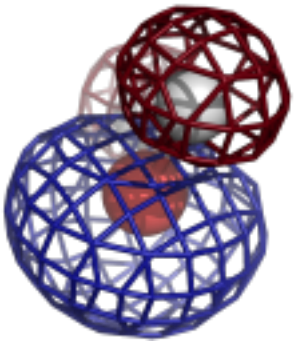
VOLUME THREE



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXII

CARBOHYDRATES, SYMMETRY MATES, HYDROGENS

Table of Contents

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

PHENIX News

New releases

phaser.MRage: molecular replacement pipeline

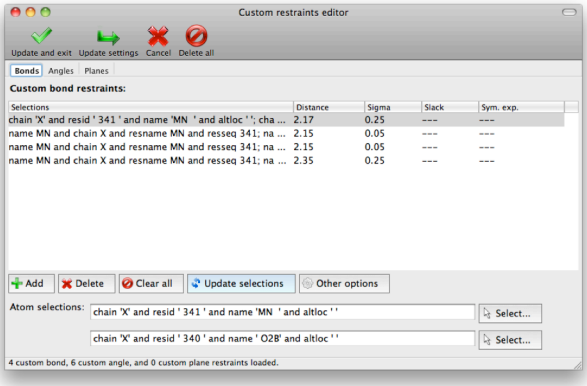
A versatile program for automatic molecular replacement, pronounced mirage, has been added to *PHENIX*. It integrates model search and generation utilities with the molecular replacement program *Phaser*. Search models can be specified in several stages. It is possible to input models that are used as 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.

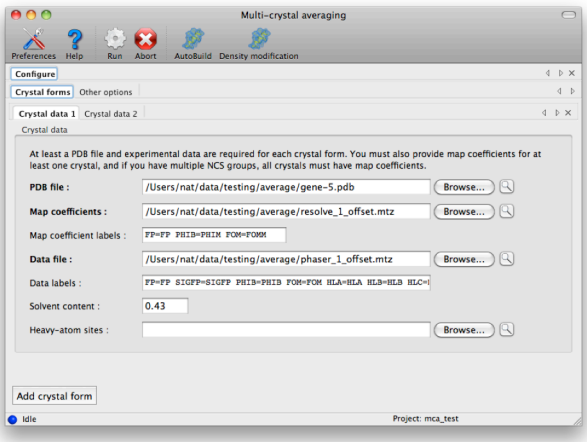
New features

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.

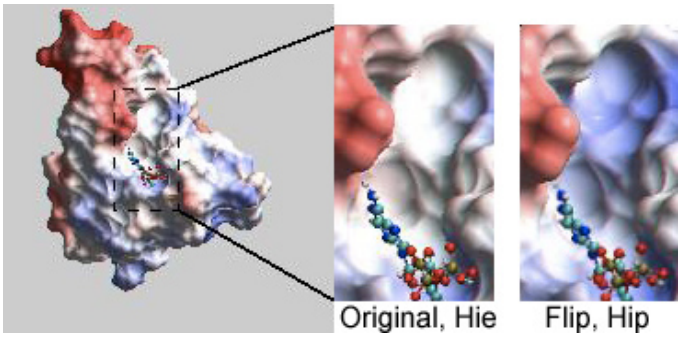


Figure 1: PDB entry `1mjh` showing electrostatic surface near `His40`.

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

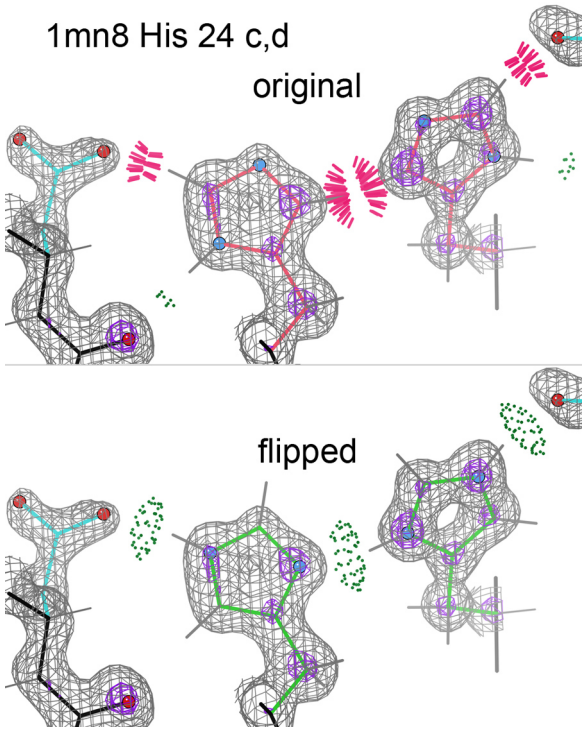


Figure 2: Improvements after flipping a residue.

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.

