ATOM      6  N   AALA A  12       8.148  29.454  12.242  0.70 40.00           N
ATOM      7  CA  AALA A  12       7.117  28.748  11.491  0.70 40.00           C
ATOM      8  C   AALA A  12       7.568  27.340  11.073  0.70 40.00           C
ATOM      9  O   AALA A  12       7.179  26.358  11.698  0.70 40.00           O
ATOM     10  CB  AALA A  12       6.636  29.557  10.322  0.70 40.00           C
ATOM     11  N   AALA A  13       8.428  27.218  10.048  0.70 40.00           N
ATOM     12  CA  AALA A  13       8.630  25.820   9.580  0.70 40.00           C
ATOM     13  C   AALA A  13       9.834  25.125  10.222  0.70 40.00           C
ATOM     14  O   AALA A  13      10.057  23.943   9.965  0.70 40.00           O
ATOM     15  CB  AALA A  13       8.706  25.851   8.065  0.70 40.00           C
ATOM     16  N   AALA A  14      10.693  25.836  10.948  0.70 40.00           N
ATOM     17  CA  AALA A  14      11.893  25.191  11.494  0.70 40.00           C
ATOM     18  C   AALA A  14      11.634  24.266  12.730  0.70 40.00           C
ATOM     19  O   AALA A  14      12.531  23.580  13.174  0.70 40.00           O
ATOM     20  CB  AALA A  14      12.908  26.282  11.773  0.70 40.00           C
TER
ATOM      6  N   BALA B   2       6.766  24.529   8.198  0.30 40.00           N
ATOM      7  CA  BALA B   2       6.833  26.021   8.208  0.30 40.00           C
ATOM      8  C   BALA B   2       8.019  26.552   9.041  0.30 40.00           C
ATOM      9  O   BALA B   2       9.077  26.857   8.469  0.30 40.00           O
ATOM     10  CB  BALA B   2       5.515  26.632   8.736  0.30 40.00           C
ATOM     11  N   BALA B   3       7.777  26.826  10.333  0.30 40.00           N
ATOM     12  CA  BALA B   3       8.789  27.424  11.187  0.30 40.00           C
ATOM     13  C   BALA B   3       9.696  26.384  11.781  0.30 40.00           C
ATOM     14  O   BALA B   3       9.907  26.409  12.941  0.30 40.00           O
ATOM     15  CB  BALA B   3       8.132  28.212  12.329  0.30 40.00           C
ATOM     16  N   BALA B   4      10.257  25.482  11.016  0.30 40.00           N
ATOM     17  CA  BALA B   4      11.422  24.727  11.513  0.30 40.00           C
ATOM     18  C   BALA B   4      12.556  24.525  10.521  0.30 40.00           C
ATOM     19  O   BALA B   4      13.441  23.714  10.771  0.30 40.00           O
ATOM     20  CB  BALA B   4      10.966  23.351  11.964  0.30 40.00           C
```

(c)

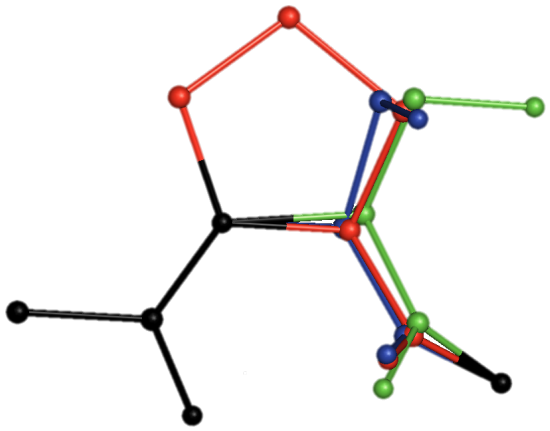
```
refinement {
  refine {
    occupancies {
      constrained_group {
        selection = chain A and resseq 12:14 and altloc A
        selection = chain B and resseq 2:4 and altloc B
      }
    }
  }
}
```

Figure 5: Different chains that overlap in space (a), PDB snippet as an example and Phenix Phil parameters showing how to handle occupancies of overlapping chains in refinement (c).

5. Site shared by several chemical moieties

PDB entry `1EJG` serves a good example of such case. Figures 6a focuses on residue number 22, which is `PRO` and two `CYS` in alternative conformations, being all together three residues sharing the same spot. Figure 6b exemplifies kinds of PDB syntax that will be automatically recognized by *`phenix.refine`*. Note, residue and chain ids are identical, while `altloc` are different, as well as residue names are allowed to be different.

(a)



(b)

```
ATOM    387  CA  THR A  21       3.664  10.576  -3.461  1.00  1.42           C
ATOM    388  C   THR A  21       4.915  11.347  -3.001  1.00  1.58           C
ATOM    389  O   THR A  21       5.844  10.729  -2.478  1.00  2.24           O
ATOM    400  N   PRO A  22       4.915  12.683  -3.102  1.00  1.83           N
ATOM    401  CA APRO A  22       6.042  13.429  -2.601  0.57  1.81           C
ATOM    402  C  APRO A  22       6.387  13.122  -1.160  0.57  1.66           C
ATOM    403  O  APRO A  22       5.480  13.006  -0.345  0.57  2.08           O
ATOM    404  CB APRO A  22       5.655  14.896  -2.744  0.57  2.86           C
ATOM    405  CG APRO A  22       4.661  14.854  -4.058  0.57  2.66           C
ATOM    406  CD APRO A  22       3.957  13.505  -3.910  0.57  2.27           C
ATOM    414  CA BSER A  22       6.034  13.399  -2.687  0.21  1.55           C
ATOM    415  C  BSER A  22       6.367  13.062  -1.223  0.21  2.92           C
ATOM    416  O  BSER A  22       5.412  13.050  -0.345  0.21  1.87           O
ATOM    417  CB BSER A  22       5.409  14.835  -2.876  0.21  2.60           C
ATOM    418  OG BSER A  22       4.760  15.243  -1.635  0.21  2.11           O
ATOM    420  CA CSER A  22       6.112  13.653  -2.656  0.22  2.31           C
ATOM    421  C  CSER A  22       6.354  13.275  -1.187  0.22  1.92           C
ATOM    422  O  CSER A  22       5.636  12.705  -0.270  0.22  1.77           O
ATOM    423  CB CSER A  22       5.605  15.097  -2.687  0.22  3.56           C
ATOM    424  OG CSER A  22       6.750  15.771  -2.280  0.22  6.38           O
ATOM    426  N   GLU A  23       7.651  13.190  -0.860  1.00  1.86           N
```

Figure 6: Example of a same space shared by different chemical moieties (a) and its representation in PDB file that is handled automatically in refinement.

