The Unimap map-maker: status and developments

Lorenzo Piazzo

DIET dept. - University of Rome 'La Sapienza'

Outline

Part I – Unimap

Overview
Basic data model
Least Squares Estimation
Noise
Reduction strategy
Full data model and optimal map
Drift removal
GLS synthesis
GLS distortion
Distortion removal

Part II – Developments

Software
PSF equalisation
Pixellisation noise
Synchronisation / Pointing error
SCUBA2

Part I

Unimap

Overview

Unimap (infocom.uniroma1.it/unimap) is a Map maker for PACS and SPIRE, developed by the University of Rome, in cooperation with the ASI, IAPS and ESAC.

Source code is Matlab. Distributed under an open source license.

Compiled version (not requiring Matlab) distributed for Linux and Mac.

First release 2012. Release 5.5.0 has been stable since 2013.

Quality images with modest hardware requirements.

Almost automatic: good images with default parameters, but tuning of a few is needed for best results.

The signal processing is presented in the following paper

[1] L. Piazzo, L. Calzoletti, F. Faustini, M. Pestalozzi, S. Pezzuto, D. Elia, A. di Giorgio and S. Molinari: 'Unimap: a Generalised Least Squares Map Maker for Herschel Data', MNRAS, 447, pp. 1471-1483, 2015

Please cite the paper(s) if you use images produced with Unimap!!

Overview

The core of Unimap is a Generalised Least Squares (GLS) image synthesis method.

The GLS method can be tracked back to 1935:

A. C. Aitken, "On Least Squares and Linear Combinations of Observations", Proceedings of the Royal Society of Edinburgh, vol. 55, pp. 42-48, 1935.

It has been used for astrophysical imaging since the nineties:

M. A. Janssen, S. Gulkis, "Mapping the Sky with the COBE Differential Microwave Radiometers", In "The Infrared and Submillimetre Sky after COBE", NATO ASI Series Volume 359, pp. 391-408, 1992.

Theoretical background is well developed, e.g.

M. Tegmark, "CMB mapping experiments: A designer's guide", Physical Review D, Vol. 56, no. 8, pp. 4514-4529, 1997.

Efficent implementations exists, e.g.

E. L. Wright, G. Hinshaw, and C. L. Bennett, "Producing Megapixel Cosmic Microwave Background Maps from Differential Radiometer Data", The Astrophysical Journal Letters, Vol. 458 No. 2, 1996.

Used by several Herschel map-makers: Madmap, Sanepic, Tamasis, Romagal.

Overview

While the GLS is well known, its application to Herschel data is not trvial.

- big data set: an efficent implementation is needed.
- glitches and jumps due to cosmic rays require specific pre-processing

Moreover there are two **key problems**

- Data are affected by **spatially correlated** noise (noise of different bolometers is correlated). This greatly complicates the GLS implementation.

- Classical GLS introduces **distortion** (signal dependent error) which lowers the image quality.

Overview

Unimap was the first GLS mapper to succesfully address both problems.

• Spatial correlation is removed by means of an Alternating Least Squares (ALS) approach:

[2] L. Piazzo, P. Panuzzo, M. Pestalozzi: "Drift removal by means of alternating least squares with application to Herschel data", Signal Processing, vol. 108, pp. 430-439, 2015.

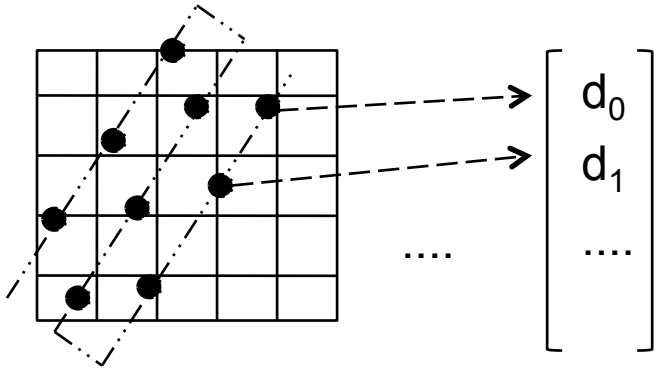
• Distortion is removed by means of the Post-processing for GLS (PGLS) algorithm.

[3] L. Piazzo, D. Ikhenaode, P. Natoli, M. Pestalozzi , F. Piacentini and A. Traficante: "Artifact removal for GLS map makers by means of post-processing", IEEE Trans. on Image Processing, Vol. 21, pp. 3687-3696, 2012.

Basic data model

Consider an image of *M* pixels, scanned with a bolometer. The bolo output is sampled to produce a **timeline** of *D>>M* readouts, represented by a *Dx1* vector *d* (**data**).

Assuming an ideal bolometer, each readout gives the value of the pixel towards which the bolometer was pointed at the sampling time.



By organising the image pixels into an *Mx1* vector *m* (**map**), this process can be written in matricial form

$$d = P\,m$$

P is the *DxM* **pointing** matrix. P_{hk} = 1 if the h-th readout was taken in the k-th pixel. Otherwise is zero. It is a sparse, binary matrix.

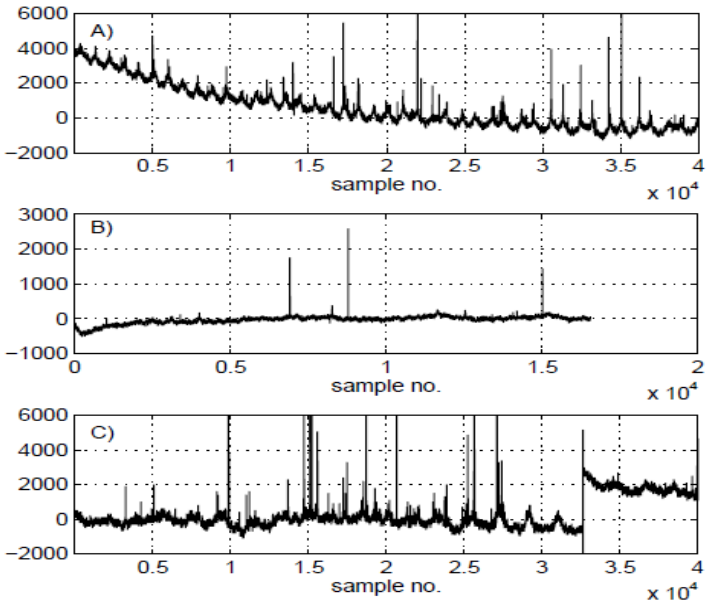
When there are many bolos, *d* and *P* are obtained by stacking the data and pointing of the various bolos.

Basic data model

The timelines are affected by several disturbances. Most important are

- Thermal and electronic noise
- Glitches and jumps
- Random offset
- Saturation

There are several other effects:coaddition, RPE, quantization noise, interference from solar panel...



Glitches, jumps and saturations are detected in the **pre-processing** (see [1]) and the corresponding readouts excluded. Typically less than 0.5% of the readouts. Minimal impact on performance.

Relative offsets are corrected by GLS, but an absolute offset will affect the map.

The noise and the offset are accounted for by adding a random vector to the model:

$$d = Pm + n$$

Least Squares Image Estimation

Given the data model

$$d = Pm + n$$

the **map-making** problem is that of estimating *m* knowing *d, P* and the statistics of the noise.

An important estimate is the one obtained solving the over-determined linear system $Pm = d$ in the Least Squares (LS) sense. The LS estimate is

$$\overline{m} = (P^T P)^{-1} P^T d$$

where $(P^T P)^{-1} P$ is the pseudo-inverse of *P*. The estimate is also called **naive map** or **rebinned map**.

Simple to compute: just average the readouts falling into each pixel.

If the noise is white and Gaussian it is the Maximum Likelihood (ML) estimate.

However PACS and SPIRE noise is correlated and not white: LS does not work.

Least Squares Image Estimation

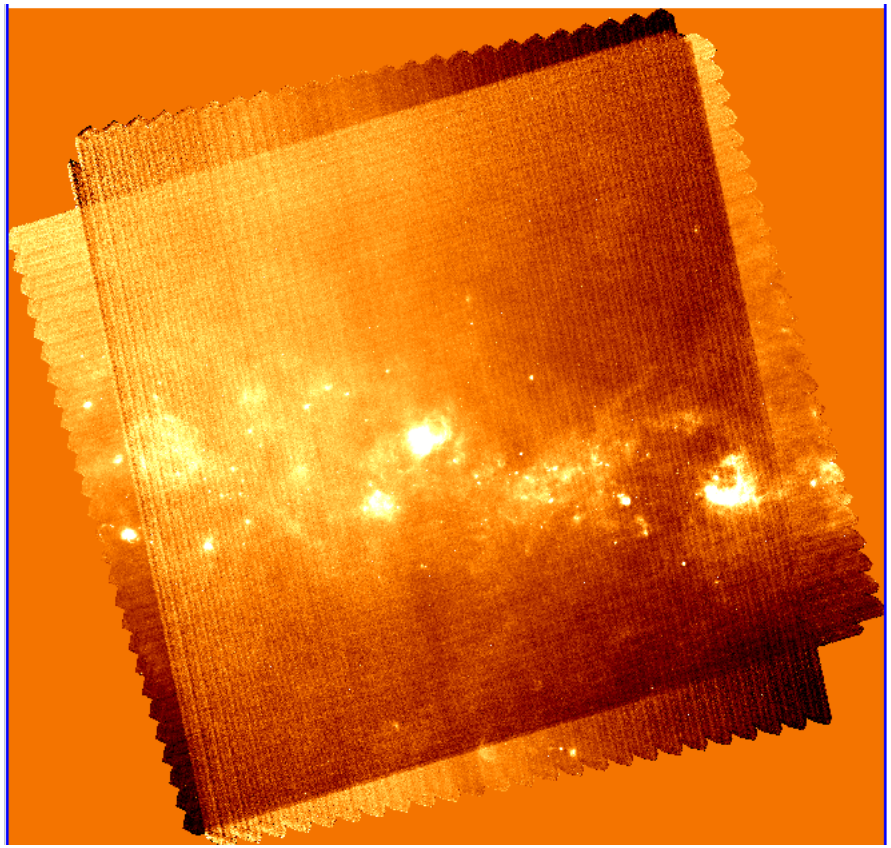


Figure shows the LS (naive) map after pre-processing (glitch and jump detection) for a Galactic tile centered on 4 degrees (L004).

