Common mode correction algorithms

Content

Common mode correction

Most of pixel array detectors produce imaging data that can not directly be used in analysis and need in corrections. Most popular corrections are

- dark rate (pedestal) subtraction,
- bad pixel masking,
- common mode correction,
- gain correction,
- etc.

In this note we discuss common mode correction algorithms. Common mode is a hardware effect of collective pixel intensity variation due to synchronous variation of potentials on sensor chip or ADC at readout process. This effect can be corrected at low sensor illumination, where the number of pixels with energy deposition from photons is small. The spectrum of pixel intensities without photons should be grouped in the peak with small offset of the average from zero (if the dark rate correction is already applied) due to the common mode effect. This offset can be evaluated and subtracted from all pixel intensities in the group of synchronously fluctuating pixels. The number and structure of commonly behaving pixel groups depend on detector hardware.

Implementation

Common mode correction is applied in the Detector interface method `calib` described in AreaDetector:

```
arr = det.calib(evt, cmpars=None)
```

and is controlled by the list/tuple of parameters `cmpars`=(<algorithm>, <mode>, <algorithm-specific parameters>,...).

Common mode correction can be turned off using `cmpars=(0,0)`.

If this list of parameters is not specified `cmpars=None`, it is searched under the experimental calib directory or substituted by default.

Each algorithm may load file with parameters from calibration directory, which by default accounts for experimnet, calibration version, data source, calibration type and run range, for example

```
/reg/d/psdm/<INS>/<experiment>/calib/<calib-version>/<data-source>/<calibration-type>/<run-range>.data

For example:
/reg/d/psdm/XPP/xppi0614/calib/Epix100a::CalibV1/NoDetector.0:Epix100a.0/pedestals/0-end.data
/reg/d/psdm/XPP/xppi0614/calib/Epix100a::CalibV1/NoDetector.0:Epix100a.0/common_mode/0-end.data
```

Content of this file depends on detector, calibration type, and algorithm, as shown below.

Algorithms

We use algorithms earlier developed for CSPAD and other detectors and currently available through the Detector package.

Selection of algorithm of particular type is controlled by the list of `cmpars` or similar list of parameters in file for `common_mode` calibration type.

#0 - common mode correction is turned off

The 1-st parameter in the list of common mode parameters represents algorithm number. If it is set to 0 common mode correction is not applied.

#1 - common mode peak finding algorithm

Valid for: CsPad-style detectors (CsPad with 32 tiles and smaller CsPad2x2 with 2 tiles).

This algorithm is similar to one developed by Andy Salnikov and implemented in https://github.com/lcls-psana/pdscalibdata/blob/master/src/CsPadCommonModeSubV1.cpp:

1. for each cspad tile an intensity histogram is filled with pedestal-subtracted ADU values (1 bin for each possible ADU value). pixels with bad status are ignored
2. software finds where histogram crosses threshold (parameter 3)
3. measure mean/rms of region that is above the threshold
4. if mean/rms is in allowed range (parameters 1 and 2) then common mode value is set to the mean.
5. if the correction is not in the allowed range, and par[4]>0.1 (see below) then the common mode value is computed from unbonded pixels (see algorithm #5)
6. subtract the common mode value from all pixels in the cspad tile

Parameters for this algorithm resides in the calibration directory in the directory named `common_mode` or can be set with the python Detector interface.

Default parameters for CSPAD and CSPAD2x2

```
1 25 25 100 1
```

- par[0] - algorithm #
- par[1] - maximal deviation of the peak mean from 0
- par[2] - maximal allowed value of the peak RMS
- par[3] -threshold on number of pixels in the ADU bin in the peak finding algorithm
- par[4] - if set to 1 then use unbonded pixel common-mode values (algorithm 5) for those panels where the standard correction failed. do not do this if set to 0

parameters for other detectors

- par[4] - number of segments for common mode evaluation
- par[5] - segment size (number of pixels for common mode evaluation)
- par[6] - stride (step for jump to the next pixel)

  For example:

  ```
  1 50 50 100 8192 128 1
  ```

#2 - MEAN algorithm

Not valid for: CsPad and CsPad2x2

It was developed by Philip Hart for test purposes;

1. for each group of pixels intensity histogram is filled for natural ADU bins, pixels with bad status are ignored,
2. a simple mean below threshold is considered as a common mode correction,
3. if correction is in the allowed range, it is applied to all pixels from the group.

Control parameters for this algorithm resides in the file for `common_mode` calibration type;

- par[0] - algorithm #
- par[1] - maximal threshold on intensity to evaluate mean for low intensities
- par[2] - maximal allowed common mode correction
- par[3] - length of consecutive pixel array for common mode evaluation
  For example, for pnCCD one can evaluate common-mode for one came chip (128 channels).

Control file: /reg/d/psdm/amo/<exp-name>/calib/PNCCD::CalibV1/Camp.0:pnCCD.0/calib/PNCCD::CalibV1/Camp.0:pnCCD.0/common_mode/0-end.data

```
2 1000 1000 128
```

- The algorithm loops over the data and evaluates consecutive arrays of specified length (which might represent for example a row of pixels in a readout chip) and finds the mean value for values below a threshold, ignoring masked pixels. It corrects all the data if the calculated common mode is less than the maximal allowed correction. The median algorithm is more stable in many cases and is recommended.

#3 - MEDIAN algorithm

Not valid for: CsPad and CsPad2x2

It was developed by Philip Hart as a replacement for the mean (#2) algorithm;

1. for each group of pixels intensity histogram is filled for natural ADU bins, pixels with bad status are ignored,
2. a half of statistics counted from bin with low intensities below threshold is considered as a common mode correction,
3. if correction is in the allowed range, it is applied to all pixels from the group.

Parameters are the same as in #2. The algorithm is as above, except that it calculates the median, or the average of the two median points if there are an even number passing the selection criteria.

```
3 100 100 128
```

#4 - MEDIAN algorithm - detector dependent

Not valid for: CsPad and CsPad2x2

It is pretty similar to one developed by Matthew Weaver that is implemented in `ami/event/FrameCalib`

In `ImgAlgos::NDArrCalib` it is implemented for Epix100a and Fccd960. The algorithm is detector-dependent, executing different code depending on whether an Epix or Fccd "Source" is given to the NDArrCalib module.

