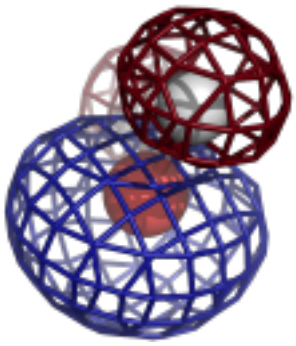
VOLUME ELEVEN



COMPUTATIONAL CRYSTALLOGRAPHY NEWSLETTER

JANUARY MMXX

ENSEMBLE REFINEMENT

Table of Contents

Editor

Nigel W. Moriarty, NWMoriarty@LBL.Gov

Phenix News

Announcements

New Phenix Release

Highlights for the 1.17 version of Phenix include:

- Improved handling of SHELX data in *phenix.reflection_file_converter*
- *eLBOW* can output files for Amber and supports the Orca QM package
- *dials.image_viewer* is used for viewing diffraction images
- Updated map smoothing
- Fix inconsistency in clashscore values in *phenix.validation_cryoem* when hydrogen atoms are in the model

Please note that this new publication should be used to cite the use of Phenix:

Macromolecular structure determination using X-rays, neutrons and electrons: recent developments in Phenix. Liebschner D, Afonine PV, Baker ML, Bunkóczi G, Chen VB, Croll TI, Hintze B, Hung LW, Jain S, McCoy AJ, Moriarty NW, Oeffner RD, Poon BK, Prisant MG, Read RJ, Richardson JS, Richardson DC, Sammito MD, Sobolev OV, Stockwell DH, Terwilliger TC, Urzhumtsev AG, Videau LL, Williams CJ, Adams PD: Acta Cryst. (2019). D75, 861-877.

A new tool, *phenix.homology*, is available in the nightly and discussed on page 5 of this newsletter.

Downloads, documentation and changes are available at phenix-online.org

Expert advice

Fitting Tip #19 – Remember to use the information from NCS copies

Christopher Williams and Jane Richardson, Duke University

Multiple copies of the molecule in a crystal asymmetric unit or in a cryoEM 3D reconstruction are both an advantage and a disadvantage. Although NCS makes the structure very much larger, a big advantage is that if density is uninterpretable for the individual copies, you can often fit a single model to the averaged map. For less extreme cases, in Phenix you can torsionally restrain the copies to match one another, with a "top-out" function to loosen the restraints where they diverge strongly (Headd 2012). However, in our experience a large fraction of structures with NCS are built and refined as independent copies that seldom seem to have been compared with each other afterward. That final comparison step is a very powerful advantage that should not be skipped (even if top-out restrained), since comparison can separate real differences from modeling-error differences and can straightforwardly correct the errors that differ.

Individual problem areas

At high to mid resolution, differences between unsymmetrized NCS copies are most commonly for sidechain conformations or for differently disordered loops or chain ends. At low resolution (2.5-4Å), in addition one often sees significant problems in the backbone even in relatively well-ordered parts of the molecule. An example is shown in Figure 1, where each chain fits correctly the region of the other's error. At low resolution, these validation displays benefit greatly from the peptide-orientation analysis of CaBLAM, which integrates information across five residues. Traditional outliers signal problems if they are present, but their absence does not mean all is well, because the broad density is compatible with models that refine away outliers without fixing the underlying problems. Once identified, many of those problems can be fixed in Coot (Emsley 2010) with peptide-flip or crankshaft moves, such as a CaBLAM inside secondary structure, or two CaBLAM outliers in a row (where the central CO's peptide's should be rotated).

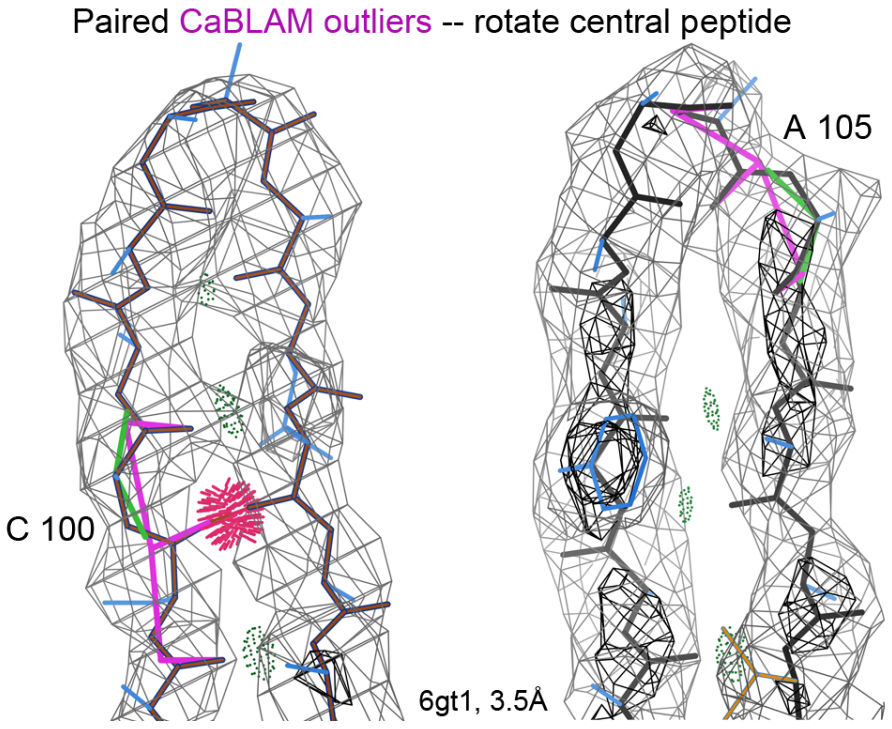


Figure 1: Models of NCS copies often have outlier problems in different places, as for this β-hairpin in chain A vs C of the 6gt1 Nek7 kinase at 3.5Å resolution (Byrne 2018). Hotpink spikes flag all-atom clashes >0.4Å, green lines are Ramachandran outliers, and the magenta dihedrals between backbone COs flag CaBLAM outliers. Gray contours are 2mFo-DFc electron density at 1.2σ and black ones at 3σ, and pillows of dark green dots are hydrogen bonds.

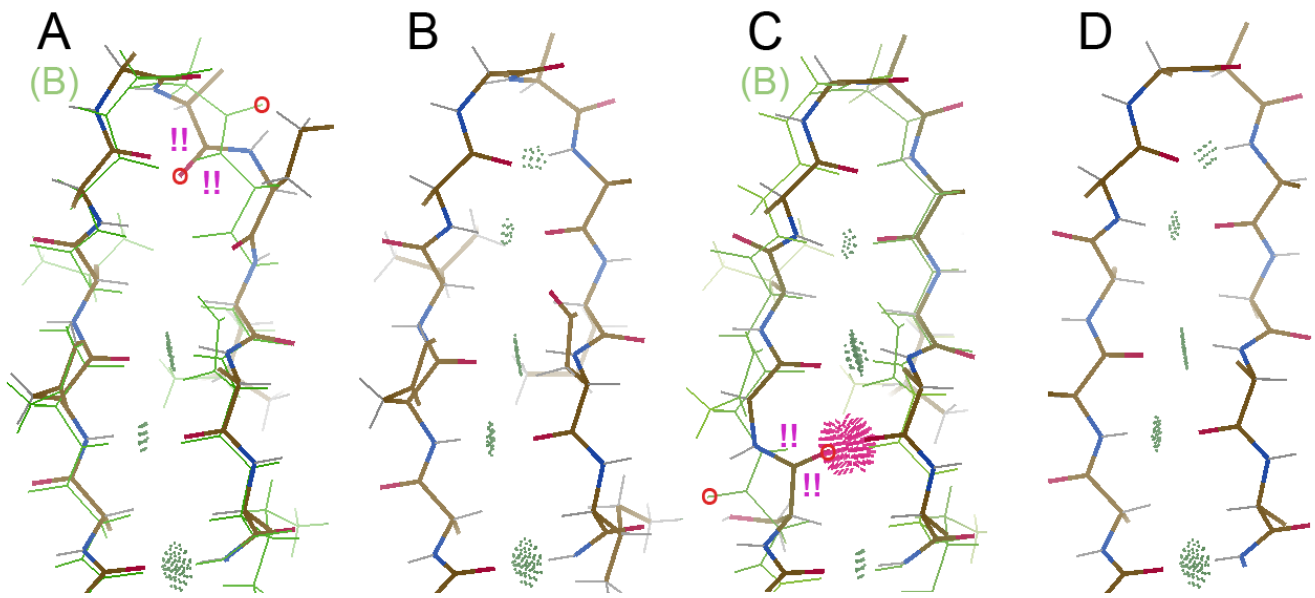


Figure 2: The same β-hairpin from 6gt1, shown for each of the 4 chains. The B chain has the best electron density and D the worst (its clean conformation was apparently copied from B). Parts A and C show a green ghost of chain B that guides how each of the errors can be fixed. The carbonyl O atoms that need rotation are marked with a red O in original and corrected positions, and the original CaBLAM outliers are represented by **!!** in magenta.

Chain C is especially easy, with two CaBLAM outliers and a huge clash flagging a classic error that happens at resolutions where bumps for the CO groups have disappeared: in a beta-strand, three COs in a row are incorrectly pointed in the same direction rather than alternating (Chen 2011). Chain A has a more complex problem at what should be a tight turn, but also rotating the central CO (105) is the first move, followed by fixing a bad-geometry adjacent group that clashes with the corrected CO and then real-space refinement of the surrounding stretch (say, 102-107). When a peptide orientation is seriously wrong, it also distorts sidechain positioning and makes sequence misalignment more likely.

[Note: KiNG (Chen 2009) is used for Figure 1, since Coot does not yet flag CaBLAM outliers. Within Phenix, the cryoEM validation GUI lists CaBLAM outliers, each with a link to center there in Coot. On the MolProbity web site (Prisant 2020), the CaBLAM outliers are included in the 3D multi-kin displayed online in the modified NGL Viewer (Rose 2015).]

Compare the NCS copies

Individual corrections are the hard way to figure this out, however. Coot not only lets you cycle through the NCS copies to find the best one, it also lets you show a "ghost" of the best one superimposed on the problem copy. That ghost both guides and validates the correction process.

In this case, locally, chain B has no outliers and a fit to its density that is very reasonable for this resolution. Chains A and C each differ from the consensus conformation in a different way, and their evidently avoidable outliers mean that those differences are surely fitting errors, not genuine differences.

The bottom line

Most of the time, the quality of your model can be significantly improved if you use the information from comparing multiple NCS copies. If resolution or density quality is really poor, it is better to fit one model to an average map. But in most cases it is better to take advantage of having an ensemble (at least a proxy for uncertainty, and sometimes for mobility). But to get the benefit from this extra information requires a step near the end where you explicitly compare the differences between models. If they differ locally with no outlier flags, then probably the local differences are real, and perhaps of interest. If all copies of a local region have problems, it's worth a try at fixing the erros, but it's hard to know what's the right answer. In the frequent and most useful case where a local region differs in both conformation and validation, then a clean copy shows how to correct a problem copy and make your overall structure better. This is an important, productive step at either high or low resolution.

References:

Byrne MJ, Cunnison RF, Bhatia C, Bayliss RW (2018) Nek7 bound to purine inhibitor, unpublished [6gt1]

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

Chen V, Williams C, Richardson J (2011) "Fitting Tip #1: Not 3 parallel COs in a row", *Comp Cryst Newsletter* **2**: 3

Emsley P, Lohkamp B, Scott WG, Cowtan K (2010) Features and development of Coot, *Acta Cryst* **D66**:486-501

Headd JJ, Echols N, Afonine PA, Grosse-Kunstleve RW, Chen VB, Moriarty NW, Richardson DC, Richardson JS, Adams PD (2012) Use of knowledge-based restraints in *phenix.refine* to improve macromolecular refinement at low resolution, *Acta Cryst* **D68**: 381-390

Prisant MG, Williams CJ, Chen VB, Richardson JS, Richardson DC (2020) New tools in MolProbity validation: CaBLAM for cryoEM backbone, UnDowser to rethink "waters", and NGL Viewer to recapture online 3D graphics, *Protein Sci* **29**: 315-329 and bioRxiv 79516

Rose AS, Hildebrand PW (2015) NGL Viewer: A web application for molecular visualization, *Nucleic Acids Res* **43**: W576-W579

FAQ

Can I control the automatic linking?

When a model with poor geometry is provided to a Phenix program, the automatic linking may generate an unwanted link. This is because distance between the two entities plays a roll in the algorithm. The stop an unwanted link use

```
exclude_from_automatic_linking {
    selection_1 = None
    selection_2 = None
  }
```

to select the two entities to not create a link.

phenix.homology: finding high-resolution matches for low-resolution models at a chain level

Yanting Xu[a,b], Li-Wei Hung[b], Oleg V. Sobolev[b] and Pavel V. Afonine[b]
[a]*International Center for Quantum and Molecular Structures, Shanghai University, Shanghai 200444, China*
[b]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

Correspondence email: pafonine@LBL.Gov

With the rise of cryo-EM, low-resolution model building and refinement becames more frequent exercise. Modeling against low-resolution data is generally tedious and error-prone because the corresponding maps lack atomic and often secondary-structure level of details while containing noise thus allowing multiple interpretations. It is not uncommon, however, that the Protein Data Bank (PDB, Burley *et al*., 2019) contains an atomic model of a structure that is sequence-similar to the structure being studied obtained using a higher resolution data. In such cases the higher resolution model can be used to help low-resolution model building, refinement and validation (Headd *et al.*, 2012; van Beusekom *et al.*, 2018).

