PACS photometer map-making with MADmap

PACS-401

for HIPE 9 user release Version

Babar Ali (NHSC)

What is MADmap?

- Mapping code written by the Berkeley CMB group to remove 1/f noise from bolometers.
  http://newscenter.lbl.gov/feature-stories/2010/02/03/madmap/

- MADmap was ported to Java for use in HIPE.
- MADmap offers the so-called optimal map-making to convert time ordered readouts to a final map.
  - Uses maximum likelihood (given a noise/probability model) to determine the optimal sky value.

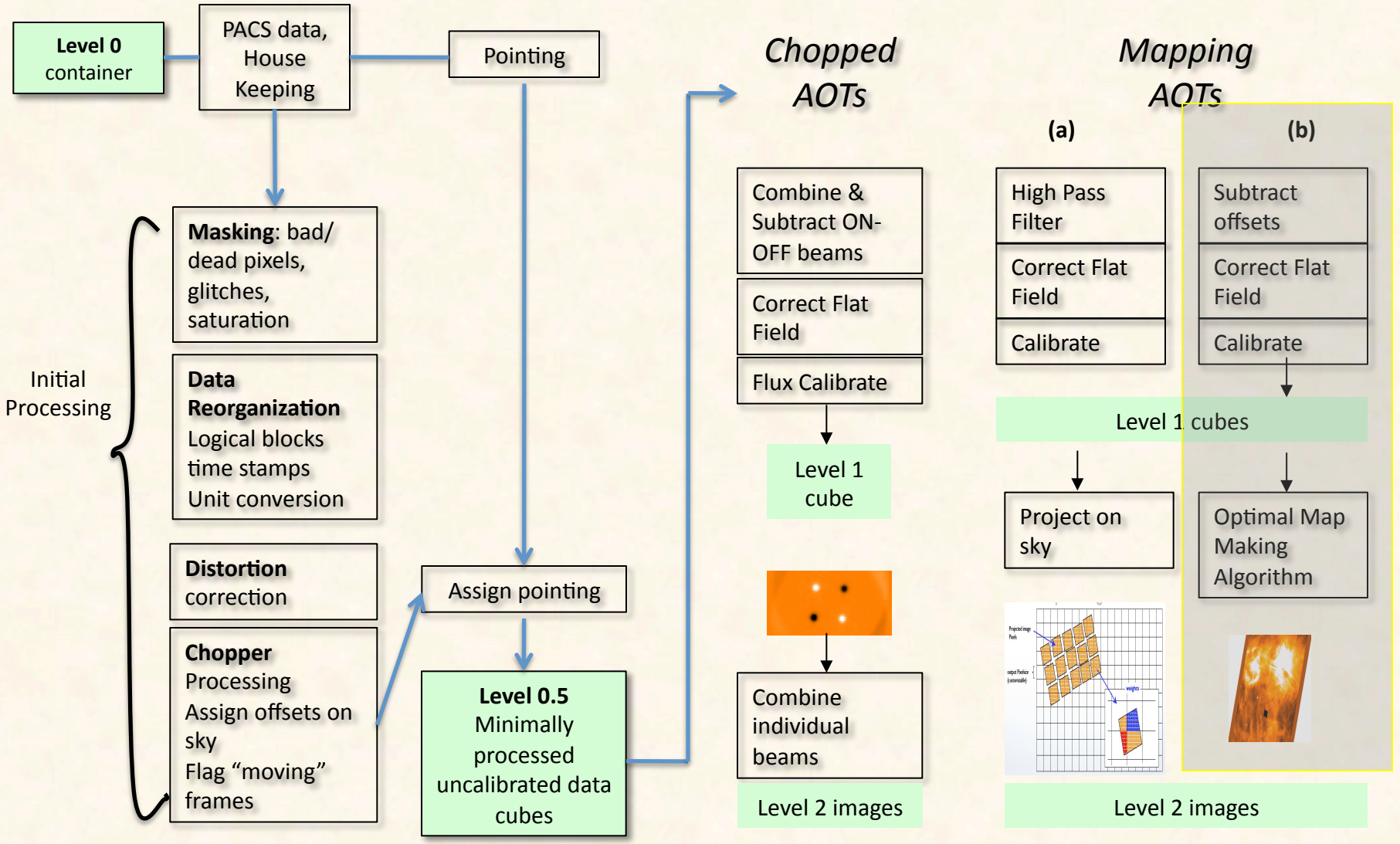
When should you use MADmap?