6. Exchangeable H/D sites

While purely hydrogenated samples are possible, for best results neutron crystallography requires samples to be deuterated or partially deuterated. This means that some of the hydrogen (H) atoms in the structure may be partially or fully substituted with deuterium (D) atoms. Figure 7 shows syntax that is used to record this situation in a PDB file. As long as PDB file is composed as illustrated in figure 7 *`phenix.refine`* will handle this situation automatically: occupancies of H and D will be coupled such that each one is in [0,1] range and their sum is exactly 1. In addition *`phenix.refine`* will constrain coordinates and B-factors of H and D to be identical.

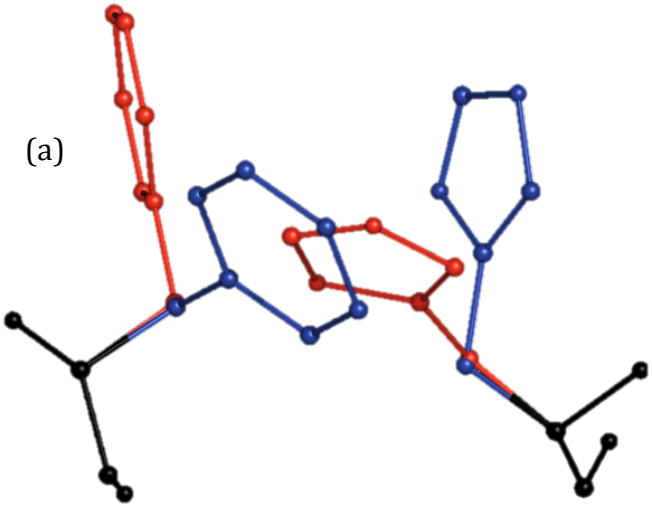
```
ATOM      1  CA  SER A  22       6.034  13.399  -2.687  1.00  1.55           C
ATOM      2  C   SER A  22       6.367  13.062  -1.223  1.00  2.92           C
ATOM      3  O   SER A  22       5.412  13.050  -0.345  1.00  1.87           O
ATOM      4  CB  SER A  22       5.409  14.835  -2.876  1.00  2.60           C
ATOM      5  OG  SER A  22       4.760  15.243  -1.635  1.00  2.11           O
ATOM      6  HB2 SER A  22       6.201  15.544  -3.119  1.00  2.60           H
ATOM      7  HB3 SER A  22       4.670  14.804  -3.676  1.00  2.60           H
ATOM      8  HG ASER A  22       4.050  14.608  -1.403  0.30  2.11           H
ATOM      9  DG BSER A  22       4.050  14.608  -1.403  0.70  2.11           D
```

Figure 7: Example of PDB representation of exchangeable H/D sites.

7. Concerted (coupled) alternative conformations

Figures 8a,b illustrate this scenario. Here two residues, PHE 73 in chain A and HIS 90 in chain B, both have two alternative conformations situated in space such that the same spot is occupied part of the time by conformation B of PHE and part of the time is occupied by conformation A of HIS. Seemingly clashing side chains of these residues are not clashing in reality because they are never present at

(a)



(b)

```
ATOM      5  N   PHE A  73      12.614  27.642  22.065  1.00 30.00           N
ATOM      6  CA  PHE A  73      12.796  27.740  23.498  1.00 30.00           C
ATOM      7  C   PHE A  73      11.723  28.575  24.143  1.00 30.00           C
ATOM      8  O   PHE A  73      11.292  29.592  23.600  1.00 30.00           O
ATOM     10  CB APHE A  73      14.228  28.251  23.830  0.70 30.00           C
ATOM     11  CG APHE A  73      15.354  27.370  23.370  0.70 30.00           C
ATOM     12  CD1APHE A  73      15.774  26.299  24.140  0.70 30.00           C
ATOM     13  CD2APHE A  73      15.993  27.613  22.165  0.70 30.00           C
ATOM     14  CE1APHE A  73      16.808  25.487  23.718  0.70 30.00           C
ATOM     15  CE2APHE A  73      17.029  26.805  21.739  0.70 30.00           C
ATOM     16  CZ APHE A  73      17.437  25.739  22.515  0.70 30.00           C
ATOM     17  CB BPHE A  73      14.173  28.323  23.818  0.30 30.00           C
ATOM     18  CG BPHE A  73      14.460  29.621  23.118  0.30 30.00           C
ATOM     19  CD1BPHE A  73      14.929  29.633  21.815  0.30 30.00           C
ATOM     20  CD2BPHE A  73      14.261  30.829  23.764  0.30 30.00           C
ATOM     21  CE1BPHE A  73      15.191  30.826  21.170  0.30 30.00           C
ATOM     22  CE2BPHE A  73      14.524  32.025  23.123  0.30 30.00           C
ATOM     23  CZ BPHE A  73      14.989  32.024  21.824  0.30 30.00           C
TER
ATOM     24  N   HIS B  90      16.239  33.606  26.584  1.00 30.00           N
ATOM     25  CA  HIS B  90      15.150  32.655  26.764  1.00 30.00           C
ATOM     26  C   HIS B  90      15.091  32.170  28.215  1.00 30.00           C
ATOM     27  O   HIS B  90      16.050  31.585  28.724  1.00 30.00           O
ATOM     28  CB BHIS B  90      15.368  31.394  25.935  0.30 30.00           C
ATOM     29  CG BHIS B  90      16.683  31.284  25.230  0.30 30.00           C
ATOM     30  ND1BHIS B  90      17.332  30.072  25.122  0.30 30.00           N
ATOM     31  CD2BHIS B  90      17.420  32.174  24.522  0.30 30.00           C
ATOM     32  CE1BHIS B  90      18.429  30.232  24.402  0.30 30.00           C
ATOM     33  NE2BHIS B  90      18.509  31.496  24.030  0.30 30.00           N
ATOM     35  CB AHIS B  90      15.468  31.394  25.935  0.70 30.00           C
ATOM     36  CG AHIS B  90      15.431  31.564  24.449  0.70 30.00           C
ATOM     37  ND1AHIS B  90      14.958  30.558  23.632  0.70 30.00           N
ATOM     38  CD2AHIS B  90      15.698  32.617  23.640  0.70 30.00           C
ATOM     39  CE1AHIS B  90      14.968  30.980  22.379  0.70 30.00           C
ATOM     40  NE2AHIS B  90      15.416  32.222  22.355  0.70 30.00           N
```

(c)

```
refinement {
  refine {
    occupancies {
      constrained_group {
        selection = chain A and resseq 73 and altloc A or \
                    chain B and resseq 90 and altloc A or
        selection = chain A and resseq 73 and altloc B or \
                    chain B and resseq 90 and altloc B or
      }
    }
  }
}
```

Figure 8: Example of concerted alternative conformations. (a) Two side chains have two alternative conformations, and two of them clash, (b) PDB snippet that exemplifies the situation, (c) Phenix Phil parameters defining how occupancies of the conformers need to be coupled in refinement.

