Welcome to the PIIC



Pointing & Imaging In Continuum

a code by Robert Zylka

Stefano Berta and Robert Zylka

Tutorial v. 5.5.3
June 27th, 2026

Acknowledgements: we thank A. Sievers, C. Kramer and A. Schmiedeke for their careful reading of the Oct. 31st 2019 version of this document, and all users who sent us comments and suggestions (A. Beelen, G. Lagache, A. Rigby, J. Macias-Perez, S. Andreon, R. Adam, L. Pagani, and many others). We are grateful to C. Kramer for his help and scripts for the Fourier analysis in App. C and to A. Beelen, C. Kramer, A. Rigby and Q. N. Luong for testing the beta version of the multi-core processing scripts. We are very thankful to N. Ponthieu for his support and help with the development of the PIIC polarimetry routines and for making available his IDL code, to P. André and I. Myserlis for their tests of PIIC-Pol and their useful suggestions, and to P. André for his precious and constructive help into figuring out the correction of instrumental polarization. Furthermore, we thank P. Zhou and M. Nonhebel for the intense brainstorming during a long winter storm that lead to the development of the IP correction using QSOs. We wish to thank J. Pety for his invaluable work with the CUBE script to perform the operations in Fourier space for the correction of the instrumental polarization, and S. Bardeau for his continuous support in the compilation, the assembly and the distribution of PIIC. Special thanks go to C. Bartels (MPA/MPE library in Garching) and D. Lutz for their help in finding the long-sought Haslam (1973) publication [2026/04]. Last but not least, SB warmly thanks A. Bongiovanni and M. Beltran for the erudite discussions at the Telescope that lead to the new quotes introduced in v.5.4 of this tutorial.

SB thanks RZ for his help, support and availability during the years spent working together on PIIC, and wishes him big adventures!

Contents

1 Welcome to PIIC! 9
- 1.1 Acknowledging and citing PIIC . . . 9
- 1.2 PIIC release . . . 10
- 1.3 The PIIC calibration files . . . 10
- 1.4 Setting up PIIC . . . 10
- 1.5 Starting PIIC . . . 11
  - 1.5.1 The graphic interface . . . 11
  - 1.5.2 Using external commands . . . 11

2 Blind use of the PIIC science pipeline 13
- 2.1 Important warning . . . 13
  - 2.1.1 Note about scan speed . . . 14
- 2.2 Weak compact sources . . . 14
- 2.3 Strong compact sources . . . 14
- 2.4 Deep fields . . . 14
- 2.5 Sunyaev-Zeldovich clusters . . . 15
- 2.6 Complex sources . . . 15

3 The science data-reduction pipeline 17
- 3.1 Principles and challenges of NIKA2 science data reduction . . . 17
- 3.2 Setup . . . 18
  - 3.2.1 The directory tree . . . 19
  - 3.2.2 The list of scans . . . 19
  - 3.2.3 `Select obsProc` type . . . 20
  - 3.2.4 The template script . . . 20
  - 3.2.5 Specific cases . . . 22
  - 3.2.6 Order of baseline correction . . . 24
  - 3.2.7 Deep field and weak source options . . . 24
  - 3.2.8 Measuring the noise r.m.s. . . . 24
  - 3.2.9 Additional parameters . . . 25
  - 3.2.10 Correlating pixels . . . 25
  - 3.2.11 Correlating pixels: the case of extended sources . . . 25
  - 3.2.12 Pixel size . . . 25

3.2.13 Exclude noisy timelines/maps . . . 26
3.2.14 Pausing . . . 26
3.2.15 Producing diagnostic diagrams and saving them as PNG figures . . . 27
3.2.16 Overwriting . . . 27
3.3 Run it! . . . 27
3.3.1 Sky opacity along the line of sight . . . 31
3.3.2 Information about re-gridding . . . 31
3.3.3 Batch mode . . . 31
3.4 Quick analysis of the 0-th iteration results . . . 33
3.4.1 Plot a map with PIIC . . . 33
3.4.2 The intensity maps . . . 34
3.4.3 The weight maps . . . 34
3.4.4 Compute r.m.s. and S/N maps . . . 34
3.4.5 Verify the source defined in the script . . . 36
3.4.6 The noise r.m.s. polygon . . . 37
3.5 Iterative mode . . . 37
3.5.1 Verify convergence . . . 37
3.5.2 Dynamically varying parameters . . . 38
3.6 Products . . . 38

4 Additional tips and advanced use 39
4.1 Non-azimuthal scans . . . 39
4.2 Scans with different centers . . . 40
4.3 Flux units . . . 40
4.4 WCS . . . 41
4.5 Define your source . . . 42
4.5.1 Offsetting the source patch . . . 43
4.5.2 Polygonal source definition . . . 44
4.6 Baseline correction and base range . . . 44
4.7 Effects of sky noise removal and of instabilities on extended and diffuse emission . . . 45
4.7.1 Faint extended sources . . . 45
4.7.2 The effect of NIKA2 instabilities . . . 46
4.7.3 Uniform emission . . . 46
4.8 Noise of the final map . . . 47
4.8.1 Noise r.m.s. across the map . . . 47
4.8.2 Trimmed edges . . . 49
4.8.3 Excluding the outer regions from the computation of the source map . . . 49
4.9 Noise of individual scans . . . 50
4.10 Number of KIDs effectively used . . . 50
4.11 Manually define, verify and optimize the noise r.m.s. polygon . . . 52
4.11.1 Produce the noise r.m.s. polygon . . . 52
4.11.2 Verify the polygon . . . 53
4.11.3 Optimize the polygon a posteriori . . . 53

4.12 Scans of very different length . . . 54
4.13 Cumulative signal and weights . . . 54
4.13.1 Saving only the last cumulative map . . . 55
4.13.2 Combining cumulative maps of sub-sets of data . . . 55
4.13.3 Linking `rgw` and effective exposure time . . . 56
4.13.4 Using `rgs` and `rgw` to produce individual scan maps . . . 56
4.14 The source map . . . 57
4.14.1 Continuing where PIIC was stopped (during the iterative mode) . . . 57
4.14.2 Using a pre-defined source map . . . 57
4.14.3 Produce the source map a posteriori . . . 58
4.14.4 Enhance detections by smoothing . . . 58
4.15 Adding artificial sources to the timeline . . . 60
4.15.1 Null maps (a.k.a. jackknife) . . . 61
4.16 Saving maps of individual scans . . . 61
4.16.1 Combine the maps of individual scans . . . 62
4.17 Hit map . . . 62
4.18 Pointing and calibration correction between scans . . . 63
4.19 What if reading errors occur? . . . 65
4.20 Saving time . . . 65
4.20.1 Reduce the read and pre-processing time . . . 65
4.21 Moving targets data processing . . . 66

5 Speed-up: multi-core processing 69
5.1 Generalities of PIIC parallel processing . . . 69
5.2 Splitting the list of scans . . . 70
5.2.1 Important caveats . . . 70
5.3 Preparation of the scripts . . . 73
5.3.1 The master script . . . 73
5.3.2 The slave script . . . 73
5.3.3 Scans with different sizes . . . 74
5.4 Run it! . . . 74
5.4.1 Running the multi-core processing in *batch mode* . . . 75
5.4.2 What to do if it fails on some scan . . . 75
5.5 Products . . . 75
5.5.1 Content of the `red/` sub-directory . . . 76
5.5.2 Content of the `stat/` sub-directory . . . 76
5.5.3 Content of the `msA*i%/` sub-directories . . . 76
5.6 Additional important notes . . . 77
5.6.1 Saving individual maps . . . 77
5.6.2 Comparison to single-core processing and advantages . . . 77

6 PIIC-Pol: reduce NIKA2 polarimetry data 79
6.1 Basic needed knowledge and suggested reads . . . 79
6.2 Further details of NIKA2 polarimetry . . . 80
6.3 Observing with NIKA2-Pol . . . 81
6.4 Operations performed by PIIC-Pol . . . 81
6.4.1 Parasitic signal . . . 82
6.4.2 Demodulation and lock-in . . . 83
6.4.3 Fourier low-pass filtering and scan speed limit . . . 84
6.4.4 One-dimensional gridding in time . . . 85
6.4.5 Iterative mode . . . 85
6.5 Prepare your scripts . . . 86
6.5.1 Extended faint signals and bright compact sources . . . 86
6.5.2 Use of an input source map . . . 86
6.5.3 Use a custom r.m.s. polygon . . . 87
6.6 Products . . . 87
6.7 Basic post-reduction operations . . . 88
6.7.1 Read, plot, split FITS cubes with PIIC . . . 88
6.7.2 Produce polarimetry source maps a posteriori . . . 88
6.7.3 Produce polarimetry r.m.s. and S/N maps . . . 88
6.8 Plot polarization degree and angle . . . 89
6.9 Correction of the instrumental polarization (a.k.a. leakage) . . . 90
6.9.1 Preparing the data . . . 91
6.9.2 Preparation of the script . . . 92
6.9.3 Execution of the script . . . 92
6.9.4 Output . . . 93
6.9.5 Caveats . . . 93
6.10 Correction of the instrumental polarization using QSOs . . . 94
6.10.1 Case i) use a combination of a QSO and a non-polarized source . . . 94
6.10.2 Case ii) use an independent monitoring of the QSO polarization . . . 97
6.11 Example of a typical PIIC-Pol data reduction . . . 99
6.12 Important, to keep in mind . . . 100
6.12.1 Beam shape dependencies . . . 100
6.12.2 Consequences on the leakage pattern . . . 101

7 The quick look monitor 103
7.1 On the fly data reduction at the telescope . . . 103
7.2 Quick Look . . . 104
7.3 Simple use of QL . . . 104
7.4 Advanced example: use the QL on a refined list . . . 105
7.5 Operations and results . . . 106
7.6 Products . . . 106
7.6.1 Save FITS files . . . 107
7.7 Additional setup parameters . . . 109

References **112**

Appendix

A Content of PIIC NIKA2 DAFs **113**
A.1 CAL files . . . 114
A.2 DRP files . . . 114
A.3 NKFR files . . . 115
A.4 NKSLEX files . . . 115
A.5 RPP files . . . 115
A.6 TAU files . . . 116

B Effects of re-gridding **117**

C Transfer function examples and challenges **119**
C.1 List of term and acronyms used in this Appendix . . . 119
C.2 Point-like sources transfer function . . . 120
3.2.1 Recovered point-like sources flux . . . 121
3.2.2 Noise properties of Null Map *vs.* Final Map . . . 121
3.2.3 The effect of changing baseline fitting order . . . 123
3.2.4 The ideal case of simulated pure Gaussian noise . . . 124
C.3 Extended sources transfer function . . . 124
3.3.1 Properties of the input artificial map . . . 127
3.3.2 The effect of the zeroing radius . . . 128
3.3.3 Re-shuffling the list of scans . . . 129
3.3.4 Caveats, warnings and null maps . . . 129
3.3.5 Instabilities induced by the scanning direction . . . 132

D PIIC dictionary **135**

E History of PIIC releases **139**

PIIC Cookbook **i**

Chapter 1

Welcome to PIIC!

Let's have a Jamon y Queso
snack at Pico Veleta

The *Pointing and Imaging In Continuum* software is developed by Robert Zylka at IRAM, with the help of Stefano Berta in the most recent times. It is the extension of the MOPSIC data reduction software (for MAMBO) to the case of NIKA2 data. It consists of two main components: the so-called *Quick Look monitor* (briefly known also as QL); and the data reduction *pipeline* aimed at producing science-graded products. As of its 9th release, PIIC also supports NIKA2 *polarimetry* data, producing Stokes I, Q, U maps.

The core of PIIC is written in Fortran-90 and defines the commands to be used for the data reduction and analysis. The data reduction is carried out by scripts that combine the core commands to perform all operations on the data. Plotting is also included in the main scripts and calls GREG/SIC functions, which are part of the GILDAS distribution.

This tutorial is a brief hand-in-hand guide to accompany the users through PIIC operations. It takes for granted that the user already has some basic knowledge of how the NIKA2 instrument is structured and works (Calvo et al. 2013; Catalano et al. 2016; Adam et al. 2018; Perotto et al. 2020), as well as has some experience with ground-based, single-dish, millimeter astronomical observations.

For data reduction support, questions, doubts, suggestions, please do not hesitate to contact us, by writing to *piic@iram.fr*.

1.1 Acknowledging and citing PIIC

If you are using PIIC for the reduction and analysis of NIKA2 (or other) data, please acknowledge the efforts of the developers by adding a note in your publications and citing the PIIC seminal paper. Please include in your acknowledgements the sentence: *The NIKA2 data were processed using the Pointing and Imaging In Continuum (PIIC) software, developed by Robert Zylka and Stefano Berta at the Institut de Radioastronomie Millimetrique (IRAM) and distributed by IRAM via the GILDAS pages. PIIC is the extension of the MOPSIC data reduction software to the case of NIKA2 data.*

Please be also so kind to cite the Zylka (2013) seminal paper, as well as this Tutorial (Berta & Zylka 2019-2025), in your publication. Thank you.

