Automated structure refinement with *phenix.refine*

Pavel Afonine

Computation Crystallography Initiative
Physical Biosciences Division
Lawrence Berkeley National Laboratory, Berkeley CA, USA

Crystallographic structure refinement

- **Today's choices for refinement programs**
  - SHELX
  - REFMAC
  - CNS
  - BUSTER-TNT
  - MOPRO
  - phenix.refine

- **Focus of next slides is:**

  phenix.refine: a highly-automated state-of-the-art structure refinement program which is part of PHENIX package

  Development mainly at Lawrence Berkeley National Lab (USA):

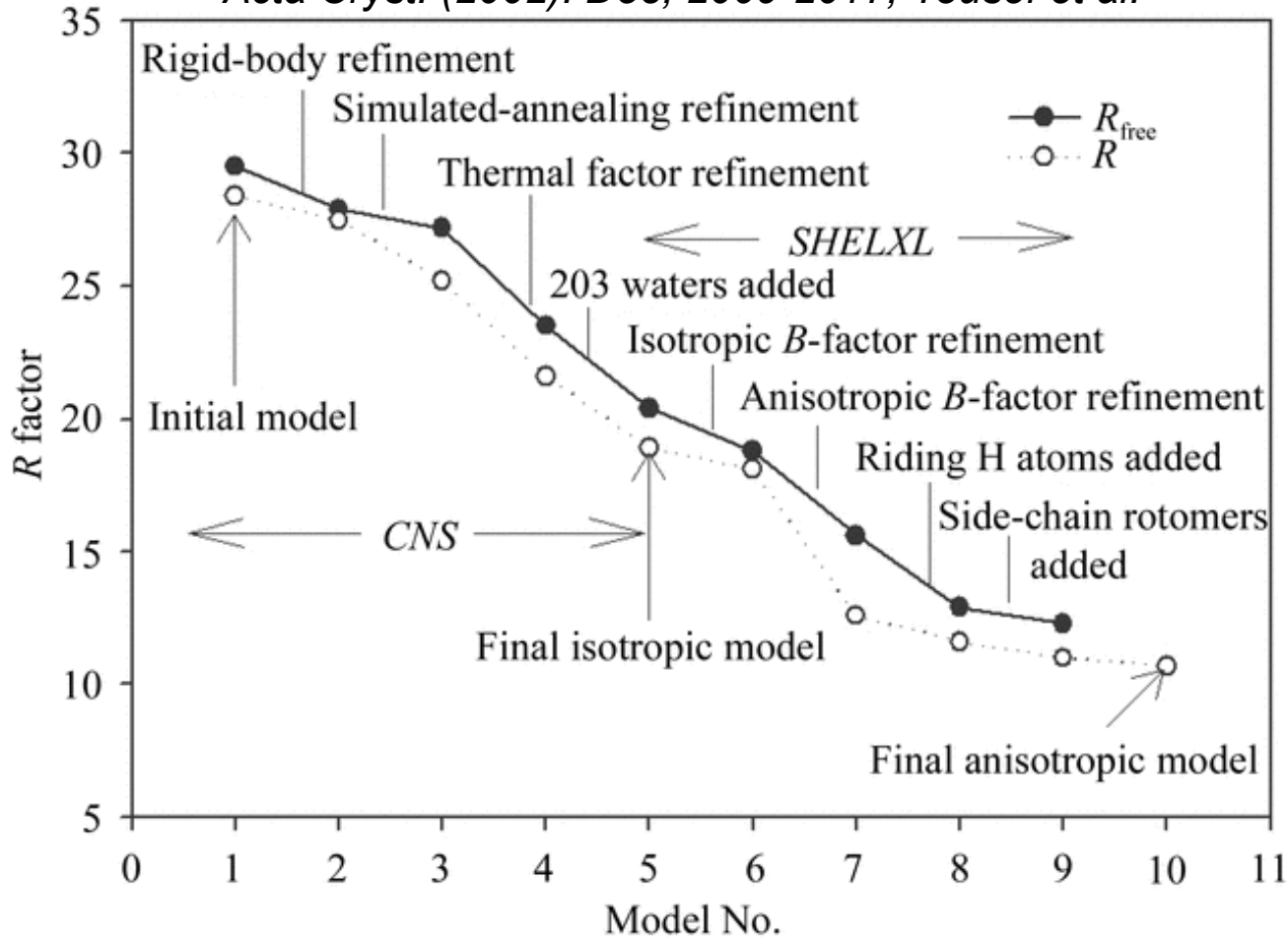
  Paul Adams, Pavel Afonine, Nathaniel Echols, Ralf Grosse-Kunstleve, Jeff Headd, Nigel Moriarty, Peter Zwart

  + *valuable contribution by many others (*Marat Mustyakimov*, Sasha Urzhumtsev, Vladimir Lunin, …)*

Automation of structure refinement

- What used to be in the past … and often still the case nowadays

Acta Cryst. (2002). D58, 2009-2017, Yousef et al.



- Clearly, the modern software should do all these steps automatically

- This is one of the goals of phenix.refine

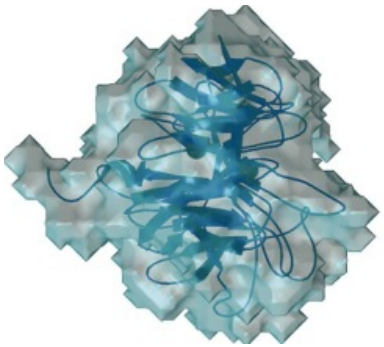
Automation of structure refinement

Round	Action taken	Program	Resolution (Å)	R (%)	R_{free} (%)
1	Simulated annealing	P	30–1.5	24.83	27.28
2	Isotropic, add solvent	P	10–1.1	15.17	16.80
3	Same atoms, isotropic	S	10–1.1	14.75	16.83
4	All-atom anisotropic	S	10–1.1	11.59	14.52
5	Add disorders (add Leu129)	S	10–1.1	10.93	14.00
6	Change resolution	S	8–1.0	11.22	13.75
7	Isotropic, water	P	8–0.65	16.78	17.24
8	Same atoms, isotropic	S	8–0.65	16.56	17.53
9	Anisotropic, add disorders	S	8–0.65	10.75	11.86
10	Isotropic, water	P	30–0.65	16.95	17.55
11	Same atoms, isotropic	S	30–0.65	16.33	17.24
12	Anisotropic, add disorders	S	30–0.65	10.71	11.69
13	Minor adjustments	S	30–0.65	10.10	11.12
14	Riding hydrogens added	S	30–0.65	9.16	10.04
15	Minor adjustments	S	30–0.65	9.00	9.95
16	Add flexible loop	S	30–0.65	8.71	9.63
17	Weighting changed	S	30–0.65	8.65	9.62
18	Restraints removed	S	30–0.65	8.48	9.59
19	Water occupancies refined	S	30–0.65	8.39	9.52
20	Free R removed	S	30–0.65	8.39	—

Wang *et al., Acta Cryst. (2007). D63, 1254-1268*

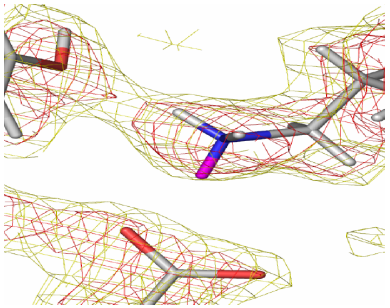
phenix.refine: single program for a very broad range of resolutions

Low



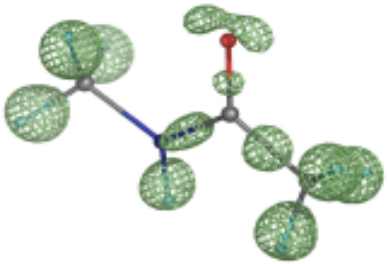
- Group ADP refinement
- Rigid body refinement
- Torsion Angle dynamics
- Reference model
- Ramachandran plot restraints
- Secondary structure restraints

Medium and High



- Restrained/constrained refinement of individual parameters
- Automatic water update

Subatomic

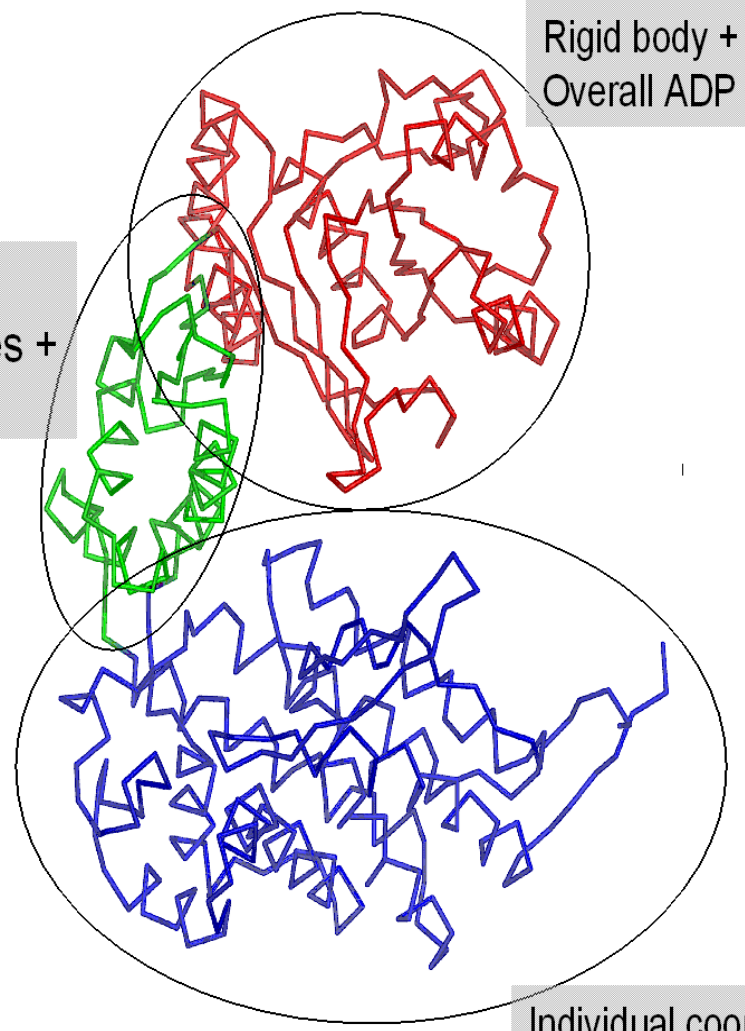


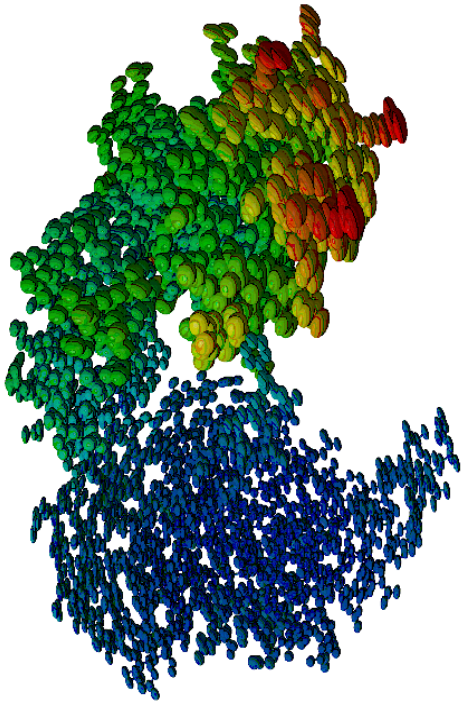
- Bond density model
- Unrestrained refinement
- FFT or direct
- Explicit hydrogens

- Automatic NCS restraints
- Simulated Annealing
- Automatic side chain rotamer fixing
- Occupancies (individual, group, automatic constrains for alternative conformations)
- Various targets: LS, ML, MLHL,…
- Dual (real/reciprocal) space refinement

- TLS refinement with automated TLS groups identification
- Use hydrogens at any resolution
- Refinement with twinned data
- X-ray, Neutron, joint X-ray + Neutron

Refine any part of a model with any strategy: *all in one run*





+ Automatic water picking

+ Simulated Annealing

+ Add and use hydrogens

Running phenix.refine

- Designed to be very easy to use
- Several ways of running:
  - command line version:

```
phenix.refine model.pdb data.hkl [parameters]
```

    - Can be highly customized (more than 300 parameters available to change)
  - can be called from (a Python) script allowing to run it within different contexts
  - GUI version

Refinement flowchart

PDB model,
Any data format
(CNS, Shelx, MTZ, …)

Input data and model processing

Refinement strategy selection

Bulk-solvent, Anisotropic scaling, Twinning parameters refinement

Ordered solvent (add / remove)

Target weights calculation

Coordinate refinement (real- and reciprocal space) (rigid body, individual) (minimization or Simulated Annealing)

ADP refinement
(TLS, group, individual iso / aniso)

Occupancy refinement (individual, group)

Output: Refined model, various maps, structure factors, complete statistics, ready for deposition PDB file

Repeated several times

Files for COOT, O, PyMol

✓ *Input data: can be intensities: French&Wilson method is used to convert Iobs to Fobs – no need to run Truncate*

Automatic Water Picking

✓ Water is updated (add/remove/refine) automatically as part of refinement run:

- No need to do it as a separate step using external tools

Input data and model processing

Refinement strategy selection

Bulk-solvent, Anisotropic scaling, Twinning parameters refinement

Ordered solvent (water picking)

Target weights calculation

Coordinate refinement
(rigid body, individual) (minimization or SA)

ADP refinement
(TLS, group, individual iso / aniso)

Occupancy refinement (individual, group)

Output: Refined model, various maps, structure factors, complete statistics, ready for deposition PDB file

- **remove** "bad" water:
  - 2mFo-DFc (peak height)
  - distances
  - map CC (2mFo-DFc, Fc)
  - B-factors and anisotropy
  - occupancy
- **add** new:
  - mFo-DFc,
  - distances
- **pre-refine** water parameters

Refinement flowchart

PDB model,
Any data format
(CNS, Shelx, MTZ, …)

Input data and model processing

Refinement strategy selection

Bulk-solvent, Anisotropic scaling, Twinning parameters refinement

Ordered solvent (add / remove)

Target weights calculation

Coordinate refinement (real- and reciprocal space) (rigid body, individual) (minimization or Simulated Annealing)

ADP refinement
(TLS, group, individual iso / aniso)