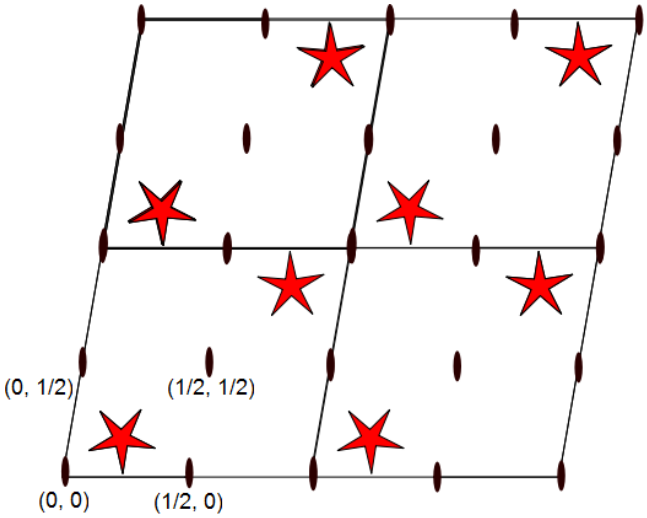


Figure 1: A two-dimensional crystal with a two-fold axis, symmetry generated copies as well as alternative origins.

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.

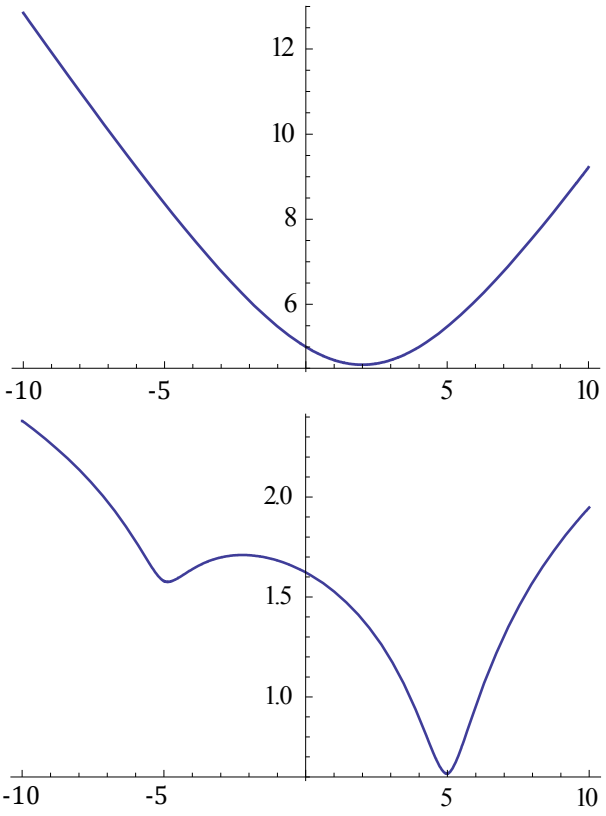


Figure 3: (Upper) The RMSD values of the hypothetical ten red and blue atoms in figure 2 as a function of sliding along the a-axis. (Lower) The MLAD values of the hypothetical ten red and blue atoms in figure 2 as a function of sliding along the a-axis.

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.

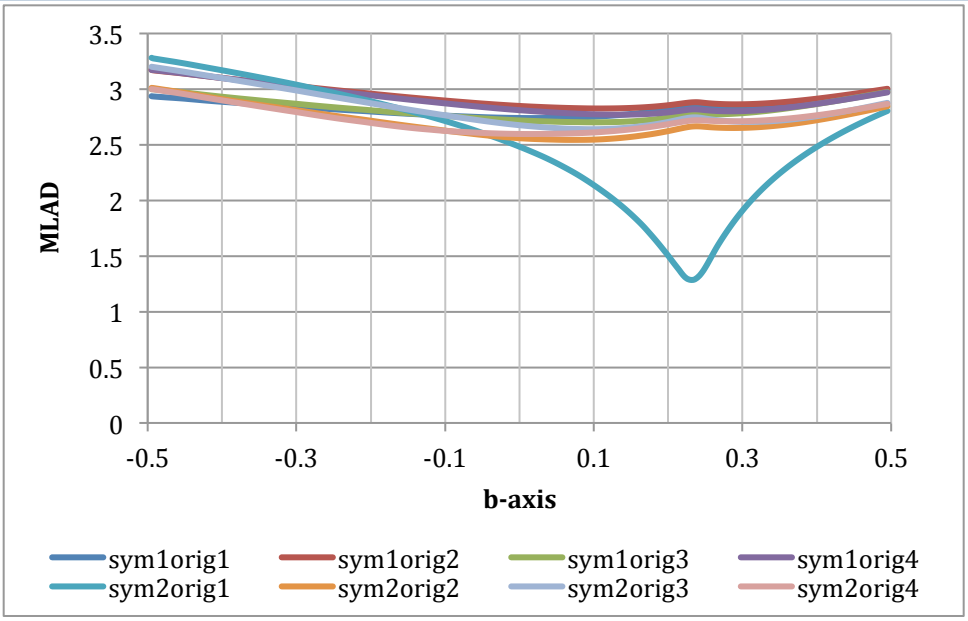


Figure 4: MLAD values between the structures with PDB code 1CM3 and a molecular replacement solution with a model derived from 1Y50 as a function of polar axis position in fractional coordinates for the eight combinations of symmetry transformation and choice of origin.

