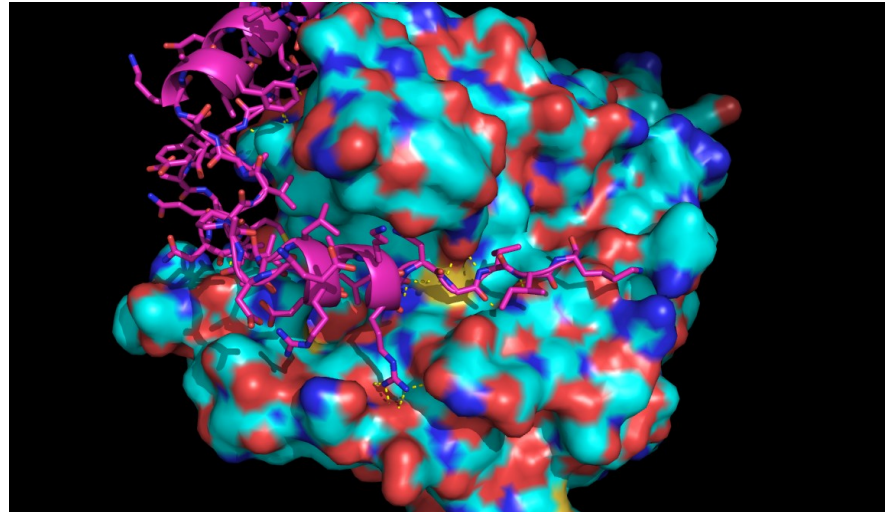


Introduction to CrystFEL

Data processing for serial crystallography



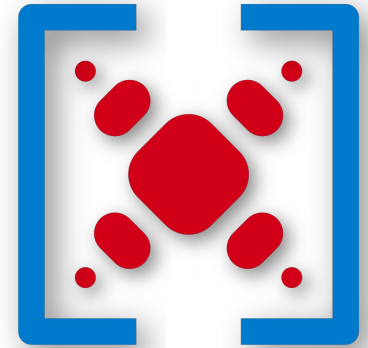
Thomas White
BioXFEL Data Analysis Workshop
Berkeley, 21 Aug 2014

Contributors to CrystFEL

- > Led by me, Thomas White (Center for Free-Electron Laser Science, DESY Hamburg).
- > Code contributions from:
 - Rick Kirian (ASU, now CFEL), Nadia Zatsepin (ASU).
 - Karol Nass (CFEL, now MPI-MF Heidelberg), Andrew Aquila, Andrew Martin, Lorenzo Galli, Kenneth Beyerlein, Chun Hong Yoon, Cornelius Gati, Anton Barty, Fedor Chervinskii, Valerio Mariani (CFEL).
 - Takanori Nakane (MRC-LMB).
 - Wolfgang Brehm (U Konstanz).
- > Testing/feedback from the above and many additional people, especially:
 - Francesco Stellato, Markus Metz, Dominik Oberthur, Oleksandr Yefanov (CFEL).
 - Shibom Basu, Haiguang Liu (ASU).
- > Words of wisdom from many people, in particular:
 - Henry Chapman (CFEL), Petra Fromme (CFEL), James Holton (ALS/UC Berkeley), John Spence (ASU/LBL), Kay Diederichs (U Konstanz).

What is CrystFEL?

- > CrystFEL: data processing for “snapshot serial” crystallography (usually using an FEL, but not necessarily).
- > Development since late 2009. Currently at version 0.5.3a.
- > Licence: GPLv3 (*free and open source* software)
- > User-oriented development.
- > <http://www.desy.de/~twhite/crystfel>



CrystFEL

Welcome to the **CrystFEL** page!

CrystFEL is a suite of programs for processing diffraction data acquired “serially” in a “snapshot” manner, such as when using the technique of **Serial Femtosecond Crystallography (SFX)** with a free-electron laser source. CrystFEL comprises programs for indexing and integrating diffraction patterns, scaling and merging intensities, simulating patterns, calculating figures of merit for the data and visualising the results. Supporting scripts are provided to help at all stages, including importing data into **CCP4** for further processing.

The primary citation for CrystFEL is as follows:

T. A. White, R. A. Kirian, A. V. Martin, A. Aquila, K. Nass, A. Barty and H. N. Chapman. “CrystFEL: a software suite for snapshot serial crystallography”. *J. Appl. Cryst.* **45**, p335–341. doi:10.1107/S0021889812002312 — [Download PDF](#) — [Article on IUCr website](#).

Recent News

- **10th January 2013:** CrystFEL version 0.4.3 released. See the [download page](#) or the [changes page](#) for more information.
- **3rd October 2012:** CrystFEL version 0.4.2 released.
- **29th August 2012:** CrystFEL version 0.4.1 released.
- **31st July 2012:** CrystFEL version 0.4.0 released. See the [release notes](#) and the [download page](#) for more information.
- **20th June 2012:** Lysozyme test images available for download! Are you looking for some test data to work with? The images from a [recent Science article](#) have been made available via the Coherent Imaging Data Bank. [Download here!](#)
- **14th March 2012:** First public release of CrystFEL!

Acknowledgements

The development of CrystFEL is led by Thomas White at the [Center for Free-Electron Laser Science](#) at the [Deutsches Elektronen-Synchrotron DESY](#) in Hamburg, Germany. DESY is a research centre of the [Helmholtz-Gemeinschaft](#). See the [contact page](#) for more contact details.

Code has been contributed to CrystFEL by: Rick Kirian, Andrew Aquila, Andrew Martin, Lorenzo Galli, Kenneth Bayerlein, Chun Hong Yoon and Nadia Zatspein. It has been thoroughly tested by Karol Nass, Francesco Stellato, Linda Johansson, David Amund, Mark Hunter and many others. The design of CrystFEL and its algorithms has been influenced by Henry Chapman, Anton Barty, Petra Fromme, James Holton and John Spence.

You can read more information about some of the algorithms used in CrystFEL in the following article: Kirian et al., *Optics Express* **18** (2010) p5713. doi:10.1364/OE.18.005713.

“Official” citation: White et al., *J. Appl. Cryst.* **45** (2012) p335

What do you get? Core programs

- > **indexamajig** Bulk indexing and integration of patterns
- > **ambigator** Resolve indexing ambiguities
- > **process_hkl** Merge using “Monte Carlo” technique
- > **hdfsee** View images in HDF5 format (with geometry)
- > **cell_explorer** Examine unit cell parameter distributions
- > **compare_hkl** Compare merged reflection data
- > **check_hkl** Evaluate merged reflection data
- > **pattern_sim** Simulate diffraction patterns
- > **partial_sim** Simulate partial reflection intensities
- > **get_hkl** Reflection data “Swiss army knife”
- > **render_hkl** Render plane sections of reciprocal space
- > **partialator** Merge using partialities and post-refinement

What do you get? Documentation

- > Manual pages: `$ man indexamajig`
- > Help messages: `$ indexamajig --help`
- > Website, particularly: tutorial, best practice, FAQ

```
18:45:04 cfe4204w ~$ ./project % indexamajig -help
Syntax: indexamajig [options]

Process and index FEL diffraction images.

-h, --help          Display this help message.
-v, --version       Print CrystFEL version number and exit.

-i, --input=<filename> Specify file containing list of images to process.
-o, --output=<filename> Write output stream to this file. '-' for stdout.
                        Default: indexamajig.stream

--indexing=<methods> Use '<methods>' for indexing. Provide one or more
                    methods separated by commas.
                    Set 'no_indexamajig' for details.
-g, --geometry=<file> Get detector geometry from file.
-b, --beam=<file>      Get beam parameters from file (provides nominal
                    wavelength value if no parshot value is found in
                    the HDFS files.
-p, --pdb=<file>      RDB file from which to get the unit cell to match.
                    Default: 'molecule.pdb'
--beamname           Remove the directory parts of the filenames.
-x, --prefix=<mp>     Prefix filenames from input file with <mp>.
--peaks=<method>      Use '<method>' for finding peaks. Choose from:
                    'zarf' - use Zeffner (2000) gradient detection.
                    'hdfs' - Get from a table in HDFS file.
                    'hdfs-peaks-mp' Find peaks table in HDFS file here.
                    'integration=with' Perform final pattern integration using <with>.

For more control over the process, you might need:

--tolerance=<tol>    Set the tolerances for cell comparison.
                    Default: 5.5, 3.1, 5.
--filter-noise       Apply an aggressive noise filter which sets all
                    pixels in each 2d region to zero if any of them
                    have negative values. Intensity measurement will
                    be performed on the image as it was before this.
--median-filter=<mp> Apply a median filter to the image. Intensity
                    measurement will be performed on the image as it
                    was before this. The scan length of the median
                    filter box will be 200x1. Default: 0 (no filter)
```

- > Mailing list for announcements

CrystFEL

Home
About CrystFEL
Download
Screenshots
Downloads
Citations

Documentation
Overview
Installation
FAQ
Tutorial
Changes
API Reference

Contact
Legal Information

DESY Homepage
CFEL Homepage
DESYXCFEL

Welcome to the CrystFEL page!

CrystFEL is a suite of programs for processing diffraction data acquired "on-site" in a "snapshot" manner, such as when using the technique of Serial Femtosecond Crystallography (SFX) with free-electron laser sources. CrystFEL comprises programs for indexing and integrating diffraction patterns, scaling and merging intensities, simulating patterns, calculating figures of merit for the data and visualizing the results. Supporting scripts are provided to help at all stages, including importing data into CCP4 for further processing.

