

## Processing XFEL data with *cctbx.xfel* and DIALS

Aaron S. Brewster<sup>a</sup>, David G. Waterman<sup>b</sup>, James M. Parkhurst<sup>c</sup>, Richard J. Gildea<sup>c</sup>, Tara M. Michels-Clark<sup>a</sup>, Iris D. Young<sup>a</sup>, Herbert J. Bernstein<sup>d</sup>, Graeme Winter<sup>c</sup>, Gwyndaf Evans<sup>c</sup> and Nicholas K. Sauter<sup>a</sup>

<sup>a</sup>Lawrence Berkeley National Laboratory, Berkeley, CA 94720

<sup>b</sup>CCP4, Research Complex at Harwell, STFC Rutherford Appleton Laboratory, OX11 0FA (UK)

<sup>c</sup>Diamond Light Source, Harwell Science and Innovation Campus, OX11 0DE (UK)

<sup>d</sup>School of Chemistry and Materials Science, Rochester Institute of Technology, Rochester, NY 14623

Correspondence email: asbrewster@lbl.gov

### 1. Introduction

Protein crystal diffraction data from X-ray free-electron lasers (XFELs) pose difficult challenges to conventional data reduction software. In a typical XFEL experiment, short pulses of photons tens of femtoseconds long containing  $10^{12}$  photons/pulse interact serially with thousands of individual crystals, each producing a single pattern before the intense light destroys the crystal. These diffraction patterns represent ‘still’ shots and must be treated differently than rotation datasets collected using a goniometer. For example, with only one shot, refining the crystal orientation around the x- and y-axes of rotation orthogonal to the beam becomes difficult, as the rotation around these axes does not affect the locations of reflections, but only which reflections are in the diffracting condition. Furthermore, without a way of measuring the rocking curve directly by transitioning reflections through the Ewald sphere during a crystal rotation, it is difficult to produce estimates of mosaicity and thus predict which weak reflections will be in the diffracting condition. For these and other reasons described elsewhere, we have implemented the package *cctbx.xfel*, based on *cctbx* (Grosse-Kunstleve *et al.* 2002), which includes specializations of known indexing and refinement algorithms specific for the stills case (Sauter *et al.* 2013, Hattne *et al.* 2014).

Further challenges unrelated to the physics of crystallography are encountered when processing XFEL data. First, the short pulse length makes photon-counting detectors unsuited for recording XFEL data. Integrating detectors such as those used with charged-

coupled devices (CCDs) are more appropriate, but in order to handle the high pulse rate typical of XFELs (120 Hz at the Linac Coherent Light Source (LCLS)), new detectors were created, such as the Cornell-SLAC Pixel Array Detector (CSPAD) (Hart *et al.* 2012). The CSPAD is composed of 32 separate sensors, the positions of which are not precisely known.

Finally, diffraction data collected at 120 Hz must be reduced using large computing clusters with memory, file I/O and network capacity capable of keeping pace with the experiment in a timely manner, so that a data reduction team can provide live feedback to beam line operators and scientists, allowing them to change sample measurement conditions as needed.

This article illustrates features and processing patterns of a typical XFEL experiment and provides commands for running *cctbx.xfel* in conjunction with the new software for reducing data from difficult systems, DIALS (Diffraction Integration for Advanced Light Sources) (Waterman *et al.* 2013). DIALS is built on the *cctbx* toolkit and takes advantage of many of its features, including image file reading from a variety of formats, crystal symmetry libraries and minimization engines. Given a collection of diffraction images, DIALS produces a series of models of the experiment, describing the detector, beam and crystal, and, as appropriate, goniometer and scan (Parkhurst *et al.* 2014). It also implements new indexing, refinement and integration algorithms, continuing in the

'toolbox' tradition of open source and object-oriented programming set in place by *cctbx*.

### 1.1. System overview

The ability to do serial X-ray crystallography has relied on the convergence of several critical technologies, such as the X-ray source and sample injection systems. Reduction of XFEL diffraction data requires further technologies including a) a detector capable of recording XFEL pulses at speeds matching the source, b) parallel computing infrastructure, c) new and adapted file formats for data storage and d) data reduction software. Here, we provide a brief overview of these technologies.

### 1.2. The detector

XFEL pulses challenge detector technology. The CSPAD detector is capable of integrating high signals from the ultra-bright XFEL source, while operating at 120 Hz. Two CSPADs are installed at the CXI end-station at LCLS; another is installed at the XPP end-station. The detector comprises 32 sensors, each consisting of 2 Application-Specific Integrated Circuits (ASICs) 194x185 pixels in dimension. The sensors are arranged in 4 quadrants of 8 sensors each. The CSPADs at CXI have their quadrants each on a diagonal rail that allows tuning the size of the central aperture through which the transmitted beam must pass, while the quadrants of the CSPAD at XPP are in a fixed arrangement. Each detector is regularly upgraded and improved and sensor positions vary as a consequence, although these are measured using an optical microscope. Indexing, predicting spot locations using a crystal orientation matrix and integrating reflection intensities require precise knowledge of the locations of these sensors in three-dimensional space (Hattne *et al.* 2014). For this reason, a portion of this article describes the calibration and refinement of the sensor or tile metrology using a reference dataset.

Each pixel in the detector can be configured to a low or high gain setting. The low gain setting

has a full well capacity of 3000 photons, while the high gain setting saturates about seven times as quickly but has a higher sensitivity (Hart *et al.* 2012). This feature allows the user to specify, for example, a circular gain mask that sets low resolution pixels to low gain mode to avoid saturation of intense low resolution reflections while keeping high resolution pixels in high gain mode to capture weak diffraction data near the detector limits.

Other detectors in use at XFEL sources for protein crystals include the octal sensor detector at SACLA (Kameshima *et al.* 2014), a Rayonix MX 170 HS detector at XPP (Chollet *et al.* 2015) and a MAR 325 CCD detector, also at XPP at LCLS, occasionally brought in from SSRL for use with fixed target experiments (Cohen *et al.* 2014). Each of these detectors has its own set of tradeoffs and *cctbx.xfel* has been used to process data from all of them.

### 1.3. Parallel computing

Recording at 120 Hz yields 72000 2.2 megapixel CSPAD images in a typical 10 minute LCLS run. Processing this volume of data without a parallel computing environment quickly becomes impractical. The clustering environment at LCLS is ideal. Hundreds of nodes can be harnessed with 12 - 16 computer cores each to greatly accelerate indexing and integration. The program **cxi.mpi\_submit**, a component of *cctbx.xfel*, provides an interface for submitting processing jobs to the LCLS cluster.

Additionally, we have collaborated with NERSC (National Energy Research Scientific Computing center) to transfer the data streams from the CSPAD detectors to the NERSC clustering systems for processing (Kern *et al.* 2014). NERSC is utilized for some of the largest data reduction problems world wide, including climate and astrophysics simulations and is ideal for efficient, parallel reduction of data from serial crystallographic experiments.

#### 1.4. Data reduction software

Diffraction data recorded on CSPADs at LCLS is streamed by dedicated Data Acquisition Systems (DAQs) to container files in XTC format. The programmatic interface to interact with these files at LCLS is *psana* (Damiani *et al.* 2016) *cctbx.xfel* was originally designed to use the older *pyana* interface (Sauter *et al.* 2013, Hattne *et al.* 2014) and has transitioned to *psana* while maintaining backward compatibility.

*psana* uses a calibration store to read frames and apply pixel corrections such as dark current subtraction and common mode correction and it is designed with computational parallelization in mind. As each image is independent, multiple computer cores can process images in parallel. *cctbx.xfel* interfaces with *psana* to read XTC streams, parse the LCLS metrology file that describes the layout of the CSPAD and load pixel data for each image. Next, *cctbx.xfel* uses DIALS or LABELIT to create detector and beam models, find spots, index the reflections, refine the experimental model and integrate the reflection intensities. The user specifies processing parameters in *cctbx*-style parameter files (PHIL files, see below) and passes the parameter file to *cctbx.xfel*, which calls *psana* and submits the job to the queuing system. Specific details are described below and in online tutorials at <http://cci.lbl.gov/xfel>.

