Unimap

L. Piazzo

DIET dept. - University of Rome 'La Sapienza'

Outline

- Unimap Overview
- Input and Output
- Pipeline:
    - TOP
    - Pre
    - Glitch
    - Drift
    - Noise
    - GLS
    - PGLS
    - WGLS
- Unihipe
- Conclusion

Overview

Overview

Unimap

Map maker (not only) for Herschel data.

Successor of ROMAGAL. Same quality. Simpler to install and use.

Development started in 2011, within the Hi-GAL project.

Now a stand alone project leaded by the University of Rome.

Partially funded by ASI.

Team

DIET-University of Rome: L. Piazzo, D. Ikhenaode.

IAPS-INAF: M. Pestalozzi , S. Pezzuto, D. Elia, E. Schisano.

ASDC-ASI: L. Calzoletti, F. Faustini.

Overview

Features

Implements a full pipeline starting from Level 1 products.

The map maker is the GLS/ML (e.g. MADMAP, Sanepic, Tamasis).

The implementation is efficient and original.

Two main novelties are introduced:

Post-processing algorithm for GLS map makers (PGLS)

- *"Artifact removal for GLS map makers by means of post-processing", IEEE Trans. on Image Processing, Vol. 21, Issue 8, pp. 3687-3696, 2012.*
- *"Distortion Analysis and Removal for PACS Instrument GLS Images", Submitted to the IEEE Sig. Proc. Lett., 2012.*

Subspace Least Square approach to drift removal (SLS)

- *"Subspace Lest Square Drift Removal with Application to Herschel Data", ArXiv 1301:1246, 2013.*

Overview

Documentation and download

Program and User’s Manual (5.3.0): w3.uniroma1.it/unimap.

Software environment

Source code is Matlab.

Can run where Matlab can run (Win, Linux, Mac).

Compiled versions can run w/out Matlab.

Compiled version currently distributed for Linux.

Efficient

Atlas blue (600 Mega samples) was reduced on a desktop with 12G RAM.

Most wshop tiles (up to 400 Msamp) were reduced on a laptop with 8G RAM.

Processing time for wide tiles ranges from 2 to 20 hours.

Overview

Flexible

Several projections and unit conversions implemented.
Each pipeline step can be executed separately.
Intermediate data saved to restart execution.
Input data can be downsampled to run fast tests.

Automatic/Interactive

Controlled by several parameters. Saves evaluation data.
In automatic mode defaults are used.
In interactive mode the user sets the parameters based on evaluation data.

Output quality is normally good in automatic mode.
Quality can be improved by tuning the parameters.

Input and output

Input and output

- **input data**

unimap_obsid_1.fits

…

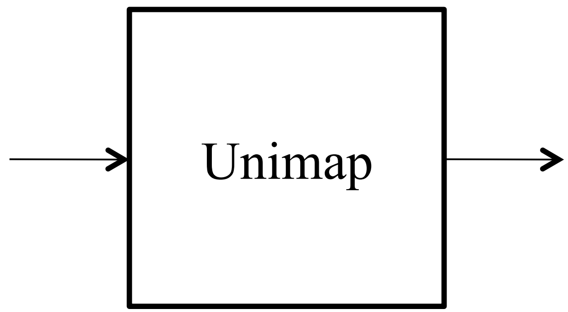
unimap_obsid_n.fits

- **par file**

unimap_par.txt

- **fits header**

unimap_header.txt



- **output images**

img_wgls.fits

img_pgls.fits

img_gls.fits

img_rebin.fits

img_high.fits

- **noise map**

img_noise.fits

- **text output**

unimap_log.txt

Several evaluation images and ancillary data are also produced.

Input and output

Parameter file. It is divided in sections corresponding to the processing steps

```
=============== % Params for Unimap 5.2.0 - Delete this file and run Unimap to produce defaults ======
../data/1004_16/        % data_path - working directory
250                     % max_ite_par - positive integer - global iteration limit
0                       % start_module - positive integer - first module to execute (1 = top, 2 = pre etc)
8                       % stop_module - positive integer - last module to execute (1 = top, 2 = pre, ..., 8 wgls)
0                       % save_eval_data - 0/1 - if 1 save evaluation data
0                       % save_tops - 0/1 - if 1 saves the intermediate tops
=============== % 1: Top ===========================================================
1                       % top_use_galactic - 0/1 - if 1 use galactic coords, if 0 keep equatorial
1                       % top_use_gnomonic      - 0 = no projection (CAR), 1 = gnomonic (TAN), 2 = cyl eq area (CEA)
1                       % top_bolo_sub          - positive integer - bolometers subsampling
30                      % top_max_bad           - real in [0,100] - max percent of flagged samples to accept bolo
0                       % top_unit  - if 0 MJy/sr, if 1 Jy/pixel, if 2 Jy/beam (SPIRE only)
............
0                       % top_cpi2 - real - ref point coord2 in the pixel plane  (pixel)
0                       % top_nax1 - Positive integer - number of pixels on the first axis - if zero use minimum
0                       % top_nax2 - Positive integer - number of pixels on the second axis
=============== % 2: Pre ===========================================================
0                       % pre_threshold - positive real - threshold for calibration detection 2
                        % pre_jump_threshold - positive real - Threshold for jump detection (0 suppress detection,.
25                      % pre_jump_hfwin - positive real - Window length for hcb (e.g. 25).
0                       % pre_onset_len - positive integer - len of onset (samples) - if 0 suppress onset removal
=============== % 3: Glitch ===========================================================
25                      %glitch_hfwin - positive integer - half len of the highpass filter (samples)
0                       % glitch_sub - positive integer - subsampling for glitch search (pixels).
0                       % glitch_max_dev - positve real - threshold to declare a readout a glitch
=============== % 4: Drift ===========================================================
............
```

Input and output

Header file. The user can customise the output fits header (only chars currently supported):

```
EQUINOX = 2000.0
WAVELNTH= 160
COMMENT = this map is very beautiful
........
```

The Input is organised in observations.

Each observation contains: timelines, pointing and flags.

Any number of observations can be processed (within RAM limits).

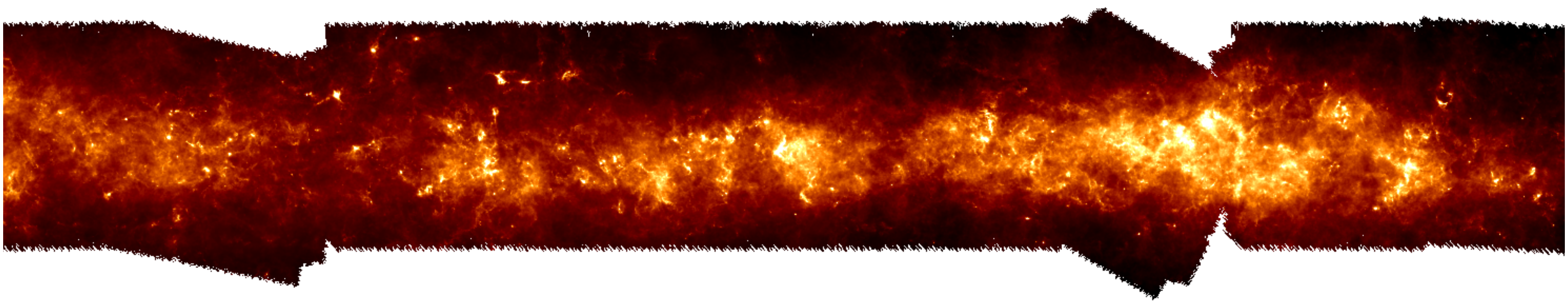


Image of 8 SPIRE PLW Hi-GAL tiles: L308-L325. Default parameters. Galactic coord. Cylindric projection.

Pipeline

Unimap Pipeline and main data

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

TOP	Pre	Glitch	Drift	Noise	GLS	PGLS	WGLS
top_base	*top_pre*	*top_glitch*	*top_drift*	*noise_spec*	*img_rebin*	*img_pgls*	*img_wgls*
cove_full	*flag_pre*	*flag_glitch*		*noise_filt*	*img_gls*	*delta_gls_pgls*	*flag_wgls*
		img_high			*img_noise*	*delta_pgls_rebin*	
					delta_gls_rebin		

TOP:	astrometry and offset removal. (Time ordered pixels: TOP).
Pre:	jumps and onset detection and removal
Glitch:	glitch detetction and removal
Drift:	drift estimation and removal
Noise:	noise spectrum estimate
GLS:	noise removal with GLS
PGLS:	removal of GLS distortion
WGLS:	minimisation of PGLS noise

TOP - Time Ordered pixels

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

TOP

➢ Open the input and stack the readouts into a NxR matrix of N timelines with R readouts. Same for the flags.

➢ Convert unit and pointing, build the pixelisation grid and assign each readout to a pixel.
Coordinate: equatorial (EQU) or galactic (GAL).
Projection: gnomonic (TAN) or cylindric (CAR, CEA).
Astrometry: automatic or controlled by the user.

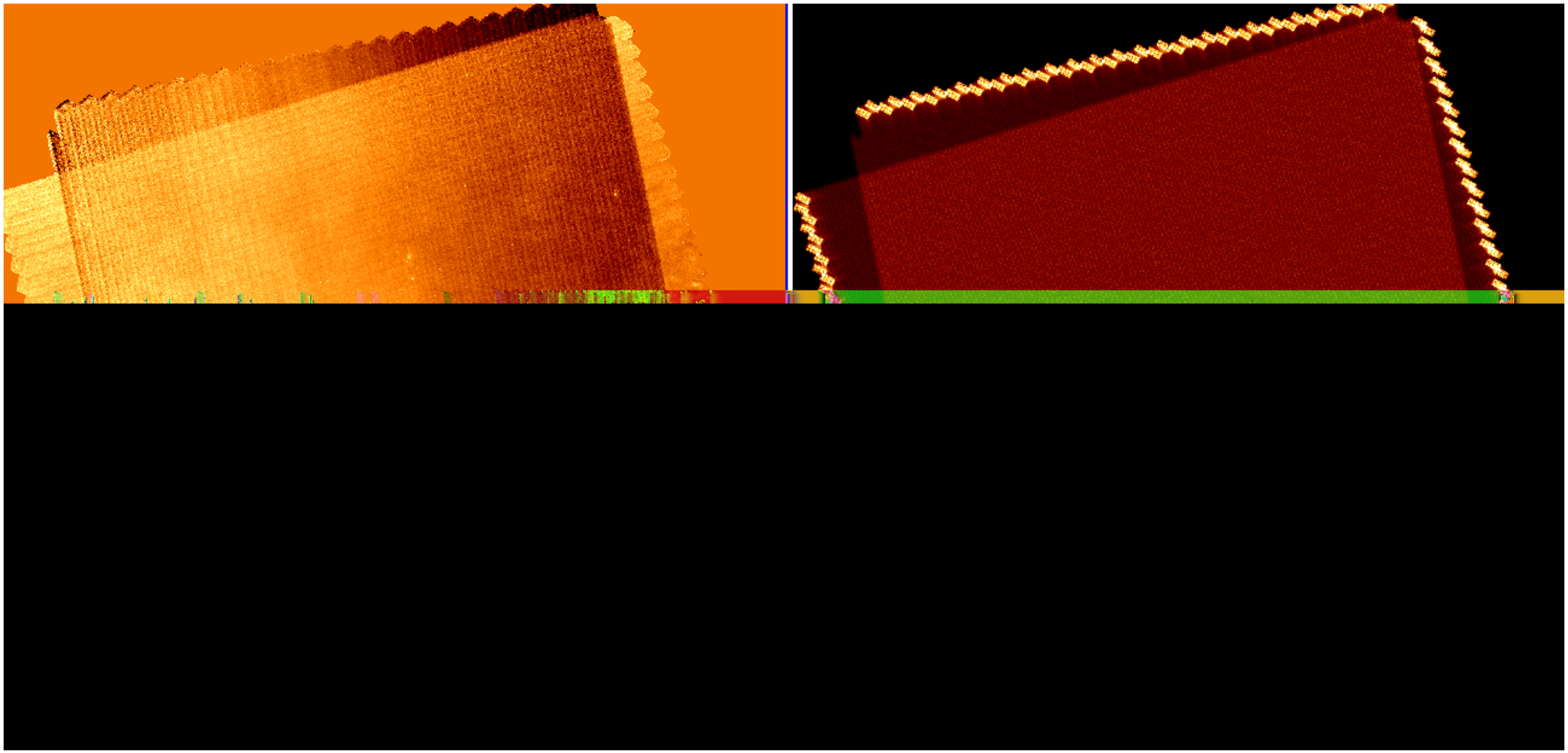
➢ Remove offsets by subtracting the median from each timeline.