1.2 PIIC release

The first release of PIIC was on October 2019. The software is distributed via the GILDAS pages[1]. It consists of a tarball file containing the main components of PIIC, an archive of the calibration files (regularly updated), a Read-Me file, and this tutorial. To install PIIC, simply follow the instructions found in the Read-Me file.
A history of PIIC releases is provided in Appendix E.

1.3 The PIIC calibration files

The PIIC calibration database (called DAFs) contains five different main types of files: calibration files (CAL), defining the response of KIDs (*kinetic inductance detectors*) for flux calibration; deleted receiver pixels (DRP) files, listing those KIDs that are known to be defective; frequency files (NKFR), listing the natural resonance frequencies of all KIDs, for different sweeps[2]; receiver pixels parameters (RPP), listing the position of each KID in the field of view of NIKA2, for different sweeps; atmospheric conditions over all observing runs (TAU files), containing the values of the atmospheric opacity, τ, produced by the Observatory's tau-meter. Appendix A describes the content of the DAFs files.

After each observing pool, the PIIC support team processes calibration data and produces new calibration files. The DAFs database is thus updated and a new version is available online. This process might take some time, therefore there is a natural delay between science observations during a pool and the release of the related DAFs. The DAFs directory includes a PDF document that summarizes the properties of the available DAFs for each science pool, including calibration uncertainties.

At the moment, DAFs cover the period from October 2017 until March 2022. This means that DAFs for all NIKA2 science pools so far have been produced and are released. There is no plan to produce DAFs for non-science weeks carried out before October 2017.

1.4 Setting up PIIC

Install PIIC following the instructions provided in the release Read-Me file. PIIC does not need GILDAS to be installed first. See the PIIC installations notes for instructions on how to avoid conflicts with pre-existing GILDAS installations on your computing machine. The PIIC installation directory contains the following sub-directories:

```
dafs/
doc/
etc/
pro/
x86_64-generic/
```

where `dafs/` contains the calibration files; `doc/` includes the PIIC online documentation files; `pro/` is the repository of all PIIC scripts; and `x86_64-generic/` is the actual code installation folder.

[1] `www.iram.fr/~gildas/dist/index.html`

[2] a *sweep* is practically a re-definition of KIDs resonance frequencies, performed regularly for each new season.

1.5 Starting PIIC

Once the necessary environmental variables are set (see the INSTALL instructions file), start PIIC by simply typing `piic`. The PIIC prompt will look like this:

```
PIIC>
```

1.5.1 The graphic interface

Plots, diagrams and image displays are visualised on a graphical window, following the GILDAS' syntax. The QL opens the graphical device automatically. The science data reduction pipeline does not, but it creates the required diagrams and saves them on file if requested (see Sect. 3.2.15). If other operations need to be performed — without running the QL or pipeline — the user needs to explicitly open the graphic device. This can be done either after starting the software, i.e. from the PIIC prompt, or at the start itself. Three graphical examples follow.

In the PIIC prompt type either

```
dev i
dev i b
dev none
```

or while starting PIIC type:

```
piic dev i
piic dev i b
piic dev none
```

An "`i`" device is a (possibly interactive) graphical window with white background; if the user likes a black background better, the chose "`i b`"; finally, if no device is wished, then use the option "`none`".

If PIIC is operated remotely — e.g. with a low-speed net — or if the user is experienced enough to trust all operations without the need for visual inspection, or if a huge workload is foreseen and the user would like to avoid slowing down the processes by displaying many diagrams, then `dev none` could be the option of choice (note that this is the default for the science pipeline but not for the QL).

It is very important to note that `dev none` is *different* from not specifying any device. If the given PIIC script plots diagrams, it will stop in case no device is specified. On the other hand it will redirect the plot to the `none` device if specified as such, thus proceeding with script operations without stopping but also avoiding slow downs due to displaying graphics.

1.5.2 Using external commands

Just as in GILDAS, to use an external command (e.g. shell) from the PIIC prompt, simply type the name of the command, preceded by the "`$`" sign, e.g. `$ls`. On the other hand, if external commands need to be included in any PIIC script, then the `sys` (`system`) command has to be used.

Chapter 2

Blind use of the PIIC science pipeline

> Aglio olio, so simple and yet so good! (And so difficult to say)

Performing data reduction with PIIC is simple. It consists of simply editing a template script, setting it properly for the given dataset, and running it. This short Chapter is a description of typical setups for common types of targets, dedicated to those users who do not want to enter much in the details of PIIC setups. A more detailed description of operations can still be found in Chapter 3.

If this is your case — after preparing PIIC and its environment as described in Section 3.2 — edit the template script as described in this Section.

We start, however, with a very important warning about the setup and optimization of the observing strategy (Sect. 2.1). Do not omit to read it!. The outcome of data processing depends also on the adopted observing setup.

2.1 Important warning

As a general rule for multi-beam, sub-mm continuum mapping observation, the size of maps along the scan direction, shall be at least:

$$\text{NIKA2 FoV} + 2 \times \text{NIKA2 beam width} + \text{source size (above the noise)} \tag{2.1}$$

plus an additional term accounting for the NIKA2 instability triggered by the 30m telescope tracking behaviour (see, e.g., the IRAM/NIKA2 call for proposals). Note that it is assumed that the source is centered in the map. If not, the map size should be increased by the value of the offsets from the center.

Using this formula ensures that the correction of data instabilities (intrinsic to NIKA2 or induced by the chosen mapping strategy) is optimized. If this condition is not fulfilled, the results might become unpredictable, because some crucial processing steps cannot be performed properly. The header of the template scripts includes some additional comments about this topic.

2.1.1 Note about scan speed

The map size is not the only parameter defining the scan strategy. The choice of the scan speed plays also an important role in optimizing the results. A high scan speed lowers the $1/f$ noise and allows to approach the stability timescale of the instrument. On the other hand, it can also induce other instabilities of the system (e.g. enhanced shaking when changing direction, magnetic fields produced by the motors, acoustic noise, etc.).

For setting up the optical observing strategy for your project, please refer to the technical description of observations on the IRAM/NIKA2 Spanish web pages[1] and to the documentation accompanying the call for proposals.

2.2 Weak compact sources

Targets are considered *compact* if their radius is < 2.5 times the half power beam width (HPBW) of NIKA2/30m. In this case, use the template script with its default settings (simply change filenames). When the source is also *weak*, use the option `let weakSou yes`.

2.3 Strong compact sources

If the target is a bright source, expected to be detected at S/N>10, and visible on individual scans, use the template script with its default settings, keeping the `weakSou` set to *no*. Note, however that 20 iterations (i.e. the default value of `nIterSource`) might be insufficient to reach a satisfactory convergence.

2.4 Deep fields

If the target is a *deep* (blank) *field*, potentially including several weak sources, edit the template scripts such that:

```
let deepField yes
let weakSou yes
let souRmxAS 0          ! as default
let souRmnAS 0          ! as default
let blOrderOrig 5       ! or higher
let nIterSource xx      ! depending on S/N, etc.
```

We remind that the re-definition of optional parameters has to be added to the data reduction script (the so called template script) *after* the call to the `mapTPoptionalSets.piic` script (see Sect. 3.2.9).

[1] `https://publicwiki.iram.es/Continuum/PoolPreparation`

2.5 Sunyaev-Zeldovich clusters

This is a special case of weak, extended sources, with moderate size. Keep the default setup, but re-define the source a priori using the available knowledge about the target: its expected position and size (see Sect. 4.5).

If using the iterative mode, in order to detect the negative signal, switch the `souSign` parameter[2] to "-". If both positive and negative signals are present (e.g. both the SZ cluster and foreground/background/lensed galaxies are in the field), set `souSign` to "+-".
Summarizing:

```
let deepField no
let weakSou yes
let souRmxAS xx          ! your source size
let souRmnAS xx          ! your source size
let souPA xx             ! your source P.A.
let souSign "-"
```

Remember that the re-definition of optional parameters has to be added to the data reduction script *after* the call to the `mapTPoptionalSets.piic` script (see Sect. 3.2.9).

Finally , take a look to Section 4.14.4 that describes how to use the "S/N map smoothing": important in the case of SZ targets.

2.6 Complex sources

By these, are meant extended objects with complex geometries, possibly including diffuse emission. Adopt the iterative mode, and do not use any a priori definition of the source. For a first, quick data processing, edit the template script changing the default settings of the following parameters:

```
let souRmxAS 0          ! as default
let souRmnAS 0          ! as default
let deeField no         ! as default
let blOrderOrig 2       ! or 3
let nIterSource xx ! more than 20 iteration might be necessary
```

As usual, the re-definition of optional parameters has to be added to the data reduction script *after* the call to the `mapTPoptionalSets.piic` script (see Sect. 3.2.9).

Don't omit to read Section 4.14.4 about how to enhance the detection of sources by smoothing the S/N map during thresholding. If the dataset consists of scans of very different lengths (e.g. in different directions), please read Sect. 4.12. Moreover, Sect. 3.2.11 gives indications about advanced sky noise evaluation for this case.

[2]The `souSign` parameter applies only to total power and Stokes I. The detection of Stokes Q and U signals includes always both positive and negative pixels, regardless of the value of `souSign`.

Chapter 3

The science data-reduction pipeline

Celebrate your data with a unique carbonara fusion!

NIKA2 observations are carried out with a raster-scan scheme (i.e. "on the fly" scanning). The field of view is moved on the sky, scanning a region centered on the target, and covering an area around it large enough to allow for a proper characterization of sky noise and instrumental effects.

The main issues to deal with, when reducing science maps, are instabilities, both due to the KIDs behaviour itself and due to changes in elevation when scanning in directions other than azimuthal.

The science data reduction consists of a combination of PIIC commands and GREG/SIC (GILDAS) macros. The combination of the two performs all needed operations and produces several diagnostic diagrams.

The data reduction of a science data set is carried out using a single PIIC script, which calls all needed procedures.

3.1 Principles and challenges of NIKA2 science data reduction

NIKA2 data include the signal of the sources, of the sky, of the system telescope+instrument and noise. Sky signal and its low-frequency noise, as well as instrumental instabilities, should be removed as much as possible before transforming the NIKA2 data into a science-quality product.

The essential steps are: flat field correction, correlation correction, sky subtraction, baseline subtraction, calibration, geometry association (given by the known RPPs), re-gridding on the final map. During these operations, particular care should be taken in properly dealing with the emission of the sources, in order to preserve it.

The sky signal and its variations are monitored and measured during the time-line of each KID (also known as receiver pixel, RP). The correlation of each KID to all others is studied and only the best correlating KIDs are used to compute the correlated noise to be subtracted from each timeline.

When performing these measurements, the records of those KIDs observing sources at the given time need to be excluded, in order to avoid compromising the measurement of the sky and instrumental properties. This is done by defining the sky area covered by sources (either as an ellipse or a polygon, see Sects. 3.2.4 and 4.5).

If the position, size and shape of the sources is known, and the size is smaller than the NIKA2 field of view (FoV), then it is possible to define the area to be avoided in the sky. If not, it is possible to let PIIC define it on the basis of the signal to noise (S/N) ratio in the final map, during the iterative process. Similarly, a pre-defined source will be improved on the basis of S/N.

The definition of a source gives correct results only if the interpolation of the corrections across the source is reliable. This is not always the case; it is a "safe" assumption only for "not weak sources". Therefore the default setup of the PIIC data reduction pipeline is for non-weak objects, without a user-defined source, and with an automatic r.m.s. polygon definition.

The data reduction procedure is an iterative (and interactive) process. At each iteration, all data reduction operations are repeated, using the improved source definition. To compute the S/N ratio, the weight map produced at the previous iteration is rescaled into a noise map, by simply measuring the r.m.s. of the signal map in a source-free region determined by PIIC or within a source-free polygon defined by the user. Thus the S/N ratio is computed for each pixel, and pixels about a certain S/N threshold are identified. In this way a *source map* is built and is subtracted from the data timeline at the next cycle of the loop.

The standard procedure can be thus summarized in two broad main steps:

1. in the so called *0-th iteration*, PIIC performs all operations using the definition of source given by the user (see Section 4.5). This can be either an ellipse, a polygon or no a-priori definition of the source. The sky area covered by the source is excluded from the computation of baselines to be subtracted from the data.

2. in the subsequent *iterative mode*, PIIC loops N times and performs again the data reduction, improving the source definition on the basis of a S/N ratio thresholding method.

Between these two main steps, it is possible to use the results of the 0-th iteration to improve and optimize the setup of PIIC (e.g. the source definition itself), before proceeding to the proper iterative mode.

3.2 Setup

Before running PIIC, we need to prepare the necessary files and environment:

- the directory and sub-directories tree in which PIIC will work;

- the list of scans to be processed (and combined together);

- the template data reduction script, that will be modified and renamed, depending on your needs, taste and on the type of data to be dealt with;

- the definition of the source on the sky (if needed).

3.2.1 The directory tree

First of all, let's set up the working directory where PIIC will be used. For example, create the directory `my_data_red_with_piic/`. Similarly to the QL case, PIIC needs a whole directory tree to work properly. Prepare it in the following way:

```
mkdir my_working_dir
cd my_working_dir/
mkdir png red stat
ln -s directory_containing_data imbfitsDir
```

The list of scans needs to be produced only once, for each array, unless scans filenames or number change.

Note that the data files in the data directory are *not* modified by the PIIC data reduction, nor is anything going to be written in the data directory. The name of the symbolic link of the data directory must be `imbfitsDir` and shall not be changed.