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

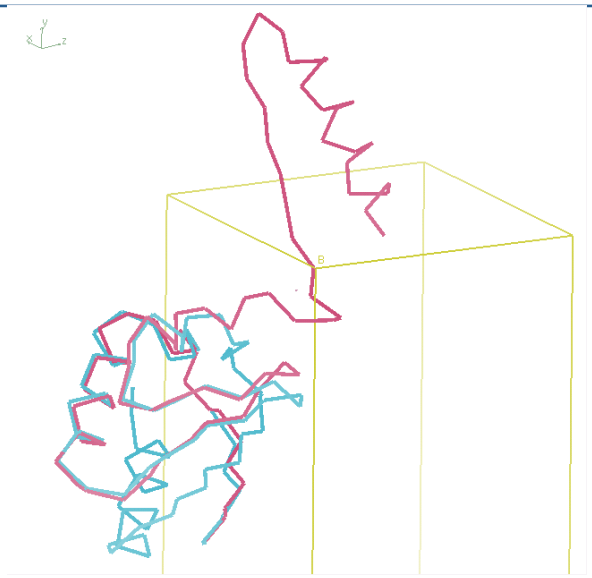


Figure 5: Cα-trace of the structures with PDB code 1CM3 (red) and a molecular replacement solution with a model derived from 1Y50 (blue) where the position along the polar axis was found to yield the smallest MLAD score among all possible symmetry copies on alternative origin for this crystal.

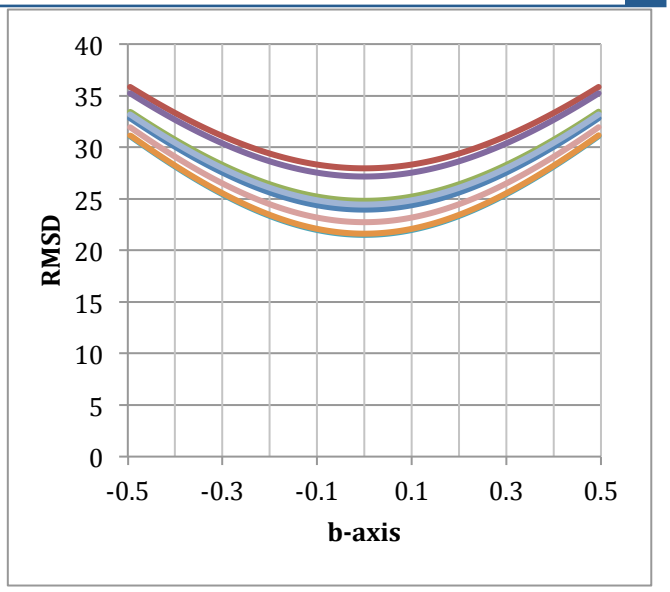


Figure 6: RMSD values between the structures with PDB code 1CM3 and a molecular replacement solution with a model derived from 1Y50 as a function of polar axis position in fractional coordinates for the eight combinations of symmetry transformation and choice of origin. Same legend as figure 4.

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

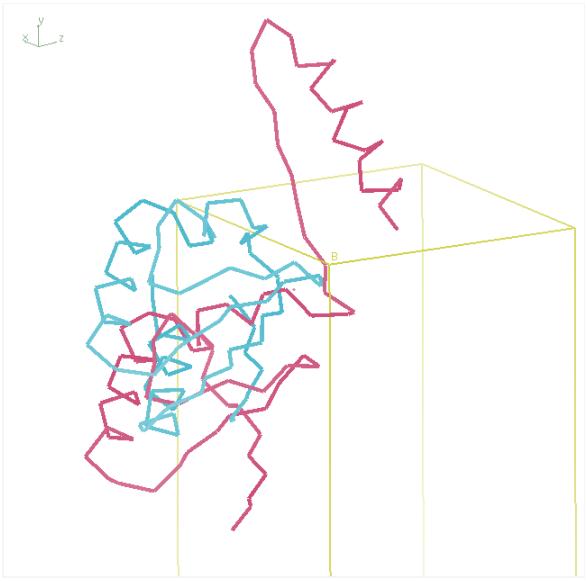


Figure 7: Cα-traces of the structures with PDB code 1CM3 (red) and a molecular replacement solution with a model derived from 1Y50 (blue) where the position along the polar axis was found to yield the smallest RMSD score among all possible symmetry copies on alternative origin for this crystal.

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).