- When spatially extended emission is present in the data. E.g. Galactic star-formation fields.
  - MADmap preserves spatially extended emission.
- When the source itself is extended. E.g. large galaxies.
- When extended structure is present around a compact source. E.g. extended halos or nebulosity.

Caveats & Warnings

- MADmap assumes that bolometer time-lines are calibrated.
  - Primarily that pixel-to-pixel instrument variations are already removed.
- MADmap assumes that 1/f noise is uncorrelated amongst pixels.
- All correlated noise (signal drift) must be removed prior to running MADmap.

Pipeline section covered here



Documentation Reference

- PACS data reduction guide, chapter 9
- **PACS-101**: Introduction to PACS tutorials
- **PACS-103**: Accessing & Storing PACS data
- **PACS-104**: Using iPipe scripts
- **PACS-201**: Level 0 to 1 processing of PACS photometer data
- **PACS-401**: MADmap map-making.

Outline of processing

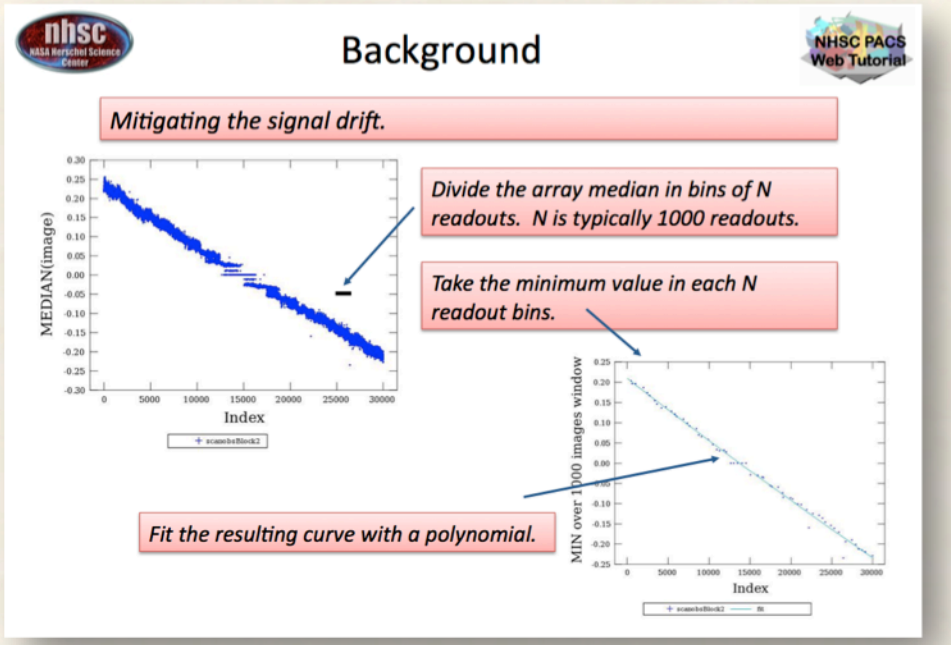
I. Pre-processing
   - Calibrate time-lines
   - Remove correlated signal drift

II. MADmap
   - Create “naïve”/simple map.
   - Create optimal map.

III. Post-processing (optional)
   - Remove point-source artifact around bright point sources.

Use the tutorials

- This talk will concentrate on the using MADmap.
- The tutorials provide step-by-step walkthroughs.
- Expected results are highlighted.
- Recommended parameters and usage is highlighted.