3.2.2 The list of scans

The list of scans is an ascii file containing the names of the IMBFITS files belonging to the dataset of interest. It can be produced with any tool. Here an example is given to use PIIC to produce the list of all array-2 scans belonging to observations of an *hypothetical* source called "Superantennae":

```
cd my_working_dir
cd imbfitsDir
ls iram30m*-2-* > ~/my_working_dir/all_ar2.LIST    ! lists all Ar2 IMBFITS files
cd ..

piic

indir imbfitsDir                  ! define the input directory
inlis all_ar2                     ! load the list
sel SUPERANTENNAE*                ! selects only scans of object SUPERANTENNAE
sel obs m                         ! select scans with observation type = map
write inlist SUPERANTENNAE_a2     ! write the list of selected scans on file (.LIST)
init inlist                       ! if needed, resets the starting list (in memory)
                                  ! the user will need to start from scratch,
                                  ! if they wish to produce a new list
```

In this way, one ascii list has been produced for array 2 containing the names of all Superantennae IMBFITS files to be processed.

If the data IMBFITS files are all stored in the same directory, then the list can include only filenames, and the `imbfitsDir` symbolic link points directly to that directory. If, instead, the data are spread over several directories, the `imbfitsDir` points to the parent directory common to all of them and the list includes paths (starting from the common parent directory). In this case, in

order to comply to Fortran's conventions[1], path+filenames need to be written in double quotes: `"/path/imbf_filename.fits"`.

The filename of the list is to be inserted in the data reduction script (see Sect. 3.2.4). By default it is `"yourSource_a"'nikaBand'`. The default file extension is `.LIST`. In the above example, for the target called "Superantennae" and array 2, the default name is therefore `Superantennae_a2`.

Very importantly, the data belonging to each array need a different list of IMBFITS data files. In fact, remember that the data of each scan are stored into three separate IMBFITS files, each belonging to one single NIKA2 array.

If the user would like to process 1.3 mm data all together, i.e. would like to combine array-1 and array-3 data together, then the list shall be labelled `a13` and shall include both array 1 and 3 scans filenames. This means that the length of the arrays 1+3 IMBFITS list must be double the one of array 1 *or* 2 *or* 3 individual lists.

3.2.3 `Select obsProc` type

The `select` command has many options and can perform a selection of different kind of data, based on several different pieces of information. We invite the user to explore its possibilities by consulting its help page (simply typing `? select`).

The `obs` option, used in the previous Section, is the short form of `obsProc` and specifies the kind of observation procedure used to observe the given scans. Possible choices are: `map (m); pointing (p); focus (f); track (t)`, or a combination of them[2].

3.2.4 The template script

PIIC comes with a template script for science data reduction. It is found in the `pro/` folder under the installation directory and it is called `mapTP_a2_template.piic`. This template script needs to be edited and optimized for the given science case and the given kind of observations (type of source, map size, scan strategy, etc).

This and the following Sections describe the key parameters of the script and give some guidelines for choosing their optimal values. The users are encouraged to test different configurations and design the best setup for their project.

This script assumes that all the needed calibration files have been already produced and are in place in the PIIC `dafs/` database.

Figure 3.1 shows an example of template script, as found in the `pro/` folder and with an additional custom commands section.

As mentioned, the template script includes several parameters that can be optimized depending on goal and on type of data/targets. Nevertheless, the default values of most parameters are already a valuable start for almost any kind of data set and it is worth to perform a first test run leaving most of them unchanged.

The most critical parameters, to which the user should take special care are:

[1] The double quotes are strictly necessary only in the first character of the line is a / (slash).

[2] until the first half of 2018, the focus scans had been marked as `m` instead of `f`. Therefore extra care must be taken when selecting old scans by their `obsProc`



mapTP_a2_template.piic

```
!-------------------------------------------------------------------------------------------------
!
!  Written by Robert W. Zylka, zylka@iram.fr
!  created 02.10.2016, version 22.11.2023
!
!      execution: piic @scriptName   (.piic is the default extension)
!
! Pipeline to reduce NIKA2 maps in the iterative mode.
!
! IMPORTANT WARNING!
! If the maps in the scanning direction are shorter than
! 2*(souRmxAS+recArRmxAS+HPBW+scanVelocity),
! where souRmxAS is the source semi-axis in this direction, recArRmxAS the NIKA2 FoV
! radius, scanVelocity is in (units of souRmxAS,recArRmxAS,HPBW)/s,
! some crucial processing steps cannot be performed. Consequently, due to inevitable
! data instabilities (intrinsic to NIKA2 or induced by the chosen mapping strategy),
! the results in such cases might be questionable.
!
! Further notes:
! - For simple weak sources the best might be to define the source region
!   (as an ellipse or using a polygon) but keep the base range set to 0, i.e.
!   brRmxAS=0 brRmnAS=0 and polBReq=" " (defined in mapTPoptionalSets).
! - As a consequence of very low integration time and data instabilities, at edges
!   of the iterative maps S/N spikes and/or "instability" sources may appear.
!   To avoid their propagation while iterating towards the map centre, either polZmEq or
!   rZmEq should be used.
! - Defaults are set for Ar2, iterative mode (nIterSource>0) and not weak sources,
!   i.e. max. signal > 5*rmsOfNIKA2pixel.
!   The average rmsOfNIKA2pixel is roughly ~33mJy/23.8Hz for Ar2, ~110mJy/23.8Hz for Ar1,
!   and ~87mJy/23.8Hz for Ar3, but many KIDs show smaller r.m.s. than average,
!   therefore sources stronger than ~100mJy/2mmBeam and ~350mJy/1mmBeam are "not weak".
!
!-------------------------------------------------------------------------------------------------

sic mkdir red
sic mkdir stat

dev none            ! if display required e.g. dev i b, see help g\device

@ mapTPdefs

let nikaBand "2"    ! process NIKA2 A2 data, 1=A1, 3=A3, 13=A1+A3; other names not accepted

!------------------ loading the list of observations and selecting only maps -----------------------

init inList
init outList
inDir  imbfitsDir
outDir red
inList source_a'nikaBand' ! e.g. Superantennae_a2, give without the default extension .LIST
select obs m              ! only maps
!?sort inList             ! enable or disable depending on needs
list

!------------------------------------------------ parameter setting ------------------------------------------

let weakSou   no    ! if yes signal >5*rmsOfNIKA2pixel (see above) masked after the sky noise subtraction

let deepField no    ! yes for e.g. GOODSNorth, COSMOS, HDF, DeepField1, ...

let souSign "+"     ! "-" if source with negative signal, e.g. SZ; "+-" if positive and negative sources

let posSeq " "      ! [H M S D AM AS] centre of the final (R.A.,DEC) map
                    ! necessary only if individual maps have different centres

let eqExtrAS    0.0 ! [arcsec] half map extent in EQ, 0 to calculate it (all maps must have the same centre)
let souRAoffAS  0.0 ! [arcsec] R.A. offset of the source relative to posSeq
let souDECoffAS 0.0 ! [arcsec] DEC  offset of the source relative to posSeq

let blOrderOrig 2   ! order of instability corrections in subscans, for compact sources >2 might be better

let nIterSource 20  ! number of iterations; higher values might be necessary

if nIterSource.ge.1 then
   let rZmEq 0.0    ! [arcsec] if >0 outside of this radius the data of the iterative source are neglected
   let polZmEq " "  ! [arcsec] relative to posSeq, action as for rZmEq but using this polygon
   let smSNRpar 0.0 ! if >0 S/N is calculated using the smoothed map but the iterative source not smoothed,
                    ! if <0 also the iterative source smoothed; the ITERATIVE MAPS ARE NEVER SMOOTHED
end if

@ mapTPoptionalSets

!---------------------------------------- Ste's custom parameters ---------------------------------------

let nrPause 0       ! =1: pause after processing of each map, =2: and after "plot; ...

!------------------------------------------------ Run it! ------------------------------------------------

pause main

@ mapTPrun

exit
```

Figure 3.1: Example of data reduction script: default template plus few custom parameters.

- `nikaBand`: the array to be processed (the accepted values are 2, 1, 3, or 13);
- `inList, inDir, ...`: basic filenames and directories;
- `blOrderOrig`: order of baseline correction (i.e. of the polynomial fit to the baseline) over individual subscans;
- `deepField`: flag that indicates if we're dealing with "deepfield" observations;
- `nIterSource`: the number of iterations to be performed;
- `rZmEq` or `polZmEq`: the radius/polygon outside of which pixels are ignored during the definition of the source map;
- `smSNRpar`: the smoothing parameter used during S/N thresholding.

In order to better understand the meaning of these parameters, a very brief introduction to the main operations performed during the PIIC NIKA2 data reduction is needed. A brief overview was given in Sect. 3.1. More details are given in Section 3.3. Additional, important parameters are described in Sect. 3.2.9.

3.2.5 Specific cases

The details and tuning of the data reduction procedure depend on the kind of sources one is dealing with. This is particularly true when it comes to properly define source areas.

Simple sources, i.e. with simple and compact geometry are easy to deal with. A typical example is one single point-like source or a small extended source. In this case, a simple ellipse covering the target is sufficient.

More complex objects (e.g. extended sources still smaller than the FoV of NIKA2) are more difficult, but can still be dealt with without problems. A polygon defining an extended source smaller than the FoV of NIKA2 can be built with success.

Note that the polygon or ellipse must be smaller than the FoV of NIKA2 and should allow some pixels to be kept, in order to allow a proper computation of sky noise. The FoV of NIKA2 has a radius of $\sim$200 [arcsec]. Hence the source "mask" should be in general $\ll$200 [arcsec] in radius and undoubtedly not larger than 180 [arcsec] in radius (which anyway is already very big).

However, as pointed out in Sect. 3.1 the definition of a source gives correct results only if the interpolation of the corrections across the source is reliable. This is a "safe" assumption only for "not weak sources", hence the default setup of the template script is for non-weak objects, without a user-defined source, and with an automatic r.m.s. polygon definition.

See Sect. 4.7 for a description of the effect of sky noise removal on extended sources and diffuse emission.

Faint sources or blank-field case

In the case of a blank field, several faint sources are present on the final map, but one does not necessarily know a priori where they are. At the moment a multi-ellipse definition of several source is not implemented in PIIC and, in order to manually exclude all sources from the baselines computation, one should define a very complicated polygon covering all (mostly point-like) sources.

The straightforward PIIC approach is to adopt the iterative method, using no source patch at all during the 0-th iteration. In order to do this, simply set the sizes of the ellipse `souRmxAs` and `souRmnAs` to 0.0 [arcsec].

An unintended advantage of this choice is that — with no source defined on sky — PIIC avoids to transform back and forth from sky coordinates to Nasmyth and FoV coordinates the source patch, thus sparing a lot of spherical trigonometry computations and computational time.

Very large objects

Very large, extended objects, larger than the NIKA2 field of view are an interesting, complicated case.

The diffuse emission extended over an area larger than the NIKA2 FoV is suppressed by definition, during sky noise subtraction (see also Sect. 4.7). Only the small, more compact structures (e.g. cores and filaments embedded in a big Galactic cloud) make it to the final map.

The small structures can be treated similarly to faint sources in a deep field (see Sect. 3.2.7), i.e. simply letting PIIC to identify them on the basis of the S/N ratio on the final map. The easiest approach is therefore to adopt the iterative processing, with no a priori source definition. In this way, the flux of these structures is preserved and is not affected by filtering/baseline suppression. On the other hand, obviously no measurement of diffuse emission is possible with this approach.

Don't omit to read Sect. 4.14.4) about enhancing the source detection by smoothing the S/N map.

Negative signals

Compton scattering of CMB (Cosmic Microwave background) photons by free electrons in the plasma of galaxy clusters produce the well known Сюняев - Зельдович effect (SZ). The NIKA2 bands are designed such that the signal in the 1.3 mm band is roughly neutral and the 2 mm band detects the negative distortion of the CMB due to the SZ effect.

In order to detect negative signals, switch the `souSign` parameter to "-". If both positive and negative signals are present (e.g. both the SZ cluster and foreground/background/lensed galaxies are in the field), set `souSign` to "+-".

Being faint and extended, the S/N thresholding is likely to miss part of the SZ source. It might therefore be needed to defined an elliptical patch covering it. Although this might seem the easiest kind of targets to deal with, the combination of instabilities, small map sizes and the choice of the ellipse and other PIIC parameters can cause the production of spurious (negative/positive) signal, depending on the adopted settings (see Sects. 2.1 and 4.5, for example). Some experimentation might be needed in order to produce the best final product. Good luck!

3.2.6 Order of baseline correction

By default, the order of the baseline correction over single sub-scans (`blOrderOrig`) is 2. This should not be confused with `btOrder`, the order of baseline correction over the whole timeline, which is set equal to 1 and and in general does not need to be modified.

Different kind of sources are processed at best with different polynomial orders. The user is encouraged to experiment with larger values of `blOrderOrig`, with the aim to find the best option for their specific targets. In the case of complex sources, it is not possible to predict in advance what is the best value of `blOrderOrig`, because it depends of the actual dataset (maps, source, instrument status, temperature, atmospheric conditions, etc.). While optimizing `blOrderOrig`, it is wise to keep in mind that enough records should be available to evaluate higher-order polynomials.