Note the effects of correlated noise:

- gradient on the edges
- non flat background
- stripes

Least Squares Image Estimation

In the presence of correlated noise, a better estimate is the Generalised Least Squares (GLS) estimate, given by

$$\overline{m} = (P^T N^{-1} P)^{-1} P^T N^{-1} \, d$$

where $N = E\{ nn^T \}$ is the covariance matrix of the noise.

Has the minimum variance (MV) among all linear estimates.

It is the ML estimate when noise is Gaussian.

Cannot be computed directly: matrices are too big to be inverted.

The GLS is a better estimate than LS, but we need an efficent, iterative way to compute it.

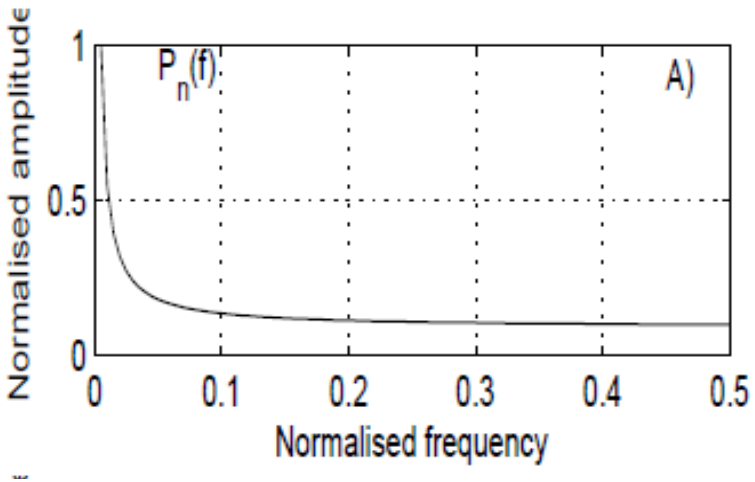
We need to study the noise covariance matrix.

Noise

Each timeline is affected by stationary **1/f** noise. The noise spectrum can be written as

$$P_n(f) = [\,(f_0 / f)^{a} + 1\,]\,N_0$$

and is the sum of of a 1/f part, $N_0(f_0 / f)^{a}$, plus a flat, white part, N_0.



Since the spectrum is not flat, the noise is **temporally correlated**: successive noise samples are statistically dependent.

It can be oserved that the noise is **spatially correlated** too: the noise of adiacent bolometers (same sub-array) is statistically dependent.

KEY PROBLEM: Spatial correlation prevents the use of efficent GLS solution techniques. We look for a solution and improve the noise model.

Noise

We separate the noise into two components.

Thermal noise (**drift)**

- due to slow temperature variations of the focal plane
- low frequency noise, contributing to the 1/f part
- responsible for spatial correlation

Electronic noise (**noise**)

- due the readout electronics of each bolometer
- contributes to the 1/f part and to the white part
- not spatially correlated

Reduction strategy

Using the noise decomposition we write the data model as

$$d = Pm + y + n$$

where *y* is the **drift** vector and *n* is the (electronic) **noise** vector.

The drift is a low frequency signal, slowly changing. It is well modelled as a smooth curve, depending on a few parameters. We can use a two steps reduction strategy:

• **Step 1** (**drift removal**) produce an estimate of the drift, denoted by $\bar{y}$, and subtract it from the data vector, to produce an updated data vector given by

$$\tilde{d} = d - \bar{y}$$

which is, ideally, drift free.

• **Step 2** (**GLS synthesis**) estimate the map from the updated data vector $\tilde{d}$ using GLS.

This approach is followed by most GLS mappers (Tamasis, Madmap, Romagal).

Full data model

In order to implement the drift removal, we need to better specify the drift.

The drift is well modelled as a set low order polynoimials, one for each subarray.

$$\begin{pmatrix} 1 & 1 & 1^2 & 1^3 & \dots & 1^{N_a} \\ 1 & 2 & 2^2 & 2^3 & \dots & 2^{N_a} \\ 1 & 3 & 3^2 & 3^3 & \dots & 3^{N_a} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & N_r & N_r^2 & N_r^3 & \dots & N_r^{N_a} \end{pmatrix}$$

A polynomial can be written as the product of a Vandermonde matrix (see figure) by a coefficent vector.

Using this fact, we can write the drift as $y=Xa$ and the data model becomes

$$d = Pm + Xa + n$$

where X is the **drift** matrix, obtained by properly stacking a Vandermonde matrix for each sub-array, and a is an unknown **coefficent** vector.

In the next slides we see that the latter model is important both to implement the drift removal and to better understand the theoretical framework.

Optimal map

Given the full model $d = Pm + Xa + n$, we can extend the LS and GLS approaches and **jointly** estimate the unknown map m and coefficents a.

Introducing $z = \begin{bmatrix} m \\ a \end{bmatrix}$ and $A = \begin{bmatrix} P & X \end{bmatrix}$ we can rewrite the data as

$$d = Az + n$$

and the problem becomes that of estimating z. The joint LS (JLS) and joint GLS (JGLS) estimates are

$$\bar{z} = \begin{bmatrix} \overline{m} \\ \overline{a} \end{bmatrix} = (A^T A)^{-1} A^T d \qquad \bar{z} = \begin{bmatrix} \overline{m} \\ \overline{a} \end{bmatrix} = (A^T N^{-1} A)^{-1} A^T N^{-1} d$$

The JGLS is optimal in the sense that it gives the MV and, in Gaussian noise, the ML estimate of the map and coefficents.

Drift removal

In principle, since the noise is no more spatially correlated, the JGLS estimate can be computed using an iterative approach.

However, for big data, convergence is slow and there are numerical stability issues.

Therefore, we stick to the two steps approach, but the JGLS remains an important theoretical bound.

The JLS estimte is simpler to compute and is used by Unimap to carry out the first reduction step, i.e. the drift removal.

Specifically, Unimap computes the JLS estimate and extracts the coefficent vector. Next it estimates the drift as