PACS 401

I: Pre-processing

Preparing the time-lines for MADmap

Calibrating Time-lines

- **Start with Level 1 products.**
  - The ipipe scripts start with level 0
  - However, the processing is similar between all pipeline branches of the PACS photometer
- The following additional steps are needed to finish calibration for MADmap
  1. **Assign (α,δ) values to each pixel**
  2. **Remove pixel-to-pixel electronic offsets**

MADmap preprocessing

Executing the loop will automatically execute these steps for both OBSIDs.

Cleanup unstable frames at the start of observation.

```
308     #
309     frames = photMadMapIgnoreFirst(frames, ignoreFirst)
310     #
311     frames = photAssignRaDec(frames, calTree=calTree)
312     #
313     frames = photOffsetCorr(frames)
314     #
```

Assign pointing to each and every pixel and readout

Apply pixel-to-pixel electronic offset correction.

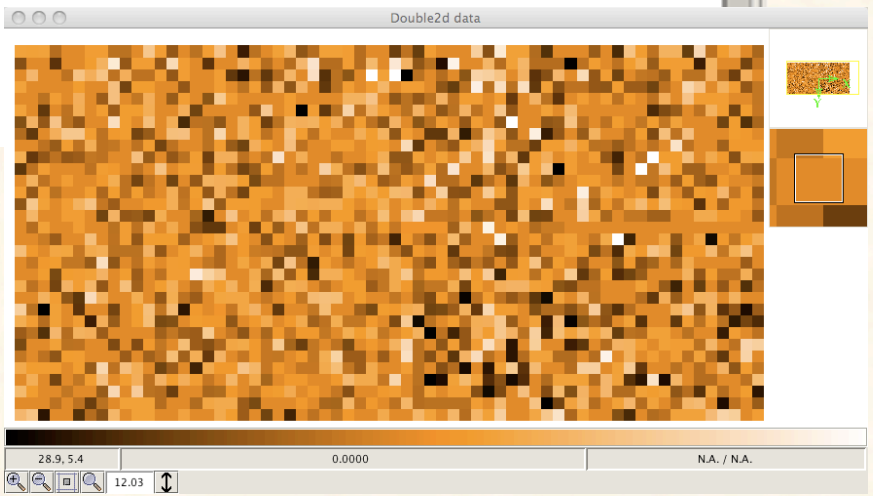
Taken from the tutorial

Check # 5: Offsets are removed

Type this command

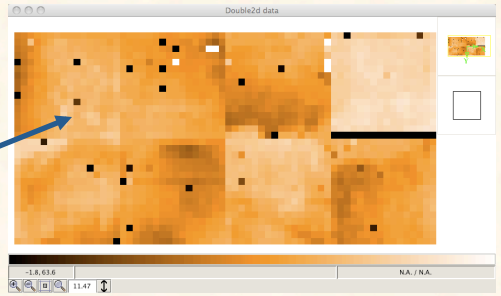
Console

```
HIPE> Display( frames.signal[:,:,100] )
```



Expected Output:
With proper offset removal, the image shows a relatively constant signal. Note: extremely bright sources may also be observed on a single image.