Output: *top_base*, *cove_full*, ancillary data.

TOP

top_base.fits Naive map after offset removal.

cove_full.fits Number of readouts in each pixel.



L004 blue. *top_base* *cove_full*

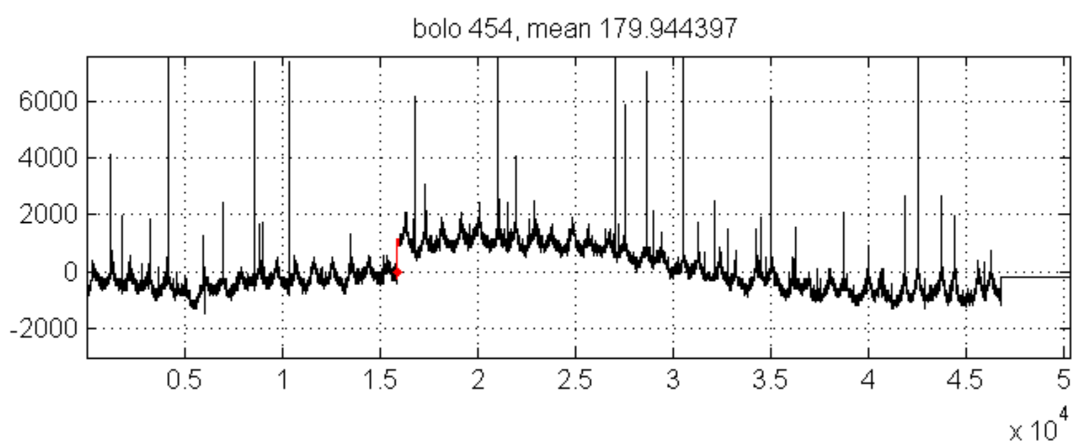
Pre - pre-processing

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

Pre - Pre-processing

```
=============== % 2: Pre ============================================================
2            % pre_jump_threshold - positive real - Threshold for jump detection
25           % pre_jump_hfwin - positive real - Window length for hcb (e.g. 25).
0            % pre_onset_len - positive integer - len of onset (samples) - if 0 suppress onset removal
```

- Detect and flag signal jumps
  - Each timeline is segmented into blocks of *2 jump_hfwin* samples.
  - Candidate detected when difference of median of blocks exceeds *jump_threshold.*
  - Morphological rules are used to eliminate false candidates.



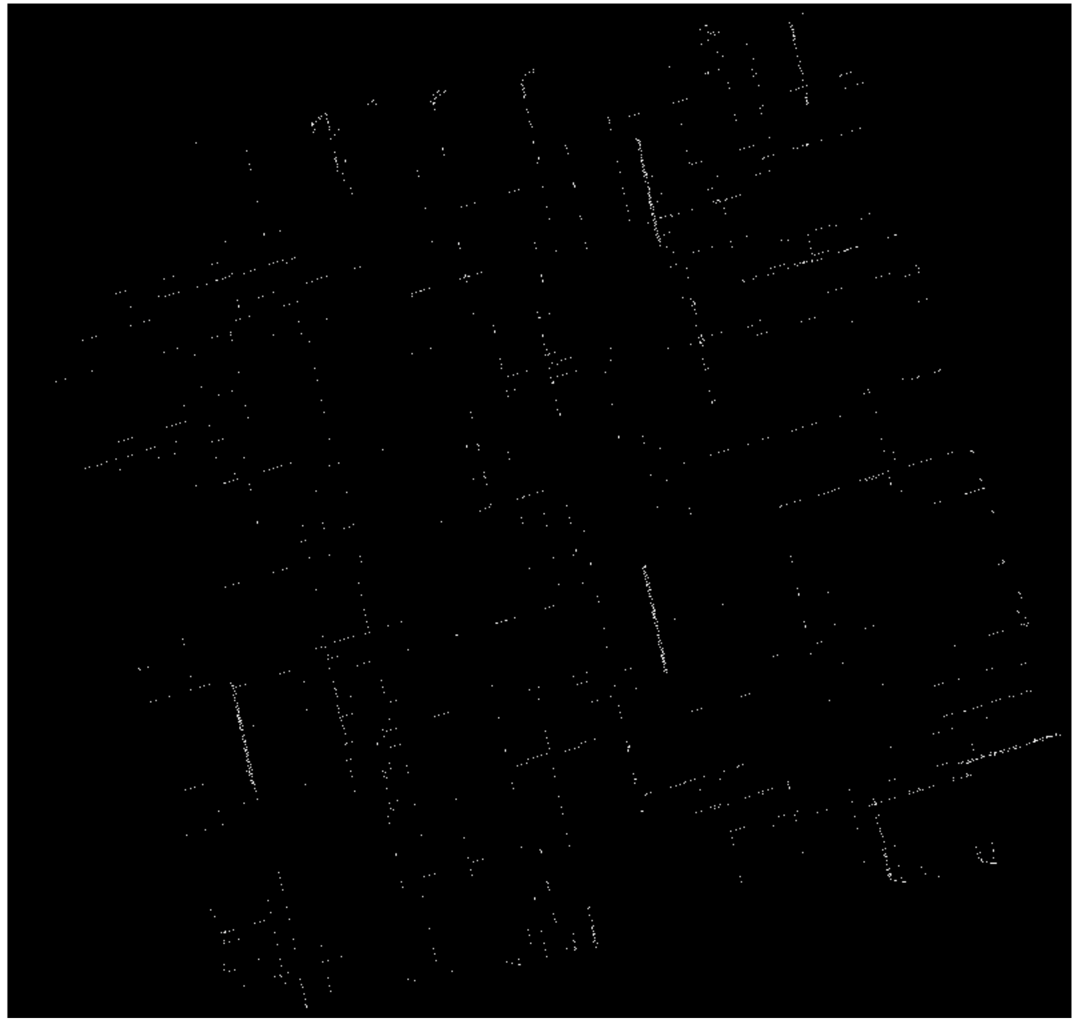
Pre

flag_pre.fits

Number of jump and base flags in each pixel.

Useful to evaluate accuracy of jump detection

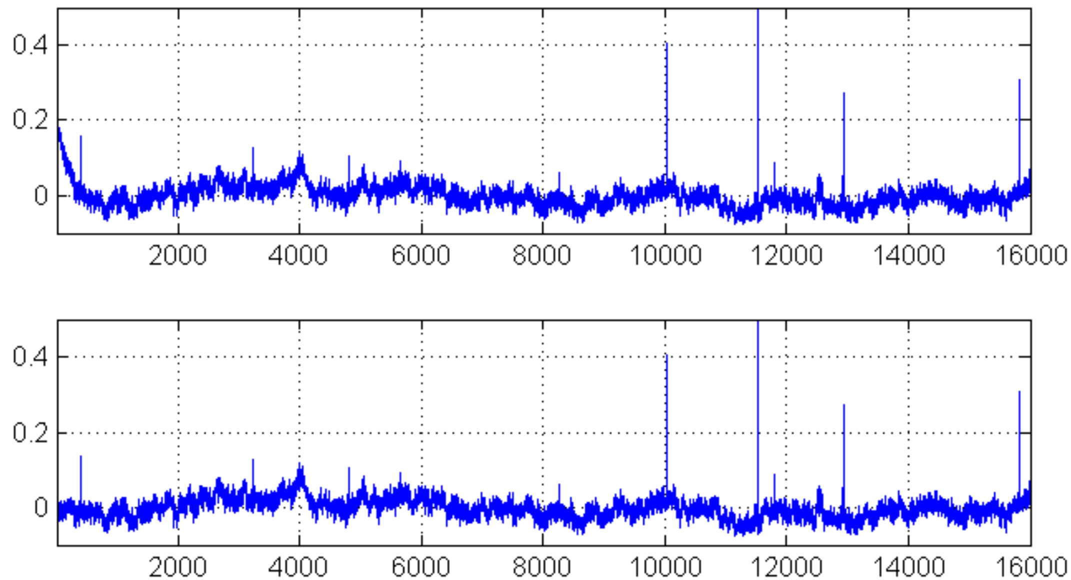
L004 blue.



Pre

➢ Correct the onset

- An exponential is fit and subtracted to the first *onset_len* samples of the timeline.

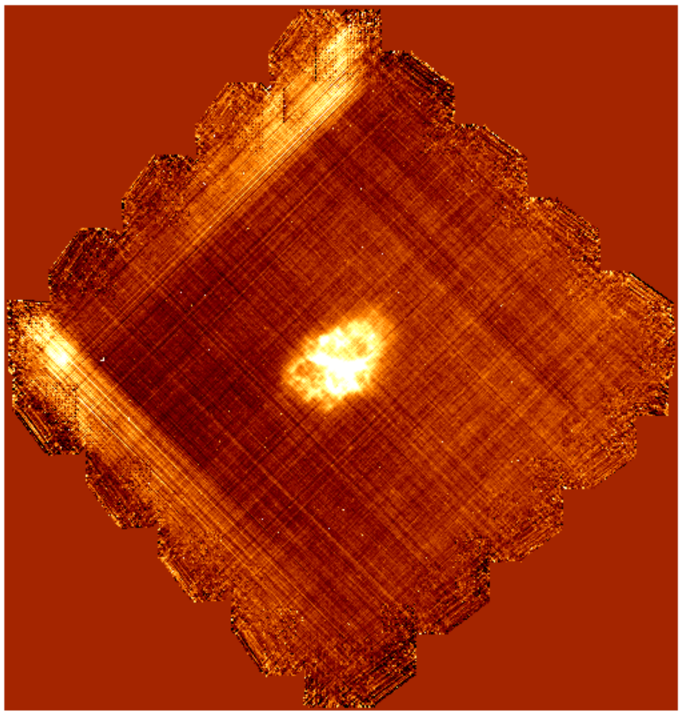


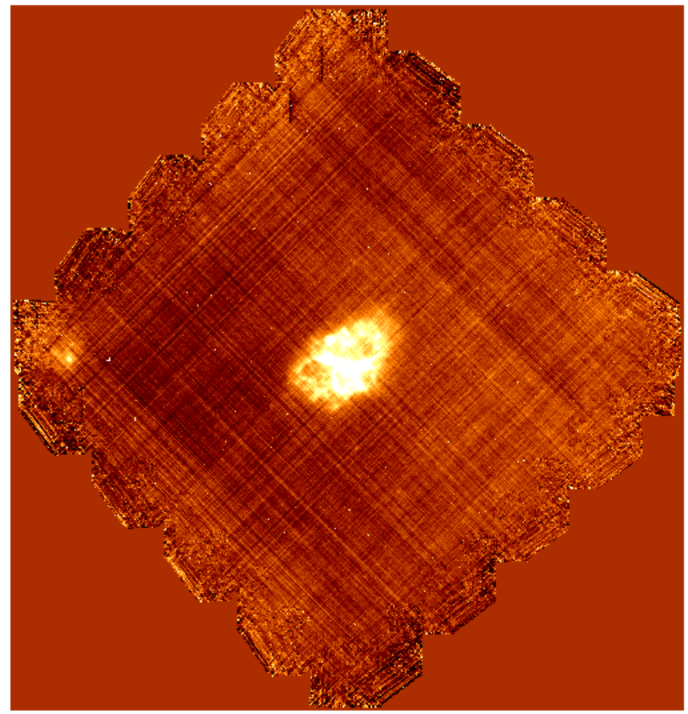
➢ Merge base flags and jump flags. Linearly fix the timelines.

