ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-TUCPR01

DEVELOPING A TOOLKIT FOR ANALYSIS OF LCLS PUMP-PROBE DATA*

S. Nelson†, SLAC National Accelerator Laboratory, Menlo Park, CA, USA

Abstract

The data format and volume at LCLS requires significant computing expertise which not all user groups can provide. We will describe the path to and current status of a Python module that enables user groups to translate and reduce their data into a format that they can easily work with. The package is developed in Python and uses the standard LCLS data analysis framework. It encapsulates knowledge of the standard beam line components and adds convenient ways to reduce the data of larger detectors. Both an event-based (best for small event sizes) and a binned approach which is able to handle larger data as megapixel size detectors are simple to setup. MPI is used for fast turn around, enabling close to real time feedback necessary to make decisions of how to use the limited amount of beam time. Jupyter notebooks are provided to demonstrate some of the available options and can serve as a convenient quick start for fast turn around analysis.

ANALYSIS ENVIRONMENT AT LCLS

The requirements for the LCLS data acquisition system (DAQ) are that within each instrument, data are acquired for all devices at the beam rate of 120 Hz, tagged with the fiducial from the timing system and a UNIX timestamp. These event data are then appended to a file in eXtended tagged container (XTC) format [1]. The DAQ system is capable of reading out 5 GB/s per instrument and is described in more detail in [2].

The DAQ software was written in C++ which was also the language of the the initial analysis framework. While the learning curve to do analysis in a complex framework is standard in HEP where scientists join a collaboration and will work in a given framework for a few years up to decades, light sources are user facilities where typical users are only present for a short amount of time for beamtime that has been assigned to one of their proposals.

In order to allow groups to analyze their data without having to develop code in C++, an hdf5 [3] translation system was set up. This is accessible from the “Data Manager”, the main interface to the experiment. Having the data in hdf5 format allows user groups to use a variety of programs that they are already familiar with, e.g. MATLAB, Igor or python. The translated files mirrored the structure of the original xtc files, which is not very intuitive. Each component of the data contains the values and timestamps and the user must perform timestamp matching when analyzing the data. The data had to be read sequentially in a single thread. For longer runs with mega pixel detectors, the datawas only be available for analysis a few hours after data taking.

Some of the earlier user groups had sufficient resources to develop frameworks for analysis based on the SLAC code stack and the detectors typically used. In particular for compute intensive analyses like crystallography a couple of different analysis frameworks were developed that are used to date. For other communities where the analysis was typically more less computationally challenging, this was not the case. Some groups developed analysis setups based on the LCLS analysis framework while other groups based their analysis on MATLAB code provided by beam line staff which was based on the standard translation provided by LCLS.

Hidden costs of user frameworks became visible during years of operations: the first is a technical cost for the facility when the user code needs to be adjusted for a new beam time or in rare cases, even for analysis of old data. Such changes are necessary when the LCLS analysis stack changes significantly, when common beam line components are upgraded, new physics detectors are used or a user group moves to a different experimental end station. In addition, the barriers for small, new user groups can be high if the analysis system provided for them requires more computing expertise than they have available. These groups will either not propose their experiment, feel the need to include other groups who have an LCLS-compatible analysis setup or take a long time to publish their data after the beam time assuming that their experiment allowed the LCLS-provided live analysis framework to work well enough so that the measurement plan could be followed.

REQUIREMENTS FOR GENERAL USER SUPPORT ANALYSIS PACKAGE

The requirements listed below have been taken from a few years of supporting analysis groups at XPP [4] and XCS [5] in particular, but they are common for other experiments conducted at other hutches. Crystallography experiments will typically use a different code stack:

- portable data (hdf5), small with timestamp aligned data
- no setup for standard beam line data
- simple interface to data reduction methods for larger data sources.
- option to keep full data of big detectors, but average multiple events
- data ready for analysis within a few minutes after the run has ended

The needs of staff that is supporting a significant number of user experiments during a year adds a few more requirements.