Here we present a new *Phenix* (Liebschner *et al*., 2019) tool, *phenix.homology*, that, given a low-resolution protein model or the corresponding sequence, can search the PDB for a set of highest-resolution models within a specified threshold of sequence identity. *phenix.homology* operates at a chain level: it searches for higher-resolution matches for each individual chain in the low-resolution model. *phenix.homology* makes use of BLAST sequence alignment tool (Altschul *et al.*, 1997) and *iotbx.bioinformatics* module of CCTBX (Grosse-Kunstleve *et al*., 2002).

Examples of usage scenarios:

– Given a low-resolution model (provided as model or sequence files) find *n* highest resolution models that satisfy the resolution and sequence identity criteria:

```
phenix.homology <model.pdb or sequence.txt> high_res=2 identity=95 n=3
```

– Find high-resolution matches for all low-resolution models in the PDB that satisfy specified resolution and sequence identity criteria:

```
phenix.homology low_res=3 high_res=2 identity=95 all_database=True
```

This command will iterate over all protein models of resolution 3Å or worse in the PDB. For each chain in the models, it will search for matching chains in all PDB entries of resolution 2Å or better and that have sequence identity above 95%.

There is a nuance about how sequence identity percentage is calculated depending on the presence and position of gaps in reference or matching models. Figure 1 illustrates five scenarios: (a) perfect match, (b) gap in the middle of reference model, (c,d) gap at either end of reference or matching model, (e) gap in the middle of matching model.

The parameter *piece_matching* controls how matching is done. If *piece_matching* is set to True, then leading and trailing gaps (Fig. 1c) are ignored in matching models. If *piece_matching* is set to False (the default), the leading and trailing gaps in matching models are accounted for in the matching. Gaps in the middle of the chain (Fig. 1b,e) are always accounted for regardless of the *piece_matching* setting. In case of matching chain being longer than reference chain, BLAST would only return aligned segment and report the identity (Fig 1d).

Another nuance is related to whether the sequence identity is considered for the whole model or per chain. This is governed by *chain_matching* parameter of *phenix.homology*. If *chain_matching* is set to True (default setting) then a match will be considered if at least one chain satisfies the sequence similarity criterion. If *chain_matching* is False then a match will not be considered if there is at least one chain out of the whole protein does not satisfy similarity criterion.

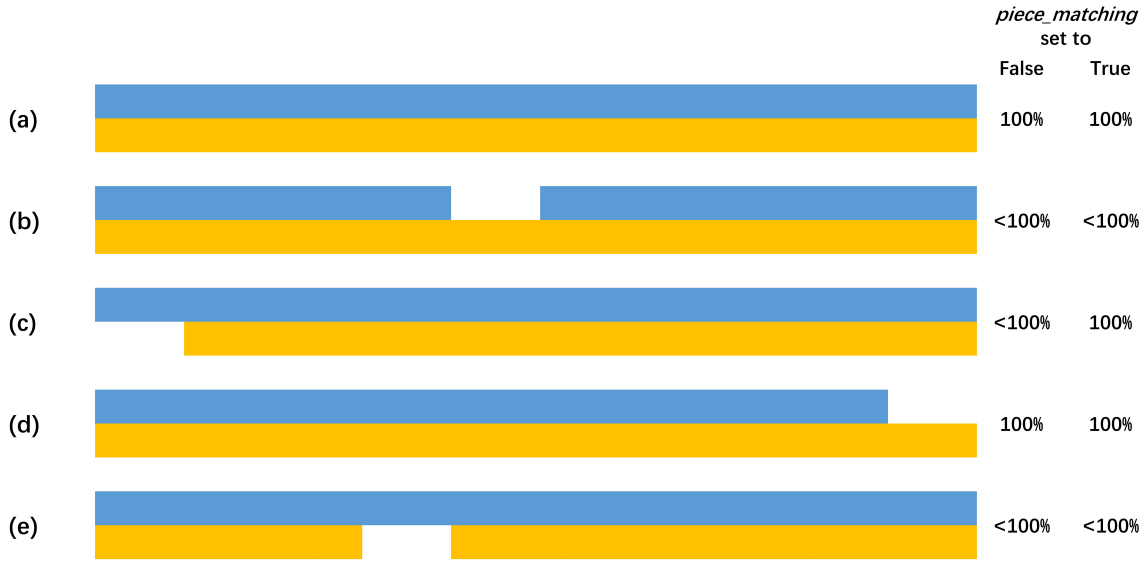


Figure 1: Example of matching scenarios: low resolution (blue), high resolution (yellow).

The tool is also accessible at the Python level:

```
from phenix.programs import homology
parameters = homology.get_default_params()
result = homology.run(sequence_string, parameters)
```

where the result contains matching high-resolution structure information: PDB code, chain ID, resolution and corresponding sequence identity.

References

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J. H., Zhang, Z., Miller, W. & Lipman, D. J. (1997). *Nucleic Acids Res* **25**, 3389-3402.

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Di Costanzo, L., Christie, C., Dalenberg, K., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranović, V., Guzenko, D., Hudson, B. P., Kalro, T., Liang, Y., Lowe, R., Namkoong, H., Peisach, E., Periskova, I., Prlić, A., Randle, C., Rose, A., Rose, P., Sala, R., Sekharan, M., Shao, C., Tan, L., Tao, Y.-P., Valasatava, Y., Voigt, M., Westbrook, J., Woo, J., Yang, H., Young, J., Zhuravleva, M. & Zardecki, C. (2019). Nucleic Acids Res. 47, D464–D474.

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

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). *Acta Crystallogr D Biol Crystallogr* **68**, 381-390.

Liebschner, D., P. V. Afonine, M. L. Baker, G. Bunkóczi, V. B. Chen, T. I. Croll, B. Hintze, L.-W. Hung, S. Jain, A. J. McCoy, N. W. Moriarty, R. D. Oeffner, B. K. Poon, M. G. Prisant, R. J. Read, J. S. Richardson, D. C. Richardson, M. D. Sammito, O. V. Sobolev, D. H. Stockwell, T. C. Terwilliger, A. G. Urzhumtsev, L. L. Videau, C. J. Williams, and P. D. Adams. (2019). Acta Cryst. **D75**, 861-877

van Beusekom, B., Joosten, K., Hekkelman, M. L., Joosten, R. P. & Perrakis, A. (2018). *Iucrj* **5**, 585-594.

Lessons from using the Cambridge Structure Database: I – Bond number specification

Nigel W. Moriarty[a*]

[a]Molecular Biosciences and Integrated Bioimaging, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

[*]Correspondence e-mail: nwmoriarty@lbl.gov

Preface

As a user of the Cambridge Structural Database (CSD), I had to learn to use the available interfaces. Several lessons were learned either by trial and error, or by noticing discrepancies in published works when attempting to replicate the searches. Clearly, an expert user of the CSD would be aware of these details but an example for the novice can be very useful both for teaching and as a cautionary tale.

Introduction

One of the techniques used to obtain accurate internal coordinate values of chemical entities involves the interrogation of experimentally determined small molecule structure databases such as the Cambridge Structure Database (CSD, Groom *et al.*, 2016) and the Crystallography Open Database (COD, Gražulis *et al.*, 2009). The former has several powerful interfaces including Conquest (Bruno *et al.*, 2002), a structure based search tool.

Notwithstanding the ample documentation, using these tools has a learning curve that can be challenging. Human nature also plays a role. Many are loathed to actually read the documentation preferring to jump into using the program. Conquest is a flexible and intuitive interface that searches the CSD for matches to a structure fragment that can be drawn in a window. Seems simple enough but there are pitfalls that can easily shallow the unsuspecting.

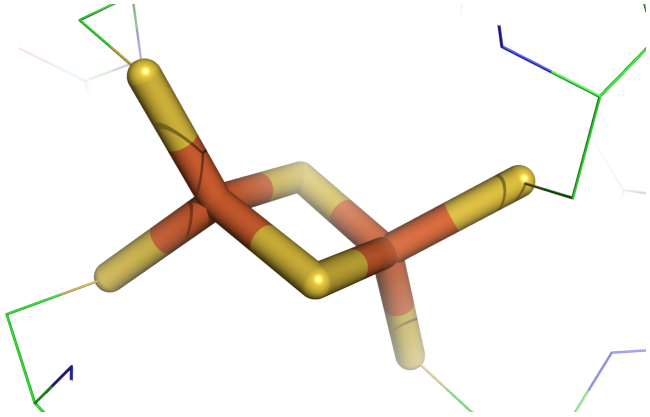
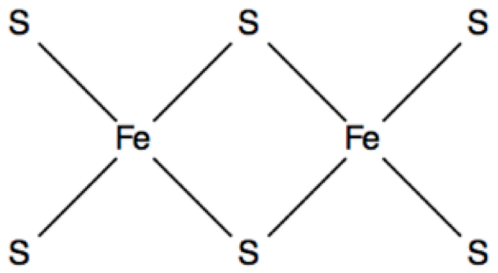


Figure 1: The Fe_2S_2 cluster (FES) from PDB entry 3wcq. The coordinated sulfur atoms from four cysteine amino acids are included in the "sticks" representation.

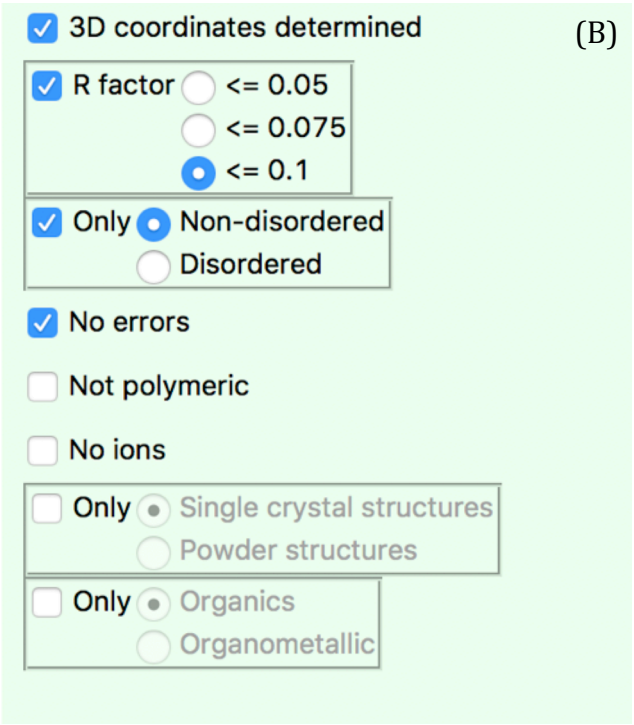
Bond number specification

The iron-sulfur cluster entry in the Chemical Component Library (CCL, Westbrook *et al.*, 2015) designated FES is a rhomboidal Fe_2S_2 entity (see figure 1) that has inaccurate geometry information in both the CCL and Monomer Library (Vagin *et al.*, 2004). This statement is based on a recent study (Moriarty & Adams, 2019) of another iron-sulfur cluster, SF4, that had similar issues. Another indicator is the high-resolution FES structure from the Protein Data Bank (PDB, Burley *et al.*, 2019) structure 3wcq shown in figure 1. This PDB entry has very different geometry values for the FES compared to the CCL and Monomer Library. Note that FES

(A)



(B)



(C)

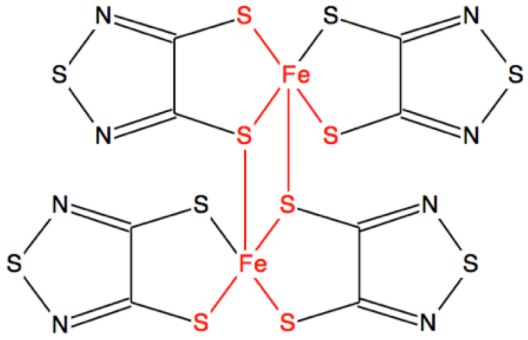


Figure 2: (A) Simple Conquest search fragment for the Fe_2S_2 cluster FES. Image taken from Conquest Draw window. (B) Conquest filter settings. (C) Example of unreasonable CSD entity with matching atoms highlighted in red.

coordinates with four cysteine sulfur atoms (included in the "sticks" representation of PyMOL (DeLano, 2002)) – two to each iron atom to form a tetra-coordinated centre. Note also that the two sulfur atoms in FES are only coordinated to the iron atoms in the FES.

Using the Conquest chemical fragment interface, a simple search can be constructed as shown in figure 2a. Searching with the filters in figure 2b results in 229 results. Verification of the results quickly shows that this search was deficient. The first result, AFAVOS, is shown in figure 2c. Focusing on the red atoms and bonds it is clear that the iron atoms are penta-coordinated and the sulfur atoms are bonded to atoms external to the group. The reason is that the number of bonds that an atom in the search fragment can be bonded to is "unspecified" – meaning any number is accepted as a hit.

One option in Conquest is to specify the exact number of bonded atoms for each atom. Once done, the search structure has T*n* where *n* is the number of bonded atoms associated with each atom as shown in figure 3. This structure returns 18 hits. Stepping though the results shows that all 18 are much closer to the FES entity topology.

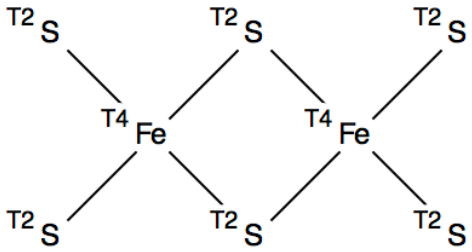


Figure 3: Conquest search fragment for the Fe_2S_2 cluster FES with bond numbers set.