$$\bar{y} = X\bar{a}$$

and produces the updated data vector as

$$\tilde{d} = d - \bar{y}$$

Drift removal

Drift removal was analysed in [2]. It was shown that

• The updated data vector is given by

$$\tilde{d} = Pm + c + \tilde{n}$$

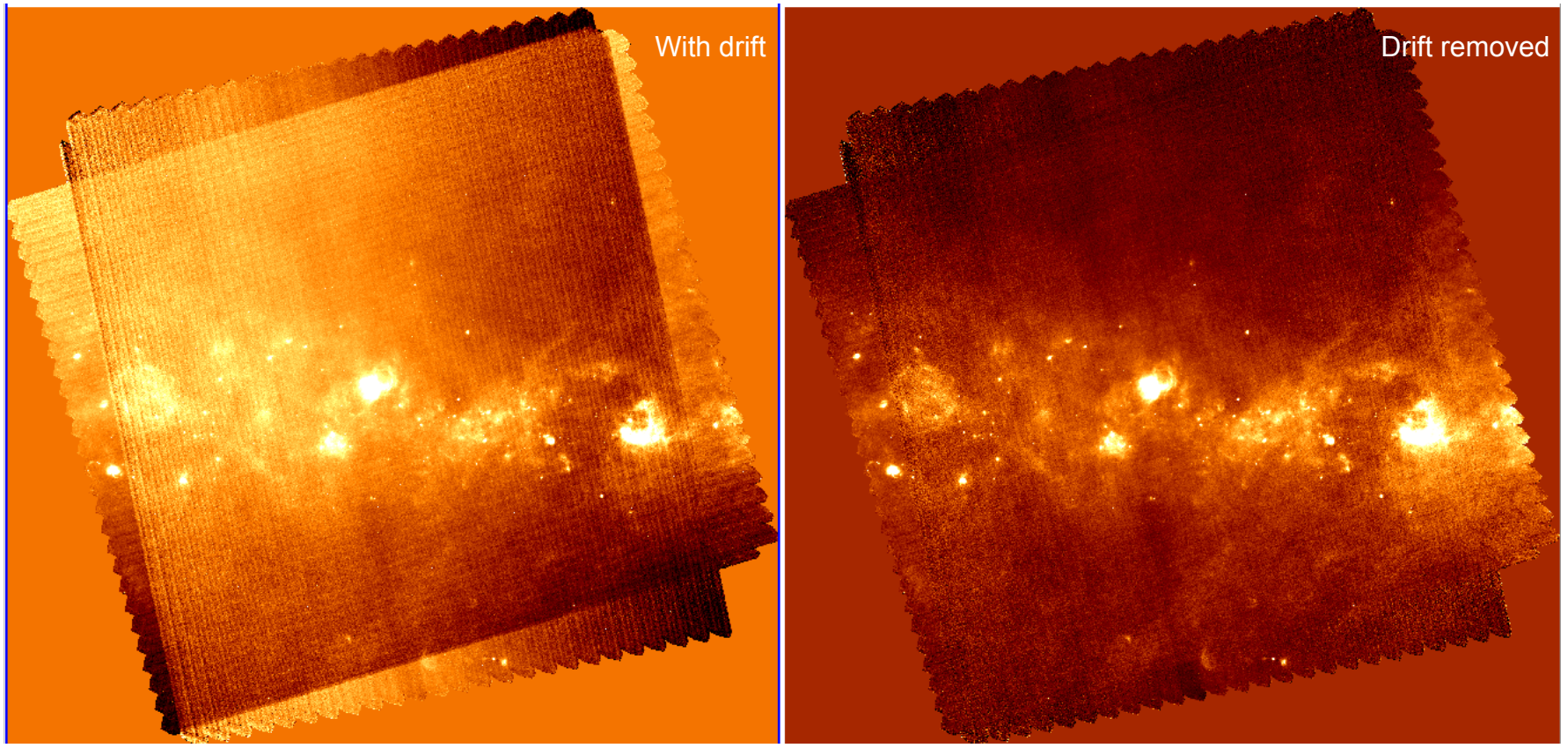
where *c* is a constant vector and $\tilde{n}$ is a zero mean noise vector. Therfore, within the limits of the polynomial model, the drift is removed except for a constant (irrelevant, map is in any case affected by an offset) and the signal is preserved.

• The map produced by the two steps approach is close to the JGLS estimate (was verified by means of simulations on reduced data sets). Therefore the two steps approach incurs a negligible performance loss and yelds a near-optimal map.

• There is an efficent way to compute the drift estimate, based on an Alternating Least Squares (ALS) approach. Similar algorithms are used by Tamasis and the SPIRE destriper.

Drift removal

We compare the naive (LS) maps before and after the drift removal. Background is flattened and striping attenuated. Residual striping is there, due to the1/f part of the electronic noise.



GLS synthesis

We now consider the second step. Starting from the updated data vector

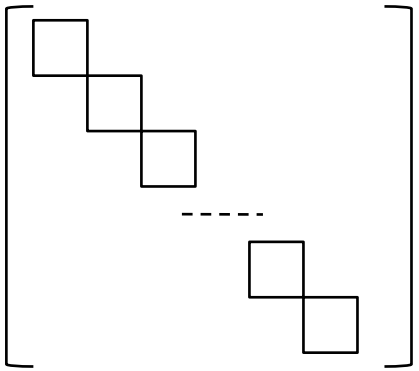
$$\tilde{d} = Pm + c + \tilde{n}$$

dropping the tilde and neglecting the irrelevant constant vector we have that, after the drift removal, the data model is

$$d = Pm + n$$

where the noise *n* is **no more spatially correlated**.

The GLS estimte is $\overline{m} = (P^T N^{-1} P)^{-1} P^T N^{-1} d$



Since the noise is stationary and not spatially correlated, the covariance matrix is block Toeplitz (see figure) with a block per timeline and is dictated by the noise spectrum.

Unimap estimates the spectrum from the data set, see[1].

When *N* is block Toeplitz, there exists an efficent and well known procedure to obtain the GLS estimate.

GLS synthesis

Starting from the expression of the GLS estimate $\overline{m} = (P^T N^{-1} P)^{-1} P^T N^{-1}\, d$ we can write

$$(P^T N^{-1} P)\, \overline{m} = P^T N^{-1}\, d$$

and introducing $A = P^T N^{-1} P$ and $b = P^T N^{-1} d$ we obtain

$$A\, \overline{m} = b$$

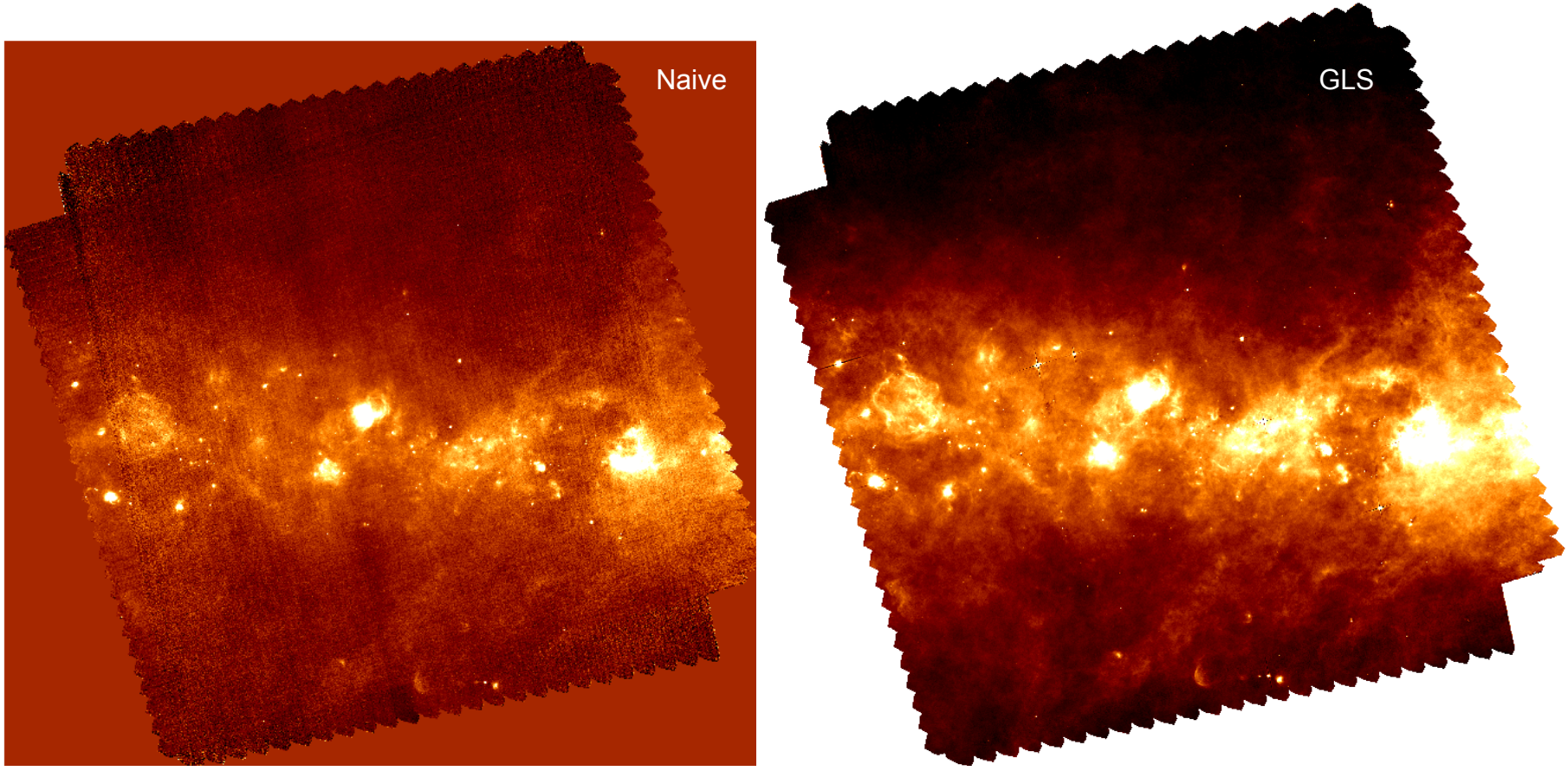
which is a linear system in the unknown vector $\overline{m}$.

The system can be solved using an iterative procedure. Typically the Parallel Conjugate Gradient (PCG) is used, which requires to compute the multiplication of a vector by A, i.e. by P, P^T and N^{-1}, at each iteration.

Multiplication by P, P^T is not difficult. Since N is block Toeplitz, multiplication by N^{-1} can be efficently realised by means of a linear filter (see e.g. [1]).