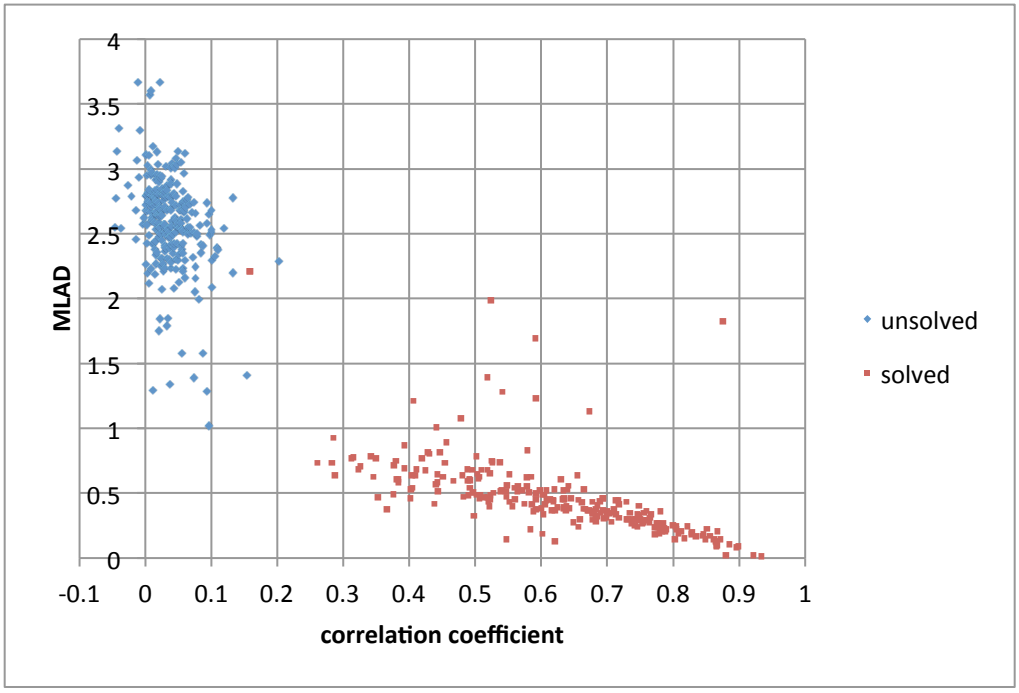


Figure 8: Map correlation coefficients and the corresponding MLAD values for 520 random molecular replacement calculations between target structures and their solutions.

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

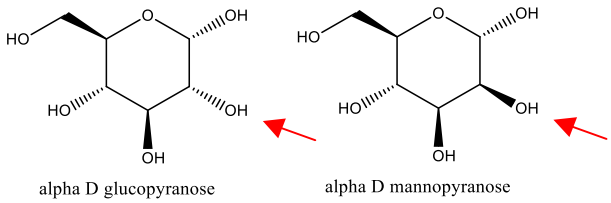


Figure 1. The position of a single hydroxyl group differentiates between α-D-glucopyranose and α-D-mannopyranose.

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

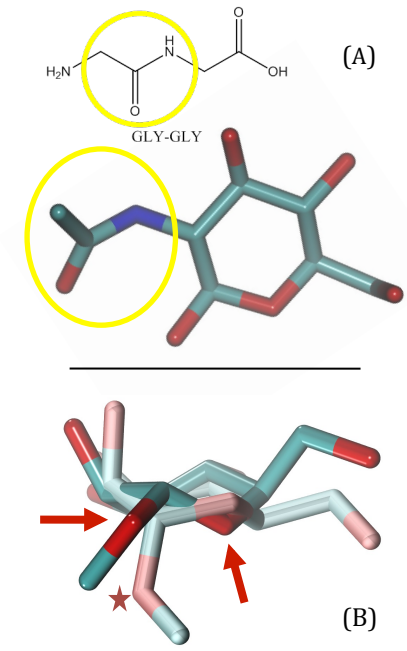


Figure 2. Two examples of unrealistic geometries assigned to carbohydrates in the PDB. In the molecular models, hydrogen atoms are omitted, carbon is cyan, oxygen is red and nitrogen is blue. (A) β-D-N-acetyl glucosamine, residue 3321(NAG) in chain A of PDB ID 1RVZ. The bend at the nitrogen atom breaks 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