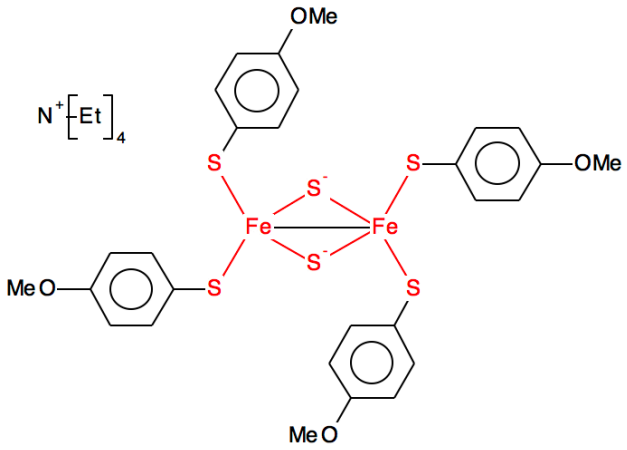


Figure 4: Search result from search fragment in figure 2A with an additional bond between the two iron atoms.

One nuance arises when considering the entry BORTOT from the first search shown in figure 4. The inclusion of the bond between the two iron atoms would appear to be an annotation of the depositor. Removing the number of bonded atoms from the iron atoms (see figure 5a) returns 24 hits. One of the hits is SIWYOM shown in figure 5b. This is clearly not an instance that can provide geometry details of FES so the stricter search is a better choice.

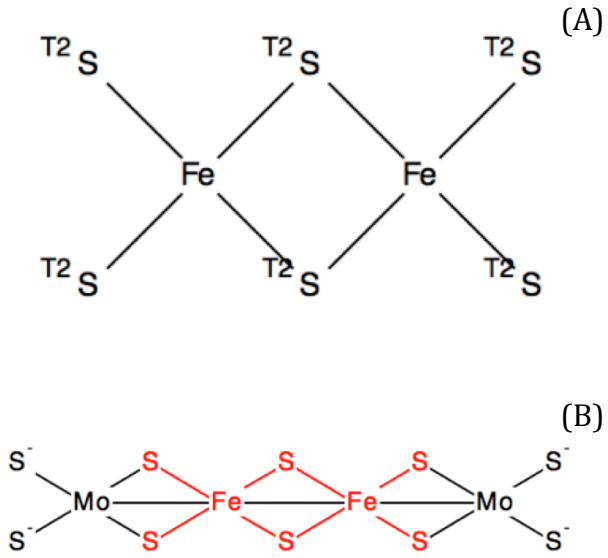


Figure 5: (A) Conquest search fragment for the Fe_2S_2 cluster FES with bond numbers set for non-iron atoms. (B) Unreasonable hit resulting from search in panel (A).

Conclusions

Always verify that the results from a structure search are reasonable. The first simple search attempt in this example resulted in many unreasonable hits for FES. Using the "number of bonded atoms" option is a powerful tool for filtering results.

References

Bruno, I. J., Cole, J. C., Edgington, P. R., Kessler, M., Macrae, C. F., McCabe, P., Pearson, J. & Taylor, R. (2002). *Acta Crystallogr. B*. **58**, 389–397.

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Costanzo, L. D., Christie, C., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranovic, V., Guzenko, D., Hudson, B. P., Liang, Y., Lowe, R., Peisach, E., Periskova, I., Randle, C., Rose, A., Sekharan, M., Shao, C., Tao, Y.-P., Valasatava, Y., Voigt, M., Westbrook, J., Young, J., Zardecki, C., Zhuravleva, M., Kurisu, G., Nakamura, H., Kengaku, Y., Cho, H., Sato, J., Kim, J. Y., Ikegawa, Y., Nakagawa, A., Yamashita, R., Kudou, T., Bekker, G.-J., Suzuki, H., Iwata, T., Yokochi, M., Kobayashi, N., Fujiwara, T., Velankar, S., Kleywegt, G. J., Anyango, S., Armstrong, D. R., Berrisford, J. M., Conroy, M. J., Dana, J. M., Deshpande, M., Gane, P., Gáborová, R., Gupta, D., Gutmanas, A., Koča, J., Mak, L., Mir, S., Mukhopadhyay, A., Nadzirin, N., Nair, S., Patwardhan, A., Paysan-Lafosse, T., Pravda, L., Salih, O., Sehnal, D., Varadi, M., Vařeková, R., Markley, J. L., Hoch, J. C., Romero, P. R., Baskaran, K., Maziuk, D., Ulrich, E. L., Wedell, J. R., Yao, H., Livny, M. & Ioannidis, Y. E. (2019). *Nucleic Acids Res.* **47**, D520–D528.

DeLano, W. L. (2002). PyMOL 0.99.

Gražulis, S., Chateigner, D., Downs, R. T., Yokochi, A. F. T., Quirós, M., Lutterotti, L., Manakova, E., Butkus, J., Moeck, P. & Le Bail, A. (2009). *J. Appl. Crystallogr.* **42**, 726–729.

Groom, C. R., Bruno, I. J., Lightfoot, M. P. & Ward, S. C. (2016). *Acta Crystallogr. Sect. B Struct. Sci. Cryst. Eng. Mater.* **72**, 171–179.

Moriarty, N. W. & Adams, P. D. (2019). *Acta Crystallogr. Sect. Struct. Biol.* **75**, 16–20.

Vagin, A. A., Steiner, R. A., Lebedev, A. A., Potterton, L., McNicholas, S., Long, F. & Murshudov, G. N. (2004). *Acta Crystallogr. D Biol. Crystallogr.* **60**, 2184–2195.

Westbrook, J. D., Shao, C., Feng, Z., Zhuravleva, M., Velankar, S. & Young, J. (2015). *Bioinformatics*. **31**, 1274–1278.

Ensemble refinement produces consistent R-free values but smaller ensemble sizes than previously reported

Stephanie A. Wankowicz[a,b] and James S. Fraser[a,b,c,d]

a - Biophysics Graduate Program, University of California San Francisco, San Francisco, CA, USA.
b - Department of Bioengineering and Therapeutic Sciences, University of California San Francisco, San Francisco, CA, USA.
c - Quantitative Biosciences Institute, University of California San Francisco, San Francisco, CA, USA.
d - Molecular Biophysics and Integrated Bioimaging Division, Lawrence Berkeley National Laboratory, Berkeley, CA
Correspondence email: jfraser@fraserlab.com

Introduction

Ensemble refinement combines molecular dynamics (MD) simulations with crystallographic data to provide a model of atomic fluctuations that are present in the crystal lattice. As implemented in *phenix.ensemble_refinement*, MD simulations are performed where the model is restrained by a time-averaged X-ray restraint (Burnley et al. 2012). Because the agreement with observed structure factors is calculated by averaging of several recent snapshots of the MD simulation, ensemble refinement differs significantly from traditional refinement where a single structure is used to calculate the agreement. To attempt to control for crystalline disorder, a Translation/Libration/Screw (TLS) model model is fitted prior to the simulation, leaving the simulation to fit the residual difference density. After the simulation is run, a procedure reduces the ensemble size down from all snapshots acquired during the period to a minimal set that will reproduce the R-free within a tolerated value. In the original paper describing *phenix.ensemble_refinement*, this yielded 39-600 ensemble members in the 20 PDB depositions that were subjected to refinement. The structural diversity across these ensemble members is a representation of the residual conformational heterogeneity after accounting for the disorder modeled by the TLS model.

We set out to run ensemble refinement on a large number of publicly available X-ray crystallography structures. Although some parameter names and default values had apparently changed since the original paper, the online documentation provided a guide to reasonable values (Phenix documentation: ensemble_refinement.html). For our analysis, all structures had a resolution between 1-2.5 Angstroms. Using Phenix version 1.15, we pursued the following workflow (code is available on github[1]).

1. Download existing model and structure factor files
2. Run *phenix.ready_set*
3. Re-refinement of model using *phenix.refine*
4. Ensemble refinement over a grid search of parameters
5. Selection of best model based on R_{free}

All input parameters for our analysis are available (https://ucsf.app.box.com/folder/95195345802).

Non-default inputs

- `wxray_coupled_tbath_offset: grid search of 2.5, 5, 10`

Errors

About 10% of the structures failed during refinement. There were numerous reasons for these failures including, a lack of appropriate

[1]https://github.com/stephaniewanko/Fraser_Lab/tree/master/phenix_pipeline

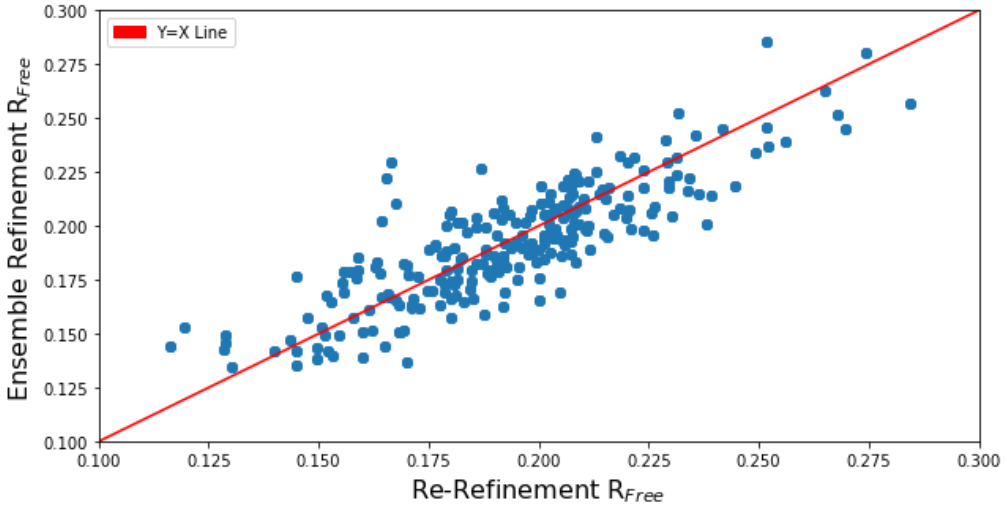


Figure 1. *R_{free} values from re-refinement and ensemble refinement are correlated (R^2=0.91). In 418 (57.6%) structures, the ensemble refinement R_{free} value was lower than the refinement R_{free} value. In 307 (42.3%) structures, the ensemble refinement R_{free} value was higher than the refinement R_{free} value.*

intensities or amplitude information, poor maps, and issues with ligands.

Conclusion

There were two major differences between our analysis and the original Burnley 2012 paper (Burnley et al. 2012). First, in the Burnley paper all 20 structures had reduced R_{free} values when subjected to ensemble refinement. In our study, overall, ensemble refinement R_{free} was comparable to re-refinement R_{free}, with 57.6% of structures having an improved R_{free} with ensemble refinement compared to traditional refinement, as shown in figure 1. This may be due to non-optimal parameter selection or insufficient model preparation. Second, it was unclear why we were getting such smaller ensemble size compared to the 2012 paper. We were expecting many ensemble sizes to be greater than 100; however, all of our structures returned ensembles <100, as demonstrated in figure 2. Although the ensembles obviously contain more diversity than single structures, we were curious as to the underlying cause of the greatly reduced ensemble size. To further investigate, we tested our ensemble refinement pipeline on the 20 PDB models originally analyzed in Burnley 2012 paper.

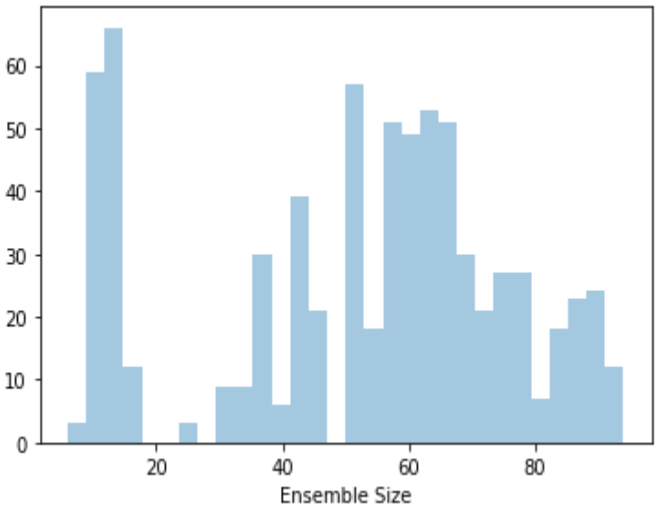


Figure 2. *Most structures have smaller ensemble sizes (the number of models in the ensemble output) than we expected based on the results in Burnley 2012.*

Recreating Burnley 2012 Paper

To recreate the results from the Burnley 2012 paper, we followed the same pipeline outlined above. Of note, while we automatically re-refined the models coming from the PDB, we did not perform any manual refinement, which

Table 1. *Input R values from Burnley 2012 compared to our input to our recreation.*

PDB	Resol.	Original Ensemble Size	R_{work} Burnley 2012	R_{free} Burnley 2012	R_{work} Recreation	R_{free} Recreation	Recreation Lowest R_{free} Ensemble Size
1kzk	1.1	600	0.125	0.153	0.155	0.179	100
3k0m	1.3	250	0.104	0.129	0.127	0.144	167
3k0n	1.4	209	0.115	0.133	0.117	0.143	167
2pc0	1.4	250	0.145	0.188	0.231	0.252	125
1uoy	1.5	167	0.104	0.137	0.136	0.165	125
3ca7	1.5	40	0.149	0.184	0.237	0.292	56
2r8q	1.5	200	0.132	0.162	0.164	0.188	125
3ql0	1.6	70	0.204	0.254	0.217	0.256	50
1x6p	1.6	400	0.121	0.149	0.141	0.163	134
1f2f	1.7	143	0.128	0.168	0.170	0.210	84
3ql3	1.8	80	0.160	0.208	0.171	0.207	56
1ytt	1.8	84	0.139	0.174	0.179	0.206	63
3gwh	2.0	39	0.160	0.200	0.198	0.230	67
1bv1	2.0	78	0.149	0.182	0.188	0.240	84
1iep	2.1	200	0.183	0.238	0.207	0.256	63
2xfa	2.1	100	0.171	0.217	0.226	0.261	60
3odu	2.5	50	0.208	0.269	0.247	0.297	32
1m52	2.6	50	0.161	0.211	0.198	0.240	32
3cm8	2.9	67	0.194	0.235	0.231	0.264	39
3rze	3.1	72	0.210	0.280	0.250	0.289	32

left us with input structures with slightly higher R_{free}/R_{work} compared to the Burnley 2012 paper (table 1). We extended our grid search to include three parameters suggested by the Phenix documentation (pTLS, wxray_coupled_tbath_offset, tx).