Output: *top_pre, flag_pre*

Pre

Onset can be seen in *top_base*. Compare with *top_pre* to asses results





Crab blue. *top_base* *top_pre*

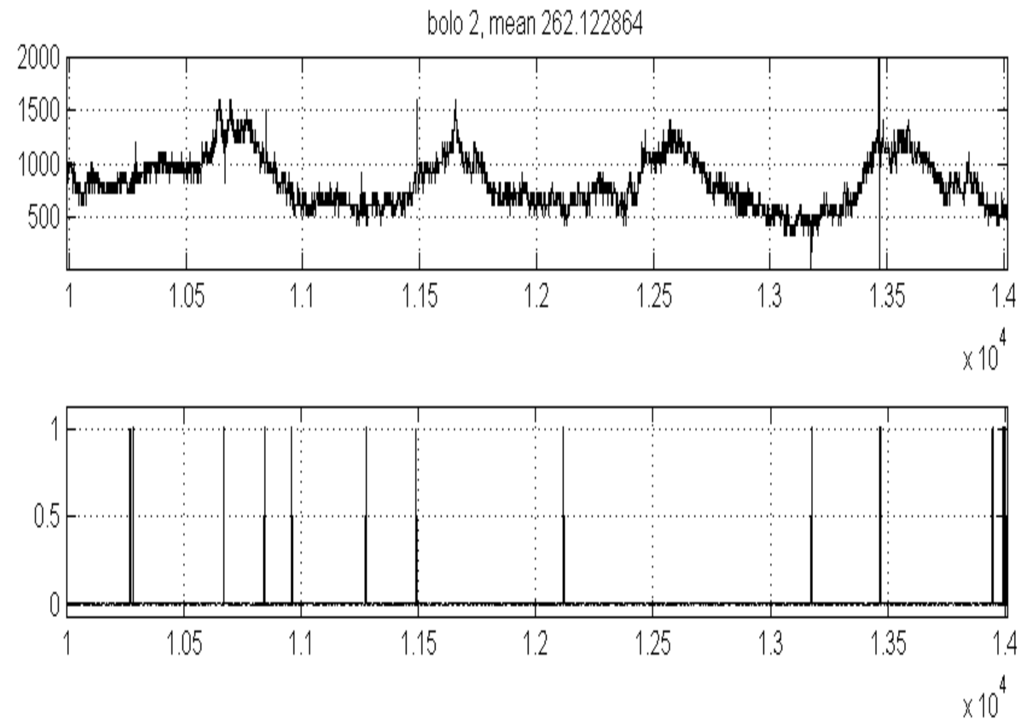
Glitch

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

Glitch

```
============== % 3: Glitch ===========================================================
25           % glitch_hfwin - positive integer - half len of the highpass filter (samples)
0            % glitch_sub - positive integer - subsampling for glitch search (pixels). If zero compute it.
0            % glitch_max_dev - positve real - threshold to declare a readout a glitch - 0 use default.
```

- Detect and flag glitches
  - Highpass filter each timeline to remove drift and 1/f noise.
  - Possibly subsample the image to guarantee enough redouts per pixel.
  - In each pixel, flag readouts with a distance from the mean exceeding the threshold
- Linearly fix the timelines.



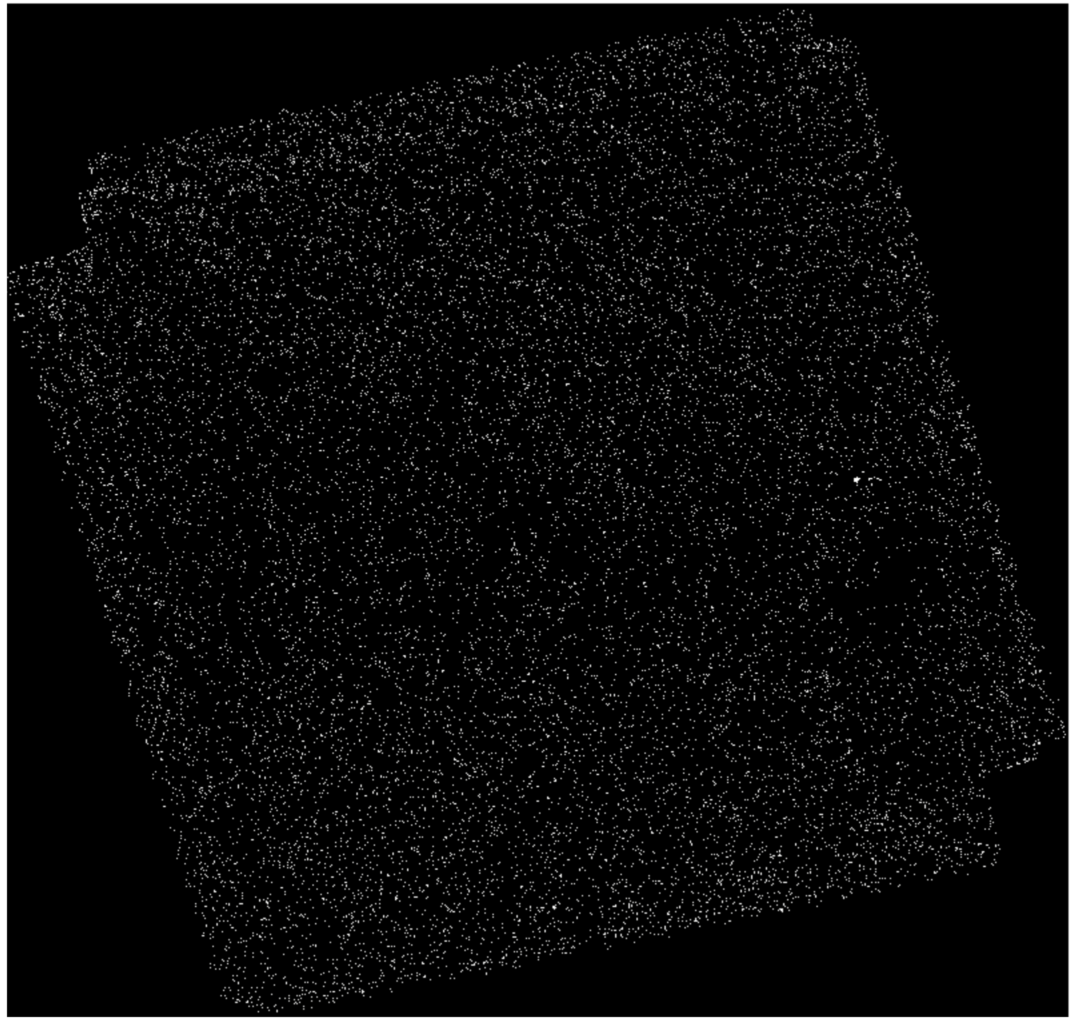
Output: *top_glitch, flag_glitch, img_high*

Glitch

flag_glitch.fits

Number of glitch flags in each pixel.

Useful to evaluate quality of glitch flagging



L004 blue.

SLS drift estimate

SLS Drift estimate

Subspace Least Square (SLS) drift estimation.

In vector space $\Re^n$ we use the following data model:

$$d \quad = \quad s + y + n \quad = \quad Pm + Xa + n$$

$s=Pm$	signal	P	pointing matrix	m	map
$y=Xa$	drift	X	Vandermonde matrix	a	coefficients
n	noise				

The signal lies in the subspace Σ spanned by the columns of P.

The drift lies in the subspace Δ spanned by the columns of X.

The signal is removed projecting d in $\Sigma^\perp$, the orthogonal complement of Σ.

The drift is estimated by means of least square in $\Sigma^\perp$.

SLS Drift estimate

The SLS estimate is $y^* = \hat{X}(\hat{X}^T \Pi \hat{X})^{-1} \hat{X}^T \Pi d$

$\hat{X}$ is obtained from X removing columns in $\Sigma \cap \Delta$

$\Pi = I - P(P^T P)^{-1} P^T$ is the orthogonal projector in $\Sigma^{\perp}$

The estimate can be obtained iteratively:

using Parallel Conjugate Gradient (PCG)

using the SPIRE destriper approach

The drift component falling in $\Sigma \cap \Delta$ is not detected in the SLS estimate.

For Herschel, this means that the estimate is affected by an offset (irrelevant).

SLS is rock solid and can be extended to other, non polynomial, drift models (e.g. sinusoidal).

Details can be found in the ArXiv report.

Drift

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

Drift

```
=============== % 4: Drift ===========================================================
3            % drift_poly_order - positive integer - polynomial order
.1           % drift_min_delta - real - minimum improvement to continue iteration (dB)
0            % drift_each_bolo - -1/0/1 - if -1 skip dedrift, 0 dedrift by subarray, 1 dedrift by bolo
```

➢ Estimate and remove the drift using SLS

Polynomial order and iteration stop are passed as parameters.

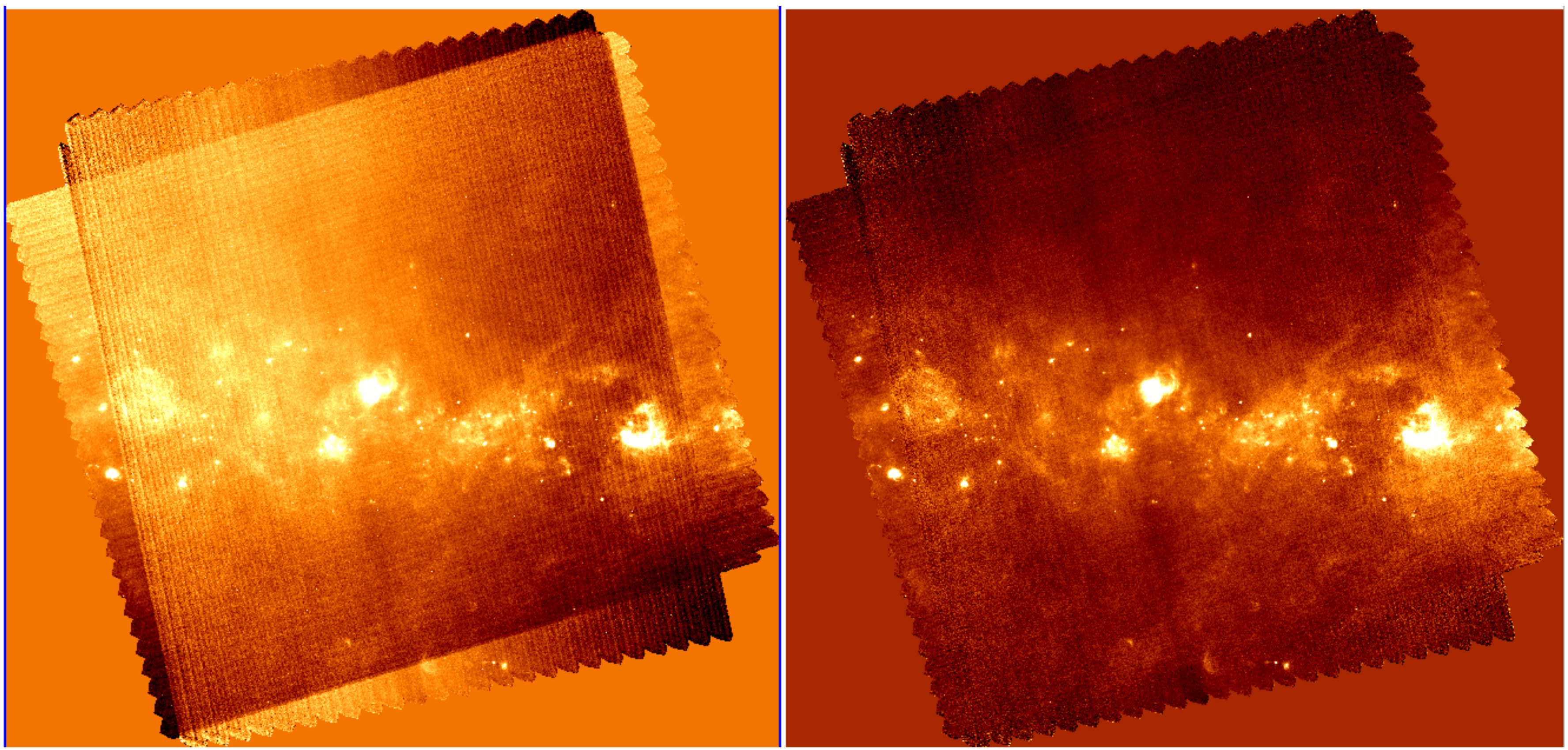
Can estimate a drift for each timeline (better but slower) or for a whole subarray (faster and GLS will take care of individual drifts).

Polynomial drift model adequate only for low poly order (say 3).