GLS synthesis

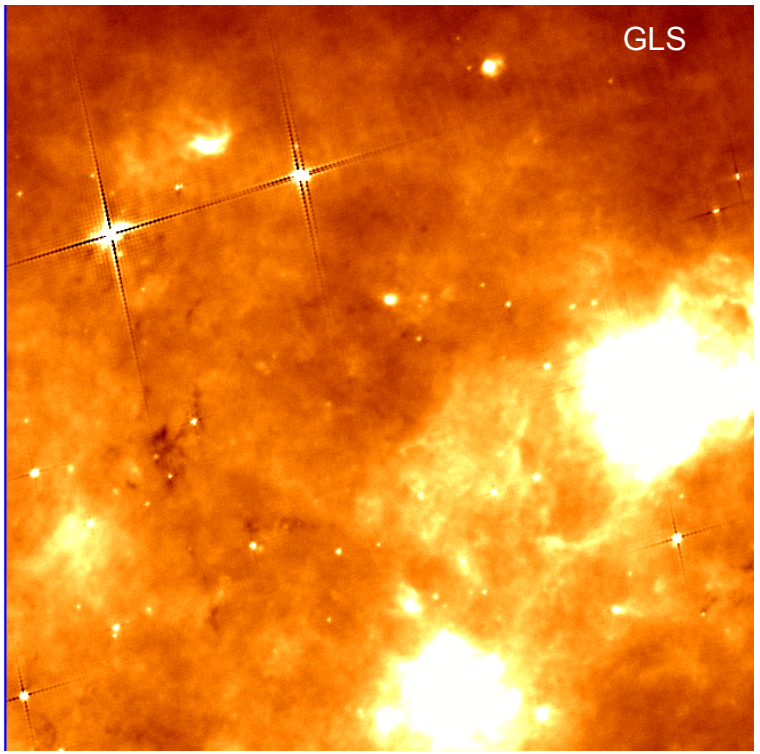
We compare the LS (naive) map after drift removal and the GLS map computed by means of the PCG. We see that the residual striping is eliminated.



GLS distortion

The GLS is effective against the 1/f noise but it has a serious drwback: it introduces distortion (signal dependent error). **KEY PROBLEM !**

Distortion is due to the approximation of the data model, which does not fully reflect the physical observation process. More on this in the second part of the talk.



To evaluate the presence of distortion an important tool is the difference of the GLS and naive maps.

Naive contains signal plus 1/f noise.

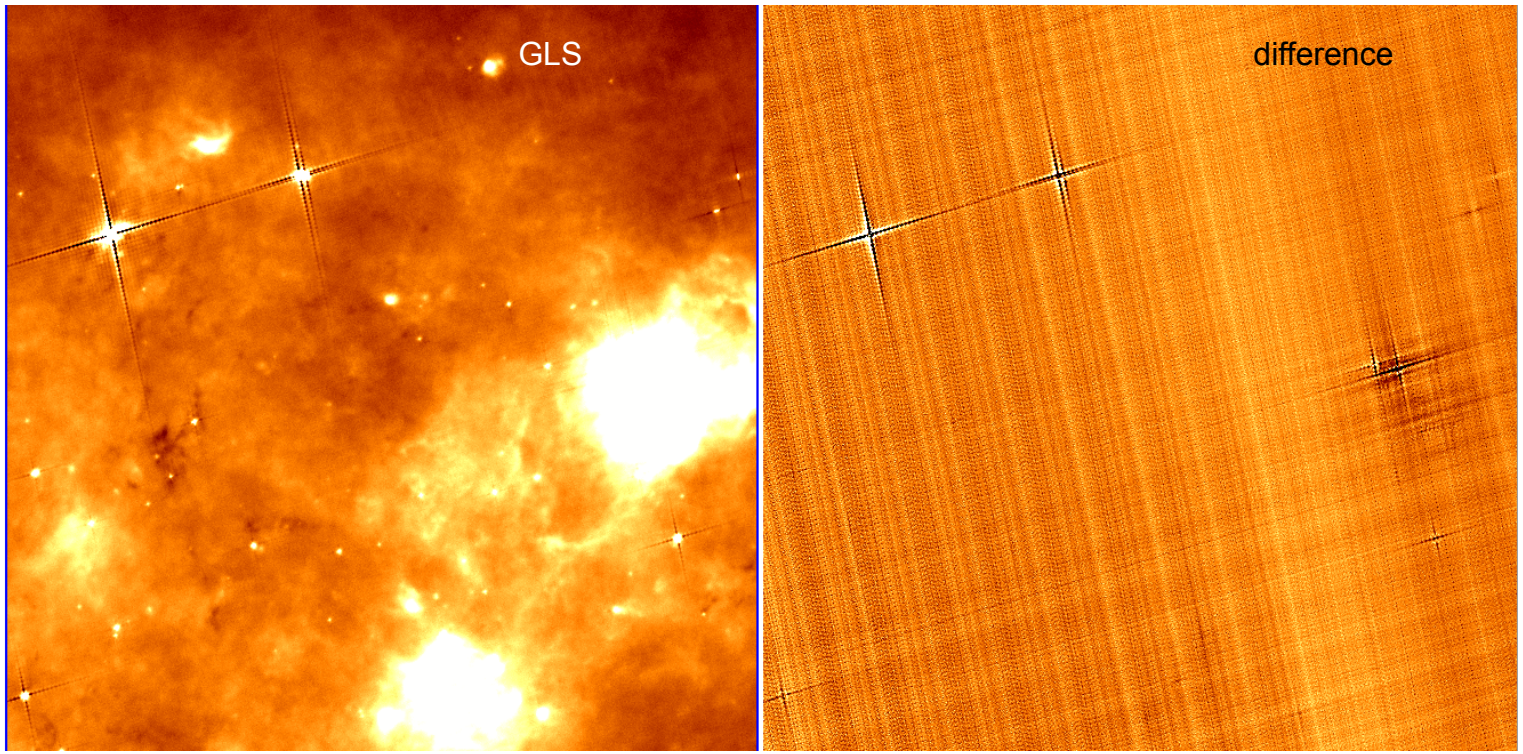
GLS contains signal plus distortion.

Difference contains distortion plus 1/f noise (with reversed polarity).

GLS distortion

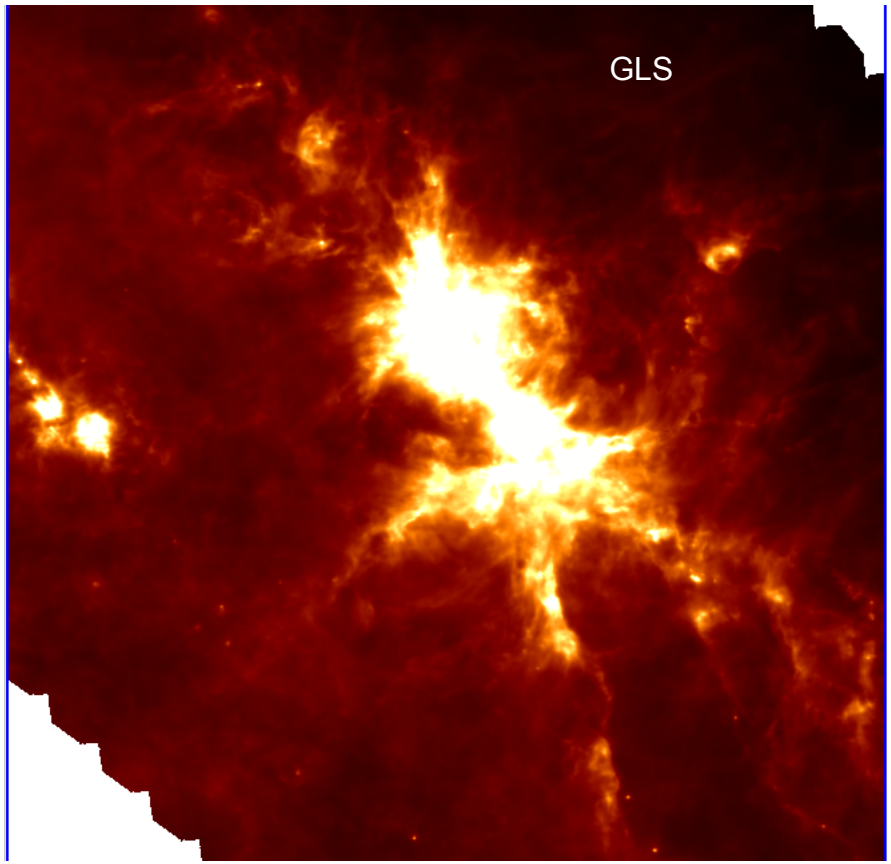
GLS and difference maps. Cross like artifacts placed on bright stars are clearly visible.