The primary citation for CrystFEL is as follows:

T. A. White, R. A. Kian, A. V. Martin, A. Aquila, K. Nass, A. Barty and H. N. Chapman. "CrystFEL: a software suite for snapshot serial crystallography." *J. Appl. Cryst.* **45**, p335-341. doi:10.1107/S0021889812022021 [CrossRef] [PubMed] [DOI]

Recent News

- 10th January 2013: CrystFEL version 0.4.3 released. See the [download page](#) or the [changes page](#) for more information.
- 3rd October 2012: CrystFEL version 0.4.2 released.
- 29th August 2012: CrystFEL version 0.4.1 released.
- 21st July 2012: CrystFEL version 0.4.0 released. See the [release notes](#) and the [download page](#) for more information.
- 20th June 2012: Lycopodium test images available for download!
- 14th March 2012: First public release of CrystFEL!

Acknowledgements

The development of CrystFEL is led by Thomas White at the [Center for Free-Electron Laser Science](#) at the [Deutsches Elektronen-Synchrotron DESY](#) in Hamburg, Germany. DESY is a research centre of the [Helmholtz Gemeinschaft](#). See the [acknowledgements](#) for more contact details.

Code has been contributed to CrystFEL by Rick Kian, Andrew Aquila, Andrew Martin, Lorenzo Gall, Kenneth Beyreiter, Chun Hong Yoon and Nadia Zabrejko. It has been thoroughly tested by Karol Nass, Francesco Stellato, Linda Johansson, David Amund, Mark Huxler and many others. The design of CrystFEL and its algorithms has been influenced by Henry Chapman, Arlon Barty, Peter Fromme, James John and John Spence.

You can read more information about some of the algorithms used in CrystFEL in the following article: Kian et al., *Optics Express* **18** (2010) p2713. doi:10.1364/OE.18.02713

```
INDEXAMAJIG(1)                                INDEXAMAJIG(1)

NAME
  indexamajig - bulk indexing and data reduction program

SYNOPSIS
  indexamajig -i filename -o output_stream -g detector-geom -b beamline-beam --peaks-method --indexing-method (options) ...
  indexamajig --help

DESCRIPTION
  indexamajig takes a list of diffraction snapshots from crystals in random orientations and attempts to find peaks, index and integrate each one. The output is a list of diffraction image files in HDFS format and some auxiliary files and parameters. The output is a long text file ('stream') containing the results from each image in turn.

  For minimal basic use, you need to provide the list of diffraction patterns, the method which will be used to index, a file describing the geometry of the detector, and a PDB file which contains the unit cell which will be used for the indexing. Here is what the minimal use might look like on the command line:

  indexamajig mypatterns.lst -i 10 -g mygeometry.geom --indexing=mosflm.dirax --peaks=hdfs -b mycell-beam -o test.stream -p mycell.pdb

  More typical use includes all the above, but might also include extra parameters to modify the behaviour. The 'HDFS' files might be in some folder a long way from the current directory, so you might want to specify a full pathname to be added in front of each filename. You'll probably want to run more than one indexing job at a time (i.e. xps).

  See man crystfel\_geometry for information about how to create a geometry description file and a beam parameters file.

PEAK DETECTION
  You can control the peak detection on the command line. Firstly, you can choose the peak detection method using --peaks=method. Currently, two values for 'method' are available. --peaks=hdfs will take the peak locations from the HDFS file. It expects a two dimensional array by default at processing/integration/peaks, whose size in the first dimension equals the number of peaks and whose size in the second dimension is three. The first two columns contain the fast scan and slow scan coordinates, the third contains the intensity. However, the intensity will be ignored since the pattern will always be re-integrated, using the unit cell provided by the user, on the basis of the peaks. You can also specify where to find this table inside each HDFS file using --hdfs-peaks=tbl.

  If you use --peaks=zarf, indexamajig will use a simple gradient search algorithm Zeffner (2000). You can control the overall threshold and square gradient for finding a peak using --threshold and --minsq-gradient. The threshold has arbitrary units matching the pixel values in the data, and the minimum gradient has the equivalent squared units. Peaks will be rejected if the 'foot point' is further away from the 'summit' of the peak by more than the inner integration radius (see below). They will also be rejected if the peak is closer than twice the inner integration radius from another peak.

  You can suppress peak detection altogether for a panel in the geometry file by specifying the "no_index" value for the panel as 'noh-zero'.

INDEXING METHODS
  You can choose between a variety of indexing methods. You can choose more than one method, in which case each method will be used in turn until one of them appears that the pattern has been successfully indexed. Choose from:
  Manual pages: indexamajig(1) line 4 [CrossRef] [PubMed] [DOI]
```



What do you get? Scripts

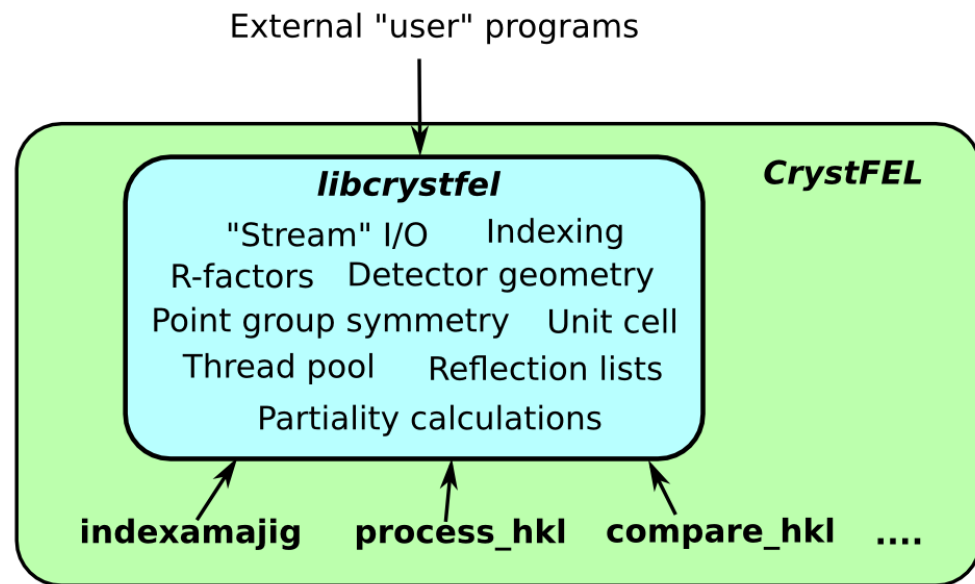
- > Auxiliary scripts for plotting graphs, collating results etc.
- > Difference from core programs: you're meant to copy them to your working directory and modify them

```
1#!/usr/bin/perl -w
2
3use strict;
4use File::Basename;
5
6my $args = join(" ", splice(@ARGV, 1, scalar(@ARGV)-1));
7if ( !($args eq "") ) {
8    printf("Extra arguments for hdfsee: %s\n", $args);
9} else {
10    # Default arguments - feel free to override!
11    $args = "--binning=2 --int-boost=10";
12    printf("Using default arguments for hdfsee: %s\n", $args);
13}
14
15open(FH, $ARGV[0]);
16open(TMP, "> list.tmp");
17
18my $in_image = 0;
```

What do you get? libcrystfel

- > libcrystfel: shared library exposing high-level functions, e.g. “index this pattern” or “calculate these partialities”.
- > Possible application: use CrystFEL for online data processing within a facility's monitoring software.

> API docs on website



Where can you get it?