*cctbx.xfel* is the XFEL data reduction package developed by the Computational Crystallographic Initiative at LBNL and is installed for all users at LCLS.

##### 1.4.1. *cctbx.xfel* indexing, refinement and integration back ends (LABELIT and DIALS)

*cctbx.xfel* was originally implemented using 1D Fourier indexing algorithms (Steller *et al.* 1997), as made available in LABELIT (Sauter *et al.* 2004). This LABELIT backend was expanded in the *cctbx* framework to include stills-specific algorithms, such as additional targets for refining crystal orientation and

refining mosaic estimates needed to determine which reflections are in the diffracting condition (Sauter *et al.* 2013, Hattne *et al.* 2014, Sauter *et al.* 2014, Sauter 2015). These procedures have been implemented and expanded in DIALS, taking advantage of the diffraction models and refinement engine made available in that platform (Waterman *et al.* 2016).

For example, the DIALS indexer provides three algorithms for determining initial sets of basis vectors during indexing: *fft3d*, *fft1d* and real space grid search. *fft1d* is an implementation of the 1D FFT algorithms described above, with the addition that the user can markedly increase the success rate of indexing by providing a target unit cell and space group based on prior knowledge (Hattne *et al.* 2014). *fft3d* is an implementation of 3D FFT methods (Bricogne 1986, Campbell 1998) and isn't relevant for stills. The real space grid search approach was described recently as a simplification of Fourier methods when unit cell dimensions are already available (Gildea *et al.* 2014). In the case of unknown unit cell parameters, real space grid search is not available and *fft1d* remains the best choice for stills. However, after an initial indexing test, the unit cells determined by *fft1d* from many still diffraction patterns can be hierarchically clustered (Zeldin *et al.* 2015) and a consensus unit cell for the sample can be determined. There are choices of lattice distance functions to be used in clustering. The most effective is the  $G^6$  space distance function (Andrews & Bernstein 2014, McGill *et al.* 2014). Then, the consensus cell can be used as a target for indexing using real space grid search. Notably, in a recent experiment with lysozyme crystals, we found that real space grid search with a well-determined set of unit cell parameters can find up to 21% more lattices than *fft1d*. However, a different experiment found that *fft1d* gave dramatically more results than *real\_space\_grid\_search*. We encourage users to try both options to determine which yields

better results according to a chosen figure of merit and we invite users to contact the authors to share their experiences.

DIALS also provides a mechanism for parameterizing experimental models that lends itself naturally to building complex refinement target functions (Waterman *et al.* 2016). The complete experiment is described through a series of models: the crystal (unit cell and orientation), the beam, the detector, the goniometer axis and orientation and the scan oscillation range and increment. While the goniometer and scan models are not applicable for stills, the crystal, beam and detector models can be refined against measured data according to stills-specific targets (Sauter *et al.* 2014). Importantly, individual parameters such as the wavelength or the detector distance, tilt or orientation can easily be fixed, *i.e.* locked into place, depending on the use case.

The *DIALS* backend for *cctbx.xfel* includes a derivation of the *DIALS* indexer optimized for stills and includes all of the stills-specific algorithms mentioned above, taking advantage of the open-source and object-oriented nature of the *cctbx* framework for which *DIALS*, *LABELIT* and *cctbx.xfel* are all derived.

#### 1.4.2. XTC and CSPAD CBF formats

The LCLS data acquisition systems stream terabytes of data to container files in XTC format. XTC is a linear, sequential-access file format, where individual events can be recorded rapidly to the file system as they are collected. ‘Derived’ metadata such as detector position and percent beam attenuation are not provided directly; instead, motor

positions and the status of LCLS instrument parameters are recorded. It is up to end user software to transform this information into the parameters needed to describe the crystallographic experiment. *psana* abstracts many of these transformations. For instance, *cctbx.xfel* interfaces with *psana* to couple the raw pixel data from the 64 ASICs in each CSPAD detector event in the XTC stream with transformed metadata to create files in Crystallographic Binary Format (CBF) that contain the pixel data and also completely describe the experiment in their headers, using standards established by the International Union of Crystallography (IUCr) (Bernstein & Hammersley 2006). This format has been described in detail previously (Brewster *et al.* 2014). Briefly, the geometry of the CSPAD detector is recorded as a series of basis transformations that move an observer from the sample interaction point (the crystal) to the detector, then to each of 4 quadrants, then to each of 32 sensors, then to each of 64 ASICs. All transformations are relative to a parent frame, which allows the positions of groups of objects (such as sensors in a quadrant) to be refined as a unit by only refining the vectors defining the parent object's frame of reference.

#### 1.4.3. PHIL format

Python Hierarchical Interface Language (PHIL) is the syntax for specifying parameters in *cctbx* (Grosse-Kunstleve *et al.* 2005, Grosse-Kunstleve *et al.* 2006, Bourhis *et al.* 2007). *Phenix* users will know it as .eff format (effective parameter file). This short example is used to configure the *DIALS* spotfinder (details explained below):

```
spotfinder {
  filter.min_spot_size=2
  threshold.xds.gain=25
  threshold.xds.global_threshold=100
}
```



PHIL format uses curly braces to establish parameter scopes among programs and uses name-value pairs to specify parameters. Here, spotfinding parameters such as minimum spot size, gain estimates and global background thresholds are provided inside the spotfinder scope.

#### 1.4.4. Intermediate DIALS formats

DIALS utilizes two intermediate file formats to store experimental models and reflection information prior to merging and scaling data and writing MTZ format files for subsequent structure solution and refinement.

##### 1.4.4.1. JSON format

DIALS represents crystallographic experiments as a series of physical models, including detector, beam, goniometer and scan, through the Diffraction Experiment Toolbox library (*dxtbx*) included in *cctbx* (Parkhurst *et al.* 2014). For stills, acquired by serial femtosecond crystallography, no

goniometer or scan objects are used. For each component in the hierarchical detector, the model includes the positional vectors  $d_0$ ,  $d_f$  and  $d_s$ .  $d_0$  points from the parent component's origin to the child's origin, while  $d_f$  and  $d_s$  define orthogonal fast and slow vectors that, when combined with the normal vector ( $d_f \times d_s$ ), specify a basis frame for the component. The 'leaves' of the detector model, *e.g.* the ASICs for the CSPAD, also contain information to convert from pixel to millimeter coordinates, such as pixel size, taking into account a parallax correction by including ASIC thickness and material composition. The beam model includes the beam vector ( $s_0$ ) and the wavelength of incident photons. All of these metadata are serialized using JavaScript Object Notation (JSON) in .json files.

The JSON files are organized thusly (here, each indentation level represents a JSON entry):

```
ExperimentList
Experiment1
  BeamID
  DetectorID
  CrystalID
Experiment2
...
Beams
  Beam1
    <beam properties i.e. wavelength, direction>
  Beam2
  ...
Detectors
  Detector1
    <detector properties i.e.  $d_0$ ,  $d_f$ ,  $d_s$  vectors>
  Detector2
  ...
Crystals
  Crystal1
    <crystal properties i.e. unit cell, orientation>
  Crystal2
  ...
```

At the top, an experiment list defines a set of experiments each containing a beam, detector and crystal model, identified using numerical IDs. Then, the set of detector, crystal and beam models referenced by the experiments are listed. An individual experiment always contains exactly one detector, beam and crystal model. A single detector model can be shared by every experiment, for example if there were multiple crystals in a single shot. Alternatively, each experiment could reference a different detector model, perhaps taking account shot-to-shot variability in detector position due to a fluctuating injection system or from other jitter. Likewise, two experiments could each share a crystal model, perhaps having different beam models in the case of a two-color experiment, or perhaps having different detector models from two different detectors. This organization provides a flexible means of organizing a variety of possible experiment types and has already been used to jointly refine multiple lattices simultaneously from multiple crystals exposed during a single rotation series (Gildea *et al.* 2014, Waterman *et al.* 2016).