that space simultaneously. The fact that "clashing" side chains have different `altloc id` instructs the refinement program to not use non-bonded repulsion term for involved atoms. While this may be automated in future, currently it's a requirement manually construct a file to instruct the refinement program how occupancies of these residues should be refined: `altloc` A should be the same for atoms in both residues, likewise for `altloc` B, and their sum adds up to 1. This can be done by composing a parameter file as shown in figure 8c.

A similar situation would be to have a single partially occupied water (or any ligand) instead of one of the residues. In such case the corresponding PDB file would have one ATOM entry and matching non-blank `altloc` for the water. It is essential to have matching non-blank `altloc` for the water even though it has just one entry (one atom) in the file because this will instruct the program to disable repulsions between this water and corresponding conformer of the residue side chain.

8. Occupancy refinement and special positions

If a single atom such as water or metal ion sits exactly on or near to a special position, for example, two-fold axis, there are two possible scenarios. One is occupancy of the atom in input is equal to 1. In this case the symmetry factor ½ is applied to the occupancy internally and the atom always stays with occupancy 1 in the file. Alternatively, if occupancy is set to 0.5 before the refinement (or, more generally, is not equal to 1) then symmetry factor is not applied internally and also the occupancy will be refined individually following *`phenix.refine`* convention to refine occupancies of atoms that are different from 0 and 1.

9. Occupancy refinement and heavy atoms

Atoms heavier than typical protein atoms (C,O,N) are more electron rich and therefore more pronounced in Fourier maps. Likewise, errors in parameters of such atoms are more visible on residual Fourier maps. It is not uncommon to observe residual map features around heavy atoms. One of several possible reasons for this is that the ion may not be fully occupied and therefore modeling it with full occupancy may not be adequate. Refining occupancies of heavy atoms in such case may be desirable.

10. Radiation damage

Radiation damage may result in structure changes such as loss of hydroxyl groups from tyrosine residues or decarboxylation of aspartates and glutamates. This effect may not be identical for all unit cells of the crystal meaning that occupation of corresponding groups may vary from unit cell to unit cell. If residual map suggests this situation then refining occupancies of these groups may be desirable.

11. Partial Selenium incorporation in SE-MET

This is typically observed as negative residual density around Selenium atoms. Enabling individual occupancy refinement of Se atoms to accommodate the fact that they may not fully substitute Sulfur is desirable in this case.

12. Single partially occupied instance of residue or ligand

If residue or ligand has multiple conformations but only one can be identified and modeled, it is desirable to refine one occupancy factor for involved atoms. This can be done in two ways. One is to assign a non-blank `altloc id` to these atoms. Another possibility is to set occupancy value to these atoms that is greater than 0 and less then 1. It is important that this partial occupancy is

(a)
```
refinement {
  refine {
    occupancies {
      constrained_group {
        selection = chain A and resseq 73 and altloc A or \
                    chain B and resseq 90 and altloc A
      }
      constrained_group {
        selection = chain A and resseq 73 and altloc B
      }
      constrained_group {
        selection = chain B and resseq 90 and altloc B
      }
    }
  }
}
```

(b)
```
refinement {
  refine {
    occupancies {
      remove_selection = (chain A and resseq 73) or (chain B and resseq 90)
  }
}
```

(c)
```
refinement {
  refine {
    occupancies {
      individual = element Zn or water
  }
}
```

Figure 9: Examples of overriding default refinement behavior. See text for details.

equal for all involved atoms; different occupancy values will trigger individual occupancy refinement.

13. Overriding default behavior

Any user-provided selections for occupancy refinement will override the default behavior of the refinement package. For example, considering the scenario shown in figure 8: it is possible to request the program to refine a single occupancy for all atoms with `altloc` A in both residues, and one occupancy per each B conformer, resulting in this case in three refinable occupancy factors. Parameter file shown in figure 9a is what's needed for this situation.

In some cases it may be desirable to keep partial occupancy value during refinement (that is not refine it as *`phenix.refine`* would do otherwise by default). This can be done by excluding selected atoms from default behavior. Using case 7 as an example, it is possible to keep occupancies of involved residues at 0.3 and 0.7 by using this syntax in figure 9b.

Finally, it is possible to enable individual occupancy refinement for atoms that otherwise by default are not subject to occupancy refinement. Example in figure 9b enables individual occupancy refinement for Zinc ion and all water molecules; refined occupancy values will range between 0 and 1.

Summary

`phenix.refine` automatically considers occupancy refinement for atoms that have occupancy greater than zero and not equal to one, as well as if atoms have non-blank `altloc` identifier. Refinement scenarios may vary

broadly and most typical cases are described above. User has a full control over occupancy refinement, such as a possibility to undo or override default occupancy refinement behavior for any selected atoms, request individual occupancy refinement for selected atoms or add constrained occupancy groups.