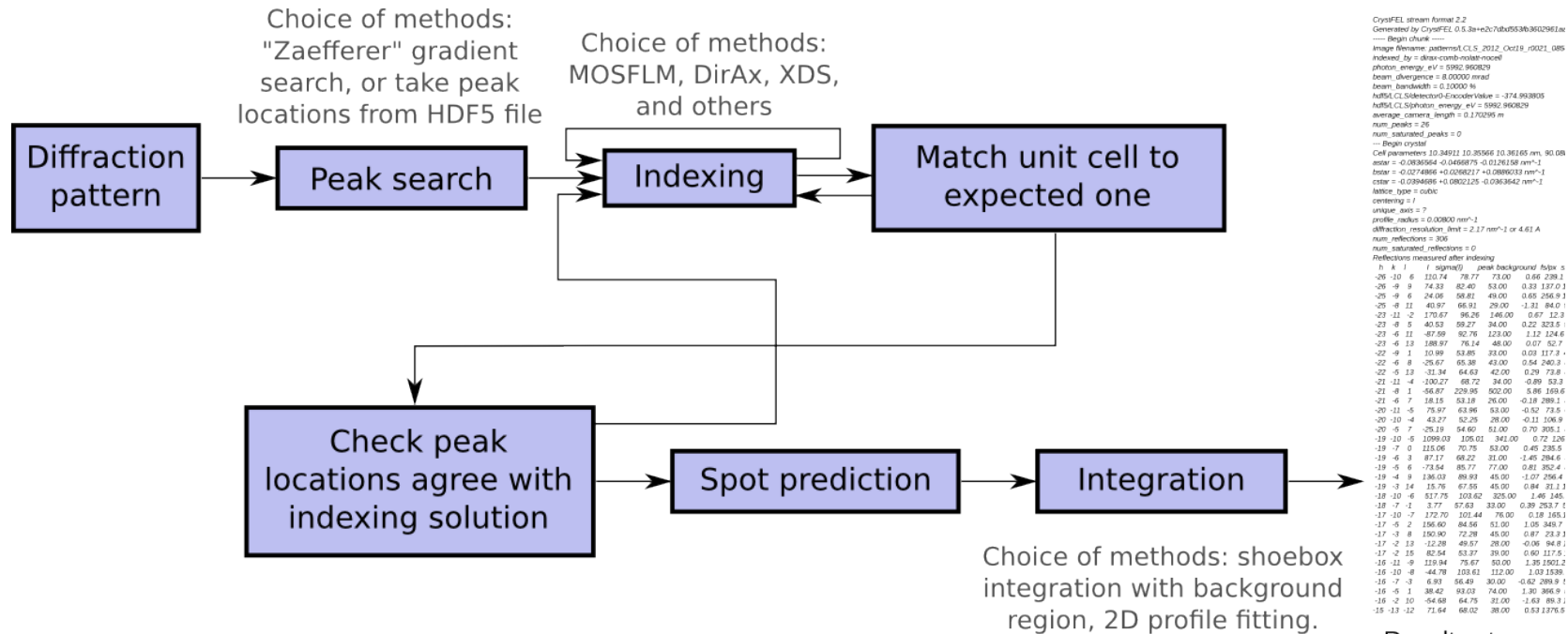
- > From the website: <http://www.desy.de/~twhite/crystfel>
(or just type “crystfel” into your favourite search engine)
- > “Release” versions: tested with a series of regression tests
- > Git repository: very latest code, maybe broken, maybe awesome, slightly harder to install (need autotools installed).
- > Packaged snapshots: latest code,
- > Ubuntu, OpenSUSE and Fedora packages. MacPort. Soon: Debian.

Where can you get it?

- > GNU General Public License (GPL):
“free” as in “freedom” (also as in “free beer”)
- > Freedom to use the software *for any purpose* (even commercially).
- > Freedom to study how the software works.
- > Freedom to change the software, e.g. to fix bugs or add features.
- > Freedom to share the software, even a modified version.
- > No license forms to sign, send etc.
- > The GPL ensures that these rights are never restricted.



Pipeline in “indexamajig”



- > Input: HDF5 (very common scientific format, many language bindings)
- > Note: choice of methods for **peak search**, **indexing** and **integration**.
- > Output: plain text “stream” - don't be afraid to look inside! (tip: use “less”, not “gedit” - file can be very large)

Peak search methods

- > `indexamajig [.....] --peaks=hdf5 [.....]`
..... means “get peak list from HDF5 file”.
- > `indexamajig [.....] --peaks=zaef [.....]`
..... means “use internal peak search algorithm”.

Indexing methods

> `indexamajig [...] --indexing=method1,method2,... [.....]`
List of indexing methods will be tried in order.

> Example method:

`mosflm-raw-latt`

Invoke mosflm to index the pattern using the peaks from the peak search

Skip the step where the unit cell parameters would be checked against the reference

Take the Bravais lattice information from the reference into account, e.g. “look for tetragonal P cells only”

Indexing methods

> `indexamajig [...] --indexing=method1,method2,... [.....]`
List of indexing methods will be tried in order.

> Example method:

mosflm-axes-latt

Invoke mosflm to index the pattern using the peaks from the peak search

Check the unit cell parameters against the reference

Take the Bravais lattice information from the reference into account, e.g. “look for tetragonal P cells only”

Indexing methods

> `indexamajig [...] --indexing=method1,method2,.. [.....]`
List of indexing methods will be tried in order.

> Example method:

`mosflm-raw-nolatt`

Invoke mosflm to
index the pattern
using the peaks from
the peak search

Do not check the
unit cell parameters

Do not use
Bravais lattice
information

Indexing methods

> `indexamajig [...] --indexing=method1,method2,... [.....]`
List of indexing methods will be tried in order.

> Example method:

`dirax-comb-nolatt`

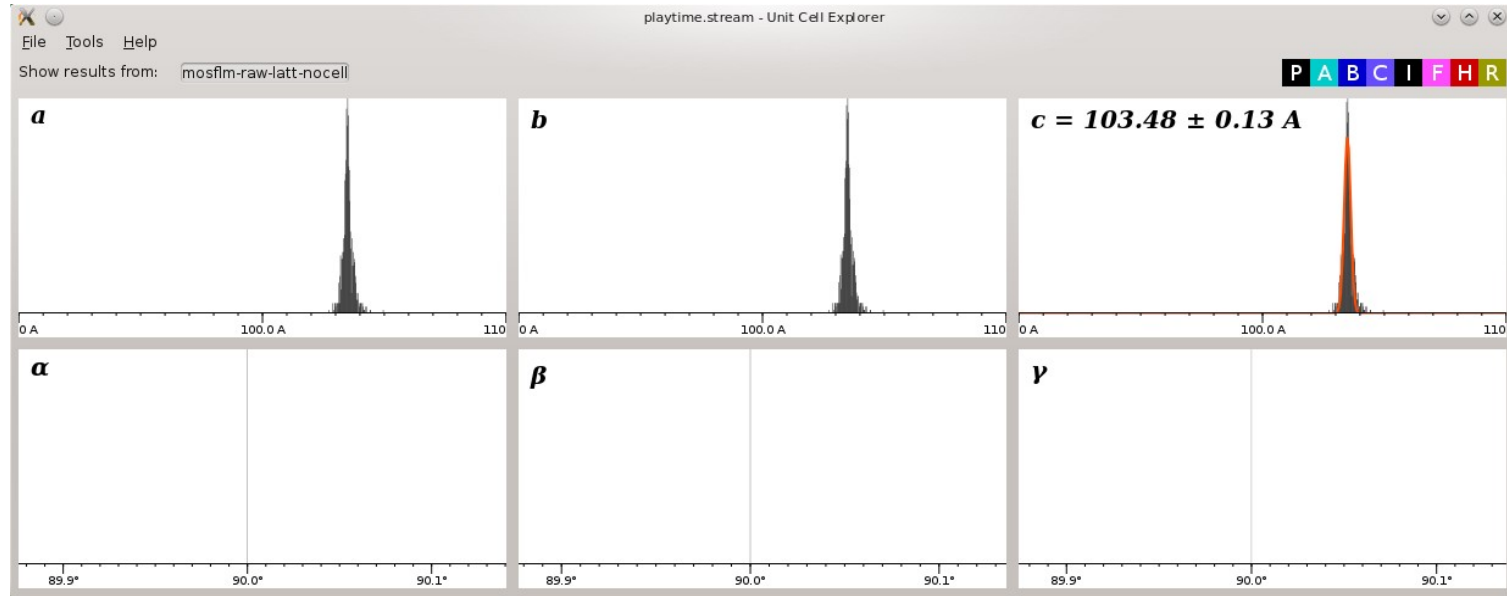
Invoke DirAx

Check the unit cell parameters against the reference, including doubling, halving, adding lattice vectors as necessary.

Do not use Bravais lattice information (DirAx cannot use this information)

The Unit Cell Explorer

> cell_explorer output.stream



$$\begin{aligned} a &= b = c = 103.5 \text{ \AA} \\ \alpha &= \beta = \gamma = 90.0^\circ \\ &\text{Body centered (I)}. \end{aligned}$$

Possible space groups: I23, I2₁3, I432, I4₁32, I422 I4₁22, I4₁, I4, I222

Indexing methods

> `indexamajig [...] --indexing=method1,method2,... [.....]`
List of indexing methods will be tried in order.

> `--indexing=dirax,mosflm,xds,mosflm-raw-latt,dirax-raw`

Invoke DirAx (using defaults, i.e. match unit cell parameters)