* This work was performed in support of the LCLS project at SLAC supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515
† † snelson@slac.stanford.edu

- consistent names for detectors across experiments
- standard feature extraction methods should have single code implementation
- production code should have known behavior
- interfaces to standard data analysis and/or calibration methods

SMALLDATA_TOOLS

smalldata_tools [6] is a python package developed to fulfill as many of these requirements as possible. It is maintained by LCLS staff involved in beam line operation and has been used during the last few years of operations for more than half of the experiments in the XPP and XCS hutches. In the following, we will describe the main concepts of smalldata_tools. MPI is used to allow to use many cores to make the production run in real time with a latency due to the file transfer time. The result hdf5 files are referred to as 'smallData' files in the following.

Default Data

Data taken at LCLS during its first 10 years has 66 different sources of data and typical experiments contain between 10 and 20. Many of these data sources are found in most experiments at a given beamline or even at all LCLS experiments. smalldata_tools will save the data from the typically used sources for a given hutch automatically in the smallData file. As the relevant data from those sources are typically small for each event, the overhead of deciding if a data source will be needed and modifying the user code to extract the information can be skipped by automatically saving that data. In addition, by having this data provided to the user automatically, we can enforce standard names for data and replace technical details of data taking procedures by the intended result. An example for that is the conversion of the presence of instrument specific events codes in the trigger object by a simple set of boolean variables that indicate the presence of X-ray and optical laser beam in the event.

Experiment Specific Data Reduction

In addition to the beamline diagnostic data, user experiments often use megapixel cameras to measure their signal. While saving the full data is technically possible, there are associated costs both in production compute resources, disk space and memory needs for analysis of these large files. smalldata_tools offers a simple interface to reduce the data from these cameras, called 'DetObject'. This class is instantiated by passing the detector name and optionally the optimal method for calibration based on the physics signal. One can then add methods to this instance, the results of which will be saved for each event in the smallData file. In addition, the parameters of this data reduction are saved as well as the detector calibration information used. Table 1 lists the currently implemented data reduction method and their results. The following sub section will discuss each method in a little more detail.

Table 1: Data Reduction Methods Offered by smalldata_tools

Method Name	Output
ROI	sum, center-of-mass signal of the highest pixel (optionally all pixels)
Binning	binned version of the data
Image	2d data of tiled detectors
Projection	lower dimensional data
Azimuthal average	data in q-φ bins
Droplets	number of droplets (optional) spectrum of droplets (optional) arrays of energy and coordinates for droplets
Photons	number of found photons (optional) full or sparse photon image

ROI This reduction method is typically used if the signal of interest is limited to a small number of pixels or if the substructure is not important. They can also used used as a quick proxy in fast feedback analyses for a more setup sensitive reduction method. ROIs are rectangular sub-arrays of the original data. This limitation was chosen to allow for simple storage of the full ROI data if requested. If a sum of the differently shaped ROI is optimal, masks can be applied.

Binning If the full resolution of a detector image is not required for the analysis, it is possible to reduce the data size by re-binning the image to a smaller array size. ndimage.zoom from scipy [7] is used.

Projections In the most simple forms, data is summed over the axes not of interest. It is also possible to provide a map of each pixel onto a different coordinate system into which the data will then be projected. This is the base of the azimuthal averaging.

Azimuthal Averaging This technique projects the data into a q-φ space. q is calculated from the beam energy, the detector-sample distance and the distance of a pixel from the beam center. Geometric corrections are applied and it is possible to also correct for the effect of the X-ray polarization.

Droplets The droplet algorithm uses a two threshold peak finding method. Thresholds can be defined in either readout units or in number of the pixel noise as measured in the calibration run. The higher threshold is used to seed the droplets in the first run of skimage.measure to find the droplet candidates. Droplets contain at least one pixel above threshold and all its direct neighbors which also pass the threshold. In a second step, all pixels directly neighboring candidate pixels passing a lower threshold are also used. Having two separate thresholds allows to rejection of candidates most likely from noise by using a high seeding threshold while allowing to recover energy resolution from charge shared by neighboring pixels passing only a lower threshold.

Photon Finding This algorithm returns an image in number of photons, trying to account for charge sharing between neighboring pixels by allowing two pixels to contribute to one photon. Parameters are the expected measured signal of a photon and the minimal threshold the photon candidates from the combined pixels have to pass. It is described in detail in [8].