The difference from algorithm #3 is quite minor; it starts to fill the intensity histogram in a quite narrow range relative to zero, but if the half of statistics is not found in this range it is extended by 1/4 of pixels (see parameter #3 below for initial guess for range of histogram). This iterations are repeated until the half of statistics is in the range, or the number of bins exceeds 10000.

Typical `common_mode` control parameters:

```
4 6 30 10    0 0 0 0    0 0 0 0    0 0 0 0    0 0 0 0
```

where

1. par[0] (4) - algorithm number; 4 stands for this median algorithm,
2. par[1] (1) - type of regions for median algorithm; 3 regions (banks/rows/columns) for Epix100a, and 2 regions (banks/rows) for Fccd960:
    a. EPIX100A has an option to turn on up to 3 regions for common mode correction, controlled by the bitword parameter #2: bit#1 - common mode for 352x96-pixel 16 banks, bit#2 - common mode for 96-pixel rows in 16 banks, bit#3 - common mode for 352-pixel columns in 16 banks
    b. FCCD960 has 2 regions, also selected by parameter #2: bit#1 - common mode correction for 1x160-pixel rows with stride 2, bit#2 - common mode correction for 480x10-pixel 96*2 supercolumns
3. par[2] (30) - initial guess for range of histogram (relative to 0) in ADU (see above description of intensity histogram). This number can grow in subsequent iterations if half of the statistics is not found in this range. It should probably start at around 3 sigma of the noise value.
4. par[3] (10) - maximal allowed absolute value of the common mode correction in ADU. If = 0 - no-limit is applied directly, indirect limit = 10000ADU (from limit on intensity histogram described above).

Other parameters are not used.

#5 - Unbond pixels common mode correction

Valid for CsPad-style detectors **only!**

In latest version of CSPAD detectors a group of pixels is not bonded to relevant electronic channels, so they see no signal. For each tile of the detector an averaged signal from these unbonded pixel-channels is used as a common mode value. It is subtracted from all other pixels.

Typical `common_mode` control parameters:

```
5 50
```

Parameters for this algorithm:

- par[0] - algorithm #
- par[1] - maximal allowed common-mode correction value (i.e. don't apply common-mode if computed value exceeds this number). In other common-mode algorithms this parameter is used to make sure that the common-mode has not been computed using pixels with signal in them. For this algorithm (that uses unbonded pixels) this parameter is less necessary, since there should never be signal in these pixels, unless there are electronic effects. One example of such electronic effects: experiments in the MEC hutch often see an electromagnetic-pulse from their high-powered lasers that causes unusual things to happen in the CSPAD readout. But this is rare in non-MEC experiments.

#6 - Epix neighbor exclusion

This algorithm ignores pixels above histoRange and, unlike algorithm #4, their neighbors. The statistics used to compute the common-mode values will be poorer because there will be fewer pixels used, but possible energy leakage from the pixels containing photons will not be included. The parameters are:

1. img - image on which the common mode is applied
2. rms - array of noise values for each pixel with same shape as img
3. maxCorr - (default = 30) maximum correction applied. If common mode correction is larger than this value, no correction will be applied
4. histoRange - (default = 30) all pixels above this parameter are masked
5. colrow - (default = 3) decides what is corrected. If 1, only the columns are corrected. If 2, only the rows are corrected. And if 3, both are corrected
6. minFrac - (default = 0.25) the minimum fraction of pixels required to be left in a row or column after applying the mask and rejecting high pixels and their neighbors
7. normAll - (default = False) if true, will subtract the mean from the full image with the masked applied

Below is an example block of code showing how to call this algorithm and pass it arguments.

```
for nevent, evt in enumerate(ds.events()):
        if nevent == 10:
                break

        # cmpars = [6] because this is the 6th common mode algorithm and it must be in a list format
        # Add arguments on as seen fit
    nda = det.calib(evt, cmpars=[6], rms=det.rms(evt), maxCorr=25)
```

#7 - MEDIAN algorithm for jungfrau and epix10ka

This algorithm evaluates common mode offset for specified by par[1] groups of pixels using method numpy.median.

- par[0] - algorithm # 7
- par[1] - mode 0/ +1/ +2/ +4 - turned off / in rows / in columns / in banks. **Each correction consumes extra time.**
- par[2] - maximal allowed common mode correction in det.raw ADU **to protect from over-correction at high illumination - needs to be adjusted by users per experiment**
- par[3] - minimal number of good pixels in mask and gain mode to evaluate median offset (optional, default =10)

rows and columns are one dimensional set of pixels limited by the shape of bank.

- jungfrau panel shape=(512,1024) consists of 2x4=8 ASICs (256,256), each ASIC contains 4 banks, bank shape=(256,64)
- epix10ka panel shape=(352,384) consists of 2x2=4 ASICs (176,192), each ASIC contains 4 banks, bank shape=(176,48)

- common mode correction is applied to pixels in **high gain mode only for jungfrau** and **H or M modes (fixed or auto-switching) only for epix10ka**.
- common mode is evaluated for good pixels only (marked by 1 in mask).
- mask is combined of 2 sources (if available)
  - det.calib(..., mbits=1,...) which is used in det.mask_comb(..., mbits=1,...). Parameter mbits controls mask of many sources, including pixel_status, borders, unbonded pixels (cspad only), and user-defined pixel_mask.
  - det.calib(..., mask=None,...) user specified mask shaped as det.raw

Default parameters:

```
7 2 10 10 0 0 0 0 0 0 0 0 0 0 0 0
```

Code:

```
cmpars=None # for default
cmpars=(7,0,10,10) # turn off common mode correction
cmpars=(7,1,10,10) # correction in rows
cmpars=(7,2,10,10) # correction in columns - works the best for epix10ka
cmpars=(7,4,10,10) # correction in banks
cmpars=(7,3,10,10) # correction in rows and columns - works the best for jungfrau
cmpars=(7,7,10,10) # correction in banks, rows and columns (in this order - banks first, then rows, then
columns)

Example:
nda = det.calib(evt, cmpars=(7,3,10,10), mbits=1, mask=None, **kwargs)
```

#8 - MEDIAN algorithm for pnCCD

Accounts for pixel mask and apply median correction for different group of pixels, selected by parameter 1:

- par[0] - algorithm # 8
- par[1] - mode of median algorithm for pixel groups: 0 - turned off | +1 - rows in banks (128 pix) | +2 - for rows (512 pix) | +4 columns (512 pix) | +8 banks (512x128 pix)
- par[2] - maximal allowed common mode correction

For each group of pixels common mode is evaluated using pixels defined by mask (1), then applied to all pixels in the group.