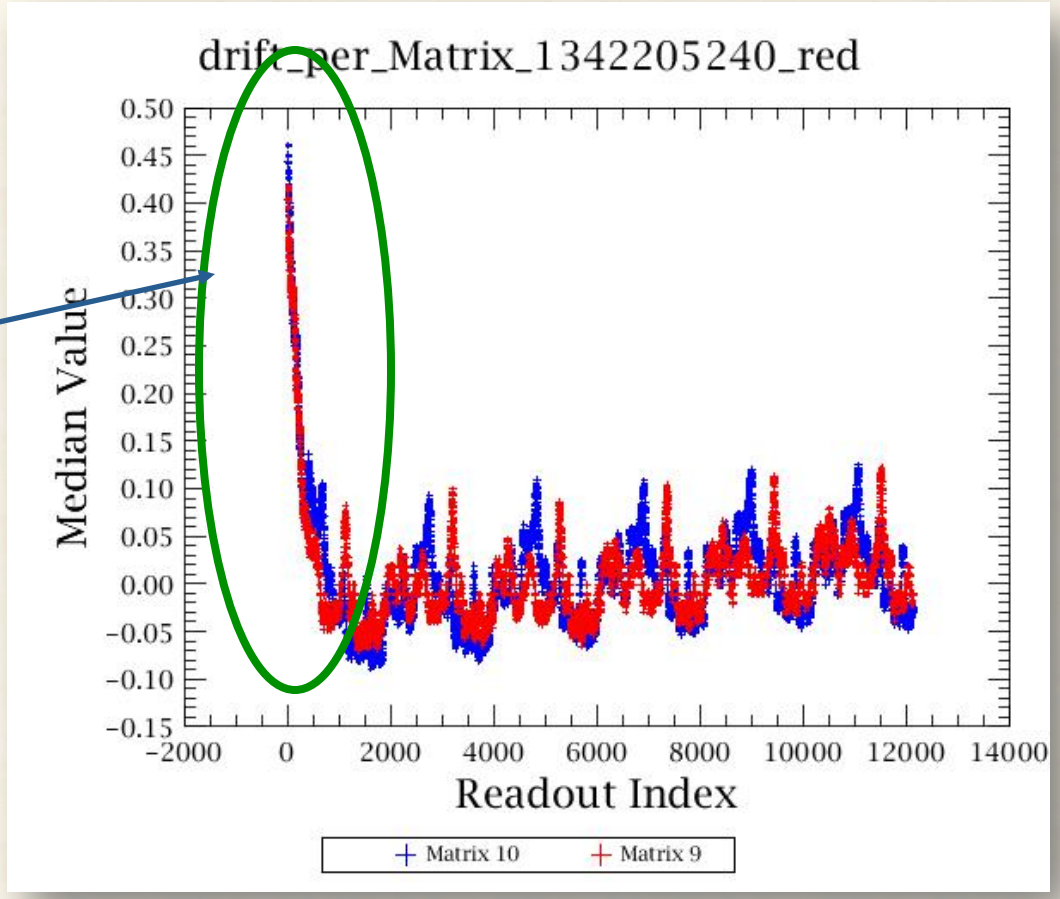
An example of improper or no pixel-to-pixel offset correction.



Masking unstable frames

Plot

MEDIAN(array) *vs* Readout #



Unstable frames:

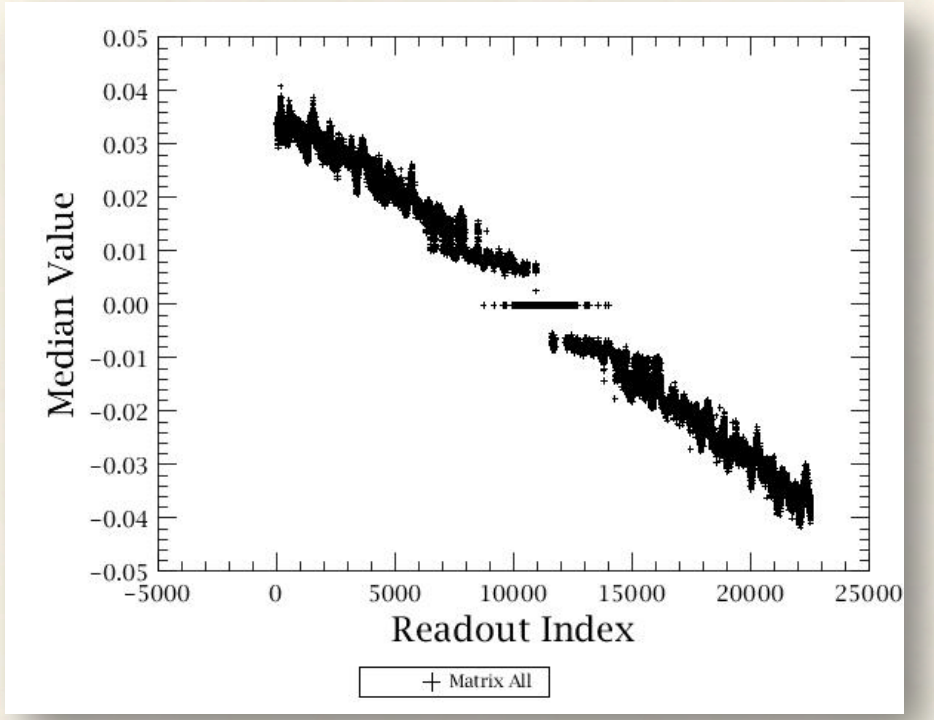
- Usually at the beginning
- Likely due to calibration pre-amble.