A context-sensitive guide to RNA & DNA base-pair & base-stack geometry

Jane S. Richardson
Department of Biochemistry, Duke University, Durham NC 27710, USA
Correspondence email: Jane.Richardson@duke.edu

Introduction

Increasingly large and biologically important nucleic-acid/protein complexes are now being solved, typically at resolutions of 3-4Å and even lower, where individual bases are seldom clearly visible. There is thus an increasing need for reliable rules of thumb to guide their modeling. This article combines prior textbook-level knowledge (e.g., Saenger 1984; Creighton 2011) with a survey of the low-B-factor parts of recent high-resolution RNA and DNA structures (≤ about 2Å) with deposited structure-factor data, for which the base positions and orientations can be confirmed as unambiguous to very high accuracy. The goal here is to define expectations for planarity and twist values of base stacking and pairing relationships for various RNA and DNA double-helix types, including common local irregularities and accommodations to complex tertiary-structures or functional sites. These expectations can then guide manual modeling at poorer resolution, and especially can form the basis for setting the parameters of automated restraints.

Base-stacking and its variation

Base stacking is a very strong effect, optimizing both van der Waals contact and pi-pi interaction. It occurs with amino acids or small molecules as well as with other bases. If backbone conformation permits, stacked bases are closely parallel at optimal 3.5Å separation, but usually sheared from complete overlap and often better stacked between successive base-pairs than between successive bases on a given strand. As typical, the green and blue dots of favorable all-atom vdW contacts (Word 1999) are extensive for the lower stack in figure 1 (2R8S; Ye 2008). When good stacking is not possible, bases still maintain vdW contact at an edge, as between the upper two bases in fig. 1. Stacking may be opened up at helix junctions, to compromise between interactions on opposite ring faces, or to allow for an inserted base (figure 2), but is essentially always near optimal within regular duplexes.

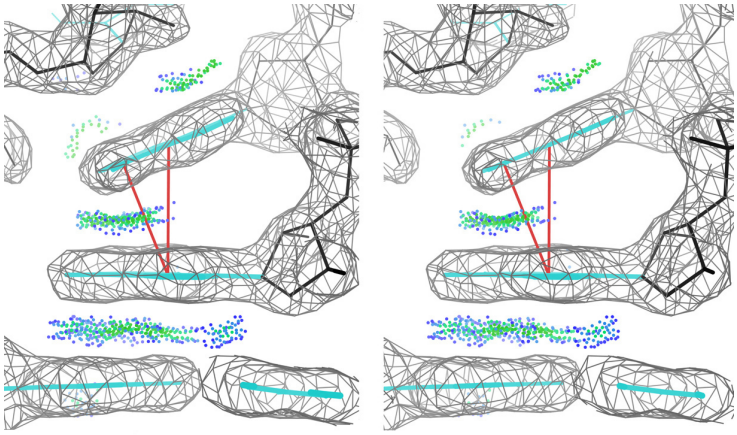


Figure 1: Optimal vs compromised base stacking. All-atom contact dots show a wide area of optimal, parallel stacking at 3.5Å separation for the lower pair of bases, but the upper pair are tilted 22° (normal in red) by their backbone conformations and touch only at one edge. 2R8S Fab/P4P6 ribozyme complex (1.95 Å), around residue 183.

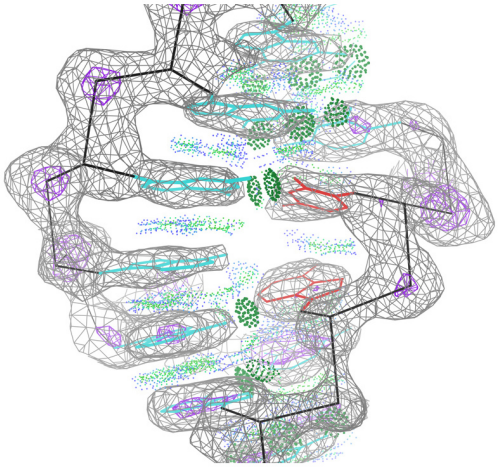


Figure 2: Stack-opening by 44° (red bases), to accommodate an inserted base in the left strand. U205-A206 in 2R8S.

Base-pair propeller-twist

The individual RNA and DNA bases are strongly planar, aromatic systems, but base-pairs are only indirectly pulled toward co-planarity, by similarity of stacking directions and by the base-pair hydrogen bonds. Their relative orientation also responds to limitations from the local backbone conformation and the demands of forming tertiary structure.

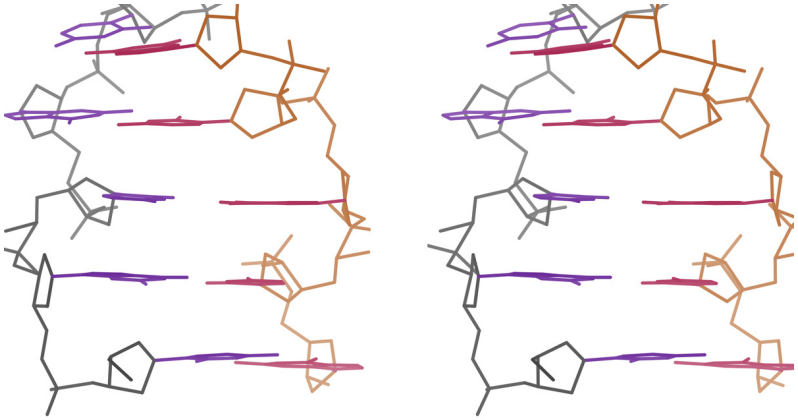


Figure 3: Stereo of co-planar base-pairs with essentially zero propeller-twist in a Z-form DNA 12-mer. 4oCB at 0.75Å.

In this context, the two most revealing of the classic parameters that measure deviation from base-pair co-planarity (Olson 2001) are the "propeller-twist" torsion around a line joining the two bases, and the "buckle" angle of their bend across the line of base-pair H-bonds. Individual nucleotides and the helices they form are both handed, so it should not be a surprise that propeller-twist is typically not 0°. The only place in this dataset that has multiple, truly flat base pairs with near-zero twist and buckle is in the rare, left-handed Z-form DNA duplexes (figure 3). Perhaps the two very different, alternating conformations that make up Z-DNA have opposite twist preferences that cancel each other out.