Code example:

```
# Call algorithm directly:
from Detector.UtilsPNCCD import common_mode_pnccd
common_mode_pnccd(nda, mask, cmp=(8,5,500))   # median in rows 128 and columns 512

# call through the Detector interface:
pnccd.common_mode_apply(evt, nda, (8,5,500), mask=mask) # new py
# OR
nda = pnccdd.calib(evt, (8,5,500), mask=mask) # generic calib method with mask parameter
```

Test of the common mode correction for pnCCD

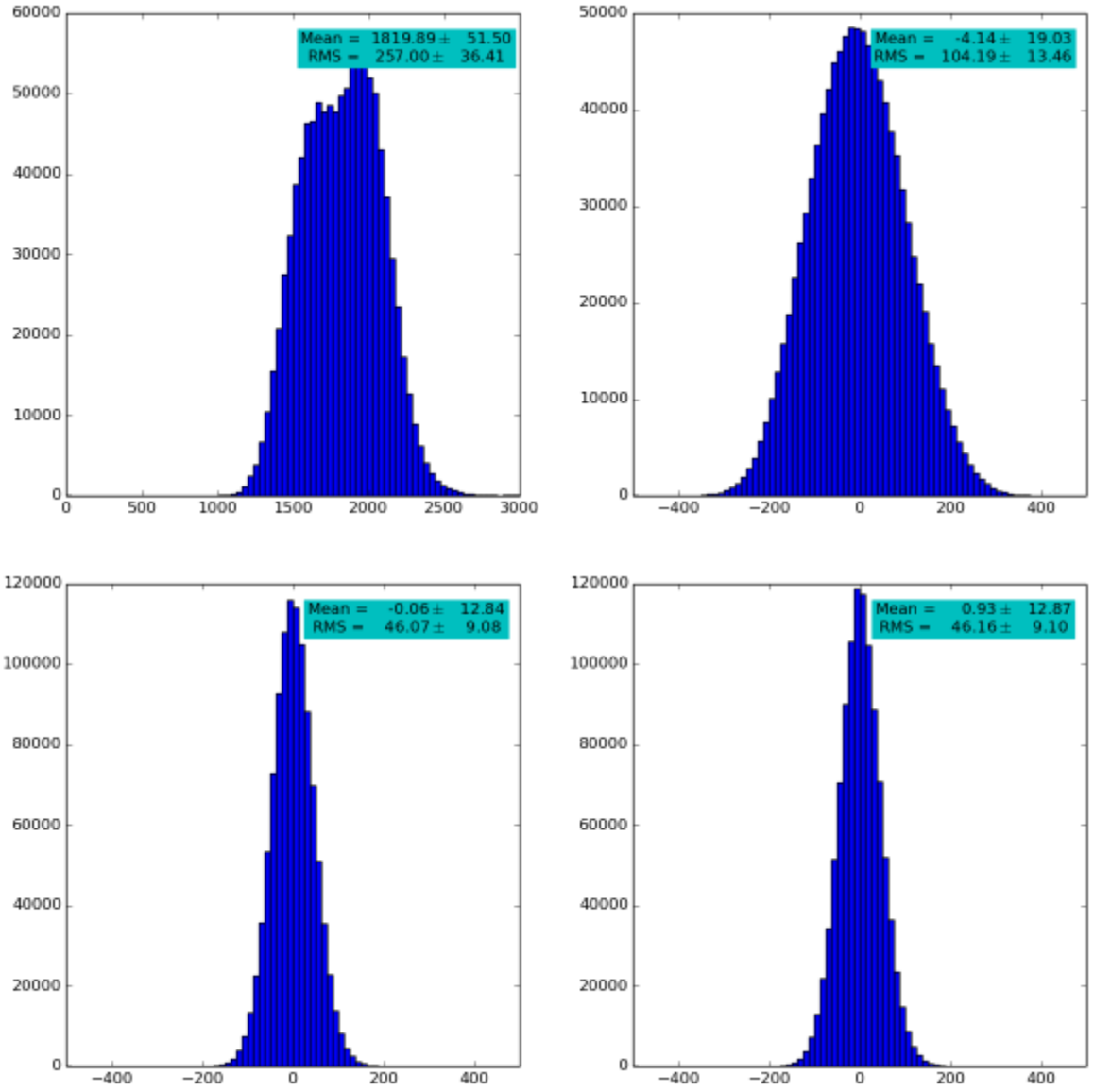
To test implementation of algorithms in `ImgAlgos::NDArrCalib` we use the same data sets as in 2014-03-25-Ankush-CommonModeNoise.pdf