Note that a higher order baseline fitting risks: *a)* to diverge at the edges; *b)* to fit not-perfectly "masked" sources or sky variations or other smooth, faint features, and thus might easily produce spurious signals; *c)* to remove real features of sources. Vice versa, under particular conditions (e.g. small maps) a lower value of `blOrderOrig` might give better results.

The definition of a source gives correct results only if the interpolation of the corrections across the source is reliable (see Sects. 3.1 and 3.2.5).

Finally, if the map is produced with scans of very different lengths (e.g. in different directions), please read Sect. 4.12.

3.2.7 Deep field and weak source options

In case of blank fields or very faint, point-like, sources, it is useful to switch the `deepField` and the weakSou parameters on. This tells PIIC that we are in presence of faint sources, i.e. sources that potentially are below the 5σ level.

When this option is active, the elliptical source definition is ignored and baselines are computed over the whole area, sources included. Therefore, for a proper use of elliptical source patches, the `deepField` option has to be switched off.

Finally, note that the `weakSou` option can be activated independently of the `deepField` parameter.

3.2.8 Measuring the noise r.m.s.

During the iterative process, source pixels are identified based on a S/N ratio threshold on the final map. PIIC does not produce noise maps explicitly, but it provides weight maps. A noise map can be produced by simply rescaling the weight map to the r.m.s. value of the final signal map (see Sect. 3.4.4). This implies that the r.m.s. should be measured in order to determine the S/N of each pixel of the final map and apply the thresholding on the iterative mode.

The r.m.s. of the signal map should be obviously measured in a region where no sources are present. By default, PIIC define automatically a polygon in an empty region of the map, in which the noise r.m.s. is measured (see Sect. 3.4.6). Alternatively, this polygon can also be defined manually by the user (see Sect. 4.11).

3.2.9 Additional parameters

The main data reduction script calls the script `mapTPoptionalSets.piic`, that defines additional parameters.

If users feel like modifying this file, for example to switch some special features on or to fine-tune some expert-mode parameters, they can proceed in two ways. They can simply re-define the "optional sets" parameters in the main (template) script, after the call to the `mapTPoptionalSets.piic` macro, or they can make a copy of the `mapTPoptionalSets.piic` script in the working directory and edit it there. In fact, PIIC finds its scripts and components in the `pro/` folder in the installation directory but — following GILDAS' conventions — equivalent scripts, with the same name, found in the working directory have priority.

3.2.10 Correlating pixels

By default the number of best correlating KIDs to be used when subtracting sky noise is `nBest=16`. One can experiment using more (e.g. 32, 64, ...) to see if the end products of the data reduction show a better source recovery. This parameter is defined in the "optional sets" macro.

3.2.11 Correlating pixels: the case of extended sources

As said, `nBest` is the number of KIDs used to calculate the approximation of sky noise. This can be computed as an average or a median over the `nBest` KIDs.

In the case of a median, point-like (or compact) objects are naturally rejected, because its signal is larger. If the source is extended, there is a chance that a subset n of the `nBest` KIDs is off-source. Two cases are possible: *a)* if $n > 50\%$ then the source is successfully removed using the median; *b)* if $n \leq 50\%$, then not.

In this latter case, the parameter `nBest2` comes to help: it specifies how many KIDs with the lowest signal out of the `nBest` are to be taken to evaluate the sky noise.

Note that a combination of `nBest` and `nBest2` is different than using a smaller value of `nBest` only. For example, `nBest`=64 combined to `nBest2`=16 is different than `nBest`=16!

Warning: the results will be correct only if not dominated by instabilities.

3.2.12 Pixel size

Among the many other available parameters, let us mention the pixel size of the final map (parameter `eqPixSizeAS` in the "optional sets" macro). By default, the pitch of the pixels of the final map are defined to be of 3 arcsec at 1 mm and 4 arcsec at 2 mm.

The HPBW is $\sim$ 11 arcsec at 1 mm and $\sim$17 arcsec at 2 mm. Therefore the chosen pixel size is more than sufficient for Nyquist sampling. A finer pixel scale is not necessary and would result into an enhanced noise and increased computation time.

3.2.13 Exclude noisy timelines/maps

PIIC checks the r.m.s. of timelines (signal as a function of time along the scan) of each KID. If such the average r.m.s. over all KIDs is above a given threshold, the entire map will be rejected from the dataset and will not be part of the final map.

In this way, scans taken in particularly unstable conditions can be discarded and will not degrade the quality of the final product.

The parameters that define these thresholds are found in the optional settings script and are called:

```
let mxA1todAvRMS 250.0       ! [mJy/beam/smplRateHz], A1 map will be excluded
                             ! if the TOI mean r.m.s. of all KIDs larger than this
let mxA2todAvRMS  70.0       ! [mJy/beam/smplRateHz], A2 map will be excluded
                             ! if the TOI mean r.m.s. of all KIDs larger than this
let mxA3todAvRMS 190.0       ! [mJy/beam/smplRateHz], A3 map will be excluded
                             ! if the TOI mean r.m.s. of all KIDs larger than this
```

These limits are set according to the statistics of NIKA2 data over the past years.

If users prefer not to exclude the noisy maps, they simply have to set these three thresholds to very high values.

The lists of scans, excluded from the final maps following this and other criteria, are saved in the `stat/` directory in files with `.excl` extension.

3.2.14 Pausing

If no problems arise the data reduction script proceeds without stopping. Nevertheless, it might be of interest to verify the operations performed on individual scans, or visualize and study the diagnostic diagrams shown on the active plotting device. It is possible to change if, when and how often the script pauses, by changing the value of the parameter `nrPause`, found in the `mapTPoptionalSets.piic` script.

By default `nrPause=0`, which means that the data processing makes no pause, if not one at the very beginning after having loaded the list of scans to be processed.

Setting `nrPause=5` implies that the data reduction script makes a pause after having completed the reduction of each individual scan. If the number of scans is large, this implies a large number of breaks. In iterative mode, the whole process is repeated `nIterSource` number of times, hence producing even a larger number of pauses, which one should then manually resume by pressing "c".

Other possible values of `nrPause` are listed in the script.

At every pause, it is possible to change the value of the `nrPause` variable, as well as of any other variable, if needed and if the user has enough experience to do that. While the script is idling, it is also possible to perform any other operation, run other scripts, display the combined image (map), verify if the polygon adopted for measuring the r.m.s. is positioned in a reliable, clean, well centered, etc, and/or verify that the data reduction is proceeding correctly. It is also possible to re-define the polygon and use a new, better, version at the next iteration (or after pressing "c").

If `nrPause` is set to 0 but one would like to make a pause and verify something or change the values of some parameters, it is always possible to interrupt the procedure, pressing `ctrl-c`. The script will pause at the first viable point, i.e. when the currently-running command ends, and it will be then possible to manually perform the desired operations.

It is possible to include the definition of `nrPause` in the template script. When doing so, take care to add it after the call to the `mapTPoptionalSets.piic` script (see Fig. 3.1).

3.2.15 Producing diagnostic diagrams and saving them as PNG figures

The pipeline produces several diagnostic diagrams and plots them on the chosen graphical interface. The user can activate or de-activate the production of different types of such diagrams, depending on their needs. Plotting takes time. In order to speed up the processing, it is suggested to switch off those plots that are not used or not interesting for the specific project. The parameters controlling which diagnostics to plot are found at the end of the `mapTPoptionalSets.piic`. The names and default values of these parameters are:

```
let plCC    yes        ! plot the Correlation Coefficient CC
let plRMS   yes        ! plot the rms in the TOD
let plRG    no         ! plot the maps of each KID
let plSig   no         ! plot the signals of each KID
let plFFc   no         ! plot the Flat Field Corrections
let plTPvar no         ! plot the total power variations
let plNuRPs yes        ! plot the Number of Used Receiver Pixels
```

Since the Spring 2021 release, PIIC does not save PNG diagrams by default anymore. This choice is driven by the evidence that not many users took advantage of the saved plots and by the wish to save space and processing time in the default setup of the data reduction template script.

All the diagnostic diagrams shown on the active plotting device can be saved on file by activating the *hardcopy* option. The parameter that controls it is called `hc` and is also found in the `mapTPoptionalSets.piic` script. By default its values is "no". Turn it to "yes" to switch it on.

3.2.16 Overwriting

The execution of the data reduction scripts halts before overwriting already existing FITS files and polygons. Cleaning or backup-ing the `red/` directory before starting a new reduction can be a valuable way to avoid these interruptions.

3.3 Run it!

After preparing the working environment, the data reduction script and associated data files, it is now time to run the PIIC data reduction. Always keep in mind the challenges and caveats described in Sect. 3.1 and throughout this tutorial document. Start PIIC in the working directory and run the script:

```
cd
cd my_data_red_with_piic/
piic

@reduce_my_data_Ar1.piic
```

PIIC will load the list of scans and pause. Verify that everything is correct and continue by pressing "c" and "enter".

The data reduction operations start and PIIC will perform in sequence the following:

1. assign sky positions to each KID; perform main-beam flat-fielding; exclude known problematic KIDs;

2. transform the KIDs signal into flux density and calibrate it to physical units [mJy/beam];

3. perform sky-noise (forward-beam) flat-fielding;

4. compute the correlation coefficient of each KID to all others; KIDs with bad correlation statistics are rejected;

5. analyse the time line of each KID and compute the r.m.s. of each KID's signal along the timeline (excluding the source positions, if requested); the most noisy KIDs are rejected;

6. subtract sky noise, using only the `nBest` KIDs best-correlating to each KID, and ignoring those records covering the source in the sky (if defined a priori);

7. subtract baselines to remove instabilities that could not be dealt with during sky-noise subtraction;

8. apply extinction correction;

9. if the NKSLEX DAFs are available for the date of the observations, evaluate the sky opacity along the line-of-sight from the scan's sky load, and refine the extinction correction and flux calibration;

10. distribute the signal of all KIDs onto the final pixel grid, using the appropriate kernel linking the NIKA2 PSF and KIDs size to the grid;

11. after the 0-th iteration (steps 1 to 10 above), the final map pixels with S/N ratio above the requested threshold are identified and a *source map* is produced;

12. in the iterative mode, the source map is subtracted from the data timeline (after calibration) and steps 4 to 11 are repeated `nIterSource` times.

Figures 3.2 to 3.5 show an example of the main steps for array 1, during a scan on the planet Uranus.

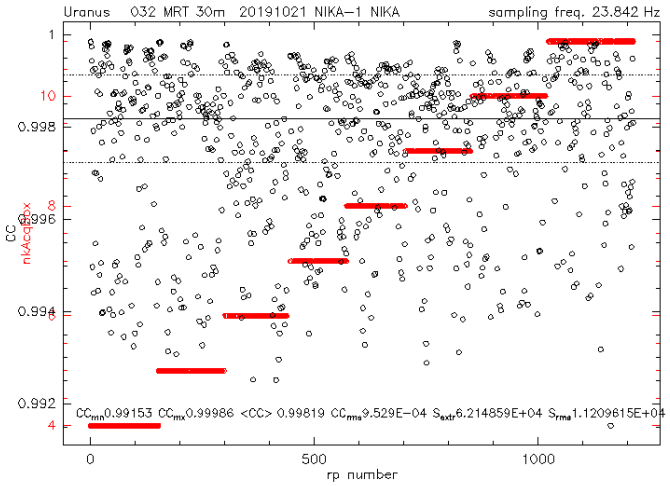


Figure 3.2: Average correlation coefficient of each receiver pixel (RP, i.e. KID) to all others, for array 1. In red are marked the RP id ranges belonging to NIKA2's 8 acquisition boxes of which array 1 consists (see also red y-axis label).

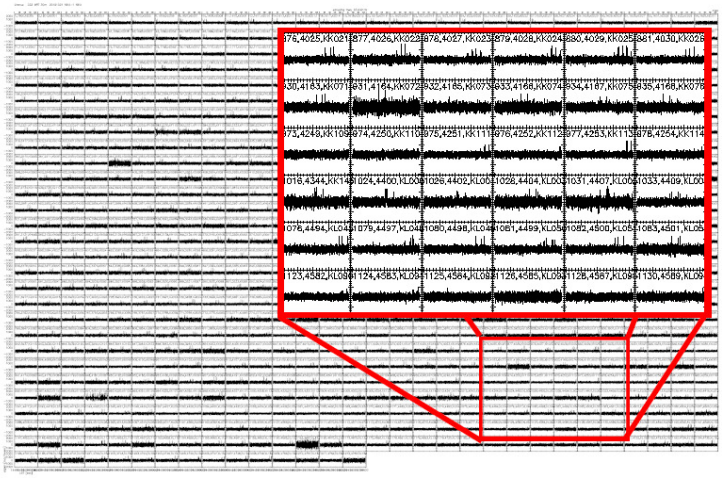


Figure 3.3: Time line of all KIDs of array 1, after the first baseline subtraction. The KIDs with the noisiest time lines are rejected (see also Fig. 3.5). The more pronounced spikes in the timelines are the bright source, crossed by the given KID at the given moment along the time line while scanning on sky.