## 2. *cctbx.xfel* operational overview at LCLS

It has been our experience that analyzing data collected using serial crystallography (SX) typically requires three distinct processing stages labeled here **calibration**, **discovery** and **batch** (figure 1). **Calibration** refers to refining the geometry of the experiment, but also includes some pre-processing steps, such as creating dark and light averages, bad pixel masks and gain masks. Using these inputs, initial parameters are derived that describe the experiment, such as detector distance, quadrant and sensor layout, any beam correction parameters needed and so forth. During **discovery**, the user examines individual diffraction patterns and searches for appropriate parameters for data reduction, including hitfinding parameters if used, spotfinding parameters, target unit cell

### 1.4.4.2. Reflection table pickles

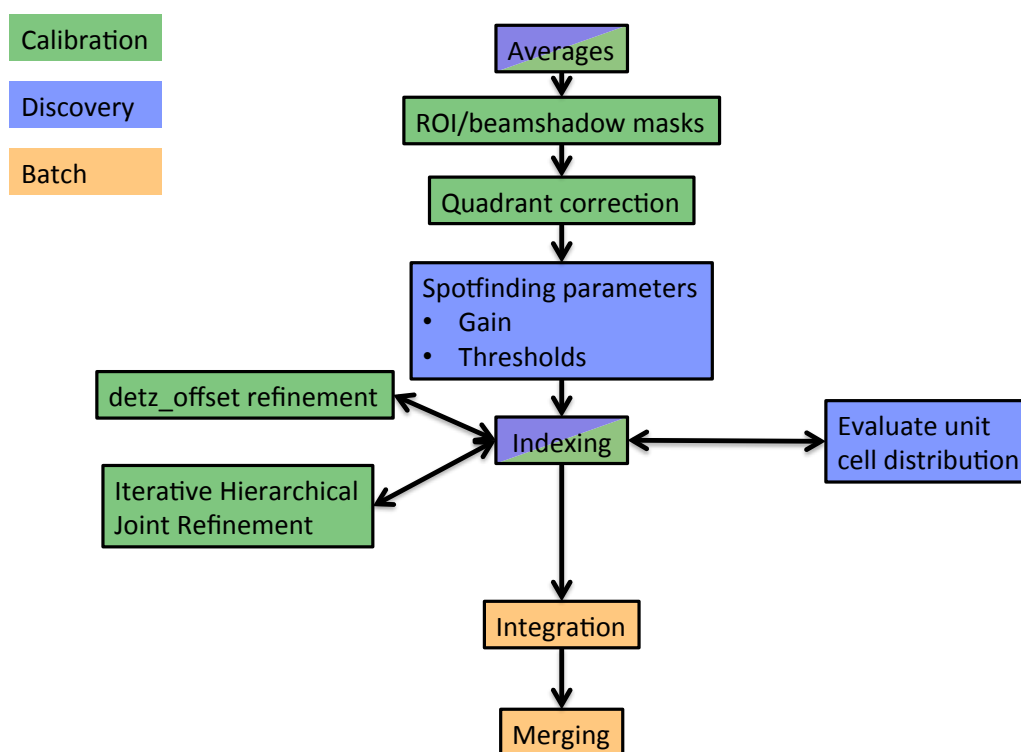
Spots found by spotfinding, indexed by indexing, or integrated during integration are recorded in reflection tables, where each spot is one row and the columns contain data such as Miller index, observed or predicted location relative to the panel origin in mm, summed intensity and variance, etc. These tables are serialized into python pickle files.

Both JSON experiment files and reflection table pickle files can be inspected with the command:

```
dials.show filename
```

Note that commands and parameters presented here will use the above formatting. Program names are shown in **bold**. Generally, additional documentation is available for each program listed with **-h**, **--help**, or **-c** (for configuration). Use **-e** to specify 'expert level' if desired (10 shows all parameters, or use 0, 1 or 2 for fewer parameters). Use **-a 2** to show documentation for each parameter.

dimensions and crystal symmetry and an optimal merging strategy. Finally, when optimal software configuration is established, the user enters **batch** processing mode, endeavoring to maximize the parallel computing options offered and, during live experiments, attempting to provide constructive feedback to beam line operators in as close to real-time as possible. After the experiment, the user will often need to reprocess the runs collected in batch mode. During batch processing, the user will continue to refine processing parameters as the results are evaluated, perhaps even revising initial metrology estimates. Thus the three stages are somewhat fluid as feedback from later stages may call for repeating earlier stages.



**Figure 1:** Operational overview at LCLS. The user will typically go through stages of processing: *calibration* (green), where detector geometry is optimized and pixel data is corrected and masked, *discovery* (blue), where the user determines sample-specific spotfinding, indexing, and integration parameters, and *batch* (orange), where large amounts of data are integrated prior to merging, either concurrent with the experiment for fast-feedback or after the experiment is completed. Several important data reduction tasks are shown in flowchart form, colored by the appropriate stage. Averaging and indexing are both useful for calibration and discovery, so they are shown with both colors.

LCLS organizes its data according to end-station name, experiment name and run number, under the global directory /reg/d/psdm. Thus, if a user was assigned experiment ID cxi84914, their data would be at /reg/d/psdm/cxi/cxi84914. The XTC streams will be in an xtc directory at that location, the calibration store, including for example geometry files and pedestals, will be in a directory named calib and the user will be able to store processing results temporarily in a scratch directory at that location, or more

permanently in a designated results directory named res. The magnitude of the data being collected — over 100 terabytes from a five-day experiment is not unusual — leads to a paradigm where the user never takes their raw data home. Instead, they reduce their data to integrated, merged and scaled intensities in MTZ format using the LCLS clustering system, then transfer that much smaller file to their home computer for downstream analysis.

### 3. Averaging CSPAD data

An important part of discovery and calibration involves averaging images together to get a sense of the behavior of the data. **cxi.mpi\_average** produces three images from a run: an average image, a standard

deviation image and a composite maximum image. In these images, every pixel is the average, standard deviation, or maximum of all pixels in that register over an entire run. Averages from a dark run, where the detector

is not exposed to X-rays, or from a light run, where the detector is exposed to X-rays, possibly in the presence of sample, are useful for several reasons. Light and dark images can be used to determine which pixels are not trusted. Inactive and non-bonded pixels, as well as hypersensitive pixels and any pixels shadowed by a beam stop should be masked out. The light images and the composite maximum in particular, also serve as a virtual, dark-subtracted powder pattern that can be used to inspect the data for diffraction. The light maximum should also flag obvious errors in metrology, because rings will be non-continuous if the quadrants are misaligned. Intensity values from the corners of the light average or maximum are also useful to estimate background when determining initial estimates of thresholds for spot-finding and integration (see below).

### 3.1. Example averaging commands

Use this command to average the CSPAD data:

```
cxi.mpi_average -x cxid9114 -r
95 -a CxiDs2.0:Cspad.0 -d 572
-v -g 6.85
```

The parameters are:

- x: experiment name
- r: run number
- a: detector address
- d: detz\_offset (defined below)
- v: verbose output
- g: gain ratio

The gain ratio is a constant multiplier for all low-gain pixels to be scaled to be at the same level as the high-gain pixels. When operating the CSPAD in mixed-gain mode, this ratio is needed to apply this correction. The value of 6.85 used here is not exact, but it is sufficient for this purpose.