Tools to Set Parameters for the Data Reduction We provide tools in an ipython environment that allows users to set the extraction parameters from their data by the use of an 'average image' that shows the features of question as shown in Fig. 1. By default, the first 100 events are used, but the number of events to sum as well as the number of events to skip can be adjusted. A threshold can be applied to the images before summation and it is also possible to apply an filter comprised of a set of square cuts to the events before before they are added to the average image.

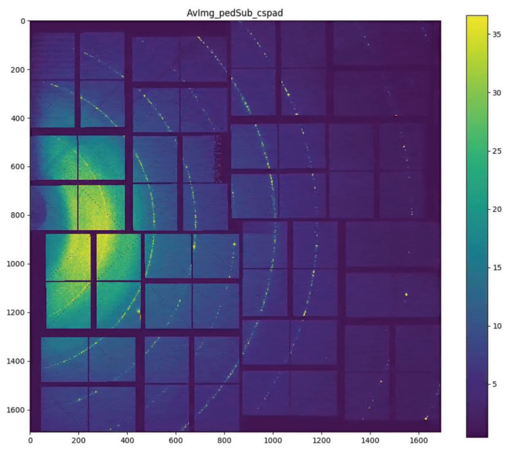


Figure 1: Example image for an average image from an X-ray detector.

Based on this method, we provide a tool to select ROIs by selecting corner points of the desired region on the image with the mouse. We also provide different methods to extract the beam center, the resulting image of one of them is shown in Fig. 2. Masks and ROIs can be created interactively from the 'average image' as well.

EVENT BINNING

In addition to the production tools, we offer simple commands to plot variables as histograms or the correlation of two variables as scatter plot or a 2-d histogram. Event selections can be presented as a series of simple square cuts with a name, they can be used for all methods that will select data. All information used in the smallData file can be used and the users can create new variables based on the information present and use those results in the event filter or for the binning. The binning method is called "cube", derived from the picture of the data of an area detector in many slices of a given binning variable, thus building a rectangular prism. A "cube" is defined by a name, the binning variable and the requested bins as well as the event filter. It is possible to add as many additional binning variables as desired. The variables from the smallData file that should be binned are then 'added' to the cube. The "cube" mechanism has most of its acceptance due to the possibility to add the full data of the large area detectors in addition to the variables present in the smallData file. The production of the hdf5 files can be run on the queue with MPI allowing the production to run fast.

Giving the users access to both data reduced in pixel space (by e.g. doing an azimuthal average) as well as in event space (by creating a "cube") allows complementary analyses which are sensitive to different systematic uncertainties.

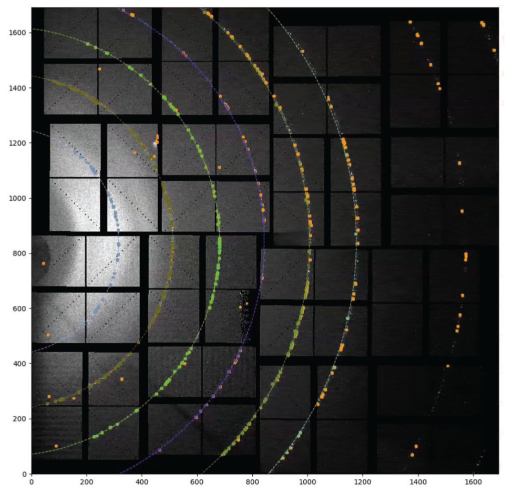
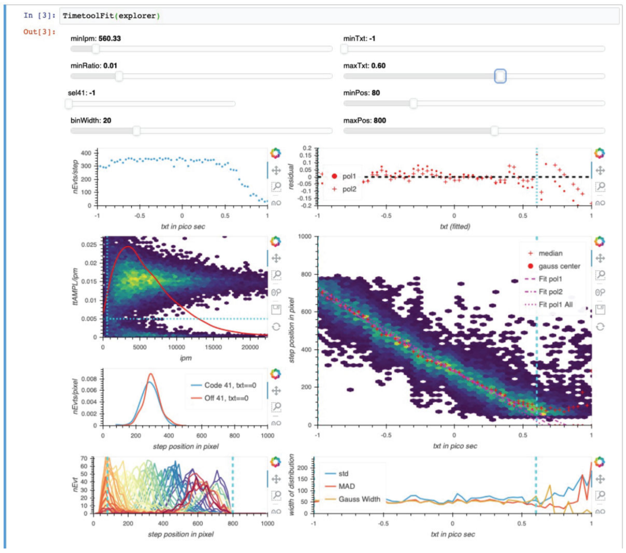


Figure 2: Beam center finding applied to the average image from 1.