To fix:

```
#
frames = photMadMapIgnoreFirst(frames, ignoreFirst)
#
```

Correlated Signal Drift

- **Examine the median value of the entire array.**
  - Strong correlation with time (or readout index)
  - No single model fits the drift (so far.
    - Behavior is inhomogeneous.

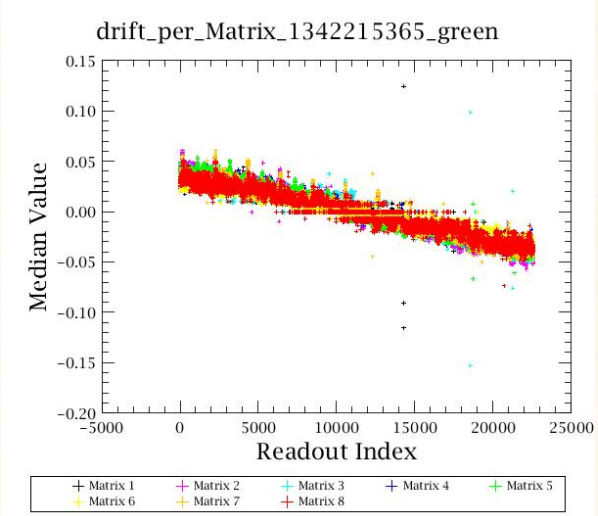


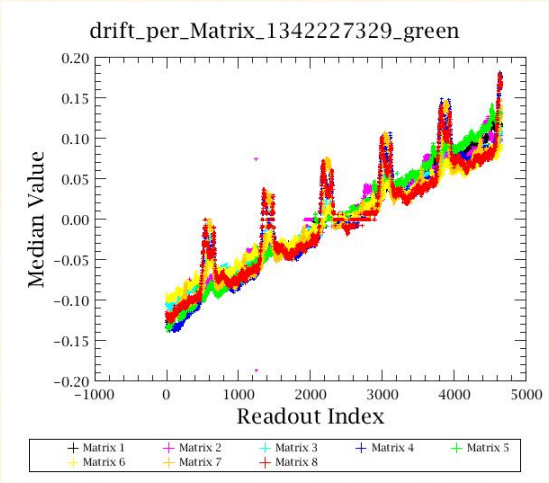
Median vs. Time

When present, this is the strongest trend in PACS cubes

Field guide to signal drifts

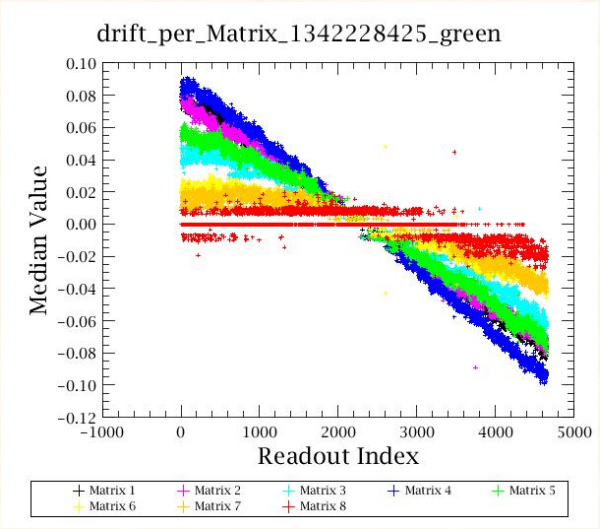
Typical.