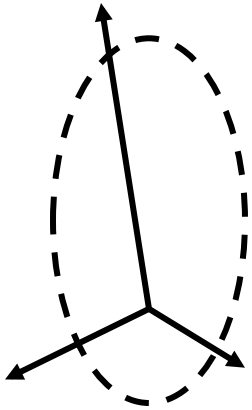
Occupancy refinement (individual, group)

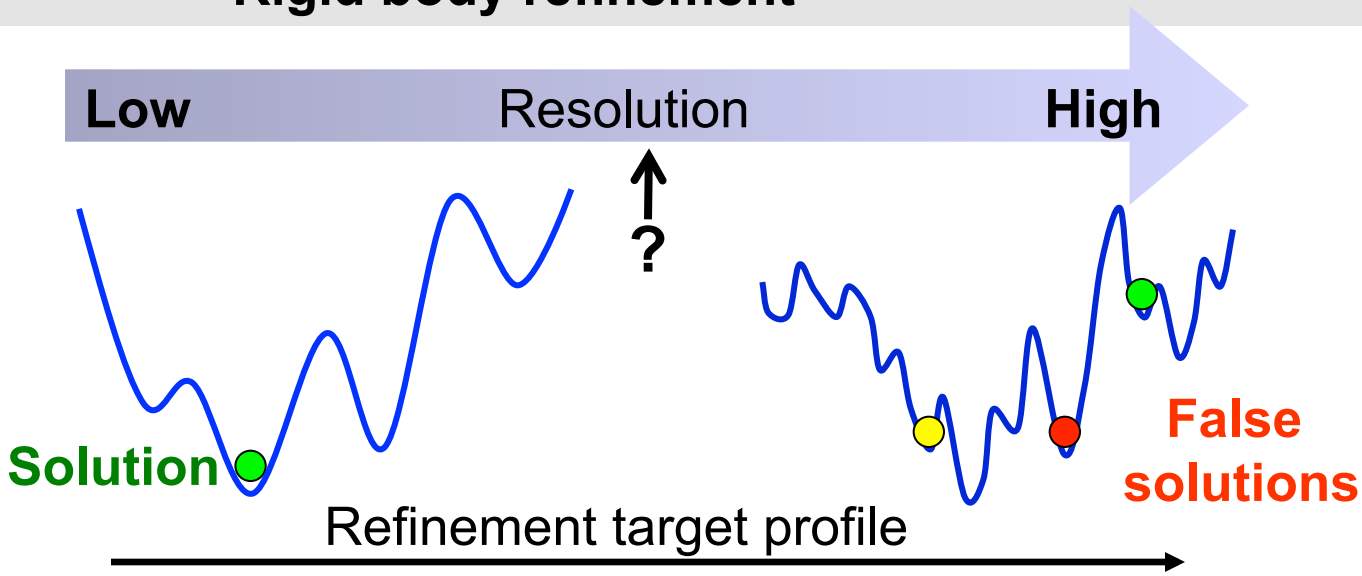
Output: Refined model, various maps, structure factors, complete statistics, ready for deposition PDB file

Repeated several times

Files for COOT, O, PyMol

Rigid body refinement

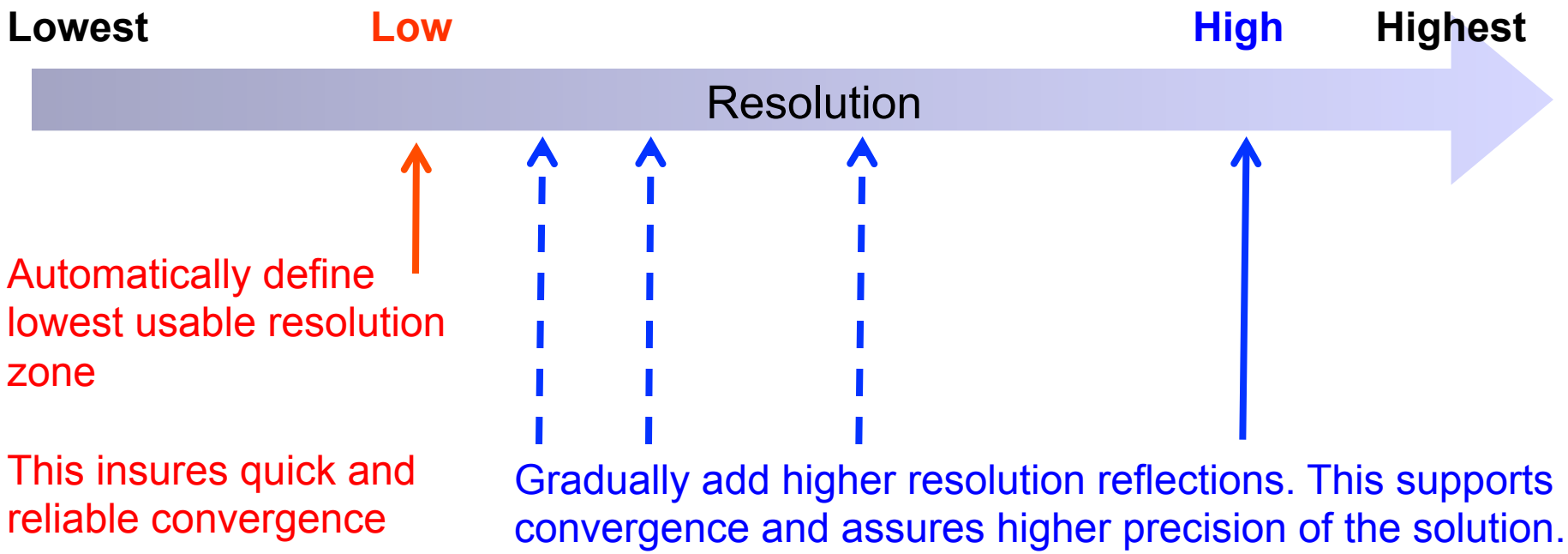




- **Rigid body refinement challenges:**
  - Need to use *low resolution* reflections to achieve a solution
    - Using too low resolution may not be good
    - Need to use higher resolution data to assure better solution
    - How to define low-high resolution border (3…4…6A)?
- **PHENIX MZ protocol** makes all these decisions automatically

Automatic multiple-zone rigid-body refinement with a large convergence radius.
P. V. Afonine, R. W. Grosse-Kunstleve, A. Urzhumtsev and P. D. Adams. J. Appl. Cryst. 42, 607-615 (2009)

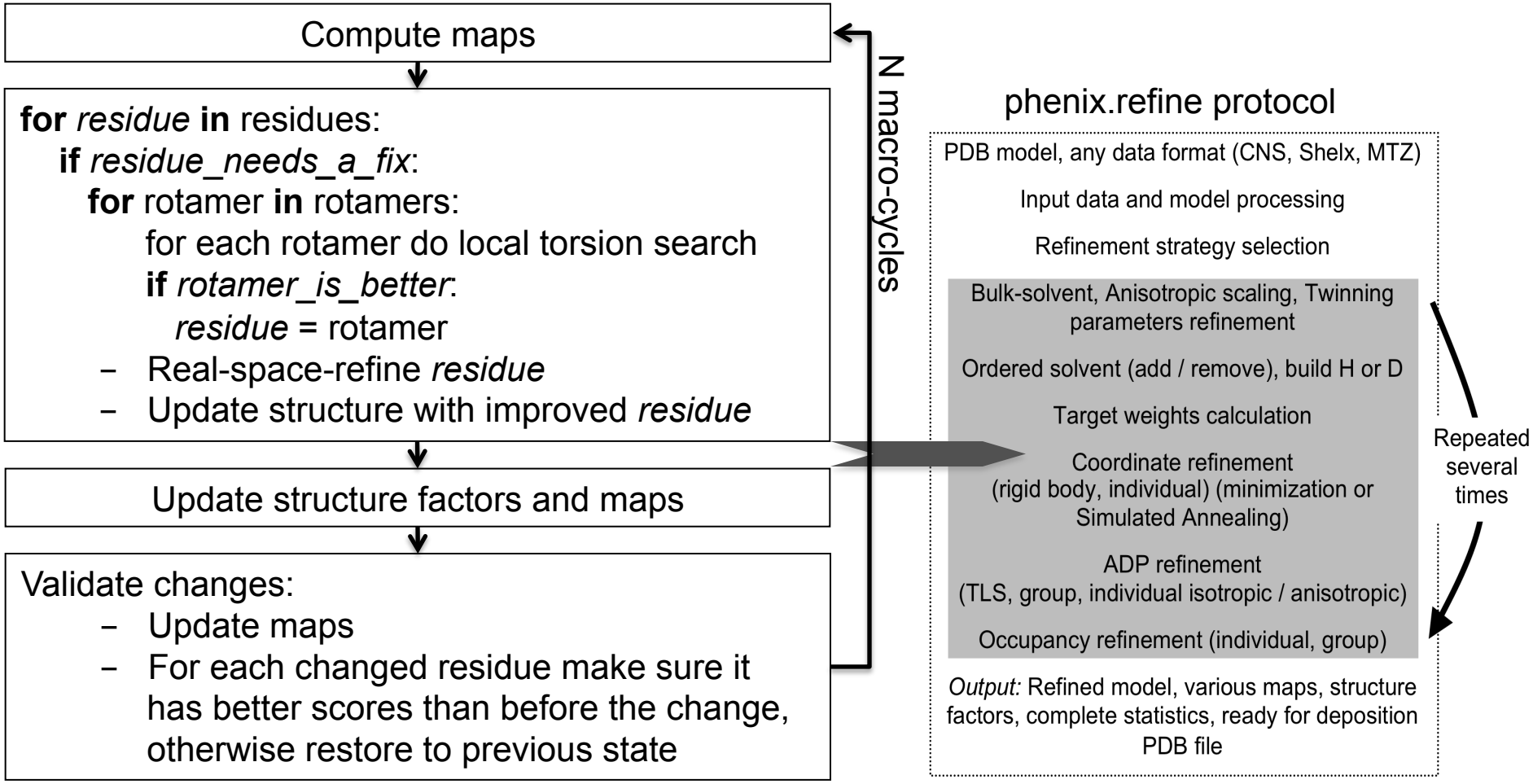
Automated Rigid Body Refinement in PHENIX (MZ protocol)



During rigid body refinement some large model movements are expected.
This invalidates the solvent mask, so the bulk-solvent model is updated at each step.

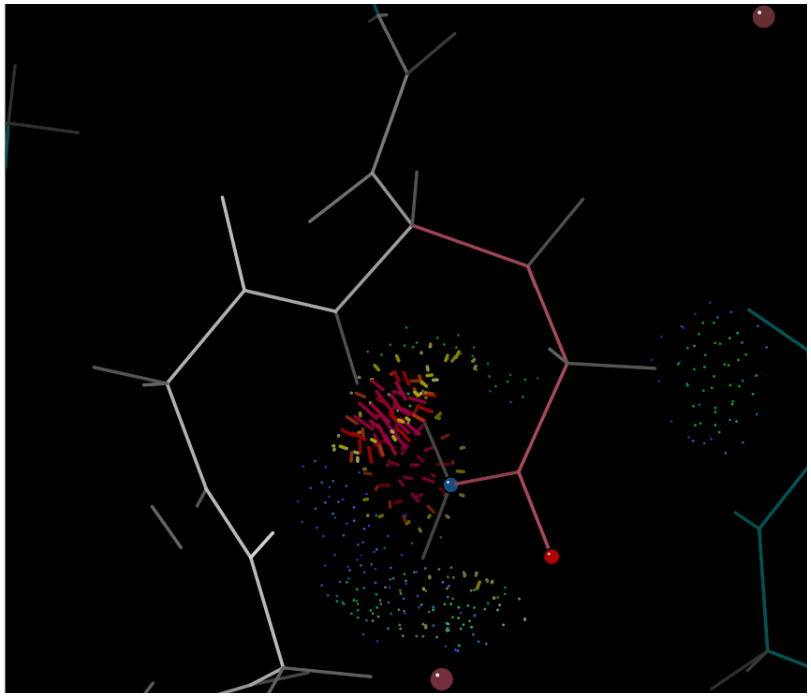
- All parameters used in the protocol are optimized to achieve the highest convergence radius at minimal runtime.
  - This is done by the grid search over ~100000 trial refinements using more than 100 different structures.