Output: *top_drift*

Drift

Compare *top_pre* (or *top_glitch)* with *top_drift* to asses the dedrift results



L004 blue. *top_pre* *top_drift*

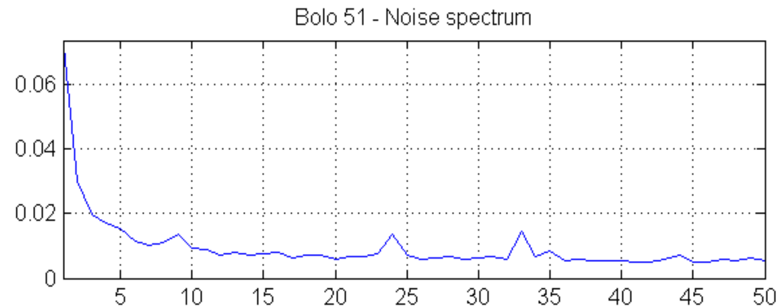
Noise

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

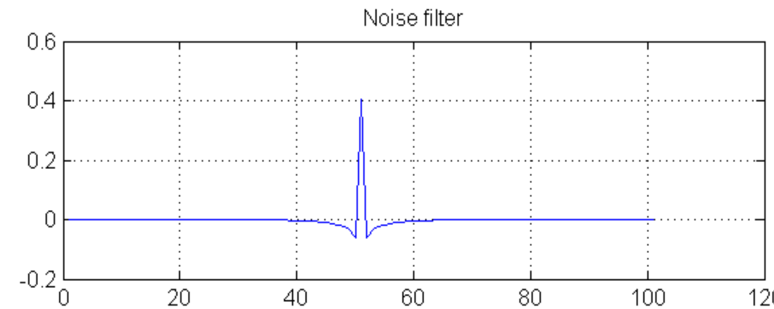
Noise

```
=============== % 5: Noise ============================================================
1   % noise_apply_flag - if 0 use the reconstructed readouts, if 1 remove flags
3   % noise_filter_type
50  % noise_filter_hflen - positive integer - half len of the gls noise filter response (samples), e.g. 50
```

- Estimate noise power and spectrum
  - Noise estimated by subtracting to each timeline the backprojection from the naive map.



- Build the GLS filter by IFFT of reciprocal of noise spectrum.



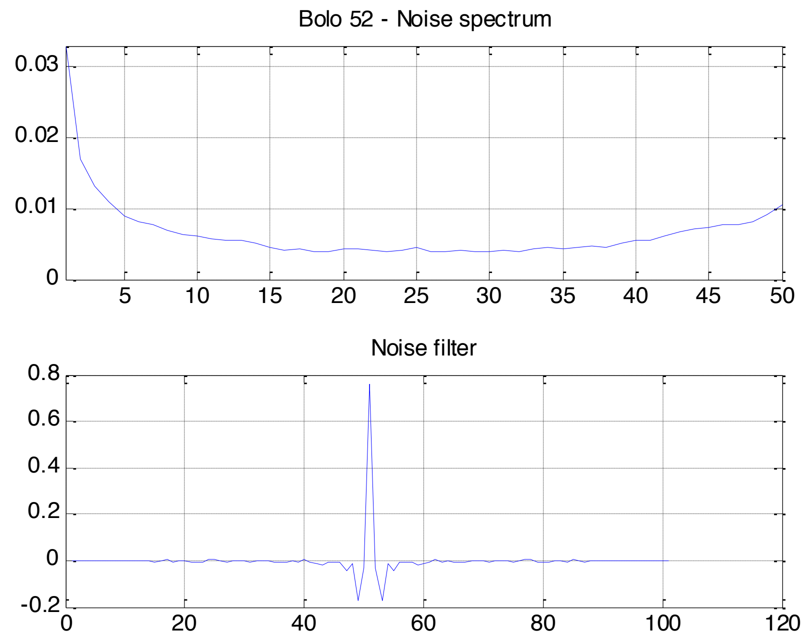
  - Filter len and type controlled by parameters.
  - Estimated or equal power
  - Fit to 1/f plus white noise or not
  - Filter per timeline or average filter

Noise

When processing SPIRE level 1 data the fit option shall be used with caution.

The noise spectrum rises at high frequencies due to the SPIRE highpass filtering.

In the example a filter produced from the raw spectrum for a PLW timeline.



- Format the timelines

  • By default flags are ignored by GLS but it is possible to use reconstructed readouts (useful if too many flagged pixels)

  • Where jumps were detected the timelines are broken in segments

Output: *noise_spec, noise_filt,* ancillary data

GLS – Generalised Least Square

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

GLS

```
=============== % 6: GLS ========================================================
0           % gls_continue - if 1 start from the saved data of the last step
1           % gls_start_image - 0 zero img, 1 rebin, 2 highpass.
0           % gls_apply_precond - 0/1 - if 1 do preconditioning
-60         % gls_min_delta - real - minimum change in the map to keep iterating (dB)
```

Data model $d = s + n = Pm + n$ with covariance noise matrix N

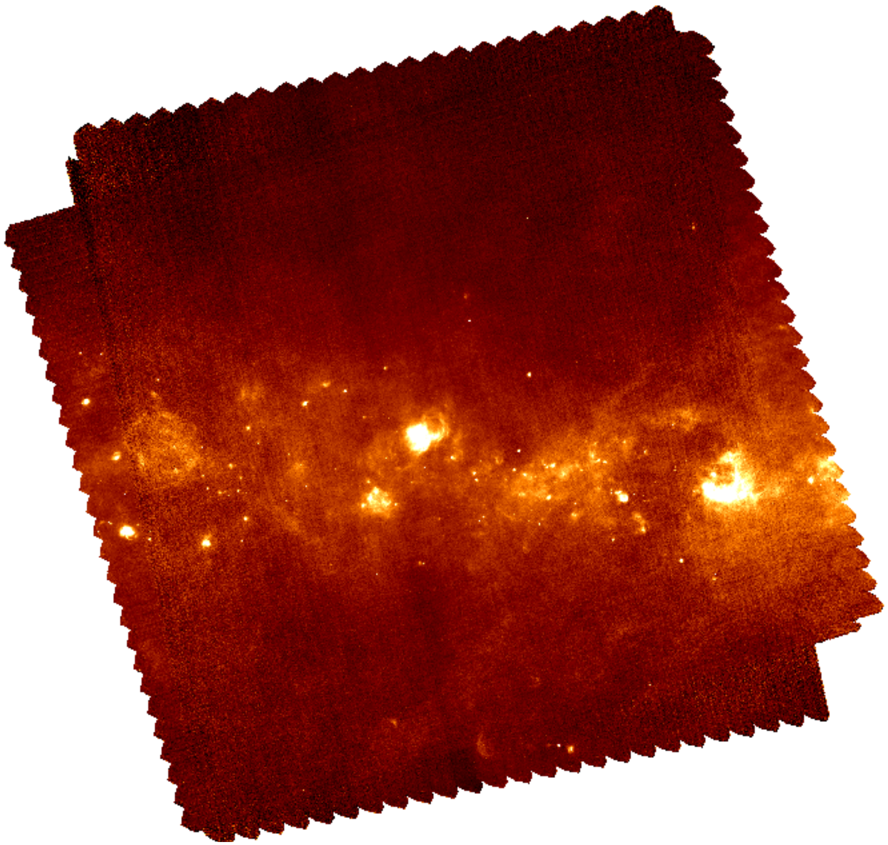
➢ Compute the GLS/ML map estimate $m^* = (P^T N^{-1} P)^{-1} P^T N^{-1} d$

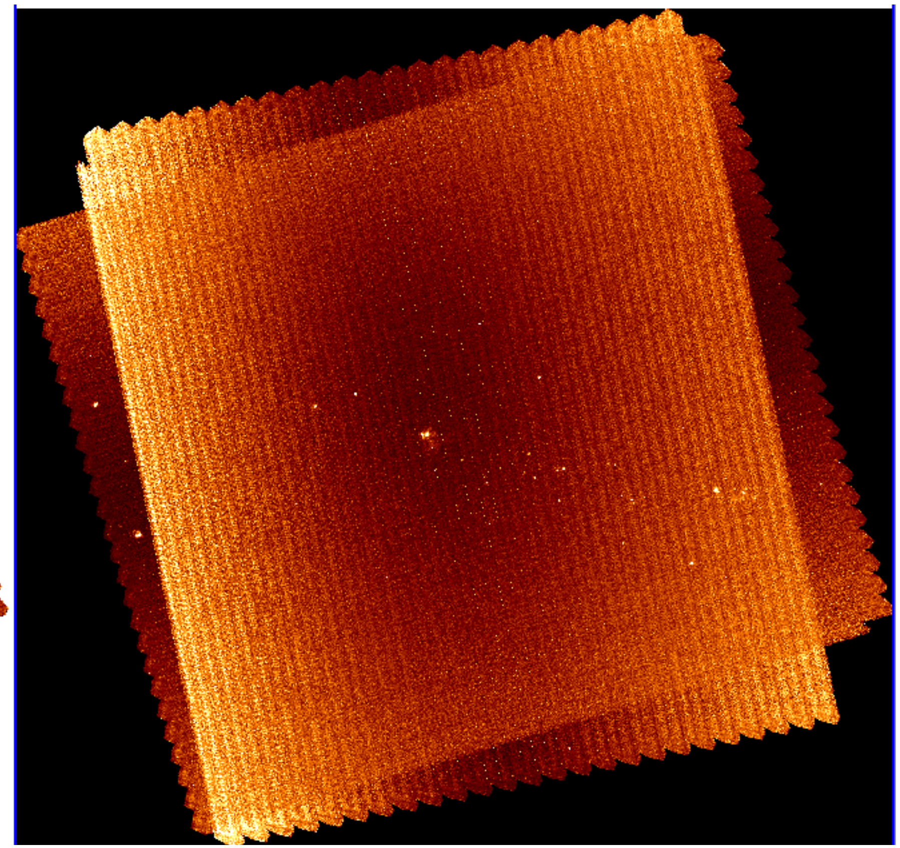
- Exploits iterative PCG solution
- Filtering is carried out in the time domain
- Timelines are segmented where jumps are detected
- Saves intermediate data to restart iterations
- Preconditioning and stop criterion controlled with parameters
- Can start from zero, naive or highpass image

Output: *img_rebin, img_noise, img_gls, delta_gls_rebin,* ancillary data

GLS

The naive map and its standard deviation are useful byproducts.





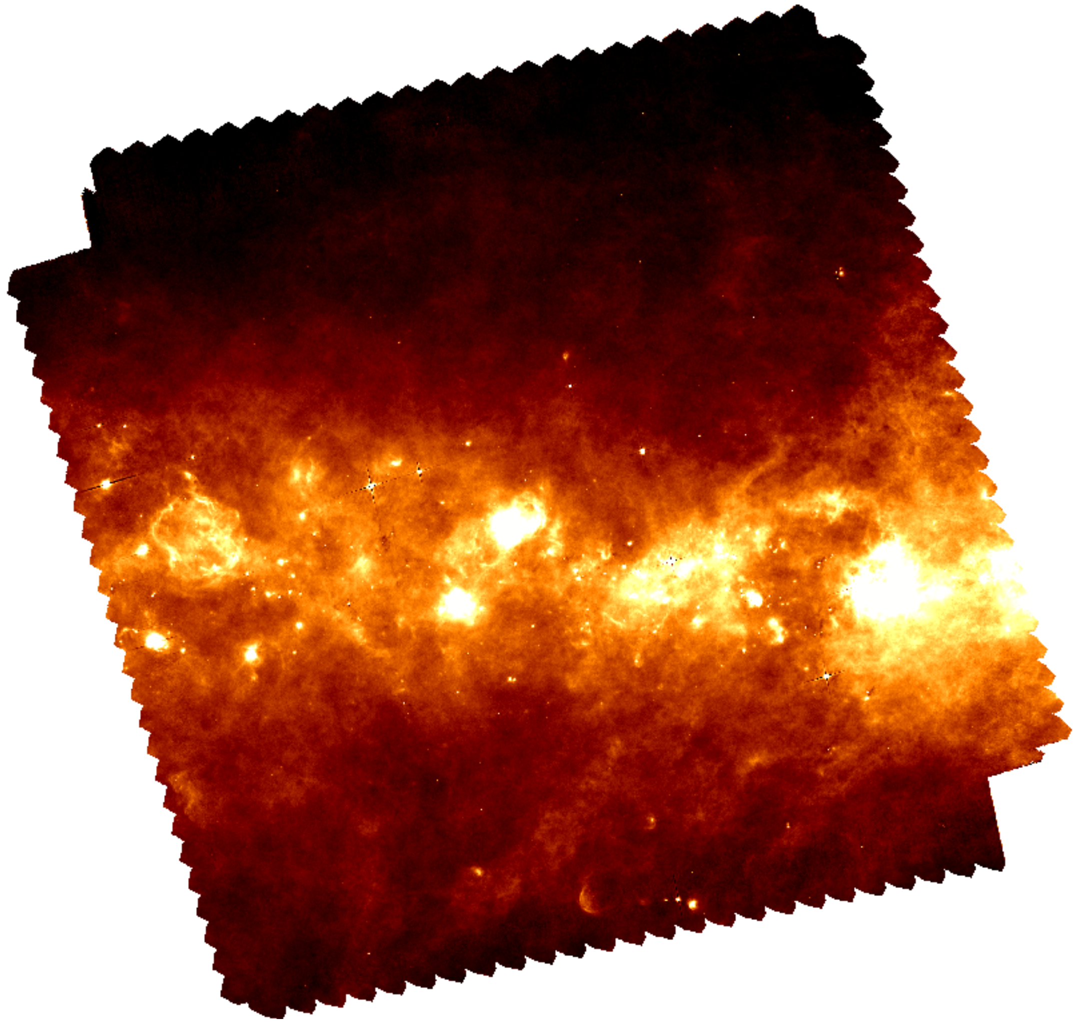
L004 blue. *img_rebin* (naive map) *img_noise* (STD of the naive map)

GLS

img_gls.fits

It is the GLSmap.