The user may find it useful to average all the available data using the queuing system and to use multiple cores to reduce processing time. This can be done using a single command:

```
for i in `seq 95 114`; do bsub -n
12 -q psanaq -o avg_r$i.log
mpirun cxi.mpi_average -x
cxid9114 -r $i -a
CxiDs2.0:Cspad.0 -d 572 -v -g
6.85; done
```

The program **bsub** is used to submit jobs to the LCLS queuing system. The parameters are:

- n: number of processors
- q: queue name
- o: log file name

**mpirun** is used to enable inter-process communication so the averaging program can dispatch images to different computing cores and gather the results when complete. For more information, please see documentation from LCLS:

<https://confluence.slac.stanford.edu/display/PCDS/Submitting+Batch+Jobs>

### 3.2. Using averages to create an untrusted pixel mask

*cctbx.xfel* uses three images created during averaging to create a mask for the CSPAD detector. From the average image of a dark run, pixels with intensities  $\leq 0$  are considered dead, while intensities  $> 2000$  are flagged as hypersensitive. From the standard deviation of a dark run, pixels with intensities  $\leq 0$  are considered dead and intensities  $\geq 10$  are too uncertain and noisy. From the composite maximum from a lighted run (*i.e.* an experimental run), pixels with intensities  $< 300$  are considered non-bonded or in shadow. The presence of diffraction is not needed for the third image, but also will not interfere. The default of 300 here is unique among the numbers listed in that it will likely vary with the sample's background while the other defaults do not usually need to be changed. To tune this value, the user can carefully examine the corners of the lighted maximum projection and choose a value lower than the ADU values displayed. Also note that there is no cutoff on the high end of the maximum



projection image specified in the mask, as that is defined as the saturation value for the detector.

Here is an example command. Note that we have chosen 50 instead of 300 for the parameter maxproj-min after examining the data:

```
cxi.make_dials_mask --maxproj-  
min=50 -o mask.pickle  
cxid9114_avg-r0089.cbf  
cxid9114_stddev-r0089.cbf  
cxid9114_max-r0096.cbf
```

The output (specified with -o) is a detector mask usable by DIALS. It can be displayed using the image viewer, where masked out pixels are shown in red:

```
dials.image_viewer cxid9114_max-  
r0096.cbf mask=mask.pickle
```

## 4. Calibrating the CSPAD detector

As described above, the CSPAD consists of 32 2x1 sensors whose position in space must be refined. Before the user arrives at the XPP or CXI end-station, the beam line operator will have used an optical microscope to measure within each quadrant where the 2x1 sensors are located relative to one another and will have deployed these measurements as a starting metrology. While each of the 8 sensors within a quadrant will be well positioned relative to each other, the overall position of each quadrant relevant to the beam is usually not well determined. Initial quadrant positioning can be done using virtual powder rings from the average of many individual crystals, as described below. Average images with strong powder rings allow the user to determine the relative placement of each quadrant by aligning them such that the rings are contiguous. With these positions, initial indexing of a subset of data can produce measured and predicted Bragg reflection positions from which tiles are refined, minimizing the difference between observed and predicted spot locations from indexed diffraction data. The user re-indexes the data and re-refines the detector metrology until convergence is reached. It is recommended that high resolution, highly reproducible diffraction data from a reference set such as lysozyme or thermolysin be

collected for calibration and that the detector is positioned such that diffraction reaches its corners.

### 4.1. Manual quadrant calibration

Typically the best powder rings come from the composite maximum (example: the file cxid9114\_max-r0113.cbf will have been generated from the above averaging command). To manually align the quadrant positions, either use calibman (see LCLS documentation) or use **cctbx.image\_viewer** with the composite maximum. Under actions, click on 'Show quadrant calibration' and then use the spinners to align the powder rings. One may find the ring tool or the unit cell tool, also under the Actions menu, to be useful visual aids during this process. When done, click 'Save current metrology' to save the changes to a .def file, which is a CBF header.

### 4.2. Automatic quadrant calibration using *cctbx.xfel*

If a quadrant is properly positioned relative to the beam center, the pixel values for a strong powder pattern will be highly correlated after rotating the quadrant 45 degrees around the beam center. **cspad.quadrants\_cbf** performs a grid search of XY offsets for each quadrant,

searching for the position with the highest rotational autocorrelation. It then writes out a new CBF file with the adjusted header:

```
cspad.quadrants_cbf cxid9114_max-r0105.cbf
```

Specify the '-p' parameter to enable plots of the grid search results for each quadrant, in addition to reporting correlation coefficients (CCs) for each quadrant. The aligned image can be inspected with the image viewer:

```
cctbx.image_viewer cxid9114_max-r0105_cc.cbf
```

It is possible that a maximum of all the runs would have more contiguous and brighter rings, leading to higher CC values. This can be done quickly using the previously generated per-run composite maxima:

```
cxi.cspad_average *_max*.cbf -m all_max.cbf
```

Once a satisfactory maximum has been obtained, the quadrants tool can be called:

```
cspad.quadrants_cbf all_max.cbf
```

If the CC values are higher than when using individual run maxima, then this is a better approach for finding a good set of quadrant positions prior to initial indexing attempts.

### 4.3. Deploying new quadrant positions

Before the new quadrant positions can be used for indexing, the new layout needs to be converted to SLAC's metrology file format. Use this command if the quadrants were aligned manually using **cctbx.image\_viewer**:

```
cxi.cbfheader2slacalib cbf_header=quadrants.def
```

Or this command if the quadrants were aligned using **cspad.quadrants\_cbf**:

```
cxi.cbfheader2slacalib cbf_header=all_max_cc.cbf
```

The resultant file, 0-end.data, needs to be copied to the calibration store for the experiment. Typically, this will be in a directory in this form:

```
/reg/d/psdm/cxi/cxid9114/calib/CsPad::CalibV1/CxiDs2.0:Cspad.0/geometry
```

There will already be a 0-end.data file in this folder. We recommend renaming this file to 0-end.data.v0, copying the new 0-end.data to this folder under the name 0-end.data.v1, then soft-linking 0-end.data.v1 to 0-end.data. This will maintain a version history, as metrology is refined for this experiment.

### 4.4. Metrology Versioning

The user may find it useful to keep track of the improvement in metrology estimates using a versioning system. In this article, we version the metrology using the conventions in table 1.

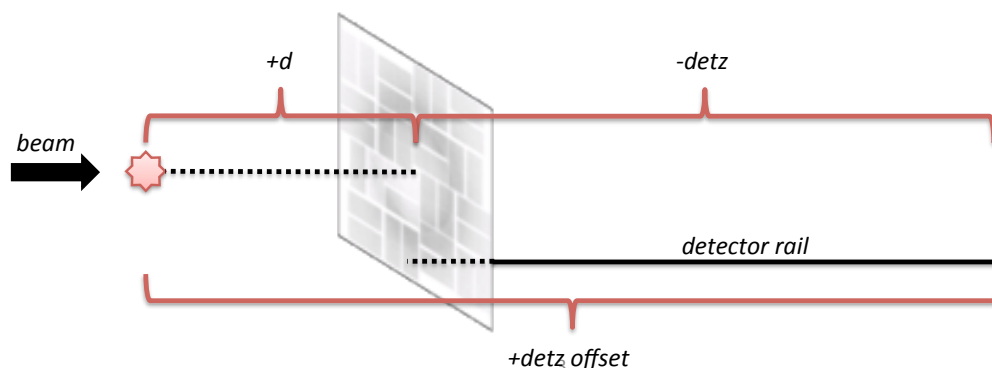
**Table 1: Metrology versioning**

Metrology version	Description
Version 0 (v0)	Initial metrology deployed by beam line operators. The tile positions are measured using an optical microscope, but, as the quadrants can move independently, they are not correctly aligned in relation to each other or to the beam center.
Version 1 (v1)	After collecting some data, virtual powder rings can be seen after averaging the events in a run. Quadrants are aligned by eye or automatically using <code>cspad.quadrants_cbf</code> .
Version 2 (v2)	After indexing the images using v1, the tile positions are refined to produce metrology v2.
Version 3 (v3)	After re-indexing the images using v2, the tile positions are re-refined to produce metrology v3.
...	And so forth until convergence.

#### 4.5. Detector distance and `detz_offset`

Each individual event in the XTC stream includes motor position settings for the detector (figure 2). The detector's position is measured from the back of the rail on which the CSPAD detectors at CXI moves to the detector itself. Call this distance `detz`. Naturally, the desired distance needed for crystallographic analysis is from the

detector's current position to the sample interaction region. For this reason, during processing it is necessary to supply an offset (`detz_offset`) from the sample interaction region to the back of the detector rail, a value that is constant over the course of a given experiment but changes between experiments when new injectors are substituted in and out of the sample chamber, a common occurrence.