Each type of double helix shows a typical average for base-pair propeller-twist, from -15° (left-handed) to +11° (right-handed), not well correlated to duplex handedness. Figure 4 shows stereos of: a) regular B-form DNA, here at -15° left-handed twist (1K61; Aishima 2002), while short B-DNAs average about -12°; b) all-GC-pair A-form RNA with -12° left-handed average twist (4MS9; Sheng 2014); c) mixed AT/GC A-form duplex within a complex RNA, with -14° average twist (2R8S); and d) a parallel poly-A duplex with +11° right-handed average twist (4JRD; Safaee 2013). The twist of individual pairs varies about ±5° from average (modulated by GC vs AT, helix bending, and specific local interactions), and thus can occasionally be quite near zero.

Base-pair buckle

Base-pairs can buckle, as well as twist, away from co-planarity. Buckle is a bend between the two base planes, folded along the line of the base-pair H-bonds. There is very little buckle in regular double-helix types for either DNA or RNA, as can be seen in figure 4. However, figure 5 shows the 33° buckle of a specific non-canonical A-G base-pair in the kink-turn binding site of 1SDS (Hamma 2004).

Large deviations from base-pair co-planarity in complex RNA structures

Quite substantial deviations from co-planar base-pairs (either in twist or buckle) are sometimes needed to make stable, unique RNA tertiary structure and to support function for ribozymes and for RNA or DNA aptamers. These occur at junctions between helices, at characteristic bends or kinks, at active sites, and at specific binding sites for other molecules. High propeller-twist can occur in Watson-Crick or non-canonical base-pairs, and can be either right- or left-handed. Figures 6a and 6b show two individual base-pairs with a high propeller-twist of about +40°, one example in end view (3G9Y; Loughlin 2009) and one in side view (2R8S).

The maximal tolerance for base-pair propeller-twist differs as a function of how many hydrogen bonds they share. 40° is high but feasible for 2-H-bond pairs, as seen in

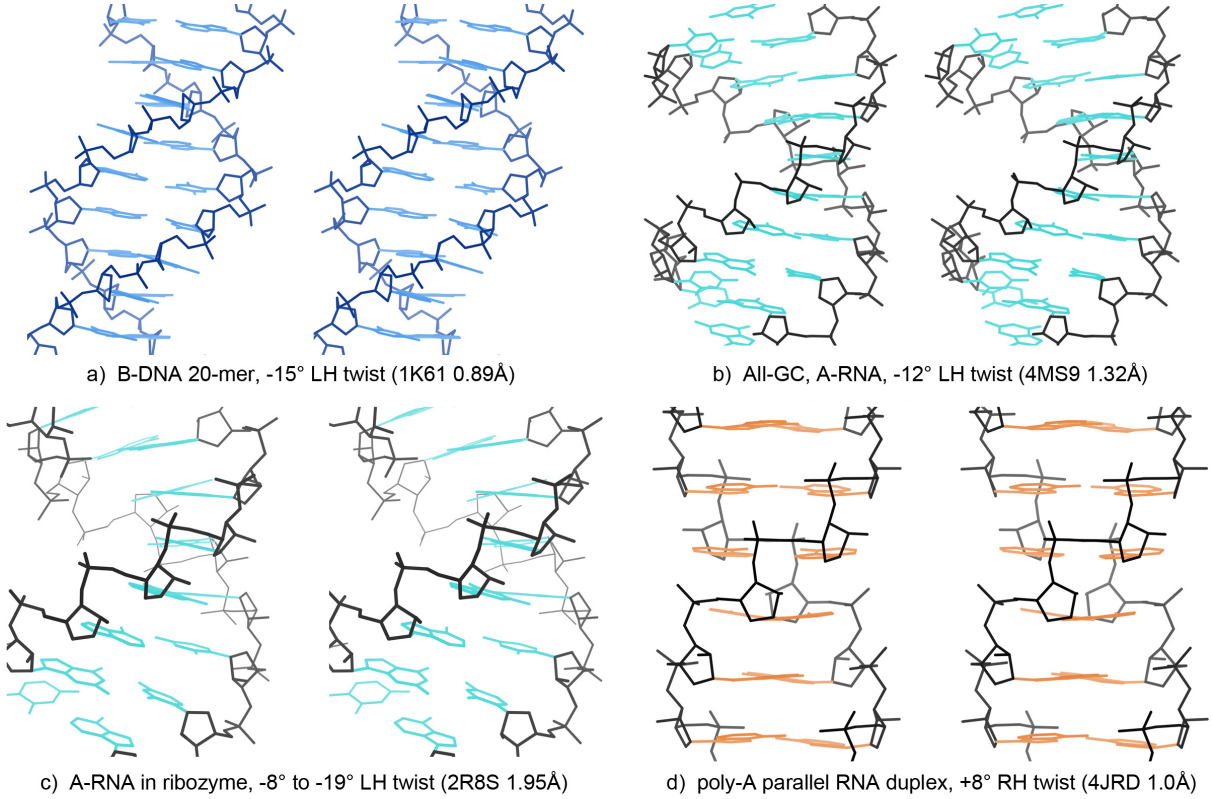


Figure 4: Basepair propeller-twist typical of different double-helix forms, best visible along basepairs where stands cross.

figure 6. When only a single H-bond joins the pair, the twist is presumably free to follow whatever else is needed, as in figure 7a. Note, though, that a pair between two bases adjacent in sequence can only make one (non-canonical) H-bond but is usually close to co-planar, as occurs in S-motifs or dinucleotide platforms. For 3-H-bond pairs the maximum twist is lower but often significant, as for the -24° propeller-twist of the GC pair in figure 7b.