- pTLS defines the fraction of atoms included in the TLS fitting procedure. This is intended to model static crystalline lattice disorder and varying this parameter results in movement being absorbed by the TLS B-factors rather than by atomic fluctuations.
- wxray_coupled_tbath_offset controls the X-ray weight. This helps ensures that the simulation runs at the target temperature.
- tx dictates the structure factor memory relaxation time. This governs the time period for which a particular conformation retains its influence. The higher the number, the more a particular conformation affects the average.

Additionally, we added harmonic restraints for all ligands in each structure. Of note, while Burnley 2012 paper reported only one ensemble structure per PDB, we had 36 ensemble structures (corresponding to a 3 x 3 x 4 grid search of the parameters pTLS, tx, wxray_coupled_tbath_offset) and choose one select ensemble structure based on the criteria of lowest R_{free}. This test was run on Phenix version dev-3584 (a mid 2019 version).

Non-default inputs

- `wxray_coupled_tbath_offset: grid search of 2.5, 5, 10`
- `pTLS: grid search of 0.6,0.8,1.0`
- `tx: 0.5, 0.8, 1, 1.5`

Outputs

As shown in table 1, in almost all cases, the ensemble sizes were lower than what was found in the Burnley 2012 paper. Figure 3 illustrates that we found that R_{free} correlated with ensemble size (R^2 =-0.61). Similarly, resolution was slightly correlated with ensemble size (R^2=-0.48). Overall, the recreated R_{free} were highly correlated with the R_{free} from the Burnley 2012 paper (R^2=0.892). We could not identify any pattern between the parameter values correlated with R_{free} and the optimal parameter value as judged by R_{free} was idiosyncratic for each structure.

As we wanted to use ensemble refinement to assess dynamics, we want to see if different parameter values (pTLS, wxray_coupled_tbath_offset, tx) change the RMSF. We examined all structures, but focus our analysis below on C-ABL kinase domain in complex with STI-571 (PDB: 1IEP).

While the RMSF values of C-ABL kinase domain in complex with STI-571 (PDB: 1IEP) were highly correlated (>0.8) across all parameter values, highlighted in figure 4, there were only some notable deviations in magnitude for the pTLS parameter values, demonstrated in figure 5.

Because of the lack of correlation between the parameters and R_{free} values, and the relative consistency of the RMSF calculations, we evaluated each PDB independently and chose parameters that yielded the lowest R_{free}. At least one of the 20 PDBs had an optimal ensemble using each of the wxray_coupled_tbath_offset and pTLS parameter values. For the tx parameter only 3 out of the 5 values were used in optimal ensembles (0.8, 1.0, 1.5).

Conclusions

Overall, we were still getting much smaller ensemble sizes compared to the Burnley 2012 paper. However, our R_{free} values correlated very well with the R_{free} values from the paper giving

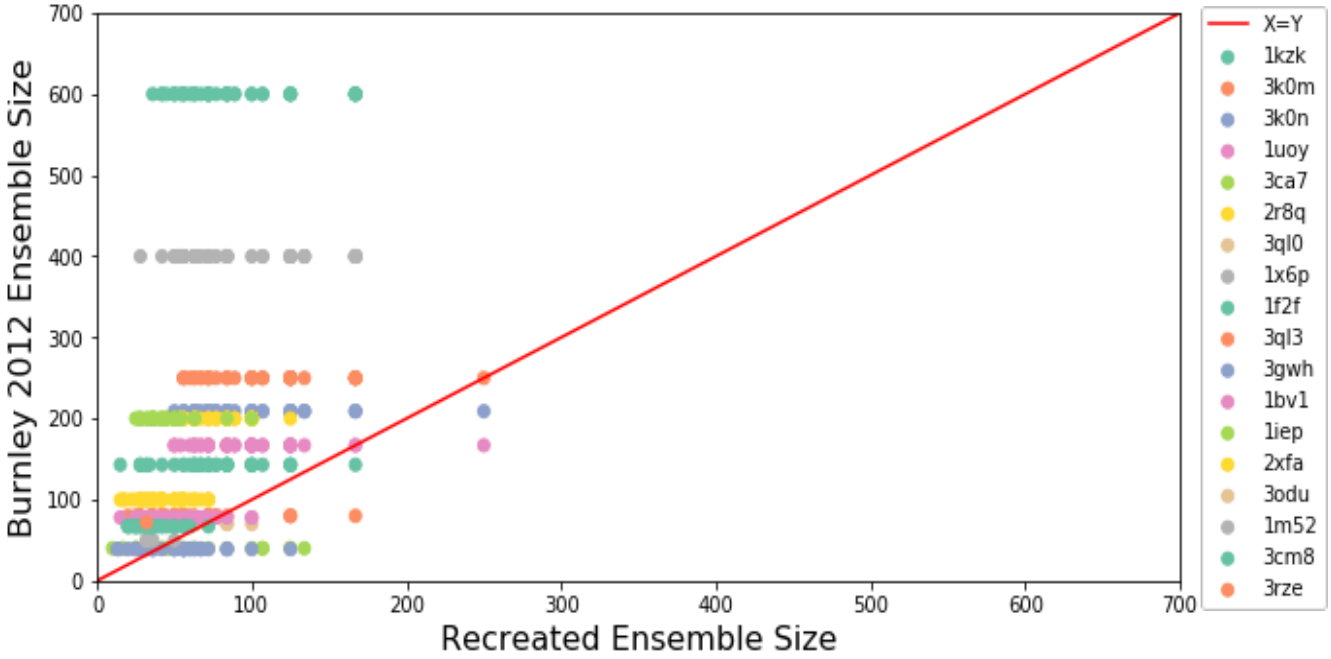


Figure 3. *Burnley 2012 paper ensemble size compared to our recreated ensemble size. In almost all cases, the ensemble sizes were lower than what was found in the Burnley 2012 paper.*

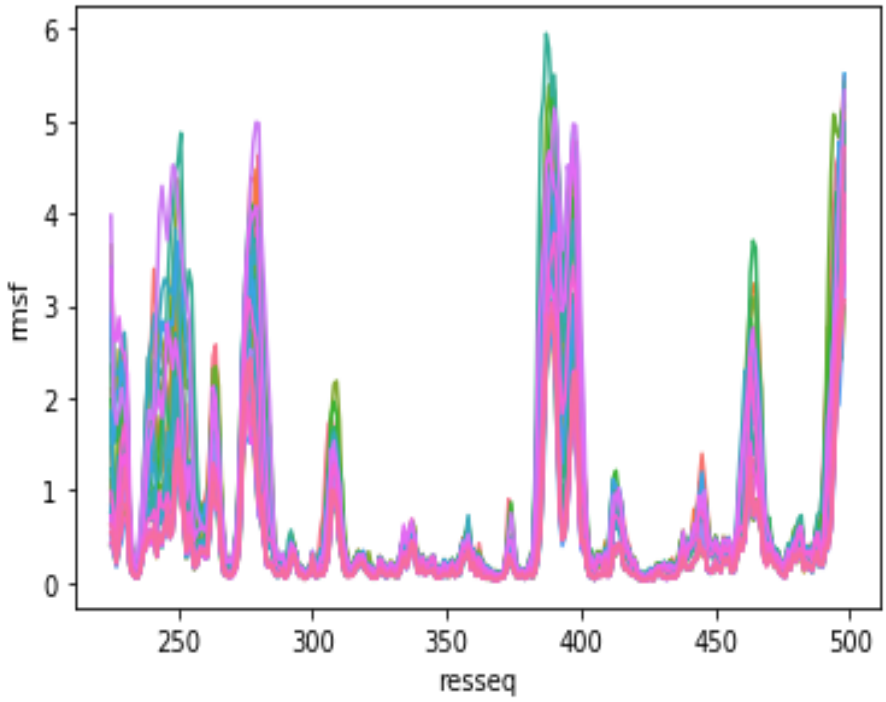


Figure 4. *RMSF of C-ABL kinase domain in complex with STI-571(PDB: 1IEP) across all 45 parameter values.*

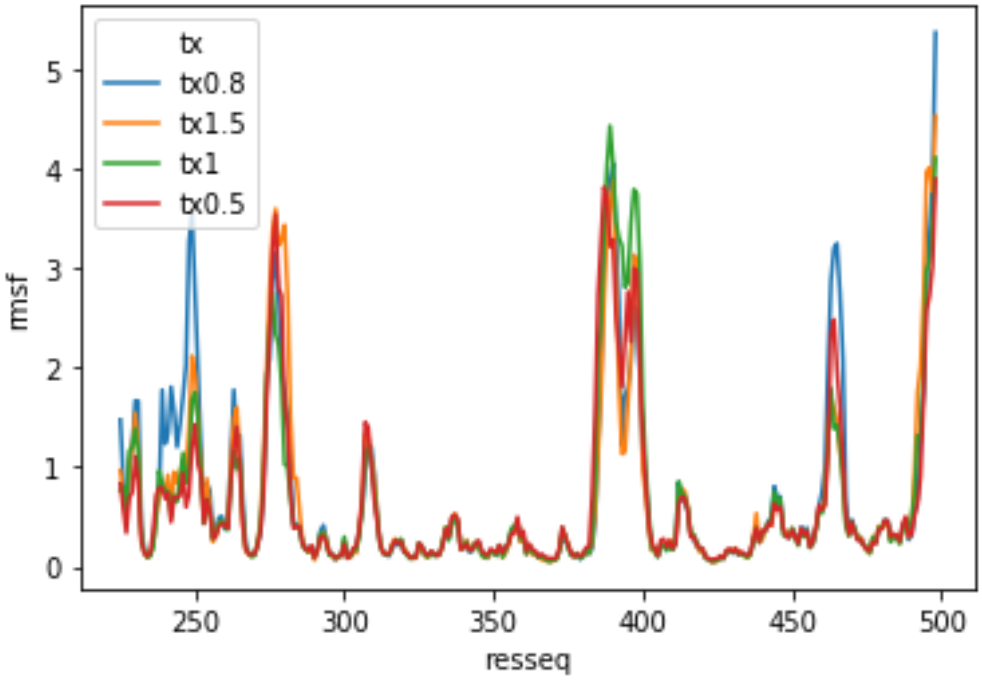


Figure 5. *RMSF of C-ABL kinase domain in complex with STI-571(PDB: 1IEP) across all tx parameter values.*

us confidence in the underlying procedure. The wxray_coupled_tbath_offset, pTLS, or tx parameter values were not correlated with R_{free}, R_{work}, or the ensemble size. In terms of RMSF changes, the only parameter that produced a major difference was pTLS, as expected. pTLS determines the percentage of atoms included in

the TLS model, which predicts the local positional displacement of atoms in a crystal structure with the underlying assumption that the atoms included are members of a rigid body. In our results, lower pTLS values (fewer atoms included in the pTLS model) have higher RMSF on average. It is unclear to us if choosing a model based on the best R_{free} will result in accurate results for protein conformational heterogeneity, especially when comparing two protein structures with different pTLS values.

Investigating the ensemble size difference

To try to resolve the discrepancy in the ensemble sizes from the original 2012 paper to our recreation of their results, we used the optimal parameter values from the test above for each PDB and tested four other keyword changes that we predicted might give us results closer to the Burnley 2012 paper.

1) Using the Phenix version released most closely to the Burnley 2012 paper (version 1.8.2, the first release to contain the *phenix.ensemble_refinement* command).
2) Removing the use of the conformation dependent restraint library.
3) Re-setting the ensemble R_{free} tolerance parameter to 0.001
4) Re-setting the ensemble reduction feature to false

Testing Phenix version 1.8.2

The Burnley 2012 paper was run using a different Phenix version than we used with our recreation. We ran ensemble refinement on Phenix version 1.8.2, which corresponds to the public release of the method after the Burnley 2012 paper.

Non-default inputs (Phenix version 1.8.2)

- `wxray_coupled_tbath_offset, pTLS, tx parameter values corresponding to the optimal` R_{free} `for each individual PDB from the previous tests.`

Errors

Only five out of the 20 structures ran ensemble refinement successfully. There were multiple reasons for failures. These included a pTLS error with chain breaks and errors reading in parameters fed into ensemble refinement.

Conclusions

Many structures failed to run ensemble refinement. However, for the five structures that finished, the ensemble sizes were still smaller than expected based on the Burnley 2012 paper and were highly correlated with our previous recreation, as seen in figure 6 and 7. Figure 8 demonstrated that we continue to observe a good correlation between the original and updated R_{free} values (R^2=0.95).