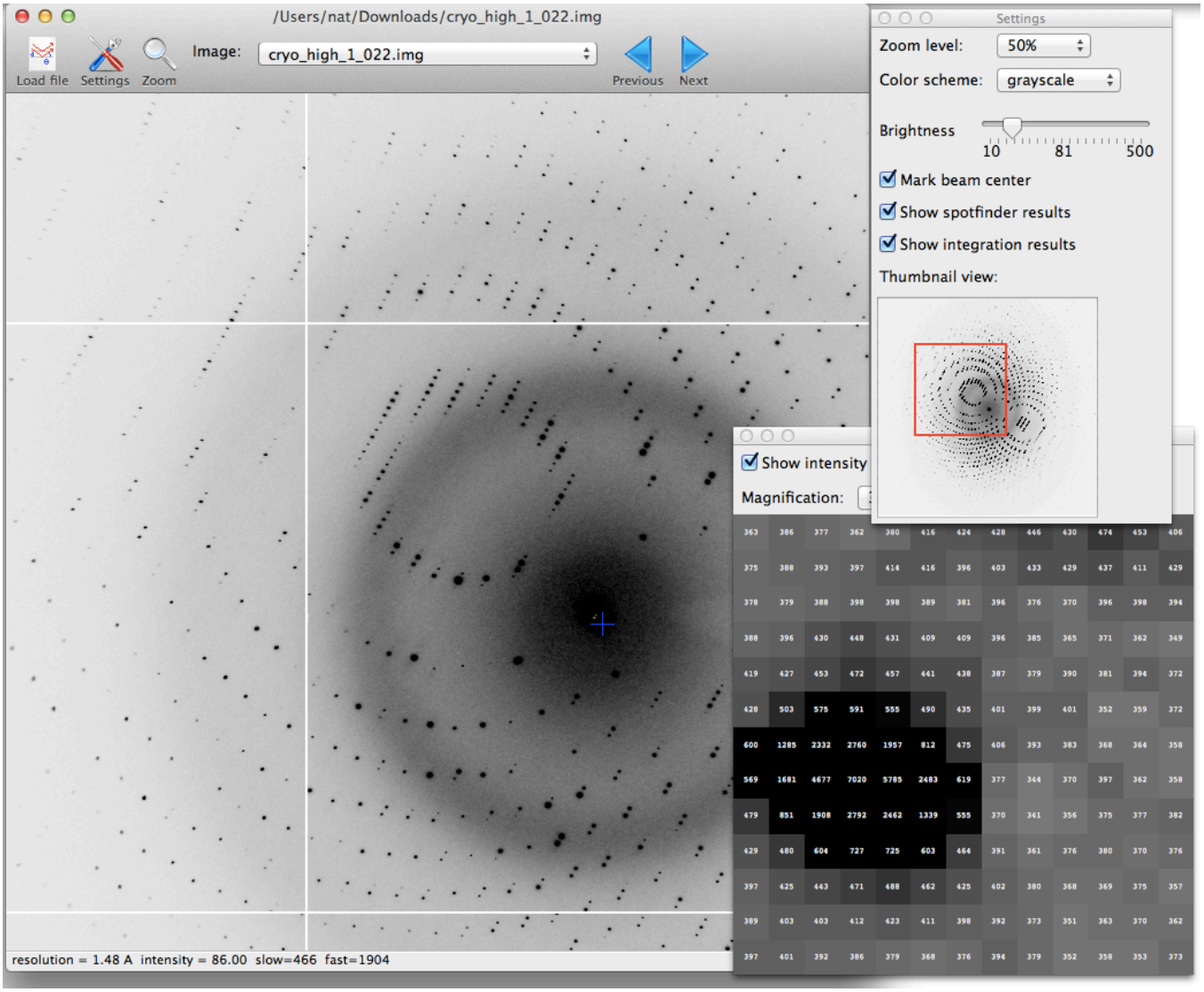


Figure 1: The main image viewer interface, showing a diffraction pattern from *E. coli* dihydrofolate reductase (provided by James Fraser, UCSF). The zoom window is visible at lower right.

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

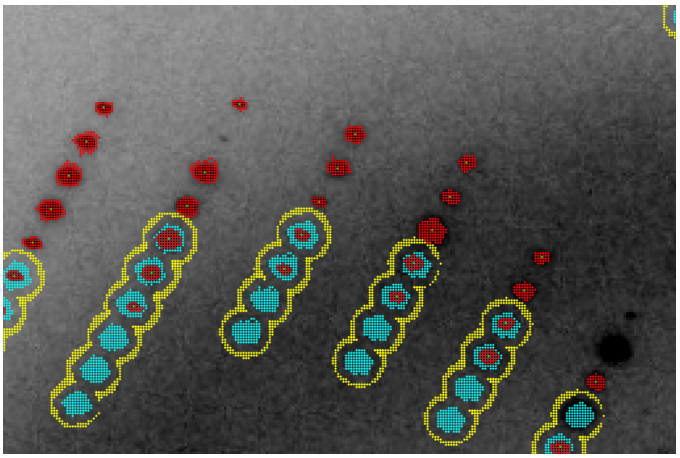


Figure 2: Closeup of the diffraction pattern shown in Figure 1 with the spotfinder results (red), integration masks (cyan), and background masks (yellow) overlaid.