**Figure 2:** Schematic of the `detz_offset` parameter. The CSPAD at the LCLS CXI endstation can be translated on a detector rail shown in black. The position of the CSPAD along this rail, `detz`, is determined from motor positions and recorded in the XTC stream for each event. As the sample injection system varies between users, the distance between the crystal and the back of the detector rail, the `detz_offset`, needs to be measured or experimentally refined. Then, the detector distance `d` can be determined for each event from the difference between the `detz_offset` and the `detz` parameter.

*cctbx.xfel* will compute the actual detector distance for each frame using the difference between *detz\_offset* and *detz*. *Detz\_offset* is available from the beam line operators but needs to be refined. See (Hattne *et al.* 2014), where *detz\_offset* is refined by screening a range of values and finding the *detz\_offset* that indexes the greatest number of images and see (Nass *et al.* 2016), where the detector distance is optimized by minimizing the standard deviation of the distribution of unit cell dimensions from P1 indexing trials at different distances. Importantly, eliminating multi-modal unit cell dimension distributions by using slightly different detector distances on a day-by-day or even run-by-run basis can increase the final quality of the data.

#### 4.6. Initial indexing

During indexing, the user parameterizes *cctbx.xfel* with settings regarding spotfinding, indexing, refinement and integration. An initial set of parameters needs to be established before indexing can be reliable. These parameters will be recorded in a PHIL file.

#### 4.7. Spotfinding

The most important parameter for the DIALS spotfinding algorithm is the gain, meaning the number of analog-to-digital units (ADU) per incident photons on a pixel, recorded on that pixel. The program *cctbx.xfel.xtc\_dump* can be used to dump CSPAD CBFs from an XTC stream to determine the flat gain value from many images at once. Here we use run 113, as its average image revealed the presence of strong diffraction:

```
cctbx.xfel.xtc_dump
dispatch.max_events=100
input.experiment=cxid9114
input.address=CxiDs2.0:Cspad.0
input.run_num=113
format.file_format=cbf
format.cbf.detz_offset=572
input.override_energy=8950
```

Then the program **dials.estimate\_gain** can be run on these files which will estimate a flat gain for each image gain based on the statistical distribution of reordered pixel values:

```
dials.estimate_gain
cspad_image.cbf
```

The final set of parameters for spotfinding is:

```
spotfinder {
  filter.min_spot_size=2
  threshold.xds.gain=25
  threshold.xds.global_threshold=100
}
```

The *min\_spot\_size* parameter specifies the minimum number of high-intensity pixels that must be present to classify the pixels as belonging to a spot. A value of 2 is appropriate for the CSPAD, which can record very small reflections. The *global\_threshold* parameter should not generally be needed for PAD data, but, here in mixed high/low gain mode, the flat gain estimate of 25 is not reliable for the entire detector. Because of this, we arbitrarily state that all pixels less than 100 ADU are background for spotfinding (but not for integrating).

The program **dials.image\_viewer** includes a mechanism for displaying which pixels will be included as signal for a given set of spotfinding parameters. This is a highly useful tool for estimating these parameters.

#### 4.8. Indexing

Initial parameters for the DIALS indexer include a target unit cell, an indexing method and a resolution cutoff:

```
indexing {
  known_symmetry {
    space_group = P43212
    unit_cell = 78.9 78.9 38.1 90 90
  }
  method=real_space_grid_search
  refinement_protocol.d_min_start=1.7
}
```

As lysozyme is well known, it is straightforward to assign known symmetry. In unknown cases, this can be left blank and the `fft1d` method can be used in place of `real_space_grid_search`. The resolution cutoff here is chosen to include the whole detector, but this may not be appropriate if the unit cell is not well known or if the resolution shown in the average composite is lower.

#### 4.9. Refinement

The DIALS refiner is used to optimize the experimental parameters after indexing. Refinement is performed after each indexing solution is determined for each diffraction pattern. Here, we change some settings in the parameterization block for the refiner:

```
refinement {
  parameterisation {
    beam.fix=all
    detector.fix_list=Dist,Tau1
    auto_reduction {
      action=fix
      min_nref_per_parameter=1
    }
    crystal {
      unit_cell {
        restraints {
          tie_to_target {
            values=78.9,78.9,38.1,90,90,90
            sigmas=1,1,1,0,0,0
          }
        }
      }
    }
  }
}
```

As these data were collected using seeded pulses instead of self-amplified stimulated emission (SASE) XFEL pulses, should have a constant energy, so we fix the beam parameters in place such that they are not refined. We also fix the detector distance (`dist`) and the rotation of the detector around the *z* axis. For SASE data, the user can consider allowing the detector distance to refine for each image by setting `detector.fix_list` equal to `Tau1` only. Regardless, either the beam model or the detector distance (if not both) should be fixed, as they are co-dependent.

When the refiner determines that there are too few observations to reliably refine a model, its behavior is determined by the `auto_reduction` parameters. Some models, such as the beam model, have few parameters, while others, such as the detector or crystal models, have many. If the number of reflections needed per parameter is set to zero, all parameters for all models will be refined regardless of how many reflections are available. Otherwise, if too few reflections are available to refine a given model, then one of three actions is taken: 1) Fail: refinement does not proceed and processing stops for this image. 2) Fix: the parameters for this model are fixed in place and not refined, but the reflections associated with it will still be used for other models if possible. 3) Remove: the model and all reflections associated with it will be removed from refinement; when refining a single still shot, this effectively means refinement will not occur for any of the models.

For stills, we find there are often not enough reflections to refine all the parameters (cell orientation, detector position, etc.) using the DIALS default minimum value of 5 reflections per parameter, because the default is optimized for a rotation experiment using a goniometer in



which there are many more reflections than there are in a single still shot. Here we set the minimum to 1, but 3 may also be a reasonable alternative, especially if the unit cell is larger. Poorly determined parameters will increase the root mean square deviation (RMSD) of the differences between observed and predicted reflections, which would affect the quality of the refined tile positions during metrology refinement later. However, during tile refinement, an image will be rejected if its overall positional RMSD is too high, so a minimum reflection count per parameter of 1 may be sufficient.

Finally, we restrain the unit cell parameters. The `tie_to_target` option uses a known set of cell dimensions and a set of weights for each dimension specified using the `sigma` parameter. Here, a `sigma` of 1 for the cell dimensions allows for some variation during refinement and is used for the edge lengths. For this orthorhombic space group we remove the restraint for the angles by setting those sigmas to zero.

#### 4.10. Integration

The *DIALS* integrator uses the stills-specific mosaic parameters estimated during integrating to predict which reflections are in the diffracting condition (Sauter *et al.* 2014). Additional parameters are shown here:

```
integration {
  profile.fitting=False
  background {
    algorithm = simple
    simple {
      model.algorithm = linear2d
      outlier.algorithm = plane
    }
  }
}
profile {
  gaussian_rs {
    min_spots.overall = 0
  }
}
```

We disable profile fitting for stills. For the background, the *DIALS* default algorithm, `glm`, is optimized for counting detectors such as

the Pilatus series from Dectris, which have a very low background. We have found this is not appropriate for the CSPAD, so we choose a “simple” linear 2d algorithm which estimates a gradient from nearby pixels for each reflection after fitting the background to a plane (Leslie 1999).

Lastly, even though we disable profile modeling, we instruct the profile modeler to accept the experiment even if the number of reflections is low (which is typically true for stills).

#### 4.11. XTC and CSPAD specific parameters

**cctbx.xfel.xtc\_process** is the program used to read the XTC streams, create CBF images in memory and then invoke the *DIALS* procedures. It needs to be parameterized in ways specific to the LCLS experiment:

```
input {
  address=CxiDs2.0:Cspad.0
}
format {
  file_format=cbf
  cbf {
    detz_offset=572.3938
    invalid_pixel_mask=mask.pickle
    gain_mask_value=6.85
    override_energy=8950
    common_mode.algorithm=custom
    common_mode.custom_parameterization=5,50
  }
}
border_mask {
  border=1
}
```

The address string identifies which CSPAD should be used to read data from in the XTC streams and can be obtained from the beam line operator. The format section specifies experiment-specific parameters to be written into the CBF file headers or to be used to correct the pixel data before adding it to the CBF main body. Here is where we specify the untrusted pixel mask created previously. The `detz_offset` parameter was chosen by indexing the data while letting the *z* axis refine, then creating a histogram of the distance values found for each image. Generally, however, the beamline operator’s estimate is a good initial

point to index enough images. A test of several different `detz_offset` values is another approach (Hattne *et al.* 2014).