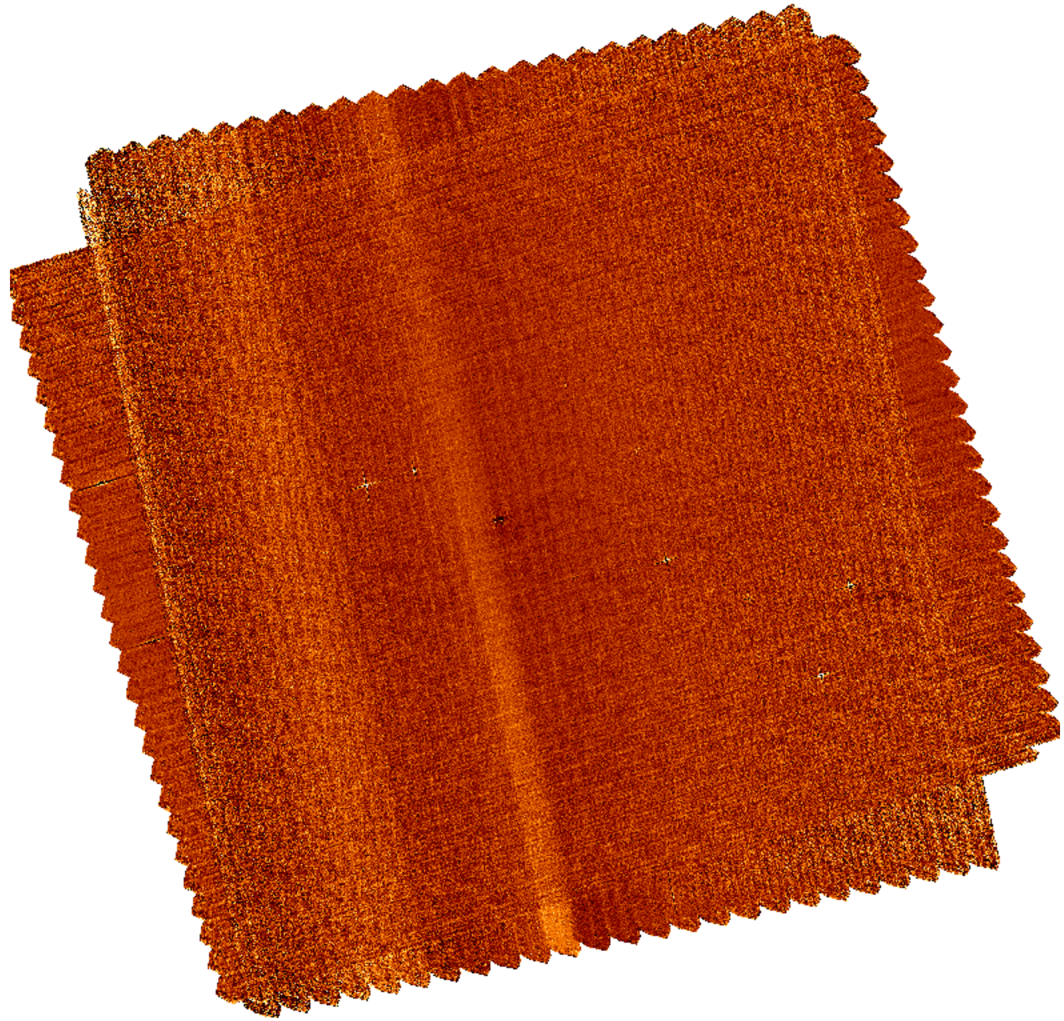
Map quality is nuch better than naive but there is distortion.



L004 blue.

GLS

delta_gls_rebin.fits



It is the GLS map minus the naive map.

Naive contains signal plus correlated noise

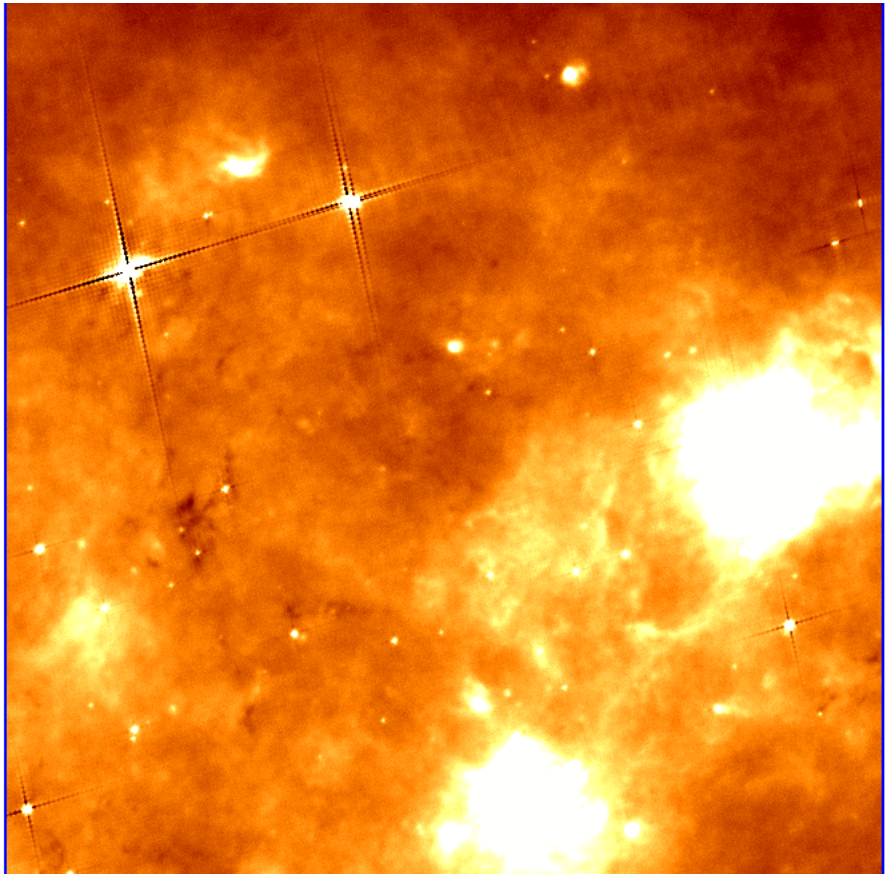
GLS contains signal plus possible distortion

Delta contains correlated noise plus possible distortion.

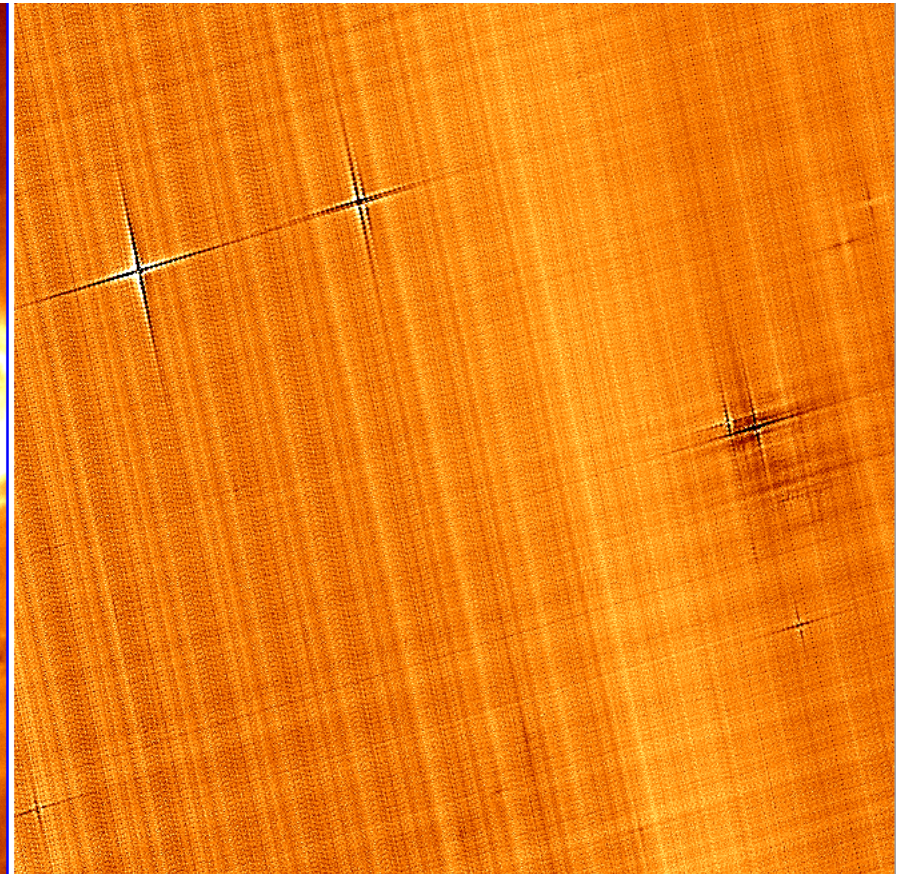
L004 blue.

GLS

GLS may introduce distortion. If distortion stronger than correlated noise it is seen in the delta map.



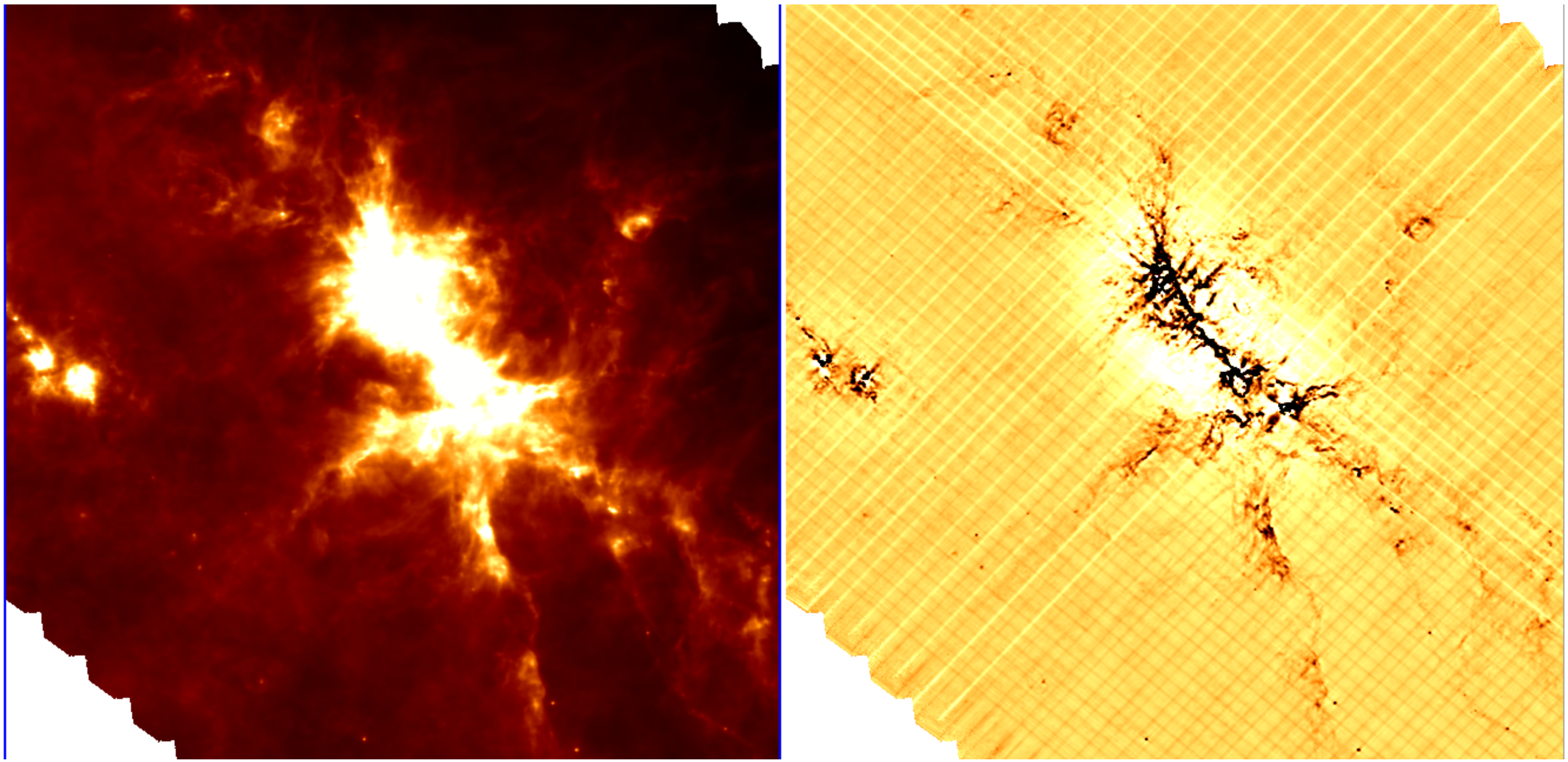
L004 blue. *img_gls*



delta_gls_rebin

GLS

Distortion may be cross-like, diffuse or absent.