Local real-space refinement (fix_rotamers)

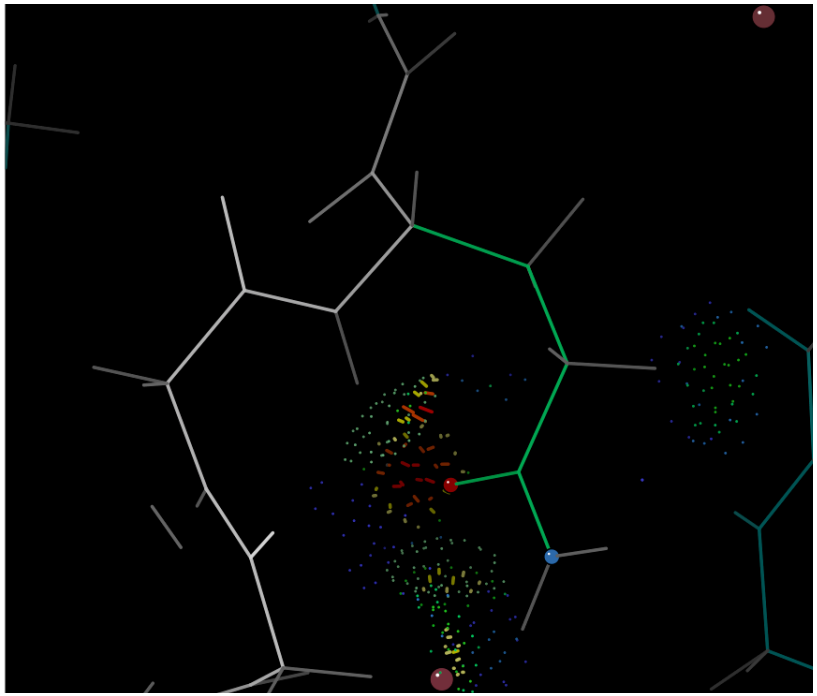


Automatic side chain flips to avoid bad clashes

- **phenix.refine** always applies side chain flips automatically (Asn/Gln/His)



Bad



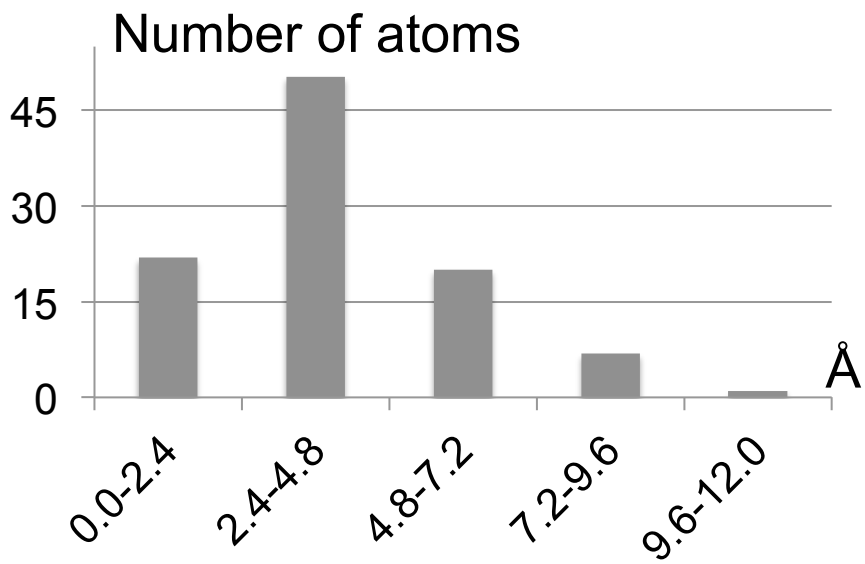
Good

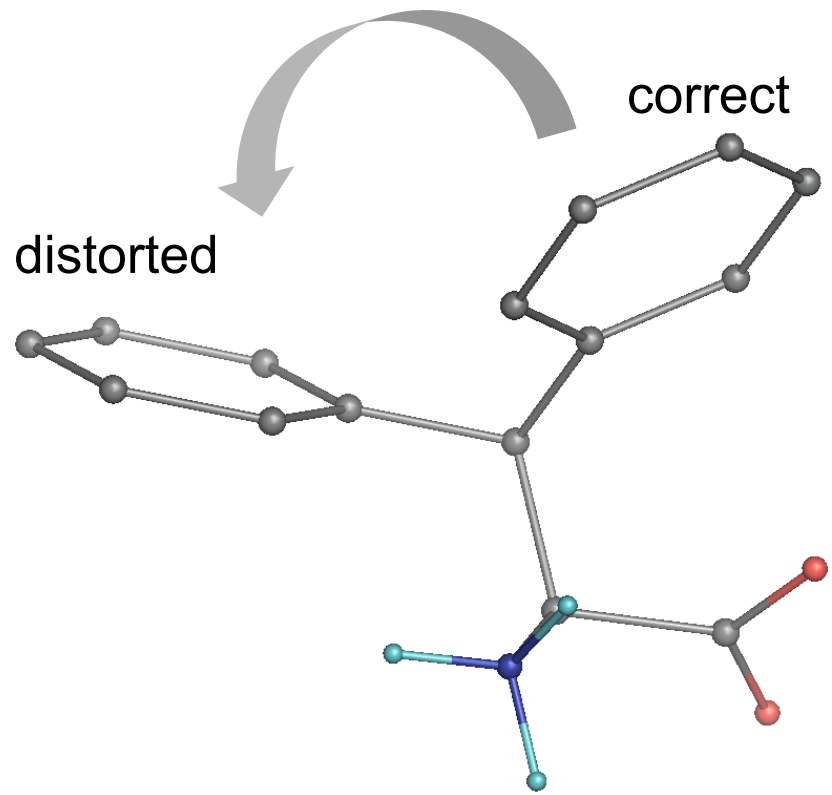
Test refinements: distorted models

- **Distorted models** (150 randomly picked from PDB structures at resolutions from 1.5 to 3Å):

  1. Remove water
  2. For each residue select the most distant rotamer
  3. Quick geometry regularization to remove bad clashes

- **Side chain distortions**:



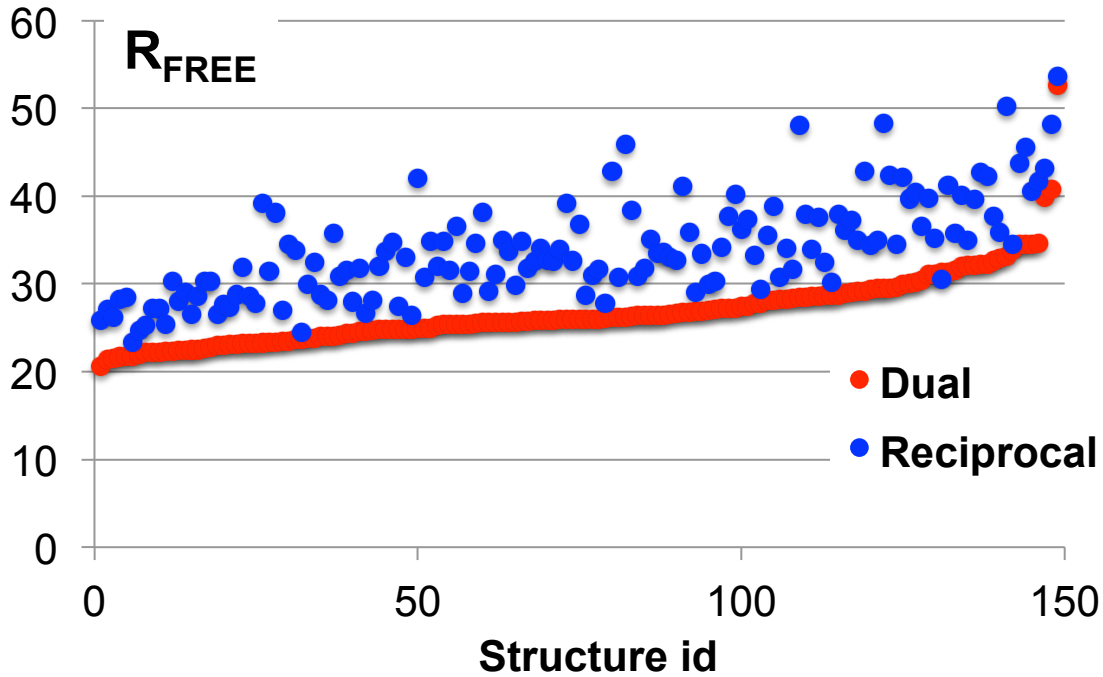


- **Main chain distortions**: rmsd~0.4 Å

Dual-space refinement: example

- **150 randomly picked from PDB structures at resolutions from 1.5 to 3Å**
  - Structures severely distorted:
    - > remove water
    - > each side-chain switched to a different rotamer
    - > geometry regularization

- **R_{FREE} after Reciprocal and Dual space refinement (sorted by R_{FREE} Dual)**



Reference Model Selections

Same chain ID:

```
refinement.reference_model.reference_group {
     reference = chain A
     selection = chain A
}
```

Different chain ID:

```
refinement.reference_model.reference_group {
     reference = chain A
     selection = chain B
}
```

Same chain ID, same residue range:

```
refinement.reference_model.reference_group {
     reference = chain A and resseq 2:119
     selection = chain A and resseq 2:119
}
```

Same chain ID, different residue range:

```
refinement.reference_model.reference_group {
     reference = chain A and resseq 130:134
     selection = chain A and resseq 120:124
}
```

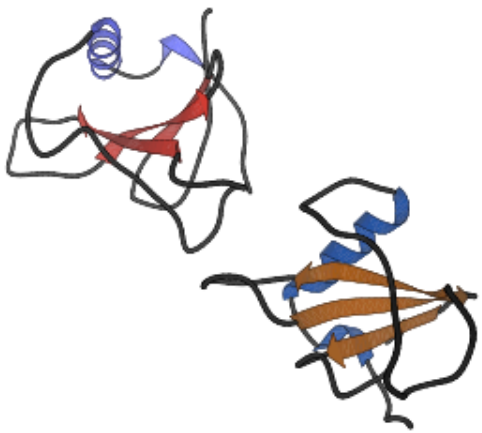
reference model refinement

MolProbity statistics after 5 macrocycles of *phenix.refine*

	Validation Criteria	1GTX, no ref.	1OHV	1GTX, 1OHV ref.	Target Value
All-Atom Contacts	Clashscore, all atoms:	**24.5**	**7.98**	**13.54**	
	Clashscore percentile	**89th**	**97th**	**97th**	
Protein Geometry	Poor rotamers:	**12.31%**	**2.30%**	**4.63%**	< 1%
	Rama outliers:	**0.65%**	**0.22%**	**0.27%**	< 0.2%
	Rama favored:	**92.88%**	**97.06%**	**96.14%**	> 98%
	Cβ dev. > 0.25Å:	**3**	**0**	**3**	0
	MolProbity score:	**3.16**	**1.87**	**2.41**	
	MP score percentile	**64th**	**94th**	**96th**	
	Res w/ bad bonds:	**0.00%**	**0.00%**	**0.00%**	0%
	Res w/ bad angles:	**0.38%**	**0.00%**	**0.43%**	< 0.1%
Residual	R-work	**0.1546**		**0.1586**	
	R-free	**0.2379**		**0.2186**	

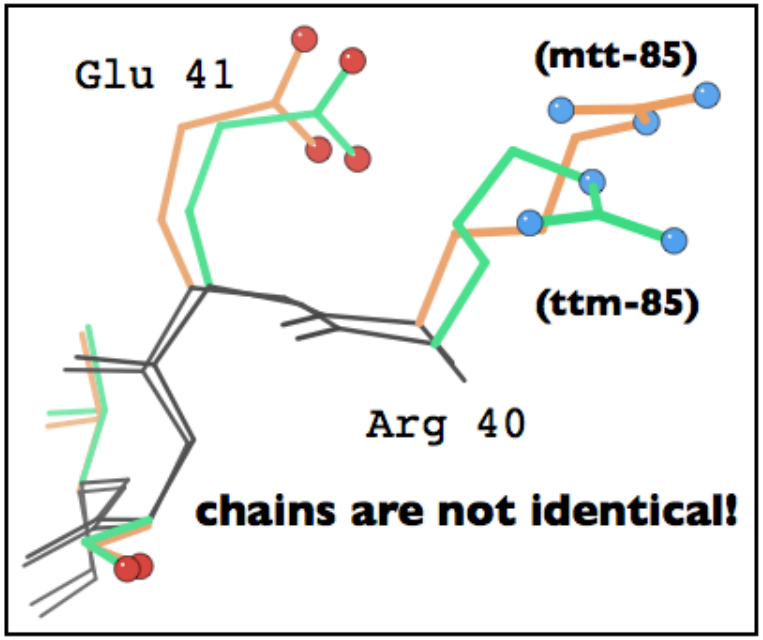
Extending Reference Model Restraints to NCS

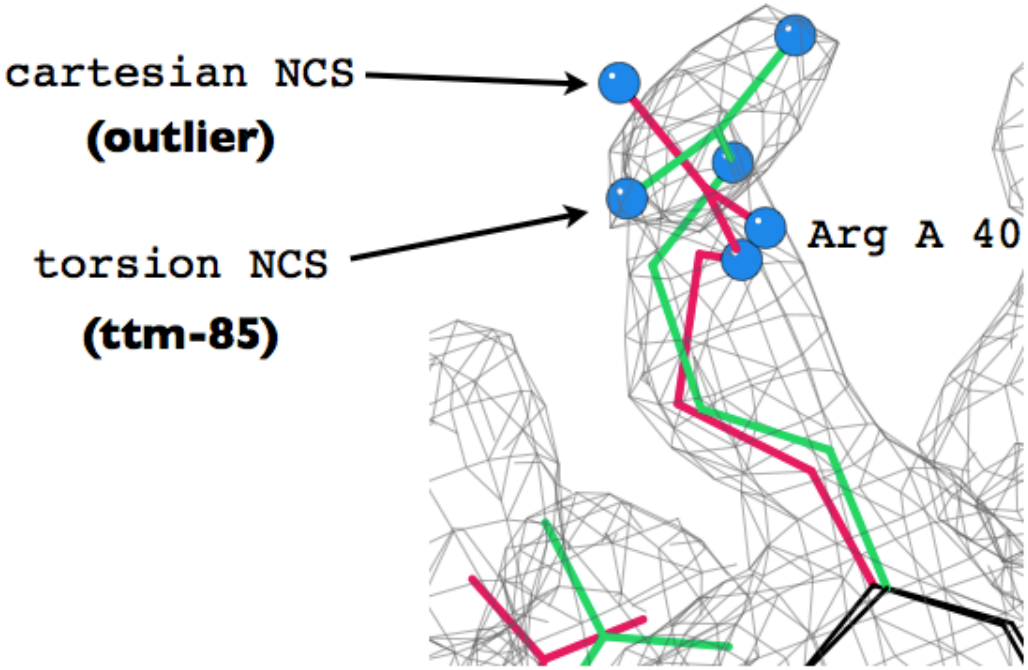
Rnase S - 2.5Å



5 macrocycles of *phenix.refine*

	R	R-free	clashscore	rotamer outliers
Default + fix_rotamers	0.1899	0.2557	7.98	1.83%
Cart. NCS + fix_rotamers	0.2106	0.2705	8.67	7.93%
Torsion NCS + fix_rotamers	**0.1890**	**0.2389**	**6.59**	**1.22%**





Refinement flowchart

PDB model,
Any data format
(CNS, Shelx, MTZ, …)

Input data and model processing

Refinement strategy selection

Bulk-solvent, Anisotropic scaling, Twinning parameters refinement

Ordered solvent (add / remove)

Target weights calculation

Coordinate refinement (real- and reciprocal space) (rigid body, individual) (minimization or Simulated Annealing)

ADP refinement
(TLS, group, individual iso / aniso)

Occupancy refinement (individual, group)

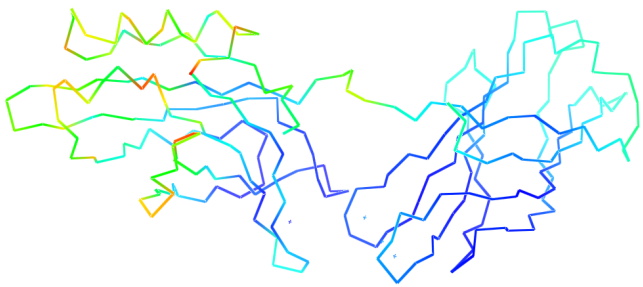
Output: Refined model, various maps, structure factors, complete statistics, ready for deposition PDB file