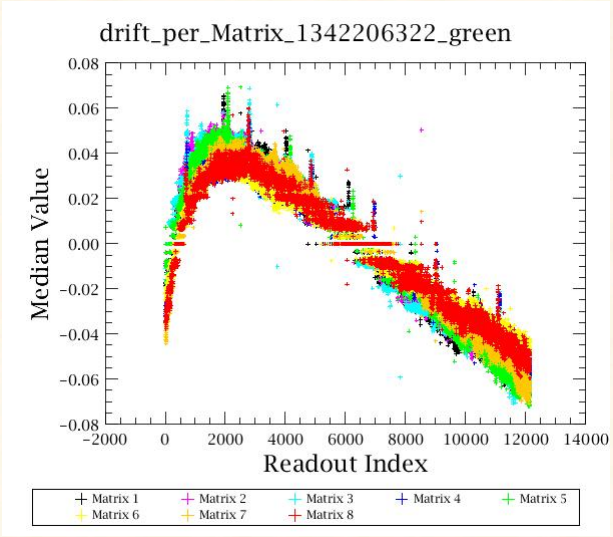




Rare.
Ascending

Strong module-to-module differences

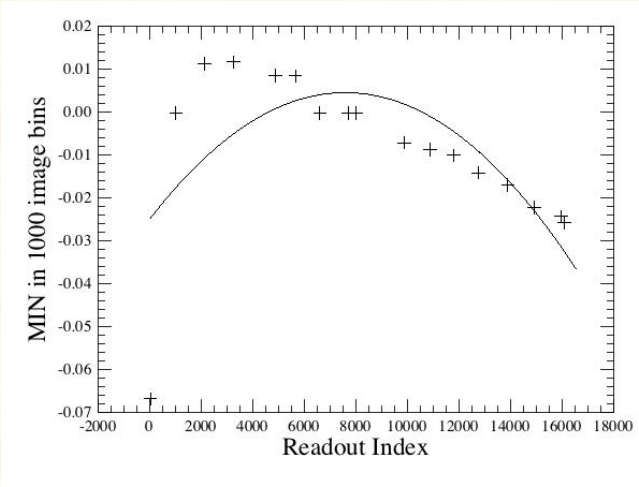


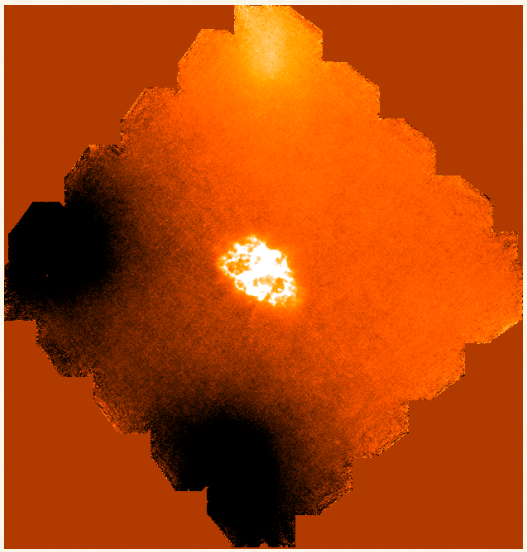


Hook

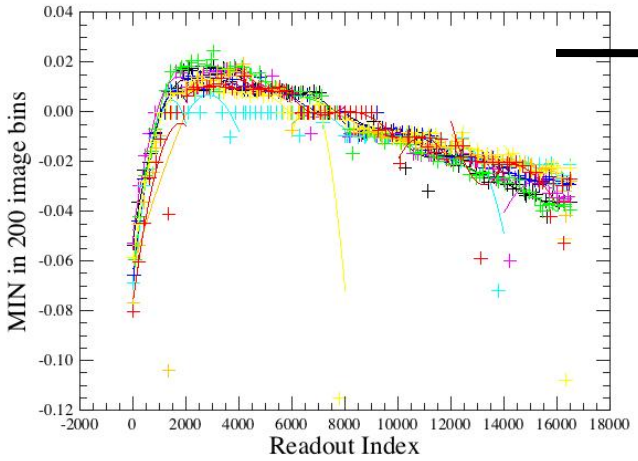
Proper correction is crucial

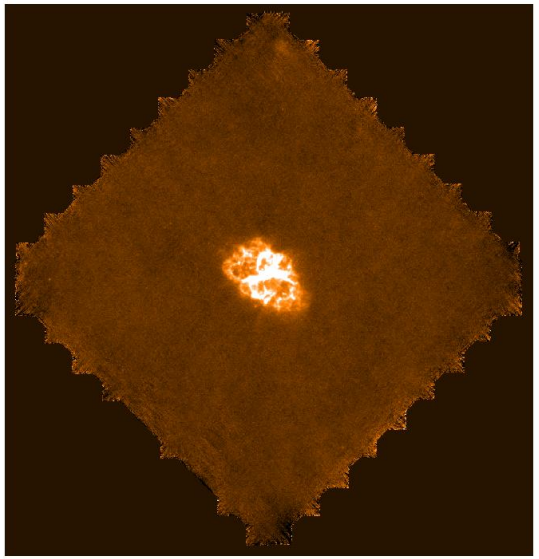
Bad





Good





What to do?

- Depending on the complexity of the drift, you may need to iterate with different bin sizes and different orders.
- Complicated module-to-module and “hook” drifts require segmenting the timeline in smaller chunks.
- Ensure that you plot the medians and the fit (doPlot=True) and you are satisfied with the fit.

Rolling your own

- You can define/fit your own baselines, using any method you feel comfortable with.
- You can do that to further customize the baseline to your data:
- These fits can be calculated individually per matrix, or for the whole array.
- These fits

II. MADmap

The actual map-making

Step 8

Execute the single line

```
339  #
340  #
341  # Make the Time Ordered Data array
342  tod = makeTodArray(joinframes, calTree=calTree, scale=scale)
343  # To scale output pixle size or to rotate the final map:
344  # tod = makeTodArray(joinframes,calTree=calTree,scale=myscale,crota
345  #
```

The ToD stands for Time-ordered-Data and is the internal format used by MADmap.
In fact, makeTodArray will create a binary file in your temporary area that has the rearranged PACS signal in the proper format.

- *The scale parameter selects the size of the output sky grid relative to the nominal PACS pixel sizes. E.g. scale=0.5 for PACS blue channel will result in final pixel sampling of 1.6”/pixel.*

Step 9

Select and execute this block of commands

```
maxRelError    = 1.e-5
maxIterations = 500

naivemap = runMadMap(tod,calTree,maxRelError,maxIterations,True)
madmap=runMadMap(tod,calTree,maxRelError,maxIterations,False)
```

Documentation Reference:

PDRG Chapter 9

Both the naive and optimal maps are created with the same call. The last parameter is set to 'True' for naive map and 'False' for optimal map. MADmap uses maximum likelihood and conjugate gradient solvers to find the optimal solution. The parameters maxRelError and maxIterations control both the convergence tolerance and the number of iterations in finding the optimal solution. See the above reference for details.