Context consequences, and conservation of non-coplanar base pairs

Twist and buckle of base pairs can best be understood as a somewhat flexible tie through the H-bonds, dominated by the directionality of each stack that contains one base of the pair. Within a helix, the two stacks are close in direction, but have a relative twist characteristic of the helix type. In a helix junction, those two base stacks almost never come from the same direction. Figure 8 shows two cases of cross-stack relationships in helix junctions of an intron (2R8S) and an aptamer (3DD2; Long 2008), with one stack in orange and the other in blue, and joined at the junction by one or more base pairs (purple H-bond dots) twisted + or - by the torsion angle between the stack directions. Such places are among the most

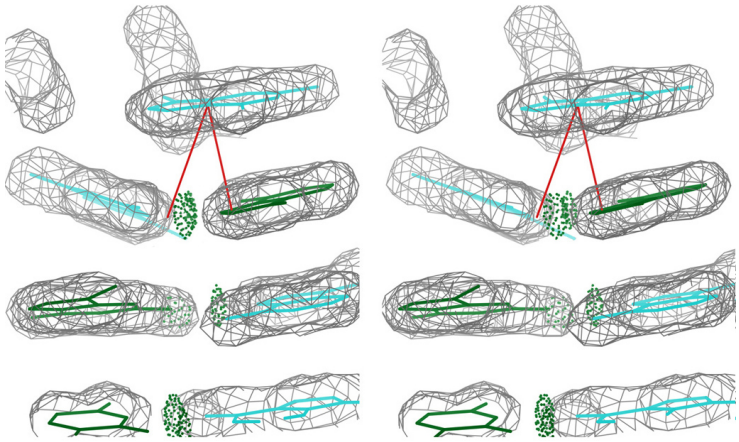
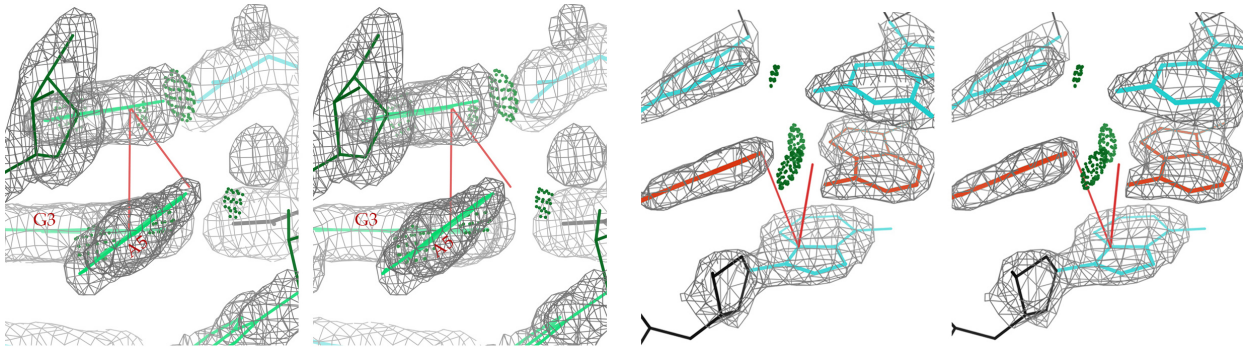


Figure 5: 33° buckle of base-pair in kink turn. Base normal in red (1SDS 1.8Å).



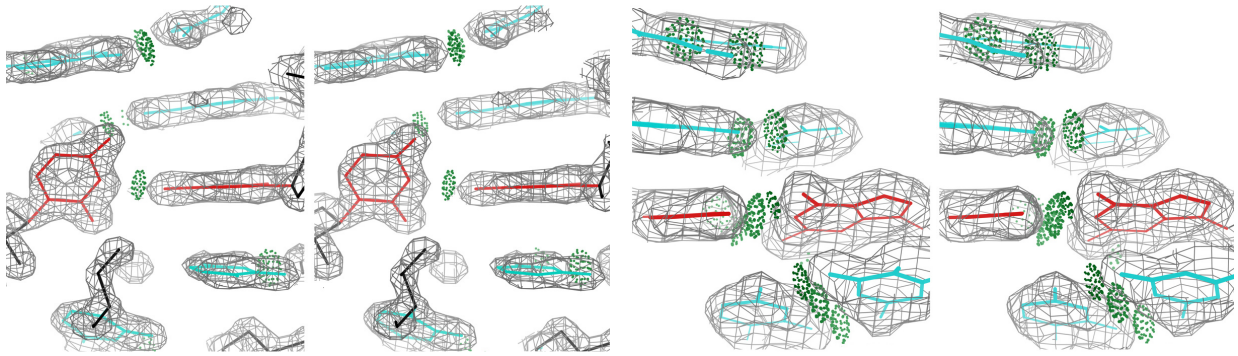
a) +37° twisted G-A base pair in splice-site/ZnF complex (3G9Y 1.4Å)

b) +43° twisted A-A base pair at helix junction in ribozyme (2R8S)

Figure 6: Stereos of high propeller-twist RNA base-pairs with 1 good H- buckle of base-pair in kink turn. Base normal in red (1SDS 1.8Å).

functionally important for large RNAs and their complexes, and sometimes also for DNAs.

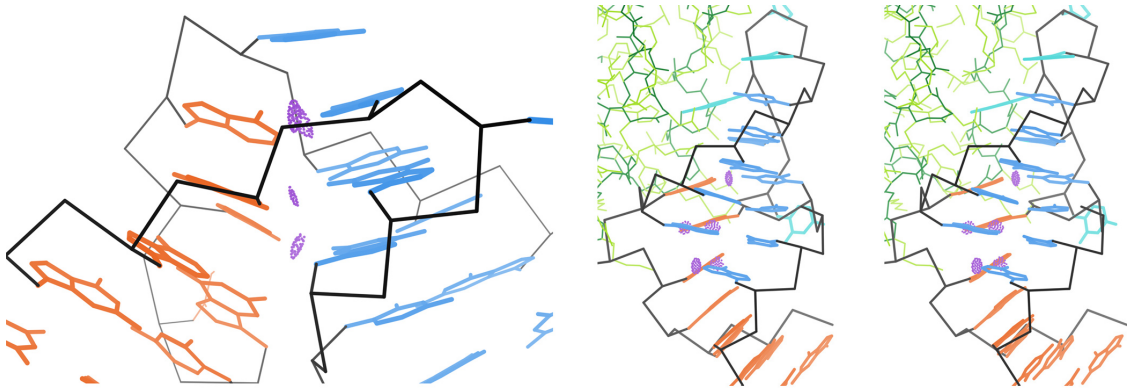
Complex regions of nucleic acid tertiary structure usually involve strongly non-coplanar base pairs and are the glue for forming specific large molecules and the arrangements that enable catalysis and binding. The strongly non-coplanar base pairs (or poorly stacked successive bases) are highlighted in red in figure 9 for overview examples of a riboswitch (4FEJ; Stoddard 2013) and a DNA aptamer (3ZH2; Cheung



a) Free twist (52°) for a 1-H-bond base pair (4FEJ riboswitch 1.5Å)

b) Lower maximum twist (-24°) for GC 3-H-bond base pair (2R8S)

Figure 7: Differing maximum propeller-twist as a function of number of base-pair H-bonds.