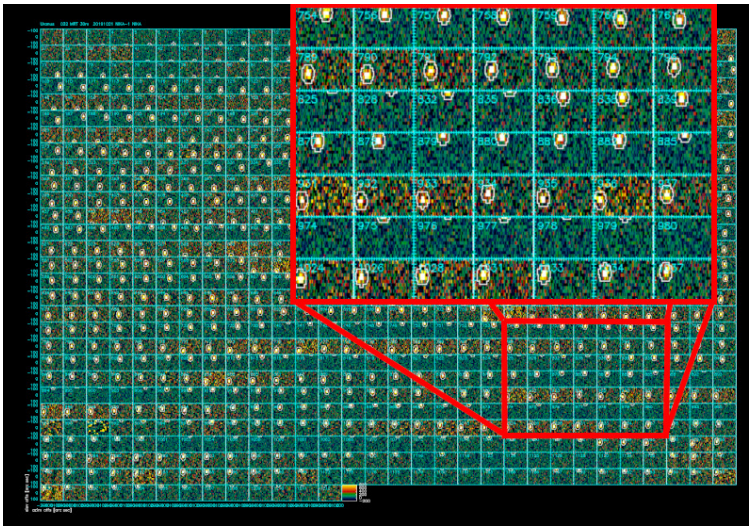


Figure 3.4: Array-1 scan maps of each individual KID, during the Uranus observation already shown in Figs. 3.2 and 3.3. The ellipse defining the area covered by the source is depicted. Those records "seeing" the source (ellipse) are not used to compute baselines.

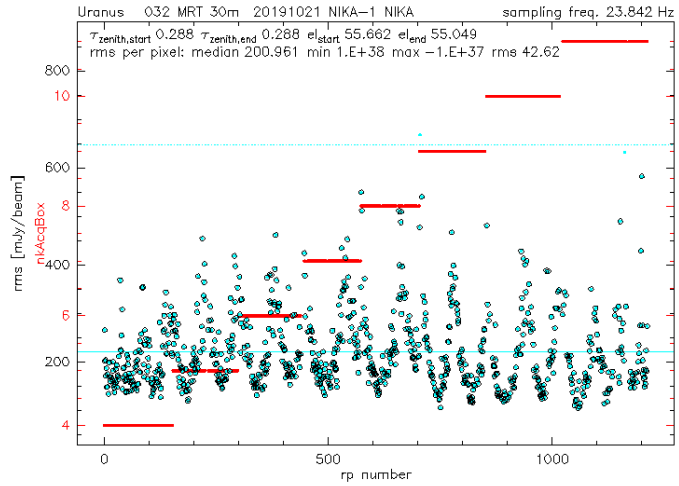


Figure 3.5: Timeline r.m.s. of all KIDs of array 1 (see also Fig. 3.3). The solid blue horizontal line marks the median value and the dot-dashed line marks a threshold set to `nOfRPrms` × the r.m.s. (corresponding to `nOfCCrms` in Fig. 3.2). As in Fig. 3.2, in red are also marked the RP id ranges belonging to the 8 electronic boxes of array 1.

3.3.1 Sky opacity along the line of sight

The absolute flux scale of the NIKA2 scans is set using observations of calibrator sources and applying the atmospheric extinction correction based on the value of the sky opacity τ measured by the IRAM Pico Veleta tau-meter (see Appendix A). This approach produces a reliable extinction correction, as the flow of tau-meter data is continuous 24h/24h, 365+ days/yr, with only few exceptions (e.g. maintenance of the instrument).

Nevertheless, the tau-meter observes the sky in a fixed direction (roughly to the West), and is mounted ~ 1.5 meters above the ground outside the control room of the Observatory. Consequently, the extinction correction comes with an intrinsic noise related to the fact that the millimetric atmosphere is not always stable and uniform across the whole 2π Celestial Hemisphere.

Starting from the 7th public release, PIIC refines the extinction correction and flux calibration evaluating the atmospheric opacity along the line-of-sight in the actual direction where the observations took place.

Because the average sky load of a scan depends on the local atmospheric conditions in the direction of observation (opacity, stability, temperature, turbulence, etc.), it can be used to evaluate the opacity and the correction along the actual line-of-sight. During the process of DAFs definition (i.e. calibration), the observed calibrators are used to define a function between the extinguished flux of the source and sky load: $\log(F_{\rm ext}) = a \times \mathrm{SkyLoad} + b$, which is stored in the NKSLEX DAFs (see Appendix A).

This function is applied to the data to refine their flux calibration. If for any reason it is not available for a given NIKA2 pool, then only the tau-meter sky opacity is used to calibrate the data.

More information about the quality of calibration and about this approach can be found in the NIKA2 PIIC Calibration documentation included as well in the PIIC package[3].

3.3.2 Information about re-gridding

The re-gridding strategy adopted as default by PIIC works in Fourier space. The kernel used to re-distribute the flux of each KID onto the final grid is a sinc function $(\sin(x)/x)$ in this space. This function is truncated at the 4th period and then weighted. Weighting involves 6 parameters, i.e. two weighing functions are used, each one having 3 parameters to be optimized. The net result is a kernel that corresponds to $\sim$43% of the HPBW.

The values of the free parameters have been optimized already in 1992. Thanks to this optimization, such regridding method reproduces the original point-like source shape to better than 1%.

The algorithm was first described by Haslam (1973) and later modified and optimized by R. Zylka, as mentioned above. Figure 3.6 shows the original concept. Appendix B analyses the effects of re-gridding on point-like sources.

3.3.3 Batch mode

For long PIIC sessions, lasting several days and involving big data sets, it might be desirable to close the terminal from which PIIC is run. In order to do this without interrupting the PIIC data

[3] file summary_PIIC_DAFs_and_figs_v*.pdf

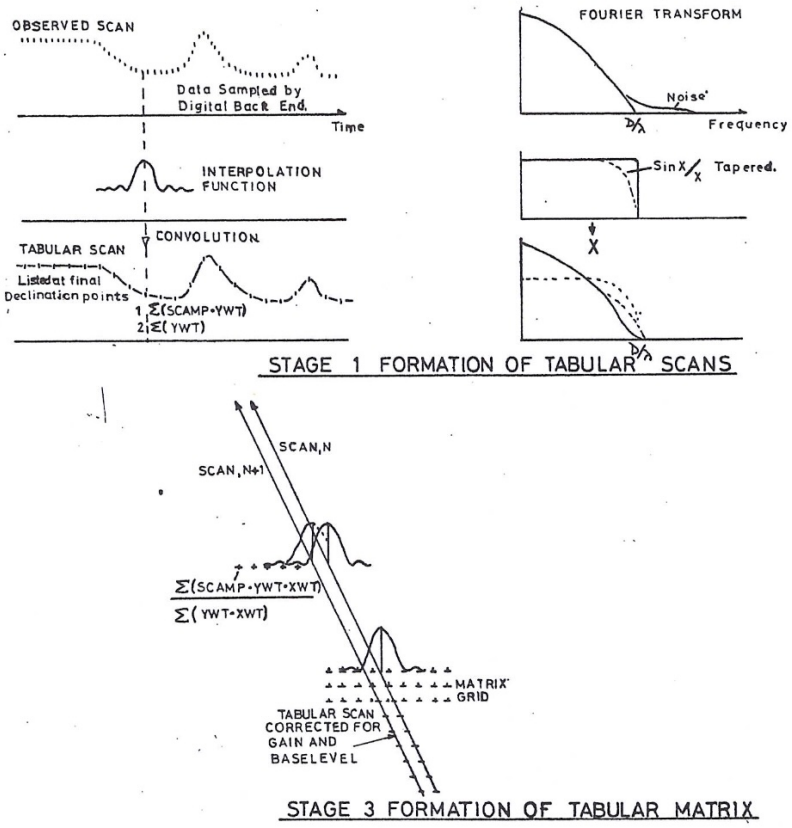


Figure 8 The process of forming the tabular scans and adding these scans into the matrix

Figure 3.6: Original re-gridding scheme by Haslam (1973), heroically found in the MPA/MPE library.

processing, it is necessary to start it in *batch mode*.

To do this, instead of starting the data reduction as described above, use the following command line from the shell:

```
piic @ reduce_my_data_Ar1 > my_batch_log_a1.log 2>&1 &
```

or

```
nohup piic @ reduce_my_data_Ar1 > my_batch_log_a1.log 2>&1 &
disown
```

or a combination of them. The actual command(s) might depend on the system setup and/or terminal type used to connect to the machine where PIIC runs or used to run PIIC.

When using the batch approach, the `pause main` command in the template script shall be commented out. Obviously the fictitious file names shall be turned into appropriate names for the user's given data set and data reduction session.

3.4 Quick analysis of the 0-th iteration results

After running the 0-th iteration, it is advisable to quickly verify that the initial assumptions were commensurate to the target and the dataset, and if necessary modify them.

The maps produced by PIIC are stored in the `outDir` directory, by default called `red/`. Here follow some examples of basic operations that can be performed on these maps.

PIIC produces three FITS files at each iteration: *1)* the signal map; *2)* the corresponding weights map; *3)* the source map, based on S/N, to be used during the next iteration, in combination with the source (ellipse or polygon) defined at the beginning. Typical filenames are:

```
1) Superantennae_a1_...0.fits
2) Superantennae_a1_...0rgw.fits
3) Superantennae_a1_...0f1.fits
```

The three dots omit part of the filenames; their content reflects the setup chosen in the data reduction script, i.e. the labels and numbers found in the filenames are basically the values of some of the key parameters of the data reduction.

As for the source map, the label `0f1` indicates that it was produced at the 0-th iteration and will be used in the 1st iteration to protect the area of sky where the source signal was detected (see Sect. 3.1).

3.4.1 Plot a map with PIIC

PIIC allows to plot any FITS map in a very simple way. First the map needs to be read and then simply displayed. In short, proceed as follows:

```
piic dev i b
read Superantennae_a1_...0.fits
plot sca lin -100 500
```

The command `plot` has several options and can display a large number of diagrams. We invite the user to get acquainted with it by consulting its help page, simply typing `? plot`.

In the simple case of displaying a map, the option `sca lin` is analogous to what found in GREG, with a slightly different syntax: it will display the map using a *linear* intensity *scale*. The two following numbers (-100 500, in the example above) represent the lower and upper limit of such scale.

3.4.2 The intensity maps

The intensity map combines the signal that hits all KIDs of all scans, excepted those KIDs and scans that were rejected during the data reduction process. In each pixel of the final grid (i.e. the final image), the intensity is the weighted mean of the intensities of all KIDs that have "seen" that given pixel

$$\frac{\sum_{i,j} s_{i,j} \times w_{i,j}}{\sum_{i,j} w_{i,j}} \tag{3.1}$$

where $s_{i,j}$ is the signal of the i-th KID of the j-th scan and $w_{i,j}$ its weight. By default, weights are proportional to the inverse of r.m.s. squared.

The intensity of each KID is distributed on the final grid according to the beam, i.e. its contribution to the intensity of each pixel of the grid implies a convolution with a proper kernel. The construction of the final map in Equatorial coordinates from the KIDS timelines of all scans is performed using the algorithm by Haslam (1973), also described by Emerson et al. (1979) and adopted in the MOPSI and MOPSIC data reduction softwares (Zylka 1998, 2013).

The top-left panel of Fig. 3.7 shows the 0-th iteration intensity map obtained for array 1 of our example Uranus case. Section 4.4 gives more information about the adopted WCS.

3.4.3 The weight maps

Along with the intensity map, the PIIC script produces the map of weights, labeled `rgw` (regular grid weight). Its value is the denominator of Eq. 3.1. Each KID of each scan enters into `rgw` with a different weight, which depends on electronic noise, KIDs instabilities, KIDs tuning, residuals of sky noises removal, beam differences, etc. Hence a possible proportionality of `rgw` to exposure time is not trivial and actually is distorted by weighting.

In order to compute a quantity that is directly proportional to the effective exposure time per pixel, one should in principle re-compute the terms (nominator and denominator) of Eq. 3.1 without weighting. Note, nevertheless, that even in such case, the proportionality constant between `rgw` and $t_{\rm eff}$ would be not straightforward to evaluate, because the contribution of each KID to the pixels of the final grid is convolved with a kernel. See Section 4.13 for more on this subject.

3.4.4 Compute r.m.s. and S/N maps

As mentioned earlier, r.m.s. maps can be computed by rescaling weight maps to the r.m.s. of the intensity map measured within a given area. This area is — for example — what is here called "the r.m.s. polygon". The PIIC `pro/` repository includes the script `creaRMS.piic`, already prepared to produce the r.m.s. map. Use it as follows:

```
@creaRMS mainName polName
```

where `mainName` is the main file name, without the `rgw` suffix, and `polName` is the polygon file name (as produced automatically by PIIC or as defined manually by the user). The result is the `mainName_rms.fits` map.

Similarly, the script `creaSNR.piic` can be used in a similar way to produce the S/N ratio map `mainName_snr.fits`.