For these seeded pulses the energy is known, so we override the energy value found for each shot in the XTC stream. For SASE data, the XTC stream contains an estimate of the overall energy for the pulse and should be used, meaning that the user should leave this field blank. Finally, LCLS provides a variety of common mode correction algorithms (see <https://confluence.slac.stanford.edu/display/PSDM/Common+mode+correction+algorithms>). The common mode is a per shot, per sensor small offset on the order of 10 ADU that occurs due to changes in potentials during readout of the sensors. Determining it for protein crystallography, which generally has a high background due to solvent scattering, is difficult. If `common_mode.algorithm` is left unspecified in the PHIL file, no correction will be applied. The user may otherwise specify 'default' for this option, in which case the current default LCLS corrections will be

applied. Currently this is algorithm 1, a pixel histogramming method applicable to weak signal and likely not applicable to protein crystallography with strong solvent scattering. The user can also specify 'custom' and pick an algorithm from the algorithms described at the above link. Algorithm 5, selected in this PHIL file, uses non-bonded pixels, i.e. pixels not bump bonded to the electronics in a given ASIC and specifies a maximum correction applied to each pixel of 50 ADU. Currently, we can offer no advice as to which is best as it is a matter of active research. Generally, we have been processing with no correction, leaving the `common_mode` parameters blank.

Finally, we specify a border mask of 1 pixel for each of the 64 tiles, because the wider edge pixels of the CSPAD are not on the same ADU scale as the rest of the pixels in the ASIC.

#### 4.12. Final PHIL file for initial indexing

Putting it all together, the initial PHIL file for indexing will look like this:

```
input {
  address=CxiDs2.0:Cspad.0
}
format {
  file_format=cbf
  cbf {
    detz_offset=572.3938
    invalid_pixel_mask=mask.pickle
    gain_mask_value=6.85
    override_energy=8950
    common_mode.algorithm=custom
    common_mode.custom_parameterization=5,50
  }
}
border_mask {
  border=1
}
spotfinder {
  filter.min_spot_size=2
  threshold.xds.gain=25
  threshold.xds.global_threshold=100
}
```

```

indexing {
  known_symmetry {
    space_group = P43212
    unit_cell = 78.9 78.9 38.1 90 90 90
  }
  method=real_space_grid_search
  refinement_protocol.d_min_start=1.7
}
refinement {
  parameterisation {
    beam.fix=all
    detector.fix_list=Dist,Tau1
    auto_reduction {
      action=fix
      min_nref_per_parameter=1
    }
    crystal {
      unit_cell {
        restraints {
          tie_to_target {
            values=78.9,78.9,38.1,90,90,90
            sigmas=1,1,1,0,0,0
            apply_to_all=True
          }
        }
      }
    }
  }
}
integration {
  integrator=stills
  profile.fitting=False
  background {
    algorithm = simple
    simple {
      model.algorithm = linear2d
      outlier.algorithm = mosflm
    }
  }
}
profile {
  gaussian_rs {
    min_spots.overall = 0
  }
}

```

### 4.13. Indexing commands

Indexing in *cctbx.xfel* is typically done in a series of trials. Our first trial will be trial 0, using metrology v1 (initial metrology from beam line operators, with quadrants corrected using one of the above techniques). The program **cximpi\_submit** will submit the indexing command to the queuing system. LCLS's LSF queue is supported and SGE and custom queuing commands are also available. Please contact the authors for advice on running the software on any queuing systems not yet supported.

We generally organize our work into numbered 'trials', where each trial represents an experimental set of parameters. The program **cximpi\_submit** creates a directory for the trial, copies all config and PHIL files as backups and submits the processing job to the LCLS cluster. For example, to do the initial indexing trial for this data, we use the command:

```
for i in `seq 95 114`; do
  cximpi_submit
  input.experiment=cxid9114
  output.output_dir=/reg/d/psdm/
  cxi/cxid9114/ftc/brewster/dial
  s mp.nproc=36 mp.queue=psanaq
  output.split_logs=True
  input.dispatcher=cctbx.xfel.xt
  c_process input.target= LD91-
  lyso-t000.phil input.trial=0
  input.run_num=$i
  dispatch.integrate=False; done
```

This for loop in bash submits runs 95 through 114, inclusive, for processing. The experiment name is specified to allow *psana* to find the XTC streams with the data. In the output directory, a run directory for each run will be created and under that a three-digit trial directory (named 000 for trial 0) will be created. Logs will be saved to a stdout directory in the trial directory. Here, we use split logs, which means that in addition to the main log file, each of the 36 processors requested here will write to a separate log file

so as not to interleave all the output from all the processors. The queue to use is specified as *psanaq*, the public queue available at LCLS. Other queues are available, as described here: <https://confluence.slac.stanford.edu/display/PCDS/Submitting+Batch+Jobs>. The dispatcher refers to which XTC processing program to run; **cctbx.xfel.xtc\_process** invokes the DIALS processing pipeline. The other program available, **cxixtc\_process**, invokes the LABELIT backend as described elsewhere (Hattne *et al.* 2014). The PHIL file for processing is specified as a target. Finally, to save time during initial indexing and metrology refinement (the discovery phase), we use `dispatch.integrate=False` to skip the integration step.

After the job completes, indexing results will be available in the results folder for each run, under trial 000 in a folder named out. For each indexed image a CBF file will be created. In addition, the files `refined_experiments.json` and `indexed.pickle` will be created, containing information about the indexing solution and the list of indexed reflections, respectively. **dials.show** is useful for displaying some summary information about the contents of these files and **dials.image\_viewer** can be invoked with a CBF file and an `indexed.pickle` file to visualize the indexed reflection positions overlaid on the image data.

**cctbx.xfel.xtc\_process** provides user control over which steps in the spotfinding, indexing and integration are executed. For example, the user could dump all images with strong reflections to CBF whether or not they indexed using `dispatch.dump_strong=True`. Use the `-c` (configuration) parameter to show the full set of options available.

### 4.14. Refinement of tile positions

After initial indexing results are obtained, the program **cspad.cbf\_metrology** is used to refine the tile positions of the CSPAD detector. This program aggregates many individual indexing results to do a joint refinement of many crystal models and a single detector

model using the differences between the observed spot locations and predicted spot locations as a target function. In brief, the steps taken are:

Use **dials.combine\_experiments** to concatenate a number of indexing results into a single combined\_experiments.json file and a single combined\_reflections.pickle file. During this step, each crystal model from each experiment is retained separately, while the detector positions are averaged together to create a single detector model.

- Filter the set of indexing solutions, rejecting images with an RMSD high enough to be considered an outlier using Tukey's rule of thumb.
- Refine the detector as a whole, including rotation and tilt, using **dials.refine**.
- Filter individual images again by RMSD.
- Refine each quadrant separately using **dials.refine**.
- Filter individual images again by RMSD.
- Refine each 2x1 sensor separately using **dials.refine**.
- Convert the final refined\_experiments.json file into a CBF header.
- Convert the CBF header to SLAC format, ready to deploy as a new metrology file.