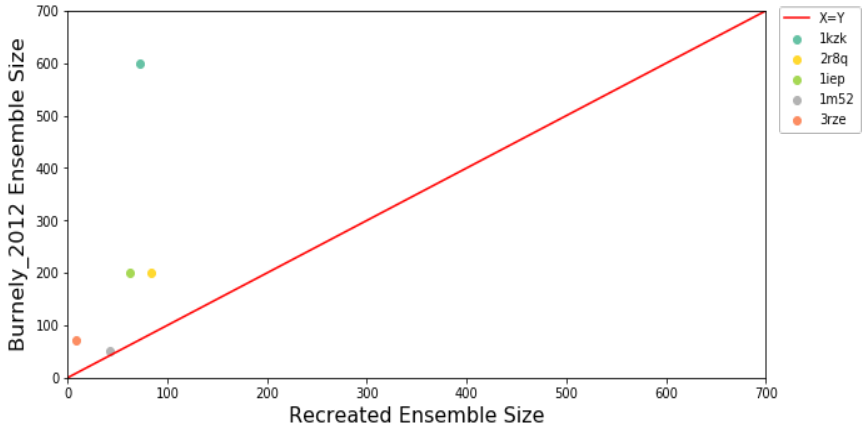


Figure 6. *Recreated ensemble sizes are smaller compared to the ensemble sizes in Burnley 2012 (R^2=0.57).*

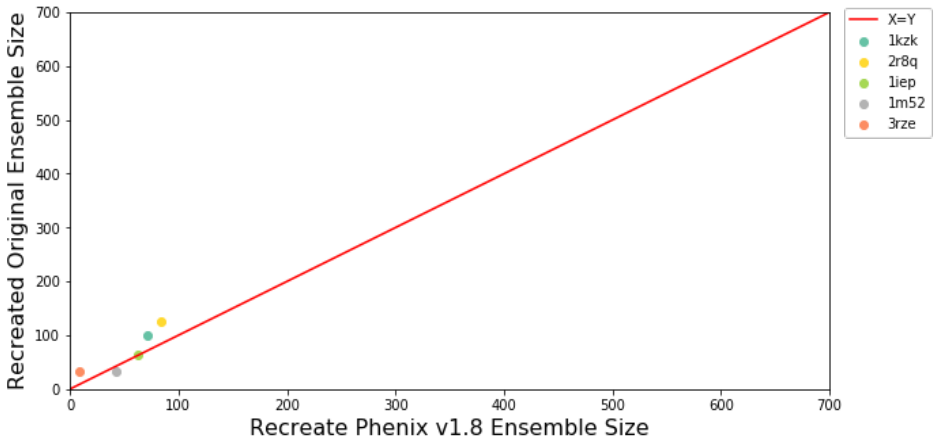


Figure 7. *Recreated ensemble sizes with Phenix version are similar to the initially recreated ensemble sizes (R^2=0.88).*

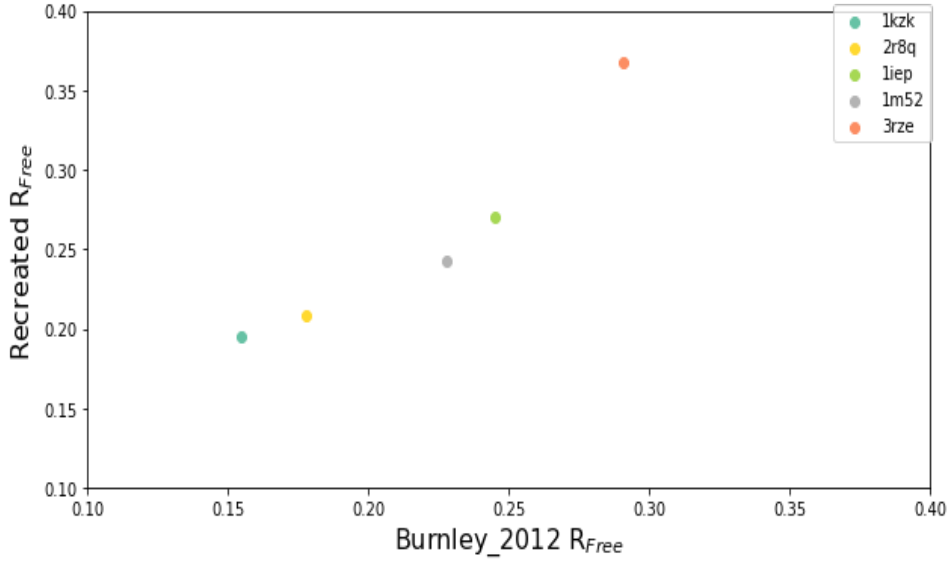


Figure 8. *Recreated R_{free} were highly correlated with the* Burnley 2012 *paper (R^2=0.95).*

Reverting R_{free} Tolerance to 0.001

The last step of ensemble refinement takes all of the snapshots saved from the MD simulation and selects the lowest number of models that together have an R_{free} within the percentage of the full ensemble R_{free}. This percentage is defined by the R_{free} tolerance parameter. The current version of Phenix (1.16), defaults this parameter to 0.0025 but in the Burnely 2012 paper, it was set to 0.001. Therefore, we tested if we could increase the ensemble size by changing this parameter back to what was used in the paper using Phenix version 1.16.

Non-default inputs (Phenix version 1.16)

- `Wxray_coupled_tbath_offset, pTLS, tx parameter values corresponding to the best` R_{free} `for each individual structure.`
- `ensemble_reduction_rfree_tolerance = 0.001`

Conclusions

Reducing the R_{free} tolerance parameter back to where it was initially set did increase our recreated ensemble size (median increase: 26 models), see figure 9. However, for many of these structures the

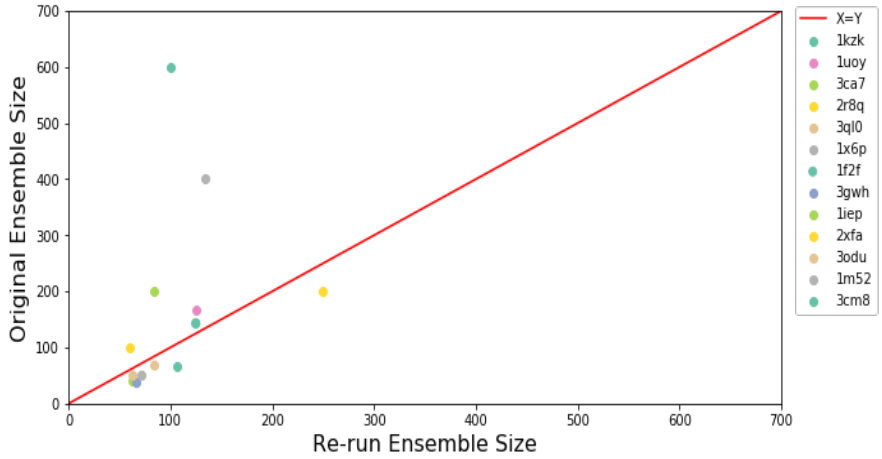


Figure 9. Burnley 2012 paper *ensemble size compared to our recreated ensemble size with an R_{free} Tolerance of 0.001 (R^2=0.307).*

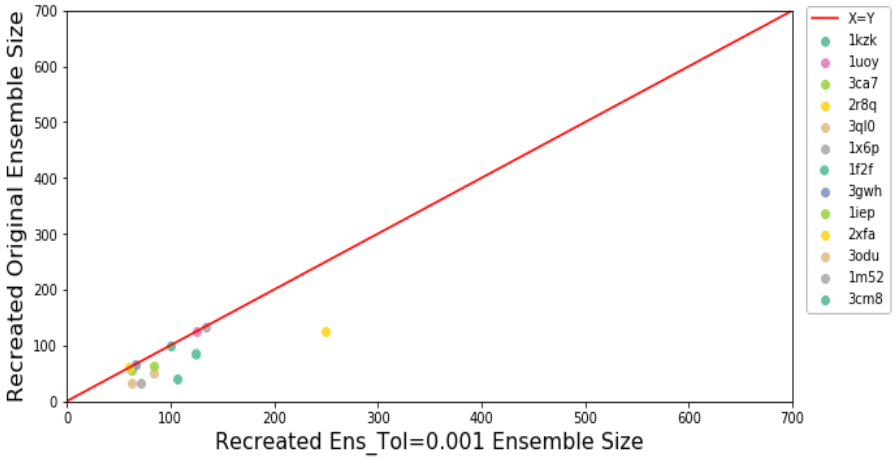


Figure 10. *Recreated ensemble sizes with ensemble_rfree_tolerence parameter =0.001 are mostly larger than the initially recreated ensemble sizes(R^2=0.71).*

number of models were still far below what was observed in the Burnley 2012 paper, as seen in figure 10. Figure 11 shows that the R_{free} correlation was still observed between the recreation on the Burnley 2012 paper.

Removing Conformational Dependent Library (CDL)

In the Burnley 2012 paper, the default restraints were Engh and Huber, but more recent versions of phenix use the conformation dependent library (CDL). One hypothesis is that the older restraints would bias ensembles to have energetically reasonable angles and bond lengths compared to the modern CDL restraints, leading the ensemble sizes to decrease under CDL restraints. Therefore, we set the cdl restraint library to false. Of note there are three other library (omega_cdl, rdl, and hpdl) that are also available as parameters but are set as false as the default.

Non-default inputs (Phenix version 1.16)

- `Wxray_coupled_tbath_offset, pTLS, tx parameter values corresponding to the best` R_{free} `for each individual structure.`
- `restraints_library_cdl = False`

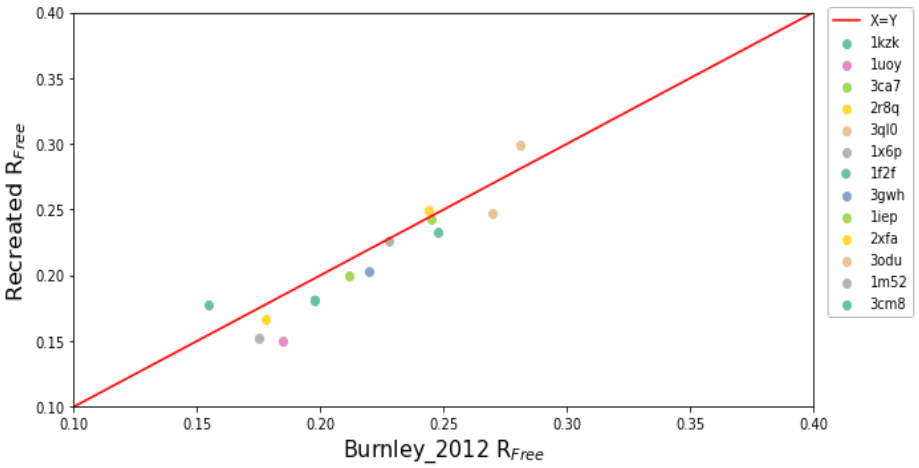


***Figure 11**. Recreated R_{free} with an R_{free} tolerance of 0.001 were highly correlated (R^2 =0.9) with the R_{free} Burnley 2012 paper*

Conclusions

By turning the CDL restraints off, we did not observe an increase in ensemble sizes, see figure 12. For one structure (PDB:3GWH), the size of the ensemble did increase, see figure 13. We suspect this is due to the input model having a high number of geometry outliers. By turning off CDL, we may have further increased the geometry problem, resulting in a larger ensemble size. There was still a high correlation between the original and recreated R_{free} (R^2=0.96) as shown in figure 14.

Testing Ensemble Reduction

When the ensemble reduction parameter is turned to false, ensemble refinement outputs all of the models in the ensemble rather than selecting down a smaller number of models to match the R_{free} tolerance value. By turning this value off, we hypothesized that the size of the ensembles would all be 500, since that is the number of models created in ensemble refinement (based on default parameters).

Non-default inputs

```
wxray_coupled_tbath_offset, pTLS, tx
parameter values corresponding to
the best Rfree for each individual
structure.
ensemble_reduction = False
```

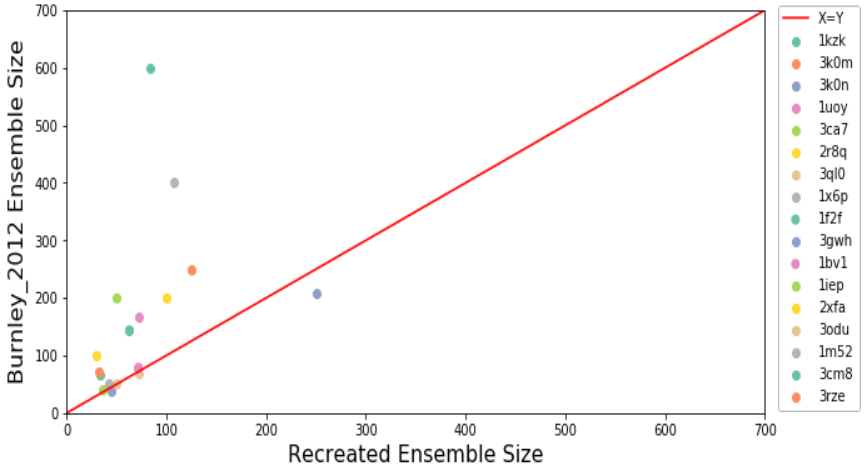


***Figure 12.** Recreated ensemble sizes are smaller compared to the ensemble sizes in Burnley 2012 (R^2=0.4).*

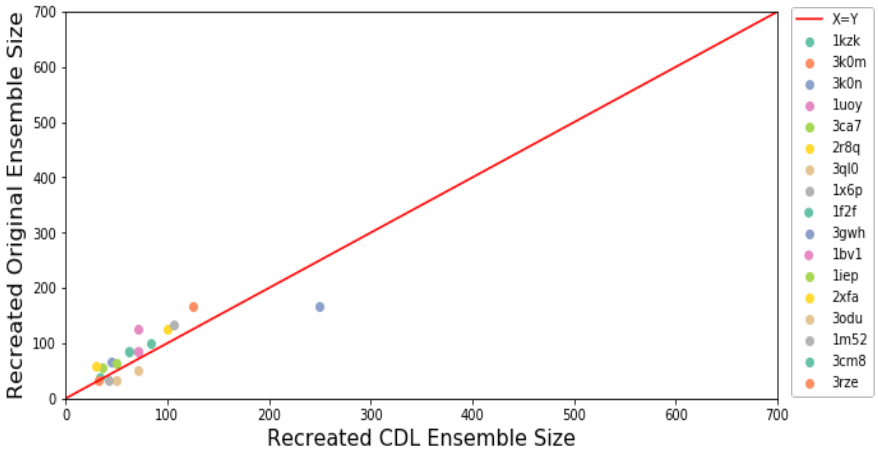


Figure 13. *Recreated ensemble sizes with CDL parameter=False are similar to the initially recreated ensemble sizes(R^2=0.82).*