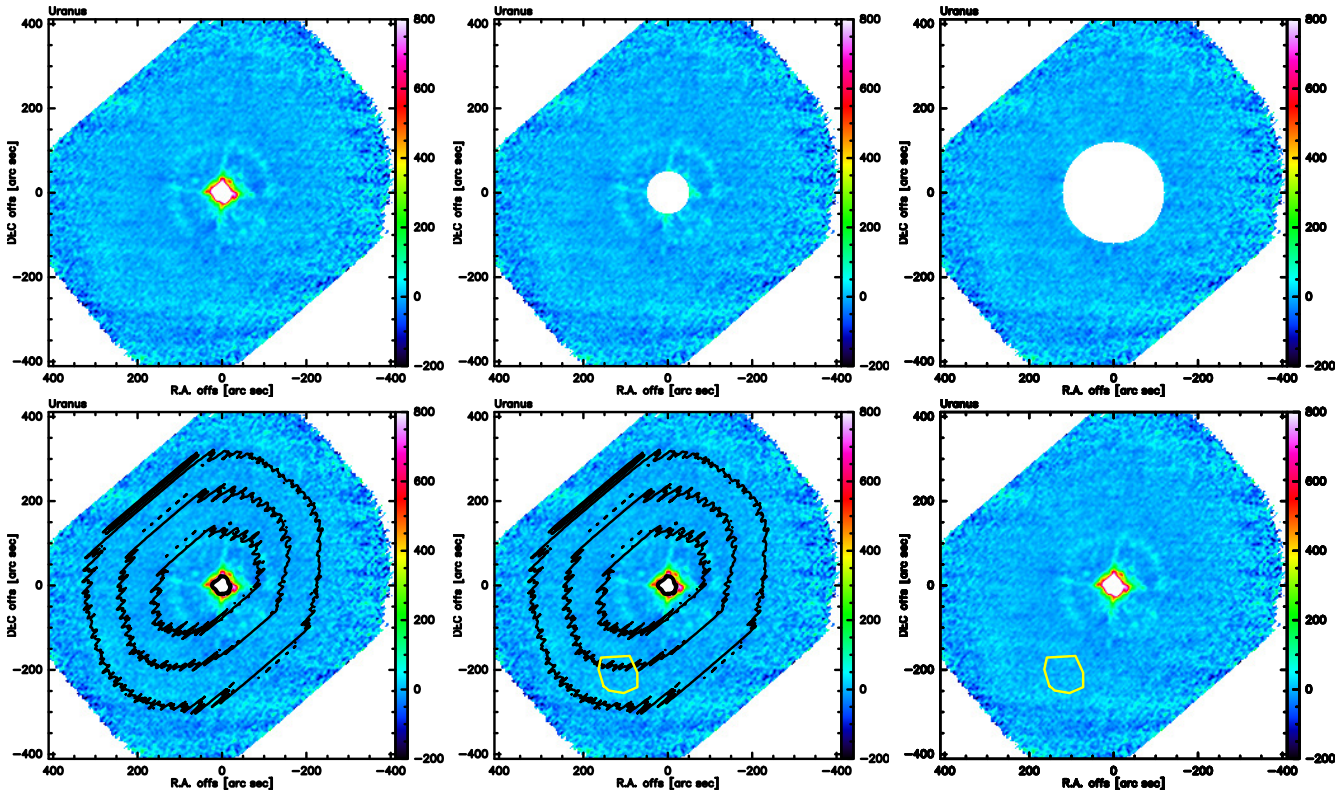


Figure 3.7: *Top-left*: array 1 intensity maps of Uranus. *Top-middle*: source area defined as a circle with radius 50 arcsec. Top-right: source definition with radius 120 arcsec. *Bottom-left*: 25%, 50% and 75% coverage levels (black lines). *Bottom-middle* and *right*: definition of the r.m.s. polygon (yellow line). See main text for details.

Manual computation of the r.m.s. map

If wished, the user can compute the r.m.s. map manually as follows:

```
read Superantennae_rgw.fits    ! read the rgw map produced for the source Superantennae
init spik /all
calc ^ 0.5                     ! take the sqrt of the rgw (note the blank space!)
pol Superantennae.pol          ! load the polygon in which to compute the r.m.s.
mean in                        ! compute the avg value of sqrt(rgw) within
                               ! the polygon
div 'MEANVAL'                  ! divide the current array by this MEANVAL
store                          ! store it in the "BACKUP" array
read Superantennae.fits        ! read the intensity map
rms in                         ! compute the r.m.s. within the polygon
take                           ! recovers what was stored before (i.e. the
                               ! BACKUP array = sqrt(rgw)/sqrt(rgw(polygon))
div 'rmsDEV'                   ! this is now sqrt(rgw/rgw(polygon))/rms(polygon)
calc ^ -1                      ! invert and obtain
```

```
                              ! rms(polygon)/sqrt(rgw/rgw(polygon))
                              ! which is nothing else than the r.m.s. map,
                              ! our final result!
write Superantennae_rms.fits  ! save on file
```

Manual computation of the S/N ratio map

The S/N ratio map can be computed manually in the following way:

```
read Superantennae_rgw.fits   ! read the rgw map produced for the source Superantennae
store rgw                     ! save it in the rgw array

read Superantennae.fits       ! read the intensity map

pol Superantennae.pol         ! load the polygon in which to compute the r.m.s.
rms in                        ! compute the r.m.s. within the polygon
calculate snr                 ! compute the S/N ratio

plot sca lin -1 5             ! plot the result

write Superantennae_snr.fits  ! save the S/N ratio map
```

3.4.5 Verify the source defined in the script

We can now verify if the definition of the source on sky, that we have adopted during the 0-th iteration was appropriate, e.g. if it includes the whole object, or if some features are missed.

In the Uranus example, initially a circular source with radius of 50 [arcsec] was used. Proceed as follows in PIIC:

```
read Uranus.fits
mask in 50 50 0      ! mask all pixels within a ellipse with
                     ! maj/min axes = 50 arcsec and PA=0 deg,
                     ! centered at (0,0)
                     ! [this specific case could also be simply written as
                     ! mask in 50 ]
plot sca lin -200 800
```

The top-middle panel of Fig. 3.7 shows the results: a 50 arcsec radius is too small for this bright object; the first diffraction ring and the signature of the tetrapod is missed. We can let PIIC "detect" it on the basis of the S/N ratio during the iterative mode, or we can enlarge the circle to 120 [arcsec] (top-right panel of the same Figure).

For more complex sources and polygonal geometries, the principle is similar, but one needs to load the source polygon and use `mask in` to verify the area actually covered.

3.4.6 The noise r.m.s. polygon

The polygon, within which the r.m.s. of the map is to be measured, is by default defined automatically. PIIC seeks for a square area on the map, with minimum size 5×5 HPBW^2 and maximum size 8×8 HPBW^2 with as low as possible noise r.m.s. In iterative mode this region might change as the processing advances, because the properties of the map change. If `nIterSource`>0, the r.m.s. polygon is re-computed at each iteration.

The polygon thus defined is saved for each iteration. Its extension is `.pol`.

Alternatively, it is possible to define the noise r.m.s. polygon also manually, and to input its file name with the parameter `polRMSeq` (in the optional settings macro). Section 4.11 describes the procedure to define and verify the r.m.s. polygon manually. It is worth to mention that, thanks to the automatic search of the region with minimum r.m.s., the polygon defined automatically produces a lower r.m.s. value than what any manually-defined polygon has done so far, even when expert hands are operating. Having the lowest possible r.m.s. produces a lower "detection" threshold and therefore a better treatment of sources' signals.

3.5 Iterative mode

After the optimization of the script and associated files has been completed, it is time to proceed with the second main step of the data reduction, i.e. the iterative loop. This time, PIIC iterates on all data reduction operations (see Sect. 3.3), each time modifying the source map based on the S/N ratio. The negative features due to poor "masking" gradually disappear.

Depending on the kind of targets observed, the number of iterations needed to reach "convergence" might vary significantly. For point-like sources, a minimum of 6 iterations and maximum of 10 are generally enough. For extended emission and complex geometries, much more than 10 iteration might be necessary. In the case of complex sources, it is not possible to predict in advance how many iterations are needed to reach convergence, because the actual number depends of the actual dataset (maps, source, instrument status, temperature, atmospheric conditions, etc.). The most difficult case is that of a strong source with faint extended structures: more than 50 iterations are needed in such objects. See also Sect. 4.14.4 for suggestion on how to enhance source detection.

3.5.1 Verify convergence

In order to verify if `nIterSource` iterations are sufficient to the project's goal, or if more are needed, one should simply compare the final intensity maps obtained with N iterations to those obtained with `nIterSource`-1 iterations only.

To do this, simply take the difference of the two. Generally speaking, if the average difference across the field is a small fraction of percent of the typical sources' flux and area, then `nIterSource` iterations are sufficient. Typical acceptance values are of the order of few percent. Each science project has indeed different needs and the users are invited to define their own "convergence" metrics. An example is shown in Fig. 4.7 (right hand panel).

3.5.2 Dynamically varying parameters

The value of some PIIC parameters does vary dynamically as the data reduction progresses through multiple iterations.

One example is the `blOrder`, but it is not an isolated case. During the first iterations, a low value is used. The polynomial order is gradually increased as the iteration number progresses, until it reaches the value `blOrderOrig` given by the user. How quickly `blOrderOrig` is reached depends on how the data reduction proceeds. The minimum number of iterations at which this happens is 5 (for example for point-like sources observed in good sky conditions), but for complex extended and faint targets it could be more.

3.6 Products

The PIIC pipeline produces a large number of files. These files are stored in the directories created at the beginning, as in Sect. 3.2.1. Here we summarize them very briefly, and we defer to specific Sections of this tutorial for further details:

- final maps are stored in the `outDir` directory (by default `red/`). Three FITS files are produced at each iteration: the signal map; the `rgw` weight map; the source map (see Sect. 3.4 and also Sect. 4.14).
- noise r.m.s. polygons produced automatically by PIIC (if `polRMSeq` is not used, see Sect. 3.4.6) have extension `.pol` and are stored in the directory `red/`.
- if requested by the user, cumulative `rgs` and `rgw` maps (see Sect. 4.13) are stored in the `outDir` directory (by default `red/`).
- files containing statistical information about each scan are stored in the `stat/` directory (e.g. see Sect. 4.9).
- the same `stat/` directory contains also files that list all scans that have been excluded from the final maps. One possible exclusion criterion is based on the r.m.s. of timelines (see Sect. 3.2.13), among others. These files have a `.err` extension.
- if the `hc` option is switched on, all plots that are displayed on the graphical device of PIIC are also printed on file and stored in the `png/` directory.

In addition, the commands given to PIIC are stored in the `PIIC.LOG#` files and the messages produced by PIIC during the execution of the commands and the data reduction are written in the `PIIC.MES#` files. The latter include the information that PIIC prints on the terminal and much more. Each new PIIC session produces a new `.LOG` and a new `.MES` file, identified by the number `#` at the end of the file's extension.

Chapter 4

Additional tips and advanced use

A well-deserved pesto from
Genoa, for the expert user!

In this Chapter we include some material for further thoughts: possible hints to improve the observing strategy; and some extra data analysis tips for the most advanced users.

4.1 Non-azimuthal scans

Scanning on the sky in a direction different from azimuth (e.g. with an angle in Equatorial coordinates) produces a quasi-periodic saw-like instability, of the order of 10's of Jy or more, mostly due to the change of airmass (a.m.) during the scan. This behaviour is not exactly periodic and the saw teeth are actually not linear. Such drifts do not correlate perfectly because each KID reacts differently to the movement of the telescope. Therefore we cannot fully correct this behaviour; we can only obtain an approximative "cleaning".

The FoV of NIKA2 is large and therefore different KIDs (e.g. one pixel at the low-el side and one at the high-el side of the array) have different airmass and cover different a.m. ranges during the scan. Therefore the saw-like quasi periodic instability is different for each KID.

Moreover the NIKA2 array(s) rotate(s) on sky as time goes by and as the scan is performed. Hence the saw-like "trajectory" (in noise space) is NOT linear (the saw teeth are not linear), but is slightly curved. The consequence of this curvature is that a linear base-line fitting approximation can cause over-/under-subtraction of instabilities, thus producing spurious negative signals on one side and spurious positive signals on the other side of the "tangent point" to the curve, where the linear fit is pivoted.

Using an iterative procedure (see, e.g., Sect. 3.1), source "protecting" is adapted and optimized. By doing this, the negative signal is avoided and corrected, but the fake positive signal cannot be removed and will stay until the end. Hence a good evaluation of the source map (i.e. those regions of the sky that contain a source and must be protected) at the very beginning is critical, in order to avoid producing spurious effects on the final map.

4.2 Scans with different centers

When large fields are observed, or different structures of complex sources are targeted, it is possible that the scans to be combined together have different centers in the sky. In this case PIIC does not know a priori where to position (center) the pixel grid on which the final map will be built.

It is therefore necessary to define the (RA, Dec) position of center of the final map using the parameter `posSeq`. Its value must be in the format "H M S D AM AS".

Similarly, the extension of the final map must be defined by setting `eqExtrAS` in units of [arcsec].

4.3 Flux units

The maps produced by the PIIC data reduction are in units of [mJy/beam]. Flux calibration is based on the DAFs CAL files, which are based on observations of standard flux calibrators, e.g. (primary calibrators) and a number of other sources (secondary calibrators). The calibration factors are obtained fitting the main beam (MB) with a Gaussian profile.