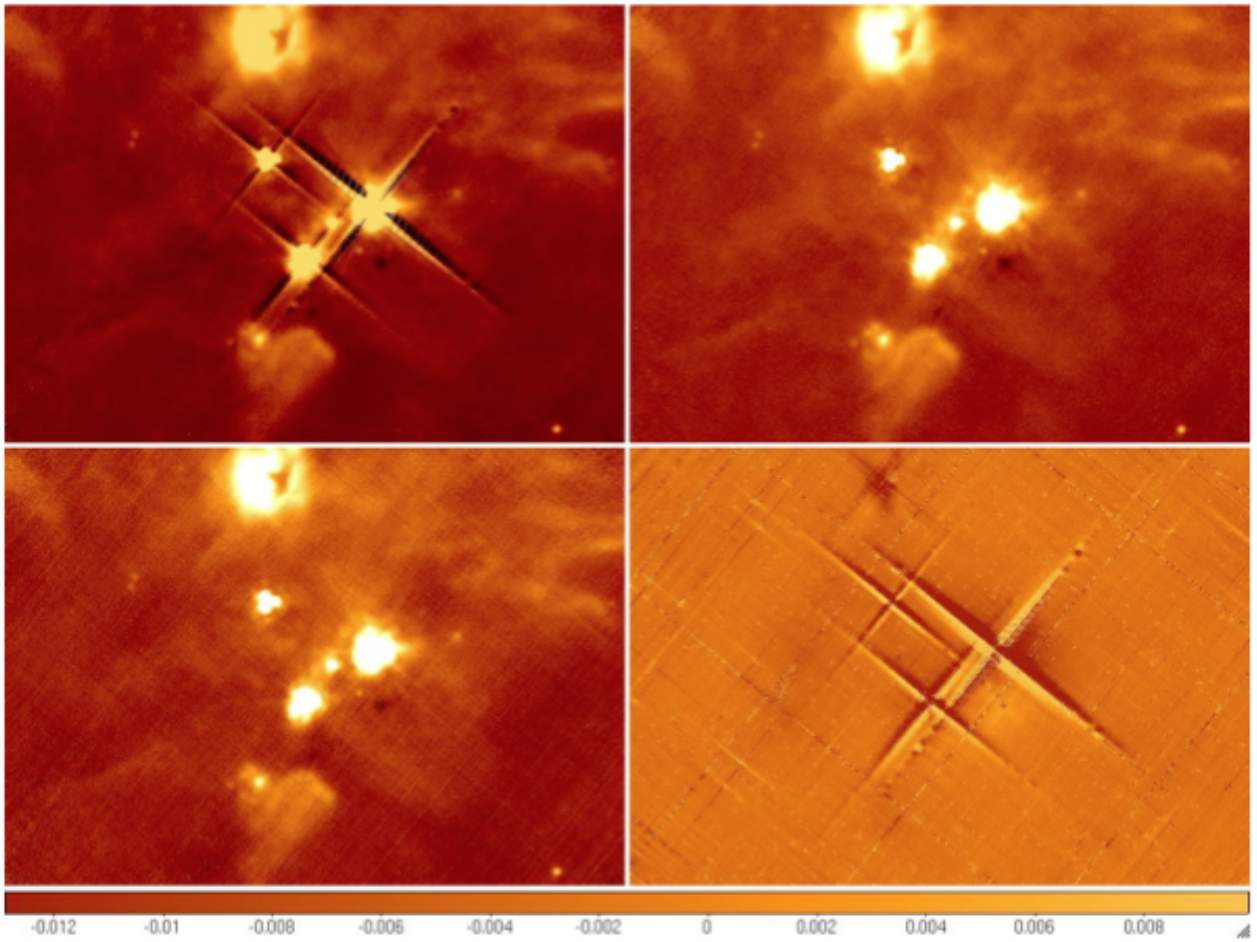
II. Post-processing

Correct the final map for point source artifacts

Point-source artifacts

Original map

Naïve map



Corrected map

Artifact map

Figure 1 Results from the point source artifact removal. Top Left. Level 2.5 map generated by MADmap. Top right, the corrected MADmap map. Bottom left the naive (simple projection) map. Bottom right. The artifact map.

How does it work?

- Reference:
  - Lorenzo et al. in prep (available on request)
  - Developed for HiGal data processing.
- Uses the optimal map as the starting point for sky emission.
- Subtract sky from original time lines, which leaves only noise +artifacts in the time-line.
- High-pass filter the noise+artifacts time-lines. This removes noise and leaves on artifact.
- In practice, several iterations are necessary.

Step 10

Execute the block of lines

```
354  #
355  # Do point source artifacts correction if requested
356  if doPGLScorrection:
357     print "do point source artifacts correction"
358     correctedmap = photCorrMadmapArtifacts(joinframes, tod, madmap, PGLS_iterations)
359     #Display(correctedmap)
360  #
```

The doPGLScorrection flag is set at the beginning of the script. If set, the correctedmap variable will contain the artifact free map.
See PDRG Section 9.5 for details.
The number of interations for the PGLS algorithm are set in the PGLS_iterations variable (at the start of the script).

When used to decode data from the bolometers in the PACS instrument aboard the Herschel satellite (left), **MADmap** yields images like this 2 x 2 degree area of the sky in the constellation of the Southern Cross.