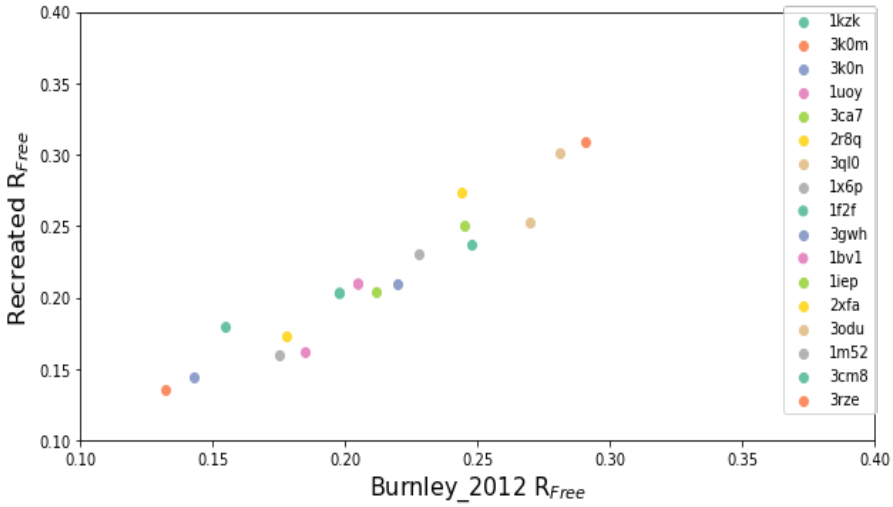


Figure 14. *Recreated R_{free} were highly correlated with the* Burnley 2012 *paper (R^2=0.96).*

Conclusion

While turning off the ensemble reduction parameter did increase the ensemble size, we only observed two ensemble size. There was not a trend of the ensemble size observed in Burnley 2012 and our recreated ensemble size (R^2=-0.03), see figure 15. There was still a high correlation between the original and recreated R_{free} (R^2=0.95) as shown in figure 16.

Using specific TLS selections from the 2012 paper

After discussing our results with the original authors, we realized that the authors used specific TLS selections and other values for the pTLS, tx, and wxray_coupled_tbath_offset parameters. We then set the following parameters from their log files in ensemble refinement Phenix version 1.16. Of note, there were additional parameters that were different between the two versions that we were not able to change.

Parameters changed:

- `pTLS`
- `tx`
- `wxray_coupled_tbath_offset`

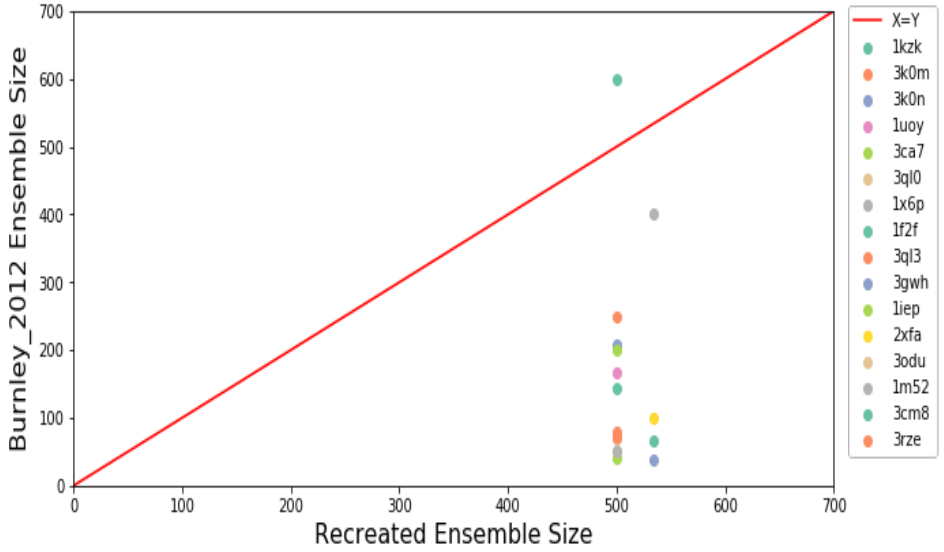


Figure 15. *Recreated ensemble sizes are larger compared to the ensemble sizes in Burnley 2012 (R^2=-0.03).*

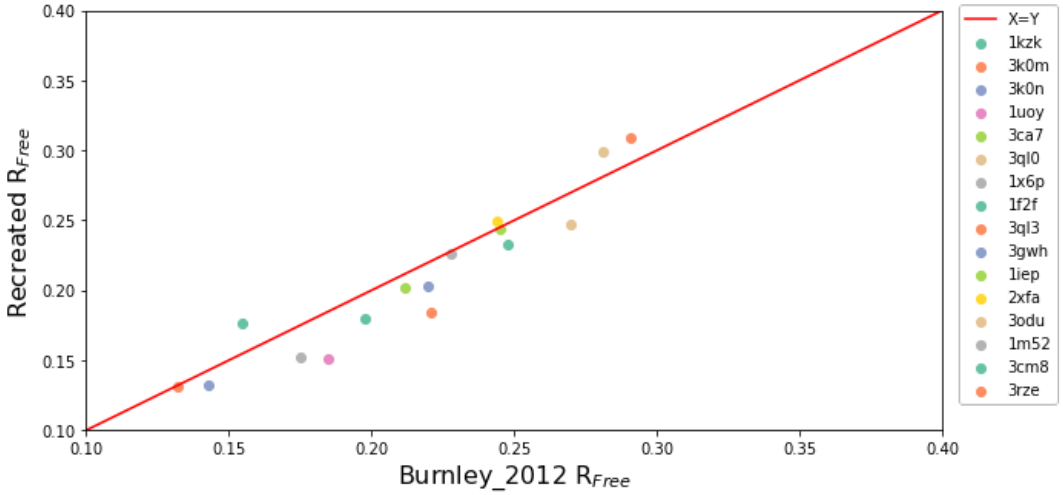


Figure 16. *Recreated R_{free} were highly correlated with the* Burnley 2012 *paper (R^2=0.95).*

- `pTLS selections`
- `harmonic restraints`

Conclusion

Altering these parameters more closely resembled our original recreation both in ensemble size and Rfree and does not underly the larger ensembles in the 2012 paper, as shown in figure 17.

Overall conclusions

While we were not fully able to recreate the ensembles in the Burnley 2012 analysis, we are confident that ensemble refinement is stable and outputs interesting representations of conformational heterogeneity. Metrics that can be used to assess those representations, such as R values or RMSF are not greatly affected by the changes to the method. We would advise future users of the ensemble refinement methods that you may observe lower number of models in each ensemble compared to the Burnley 2012 paper. The only parameter change that seemed to increase the ensemble size was changing

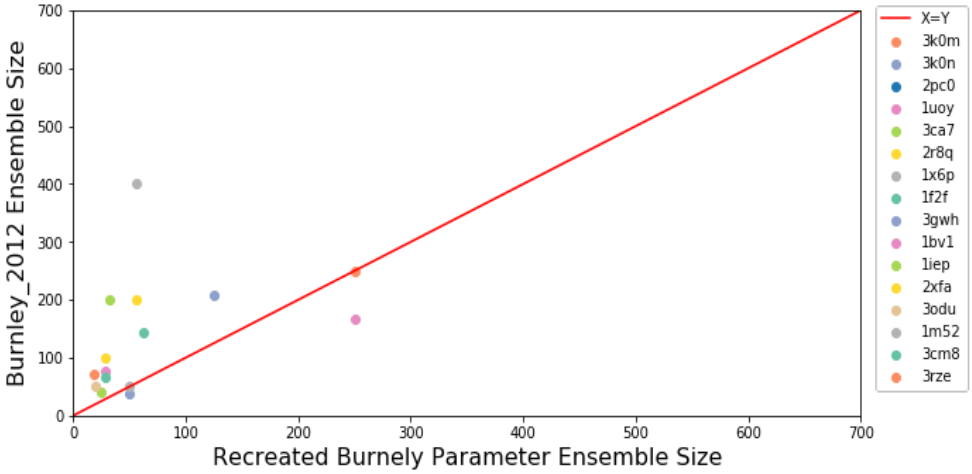


Figure 17. *Recreated ensemble sizes with the Burnley 2012 parameters not correlated to the Burnley 2012 parameters (R^2=0.41).*

ensemble_reduction to false but these values did not correlate with the ensemble sizes observed in Burnley 2012 paper. As advised on the phenix website, we suggest that users perform a grid search over the parameters of tx, wxray_coupled_tbath_offset, and pTLS. Additionally, adding harmonic restraints on all non-water HETATMS is important. All input parameters for our analysis are available (https://ucsf.app.box.com/folder/95195345802) . Moving forward, we would like to encourage publishing the exact parameters used in any refinement procedure for reproducibility.

References

Burnley, B. Tom, Pavel V. Afonine, Paul D. Adams, and Piet Gros. 2012. "Modelling Dynamics in Protein Crystal Structures by Ensemble Refinement." *eLife*.

dtmin - a Domain Tunable Python Minimizer

Duncan H. Stockwell[a*], Randy J. Read[a] and Airlie J. McCoy[a]
[a]*Department of Haematology, Cambridge Institute for Medical Research, Clinical School of Medicine, University of Cambridge, CB2 0XY, UK*
*Correspondence email: dhs37@cam.ac.uk

Introduction

Phaser's minimizer, written in C++, has been developed to meet the needs of optimizing likelihood target functions for molecular replacement (MR) and single-wavelength anomalous diffraction (SAD) phasing. Optimizing likelihood targets, by minimizing the minus-log-likelihood gain (see (McCoy, 2004) for review), is intrinsic to *Phaser's* anisotropy correction, translational non-crystallographic symmetry epsilon factor correction, rigid body refinement of posed ensembles, gyre refinement of oriented ensembles, single atom molecular replacement, SAD substructure content analysis, SAD substructure refinement, and log-likelihood pruning (see (McCoy *et al.*, 2009) for documentation). The minimization takes initial parameter value estimates (e.g. approximate global minima obtained from grid search methods) and improves them to give optimal estimates of structure factor phases and map coefficients throughout the MR and SAD phasing pathways. *Phaser* is currently being reconfigured as *phasertng*, to support directed acyclic graph data structures; and *phasertng* is central to our development of *phaser.voyager*, which will leverage the directed acyclic graph data structure to allow dynamic decision making in phasing strategies. As part of this project, we have developed a python minimizer library, *dtmin*, that has the same advanced functionality as the *phaser* minimizer and made it available in the *cctbx* library. The advanced functionality includes the ability to change the subset of refined parameters each refinement 'macrocycle' (see below), bounding parameters, logarithmic reparameterization, outlier rejection, use of large shift estimation, and a mechanism for debugging derivative calculations ('study_parameters'). This article documents how to use *dtmin* and briefly compares it to the *cctbx* minimizer *scitbx.lbfgs*.

Terms

Target Function: the function to be minimized. There are various names for this in the literature and in different fields, such as 'cost' function, or 'loss' function. In crystallography this is typically of the form:

$$\text{Target} = -\sum_i \log\left(P(x_i|\,\theta_1,\ldots,\theta_J)\right) + \sum_j k_j\left(\theta_j-\theta_{0_j}\right)^2$$

where $P(x_i|\theta_1,\ldots,\theta_J)$ is the likelihood of the ith data value given the model parameters, and $k_j\left(\theta_j-\theta_{0_j}\right)^2$ is a restraint term for the jth model parameter θ_j. The term 'restraint' is used in crystallography and molecular dynamics; in other literature it is known as a 'penalty function'.

Gradient: the array of first derivatives, with respect to the parameters, of the function to be minimized. The gradient may be calculated analytically or using finite differences; which method is faster depends critically on the function and its parameterization. The target function is often an intermediate in the calculation of the gradient.

Hessian: the matrix of second derivatives of the function to be minimized. As with the gradient function, this may be calculated analytically or with finite differences. Some minimization methods do not require all elements of the matrix to be calculated. Whether the Hessian is more computationally intensive to compute than the gradient depends on the complexity of the second derivatives and the number of elements in the Hessian that are calculated.

Bounds: range of values that a parameter is permitted to take. For example, σ_A is restricted to values between 0 and 1. Bounds need not only be the mathematically allowed values, they can be used to restrict the range of values for good convergence. For example, B-factors may be restricted to values between -20Å^2 and 500Å^2, although mathematically they may take any value. Bounds are optional for each parameter.

Reparameterization: Some parameters have more quadratic behavior if reparameterized for refinement, e.g. B-factors. Shifts are calculated in terms of the reparameterized variable. Although many reparameterizations are possible, in practice we have found that the most effective reparameterization is logarithmic. When performing logarithmic reparameterization, an 'offset' is applied to the parameter before taking the logarithm. The offset supports reparameterization of negative parameter values while different values should be tested to optimize the convergence. *dtmin* has a logarithmic reparameterization available by flagging each parameter true/false and supplying an 'offset' value in the case of true.

Outliers: data points that should be excluded from the target value calculation. This may be because they take disallowed values, will bias the refinement, or are extremely unlikely to be good estimates of the true values. Data points can also be excluded to speed up the target value calculation. Good minimization should always include a robust method for outlier rejection.

Protocol: The protocol specifies which subset of the parameters are refined during each *macrocycle* of refinement. For stable refinement, it is often useful to refine the parameters in steps, first minimizing values whose initial values are likely to be far from convergence and then adding less significant parameters in later macrocycles. Within each macrocycle, there are *microcycles* each of which consists of finding a direction and performing a line-search. See figure 1.