Use data from experiment amob5114

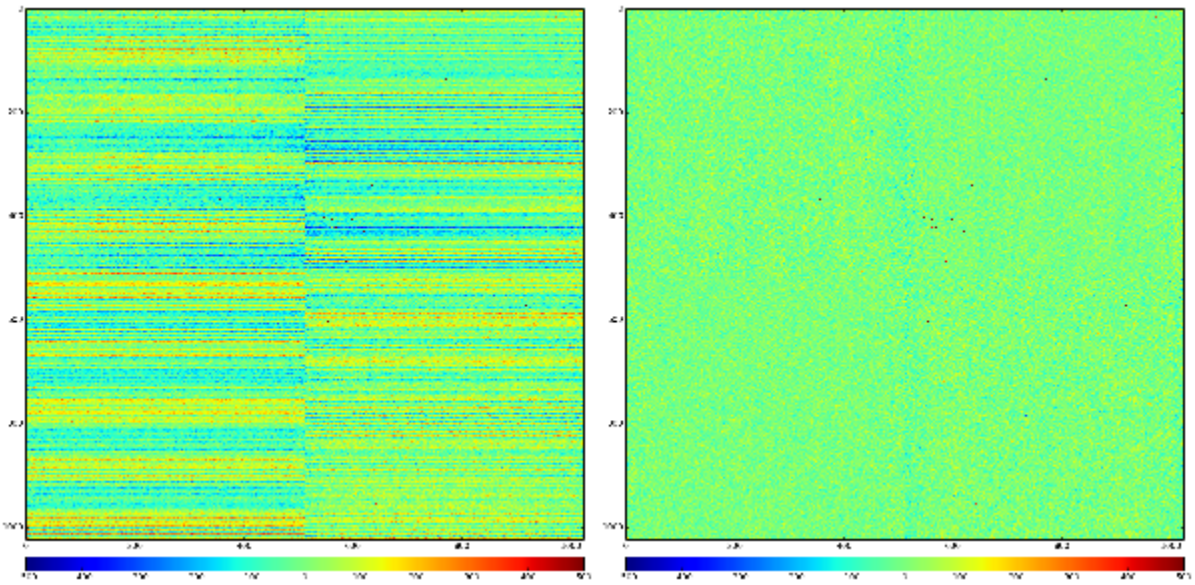
High gain pnCCD run 121

```
2(or 3) 1000 1000 128
```

Spectra for 1) raw data, 2) subtracted pedestals, 3) subtracted common mode correction algorithm #2 and 4) algorithm #3:



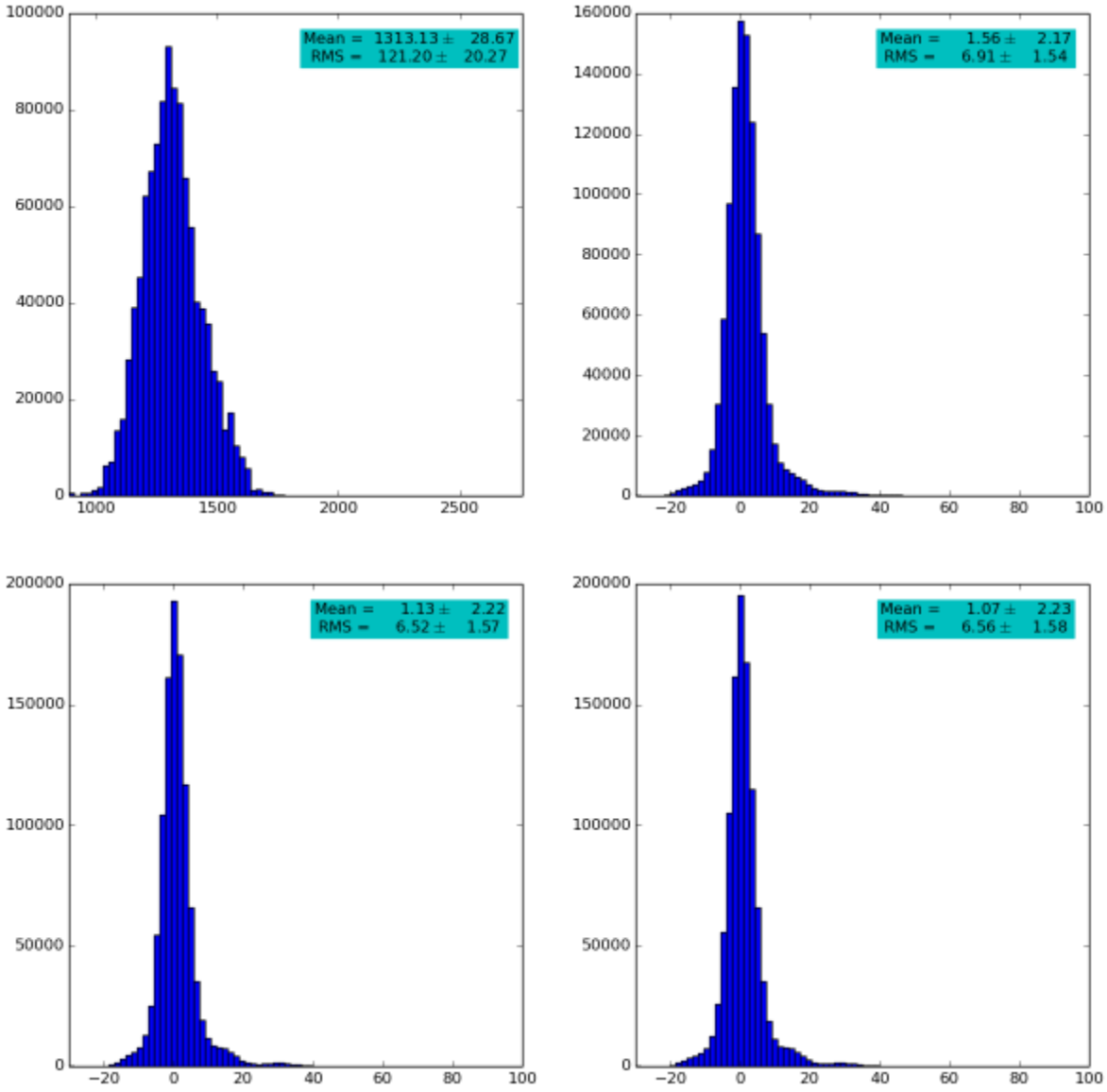
Images 1) for subtracted pedestals and 2) common mode correction algorithm #2:



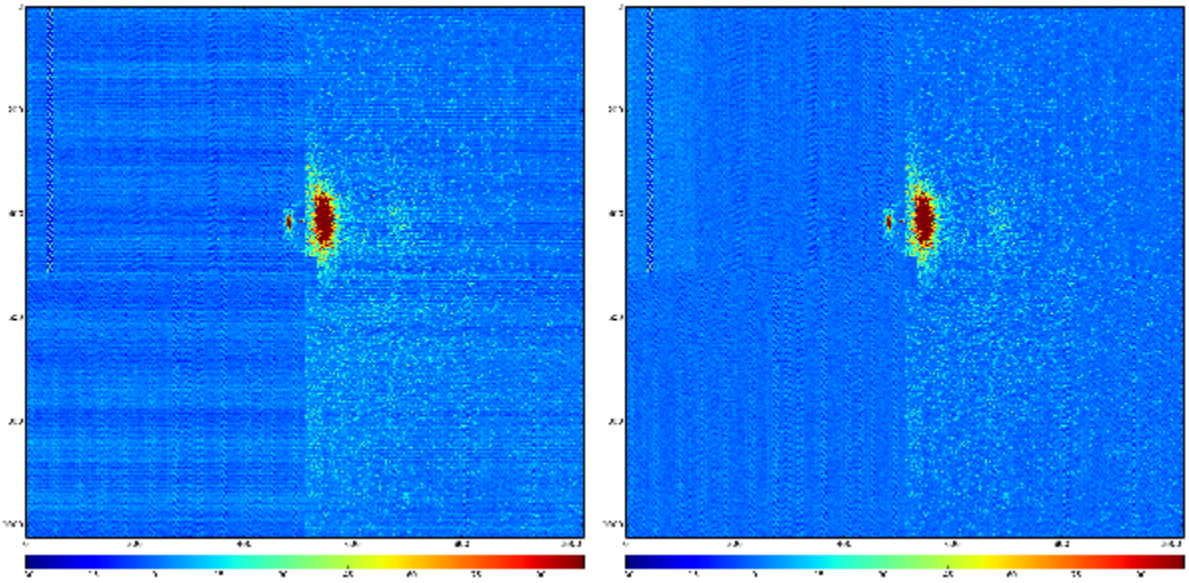
High gain pnCCD run 329

```
2(or 3) 1000 1000 128
```

Spectra for 1) raw data, 2) subtracted pedestals, 3) subtracted common mode correction algorithm #2 and 4) algorithm #3:



Images 1) for subtracted pedestals and 2) common mode correction algorithm #2:



Summary for pnCCD

Common mode correction for pnCCD

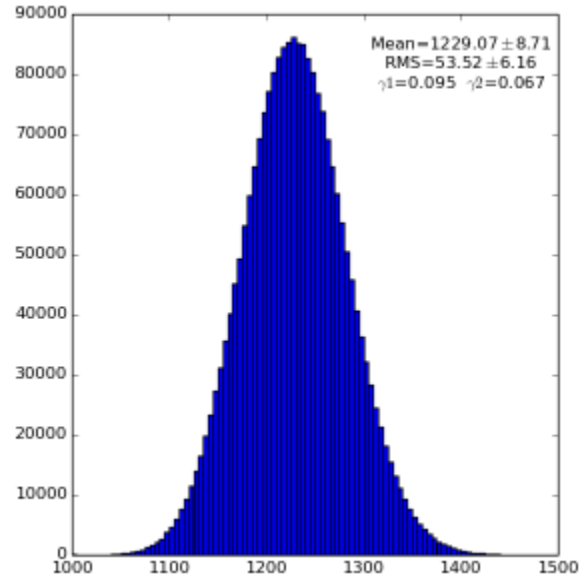
- gives significant effect in low gain mode and is negligible in high gain mode
- algorithm #2 gives the best results, #3 a little bit worse, #1 - does not work for pnCCD

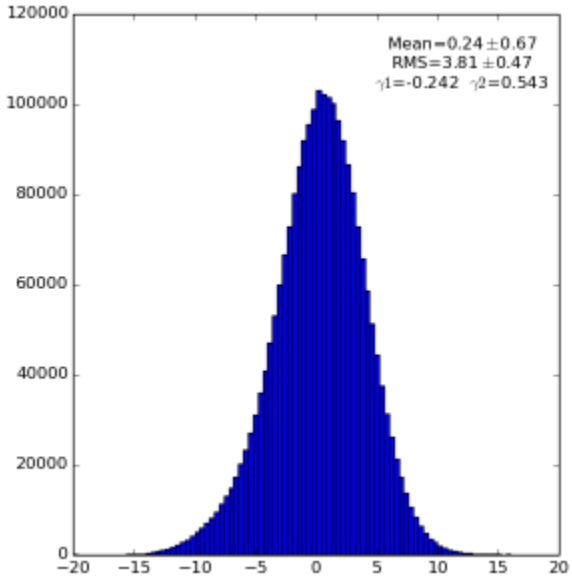
Test of common mode correction for CSPAD

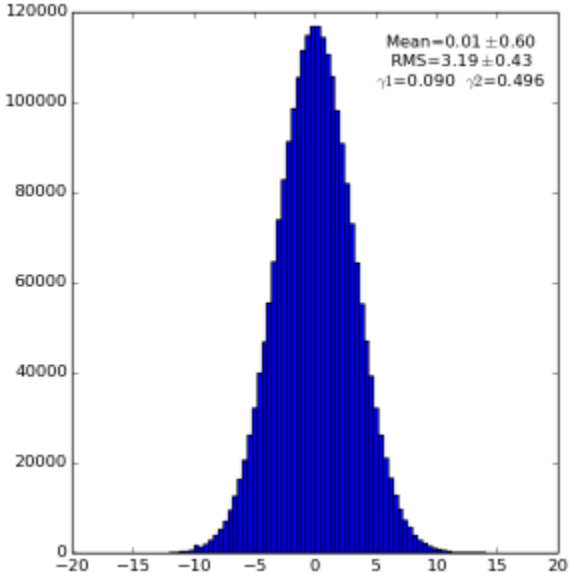
Here we use cxi83714-r0136: lysozyme crystallography data with bright bragg spots on top of a varying water background. We use these common-mode parameters (turning off the fall-back unbonded correction):

```
1 10 10 100
```