Repeated several times

Files for COOT, O, PyMol

ADP refinement: example

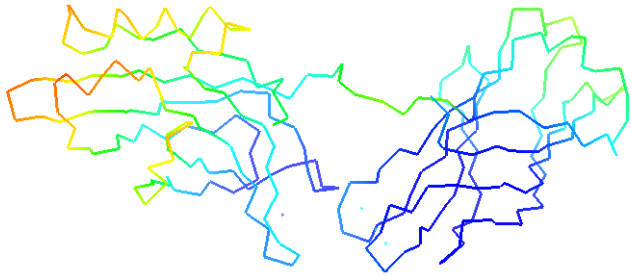
Synaptotagmin refinement at 3.2 Å



Original refinement (PDB code: 1DQV)

R-free = ***34*** *%*

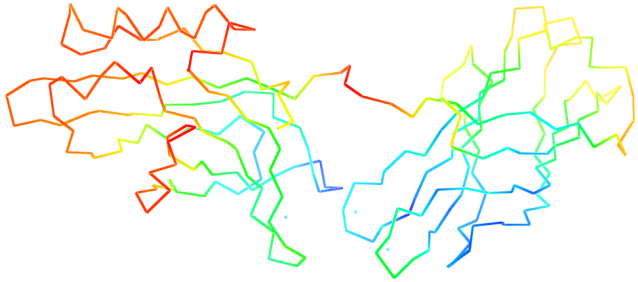
R = ***29*** *%*



PHENIX – Isotropic restrained ADP

R-free = ***28*** *%*

R = ***23*** *%*



PHENIX – TLS + Isotropic ADP

R-free = ***25*** *%*

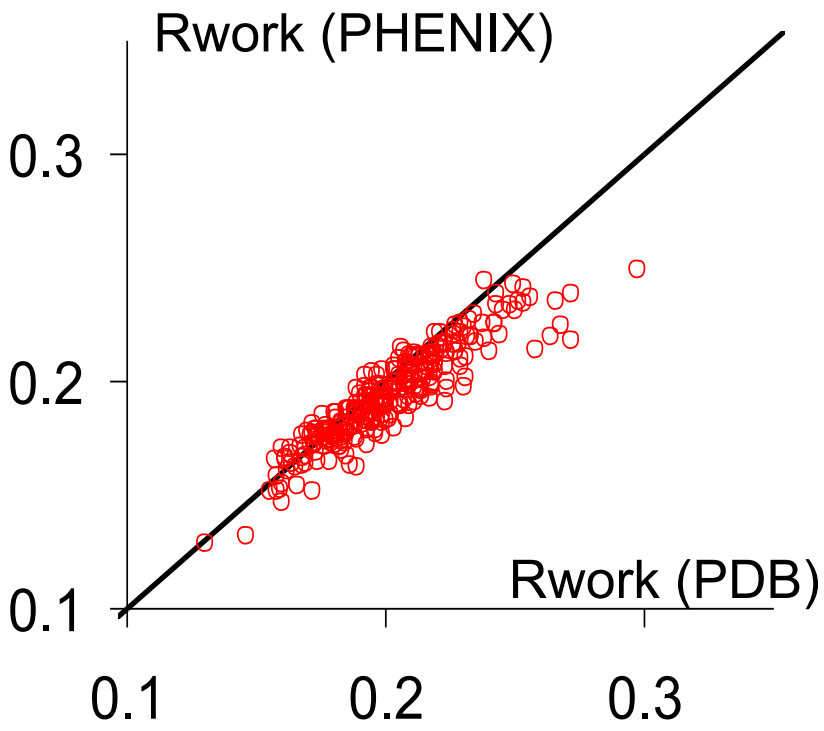
R = ***20*** *%*

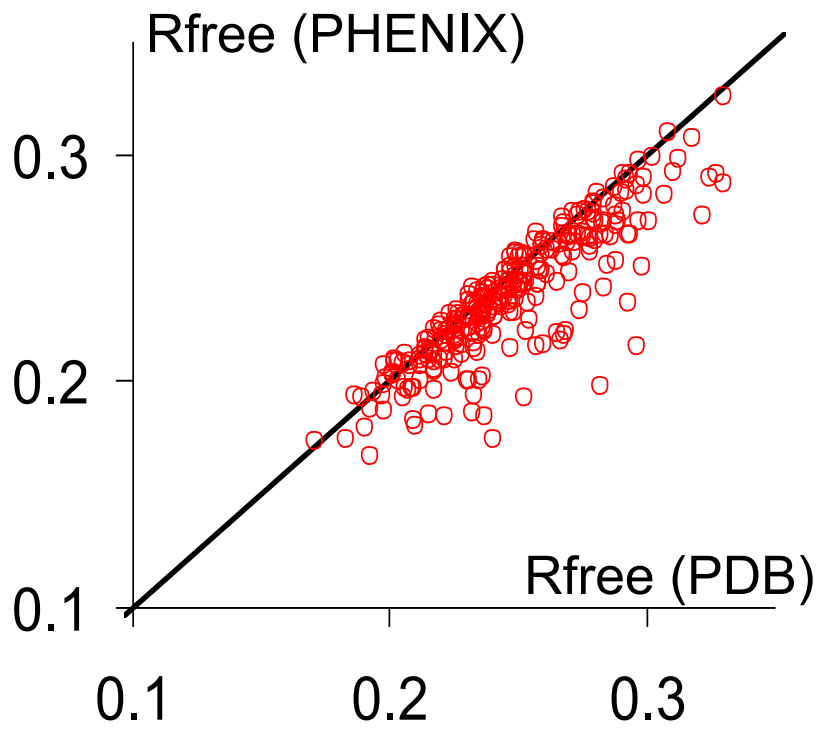
9% improvement in both Rwork and Rfree !

TLS groups determined automatically…

TLS refinement in PHENIX: robust and efficient

- Highly optimized algorithm based on systematic re-refinement of ~350 PDB models
- In most of cases *phenix.refine* produces better R-factors compared to published
- Don't crash or get "unstable"





ADP refinement: what goes into PDB

phenix.refine outputs TOTAL B-factor (iso- and anisotropic):

$$U_{TOTAL} = U_{ATOM} + U_{TLS} + U_{CRYST}$$

Isotropic equivalent

```
ATOM    1  CA  ALA  1    37.211  30.126  28.127  1.00 26.82        C
ANISOU  1  CA  ALA  1  3397    3397    3397    2634    2634    2634  C
```

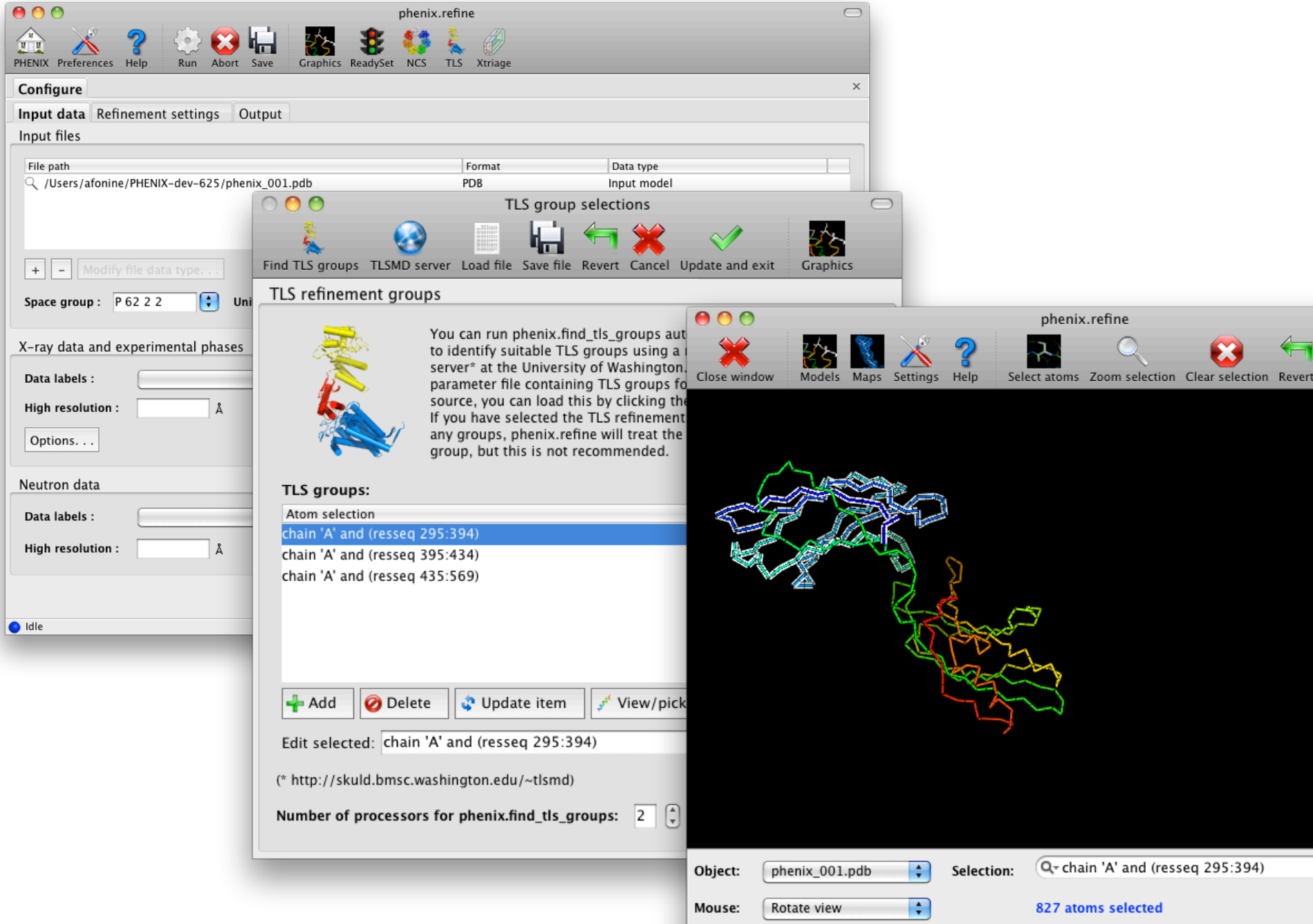
$$U_{TOTAL} = U_{ATOM} + U_{TLS} + U_{CRYST}$$

Stored in separate record in PDB file header

Atom records are self-consistent:

- ✓ Straightforward visualization (color by B-factors, or anisotropic ellipsoids)
- ✓ Straightforward computation of other statistics (R-factors, etc.) – no need to use external helper programs for any conversions.

TLS groups for refinement automatically (well, in three clicks!)

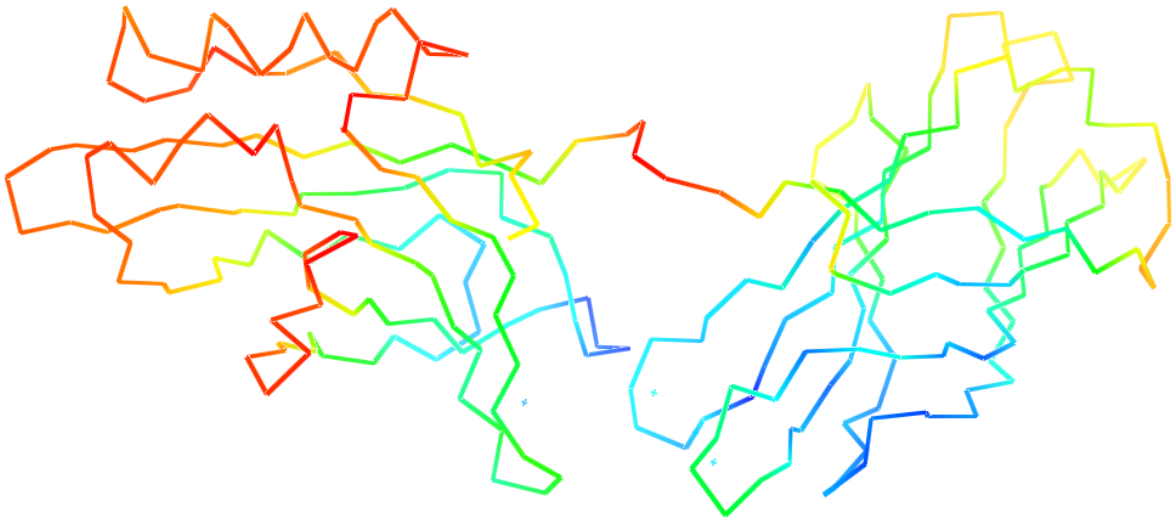


TLS

- Using TLS in refinement requires partitioning a model into TLS groups. This is typically done by
  - visual model inspection and deciding which domains may be considered as rigid
  - using TLSMD method

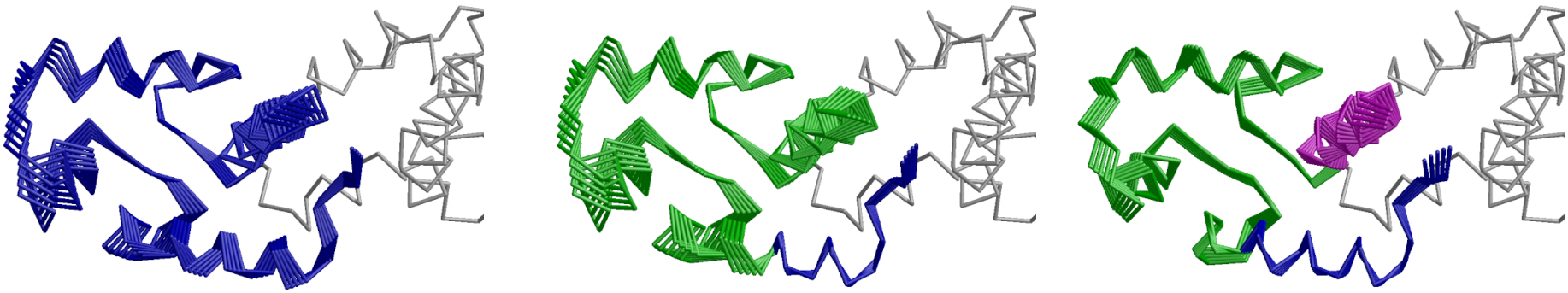
    Painter & Merritt. (2006). Acta Cryst. D62, 439-450

    Painter & Merritt. (2006). J. Appl. Cryst. 39, 109-111



TLS

- Split a model into 1, 2, 3, …, N contiguous segments.