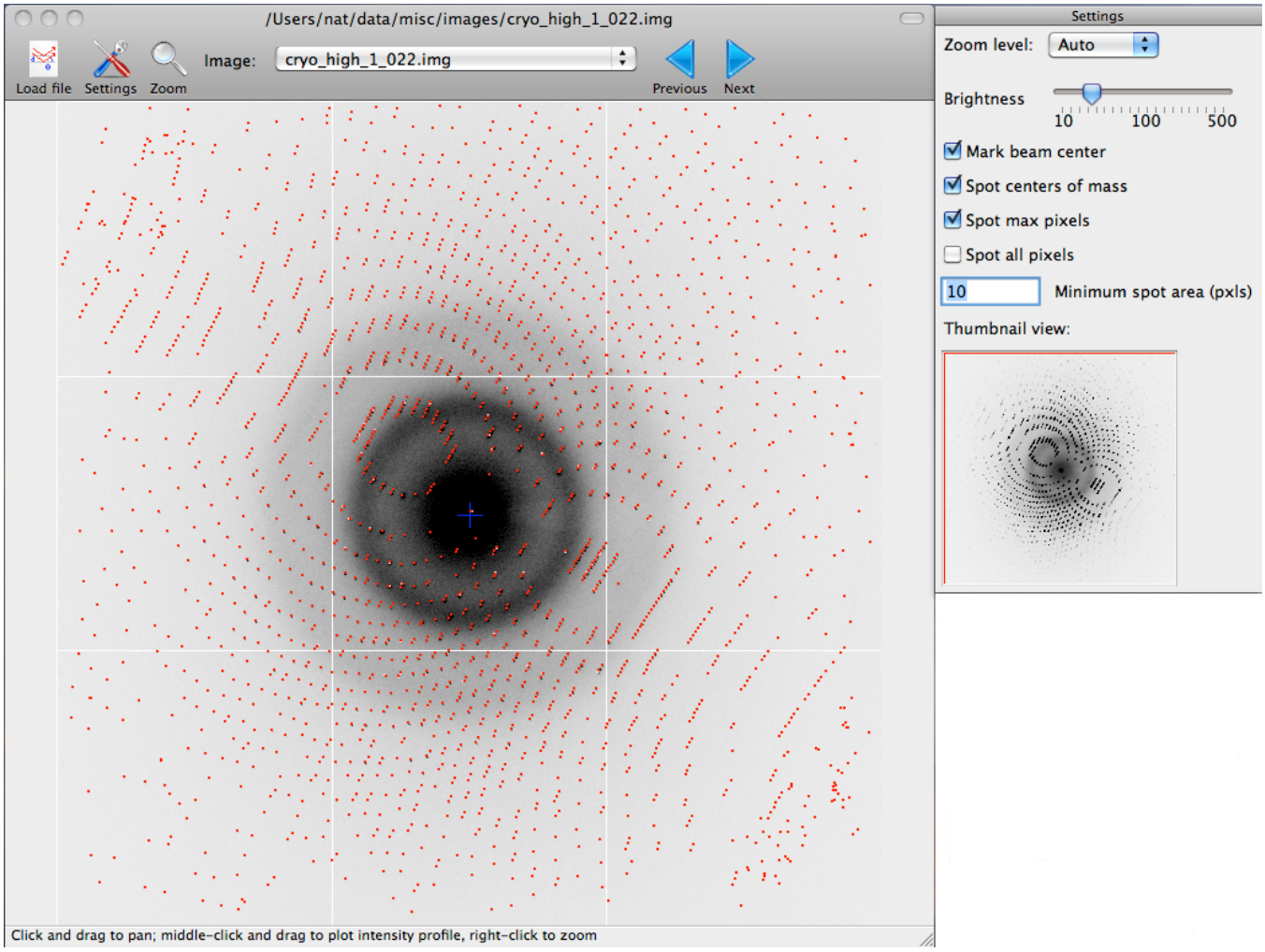


Figure 3: The spotfinder front end, *distl.image_viewer*.

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.

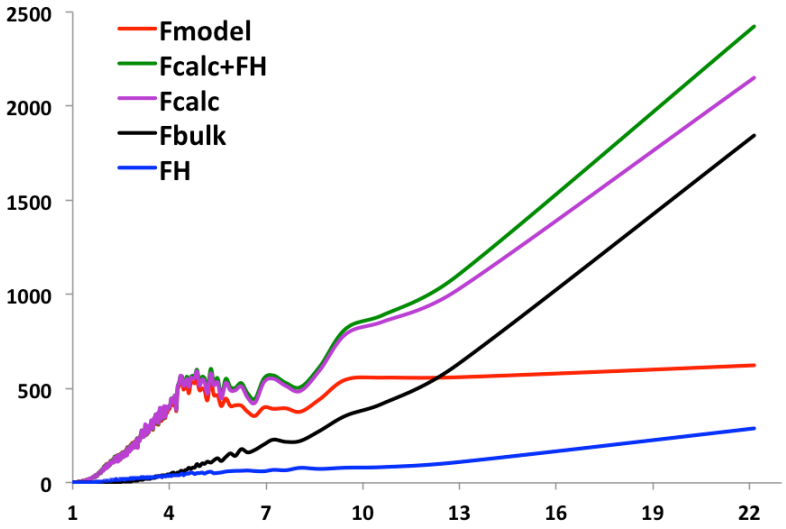


Figure 1. Resolution bin averaged values of each term in formula 1 calculated for a structure with code 1F8T using all theoretically possible reflections up to 1Å resolution.