Large Shifts: The maximum distance that you think each parameter should reasonably be able to move in one *microcycle*. They must be specified for each parameter and are used to stabilize the minimizer by damping parameters shifts that try to move parameters more than their 'large shift' in the first step of a line search. For example, 0.1 is a large shift for a fractional coordinate shift, but negligible for an atomic B-factor; appropriate 'large shift' values for these two parameters could be 0.01 and 10. Large shift values can also be used for rough curvature estimates by taking the reciprocal of the 'large shift' squared, effectively putting different parameter types on a common scale. Different large shift values will have a significant effect on the behavior of the minimizer and should be carefully optimized.

General properties

dtmin minimizes a real valued function of *n* real parameters using an iterative line-search based method. Given some starting values for the parameters, a direction in the *n*-dimensional parameter space is chosen and a 1D minimization procedure (the 'line-search') carried out along this direction. In general, the direction chosen for a line search differs with minimization method. For example, in the method of 'steepest descent', the negative gradient of the function at the starting point is taken as the line-search direction. This method, although simple, has slow convergence (requires a great many iterations) for functions that have 'valleys' rather than 'holes'. *dtmin* has implementations of Newton's method and the BFGS (Broyden–Fletcher–Goldfarb–Shanno) method for finding line-search directions. Both Newton's method and the BFGS algorithm make use of function gradients and the Hessian. While using the Hessian typically decreases the number of iterations to convergence over methods relying solely on gradient evaluations, Hessian evaluation increases the computational cost of each iteration over purely gradient driven methods. In some cases, the computational cost may be very high. Gradient driven methods can be implemented by setting the Hessian to the identity matrix

(equivalent to the method of 'steepest descent'), or to a constant estimate of the Hessian at the minimum, such as from "large shift" values for each parameter (see below). Alternatively, the computational cost of Hessian evaluation can be reduced by only providing the diagonals of the Hessian matrix (the 'curvatures').

In *dtmin*, termination occurs when one of the following criteria is met: the gradient values are all exactly zero, meaning we have found a true minimum (this is unlikely both due to numerical imprecision and because other termination criteria will be met first); every parameter is bounded, we are at the bounds and the step direction wants to push all the parameters over their bounds; none of the parameters are shifted by a significant amount, where significance is calibrated by scales derived from the diagonals of the Hessian; the function does not decrease by a significant amount, where significance is calibrated by the current function value and the numerical precision; or the maximum number of microcycles has been reached.

Architecture

The *dtmin* is architected as two main classes. The first, the 'minimizer' class, performs the overall iteration to convergence in its 'run' method and should not be modified. The second, the user implemented 'refinement' class, is specific to your problem – this is where the implementation for your target function goes! It is derived from a single RefineBase class, which in turn is derived from four base classes (Compulsory, Optional, Logging and Auxiliary; see below). Only the functions in Compulsory must be implemented by the user for the minimizer to run.

Minimizer:

The minimizer controls the minimization strategy, shown in Figure 1. It calls functions defined in the RefineBase class. To run the minimizer, derive your implementation-specific Refine Class from RefineBase and pass it to the 'run' function of the Minimizer class, along with the protocol for refinement (which parameters to refine in each macrocycle), the maximum number of microcycles per macrocycle (typically around 50), the minimizer to use (either 'newton' or 'bfgs') and whether or not to output the study_parameters gradient and curvatures debugging information (this will be 'False' for production code).

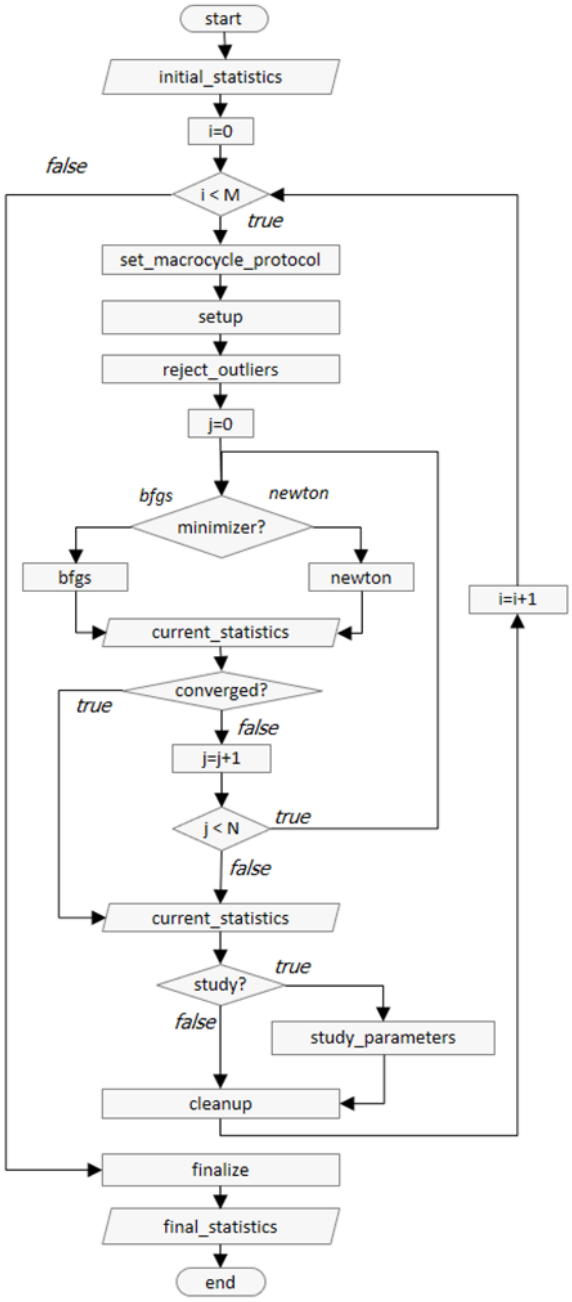


Figure 1: Flow diagram of the overall minimization strategy of the Minimizer class, where *i* is the macrocycle index and M is the number of macrocycles, *j* is the microcycle index and N the maximum number of microcycles per macrocycle.

Initial parameters: The starting values for the refinement are set in the initialization of the Refine Class. No functions are provided for initial setup.

Study parameters: The study_parameters function is used in development for checking values of the implemented gradients and curvatures against values computed using finite differences. The function outputs values obtained from the calls to target_gradient_hessian and values obtained from finite difference first derivatives, and finite difference second derivatives calculated from function values or from analytic gradients. The results can be interrogated either by inspection or using a 'mathematica' (Wolfram Research Inc., 2019) notebook available for download from phaserwiki (McCoy *et al.*, 2009). Note that the function target_gradient_hessian itself may have been implemented to return gradients and Hessians derived from finite difference methods, in which case the study_parameters protocol is superfluous.

Output: Output from the minimizer is controlled by the level of output requested. Default output from the minimizer reports the path through the minimization and can be used to optimize the path through refinement. If the functions in the Logging class are implemented (see below), intermediate statistics can be reported during the minimization.

Result: After the minimizer's run method is called, the Refine object is left in the minimized state.

RefineBase:

Your function to minimize should be defined in a class derived from RefineBase. RefineBase is, in turn, derived from four classes:

Compulsory: Compulsory functions must be implemented and will throw an exception when called by the minimizer if they are not.

Optional: Optional functions have default implementations, although overriding the defaults is recommended for optimal performance of the minimizer.

Logging: Logging functions produce no output by default and should be customized to generate output before, during and after minimization.

Auxiliary: Helper functions required for minimization, which should not be modified by the user.

See Table 1 for a summary of the functions in classes Compulsory, Optional and Logging. We have provided two example implementations, which are described below, to help guide users in how to implement the necessary functions for their problem. The function to set the protocol at the beginning of the macrocycle (set_macrocycle_parameters) is used to define the set of parameters that are refined by the macrocycle. The RefineBase member variable 'nmp', which is the number of parameters that are refined by the macrocycle, must be set by this function. If parameter selection is required by the user's refinement protocol, it will be necessary to keep an internal record of the list of parameters and the selection within the Refine object. Note that changing the set of function parameters refined between macrocycles, via set_macrocycle_protocol, will require a concomitant change in the functions that return or accept arrays depending on the set of function parameters refined between macrocycles (Table 1).

Example 1

The twisted Gaussian function was used as a development test case for *scitbx.lbfgs* and is implemented in the script *scitbx/lbfgs/dev/twisted_gaussian.py*. This script performs minimizations starting from 100 random start points, both with and without the use of curvatures to prime the Hessian approximation. As a reference implementation of the *dtmin,* we provide the script *scitbx/dtmin/twisted_gaussian.py*, which implements the same twisted Gaussian function minimizations. Although these two scripts minimize the same target function, differences in implementation will give different results. The

Table 1: Description of the functions, their arguments and return types and a summary of each function. Lists with an asterisk (*) are required to be the length of the number of refined parameters in the macrocycle (nmp) which is set in the function set_macrocycle_protocol. The array (Hessian matrix) with double asterisk (**) is required to be of length (nmp squared). Functions that return or accept arrays depending on nmp are indicated with a dagger (†).

Function	Arguments	Return type	Description	Default
Compulsory Functions				
target	None	Float	Target function (lower is better). Includes restraint terms (if any).	Raise NotImplementedError
get_macrocycle_parameters†	None	List of Float*	Get the parameters being refined this macrocycle	Raise NotImplementedError
set_macrocycle_parameters†	List of Float*	None	Set the current values of the macrocycle parameters	Raise NotImplementedError
macrocycle_large_shifts†	None	List of Float*	Array of large shift values for the parameters being refined this macrocycle	Raise NotImplementedError
set_macrocycle_protocol	List of String	None	Sets up the refine object for the current macrocycle.	Raise NotImplementedError
macrocycle_parameter_names †	None	List of String*	Names of the parameters being refined this macrocycle	Raise NotImplementedError
Optional Functions				
target_gradient†	None	Float, List of Float*	Target and gradient for the function to be minimized	Finite difference gradient
target_gradient_hessian†	None	Float, List of Float*, Array of Float**, Bool	Target, Gradient and Hessian of the parameters being refined in the current macrocycle. Also a bool to indicate whether the Hessian is diagonal or not so that the minimizer can do a simplified inverse calculation	Finite difference gradient. Hessian whose diagonals are the reciprocal of the square of the large shifts of the parameters
bounds†	None	List of 'Bounds' objects*	Bounds (minimum and/or maximum or no bounds) of each parameter being refined this macrocycle.	No bounds
reparameterize†	None	List of 'Reparams' objects*	Flags and offset (if true) for reparameterization of the parameters being refined this macrocycle	No reparameterization
reject_outliers	None	None	Flag data points for exclusion from the target function calculations this macrocycle	No outlier rejection
setup	None	None	Any preparation of the Refine Class prior to minimization	None
cleanup	None	None	Any reconfiguration of Refine Class between macrocycles	None
finalize	None	None	Any finalization of the Refine Class before exit	None
maximum_distance_special†	Float	List of Float*, List of Float*, List of Float*, List of Bool*, Float	Specialist function for restricting the line-search distance when there are correlated parameters	None
Logging Functions				
initial_statistics	None	None	Report initial statistics	None
current_statistics	None	None	Report current statistics	None
final_statistics	None	None	Report final statistics	None

example script is implemented to minimize the function with respect to all parameters in each macrocycle. Therefore, the set_macrocycle_protocol function does not perform any parameter selection.

The major architectural difference between *scitbx.lbfgs* and *dtmin* is that *dtmin* performs the overall refinement to convergence (with a configurable strategy), whereas in *scitbx.lbfgs,* the overall refinement strategy is not implemented in the library. The *scitbx* implementation of the twisted Gaussian minimization consists of the definition of the target function, gradients and curvatures, and then calls to steps in the *lbfgs* minimizer, such as requests_f_and_g() and requests_diag() (see scitbx documentation for more details). The *dtmin* implementation of the twisted Gaussian consists of a definition of the

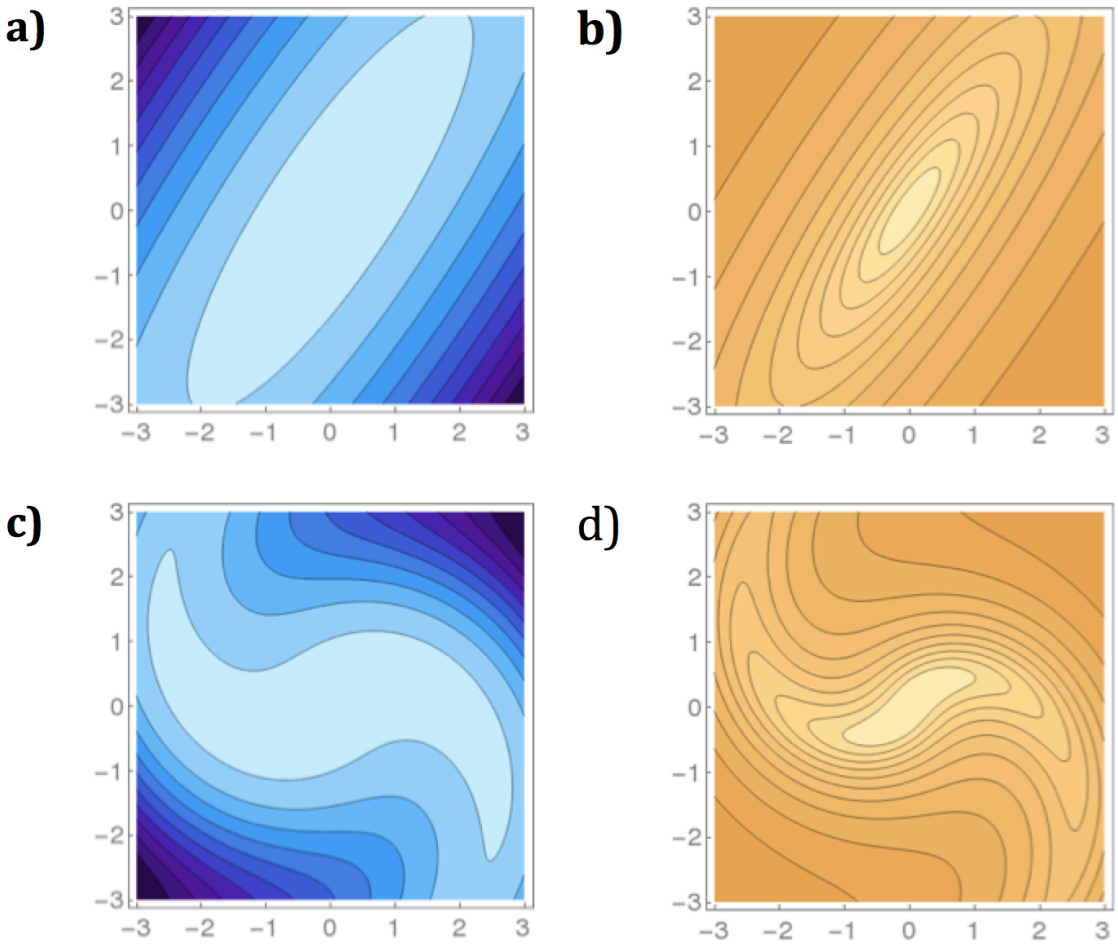


Figure 2: 'Twisted Gaussian' plotted with linear contour levels in (a) and (c) and with log-log scaled contour levels to accentuate the visualization of the gradient near the minimum in (b) and (d). (a) and (b) The functional form is the negative of the log of a bivariate Gaussian with covariate matrix entries s11=1.0, s12= 1.2, s22=2.0 and the twist parameter *t*=0, i.e. 'untwisted' Gaussian. (c) and (d) as in (a) and (b) with the twist parameter *t*=0.5.

refinement class (RefineTG, derived from RefineBase as described above) and one call to the Minimizer class 'run' function to perform the minimization to convergence. In the example, *dtmin* is configured to run with two macrocycles.