- Compute residual for all segment partitions:

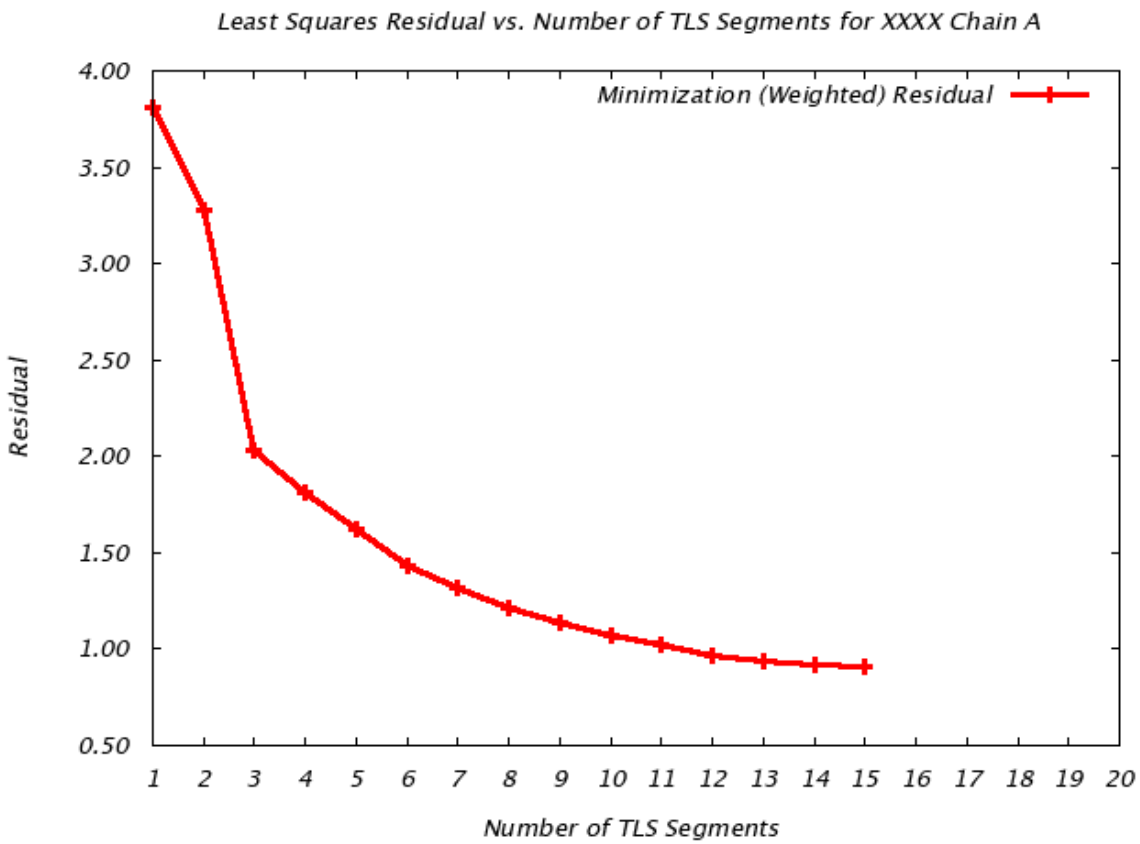
$$R = w \sum_{\text{atoms in segment}} W\left(\sum_{i=1}^{6}\left(U_{ATOM}^{i} - U_{TLS}^{i}\right)^{2}\right)$$

where TLS contribution of an individual atom participating in a TLS group:

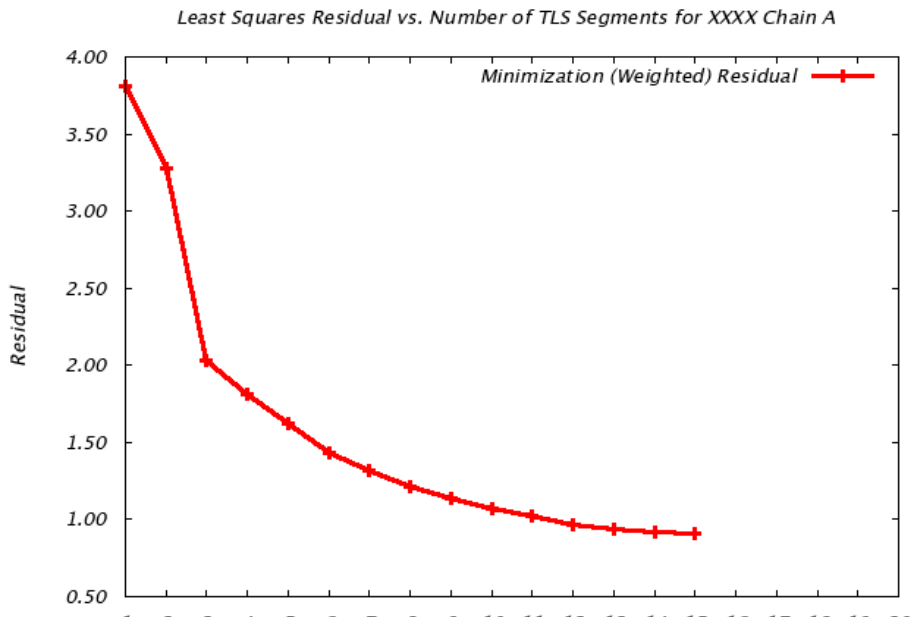
$$\mathbf{U}_{TLS} = \mathbf{T} + \mathbf{ALA}^{t} + \mathbf{AS} + \mathbf{S}^{t}\mathbf{A}^{t}$$

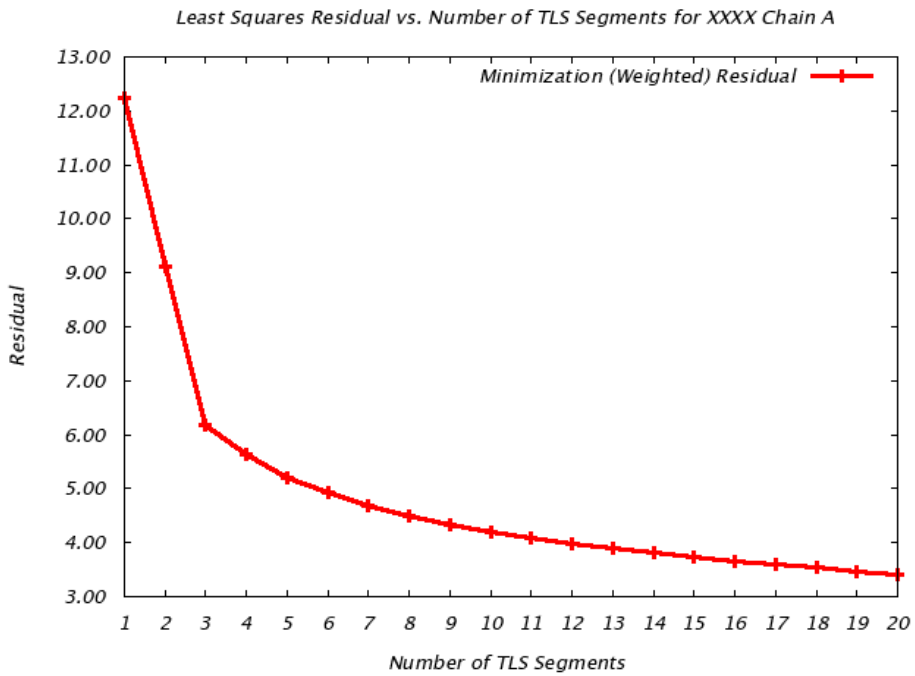
(20 TLS parameters per TLS group)

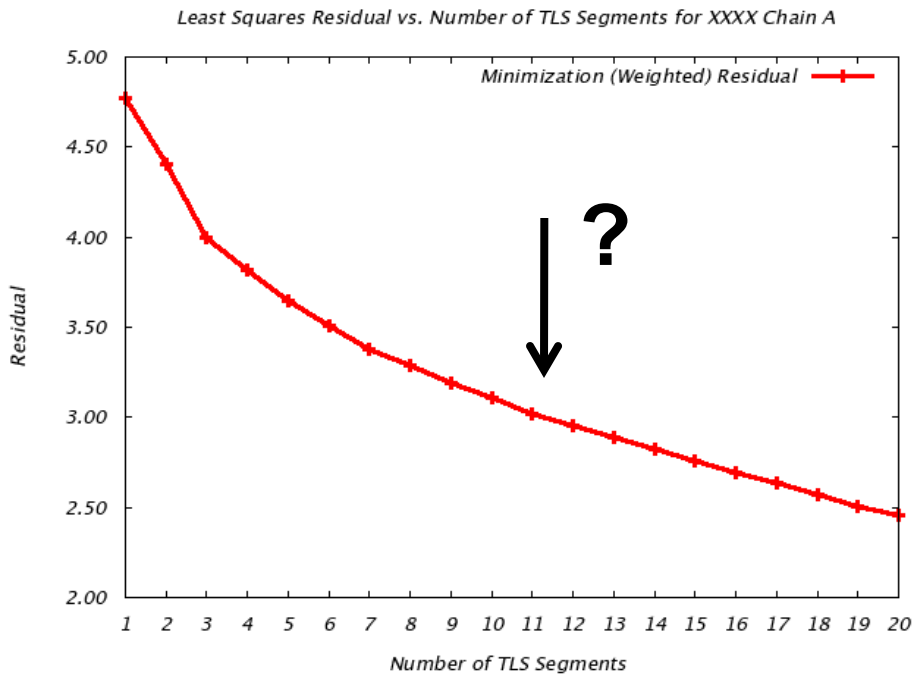
TLS


Least Squares Residual vs. Number of TLS Segments for XXXX Chain A
Minimization (Weighted) Residual
Residual
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Number of TLS Segments

TLS







How to pick up the *right* number of TLS groups?

One can try all 20 and see which one using in refinement results in best Rwork and Rfree

PHENIX approach to finding TLS groups

- **Goals:**
  - Have it as integrated part of PHENIX system:
    - No need to run external software or use web servers (=send your data somewhere, which your policy may even not allow you to do).
    - Use it interactively as part of refinement (update TLS group assignment as model improves during refinement).
    - Make it faster
  - Eliminate subjective decisions (procedure should give THE UNIQUE answer and not an array of possible choices leaving the room for subjective decisions).

PHENIX approach to finding TLS groups

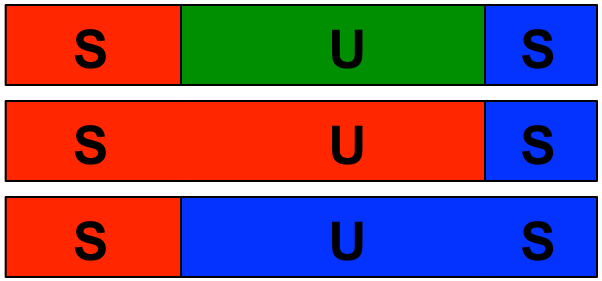
Step 1: For each chain find all secondary structure and unstructured elements

- *Number of elements defines maximum possible number of TLS groups*
- A secondary structure element can't be split into multiple TLS groups. Large unstructured elements, can be split into smaller pieces.

Chain

S – Secondary structure element
U – Unstructured stretch of residues (loop)

Step 2: Find all possible contiguous combinations

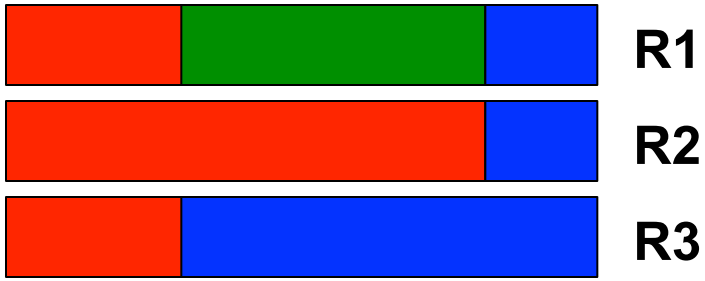


3 groups
2 groups
2 groups

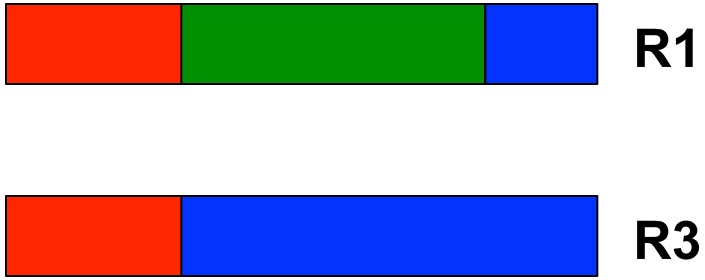
$N_{ELEMENTS} : N_{POSSIBLE\ PARTITIONS}$ 3:3, 4:7, 5:15, 6:31, …, 10:511, …

PHENIX approach to finding TLS groups

Step 3: For each partition fit TLS groups and compute the residual



Step 4: Find the best fit among the groups of equal number of partitions. In this example, if **R3**<**R2**:



Step 5: Find the best partition…

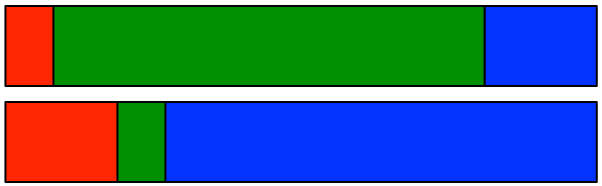
- Challenge: we can't directly compare **R1** and **R3** because they are computed using different number of TLS groups (different number of parameters)

PHENIX approach to finding TLS groups

<u>Step 5 (continued)</u>: Find the best partition…

- Randomly generate a pool of partitions for each candidate, fit TLS matrices compute, average residuals, and compute score:

R1



R11

R12

… many (20-50)

Average residuals: $\mathbf{R1_{AVERAGE}}$

Score = $(\mathbf{R1_{AVERAGE}} - \mathbf{R1})/(\mathbf{R1_{AVERAGE}} + \mathbf{R1})*100$

Do the same for the next candidate:

R3

<u>The final solution is the one that has the highest score.</u>

PHENIX approach to finding TLS groups: examples

GroEl structure (one chain):