NGC6334 red. *img_gls* *delta_gls_rebin*

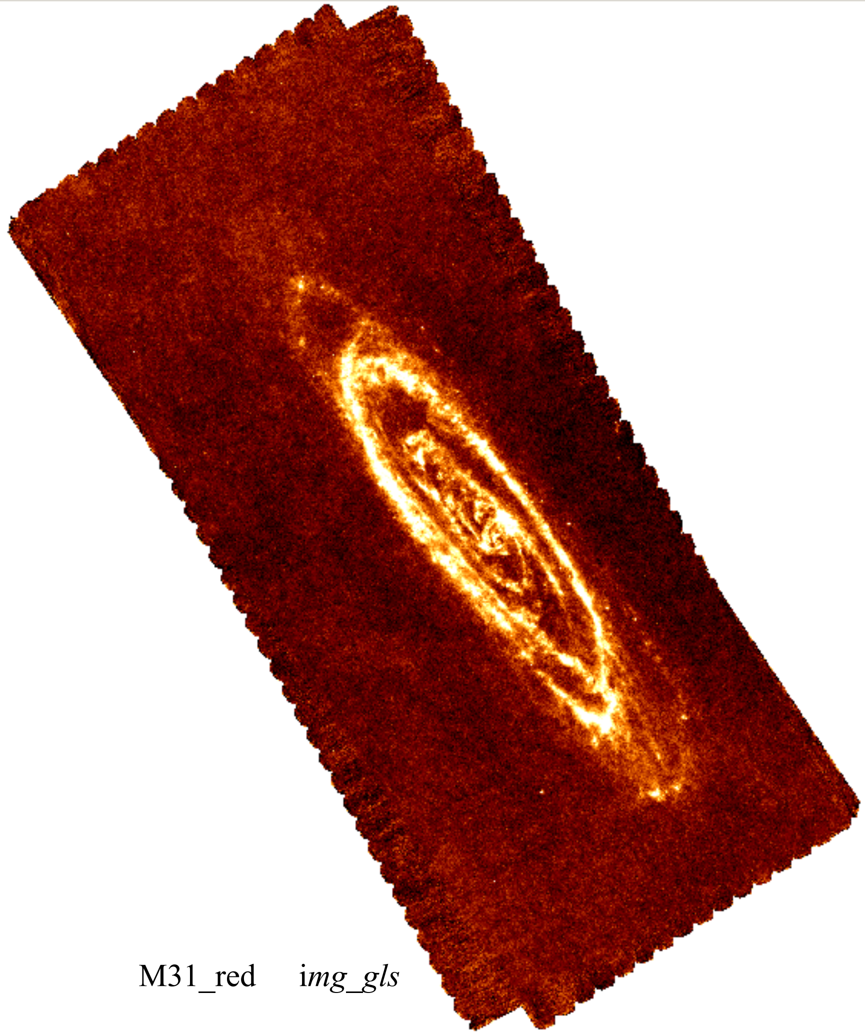
GLS

GLS is the most time consuming step.

Signal rich images are easier to reduce and require less iterations. Start from the naive map.

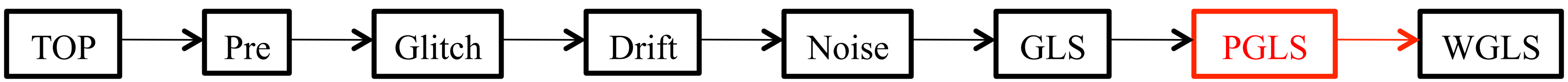
Images dominated by the background are more difficult and require more iterations. Start from a zero map.

A non flat background can often be corrected with more GLS iterations.



M31_red *img_gls*

PGLS – Post processing for GLS



PGLS

```
=============== % 7: PGLS ============================================================
0            % pgls_continue - if 1 start from the saved data of the last step
20           % pgls_hfwin - positive integer - half len of the pgls highpass filter (samples) e.g. 20
4            % pgls_num_ite - positive integer - number of pgls iterations to perform
```

➢ Estimate the GLS distortion and subtract from the GLS image to produce a clean PGLS image

- Intermediate data saved to restart iterations
- Number of iterations controlled with parameter
- Distortion estimated by means of median filtering and naive projection
- Noise level is increased. Negligible when image has high SNR

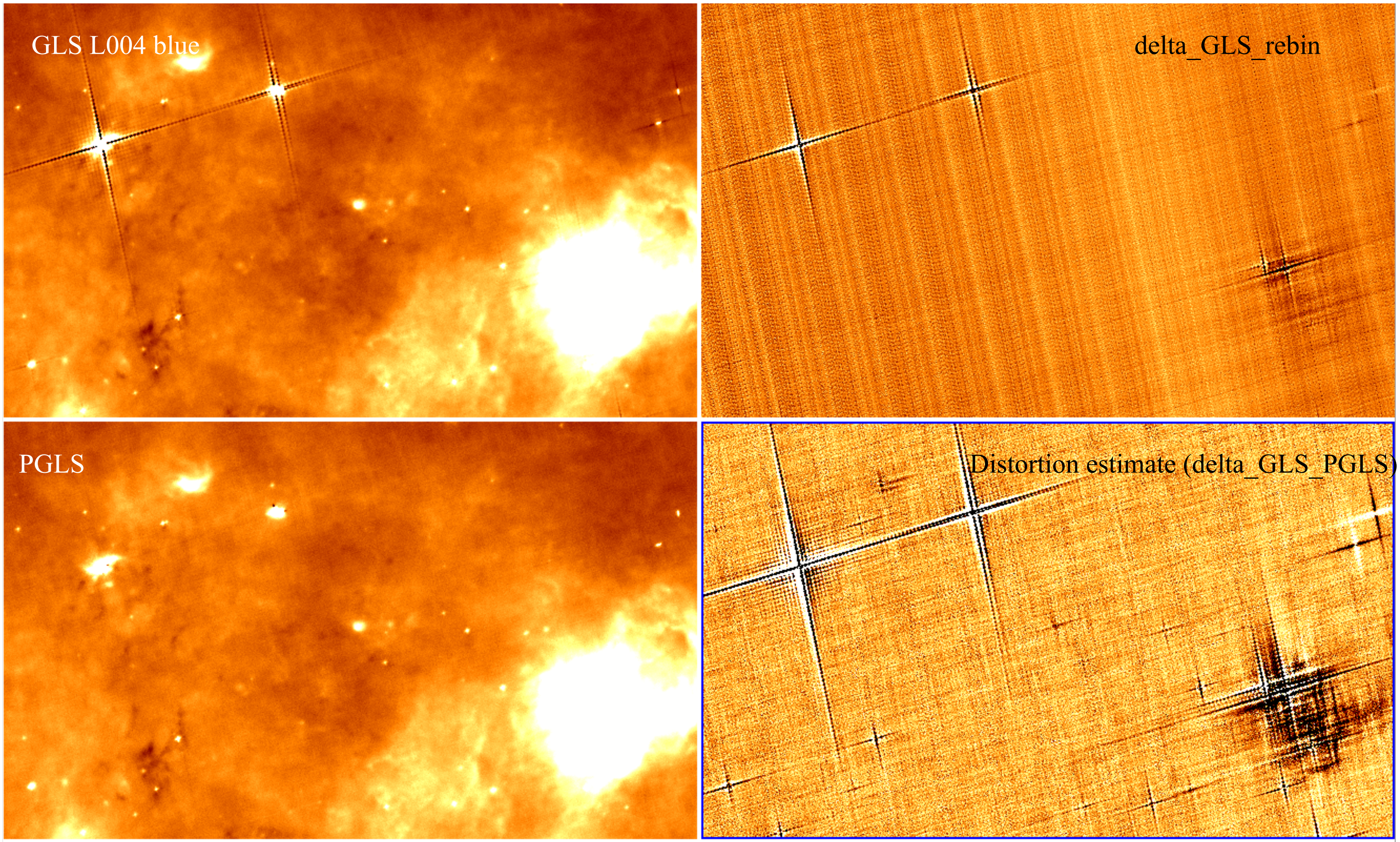
Details can be found in the paper.

PGLS is not needed if the GLS did not introduce distortion.

Output: *img_pgls, delta_gls_pgls, delta_pgls_rebin,* ancillary data

PGLS

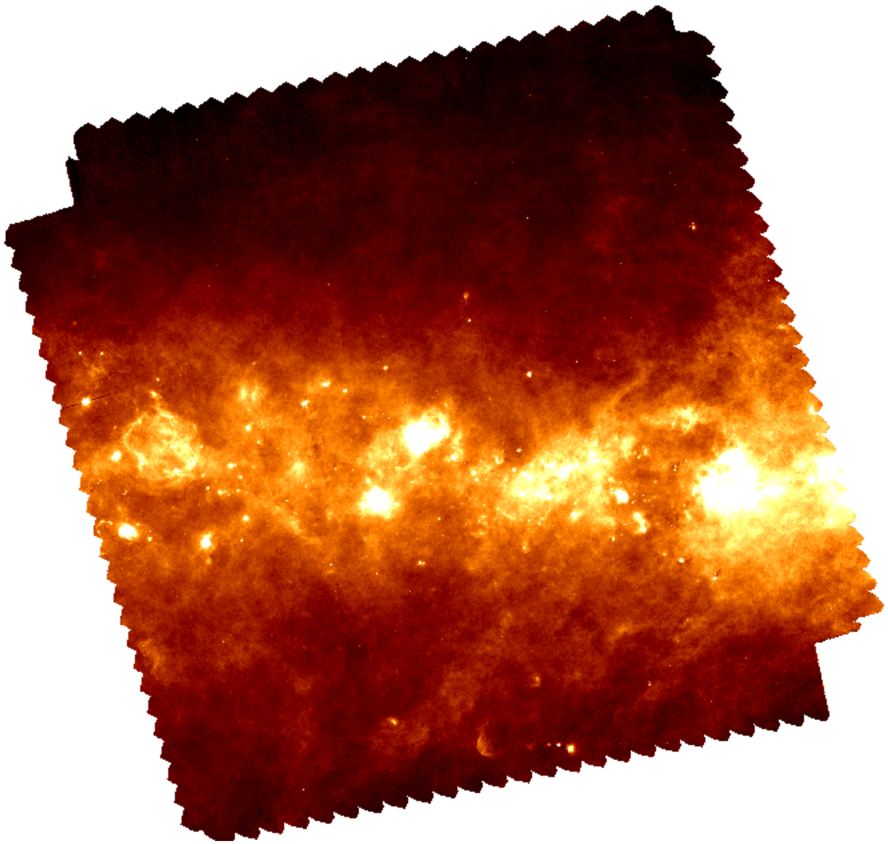
Compare *delta_gls_rebin* and *delta_gls_pgls* to asses the estimation accuracy.