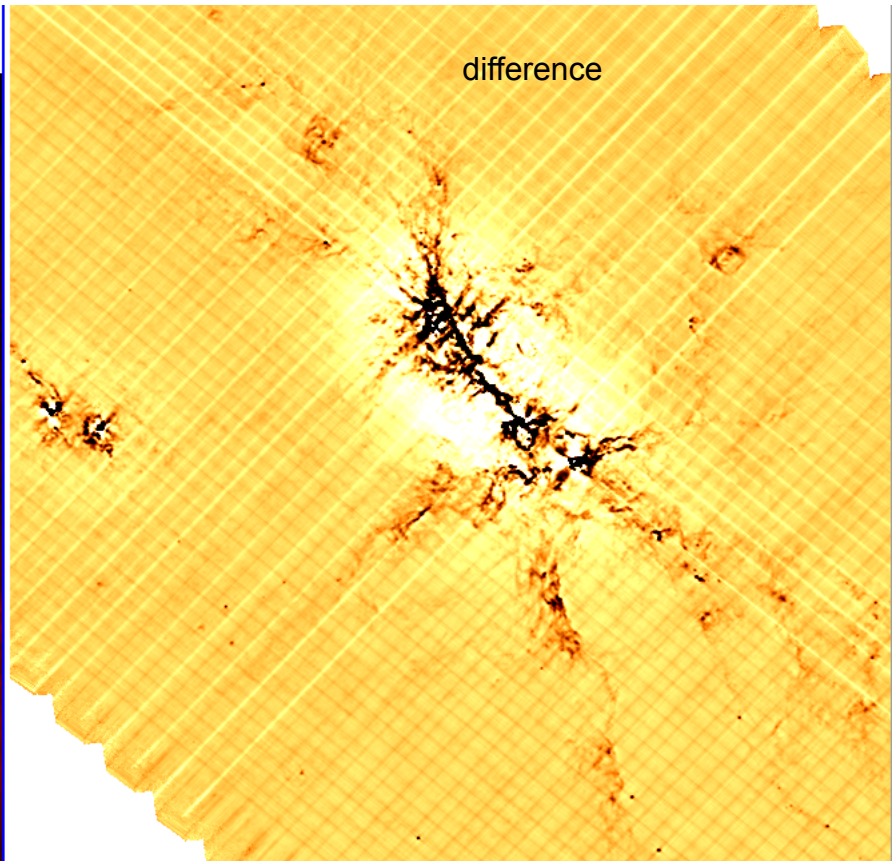
Additional distortion becomes visible.



GLS distortion

Distortion may also be diffused. Difficult to see in the GLS map but clearly revealed by the difference map.





Distortion removal

In [3] the Post-Processing for GLS (PGLS) algorithm was introduced to tackle the distortion problem.

The PGLS is an iterative algorithm producing an estimate of the GLS distortion. The PGLS map is obtained by subtracting the estimated distortion from the GLS map.

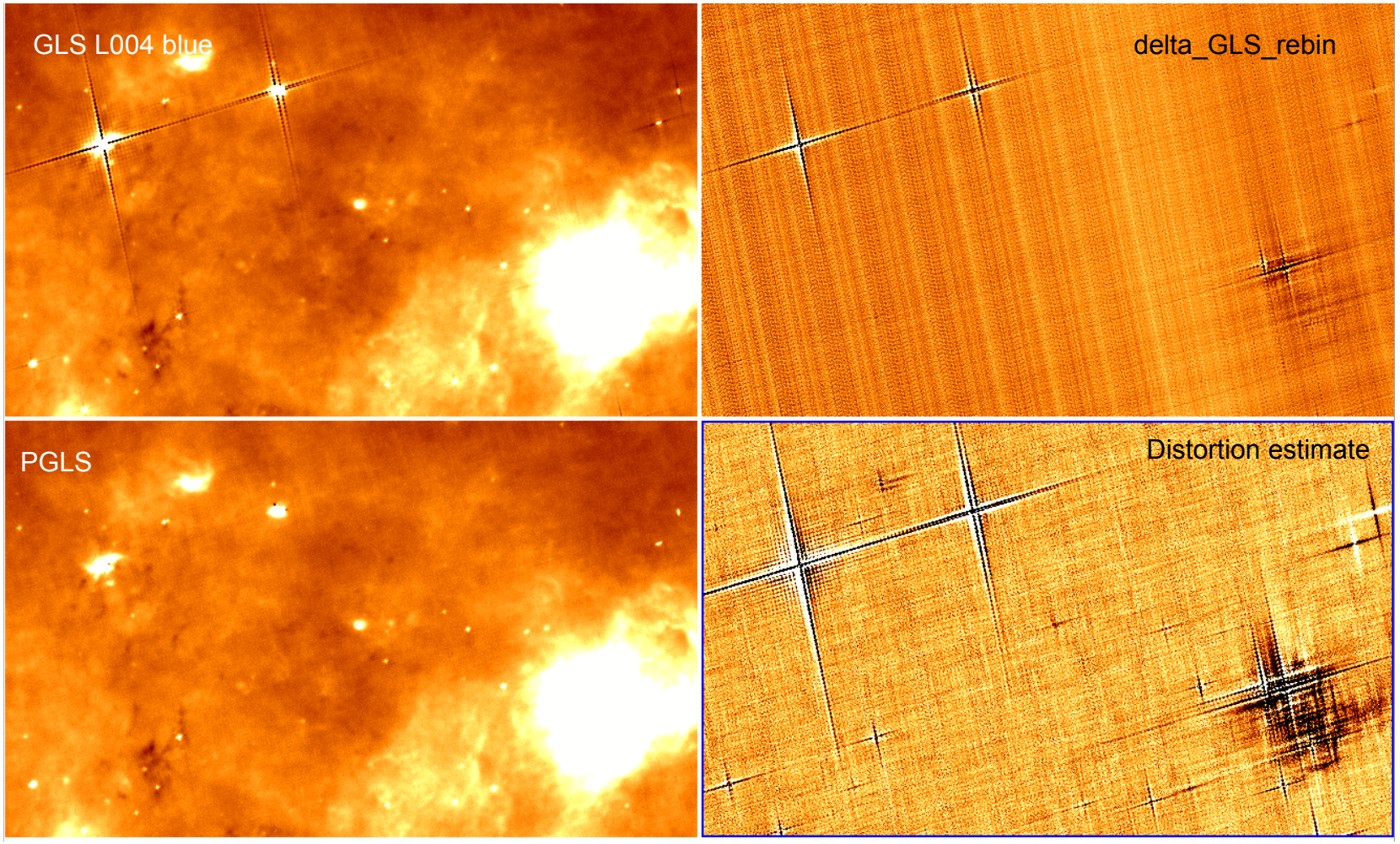
PGLS effectively removes the distortion and can be efficently implemented.

The main drawback is that the background noise is increased.

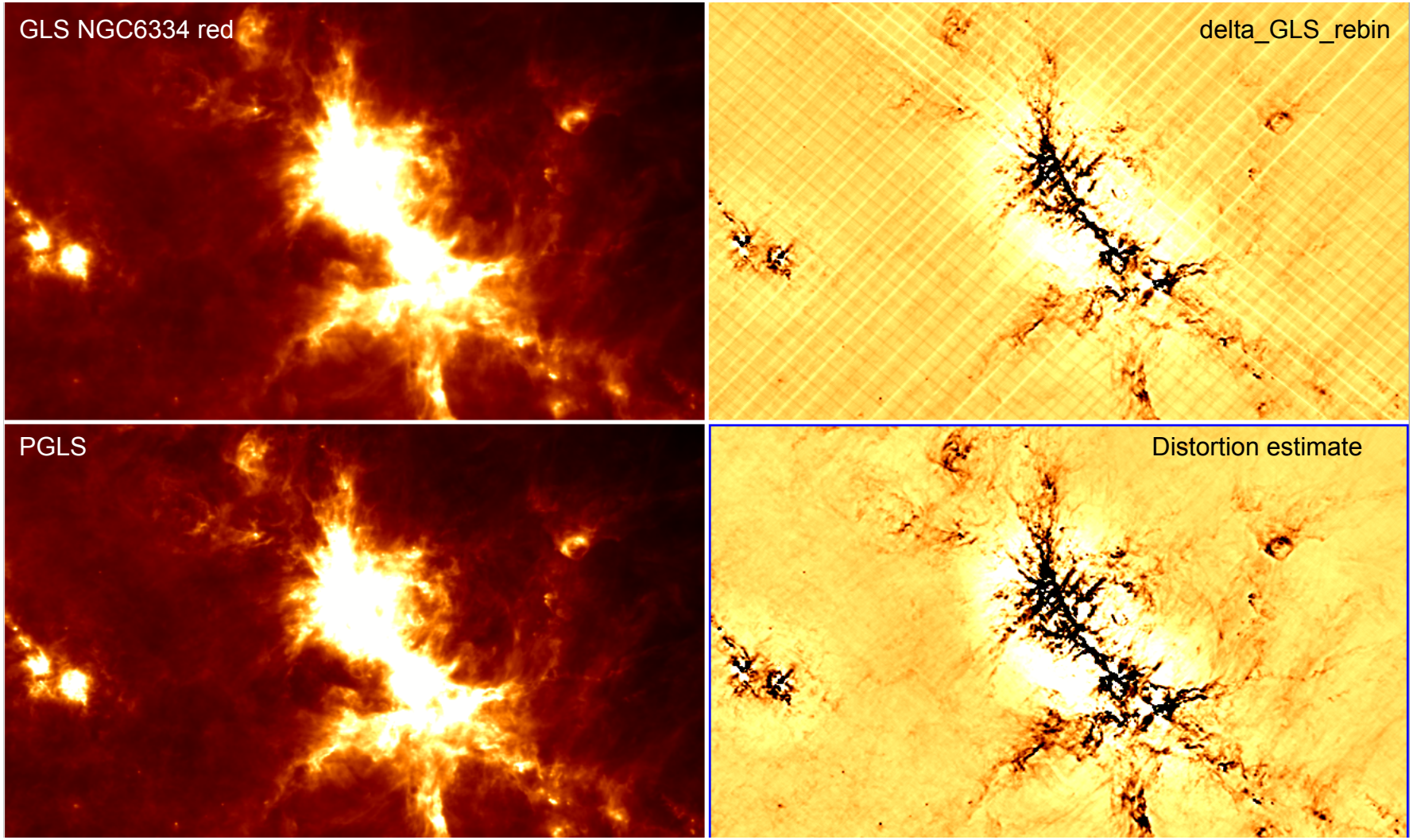
A minor drawback is that there is a non-linear step (median filtering) which makes the theoretical analysis difficult. Performance was evaluated by means of simulations.