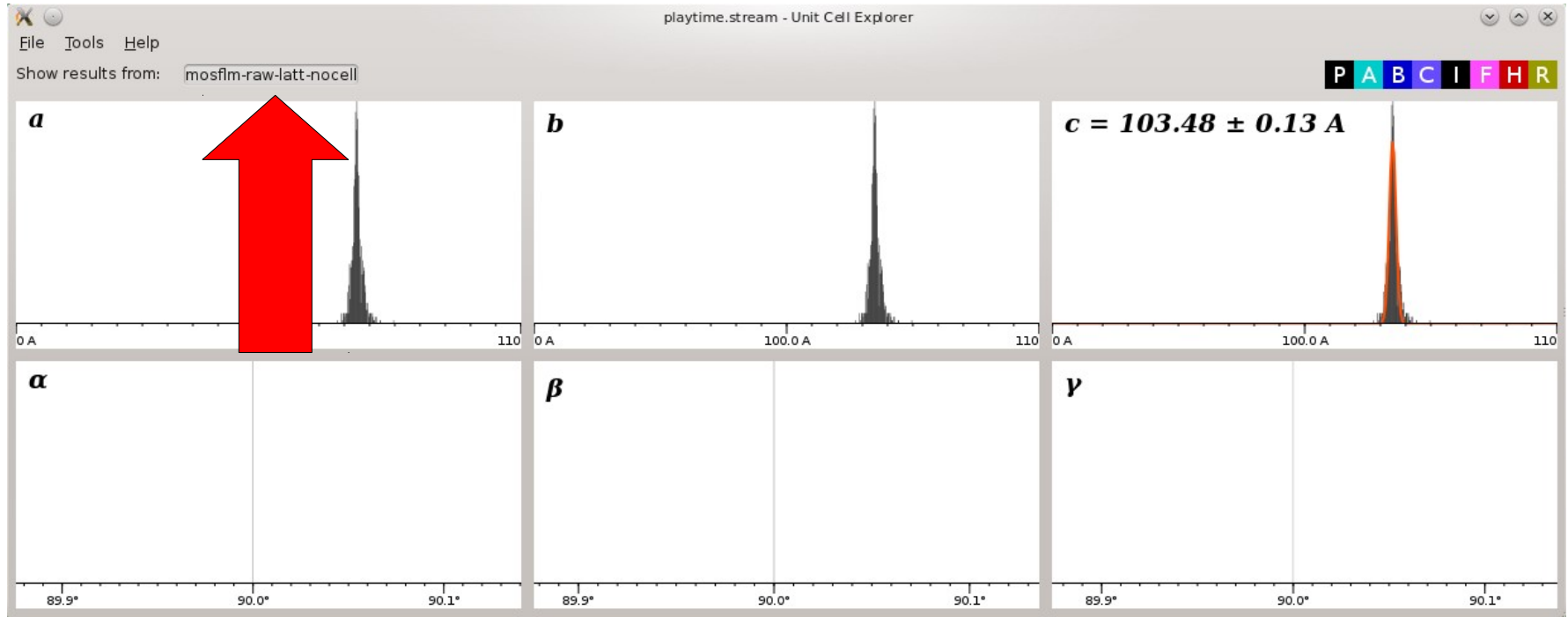
Invoke Mosflm (using defaults, i.e. match unit cell parameters, use lattice information)

Try XDS (defaults)

Still not indexed?
Try getting Mosflm to look for our lattice type, but any parameters

Last resort: see what DirAx finds on its own.

The Unit Cell Explorer



Geometry and beam files

```
> indexamajig [...] --beam=lcls.beam --geometry=cspad.geom [.....]
```

```
beam/bandwidth = 0.001  
beam/divergence = 0.008  
profile_radius = 0.008e9  
beam/photon_energy = /LCLS/photon_energy_eV  
beam/photon_energy_scale = 1.0
```

← Important for spot prediction

```
beam/fluence = 1.0e12  
beam/radius = 10.0e-6
```

← Used only by pattern_sim - ignored for indexing

Beam files will probably go away in CrystFEL 0.6.0.

← A menace ☹️

“man crystfel_geometry”, templates distributed with CrystFEL, or ask a friendly PCS collaborator, beamline scientist or us.

Analysis in CrystFEL can be surprisingly robust to inaccurate geometry.

“Calibration mode” in hdfsee.

Geometry refinement tool to be added in CrystFEL 0.6.0.

Integration methods

> `indexamajig [...] --integration=method [.....]`
(only one integration method allowed)

> Example method:

`rings-sat-cen`

Use “basic”
summation integration
(with background
subtraction)

Include saturated
reflections

Center the
integration position
on the peak first

Integration methods

> `indexamajig [...] --integration=method [.....]`
(only one integration method allowed)

> Example method:

`prof2d-nocen`

Use 2D profile fitting

Exclude saturated reflections (this is the default)

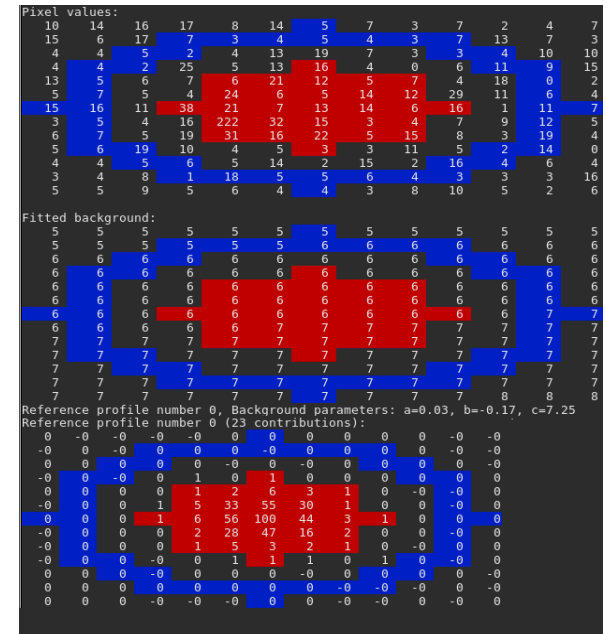
Do not center the integration box

Integration diagnostics

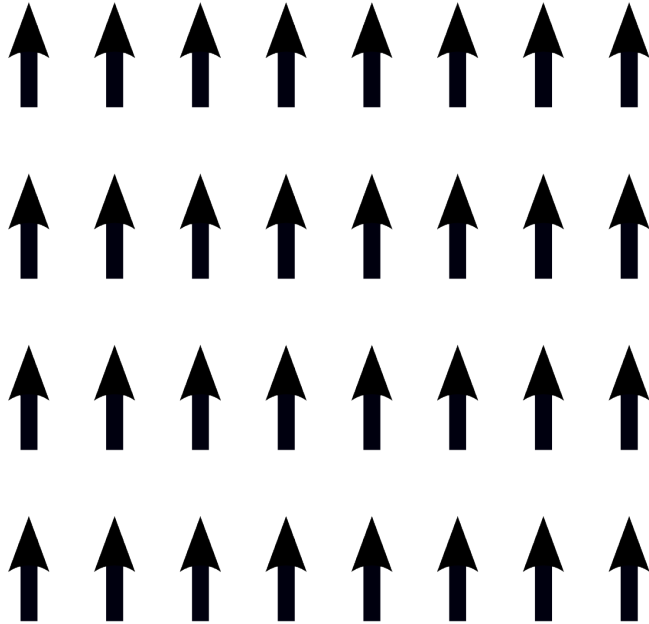
> `indexamajig [...] --int-diag=<something>`

> <something> can be:

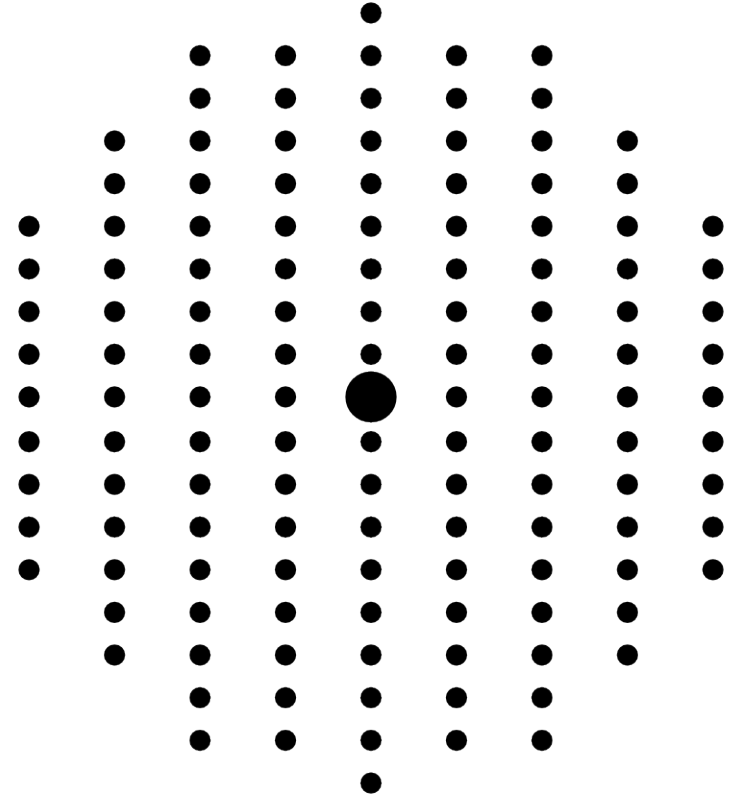
- none (this is the default)
- A set of indices, e.g. 4, 3, 5
- random – show a random 1% of reflections
- implausible – show reflections with $I < -5\sigma$
- negative – show reflections with $I < -3\sigma$