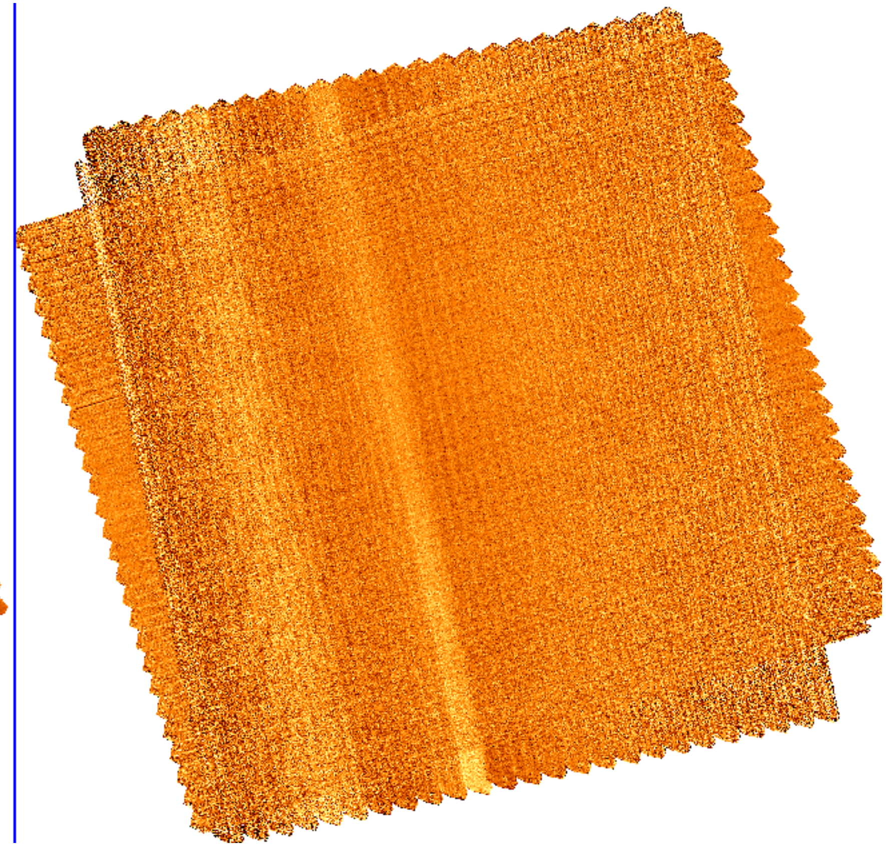


PGLS

PGLS

Check *delta_pgls_rebin* to evaluate distortion removal (verify that it contains only correlated noise). Increase the filter len if distortion still present.





L004 blue. *img_pgls* *delta_pgls_rebin*

WGLS – Weighted PGLS

TOP → Pre → Glitch → Drift → Noise → GLS → PGLS → WGLS

WGLS

```
=============== % 8: WGLS ============================================================
0    % wgls_mtype - 0/1/2 - mask type: 0 cross, 1 weight with image, 2 weigth with highpass image
3    % wgls_max_dev - positve real - threshold to declare a readout an artefact (lower = more detection)
3    % wgls_hflen - integer - half len of the filter (pixels)  (e.g. 3)
```

➢ Reduce noise increase due to PGLS

- Filter PGLS distortion estimate to amplify it.
- Detect distortion by comparing with a threshold.
- Compute WGLS image by subtracting detected PGLS distortion.

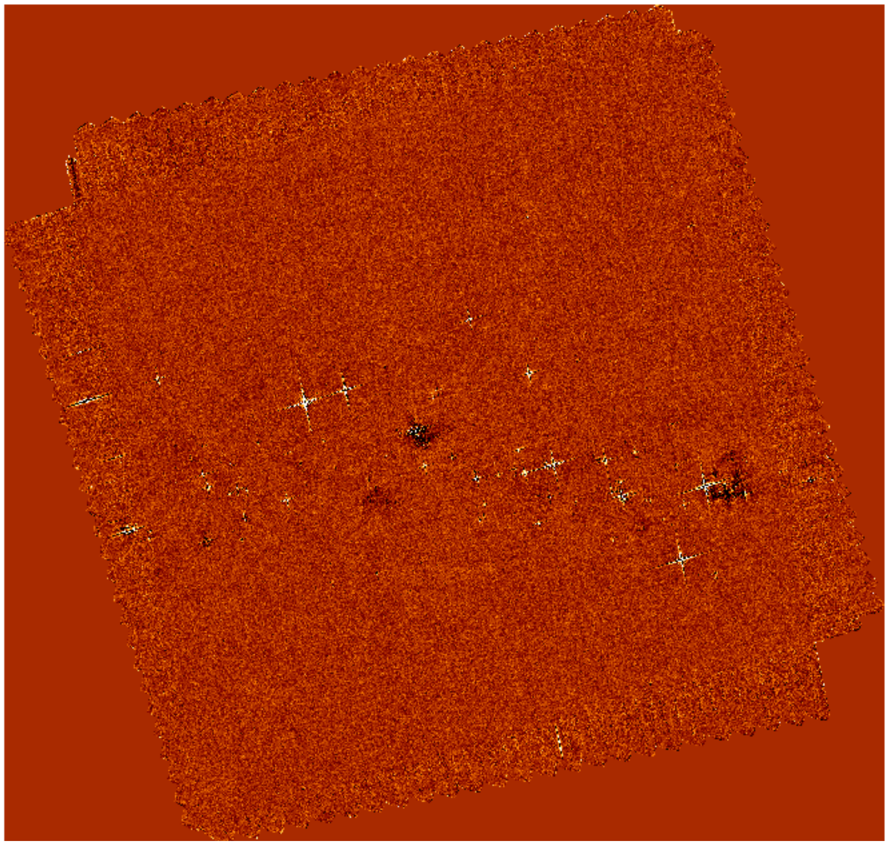
Three possible filters/masks.
The threshold cannot be set well by default. It needs trial and error.
WGLS is not needed if the PGLS noise increase was negligible.

Output: *img_wgls, flag_wgls, delta_wgls_rebin*

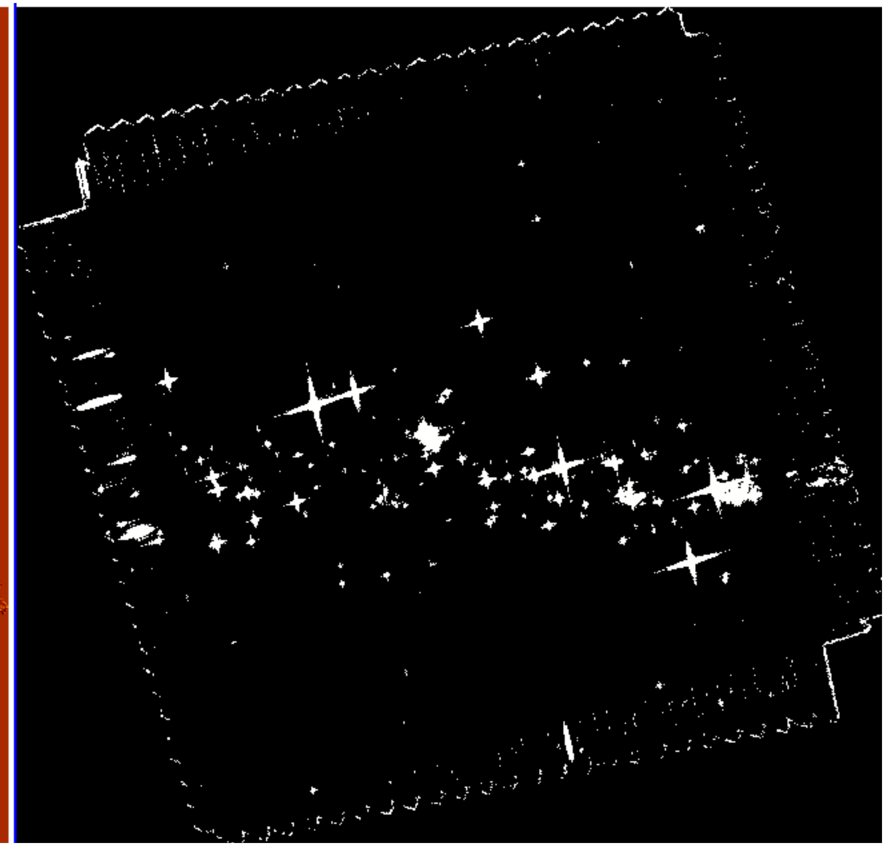
WGLS

Compare *delta_gls_pgls* with *flag_wgls* to evaluate the mask.

Here, mask type: 0. Distortion is amplified by convolution with cross filter.



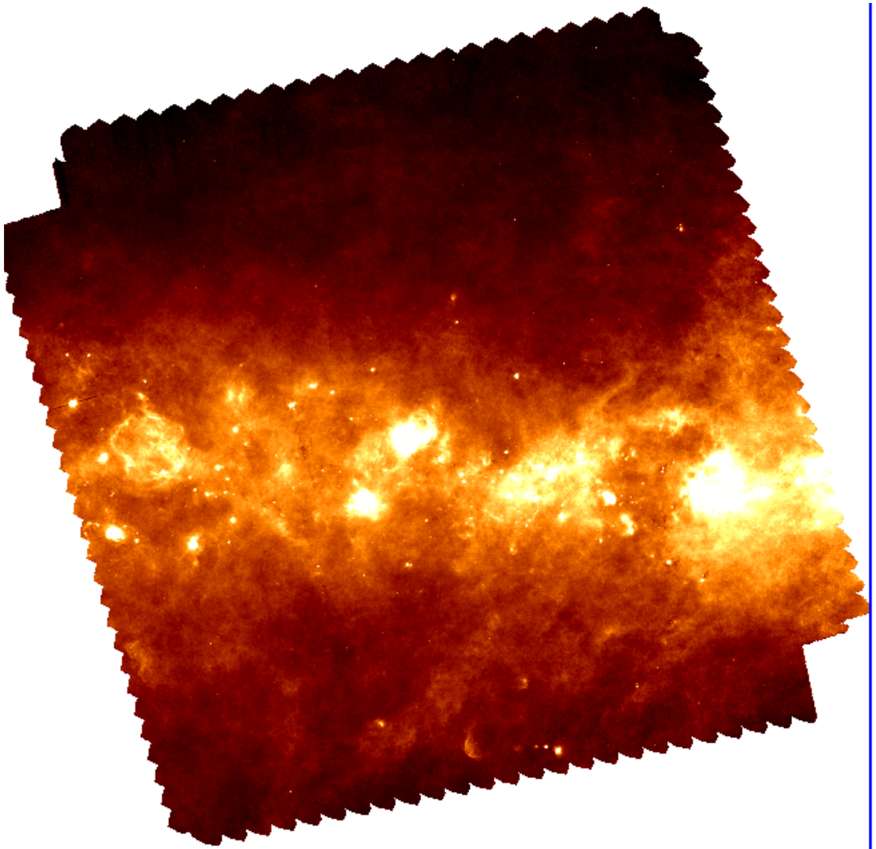
L004 blue. *delta_gls_pgls*

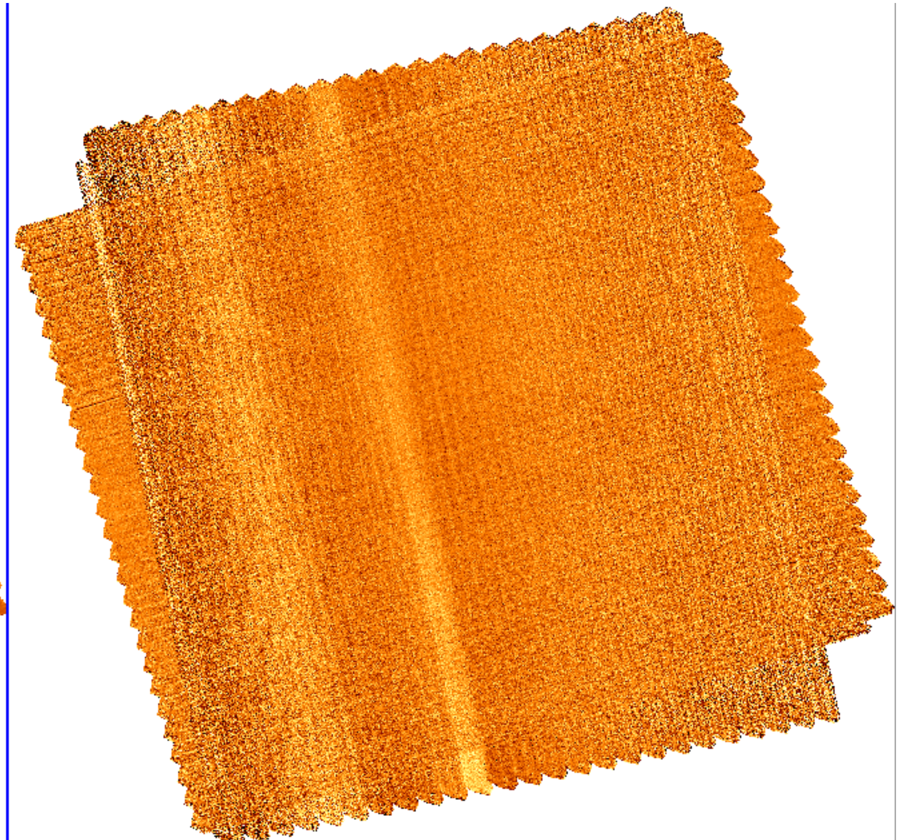


flag_wgls

WGLS

Use *delta_wgls_rebin* to check distortion: only correlated noise should be seen.