Let's say that one would like to transform the map in different flux units, for example in [MJy/sr], which is another typical unit used in the literature, for example to build color maps with Herschel.

The equivalent source angle under a Gaussian profile, in units of [arcsec^2], is $2\pi\sigma_x\sigma_y$, where $\sigma_{x,y}$ is expressed in [arcsec] and is related to the FWHW of the Gaussian by the well known relation $\sigma = \mathrm{FWHM}/\left(2\sqrt{2\ln(2)}\right)$. In case of a circular Gaussian profile and point-like sources, we can write $\mathrm{FWHM}_x = \mathrm{FWHM}_y = \mathrm{FWHM} = \mathrm{HPBW}$.
Therefore, expressing the HPBW in units of [arcsec], we obtain:

$$[\mathrm{mJy/arcsec^2}] = \frac{[\mathrm{mJy/beam}]}{2\pi\sigma^2} = \frac{[\mathrm{mJy/beam}]}{\frac{\pi\times\mathrm{HPBW}^2}{4\ln(2)}} \tag{4.1}$$

Converting to steradians, one obtains[1]:

$$[\mathrm{MJy/sr}] = \frac{[\mathrm{mJy/beam}]}{\frac{\pi\times\mathrm{HPBW}^2}{4\ln(2)}} \times \frac{(206264.8062)^2}{10^9} \tag{4.2}$$

Additionally, if units of [mJy/pixel] are wished, the pixel size in [arcsec^2] needs to be taken into account. The standard PIIC choice is to have 3×3 arcsec^2 pixels at 1.2 mm and 4×4 arcsec^2 at 2 mm. For example, at 1.2 mm one obtains:

$$[\mathrm{mJy/pix}] = \frac{[\mathrm{mJy/beam}]}{\frac{\pi\times\mathrm{HPBW}^2}{4\ln(2)}} \times \frac{1}{3\times 3} \tag{4.3}$$

The size and shape of the NIKA2 beam depend on a several factors, some intrinsic to the instrument, some related to the observing conditions:

[1] the number of arcsec in one radian is $3600 \times 180/\pi = 206264.8062$.

- the focal surface changes from the centre to the edges of the arrays, e.g. at the frequency of 1 mm this variation is of ∼0.8 mm; this results in different beams at different location across on one given array;

- the three arrays are not perfectly at a common "average" focus, e.g. array 1 and array 3 are off by > 0.1 mm; this results in different beams on different arrays;

- day and night observations are taken under different environmental conditions; for example during the day the main dish is under direct sun illumination, which heats it and induces distortions on the primary surface of the telescope;

- the coating of the telescope dish evolves with time, thus so do also the environmental effects on the beam.

The ideal beams — defined under perfect conditions (telescope, instrument, atmosphere, ... — have HPBW ~ 11.3 arcsec at 1 mm and ~ 17.7 arcsec at 2 mm, but these widths are never observed, even for really good sky conditions. Therefore more conservative values of 11.6 and 18.0 arcsec are used here, as measured for average good conditions (see for example Fig. 4.3).

4.4 WCS

The coordinates of the pixels of the final maps produced by PIIC are defined as true angles on a sphere, counted as offsets from the reference position (the center of the map). These angle coordinates are basically given by the Antenna Mount Drive of the IRAM 30m telescope and do not undergo any further projection during the map making process of PIIC.

There is a conceptual difference here between a "traditional" 2D image (e.g. taken all at once with an optical camera) and a beam (the telescope) moving in the sky (the NIKA2 OTF scans) to be then used to reconstruct the map of a given field in spherical sky coordinates. In other words, we are dealing with a mathematical table of coordinates (spherical angles relative to the map center).

Being based on offsets, the world coordinates system (WCS) of the final maps include already the $\cos\delta$ term. Schematically:

$$x = (\alpha - \alpha_0) \times \cos\delta \quad (4.4)$$

$$y = \delta - \delta_0 \quad (4.5)$$

In WCS FITS files conventions, this representation is a pseudo-cylindrical, sinusoidal, equal area projection (GLS, also known as "radio" in GILDAS). See Calabretta & Greisen (2002) to know more details on its definition and its subtle difference with respect to the proper SFL (Sanson-Flamsteed) projection.

The absolute coordinates system for sidereal targets is Equatorial FK5 J2000. On the other hand, the Epoch of the Equatorial coordinates of moving targets (e.g. Planets, Comets, etc.) is the epoch of observations, because in 2000.0 they would have simply been at a different position in the sky.

4.5 Define your source

During the data reduction, sources need to be excluded from sky noise computation. The *iterative* mode automatically accounts for this but in some cases a manual initial definition of the source might help to converge or to improve the results. The parameters related to the manual source definition are found in the `mapTPoptionalSets.piic` script.

At best, the data would have already been gone through the QL (see Chapter 7), so to have a first initial guess of where the source lies, in order to be able to have a first guess of the region of sky it covers. Note, however, that the QL does not combine maps together, hence if the source is faint, it won't be seen in the QL results yet.

A source can be defined as a circular/elliptical or a polygonal shape. Such source area is defined on the final map, i.e. on sky coordinates, using GILDAS/PIIC conventions: the (0,0) point is at the center of the map; units are [arcsec][2]; x-axis values increase to the left and y-axis values increasing towards the top, i.e. as in a RA-Dec map with East to the left and North up. The position angle is defined starting from the y-axis, in the counter-clockwise direction. Figure 4.1 depicts these conventions.

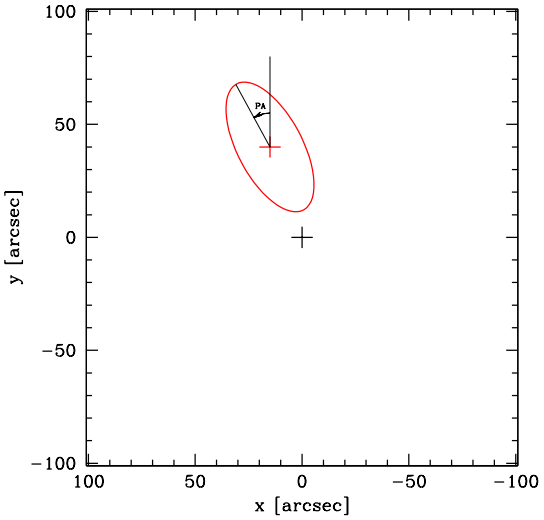


Figure 4.1: Scheme of GILDAS graphical conventions. The pixels coordinates origin (0,0) is at the center. Coordinates are defined in units of [arcsec] as in Equatorial coordinates (North is up, East is to the left). Position angles are defined starting from the y-axis, in the counter-clockwise direction. In this example, an ellipse centered at (15,40), with major and minor semi-axes equal to 15 and 32 [arcsec], and position angle of 30 degrees was defined.

When only a simple source is present in the maps, a simple circular/elliptical patch covering it all could be the easy and correct way to proceed. The parameters `souRmxAs` and `souRmnAs` define the major and minor axis of an ellipse defining the source on sky, in arcseconds. The position angle of

[2]Note that some GILDAS component have [radians] units default. Therefore it is recommended to use the `polygon` command within PIIC.

the major axis, with respect to the vertical axis of the image (North in Equatorial coordinates) is given by `souPA`.

If the parameters `souRmxAs` and `souRmnAs` are set to 0.0, then simply no source patch is defined, nor used to exclude sky areas. This option is used, for example, when adopting the iterative mode with no initial source guess.

For a proper use of elliptical source patches, the `deepfield` needs to be set to "*no*" (see Sect. 3.2.7).

Although conceptually very simple, defining the source correctly might not be always trivial. Such an ellipse should not be too big, in order to leave enough KIDs for a proper computation of sky noise. This means that the ellipse should certainly be smaller than the array size. Note that — however — this simple precaution might not be enough to produce a reliable science-quality final map, e.g. if the final map is small or if the sky conditions were particularly unstable, or if KIDs were not well tuned. It is therefore recommended to experiment with different ellipse sizes and geometries, if the results look doubtful.

A sub-optimal definition of the source patch can generate spurious features in the final map. It can produce "fake" sources, both positive and negative. If a negative spurious signal is produced (e.g. a negative lobe), and if the iterative method is used, then the adaptive protection of positive sources based on the S/N ratio can gradually make it disappear. On the contrary, a positive spurious signal is detected by the iterative process and recognized as a source. Hence it is much more difficult to identify a positive spurious signal. If a positive spurious signal is produced by a improper source definition, then it will persist on the final map, no matter how many iterations are employed. Finally note, that even negative spurious sources can survive the iterative process and can end up on the final product.

The overall message here is that — although conceptually simple — defining the source area to be excluded from computations is a delicate process and its complex consequences should not be underestimated.

A typical data reduction performs the 0-th iteration using a first-guess ellipse, for then verifying on the product map if the initial choice was properly designed or if it needs to be modified, optimised, or turned into a polygon. With an improved source definition, the reduction proceeds either directly in the iterative mode, or repeating the 0-th step.

Keep in mind that — for example — a 10-iterations process on a typical (small-ish) map of a Galactic cloud (SF region) can take approximately 24h to process. Hence it is better to spend some more time defining the source region and optimize it, rather than finding out later that it was not correctly defined and thus wasting a 24h time.

4.5.1 Offsetting the source patch

In some specific cases, it might be necessary to apply an offset to the ellipse that defines the source. An offset might also be relevant for a polygonal region, but one could imagine that polygons are defined directly at the position of the source.

The parameters that define the offset of the ellipse with respect to the center of the final map are called `souRAoffAS` and `souDECoffAS`. If all maps are centered at the same coordinates (i.e. all scans where observed using the same scan center), then defining `souRAoffAS` and `souDECoffAS` is

sufficient.

On the contrary, if the scans to be combined have different scan centers, then PIIC does not know a priori where the final map will be centered, and the central coordinates of the final map must be given manually defining the parameter `posSeq` (in Equatorial coordinates). In this case, the offsets are computed with respect to the position given by `posSeq` itself.

Summarizing, six parameters need to be used in order to entirely define the size, shape and position of the elliptical source patch:

```
souRmxAS     ! size of major half-axis in arcsec
souRmnAS     ! size of minor half-axis in arcsec else
souPA        ! source position angle on sky, in degrees (equatorial conventions)
posSeq       ! equatorial coordinates of the center of the final image,
             ! in the format "H M S D AM AS"
souRAoffAS   ! R.A. offset of the source relative to posSeq (in arcsec)
souDECoffAS  ! Dec  offset of the source relative to posSeq (in arcsec)
```

4.5.2 Polygonal source definition

A polygonal source definition follows the same principle as the ellipse, but is defined ad-hoc to cover sources with irregular or complex shapes. It can be easily defined using the GILDAS command `polygon`. The polygon needs to be written in an ascii file, listing in two columns the coordinates (x,y) of each vertex of the polygon.

Importantly, the vertices sequence must be clockwise or counterclockwise, but not mixed. This means that one should not mix the vertices, neither make intersections while defining the polygon, because otherwise its "inside" and "outside" region might get inverted (see GILDAS tutorials[3]).

Remember that, if multiple objects are present, it is necessary to define a single polygon covering them all. This can make intersections difficult to avoid if the geometry is very complex. All we need is just a little patience.

Finally, also for a proper use of polygonal source patches, the `deepfield` needs to be set to "*no*" (see Sect. 3.2.7).

The filename of the polygonal source definition is given in input as parameter `polSeq`. This is to be found in the script `mapTPoptionalSets.piic` (see Sect. 3.2.9).

4.6 Baseline correction and base range

Sources need also to be excluded from the computation of baselines, in order to avoid on one side to influence the baseline subtraction and on the other to alter the flux and the profile of the sources themselves.

Similarly to the case of sky noise computation (see Sect. 4.5) the *iterative* mode accounts automatically for this, but a manual definition of the region to avoid can help to improve the results.

[3]Note that some GILDAS components adopt [radians] units, while PIIC uses [arcsec]. Therefore use the `polygon` command from within PIIC.

The parameters that control the size and shape of the region to avoid during the baseline computation are called *base range* parameters: `brRmxAS`, `brRmnAS`, `brPA`, and `polBReq`. The first three parameters define an ellipse in the usual way; the fourth defines, in alternative, a polygon.

It is worth to note that these parameters are by default set to be equal to those controlling the source shape and size (Sect. 4.5) in the `mapTPoptionalSets.piic` script, where they are set to zero.

In the case of simple or compact strong sources, if the source parameters are set in the main `mapTP_a2_template.piic` script after the call to the optional sets, it is recommended to also set the *base range* parameters in the same place. For simplicity they can be defined as the source parameters, but it is also advisable to optimize them independently. The source size is limited by the field of view of NIKA2, while the base range is limited by the size of the maps (scans).

4.7 Effects of sky noise removal and of instabilities on extended and diffuse emission

When dealing with faint extended sources or diffuse emission, the effects of sky noise removal and of the NIKA2 instabilities can play a non negligible role. In the former case, the retrieved flux and source profile can be affected. In the latter, the signal can be suppressed or enhanced.

4.7.1 Faint extended sources

For extended sources — smaller than the NIKA2 FoV — an ellipse or a polygon are used to define the source boundaries in the sky. The records with coordinates inside these boundaries are not used during the subtraction of sky noise.