Use a command like this one to refine the metrology, assuming the results folder is in a directory at the same level as the current folder:

```
cspad.cbf_metrology tag=t000_2k
  ../results/r0*/000/out
  n_subset=2000
  split_dataset=True cxd9114-
  refine.phil
```

The tag is prepended to every output file name. `n_subset` selects a random group of images for refinement from the input set of folders. `split_dataset` instructs the program to perform the refinement twice with a different random subset of 2000 each time. This is useful to verify the precision of the refinement, meaning how likely it is for the refinement to produce the same detector metrology given a different set of indexing

results. The PHIL file provided parameterizes the `dials.refine` steps. Here we only specify a light restraint to the target unit cell as described above:

```
refinement {
  parameterisation {
    crystal {
      unit_cell {
        restraints {
          tie_to_target {
            values=78.9,78.9,38.1,90,90,90
            sigmas=1,1,1,0,0,0
            apply_to_all=True
          }
        }
      }
    }
  }
}
```

Depending on the quality of the calibration dataset, the user may decide to forgo filtering individual images with high RMSD out of the dataset. Use `rmsd_filter.enable=False` to do this. An additional parameter can specify which set of reflections to use during metrology refinement. `reflections=indexed` is the default and specifies using only indexed, strong reflections. Refinement can also be done against integrated reflections. During indexing, `dispatch.reindex_strong` can be used to re-index the bright reflections found during spotfinding that weren't matched during indexing. Then, during metrology refinement, `reflections=reindexed_strong` can be used to include more strong reflections than just the ones from initial indexing. This provides more data to use during refinement without adding weak reflections from integration, which may have poorly determined centroids.

Finally, the user can use two different methods of selecting the subset of data to use for refinement. The default method picks randomly from the available images to create a dataset with which to refine metrology. Alternatively, the user can specify `n_subset_method=n_refl` to instead pick the subset with the greatest number of reflections per image to increase the likelihood of measuring diffraction to the corners of the detector.



The final output file, 0-end.data.t000\_2k\_1 in this example, is ready to use as a new version of the metrology (here, version 3, or v3). Deploy it to the calibration folder as described above.

#### 4.15. Evaluating metrology convergence

After deploying the metrology, it is recommended that the user repeat indexing and metrology refinement until tile positions converge, meaning until the tile positions are no longer changing significantly and the quality of the positional RMSDs is no longer improving. Two programs are available to assist in evaluating this: **cspad.detector\_shifts** and **cspad.detector\_statistics**.

**cspad.detector\_shifts** can be used to see changes in panel positions during refinement:

```
cspad.detector_shifts
t000_2k_1_filtered_experiments
.json
t000_2k_1_filtered_reflections
.pickle
t000_2k_1_filtered_experiments
_level2.json
t000_2k_1_filtered_reflections
_level2.pickle
```

This test compares the unrefined metrology (first two files) with the results after sensor refinement (level 2). The CSPAD has four levels of metrology available to refine, level 0 (detector as a whole), level 1 (4 quadrants), level 2 (32 2x1 sensors) and level 3 (64 individual ASICs). As the ASICs are physically connected, level 3 need not be refined. Some example output could look like this:

Hierarchy Level (microns)	Delta XY (microns)	Delta XY Sigma
0	6.2	0.0
1	78.1	53.2
2	14.5	11.4
3	0.0	0.0

This output reveals that the detector as a whole (level 0) moved 6 microns during refinement in the XY plane (orthogonal to the beam). Each quadrant moved 78+/-53 microns and each sensor moved 14.5 ± 11.4 microns. As the quadrants are the least well determined, it is reasonable their positions should move the most. The full output contains more information, such as z offsets and rotations.

Use **cspad.detector\_statistics** to evaluate the precision of refinement. This program compares the two independent refinements performed by **cspad.cbf\_metrology** when `split_dataset=True`.

```
cspad.detector_statistics
tag=t000_2k
```

This program produces a lot of output, comparing tile positions at each level of the detector hierarchy. We will focus on a few of these statistics.

Each refinement job, two in this case, produces one measurement of the position of any given CSPAD tile. The F offset sigma and S offset sigma values report weighted standard deviations of the two measurements of the each sensor's fast (F) or slow (S) coordinate. These values are reported for each 2x1 sensor and after several rounds of indexing and refinement using monochromatic data from this experiment, the overall weighted averages of the fast and slow offset sigmas were 1.6 and 1.7 μm, respectively, indicating the overall precision of this refinement was around 1 μm. This is much smaller than the pixel size of the CSPAD, 110 μm.

Also reported is the observational RMSD of the differences between observed and predicted reflection positions, which for this experiment was 38.3 μm for the first split dataset and 37.5 μm for the second. In all cases, weighted averages or standard deviations are computed using the number of reflections observed as the weight.

## 5. Batch integration, scaling and merging of XFEL data

After metrology refinement is complete, the user can process data in batch mode using **cximpi.submit** as described above, using `dispatch.integrate=True`. During integration, in addition to `experiment.json` and `integrated.pickle` files in DIALS format, the integration results are also stored in *cctbx.xfel* integration files using a convention of `int-0-`

`<timestamp>.pickle`. The files are the direct inputs to scaling and merging programs available through *cctbx.xfel*, namely **cximerge** (Sauter 2015) and **prime.postrefine** (Uervirojnangkoorn *et al.* 2015). Directions for the use of these programs are provided on the *cctbx.xfel* wiki (<http://cci.lbl.gov/xfel>).

## 6. Processing serial crystallographic data from other sources

The main program used for processing data presented here, **cctbx.xfel.xtc\_process**, is an interface between the LCLS system *psana* used for writing and reading XTC streams, the libraries in *cctbx.xfel* needed for creating and using CSPAD CBF files and the stills-specific algorithms implemented in DIALS for indexing, refinement and integration. Users collecting serial crystallographic data at synchrotron sources on new, high performance detectors such as the Eiger by Dectris have the option of using these same

algorithms for still shots with the program **dials.stills\_process**. This program accepts a PHIL file as described above, with the exception that the format section is omitted, as the file headers are used directly to assemble the appropriate detector, beam and crystal models. The program can be used on a cluster using **mpirun** if the `mp.method` parameter is set to `mpi`. For further detail, use the `-c` parameter as described above to see available options.

## 7. Future directions and software availability

*cctbx.xfel* is installed for all users at LCLS in `/reg/g/cctbx`. Instructions for setting up the appropriate environment for its use are available at <http://cci.lbl.gov/xfel>. *cctbx.xfel* bundled with DIALS is completely open source and is distributed through SourceForge and Git (see installation instructions for standalone packages at <http://dials.github.io>).

A graphical user interface (GUI) for processing XFEL data at LCLS using *cctbx.xfel* has been developed. Users interested in helping beta test the interface are invited to contact the authors. Further, example data useful for practicing the commands presented here are available on request.

## 8. Acknowledgements

This work was supported by NIH grants GM102520 and GM117126 for data processing methods (N.K.S.). This research was supported in part by the European Community's Seventh Framework Programme (FP7/2007–2013) under BioStruct-X (grant agreement No. 283570).

## 9. References

Andrews LC and Bernstein HJ (2014). "The geometry of Niggli reduction: BGAOL–embedding Niggli reduction and analysis of boundaries." *J. Appl. Crystallogr.* **47**: 346–359.