The PGLS algorithm has been implemented in Madmap and Romagal too.

Distortion removal



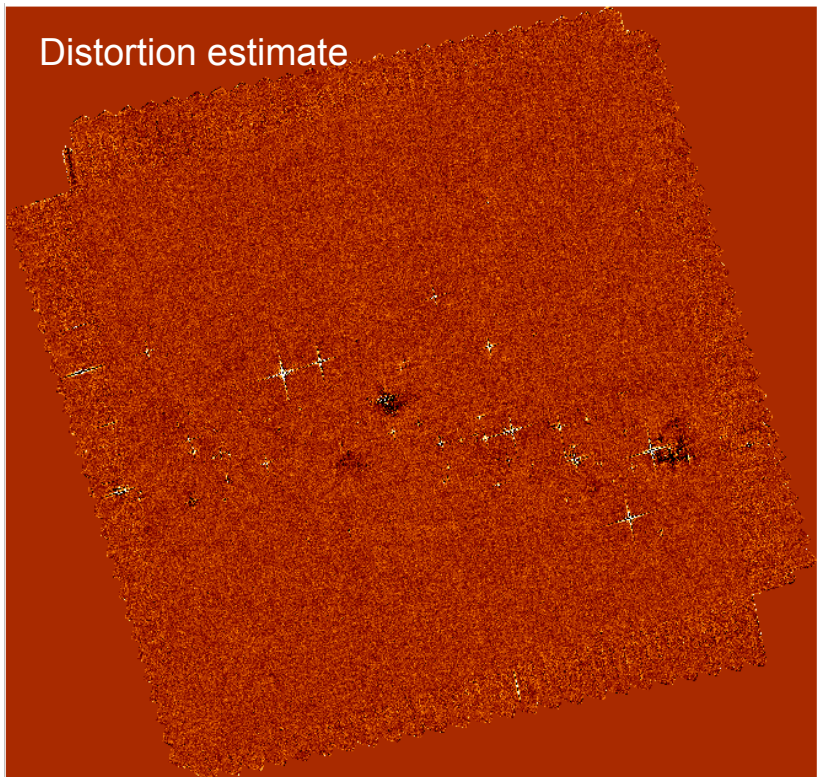
Distortion removal



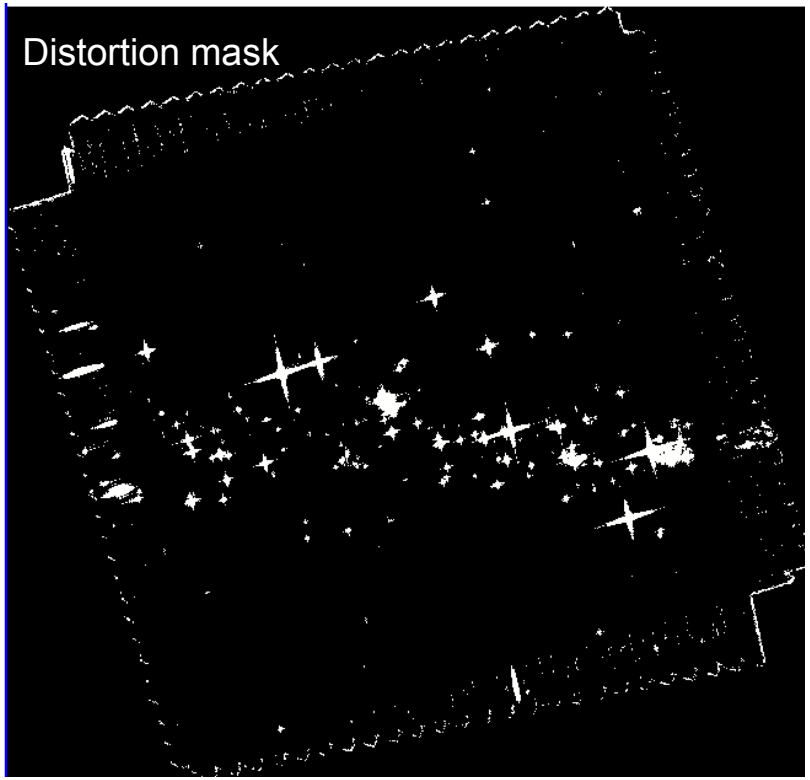
Distortion removal

Final step is Weighted PGLS (WGLS). Reduces the noise injected by PGLS.

A distortion mask is produced and the distortion is subtracted only where it has been detected, avoiding the noise increase in the pixels not affected by distortion.



delta_gls_pgls (distortion estimate)



flag_wgls (distortion mask)

Part II

Developments

Software

Code cleaning and optimisation

The source code is decently written but could be improved and cleaned.

Recently the GLS was optimised for speed (rels 553 and 554, maps identical to 550) and a reduction of 50% of the processing time was obtained. This is a 25% reduction of the overall processing time.

An additional reduction of the overall processing time (say 20%) could be obtained by optimising the rest of the code.

Parameter settings

Automatic setting of some parameters could be investigated.

Documentation

A User's Manual exists, explaining how to use the program.

A Reference Manual, explaining how the software is written, is lacking and would be useful to fully exploit the Open Source release.

Software

Parallelisation

Current version is not parallel, but all the processing steps (preprocessing, drift removal, GLS and PGLS) could be parallelised.

Implementing a parallel code should appreciably reduce the computation time.

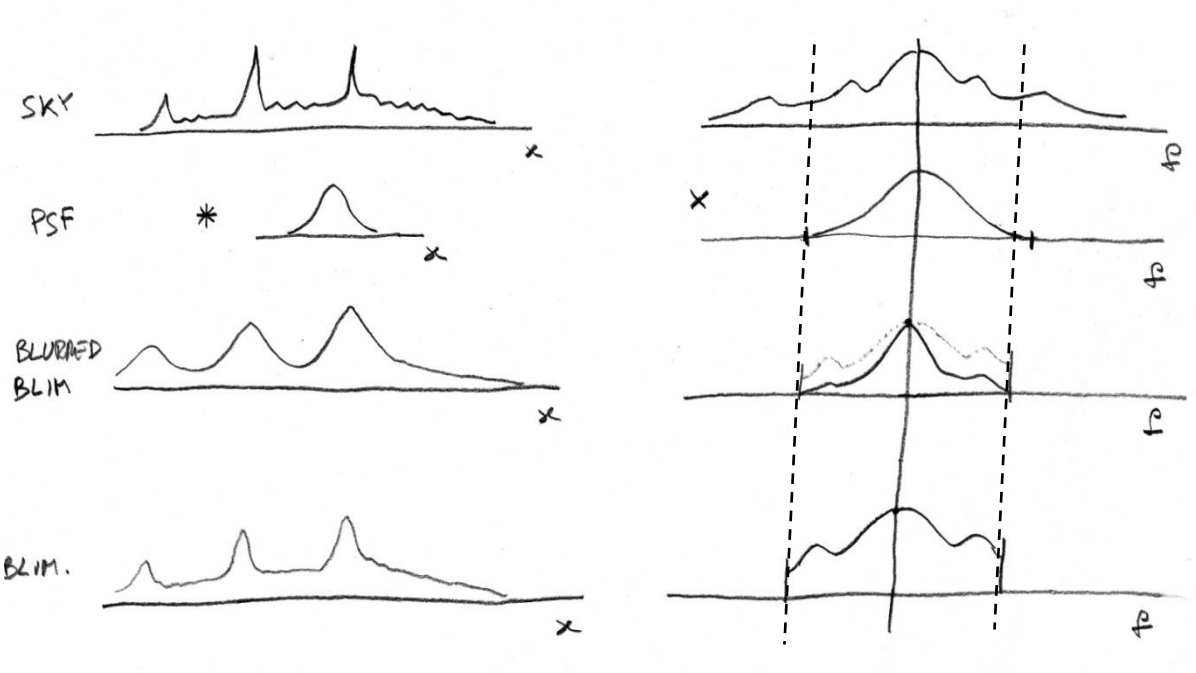
Actual reduction depends on the number of processors and on the overhead due to parallelisation (e.g. processors communications).

In the next slides we illustrate several planned signal processing improvements.

As a general comment, we note that the map quality is already high, so only percentual improvements should be expected.

PSF equalisation

The idea is quite simple: consider the observation process in both the space and frequency domains (using 1D for ease of visualisation and neglecting noise).