a) + twist helix-junction base pairs, P4P6 intron (2R8S)

b) - twist junction base pairs, thrombin RNA aptamer (3DD2)

Figure 8: Twisted cross-stack pairing across helix junctions. Stacks orange & blue, cross-stack H-bonds purple, protein green.

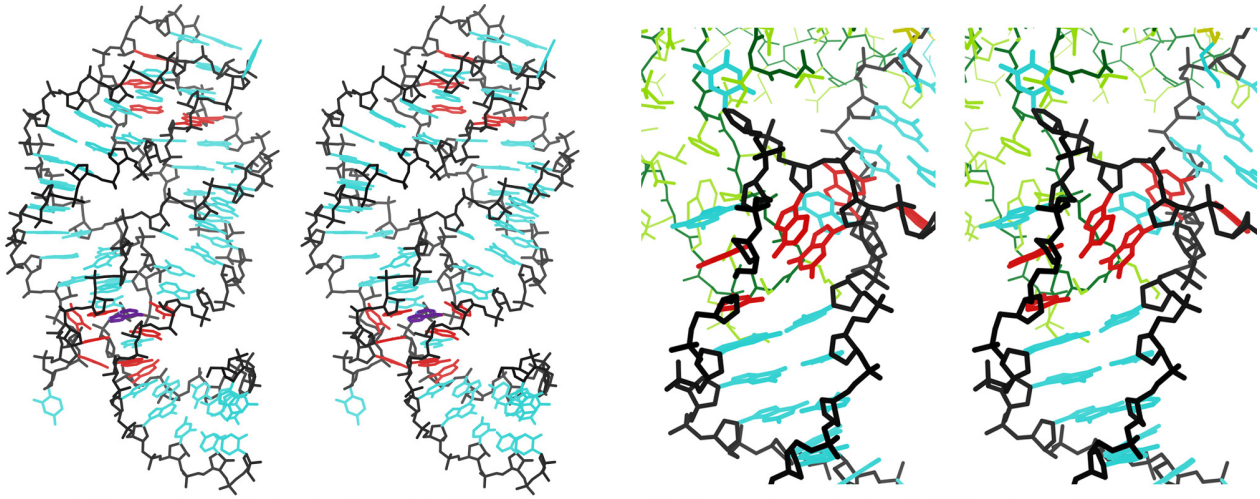


Figure 9: Context of strongly non-coplanar base-pair or non-stacked base clusters (red); effector purple, protein green.

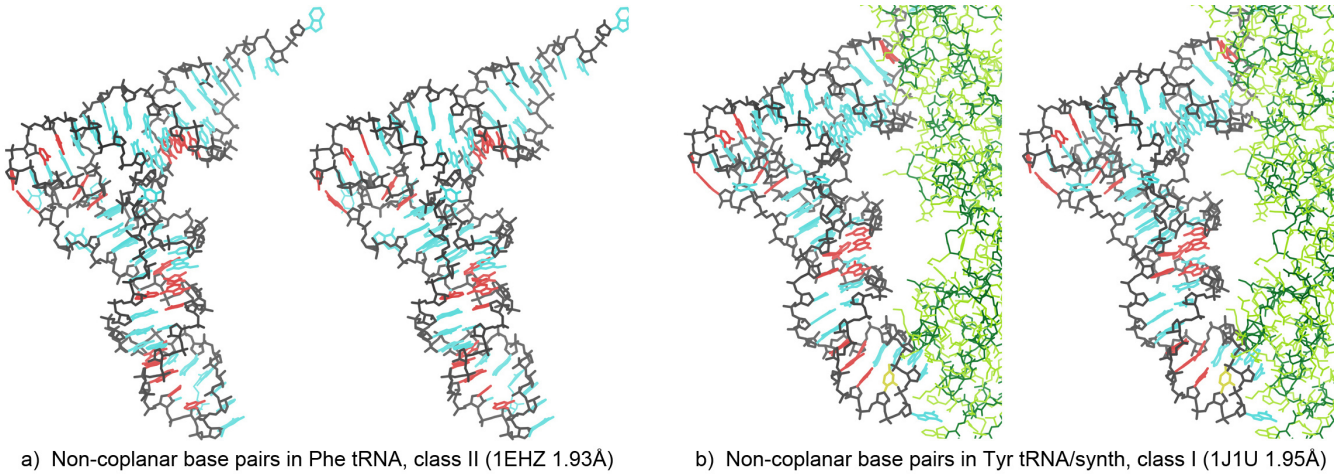


Figure 10: Conservation of 4 of the 5 clusters of non-coplanar base-pairs (red), between tRNA of the two distinct synthetase classes.

2013), to show their concentration at the functional binding sites.

Such clusters of highly non-coplanar or non-parallel bases are evolutionarily conserved, more so than the more obvious irregularities of flipped-out bases. As an example, figure 10 compares the clusters in tRNAs that are charged by tRNA synthetases from the two very different classes, Class I for the Tyr tRNA (1J1U; Kobayashi 2003) and Class II for the classic Phe tRNA (1EHZ; Shi 2000).

Conclusions

Very high-resolution structures show accurately the relationships of base-pair geometry and their response to overall context in the molecule, and those rules of thumb can be used at lower resolution to inform model-building and refinement protocols and parameters.