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

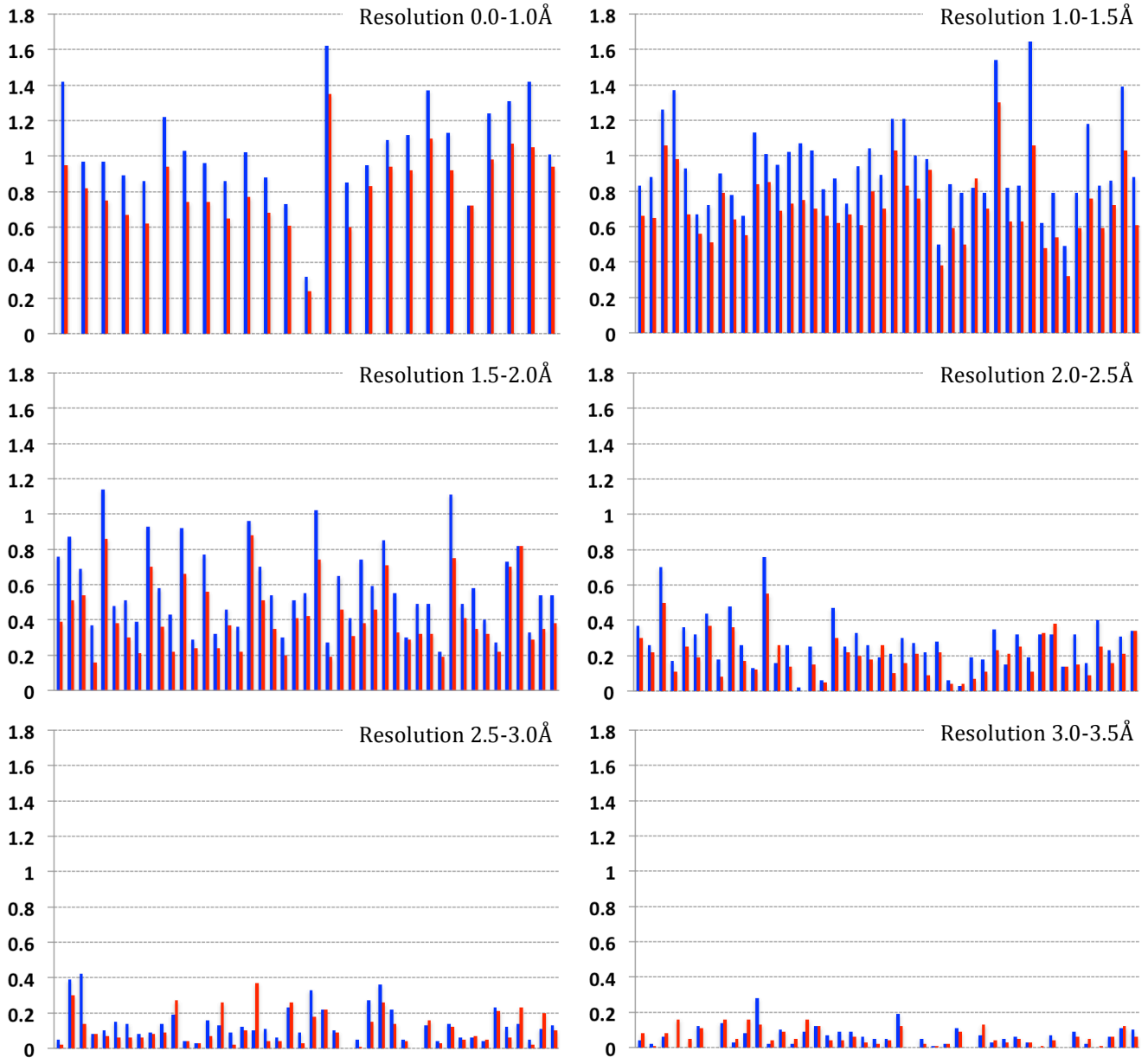


Figure 2. See text for details. R_{noH}-R_H(blue) and R_{noH}-R_{H_unique} (red). PDB codes of structures used in calculations in the order of their appearance in the plots; (0,1): *1mnz, 1nwz, 1r2m, 2ddx, 2e4t, 2fvy, 2gg2, 2gud, 2h5c, 2h5d, 2ov0, 2pne, 2ppp, 2rh2, 2zq7, 3agn, 3ago, 3cnj, 3f7l, 3gyi, 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).