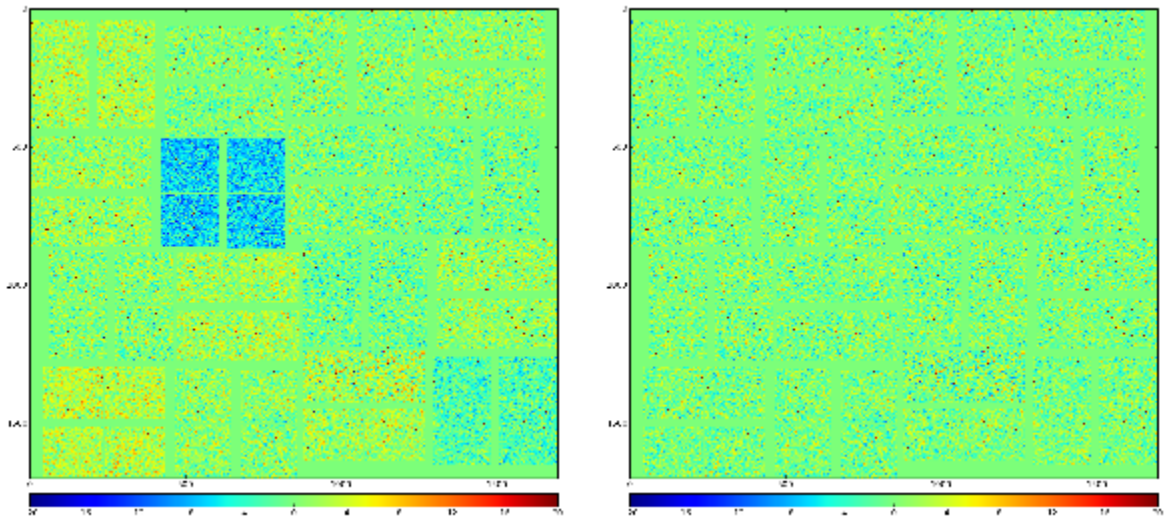
Spectra for 1) raw data, 2) subtracted pedestals, 3) subtracted common mode correction algorithm #1:







Images 1) for subtracted pedestals and 2) common mode correction algorithm #1:



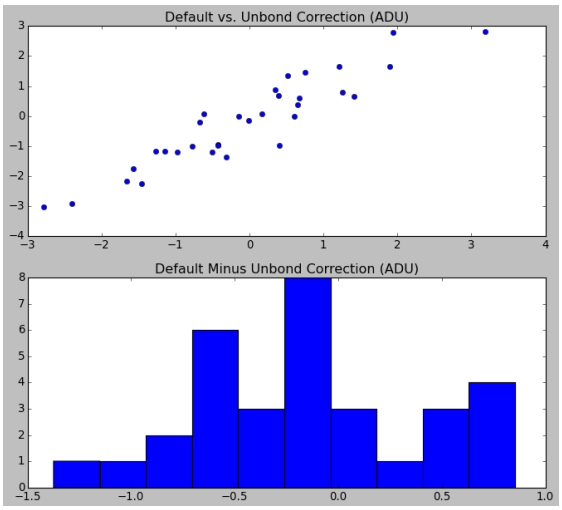
Comparison of Unbonded vs. Default Common Mode

Dark Case

This is for a **dark** run, and was generated using this script:

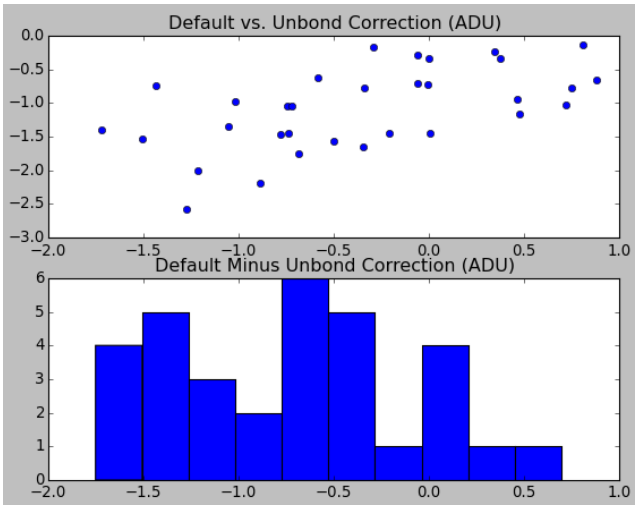
```
import psana
from matplotlib import pyplot as plt
import numpy as np
dataset_name = "exp=cxil5316:run=1"
ds = psana.DataSource(dataset_name)
psana_det = psana.Detector("DscCsPad")
evt = ds.events().next()
pedestal = psana_det.pedestals(evt)
data = psana_det.raw_data(evt) - pedestal
data_cm = psana_det.raw_data(evt) - pedestal
unbond_cm = psana_det.common_mode_correction(evt, data_cm, [5, 50])
default_cm = psana_det.common_mode_correction(evt, data_cm, [1, 50, 10, 100])
unbond=unbond_cm[:,0,0]
default=default_cm[:,0,0]
plt.subplot(2,1,1)
plt.title('Default vs. Unbond Correction (ADU)')
plt.plot(unbond,default,'o')
plt.subplot(2,1,2)
plt.title('Default-Unbond Correction (ADU)')
plt.hist(default-unbond)
plt.show()
```

This yields the following plots comparing the size of the two corrections:



Brighter Case

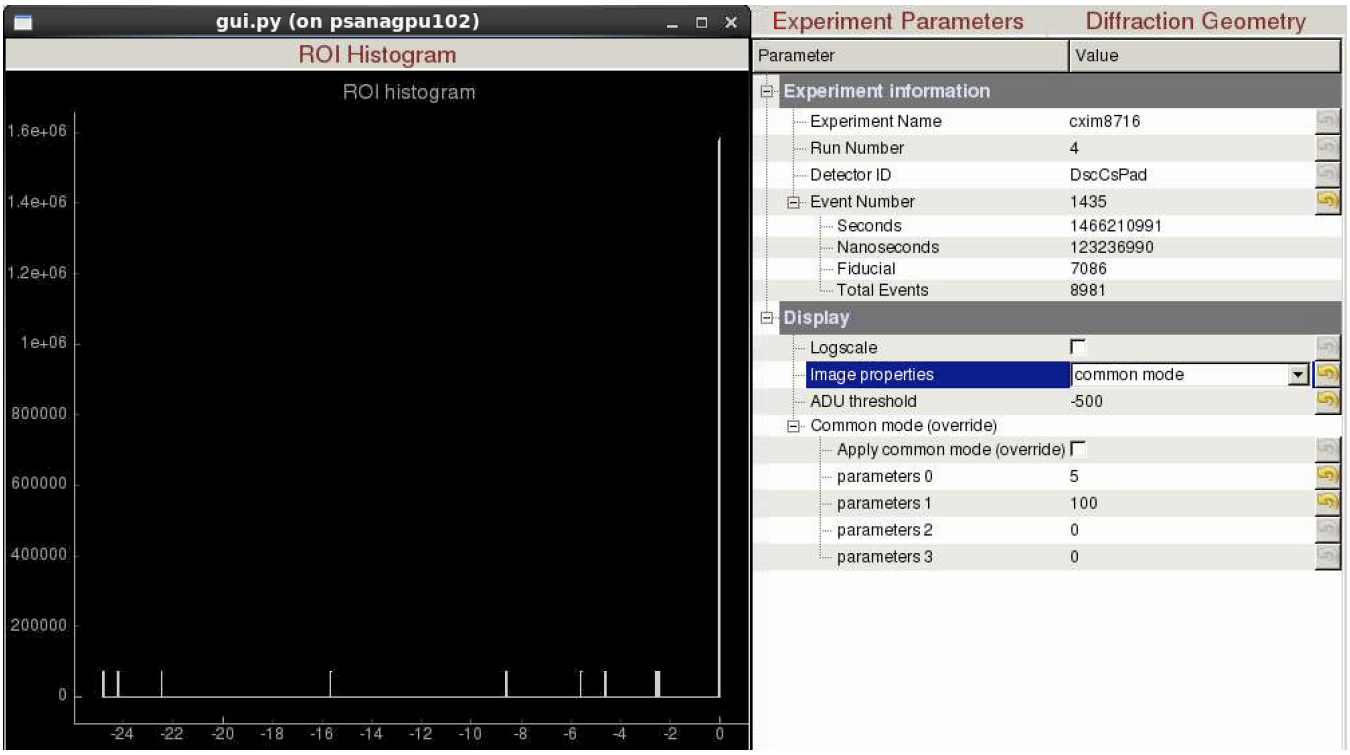
This is the same plot for an event with many photons. This is for exp=cxid9114:run=96, and the event has unix-timestamp 6025277111415285948. There is one entry in the plot for the common-mode value for each tile of the CSPAD.



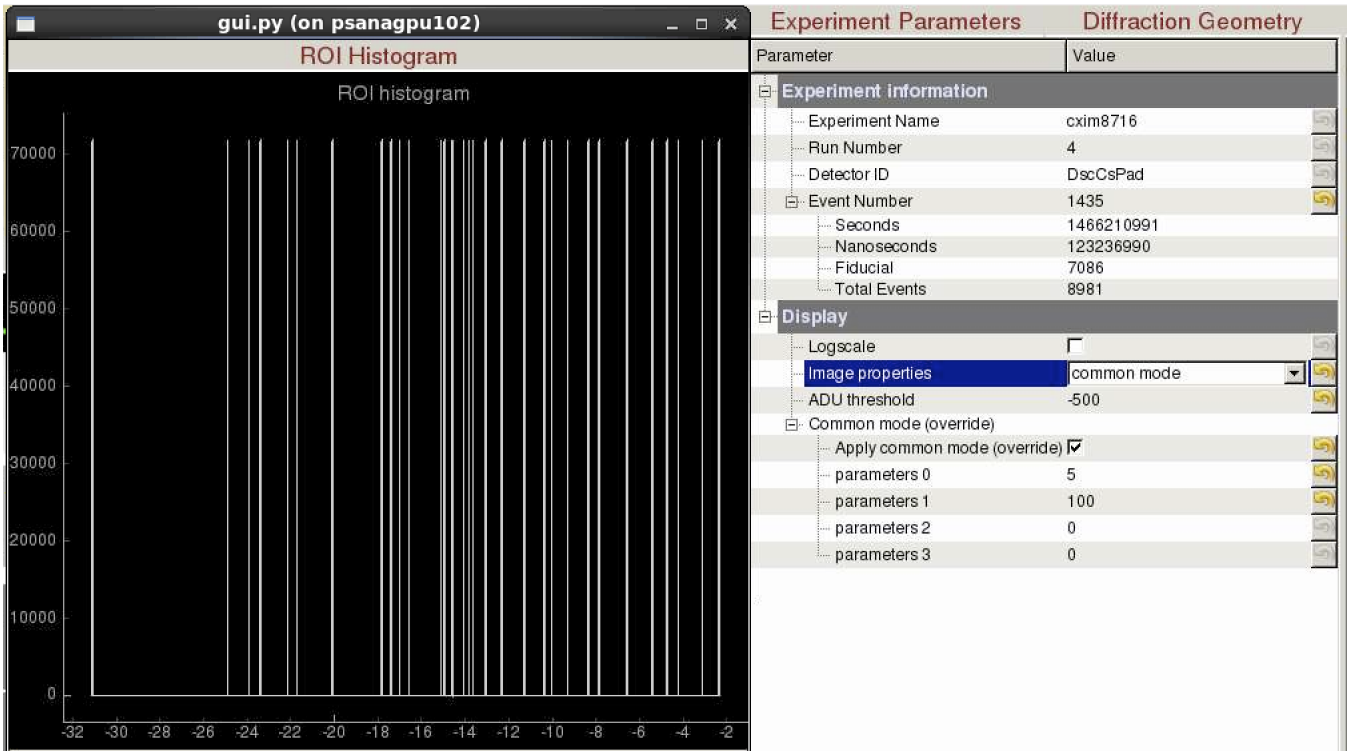
The common-mode methods still correlate, but the default method perhaps over-subtracts because of leakage into the zero-photon peak from neighbor-pixels of real photons.

Crystallography Case

Here are the common mode corrections applied to a crystal diffraction pattern with a strong water ring using algorithms 1 vs 5. Algorithm 5 seems to perform better in this case.