No. of groups	Targets best	rand.pick	diff.	score
2	680.7	869.2	188.5	12.2
3	297.1	665.7	368.6	38.3
4	260.4	448.3	187.8	26.5
5	206.2	342.1	135.8	24.8
6	188.4	264.7	76.3	16.8
7	182.3	251.2	68.9	15.9
8	176.9	229.5	52.5	12.9
9	173.1	207.3	34.1	9.0
10	170.2	196.8	26.6	7.2
11	167.8	183.1	15.2	4.3
12	165.6	179.0	13.4	3.9
13	163.9	170.8	6.9	2.1

PHENIX approach to finding TLS groups: examples

GroEl:

phenix.refine refinement:

- Using TLSMD groups: R_{WORK} = 0.2044 R_{FREE} = 0.2454
- Using PHENIX determined TLS groups: R_{WORK} = 0.2054 R_{FREE} = 0.2448

Synaptotagmin:

phenix.refine refinement:

- Using TLSMD groups: R_{WORK} = 0.1967 R_{FREE} = 0.2546
- Using PHENIX determined TLS groups: R_{WORK} = 0.1970 R_{FREE} = 0.2599

1n0y:

phenix.refine refinement:

- Using TLSMD groups: R_{WORK} = 0.2093 R_{FREE} = 0.2287
- Using PHENIX determined TLS groups: R_{WORK} = 0.2088 R_{FREE} = 0.2274

1yqo:

phenix.refine refinement:

- Using TLSMD groups: R_{WORK} = 0.1580 R_{FREE} = 0.1943
- Using PHENIX determined TLS groups: R_{WORK} = 0.1584 R_{FREE} = 0.1955

Automatic TLS

- **Find optimal partition of a model into TLS groups:**

```
phenix.find_tls_groups model.pdb nproc=2
```

Examples:

GroEL structure (3668 residues, 26957 atoms, 7 chains:

135 seconds using 1 CPU

44 seconds using 10 CPUs

Analogous job using TLSMD server: 3630 seconds (plus lots of clicking to upload/download the files)

Lysozime structure:

9.5 seconds with one CPU

2.5 seconds using 10 CPUs

Automatic TLS

- **Why it is faster:**
  a) Use isotropic TLS model,
  b) Solve $R = w \sum_{\text{atoms in segment}} \left(U^i_{ATOM} - U^i_{TLS}\right)^2$ analytically (no minimizer used)

$$U_{TLS} = T_{iso} + \frac{1}{3}\left[L_{11}(y^2+z^2) + L_{22}(x^2+z^2) + L_{33}(x^2+y^2) - 2L_{12}xy - 2L_{13}xz - 2L_{23}yz + 2S_1z + 2S_2y + 2S_3x\right]$$

$$U_{TLS} = T_{iso} + L_{11}\frac{y^2+z^2}{3} + L_{22}\frac{x^2+z^2}{3} + L_{33}\frac{x^2+y^2}{3} - \frac{2}{3}xyL_{12} - \frac{2}{3}xzL_{13} - \frac{2}{3}yzL_{23} + \frac{2}{3}zS_1 + \frac{2}{3}yS_2 + \frac{2}{3}xS_3$$

$$X = \frac{y^2+z^2}{3};\ Y = \frac{x^2+z^2}{3};\ Z = \frac{x^2+y^2}{3};\ W = -\frac{2}{3}xy;\ V = -\frac{2}{3}xz;$$
$$T = -\frac{2}{3}yz;\ S = \frac{2}{3}z;\ R = \frac{2}{3}y;\ Q = \frac{2}{3}x$$

$$U_{TLS} = T_{iso} + XL_{11} + YL_{22} + ZL_{33} + WL_{12} + VL_{13} + TL_{23} + S\cdot S_1 + R\cdot S_2 + Q\,S_3$$

$$LS = \sum_{ATOMS}(U_{TLS} - U_{iso})^2;\quad LS = \sum_{ATOMS}\left(U_{TLS}^2 - 2U_{TLS}U_{iso} + U_{iso}^2\right)$$

$$LS = \sum_{ATOMS}\left(U_{TLS}^2 - 2U_{TLS}U_{iso}\right) + \sum_{ATOMS}U_{iso}^2$$

$$\frac{\partial LS}{\partial PAR} = \sum_{ATOMS}\left(2U_{TLS}\cdot\frac{\partial U_{TLS}}{\partial PAR} - 2U_{iso}\cdot\frac{\partial U_{TLS}}{\partial PAR}\right)$$

$$\frac{\partial LS}{\partial PAR} = 2\sum_{ATOMS}\left(U_{TLS}\frac{\partial U_{TLS}}{\partial PAR} - U_{iso}\cdot\frac{\partial U_{TLS}}{\partial PAR}\right) \quad (1)$$

$$\frac{\partial LS}{\partial PAR} = 2\sum_{ATOMS}\left([U_{TLS} - U_{iso}]\cdot\frac{\partial U_{TLS}}{\partial PAR}\right)$$

$$\frac{\partial LS}{\partial L_{11}} = 2\sum(U_{TLS} - U_{iso})\cdot X = 2\sum(U_{TLS}\cdot X - U_{iso}\cdot X) = 2\sum(U_{TLS}\cdot X) - \underbrace{2\sum U_{iso}\cdot X}_{P_X} = \boxed{2\sum(U_{TLS}\cdot X) + P_X} \quad (2)$$

$$\frac{\partial LS}{\partial L_{22}} = 2\sum(U_{TLS}\cdot Y) + P_Y$$

$$\frac{\partial LS}{\partial L_{33}} = 2\sum(U_{TLS}\cdot Z) + P_Z$$

$$\frac{\partial LS}{\partial L_{12}} = 2\sum(U_{TLS}\cdot W) + P_W$$

$$\frac{\partial LS}{\partial L_{13}} = 2\sum(U_{TLS}\cdot V) + P_V$$

$$\frac{\partial LS}{\partial L_{23}} = \sum(U_{TLS}\cdot T) + P_T$$

$$\frac{\partial LS}{\partial S_1} = \sum(U_{TLS}\cdot S) + P_S$$

$$\frac{\partial LS}{\partial S_2} = \sum(U_{TLS}\cdot R) + P_R$$

$$\frac{\partial LS}{\partial S_3} = \sum(U_{TLS}\cdot Q) + P_Q$$

$$\frac{\partial LS}{\partial T_{iso}} = 2\sum_{ATOMS}\left([U_{TLS} - U_{iso}]\cdot 1\right) = 2\sum_{ATOMS}(U_{TLS} - U_{iso})$$

$$U_{TLS}\cdot X = XT_{iso} + X^2L_{11} + XYL_{22} + XZL_{33} + XWL_{12} + XVL_{13} + XTL_{23} + XS\cdot S_1 + XR\cdot S_2 + XQ\,S_3$$
$$U_{TLS}\cdot Y = YT_{iso} + YXL_{11} + Y^2L_{22} + YZL_{33} + YWL_{12} + YVL_{13} + YTL_{23} + YS\cdot S_1 + YRS_2 + YQ\,S_3$$
$$U_{TLS}\cdot Z = ZT_{iso} + ZXL_{11} + ZYL_{22} + Z^2L_{33} + ZWL_{12} + ZVL_{13} + ZTL_{23} + ZS\cdot S_1 + ZRS_2 + ZQ\,S_3$$
$$U_{TLS}\cdot W = WT_{iso} + WXL_{11} + WYL_{22} + WZL_{33} + W^2L_{12} + WVL_{13} + WTL_{23} + WS\cdot S_1 + WR\cdot S_2 + WQ\cdot S_3$$
$$U_{TLS}\cdot V = VT_{iso} + VXL_{11} + VYL_{22} + VZL_{33} + VWL_{12} + V^2L_{13} + VTL_{23} + VS\cdot S_1 + VR\cdot S_2 + VQ\cdot S_3$$
$$U_{TLS}\cdot T = TT_{iso} + TXL_{11} + TYL_{22} + TZL_{33} + TWL_{12} + TVL_{13} + T^2L_{23} + TS\cdot S_1 + TR\cdot S_2 + TQ\cdot S_3$$
$$U_{TLS}\cdot S = ST_{iso} + SXL_{11} + SYL_{22} + SZL_{33} + SWL_{12} + SVL_{13} + STL_{23} + S^2\cdot S_1 + SR\cdot S_2 + SQ\cdot S_3$$

7 more pages …

Refinement flowchart

PDB model,
Any data format
(CNS, Shelx, MTZ, …)

Input data and model processing

Refinement strategy selection

Bulk-solvent, Anisotropic scaling, Twinning parameters refinement

Ordered solvent (add / remove)

Target weights calculation

Coordinate refinement (real- and reciprocal space) (rigid body, individual) (minimization or Simulated Annealing)

ADP refinement
(TLS, group, individual iso / aniso)

Occupancy refinement (individual, group)

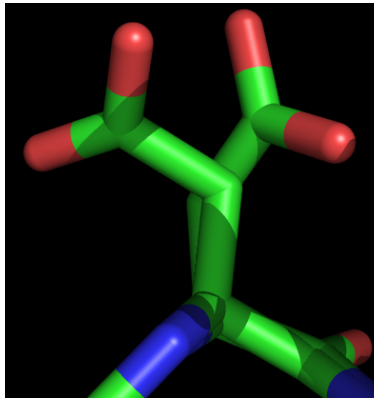
Output: Refined model, various maps, structure factors, complete statistics, ready for deposition PDB file

Repeated several times

Files for COOT, O, PyMol

Occupancy refinement

- Automatic constraints for occupancies of atoms in alternate locations



- Any user defined selections for individual and/or group occupancy refinement can be added on top of the automatic selection.

ATOM	1	N	AARG	A	192	-5.782	17.932	11.414	0.72	8.38	N
ATOM	2	CA	AARG	A	192	-6.979	17.425	10.929	0.72	10.12	C
ATOM	3	C	AARG	A	192	-6.762	16.088	10.271	0.72	7.90	C
ATOM	7	N	BARG	A	192	-11.719	17.007	9.061	0.28	9.89	N
ATOM	8	CA	BARG	A	192	-10.495	17.679	9.569	0.28	11.66	C
ATOM	9	C	BARG	A	192	-9.259	17.590	8.718	0.28	12.76	C
ATOM	549	AU		A	34	-23.064	7.146	-23.942	0.78	15.44	Au
ATOM	549	HA3	ARG	A	34	-23.064	7.146	-23.942	1.00	15.44	H
ATOM	550	H	AARG	A	34	-24.447	7.644	-21.715	0.15	8.34	H
ATOM	551	D	BARG	A	34	-24.447	7.644	-21.715	0.85	7.65	D
ATOM	552	N	ARG	A	35	-22.459	9.801	-22.791	1.00	8.54	N
ATOM	6	S	SO4		1	1.302	1.419	1.560	0.70	13.00	
ATOM	7	O1	SO4		1	1.497	1.295	0.118	0.70	11.00	
ATOM	8	O2	SO4		1	1.098	0.095	2.140	0.70	10.00	
ATOM	9	O3	SO4		1	2.481	2.037	2.159	0.70	14.00	
ATOM	10	O4	SO4		1	0.131	2.251	1.823	0.70	12.00	

Occupancy refinement – more examples

```
ATOM   3690  O2 AEDO C   1      23.106  -3.999  -8.239  0.58 15.69           O
ATOM   3691  C2 AEDO C   1      21.710  -4.102  -8.630  0.58 15.43           C
ATOM   3692  C1 AEDO C   1      20.965  -2.841  -8.282  0.58 16.78           C
ATOM   3693  O1 AEDO C   1      21.111  -2.587  -6.901  0.58 19.33           O
ATOM   3687  I  BIOD C   1      21.798  -3.596  -7.915  0.42 34.88           I
```

Refinement with twinned data

- Two steps to perform twin refinement:

  - run phenix.xtriage to get twin operator (twin law):

```
% phenix.xtriage data.mtz
```

  - run phenix.refine:

```
% phenix.refine model.pdb data.mtz twin_law="-h-k,k,-l"
```

- Taking twinning into account makes (big) difference:

Interleukin mutant (PDB code: 1I2h)

	R/R-free (%)
PHENIX (no twinning):	24.9 / 27.4
PHENIX (twin refinement):	15.3 / 19.2

Hydrogen atoms in refinement

- **Some facts about hydrogen atoms:**
  - H atoms are not visible in X-ray maps at “typical macromolecular” resolutions, that is ~1Å and lower. This is because:
    - H atom is a weak scatterer (much weaker than C, N or O atoms)
    - models contain too much noise so the H contribution is hidden in it. Ideally (nearly error free model) one would see H even at ~2Å resolution.
  - Some or most of H atoms can be seen in maps at ultra-high resolutions (~1Å and higher):
    - The resolution itself is not the sufficient condition to see H: the noise level should be low (small *R*-factor).
  - Hydrogen atoms constitute nearly 50% of the total atoms in protein structures. Typical example: Fab structure (PDB code: 1f8t): 3593 non-H atoms, 3269 H atoms.
  - Since H is a weak scatterer, it mostly contributes to the low resolution (and not to the high!). The reason why we see H atoms only in structures corresponding to high resolution data is because these structures are typically accurate enough and complete so the noise level is small (small *R*-factor).

Hydrogen atoms in refinement

- phenix.refine: handling H atoms at ***any*** resolution:

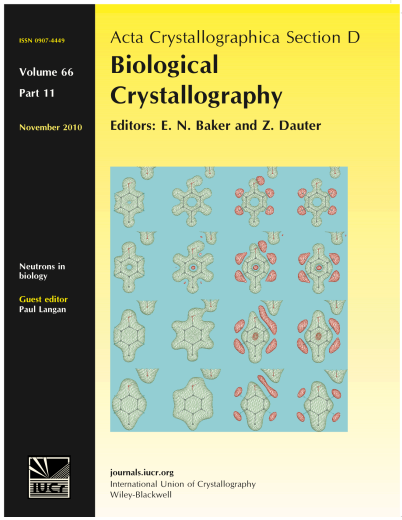
  - Riding model (low-high resolution)
  - Individual atoms (ultrahigh resolution or neutron data)
  - Account for scattering contribution or just use to improve the geometry

- Using the H atoms in refinement:

  - Improve R-factors
  - Improve model geometry (remove bad clashes)
  - Model residual density at high resolution or in neutron maps

- Example: automatic re-refinement of 1000 PDB models with and without H:

pdb	resolution	Rfree(no H) – Rfree(with H)
1akg	1.1	1.9
1byp	1.75	1.41
1dkp	2.3	0.93
1rgv	2.9	0.50



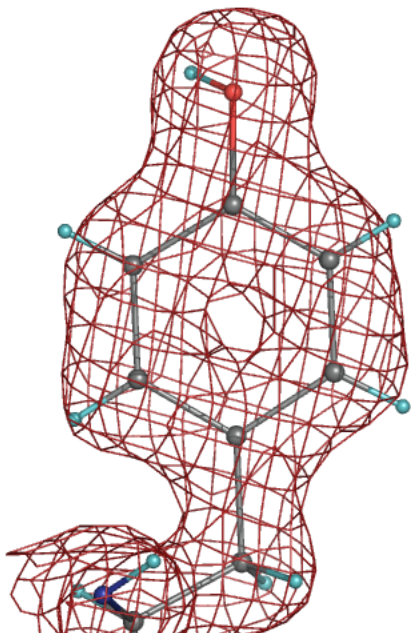
Review and latest developments:
Afonine, et al. (2010). Joint X-ray and neutron refinement with phenix.refine. Acta Cryst. D66, 1153-1163.