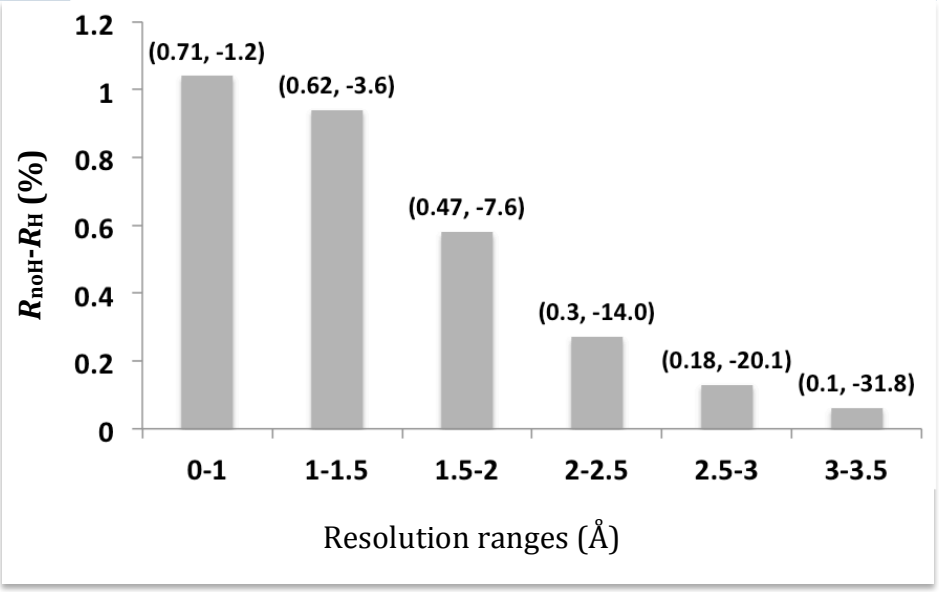


Figure 3. Averaged R_{noH}-R_H values shown for six resolution ranges. Each bar caption shows corresponding bin-averaged pairs of (k_h , B_h).

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*

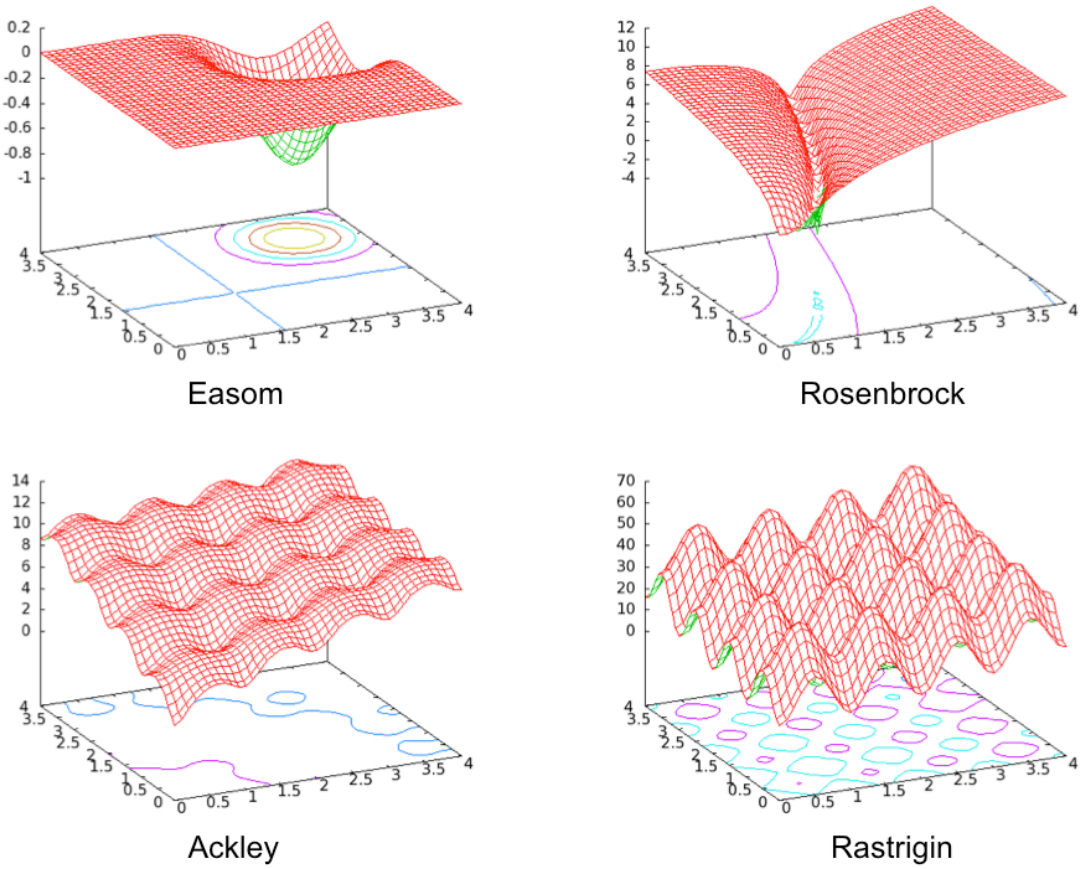


Figure 1: Test functions for optimization routines.

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.