Common mode histogram using algorithm 1



Common mode histogram using algorithm 5

Thoughts from Phil on complexities involved with CsPad Common Mode:

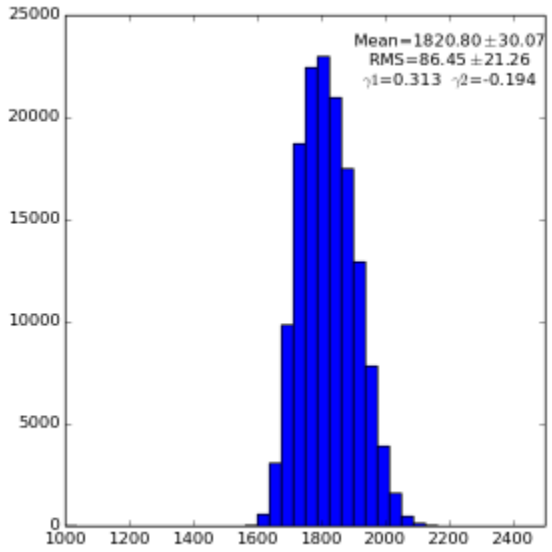
- statistics of unbonded pixels is a problem
- poisson statistics on bragg spots is bigger than the common mode
- don't correct common-mode for low-gain pixels
- the larger shifts seen in bright events are "cross-talk" caused by the "ramp" where pixels nearer the mean-pedestal have a lower gain
- should do per-asic common-mode

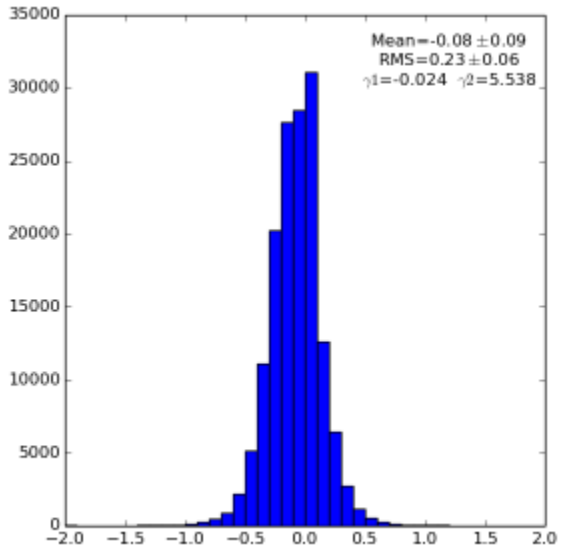
Test of common mode correction for CSPAD2x2

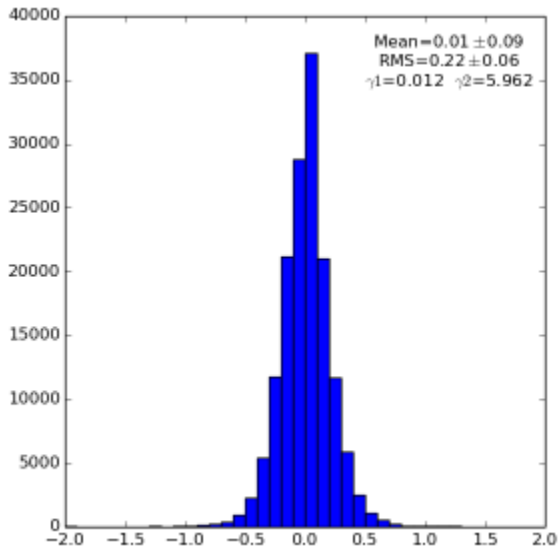
Use meca1113-r0045 with

```
1 50 10 (100) - last parameter is set by default
```

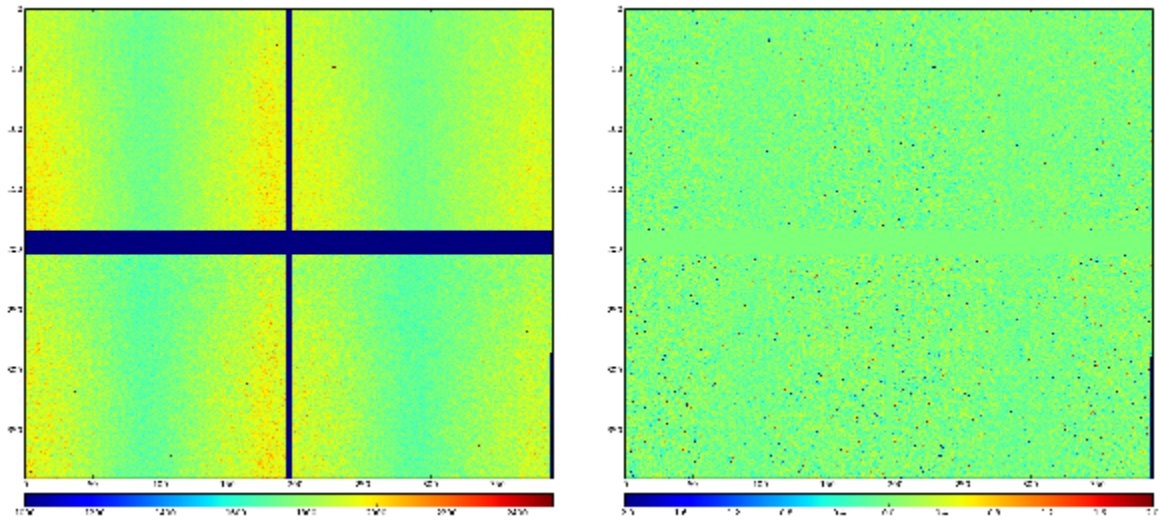
Spectra for 1) raw data, 2) subtracted pedestals, 3) subtracted common mode correction algorithm #1:







Images 1) raw data and 2) subtracted pedestals with common mode correction algorithm #1:



Summary for CSPAD2x2

Common mode correction for CSPAD2x2 in this example shows minor improvement.

References

- AreaDetector - Detector Interface is described in the head of this file.
- 2014-03-25-Ankush-CommonModeNoise.pdf - stand-alone test of common mode correction for pnCCD
- psana - Module Catalog - Module ImgAlgos::NDArrCalib
- Epix100a
- Fccd960-Detector
- numpy.median