- Bernstein HJ and Hammersley AP (2006). *Specification of the Crystallographic Binary File (CBF/imgCIF), International Tables for Crystallography Volume G: Definition and exchange of crystallographic data*. Netherlands, Springer.
- Bourhis LJ, Grosse-Kunstleve RW and Adams PD (2007). "cctbx news." *IUCr Comput. Comm. Newsl.* **8**: 74-80.
- Brewster AS, Hattne J, Parkhurst JM, Waterman DG, Bernstein HJ, Winter G and Sauter NK (2014). "XFEL Detectors and ImageCIF." *Computational Crystallography Newsletter* **5**: 19-24.
- Bricogne G (1986). "Proc. of the EEC Cooperative Workshop on Position-Sensitive Detector Software (Phase III), p. 28. Paris: LURE.
- Campbell JW (1998). "The Practicality of Using a Three-Dimensional Fast Fourier Transform in Auto-Indexing Protein Single-Crystal Oscillation Images." *J. Appl. Crystallogr.* **31**: 407-413.
- Chollet M, Alonso-Mori R, Cammarata M, Damiani D, Defever J, Delor JT, Feng Y, Glowia JM, Langton JB, Nelson S, Ramsey K, Robert A, Sikorski M, Song S, Stefanescu D, Srinivasan V, Zhu D, Lemke HT and Fritz DM (2015). "The X-ray Pump-Probe instrument at the Linac Coherent Light Source." *J. Synch. Rad.* **22**: 503-507.
- Cohen AE, Soltis SM, Gonzalez A, Aguila L, Alonso-Mori R, Barnes CO, Baxter EL, Brehmer W, Brewster AS, Brunger AT, Calero G, Chang JF, Chollet M, Ehrensberger P, Eriksson TL, Feng Y, Hattne J, Hedman B, Hollenbeck M, Holton JM, Keable S, Kobilka BK, Kovaleva EG, Kruse AC, Lemke HT, Lin G, Lyubimov AY, Manglik A, Mathews, II, McPhillips SE, Nelson S, Peters JW, Sauter NK, Smith CA, Song J, Stevenson HP, Tsai Y, Uervirojnangkoorn M, Vinetsky V, Wakatsuki S, Weis WI, Zadvornyy OA, Zeldin OB, Zhu D and Hodgson KO (2014). "Goniometer-based femtosecond crystallography with X-ray free electron lasers." *Proc. Natl. Acad. Sci. USA* **111**: 17122-17127.
- Damiani D, Dubrovin M, Gaponenko I, Kroeger W, Lane TJ, Mitra A, O'Grady CP, Salnikov A, Sanchez-Gonzalez A, Schneider D and Yoon CH (2016). "Linac Coherent Light Source data analysis using psana." *J. Appl. Crystallogr.* **49**: 672-679.
- Gildea RJ, Waterman DG, Parkhurst JM, Axford D, Sutton G, Stuart DI, Sauter NK, Evans G and Winter G (2014). "New methods for indexing multi-lattice diffraction data." *Acta Crystallogr. D* **70**: 2652-2666.
- Grosse-Kunstleve RW, Afonine PV, Sauter NK and Adams PD (2005). "cctbx news: Phil and friends." *IUCr Comput. Comm. Newsl.* **5**: 69-91.
- Grosse-Kunstleve RW, Sauter NK, Moriarty NW and Adams PD (2002). "The *Computational Crystallography Toolbox*: crystallographic algorithms in a reusable software framework." *J. Appl. Crystallogr.* **35**: 126-136.
- Grosse-Kunstleve RW, Zwart PH, Afonine PV, Ioerger TR and Adams PD (2006). "cctbx news." *IUCr Comput. Comm. Newsl.* **7**: 91-104.
- Hart P, Boutet S, Carini G, Dubrovin M, Duda B, Fritz D, Haller G, Herbst R, Herrmann S, Kenney C, Kurita N, Lemke H, Messerschmidt M, Nordby M, Pines J, Schafer D, Swift M, Weaver M, Williams G, Zhu D, Van Bakel N and Morse J (2012). "The CSPAD megapixel X-ray camera at LCLS." *Proc. of SPIE* **8504**: 85040C.
- Hattne J, Echols N, Tran R, Kern J, Gildea RJ, Brewster AS, Alonso-Mori R, Glöckner C, Hellmich J, Laksmono H, Sierra RG, Lassalle-Kaiser B, Lampe A, Han G, Gul S, DiFiore D, Milathianaki D, Fry AR, Miahnahri A, White WE, Schafer DW, Seibert MM, Koglin JE, Sokaras D, Weng T-C, Sellberg J, Latimer MJ, Glatzel P, Zwart PH, Grosse-Kunstleve RW, Bogan MJ, Messerschmidt

- M, Williams GJ, Boutet S, Messinger J, Zouni A, Yano J, Bergmann U, Yachandra VK, Adams PD and Sauter NK (2014). "Accurate macromolecular structures using minimal measurements from X-ray free-electron lasers." *Nat. Methods*.
- Kameshima T, Ono S, Kudo T, Ozaki K, Kirihara Y, Kobayashi K, Inubushi Y, Yabashi M, Horigome T, Holland A, Holland K, Burt D, Murao H and Hatsui T (2014). "Development of an X-ray pixel detector with multi-port charge-coupled device for X-ray free-electron laser experiments." *Rev. Sci. Instrum.* **85**: 033110.
- Kern J, Tran R, Alonso-Mori R, Koroidov S, Echols N, Hattne J, Ibrahim M, Gul S, Laksmono H, Sierra RG, Gildea RJ, Han G, Hellmich J, Lassalle-Kaiser B, Chatterjee R, Brewster AS, Stan CA, Glockner C, Lampe A, DiFiore D, Milathianaki D, Fry AR, Seibert MM, Koglin JE, Gallo E, Uhlig J, Sokaras D, Weng TC, Zwart PH, Skinner DE, Bogan MJ, Messerschmidt M, Glatzel P, Williams GJ, Boutet S, Adams PD, Zouni A, Messinger J, Sauter NK, Bergmann U, Yano J and Yachandra VK (2014). "Taking snapshots of photosynthetic water oxidation using femtosecond X-ray diffraction and spectroscopy." *Nat. Comm.* **5**: 4371.
- Leslie AGW (1999). "Integration of macromolecular diffraction data." *Acta Crystallogr. D* **55**: 1696-1702.
- McGill KJ, Asadi M, Karakasheva MT, Andrews LC and Bernstein HJ (2014). "The geometry of Niggli reduction: SAUC- search of alternative unit cells." *J. Appl. Crystallogr.* **47**: 360-364.
- Nass K, Meinhart A, Barends TR, Foucar L, Gorel A, Aquila A, Botha S, Doak RB, Koglin J, Liang M, Shoeman RL, Williams G, Boutet S and Schlichting I (2016). "Protein structure determination by single-wavelength anomalous diffraction phasing of X-ray free-electron laser data." *IUCrJ* **3**: 180-191.
- Parkhurst JM, Brewster AS, Fuentes-Montero L, Waterman DG, Hattne J, Ashton AW, Echols N, Evans G, Sauter NK and Winter G (2014). "dxtbx: the diffraction experiment toolbox." *J. Appl. Crystallogr.* **47**: 1459-1465.
- Sauter NK (2015). "XFEL diffraction: developing processing methods to optimize data quality." *J. Synch. Rad.* **22**: 239-248.
- Sauter NK, Grosse-Kunstleve RW and Adams PD (2004). "Robust indexing for automatic data collection." *J. Appl. Crystallogr.* **37**: 399-409.
- Sauter NK, Hattne J, Brewster AS, Echols N, Zwart PH and Adams PD (2014). "Improved crystal orientation and physical properties from single-shot XFEL stills." *Acta Crystallogr. D* **70**: 3299-3309.
- Sauter NK, Hattne J, Grosse-Kunstleve RW and Echols N (2013). "New Python-based methods for data processing." *Acta Crystallogr. D* **69**: 1274-1282.
- Steller I, Bolotovskiy R and Rossmann MG (1997). "An Algorithm for Automatic Indexing of Oscillation Images using Fourier Analysis." *J. Appl. Crystallogr.* **30**: 1036-1040.
- Uervirojnangkoorn M, Zeldin OB, Lyubimov AY, Hattne J, Brewster AS, Sauter NK, Brunger AT and Weis WI (2015). "Enabling X-ray free electron laser crystallography for challenging biological systems from a limited number of crystals." *eLife* **4**.
- Waterman DG, Winter G, Gildea RJ, Parkhurst JM, Brewster AS, Sauter NK and Evans G (2016). "Diffraction-geometry refinement in the DIALS framework." *Acta Crystallogr. D* **72**: 558-575.
- Waterman DG, Winter G, Parkhurst JM, Fuentes-Montero L, Hattne J, Brewster A and Sauter NK (2013). "The DIALS framework for integration software." *CCP4 Newsletter on Protein Crystallography* **49**: 16-19.
- Zeldin OB, Brewster AS, Hattne J, Uervirojnangkoorn M, Lyubimov AY, Zhou Q, Zhao M, Weis WI, Sauter NK and Brunger AT (2015). "Data Exploration Toolkit for serial diffraction experiments." *Acta Crystallogr. D* **71**: 352-356.