Indexing ambiguities

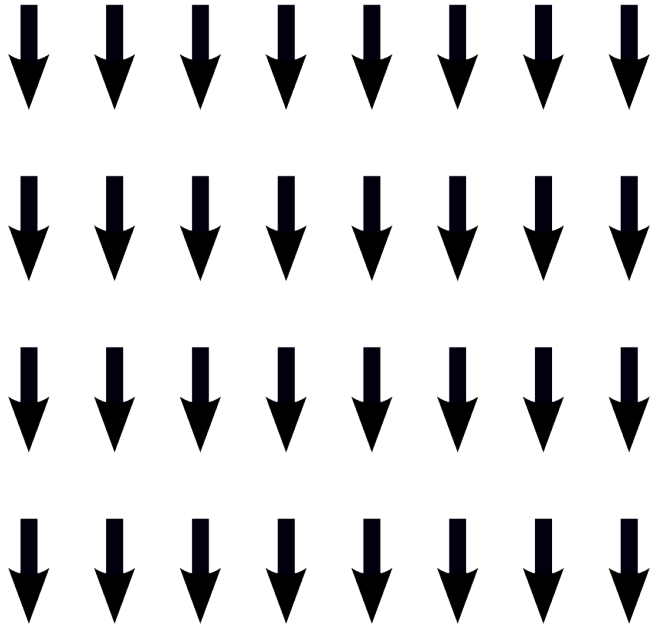


A crystal structure

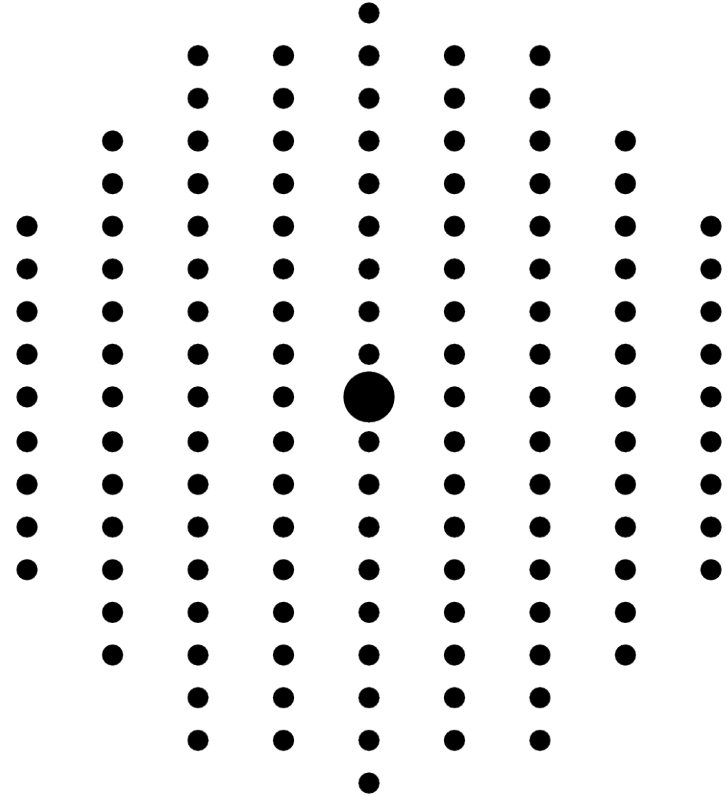


Schematic diffraction pattern

Indexing ambiguities

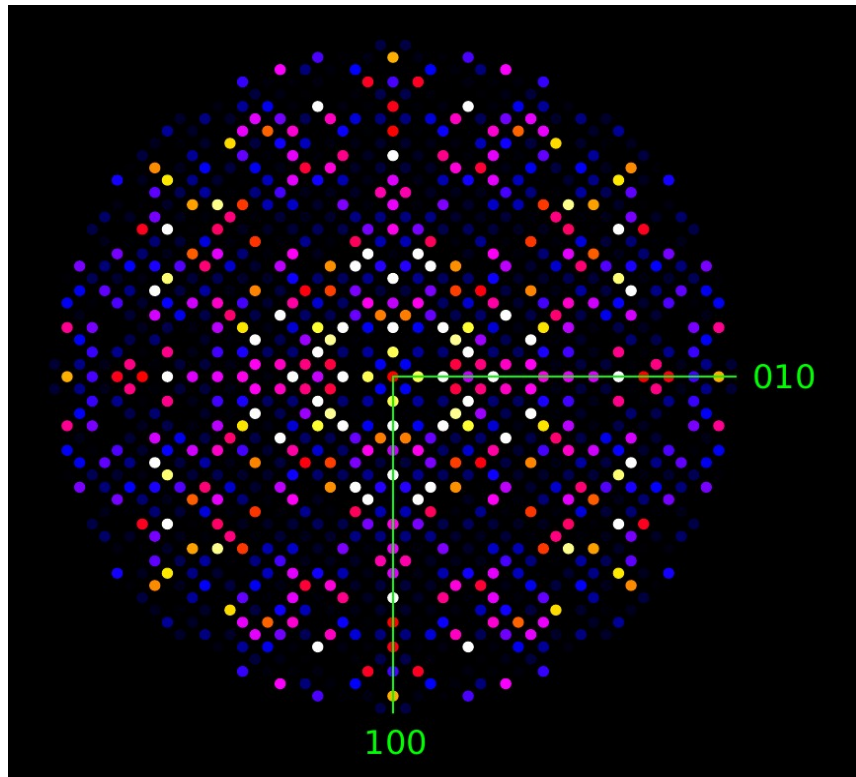


A crystal structure

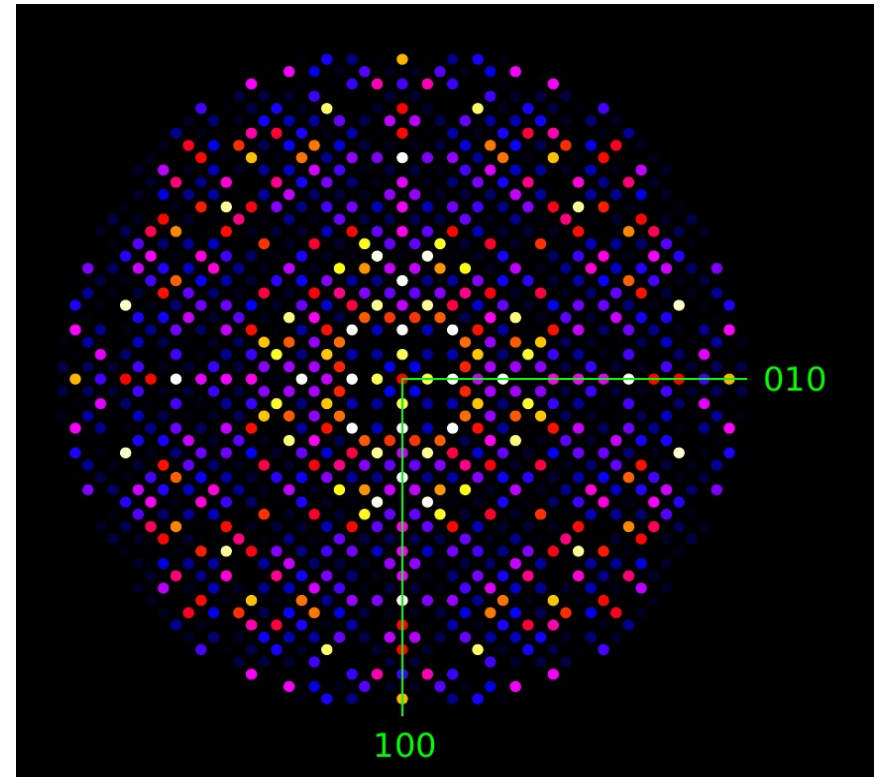


Schematic diffraction pattern

Indexing ambiguities



Actual zone axis intensities
Twofold symmetry



Zone axis intensities if indexing
ambiguity is not resolved.
Fourfold symmetry

Indexing ambiguities

Triclinic lattice								
$\bar{1}$	1	P1		P1				
Monoclinic lattice								
	m			Pm, Pc, Cm, Cc				
2	$\frac{2}{m}$	P2, P2 ₁ , C2		P2/m, P2 ₁ /m, C2/m, P2/c, P2 ₁ /c, C2/c				
Orthorhombic lattice								
	mm2			Pmm2, Pmc2 ₁ , Pcc2, Pma2, Pca2 ₁ , Pnc2, Pmn2 ₁ , Pba2, Pna2 ₁ , Pnn2, Cmm2, Cmc2 ₁ , Ccc2, Amm2, Aem2, Ama2, Aea2, Fmm2, Fdd2, Imm2, Iba2, Ima2				
222	mmm	P222, P222 ₁ , P2 ₁ 2 ₁ 2, P2 ₁ 2 ₁ 2 ₁ , C222 ₁ , C222, F222, I222, I2 ₁ 2 ₁ 2 ₁		Pmmm, Pnnn, Pccm, Pban, Pmma, Pnna, Pmna, Pcca, Pbam, Pccn, Pbcm, Pnnm, Pmnm, Pbcn, Pbca, Pnma, Cmcm, Cmce, Cmmm, Cccm, Cmme, Ccce, Fmmm, Fddd, Immm, Ibam, Ibca, Imma				
Tetragonal lattice								
4	$\bar{4}$			4mm	P4, P4 ₁ , P4 ₂ , P4 ₃ , I4, I4 ₁	P4, I4	P4mm, P4bm, P4 ₂ cm, P4 ₂ nm, P4c, P4nc, P4 ₂ mc, P4 ₂ bc, I4mm, I4cm, I4 ₁ md, I4 ₁ cd	
	$\bar{4}2m$	$\bar{4}m2$	$\frac{4}{m}$					P42m, P42c, P4 ₂ m, P4 ₂ 1c, I42m, I42d
422	4/mmm			P422, P4 ₂ 2, P4 ₂ 22, P4 ₂ 2, P4 ₂ 22, P4 ₂ 2, P4 ₂ 22, P4 ₃ 2, I422, I4 ₂ 22	P4/mmm, P4/mcc, P4/nbm, P4/nnc, P4/mbm, P4/mnc, P4/nmm, P4/ncc, P4 ₂ /mcm, P4 ₂ /nbc, P4 ₂ /nmm, P4 ₂ /mbc, P4 ₂ /mnm, P4 ₂ /nmc, P4 ₂ /ncm, I4/mmm, I4/mcm, I4 ₁ /amd, I4 ₁ /acd			
Rhombohedral lattice								
3	$\bar{3}$	3m	R3 (H3)		R3 (H3)	R3m (H3m), R3c (H3c)		
32	$\bar{3}m$		R32 (H32)		R3m (H3m), R3c (H3c)			
Hexagonal lattice								
6	3	$\bar{3}$			6mm	P3, P3 ₁ , P3 ₂	P3	P6mm, P6cc, P6 ₃ cm, P6 ₃ mc
		3m1	$\bar{6}$	31m				
622	6/mmm			P6 ₃ 1, P6 ₃ 2, P6 ₂ , P6 ₄ , P6 ₃	P6m2, P6c2	P6 ₂ m, P6 ₂ c	P31m, P31c	P6/mmm, P6/mcc, P6 ₃ /mcm, P6 ₃ /mmc
Cubic lattice								
23	$\bar{4}3m$	$\bar{m}\bar{3}$	P23, F23, I23, P2 ₁ 3, I2 ₁ 3		$\bar{4}3m, \bar{F}43m, I43m, P43n, F43c, I43d$	$\bar{P}m\bar{3}, Pn\bar{3}, Fm\bar{3}, Fd\bar{3}, Im\bar{3}, Pa\bar{3}, Ia\bar{3}$		
432	$\bar{m}\bar{3}m$		P432, P4 ₂ 32, F432, F4 ₁ 32, I432, P4 ₃ 32, P4 ₁ 32, I4 ₁ 32		Pm3m, Pn3n, Pm3n, Pn3m, Fm3m, Fm3c, Fd3m, Fd3c, Im3m, Ia3d			