The sky is convolved/multiplied with the PSF (telescope + bolometer footprint) to produce a **band-limited** and **distorted** sky, which is sampled and is our current estimation goal.

By dividing (in frequency) by the PSF (where different from zero) we can equalise the PSF and obtain a band-limited but not distorted sky.

This is a better estimation goal, yielding improved resolution..

PSF equalisation

In practice the divison by the PSF cannot be done due to the presence of noise and more sophisticated procedures are required.

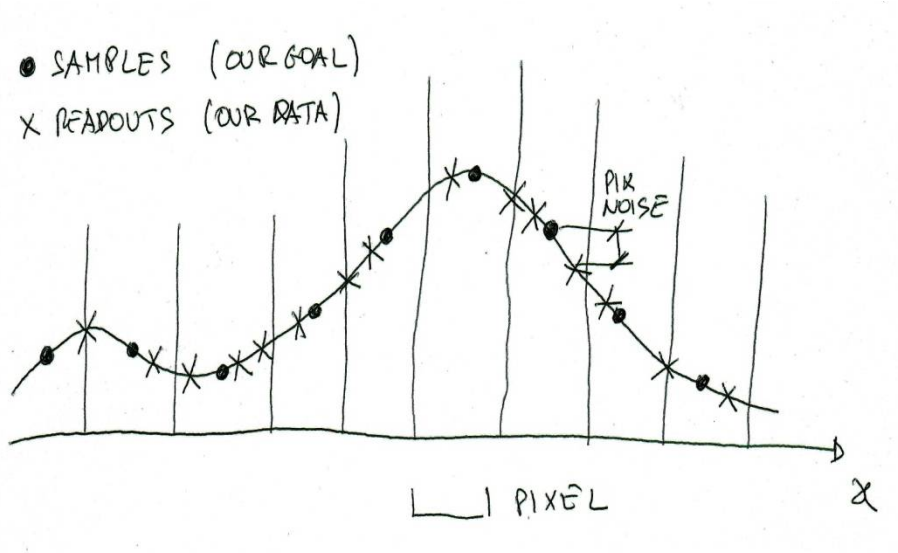
A preliminary attempt to realise the equalisation, based on an improved pointing matrix, turned unsuccesful. Resons not yet clear. But this attempt showed that this is a tough problem.

We plan to tackle the problem in the near future.

Pixellisation noise

Since the sky is band-limited after PSF convolution, it can be represented by its samples taken at the Nyquist rate.

These samples (the black dots) are our goal and are the ideal pixel value. However the readouts (the Xs) are not taken exactly on the samples neither on any regular grid.



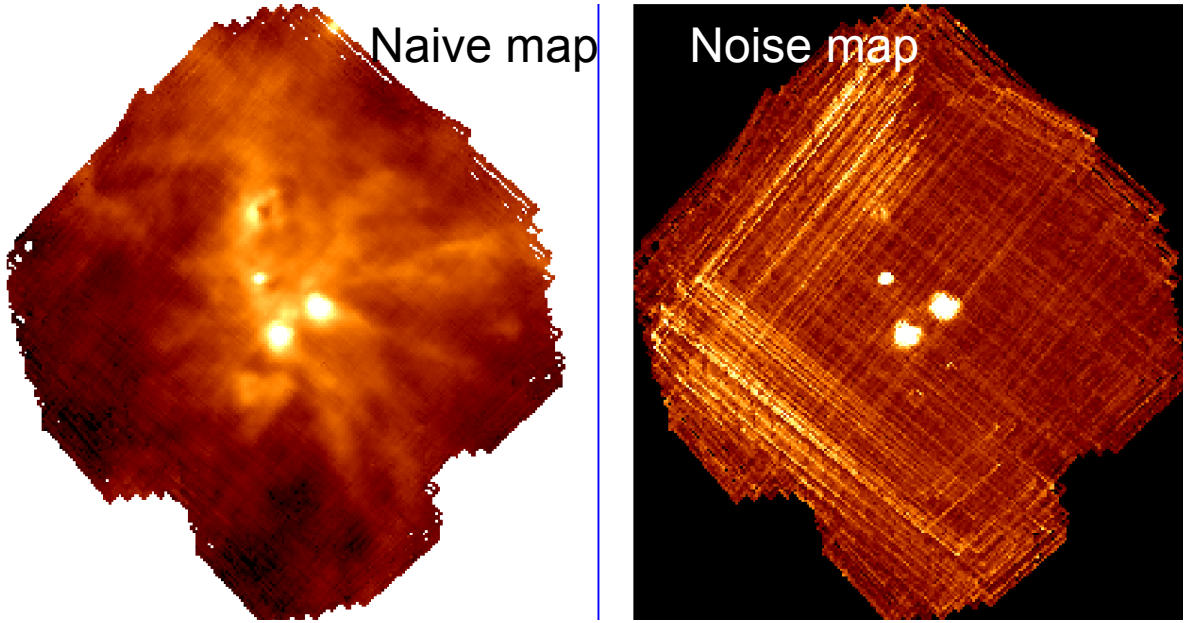
As a result, there will be a diference between the readouts and the desired pixel value **even in the absence of thermal and electronic noise**.

This difference can be called the **pixellisation noise** and can be thought as an additional noise affecting the readouts.

Pixellisation noise

Unimap saves a **noise map**, giving an estimate of the noise in the naive map (standard deviation of the readouts falling into each pixel).

The pixellisation noise is immediately seen in the noise map !



As expected, it peaks around the bright sources, where it is the **dominant** noise: the 1/f noise (the stripes) is much lower.

While it is a strong noise, pix noise is **not accounted for in any map-maker**.

Indeed, it is **not accounted for in the data model** used for the GLS estimation: we claim (and will verify) that this is the **main cause of the GLS distortion**.

Pixellisation noise

It is not difficult to modify the data model in order to account for the pix noise. We simply write

$$d = Pm + n + x$$

where *x* is the pix noise vector. The difficult part is to derive the covariance matrix.

We have developed a procedure to estimate the pix noise covariance matrix, $Y = E\{xx^T\}$, from the data.

The pix noise is statistically independent from the electronic noise: therefore the covariance matrix of the total noise (pix + electron) is $N+Y$.

Based on the latter fact, the GLS estimate is

$$\overline{m} = [P^T (N+Y)^{-1} P]^{-1} P^T (N+Y)^{-1} d$$

Pixellisation noise

Given the GLS estimate

$$\overline{m} = [P^T (N+Y)^{-1} P]^{-1} P^T (N+Y)^{-1} \, d$$

we can still use the PCG to compute it.

However we now have to perform a multiplication by $(N+Y)^{-1}$ instead that with N^{-1}. This is a significant complication. No more block Toeplitz.

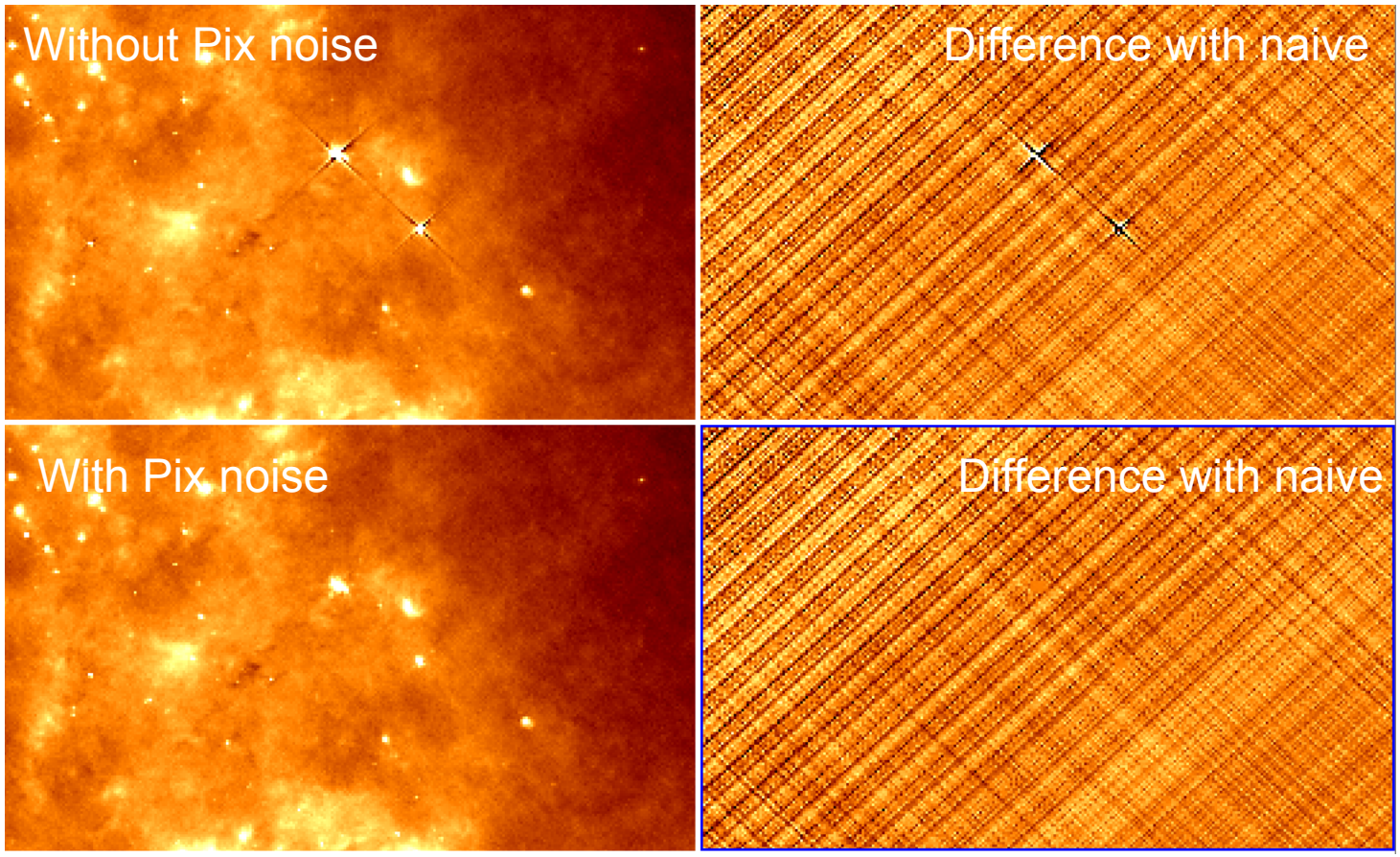
Currently, we realised a prototype that, in order to perform the multiplication, exploits a second PCG procedure, nested within the main PCG used to compute the GLS.