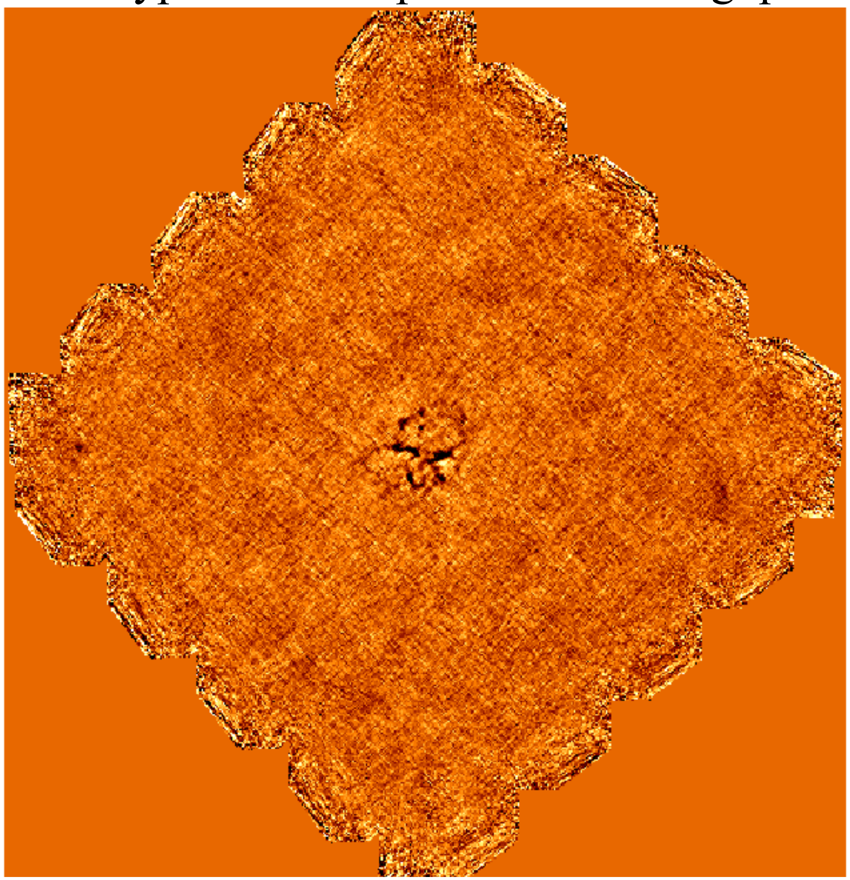


L004 blue. *img_wgls*

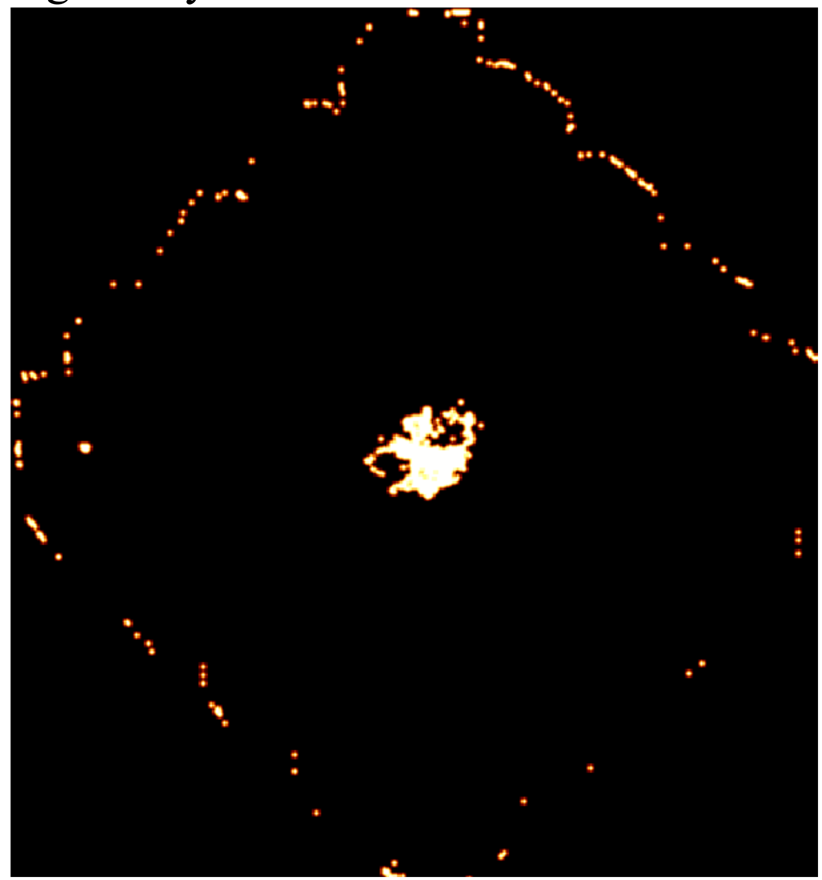
delta_wgls_pgls

WGLS

Here, mask type: 1. Distortion is amplified by mutliplication with the image.
Mask type 2 is multiplication with highpass image and yields similar masks.



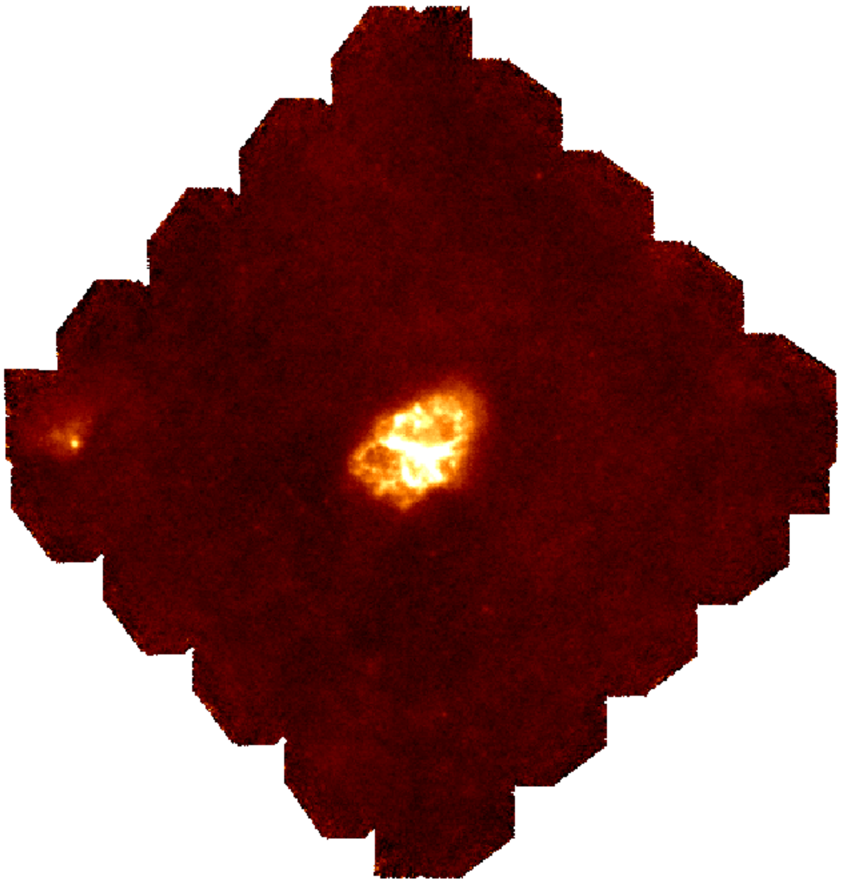
Crab red. *delta_gls_pgls*

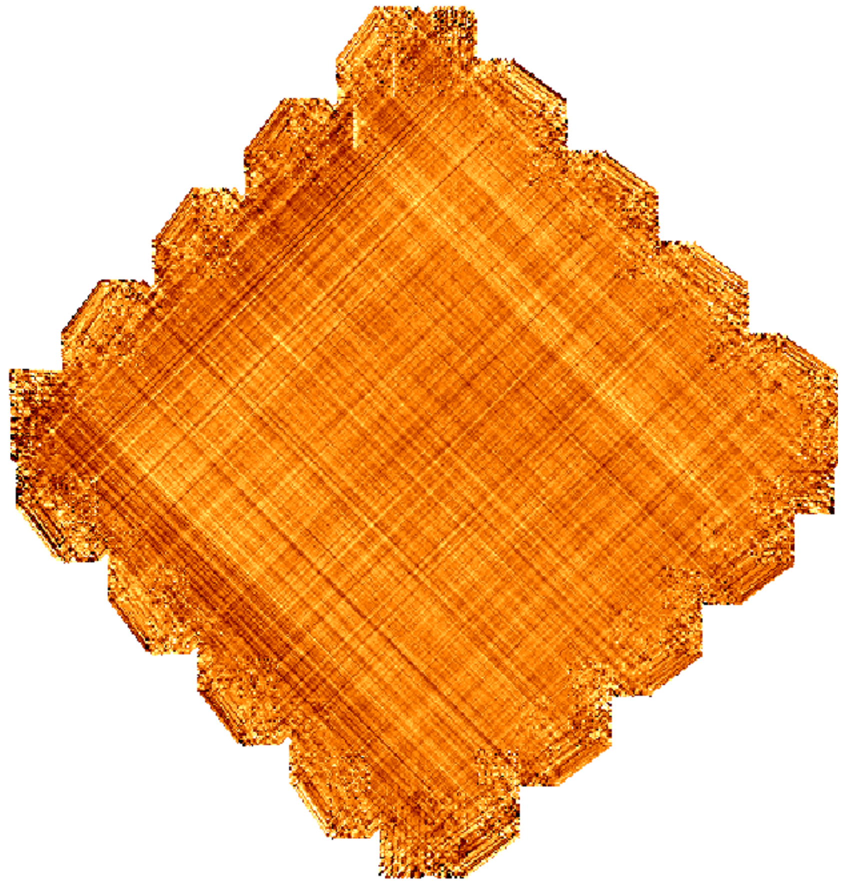


flag_wgls

WGLS

Use *delta_wgls_rebin* to check distortion: only correlated noise should be seen.





Crab red. *img_wgls* *delta_wgls_rebin*

Unihipe

Unihipe

Unihipe is an interface with HIPE and can be used to produce Unimap input data.

It is developed by ASDC and can be found at
http://herschel.asdc.asi.it/index.php?page=unimap.html

It works where HIPE is installed.

Unimap input is produced starting from Level 1 products .

The data can be downloaded or loaded from a locally saved file.

Level 1 products for PACS are essentially raw data. They are a perfect input for Unmap.

Level 1 products for SPIRE undergo some processing. Unimap performance could be (slightly) improved by extracting the data at a lower HIPE stage.

Unihipe

SPIRE Level 1 products are deglitched and highpass filtered (to remove bolometer response).

The HIPE deglitching is heavy. Easily flags more than 3% of the data.
If flagged data are removed void (NaN) pixels may appear.
If linearly reconstructed readouts are used some distortion may be visible.

It would be better to suppres HIPE deglitching.
But if glitches are passed to the HIPE highpass filter spikes appear in the timelines.

Optimal solution is to move the HIPE highpass filter in the GLS mapper.
Also the noise spectrum would be better conditioned (follow 1/f plus white).

Conclusion

Conclusion

Unimap is a robust, simple and efficient mapper.
Code and documentation are now stable (well, almost) and available in the home page.

Future work will follow these research lines:

- Dedrifting with non polynomial models
- Detect and remove RPE
- Compensate for coaddition in PACS
- Improve input data format for SPIRE
- Improve data model taking the PSF into account

The workshop was an excelent opportunity to gain expertise and a strong push to improve Unimap.   Sincere thanks to Babar, Roberta and all the mappers/experts.