Indexing ambiguities

```
> ambigator input.stream -o output.stream  
-y 6/m -w 6/mmm -j 8 --lowres=6 --highres=3
```

Actual symmetry
(note: centre of
symmetry added, i.e.
consider Friedel pairs
as equivalent)

Apparent symmetry
(can also give the
ambiguity operator
directly, using
--operator)

Use 8 threads for
calculations (go
faster).

Resolution range (in
Å) to be considered.
Needs to be big, but
not too big.

Merging

```
> process_hkl -i input.stream -o output.hkl -y 6/m
```

Output: text file, can be imported into most structure solution programs. Alternatively: “create-mtz” and “create-xscale” scripts.

Actual symmetry (note: centre of symmetry added, i.e. merge Friedel pairs)

Confused about point group notation? “man crystfel” for a list of all possibilities.

“Best practice” guidelines



Information

- [Home](#)
- [About CrystFEL](#)
- [Download](#)
- [Extra programs](#)
- [Screenshots](#)
- [Development](#)
- [Citations](#)

Documentation

- [Overview](#)
- [Installation](#)
- [Manual](#)
- [Best Practice](#)
- [FAQ](#)
- [Changes](#)
- [Tutorial](#)
- [API Reference](#)

- [Contact](#)
- [Legal information](#)

- [DESY Homepage](#)
- [CFEL Homepage](#)
- [DESY@CFEL](#)

CrystFEL processing best practice

This page contains our most recent recommendations to processing SFX data using CrystFEL, as well as some hints and tips.

General

- Read all the messages written to the terminal by the various CrystFEL programs. They contain important information and useful advice.

Peak search

- Good peak detection is essential for indexing, so it's worth spending a long time optimising the peak search. Experiment with and use whichever combination you like from the peak detection options, which are: `--peaks=zaef`, `--peaks=hd15`, `--threshold`, `--gradient`, `--min-snr`, `--section-filter`, `--filter-noise`, `--no-use-saturated`, `--no-reval1data` and `--check-hd15-snr`. None of these options can do anything other than increase or decrease the indexing success rate.
- Note that `--use-saturated` has been the default since CrystFEL 0.5.1. This option states that saturated peaks are to be given to the indexing procedure. It does not mean that their intensities will be integrated and included in the final data set.
- `--peaks=hd15` was the default behaviour prior to CrystFEL 0.5.3. Since then, the default behaviour has been `not` to check the SNR of peaks if you use `--peaks=hd15`. If your indexing rate dropped after upgrading to CrystFEL 0.5.3, add this option to restore the old behaviour.

Indexing

- When initially determining the unit cell parameters, try indexing with `mosflm-raw-no-lact-no-cell` first. Mosflm is the only indexing method which can guess the lattice type. Once you're convinced about the lattice type, use `mosflm-ref-test` to constrain the search.
- Once you've determined the cell parameters, use as many indexing methods as you can, provided that they all use `--cell`, `--axes` or `--comp`. The more, the merrier! You can sometimes increase the indexing rate by combining multiple invocations of the same indexing program, for example: `--indexing mosflm-latt mosflm-no-latt`.
- At all stages, avoid using `--bad` anywhere in your list of indexing methods. Add this option only if it's really necessary.
- Do not use `--comb` if once of the cell parameters is a simple multiple of the others, e.g. if `a` is approximately equal to `2b`. **This happens with tetragonal lysozyme**, so don't assume it won't happen to you. `--comb` is the default for many indexing methods, so specify `--axes` explicitly. See the [tutorial](#) (step 10) for a discussion.
- Set the unit cell tolerances (`unit_cell_tol`) wide enough to capture the entire distributions of all the parameters, unless you have some reason to restrict it. The widths of the distributions vary between samples, so you have to

“Definitely use this option”

“Don't use this option except for testing”

“Try this option and see what happens, might be good or bad”

“Watch out for this 'gotcha'”

... etc ...



Future developments: CrystFEL 0.5.4 and 0.6.0

- > Version 0.5.4: after this workshop, unless we find any show-stoppers.
 - Incremental update, minor new features etc.

- > Version 0.6.0: in a few months - bigger changes.
 - Multiple images in one HDF5 file (use SLAC HDF5s directly, reduce load on filesystem).
 - No more “beam” files.
 - No more “-y” options (this can be determined automatically).
 - ... some other goodies ...

Future developments: multi-lattice indexing

- > Core of CrystFEL can already deal with any number of crystals per pattern.
- > Limitation: underlying indexing algorithms.
(indexing algorithms have not been a main theme of CrystFEL development - we take advantage of previous work by calling other programs).
- > Coming soon (CrystFEL 0.6.0):
 - Multi-lattice MOSFLM
 - Felix (related to GrainSpotter / TotalCryst)
 - Multi-lattice indexing using internal “retry” mechanism (from Takanori Nakane).

Future developments: partialities and post-refinement

```
> partialator -i input.stream -o output.hkl -y mmm  
--model=sphere --iterations=1
```

Partiality model (options: sphere, gaussian, scsphere, thin, unity).

Number of cycles of post-refinement

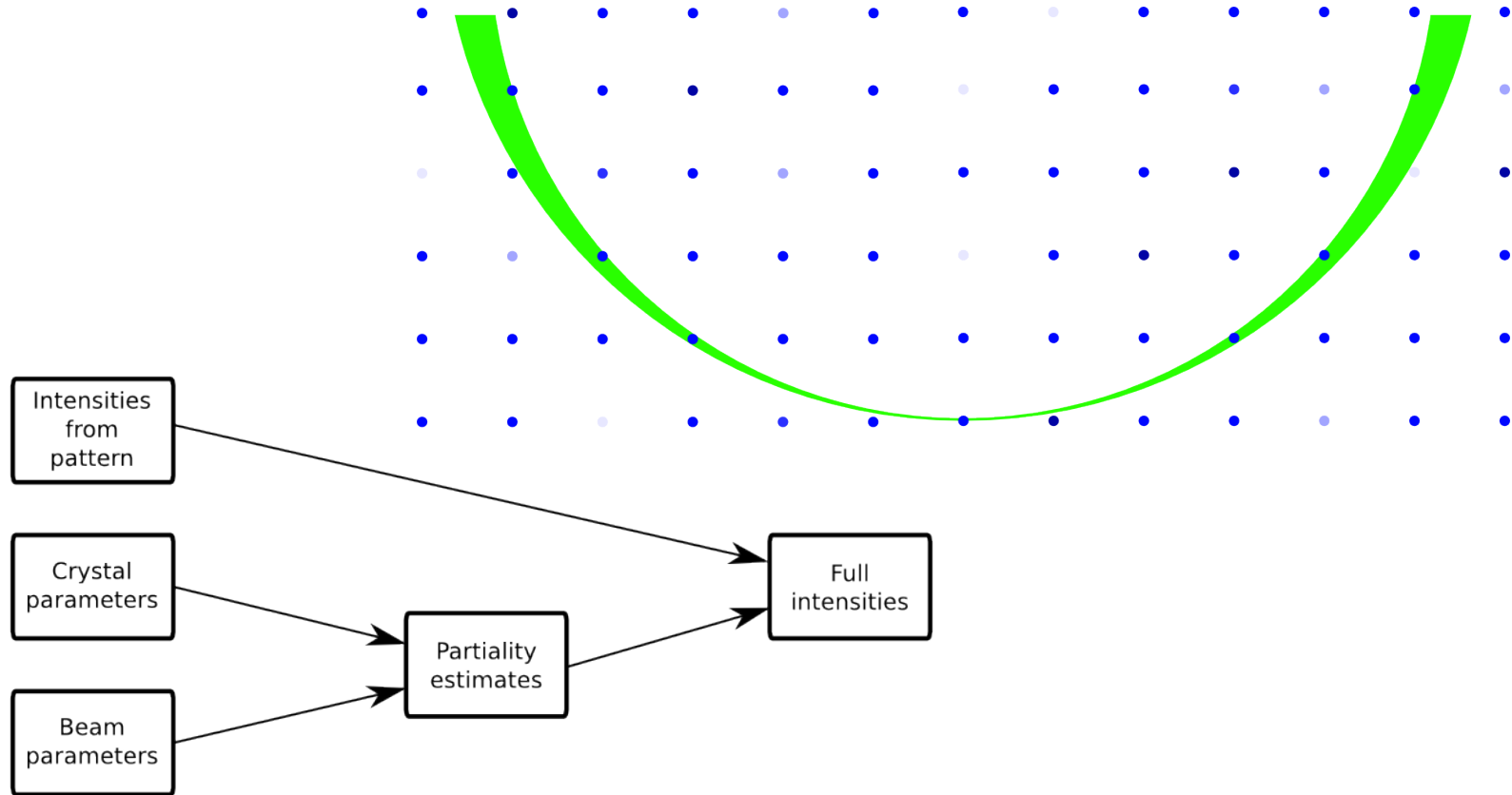
Try --iterations=0 to simply apply the partialities according to the model.