Refinement using X-ray and Neutron diffraction data

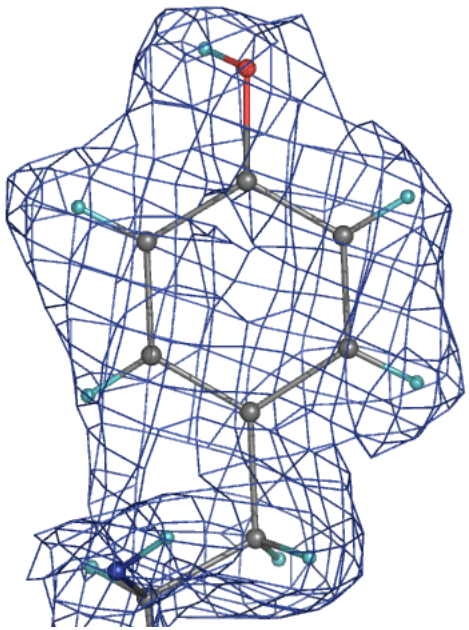
- Different techniques – different information (neutron maps show hydrogen atoms)

2mFo-DFc maps

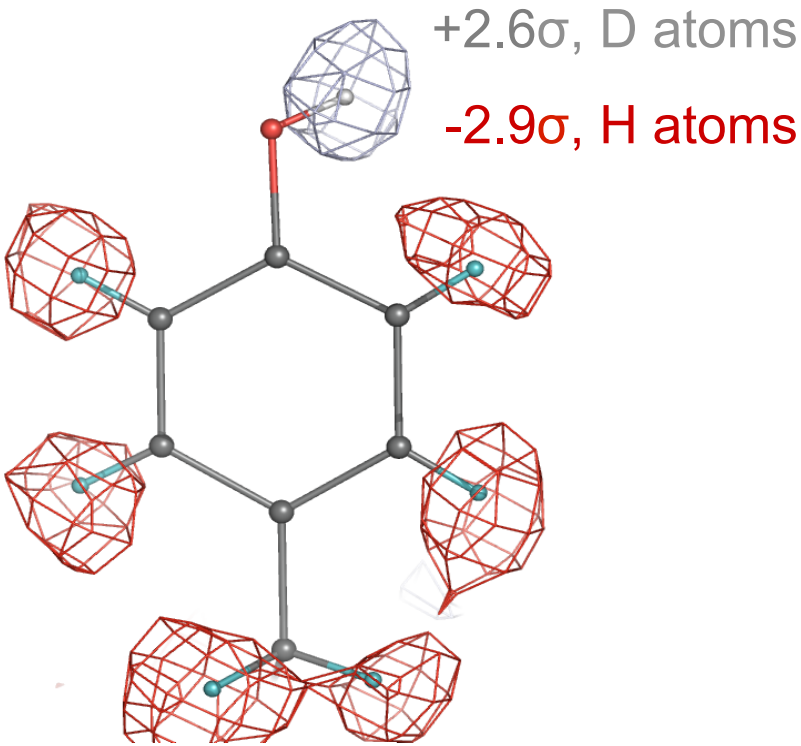
X-ray (1.8 Å)



Neutron (2.2 Å)



Fo-Fc, (H-, D-omit **neutron map**), **1.6 Å** resolution

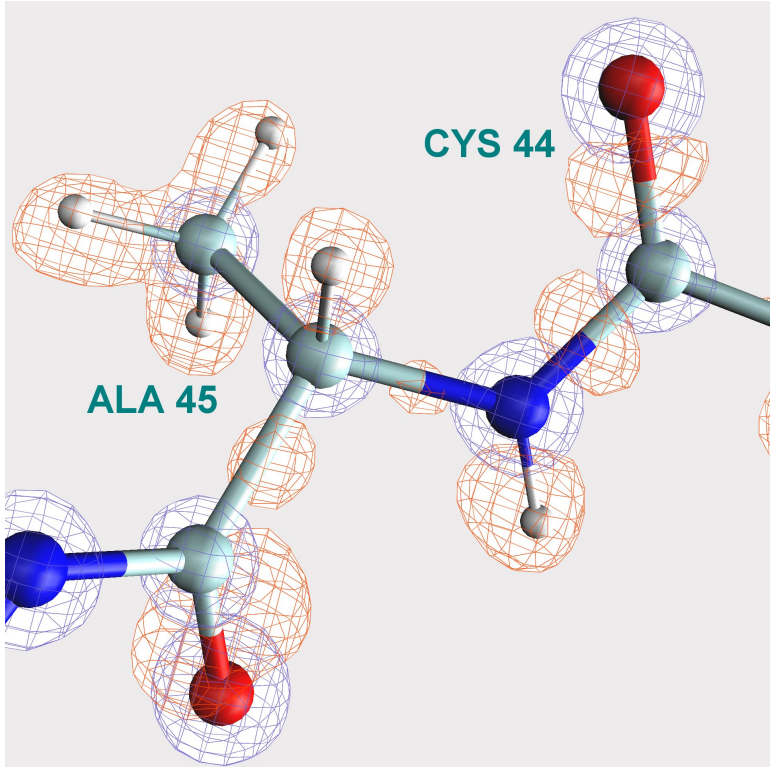


- *phenix.refine* can refine a structure against neutron data or both X-ray and neutron simultaneously

Refinement at subatomic resolution

~340 structures in PDB at resolution higher than 1.0 Å

Aldose Reductase (0.66 Å resolution)



Fo-Fc (orange) 2Fo-Fc (blue)

✓ phenix.refine has unique set of tools to correctly refine such structures

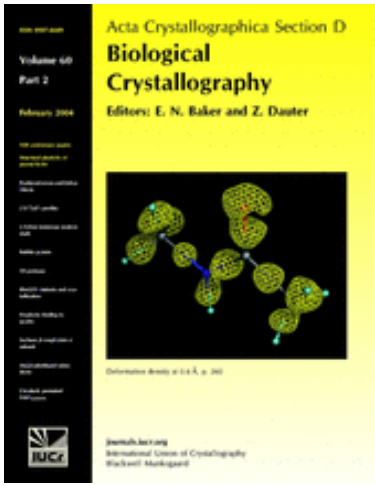
Modeling at subatomic resolution: IAS model

- Basics of IAS model:

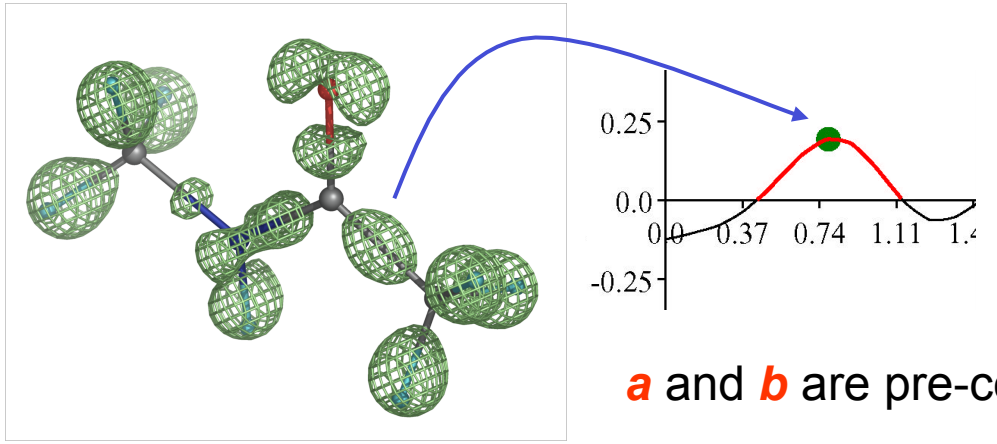
  Afonine et al, Acta Cryst. D60 (2004)

- First practical examples of implementation and use in PHENIX:

  Afonine et al, Acta Cryst. D63, 1194-1197 (2007)



IAS modeling in PHENIX



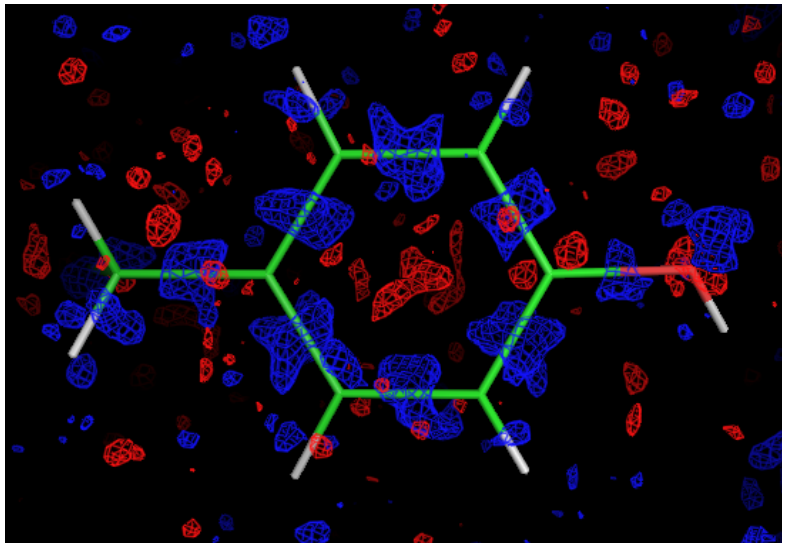
Simple Gaussian is good enough:

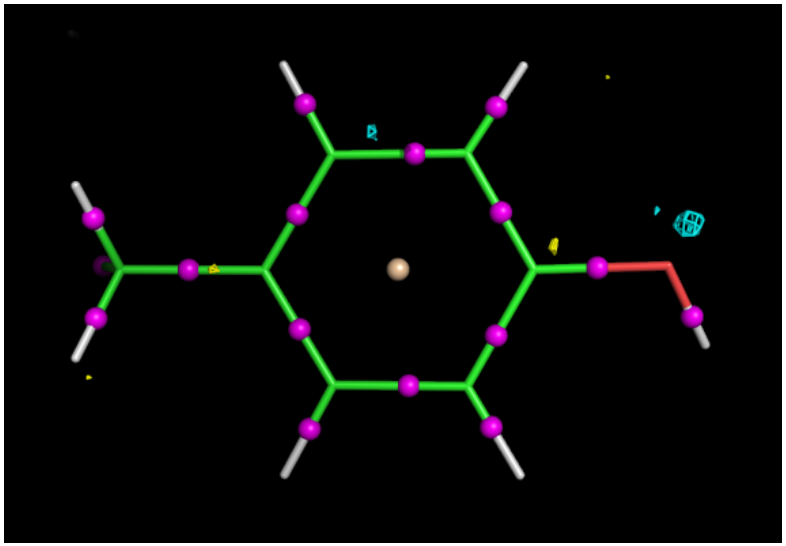
$$f_{bond\_scatterer}(\mathbf{s}) = \boldsymbol{a}\ \exp(\ \boldsymbol{b}\ \mathbf{s}^2)$$

a and ***b*** are pre-computed library for most bond types

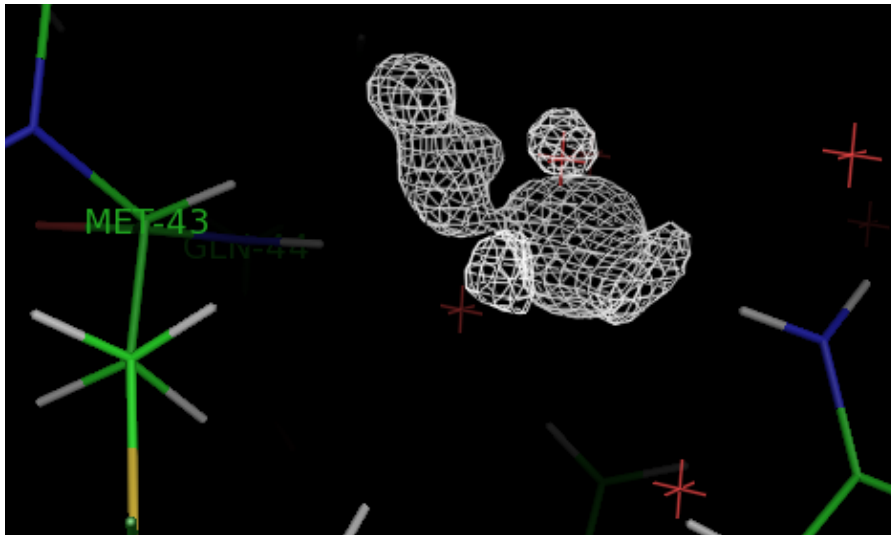
IAS modeling: benefits

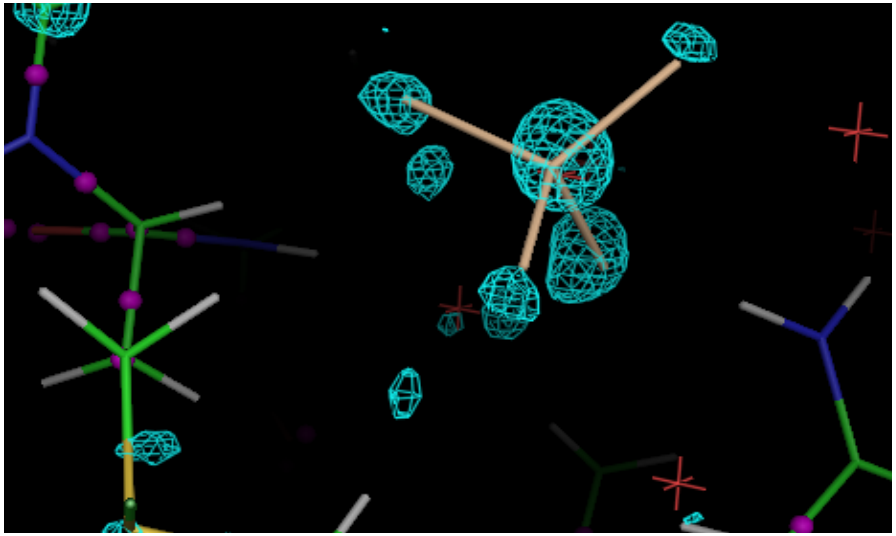
- Improve maps: reduce noise. Before (left) and after (right) adding of IAS.