Figure 2 shows the twisted Gaussian function (details below) with two different degrees of twist, one with no twist (untwisted) and the other with a non-zero twist parameter (*t=0.5*). Figures 3 and 4 show the minimization paths taken by different minimizers in the case of the untwisted and twisted Gaussian, respectively, starting at three different positions. For the untwisted Gaussian, the path to the minimum takes fewest steps when the full Hessian is used and can be reached in one step with either the Newton (3j) or BFGS (3h) minimizer. Note that the minimization path is the same for these two because the function is quadratic. For the twisted Gaussian, the BFGS minimizer takes fewer steps than the Newton minimizer, although for one starting position (2,2) and the Hessian primed with the curvatures (4g), the minimizer does not reach the minimum in the two macrocycles of the protocol. In figure (4b) the starting position (2,-2) fails to refine using *scitbx.lbfgs* because the calculation generates a negative element in the diagonals of the inverse Hessian; the comparable test with the *dtmin* BFGS algorithm in figure (4g) succeeds because *dtmin* uses a heuristic involving the large shift values to repair negative curvatures. Newton

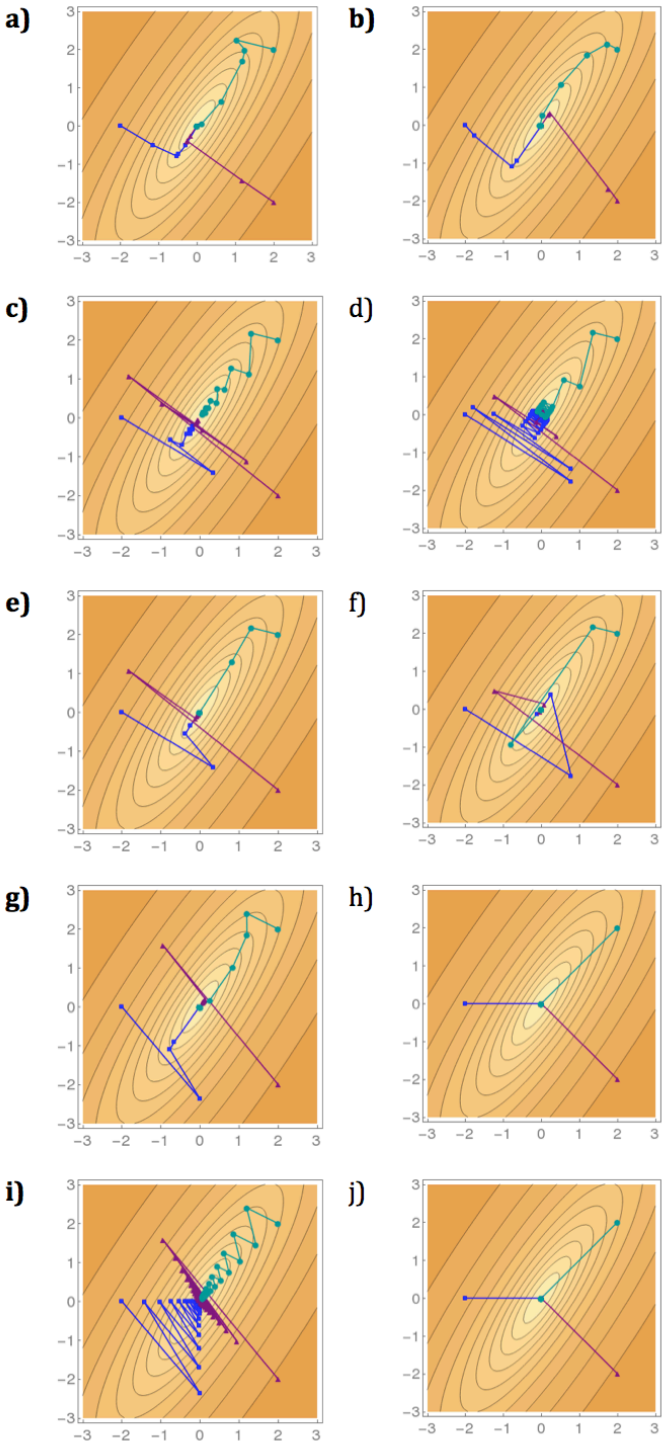


Figure 3: Path to minimum taken by different minimization methods for the 'untwisted' Gaussian in Figure 2 starting at three different positions: (2,2) in cyan, (2,-2) in purple and (-2,0) in blue. (a) and (b). a) *scitbx.lbfgs* minimizer without curvatures b) *scitbx.lbfgs* minimizer with curvatures c) *dtmin* Newton minimizer with the Hessian set to the identity matrix d) *dtmin* Newton minimizer with the diagonals of the Hessian set to the reciprocal of the square of the large shift value for each parameter e) *dtmin* BFGS minimizer with the Hessian set to the identify matrix f) *dtmin* BFGS minimizer with the diagonals of the Hessian set to the reciprocal of the square of the large shift value for each parameter g) dtmin BFGS minimizer with the diagonals of the Hessian set to analytical curvatures h) *dtmin* BFGS minimizer with the full Hessian i) *dtmin* Newton minimizer with the diagonals of the Hessian set to analytical curvatures j) *dtmin* Newton minimizer with the full Hessian.

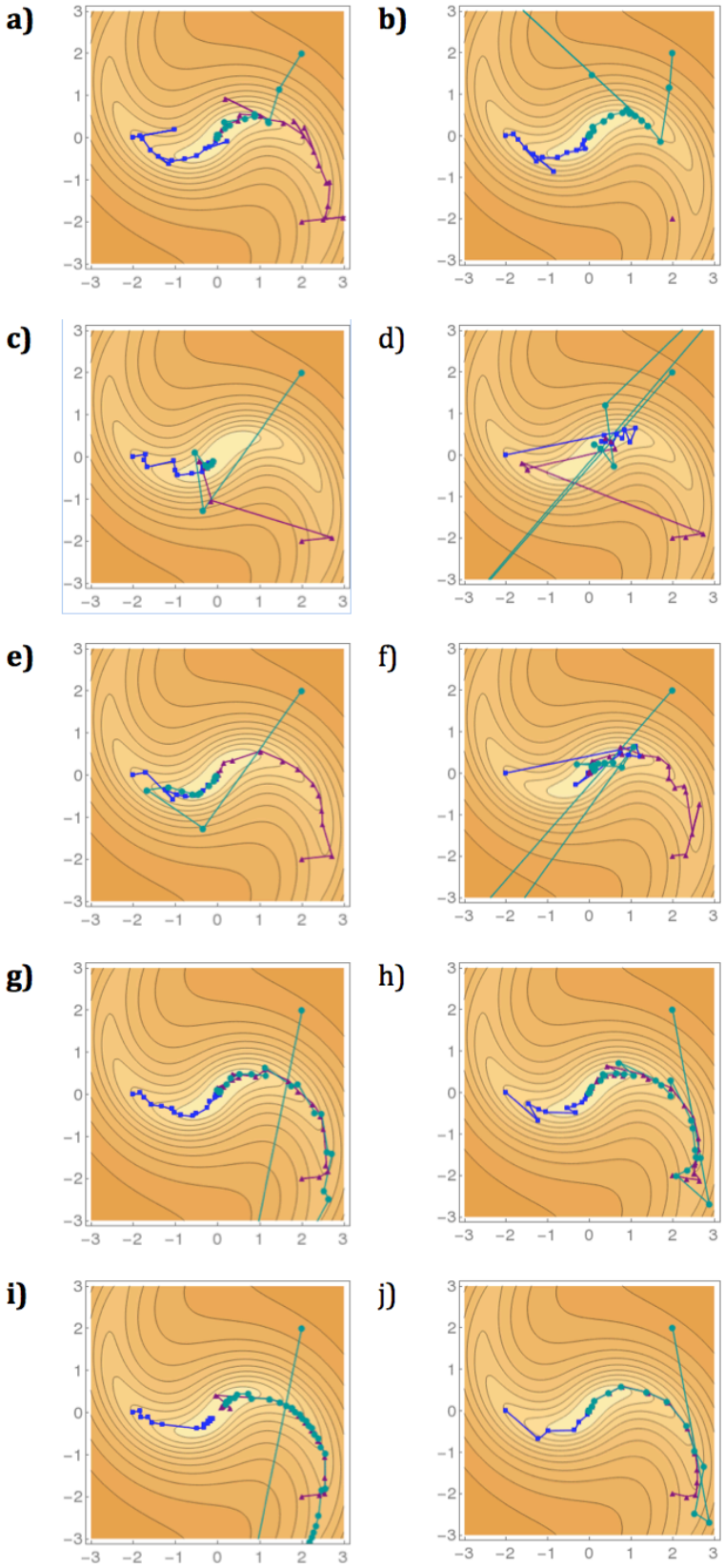


Figure 4: Path to minimum taken by different minimization methods for the 'twisted Gaussian in Figure 2 (c) and (d) starting at three different positions: (2,2) in cyan, (2,-2) in purple and (-2,0) in blue. Panels as in Figure 3.

minimization with the Hessian set to the identity matrix is equivalent to steepest descent. The appropriate minimizer and minimization protocol to use in any given case will depend on the properties of the function to be minimized and its parameterization.

The plots in Figures 2, 3 and 4 can be generated using scripts available at phaserwiki.

The functional form of the twisted Gaussian is as follows:

$$\mathrm{TG}(x,y) = -\log\left(\frac{1}{\sqrt{4\pi(s_{11}s_{22}-s_{12}^2)}}\exp\left(-\frac{s_{22}x_t^2-2s_{12}x_ty_t+s_{11}y_t^2}{2(s_{11}s_{22}-s_{12}^2)}\right)\right)$$

$$\text{where } \begin{bmatrix}x_t\\y_t\end{bmatrix} = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}, \text{and } \theta = t\sqrt{x^2+y^2}$$

In Figures 3 and 4 the large shift values for both *x* and *y* were two while two macrocycles of minimization were performed, both for all parameters (*i.e. x* and *y*).

Example 2

A second example script is provided in *scitbx/dtmin/regression/tst_dtmin_basic.py*. This script is a minimal template script that minimizes a quadratic and is intended for copying and editing. The architecture of the file is shown in Figure 5.

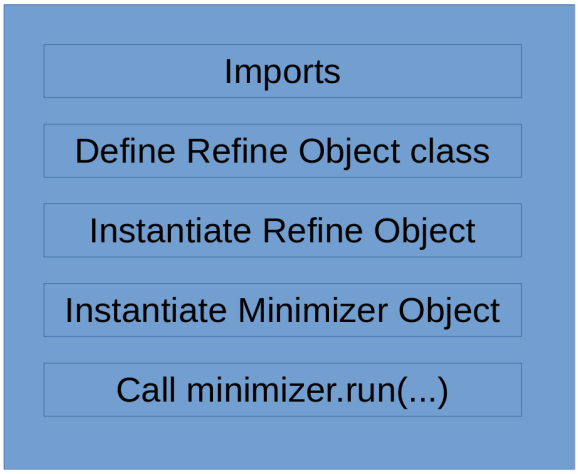


Figure 5: Architecture of a minimal python script for application of mintbx to a minimization problem. At the top of the RefineBase and Minimizer classes are imported. The function to be minimized is implemented in a 'Refine' class that inherits from RefineBase; compulsory and optional functions are described in Table 1. The derived 'Refine' class object is instantiated and passed to the Minimizer object, along with parameters to control the refinement protocol. The minimization is performed with a call to the Minimizer's run() function, and the 'Refine' object left in the minimized state.

References

McCoy, A. J. (2004). *Acta Crystallogr. D*. **60**, 2169–2183.

McCoy, A. J., Read, R. J., Bunkóczi, G. & Oeffner, R. D. (2009).

Phaserwiki, http://www.phaser.cimr.cam.ac.uk.

Wolfram Research Inc. (2019). https://www.wolfram.com/mathematica.