Note: detector geometry, beam file and images not needed.

Future developments: partialities and post-refinement

> Basic scheme (in SFX context):

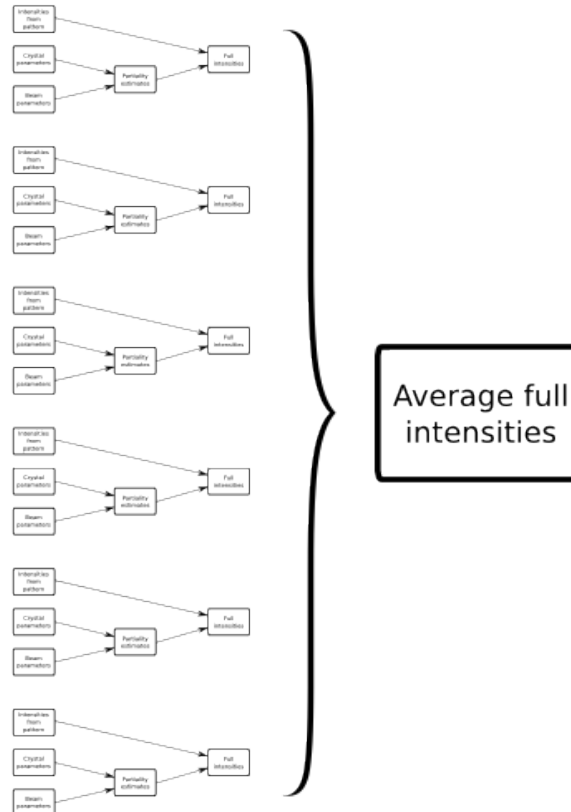
T. A. White, Phil. Trans. R. Soc. B (2014) 369 20130330.



Future developments: partialities and post-refinement

> Basic scheme (in SFX context):

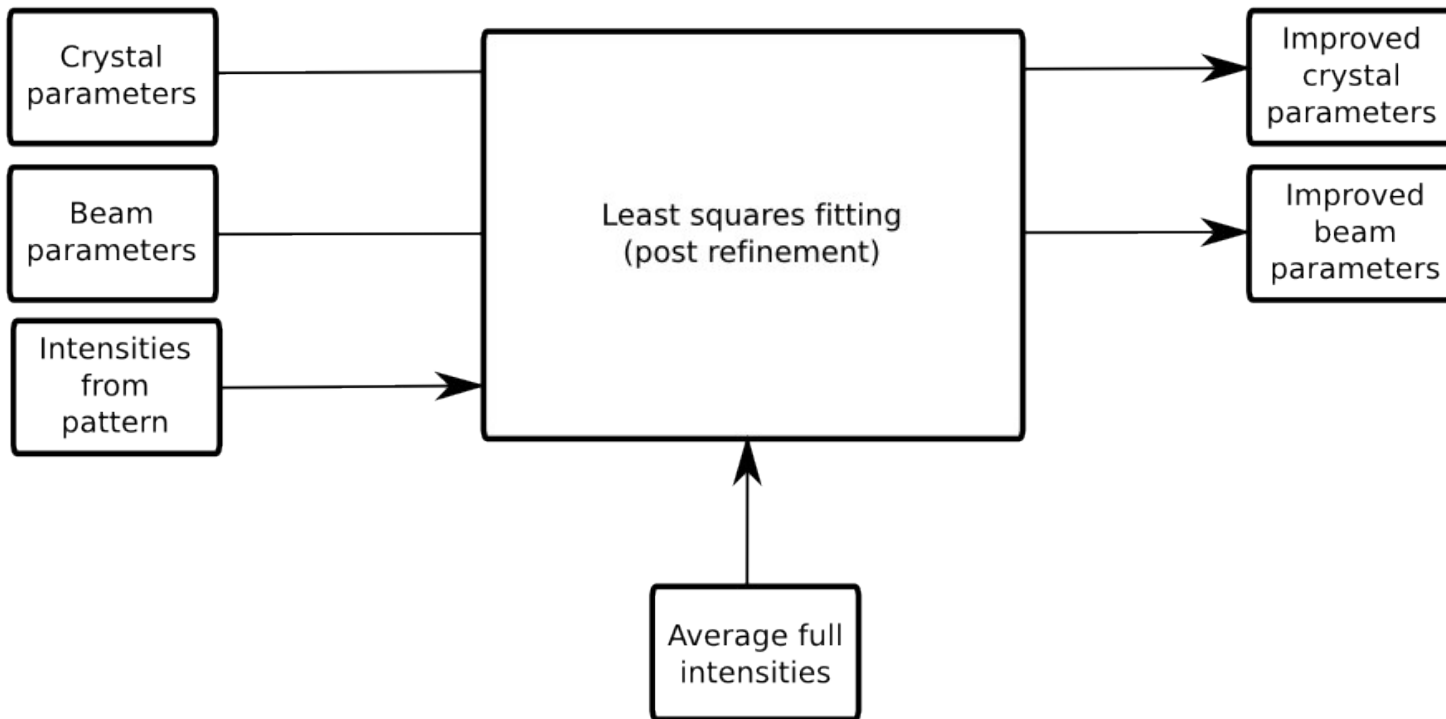
T. A. White, Phil. Trans. R. Soc. B (2014) 369 20130330.



Future developments: partialities and post-refinement

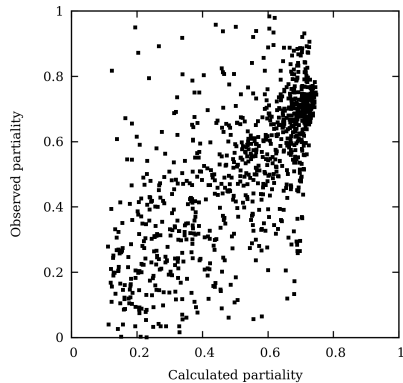
> Basic scheme (in SFX context):

T. A. White, Phil. Trans. R. Soc. B (2014) 369 20130330.

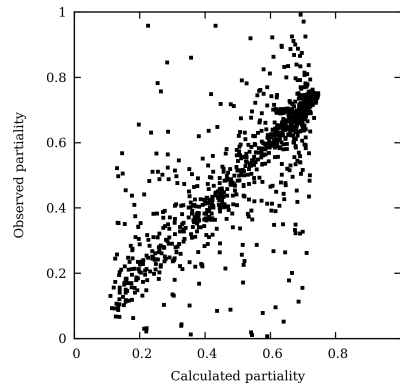


Future developments: partialities and post-refinement

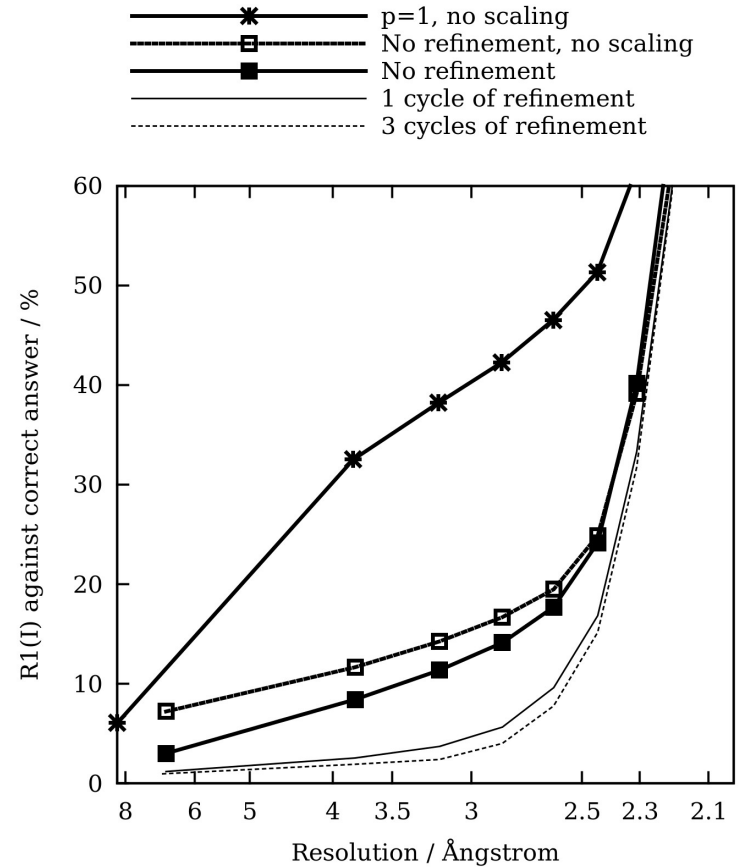
- Works really nicely for simulations of different kinds.
- Difficult to make it work on real data.