Base stacking is a strong interaction; when backbone conformation allows, stacking is nearly always parallel and at the 3.5Å separation optimal for van der Waals interaction. In contrast, base pairs are seldom completely coplanar, with each type of double-helix having its own characteristic value of base-pair propeller-twist, from -15° to +11°. In tertiary-structure interactions such as helix junctions, or in molecular

binding sites, base-pair twist or buckle can be quite large — it is controlled by the rather flexible base-pair H-bonds and the dominant effect of directionality of the stacking for each of the two bases. Those strongly non-coplanar base pairs tend to occur in places critical to forming the overall molecular structure or to providing its biological functionality.

Notes on measures and figures

The structures were displayed and examined in KiNG (Chen 2009). $2mF_{obs}$-dF_{calc} electron density maps (contours at 1.2σ) were downloaded from the Electron Density Server (Kleywegt 2004; eds.bmc.uu.se). All-atom contacts for hydrogen bonds (pillows of dark green dots) and van der Waals interactions (blue and green dots) were calculated on the MolProbity website (Chen 2010; molprobity.biochem.duke.edu). Base normals were constructed in Mage (Richardson 2001) and twist torsions and buckle angles were measured from them; propeller-twists were measured as dihedral angles around an axis between N1/9 atoms.

References

Aishima J, Gitti RK, Noah JE, Gan HH, Schlick T, Wolberger C (2002) A Hoogsteen base pair embedded in undistorted B-DNA, Nucleic Acids Res 30: 5244-5252 (1K61)

Chen VB, Davis IW and Richardson DC (2009) KiNG (Kinemage, Next Generation: A versatile interactive molecular and scientific visualization program, Protein Sci 18: 2403-2409

Chen VB, Arendall WB III, Headd JJ, Keedy DA, Immormino RM, Kapral GJ, Murray LW, Richardson JS, Richardson DC (2010) MolProbity: all-atom structure validation for macromolecular crystallography, Acta Crystallogr D66: 12-21

Cheung YW, Kwok J, Law AWL, Watt RM, Kotaka M, Tanner JA (2013) Structural basis for discriminatory recognition of Plasmodium lactate dehydrogenase by a DNA aptamer, Proc Natl Acad Sci USA 110: 15967 (3ZH2)

Creighton T (2011) The Biophysical Chemistry of Nucleic Acids, Helvetian Press

Hamma T, Ferre-D'Amare A (2004) Structure of protein L7Ae bound to a K-turn derived from an archaeal box H/ACA sRNA at 1.8 A resolution, Structure 12: 893-903 (1SDS)

Kleywegt GJ, Harris MR, Zou JY, Taylor TC, Wählby A, Jones TA (2004) The Uppsala Electron-Density Server, Acta Crystallogr D60: 2240-2249

Kobayashi T, Nureki O, Ishitani R, Yaremchuk A, Tukalo M, Cusack S, Sakamoto K, Yokoyama S (2003) Structural basis for orthogonal tRNA specificities of tyrosyl-tRNA synthetases for genetic code expansion, Nature Struct Biol 10: 425-432 (1J1U)

Long, S.B., Long, M.B., White, R.R., Sullenger, B.A (2008) Crystal structure of an RNA aptamer bound to thrombin, RNA 14: 2504-2512 (3DD2)

Loughlin FE, Mansfield RE, Vaz PM, McGrath AP, Setiyaputra S, Gamsjaeger R, Chen ES, Morris BJ, Guss JM, Mackay JP (2009) The zinc fingers of the SR-like protein ZRANB2 are single-stranded RNA-binding domains that recognize 5' splice site-like sequences, Proc Natl Acad Sci USA 106: 5581-5586 (3G9Y)

Luo Z, Dauter M, Dauter Z (2014) Phosphates in the Z-DNA dodecamer are flexible, but their P-SAD signal is sufficient for structure solution, Acta Crystallogr D70: 1790-1800 (4OCB)

Olson WK, Bansal M, Burley SK, Dickerson RE, Gerstein M, Harvey SC, Heinemann U, Lu X-J, Neidle S, Shakked Z, Sklenar H, Suzuki M, Tung C-S, Westhof E, Wolberger C, Berman HM (2001) A standard reference frame for the description of nucleic-acid base-pair geometry, J Mol Biol 313:: 229-237

Richardson DC, Richardson JS (2001) MAGE, PROBE, and Kinemages, Chap. 25.2.8 in International Tables for Crystallography Vol. F (Eds. M.G. Rossmann and E. Arnold), pp 727-730. Kluwer Academic Publishers, The Netherlands, Dordrecht

Saenger W (1984) Principles of Nucleic Acid Structure, Springer-Verlag

Safaee N, Noronha AM, Rodionov D, Kozlov G, Wilds CJ, Sheldrick GM, Gehring KB (2013) Crystal Structure of the Parallel Double-Stranded Helix of Poly(A) RNA, Angew Chem Int Ed Engl DOI: 10.1002/anie.201303461 (4JRD)

Sheng J, Li L, Engelhart AE, Gan J, Wang J, Szostak JW (2014) Structural insights into the effects of 2'-5' linkages on the RNA duplex, Proc Natl Acad Sci USA 111: 3050-3055 (4MS9, the native 10-mer)

Shi H, Moore PB (2000) Crystal structure of yeast phenylalanine tRNA at 1.93 A resolution: a classic structure revisited, RNA 6: 1091-1105 (1EHZ)

Stoddard CD, Widmann J, Trausch JJ, Marcano-Velazquez JG, Knight R, Batey RT (2013) Nucleotides adjacent to the ligand-binding pocket are linked to activity tuning in the purine riboswitch, J Mol Biol 425: 1596-1611 (4FEJ)

Word JM, Lovell SC, LaBean TH, Zalis ME, Presley BK, Richardson JS, Richardson DC (1999) "Visualizing and Quantitating Molecular Goodness-of-Fit: Small-probe Contact Dots with Explicit Hydrogen Atoms", *J Mol Biol* **285**: 1711-1733

Ye JD, Tereshko V, Frederiksen JK, Koide A, Fellouse FA, Sidhu SS, Koide S, Kossiakoff AA, Piccirilli JA (2008) Synthetic antibodies for specific recognition and crystallization of structured RNA, Proc Natl Acad Sci USA 105:82-87 (2R8S)