This causes a dramatic increase in the running time (from 10 to 100 times !) making the approach unfeasible for big data.

We are working to find a smarter procedure.

Pixellisation noise

Figure shows the GLS map and the difference with rebin for classical GLS (top) and GLS with pix noise (bottom). Distortion disappears when pix noise is accounted for.



Pixellisation noise

Is this procedure revolutionary? No. Using the PGLS we can produce maps having comparable quality.

Is this procedure useful? Yes (provided that we can reduce the computational complexity). These are the main advantages we see:

Theoretical

• Since the distortion disappears, we conclude that pix noise is the source of the distortion.

• The procedure yields the MV and ML map. This cannot be proven when PGLS/ WGLS are used (even thoguh this is essentially true)

Practical

• The procedure avoids the PGLS noise increase.

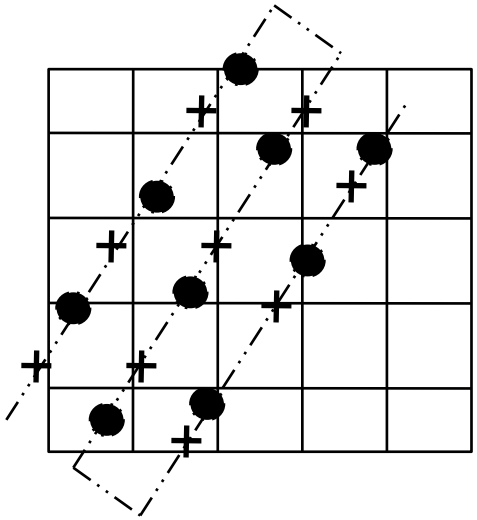
• It reduces the number of parameters to tune (PGLS and WGLS no more needed).

• It opens the way to the computation of a meaningful **noise map**, yielding the variance of the noise in each pixel of the final, GLS map.

Synchronisation / Pointing Error

The timelines may be shifted with respect to their correct position. There are two main causes

- systematic error in the sampling time
- delay inflicted by filtering or processing operations



Any time shift in the timelines translates into an erroneous pointing information. The samples are shifted along the scan line.

This problem surely affected the first PACS and SPIRE SPGs. The pointing computation was improved in the latest SPGs, and this problem is now minor.

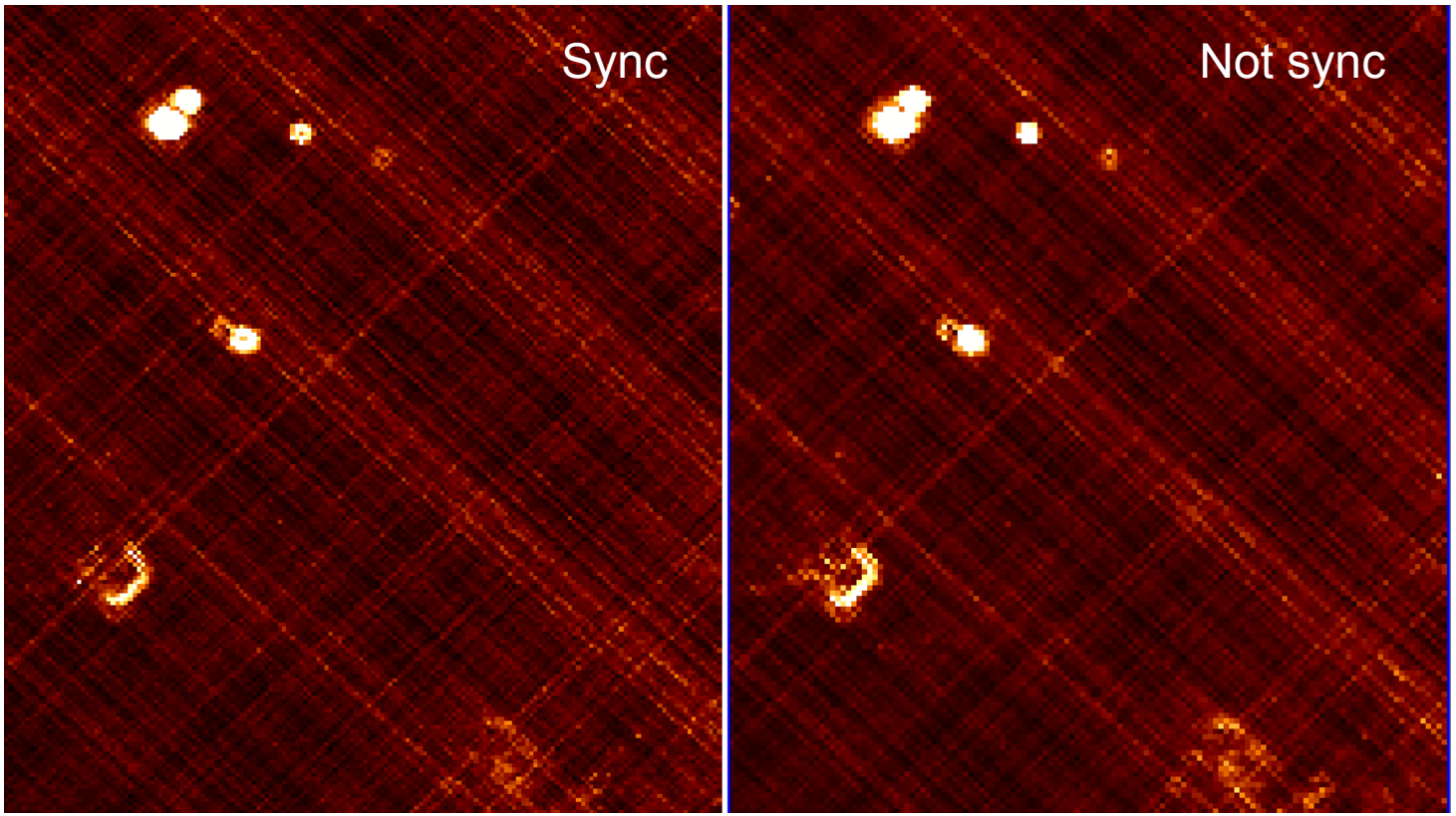
Never the less it is worth to better investigate this topic: a synchronisation procedure has been developed and a working prototype exists.

Absolutely preliminary results indicate that a small shift still exists, even in SPG13 products.

Synchronisation / Pointing Error

For M16 red (SPG13) we measure a shift of 0.7 / 0.33 arcsec for the nominal / othogonal scans (pixel is 3.2 arcsec, sampling space is 2 arcsec).

Figure shows the noise map with and without synchronisation. A noise reduction is observable around the sources. Also the GLS distortion is reduced.



Synchronisation / Pointing Error

Is this procedure a major improvement? No.

• The difference in the maps is minimal: few percents on bright sources.

Is this procedure useful? Perhaps.

• Even a few percent improvement may be worth: every little helps.

• The procedure can be efficently implemented and marginally increases the overall computation time (say 5-10%).

• The procedure can be used to check the quality of the pointing.

• For SPIRE we have the feeling that some of the filters implemented in the pipeline prior to level 1 amplify the noise. We plan to remove these filters and replace them with the synchronisation procedure.

SCUBA2

A Unimap version for SCUBA2 data has been implemented (prototype).

SCUBA2 turned out to be a much harder problem than PACS and SPIRE.

This is due to the presence of a Scan Synchronous Noise (SSN). This is a noise dependent on the pointing.

The data model is as follows

$$d = Pm + Ps + n$$

where *Ps* is the SSN. It can be seen that this noise is indistinguishable from the map. No linear approach can remove it.

As a result non-linear approaches are currently being explored, but some distortion will always be there.