- Find new features: originally wrong water (left) replaced with SO4 ion (right) clearly suggested by improved map after adding IAS



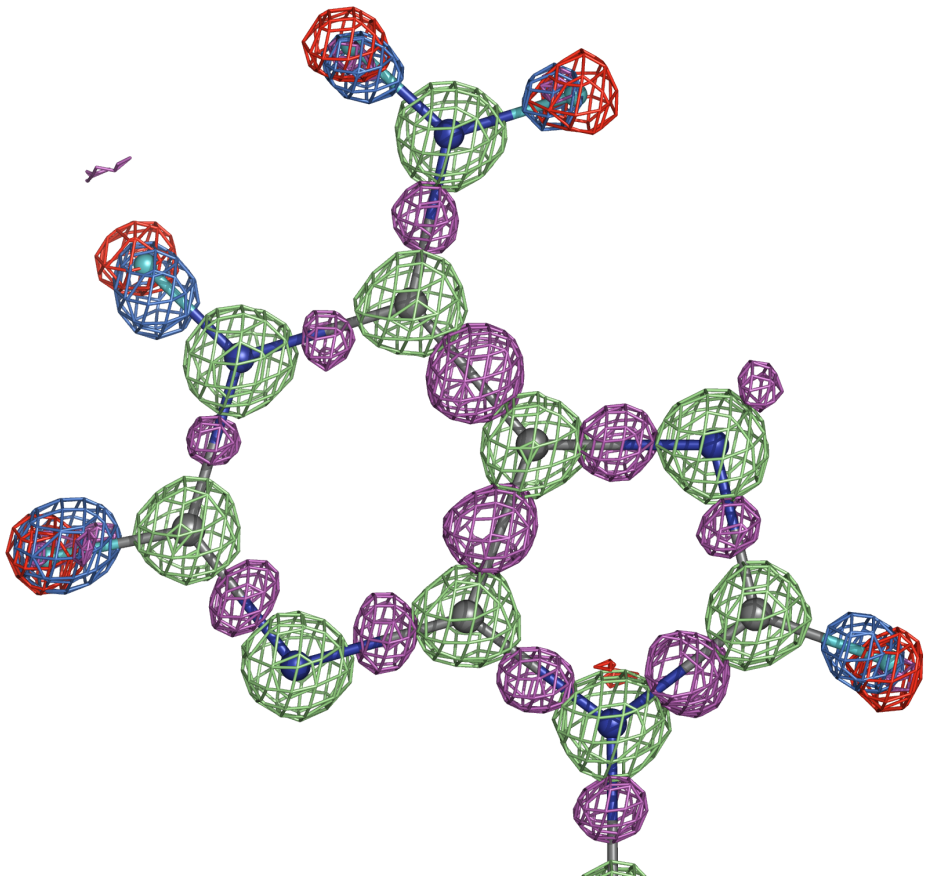


X-ray and Neutron Crystallography: Complimentary Methods

- Still complimentary even at subatomic resolution (NAD structure)

Neutron 2mFo-DFc map at 0.65 Å resolution, ±2.4σ, green (positive), red (negative)

X-ray mFo-DFc map at 0.6 Å resolution, blue: H omit, 5σ, magenta: 2.8σ all atoms included



Running phenix.refine (command line)

Model refinement

- Designed to be very easy to use

```
phenix.refine model.pdb data.hkl [parameters]
```

Some basic examples of running phenix.refine from the command line

- Refinement of individual coordinates, B-factors, and occupancies for some atoms:

```
phenix.refine model.pdb data.hkl
```

- Add water picking and Simulated Annealing to default run above:

```
phenix.refine model.pdb data.hkl simulated_annealing=true
ordered_solvent=true
```

- Refinement of individual coordinates and B-factors using neutron data:

```
phenix.refine model.pdb data.hkl
main.scattering_dictionary=neutron
```

- To see all parameters (a few hundreds):

```
phenix.refine --show_defaults=all
```

Running phenix.refine

```
% phenix.refine model.pdb data.hkl parameter_file
```

where **parameter_file** contains following lines:

```
refinement.main {
   high_resolution = 2.0
   low_resolution = 15.0
   simulated_annealing = True
   ordered_solvent = True
   number_of_macro_cycles = 5
}
refinement.refine.adp {
    tls = chain A
    tls = chain B
}
```

For typing enthusiasts, the equivalent command line run would be:

```
% phenix.refine model.pdb data.hkl xray_data.high_resolution=2
xray_data.low_resolution=15 simmulated_annealing=true
ordered_solvent=True adp.tls="chain A" adp.tls="chain B"
main.number_of_macro_cycles=5
```

Typical way of phenix.refine run from the command line

1. Get the file with all parameters:

```
% phenix.refine --show-defaults=all > parameter_file
```

2. Edit the file `parameter_file`:
   - Remove all parameters that you are not planning to change (make sure to have all { } matched)
   - Change the rest of parameters

3. Run phenix.refine as following:

```
% phenix.refine model.pdb data.hkl parameter_file
```

or (If `model.pdb` and `data.hkl` are included into `parameter_file` file)

```
% phenix.refine parameter_file
```

Useful tip: to compare the set of parameters in your `parameter_file` file against the set of all default parameters, type:

```
% phenix.refine --diff-params parameter_file
```

Some refinement runs require two steps: hydrogens and ligands

- When running: `% phenix.refine model.pdb data.hkl`

each item in model.pdb is matched against the CCP4 Monomer Library to extract the topology and parameters and to automatically build corresponding restraints.

- If model.pdb contains an item not available in CCP4 Monomer Library, e.g. a novel ligand, use ReadySet! program to generate topology and parameter definitions for refinement:

```
% phenix.ready_set model.pdb
```

This will produce the file LIG.cif and updated PDB file `model.updated.pdb` with all H atoms added which can be used for refinement:

```
% phenix.refine model.pdb data.hkl LIG.cif
```

Some refinement runs require two steps: twinning

- Two steps to perform twin refinement:

  - run *phenix.xtriage* to get twin operator (twin law):

```
% phenix.xtriage data.mtz
```

  - run *phenix.refine*:

```
% phenix.refine model.pdb data.mtz twin_law="-h-k,k,-l"
```

Model refinement - output

- Input command

```
phenix.refine model.pdb data.mtz [parameters]
```

- Output files

`model_refine_001.eff`	summary of all input parameters
`model_refine_001.geo`	summary of all restraints used
`model_refine_001.log`	complete information about refinement
`model_refine_001.pdb`	refined structure
`model_refine_001.mtz`	Fourier map coefficients, Fcalc, etc.
`model_refine_002.def`	parameters for the next run

If data file is not in MTZ format, or there are multiple data files at input (example: one with Fobs and the other one with free-R flags), then phenix.refine will combine them into one MTZ data file called: `model_data.mtz` and this file should be used in all subsequent runs.

MTZ phenix.refine

A run

```
phenix.refine model.pdb data.mtz
```

does not output anymore

```
model_001_map_coeffs.mtz
```

Instead, it always outputs a MTZ file

```
model_001.mtz
```

that contains:

MTZ phenix.refine

```
 Number of datasets: 4
  Dataset 1:
    Name: Original-experimental-data
(...)
    label           #valid  %valid  min     max type
    H                17129 100.00% 0.00   18.00 H: index h,k,l
    K                17129 100.00% 0.00   35.00 H: index h,k,l
    L                17129 100.00% 0.00   45.00 H: index h,k,l
    I-obs            16775  97.93% 0.00 1250.45 K: I
    SIGI-obs         16775  97.93% 0.00   46.36 M: standard deviation
  Dataset 2:
    Name: Experimental-data-used-in-refinement
(...)
    label                #valid %valid  min   max type
    F-obs-filtered        16464 96.12% 1.15 35.36 G: F
    SIGF-obs-filtered     16464 96.12% 0.05  2.96 L: standard deviation
  Dataset 3:
    Name: Model-structure-factors-(bulk-solvent-and-all-scales-included)
(...)
    label         #valid %valid     min    max type
    F-model(+)     16464 96.12%    0.00  83.91 G: F(+) or F(-)
    PHIF-model(+)  16464 96.12% -180.00 180.00 P: phase angle in degrees
    F-model(-)     14382 83.96%    0.02  91.93 G: F(+) or F(-)
    PHIF-model(-)  14382 83.96% -179.97 179.95 P: phase angle in degrees
  Dataset 4:
    Name: Fourier-map-coefficients
(...)
    label             #valid  %valid     min    max type
    2FOFCWT            17129 100.00%    0.00  43.21 F: amplitude
    PH2FOFCWT          17129 100.00% -180.00 180.00 P: phase angle in degrees
    2FOFCWT_no_fill    16657  97.24%    0.00  41.96 F: amplitude
    PH2FOFCWT_no_fill  16657  97.24% -180.00 180.00 P: phase angle in degrees
    FOFCWT             16657  97.24%    0.00  58.25 F: amplitude
    PHFOFCWT           16657  97.24% -180.00 180.00 P: phase angle in degrees
    ANOM               14189  82.84%    0.00   2.19 F: amplitude
    PANOM              14189  82.84% -180.00 179.96 P: phase angle in degrees
```

Example of a complex refinement run

- Do the following:
  - refine individual coordinates for all atoms using minimization and Simulated Annealing
  - refine coordinates of three rigid body groups:
    - chain A
    - chain B and chain C
    - chain D
  - individual anisotropic ADP for all Uranium atoms
  - individual isotropic ADP for all other atoms
  - three TLS groups:
    - atoms in residues from 1 to 300 of chain A and whole chain B
    - atoms from 301 to 500 in chain A
    - whole chain D
  - update water during refinement
  - use NCS in refinement
  - output everything into a files with prefix *test*

```
% phenix.refine model.pdb data.hkl parameters.eff
```

where **`parameters.eff`** contains following lines: see next slide…

Example of a complex parameter file

```
refinement {
  output {
    prefix = test
  }
  refine {
    strategy=*individual_sites individual_sites_real_space *rigid_body \
             *individual_adp group_adp *tls *occupancies group_anomalous
    sites {
      rigid_body = chain A
      rigid_body = chain B or chain C
      rigid_body = chain D
    }
    adp {
      individual {
        isotropic = not (element U)
        anisotropic = element U
      }
      tls = chain A and resseq 1:300 or chain B
      tls = chain A and resseq 301:500
      tls = chain D
    }}
  main {
    simulated_annealing = True
    ordered_solvent = True
    ncs = True
}}
```

Reporting bugs, problems, asking questions

- **Something didn't work as expected?... program crashed?... missing feature?...**

  Not Good: silently give up and run away looking for alternative software (or write your own program).

  Good: report us a problem, ask a question, request a feature (explain why it's good to have), ask for help.

- **Reporting a bug:**

  Not good: "Hi! PHENIX crashed, I don't know what to do."

  Good: "Hi! PHENIX crashed. Here are:
  1) PHENIX version;
  2) Command and parameters I used;
  3) Input and output files (at least logs)."

Subscribe to PHENIX bulletin board: www.phenix-online.org

Phenix

http://www.phenix-online.org/

Apple Yahoo! Google Maps YouTube Wikipedia News (1731) Popular

Phenix

NEW Development release of PHENIX version 1.4 now available

*Python-based **H**ierarchical **EN**vironment for **I**ntegrated **X**tallography*

PHENIX is a new software suite for the automated determination of macromolecular structures using X-ray crystallography and other methods.

Citing PHENIX:
PHENIX: building new software for automated crystallographic structure determination P.D. Adams, R.W. Grosse-Kunstleve, L.-W. Hung, T.R. Ioerger, A.J. McCoy, N.W. Moriarty, R.J. Read, J.C. Sacchettini, N.K. Sauter and T.C. Terwilliger. *Acta Cryst.* D58, 1948-1954 (2002)

Download the latest development release (1.4-3) [First request download password]

Help: FAQ Mailing List Subscription List Archives Report a Bug Email for Help

Using PHENIX (release 1.4-3): **Full Documentation** **PDF**

- Assessing data quality with phenix.xtriage
- Automated structure solution with AutoSol
- Automated molecular replacement with AutoMR
- Automated model building and rebuilding with AutoBuild
- Automated ligand fitting with LigandFit
- Structure refinement with phenix.refine
- Generation of ligand coordinates and restraints with elbow
- The PHENIX Graphical User Interface

Documentation for 1.3-final

The PHENIX system also includes SOLVE/RESOLVE, Phaser, Textal, the CCI Applications (phenix.xtriage, phenix.refine, elbow and many more), components from Molprobity, and the Computational Crystallography Toolbox in a Python framework.

Funding for PHENIX: Protein Structure Initiative (NIH General Medical Sciences)

Introduction to PHENIX
Using PHENIX
Platforms
Licensing
Download
Recent Changes
Publications
Presentations
Computational Crystallography Toolbox
Contact Us
The PHENIX Team
Acknowledgments
Intranet

The PHENIX Industrial Consortium

For-profit groups can obtain access to PHENIX through a Consortium agreement. This provides a license to use PHENIX and research funds to develop new features in PHENIX tailored to the needs of commercial users.

Information
Members
Download
Contact Us

Groups developing PHENIX:

Paul Adams *Randy Read* — Berkeley Lab, University of Cambridge

Jane & Dave Richardson — Duke

Tom Terwilliger — Los Alamos National Laboratory

Tom Ioerger & *Jim Sacchettini* — Texas A&M

Privacy and Security Notice

About this website

This presentation (PDF file) and much more

www.phenix-online.org