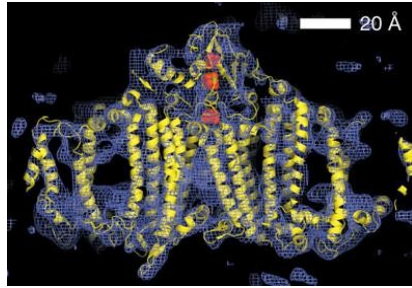
Before refinement



After 1 cycle of refinement



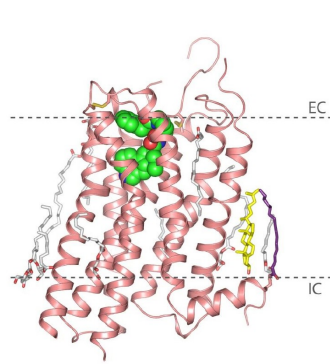
CrystFEL in the PDB (highlights)



3PCQ

Photosystem I (the first SFX experiment)

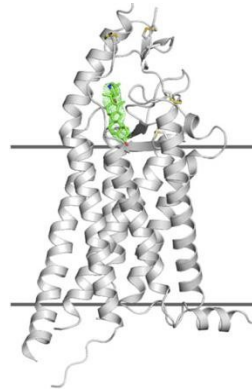
Chapman et al., Nature 2011



4NC3

Serotonin receptor bound to ergotamine

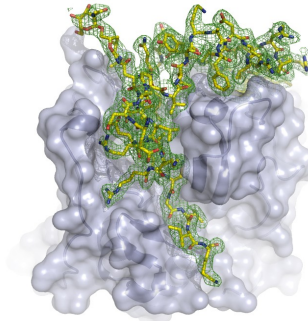
Lui et al., Science 2013



4O9R

Smoothed receptor using LCP injector

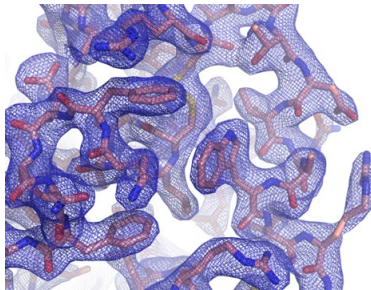
Weierstall et al., Nature Communications 2014



4HWY

Natively inhibited Cathepsin B

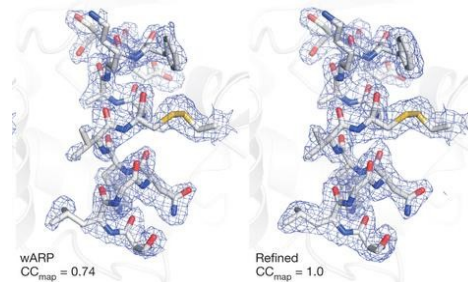
Redecke, Nass et al., Science 2013



4O34

Serial crystallography using a synchrotron beamline

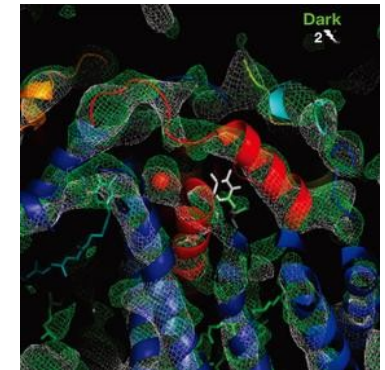
Stellato et al., IUCrJ 2014



4N5R

Lysozyme (Gd derivative) ab initio phasing using SAD

Barends et al., Nature 2013



4Q54

Photosystem II in putative S_3 excited state

Kupitz, Basu et al., Nature 2014

CrystFEL in the PDB

2014

- C. Kupitz, S. Basu, I. Grotjohann, R. Fromme et al. "Serial time-resolved crystallography of photosystem II using a femtosecond X-ray laser". *Nature* (2014). **4PB**U and **4Q54**.
- M. Caffrey, D. Li, N. Howe and S. T. A. Shah. "'Hit and run' serial femtosecond crystallography of a membrane kinase in the lipid cubic phase". *Phil Trans. Roy. Soc. B* 369 (2014) 20130621.
- O. Yefanov, C. Gati, G. Bourenkov, R. A. Kirian et al. "Mapping the continuous reciprocal space intensity distribution of X-ray serial crystallography". *Phil. Trans. Roy. Soc. B* 369 (2014) 20130333.
- T. A. White. "Post-refinement method for snapshot serial crystallography". *Phil. Trans. Roy. Soc. B* 369 (2014) 20130330.
- J. C. H. Spence, N. A. Zatsepin and C. Li. "Coherent convergent-beam time-resolved X-ray diffraction". *Phil. Trans. Roy. Soc. B* 369 (2014) 20130325.
- F. Stellato, D. Oberthür, M. Liang, R. Bean et al. "Room-temperature macromolecular serial crystallography using synchrotron radiation". *IUCrJ* 1 (2014). **4O34**.
- K. Qu, L. Zhou and Y.-H. Dong. "An improved integration method in serial femtosecond crystallography". *Acta Cryst. D70* (2014) p1202.
- M. Frank, D. B. Carlson, M. S. Hunter, G. J. Williams et al. "Femtosecond X-ray diffraction from two-dimensional protein crystals". *IUCrJ* 1 (2014) p95.
- C. Gati, G. Bourenkov, M. Klinge, D. Rehders et al. "Serial crystallography on in vivo grown microcrystals using synchrotron radiation". *IUCrJ* 1 (2014) p87. **4N4Z**.
- U. Weierstall, D. James, C. Wang, T. A. White et al. "Lipidic cubic phase injector facilitates membrane protein serial femtosecond crystallography". *Nature Communications* 5:3309 (2014). **4O9R**.
- C. Song, K. Tono, J. Park, T. Ebisu et al. "Multiple application X-ray imaging chamber for single-shot diffraction experiments with femtosecond X-ray laser pulses". *J. Appl. Cryst.* 47 (2014) p18.

2013

- W. Liu, D. Wacker, C. Gati, G. W. Han et al. "Serial Femtosecond Crystallography of G Protein-Coupled Receptors". *Science* 342 (2013) p1522. **4NC3** and **CXIDB ID 21**.
- L. C. Johansson, D. Arnlund, G. Katona, T. A. White et al. "Structure of a photosynthetic reaction centre determined by serial femtosecond crystallography". *Nature Communications* 4:2911 (2013). **4CAS**.
- T. R. M. Barends, L. Foucar, S. Botha, R. B. Doak et al. "De novo protein crystal structure determination from X-ray free-electron laser data". *Nature* (2013). **4N5R** and **CXIDB ID 20**.
- H. Dermirci, et al. "Serial femtosecond X-ray diffraction of 30S ribosomal subunit microcrystals in liquid suspension at ambient temperature using an X-ray free-electron laser". *Acta Crystallographica F69* (2013) p1066.
- T. A. White, A. Barty, F. Stellato, J. M. Holton, R. A. Kirian, N. A. Zatsepin and H. N. Chapman. "Crystallographic data processing for free-electron laser sources". *Acta Cryst. D69* (2013), p1231.
- T. R. M. Barends, L. Foucar, R. L. Shoeman, S. Bari et al. "Anomalous signal from S atoms in protein crystallographic data from an X-ray free-electron laser". *Acta Crystallographica D69* (2013) p838.
- L. Redecke, K. Nass et al. "Natively Inhibited Trypanosoma brucei Cathepsin B Structure Determined by Using an X-ray Laser". *Science* 339 (2013) p227. **4HWY**.

2012

- S. Boutet, L. Lomb, G. J. Williams, T. R. M. Barends et al. "High-Resolution Protein Structure Determination by Serial Femtosecond Crystallography". *Science* 337 (2012) p362. **4ET8**, **4ET9** and **CXIDB ID 17**.
- L. C. Johansson, D. Arnlund, T. A. White, G. Katona et al. "Lipidic phase membrane protein serial femtosecond crystallography". *Nature Methods* 9 (2012) p263. **4AC5**.
- R. Koopmann, K. Cupelli, L. Redecke, K. Nass et al. "In vivo protein crystallization opens new routes in structural biology". *Nature Methods* 9 (2012) p259.
- A. Aquila, M. S. Hunter, R. B. Doak et al. "Time-resolved protein nanocrystallography using an X-ray free-electron laser". *Optics Express* 20 (2012) p2706.
- A. Barty, C. Caleman, A. Aquila, N. Timneanu et al. "Self-terminating diffraction gates femtosecond X-ray nanocrystallography measurements". *Nature Photonics* 6 (2012) p35.

2011

- H. N. Chapman, P. Fromme et al. "Femtosecond X-ray protein nanocrystallography". *Nature* 470 (2011) p73. **3PCQ**.

(see the website)