JUPYTER NOTEBOOKS

LCLS offers a Jupyterhub with access to the data on the LCLS file system. Kernels with both the analysis framework in python2.7 as well as python3 kernels are available. For analysis of hdf5 files, either version will work. These notebooks allow user to further analyze data from their home institutes just as they did during the experiment. As notebooks allow to add text and images, they are an ideal tool to guide users in the analysis of their data. We have started to provide a few example notebooks with smalldata_tools and will expand on this for the upcoming run. Moreover, we will provide a method to save the analysis image to the Data Manager interface where it can be viewed by all collaborators.

Jupyter notebooks built on our standard smallData hdf5 file variables can be used to perform detailed analysis of calibrations runs. An example is the calibration data for the LCLS timetool [9]. The data is are taken as a step scan of the motor controlling the time delay between the optical laser and the X-ray. The pixel position of a step in the projection of an image of a camera relative to a baseline encodes the laser-X-ray in this event. We extract a calibration of this pixel position in pico seconds by fitting the data from 'good' shots in a 'reasonable' range. The notebook presents the experiment with several plots of the analysis chain and the definition of a 'good' shot and the 'reasonable' range can be adjusted with sliders. An example is shown in Fig. 3.



```
In [3]: TimetoolFit(explorer)
Out[3]:
```

minIpm: 560.33
minTxt: -1
minRatio: 0.01
maxTxt: 0.60
sel41: -1
minPos: 80
binWidth: 20
maxPos: 800

Figure 3: Panel with figures illustrating the timetool calibration extraction with interactive sliders for parameter adjustment.

Once the experimenter is satisfied with the result, the notebook with the selected parameters and fit result can be published to the Data Manager from the notebook.

The analysis system for LCLS-II will also provide a simple setup for users to create 'small' hdf5 files. Efficient data access methods are developed to both work in sequential 'smalldata' mode as well as allow access to events in a single 'bins' of a selected binning variable.

ACKNOWLEDGMENTS

I would like to thank the following people for their contributions in terms of ideas, code and/or infrastructure: Diling Zhu, Tim van Driel, Chris O'Grady, T.J. Lane, Murali Shankar, Henrik Lemke, Clemens Weninger, Jason Koglin and many more.

REFERENCES

[1] B. Aubert *et al.*, "The BaBar detector: Upgrades, operation and performance," *Nucl. Instrum. Methods Phys. Res. Sect. A.*, vol. 729, pp. 615–701, 2013. doi:10.1016/j.nima.2013.05.107

[2] J. Thayer *et al.*, Data system for the Linac coherent light source," *Adv. Struct. Chem. Imag.*, vol. 3, 2017. doi:10.1186/s40679-016-0037-7

[3] The HDF Group: *Hierarchical Data Format, ver 5.*, 1997. http://www.hdfgroup.org/HDF5/

[4] M. Chollet *et al.*, "The X-ray Pump-Probe instrument at the Linac Coherent Light Source," *J. Synchrotron Rad.*, vol. 22, pp. 503-507, 2015. doi:10.1107/S1600577515005135

[5] R. Alonso-Mori *et al.*, "The X-ray Correlation Spectroscopy instrument at the Linac Coherent Light Source," *J. Synchrotron Rad.*, vol. 22, pp. 508-513, 2015. doi:10.1107/S1600577515004397

[6] smalldata_tools, https://github.com/slac-lcls/smalldata_tools/

[7] SciPy.org, http://www.scipy.org

[8] Photon Finding, https://confluence.slac.stanford.edu/display/PSDM/Hit+and+Peak+Finding+Algorithms

[9] M. Bionta *et al.*, "Spectral encoding method for measuring the relative arrival time between x-ray/optical pulses,", *Rev. Sci. Instrum.*, vol. 85, p. 083116, 2014. doi:10.1063/1.4893657