It is possible that the source profile extends beyond the boundaries defined by the ellipse or the polygon, still being well contained in the FoV of NIKA2. In this case, its emission outside the source boundaries is recovered during the iterative processing, if it is above the S/N threshold defined by `snrLevel` (found in the optional settings script). On the other hand, if it's too faint then it is lost when subtracting the sky noise.

A smoothing of the source map can be applied (`smSNRpar`; see Sect. 4.14.4) to help detect more extension of the sources[4]. This is highly recommended for faint extended targets.

Figure 4.2 shows the fraction of recovered flux, as a function of the source radius. A circular source was defined in the data reduction setup, i.e. r =`souRmxAS=souRmnAS`. A `blOrderOrig`=2 was used. The smaller the radius of the circle used to isolate the source emission, the larger is the fraction of flux lost due to sky-noise subtraction. Note that this is not an artificial source. The curve is normalized to 100% at 120 arcsec radius. This is just an example limited to a given intrinsic size of the source, a given flux and a given noise r.m.s. reached by the observations. If any of these factors changes, the curves in Fig. 4.2 change. The users shall compute their own curves, if needed by their analysis (see also Sect. 4.15).

In reality, some flux is also lost within the radius r, because the real profile of the source is not a sharp box. How much inside r this effect extends, depends on the actual profile of the object.

[4]only the source map is smoothed, the data are not.

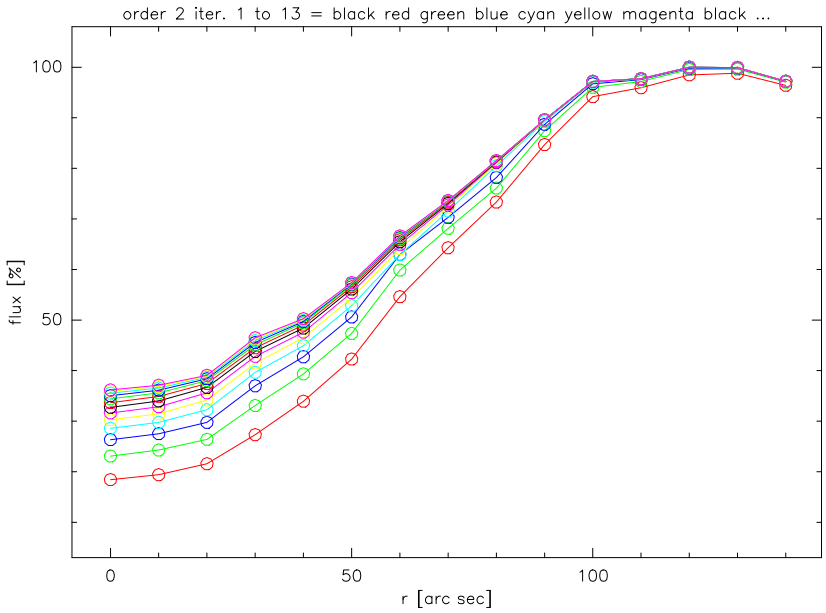


Figure 4.2: Fraction of flux recovered for an extended faint source, as a function of the source radius defined by `souRmxAS=souRmnAS`. Different lines (colours) refer to a different iteration, from 1 to 13. A `blOrderOrig`=2 was used.

The fraction of recovered flux is exactly =1 *only* if the source is *entirely* contained inside the circle. To be precise, this is true only if the median calculated using `nBest` records outside the source boundaries is not polluted by the source signal (see Sect. 3.2.11). In practice, ∼100% of the flux is recovered, if the source is smaller than $r + x \times$ HPBW. Here, x is a constant that depends on the profile shape, and whose value is usually between 1 and 2.

Similar considerations can be made for elliptical or polygonal shapes.

4.7.2 The effect of NIKA2 instabilities

Inside the boundaries of the source area (defined either by an ellipse or a polygon), a second effect plays an important role: NIKA2 instabilities, related to the atmospheric condition, the observing strategy, the reply of KIDs to the sky load, and so on.

Instabilities can decrease or increase the source flux, depending on the order of baseline corrections, and on the location of the source in the map (i.e. if it falls onto an instability "valley" or a "hill").

Under strongly unfavorable conditions, even spurious signals can be generated (see Sect. 2.1).

The map size, the scanning strategy, and the choice of blOrderOrig are key factors to be carefully defined at the observing stage and when using the PIIC pipeline.

4.7.3 Uniform emission

Very extended, diffuse emission — which can extend beyond the size of the NIKA2 FoV — is suppressed by the sky noise subtraction. This suppression extends also inwards, inside the source ellipse

(or polygon), because a common "background" is subtracted, and it includes the diffuse emission too.

A straightforward way to show this effect is to study an almost uniform signal: the beam of NIKA2, extended over large scales.

Observations of Mars were taken in excellent conditions: the source was ∼1000 [Jy], and any sky effects were negligible in comparison. This allowed to produce the beam without any sky-noise subtraction. Figure 4.3 compares the NIKA2 beam obtained in this way to the one produced subtracting the sky-noise (see also Sect. 4.3 for notes about the possible effects that can affect the NIKA2 beam).

When applying the sky-noise subtraction, a circle of radius 190 [arcsec] defined the source in the sky. Remember that the radius of the NIKA2 FoV is ∼200 [arcsec]. Using such a large source radius was possible only thanks to the special conditions of these observations.

The suppression of the uniform/diffuse emission are clear.

4.8 Noise of the final map

The final map comes with an associated noise r.m.s. map, that can be produced following the instructions given in Sect. 3.4.4. The noise r.m.s. map is not uniform and its value can show large variations across the field observed with NIKA2.

The outer regions (or edges) of maps are always noisier than the central parts, because they are covered by a smaller number of KIDs while scanning in the sky. In low number conditions the usual law of statistics are not a good representation of the noise, and the r.m.s. at the edges can be thus erroneously measured. For this reason, scaling the r.m.s. from the central to the outer regions of the map on the solely basis of the lower exposure (e.g. lower weights) can lead to an erroneous (underestimated) noise prediction at the edges.

For the same reason, associating one single value of the noise r.m.s. (or depth) to an observed map does not properly characterize the map nor the performance of the instrument during that specific data set.

4.8.1 Noise r.m.s. across the map

Figure 4.4 presents three examples, built for three different data sets that observed areas of different sizes in the sky: a very large field, observed in two different scan directions parallel to the Equatorial coordinates; an intermediate-size field observed in two directions rotated with respect to the coordinate system; and a small field observed at multiple scanning directions.

The bottom panels of the Figure represent the noise r.m.s. map, normalized to its deepest level (at the center of the map). The top panels show the distribution of the normalized noise r.m.s. along $\delta\,(\mathrm{Dec}) = 0.0$ (i.e. along the x-axis).

As can be seen, the noise r.m.s. literally "explodes" in the outer regions of the maps, increasing by up to a factor much larger than 10 at the edges. In the case of smaller maps (central panel), the region with enhanced noise has a larger impact in relative terms: the area with "flat" r.m.s. value near the central minimum covers a smaller fraction of the map. In the case of the smallest map, the central region with minimum r.m.s. nearly becomes curved and the flat area almost disappears.

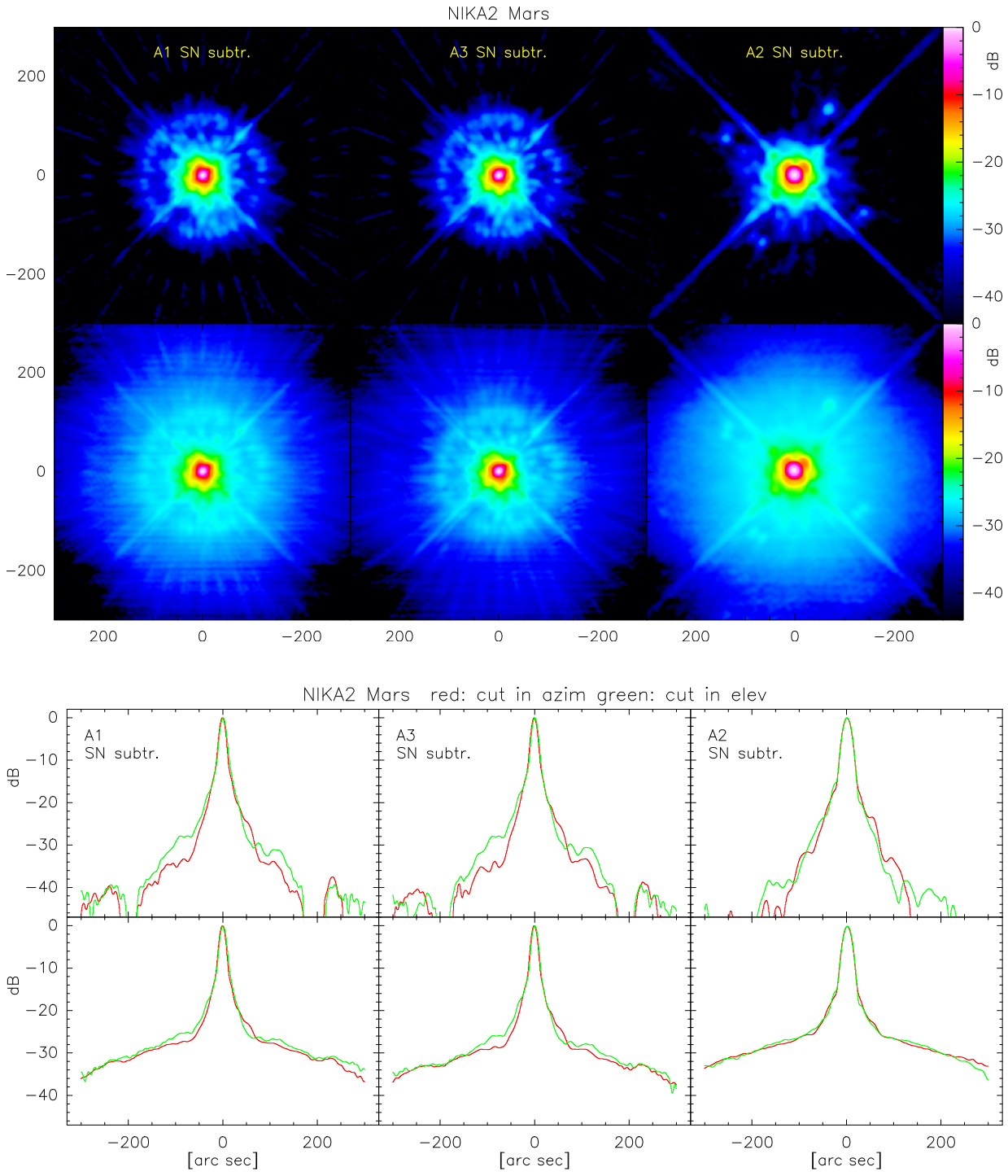


Figure 4.3: NIKA2 beam, based on Mars observations. The *top* maps and profiles show the beam as obtained with sky-noise (SN) removal. The *bottom* ones without it. The *left, middle, right* panels belong to arrays 1, 3, and 2, respectively. The profiles were obtained cutting the 2D maps along the central line and column (red/green lines).

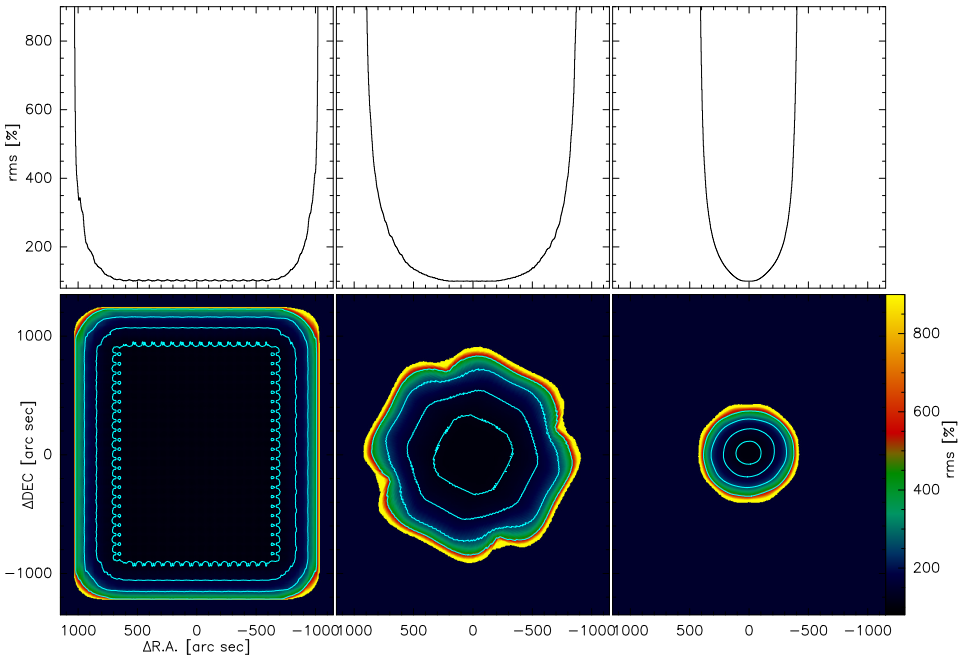


Figure 4.4: Noise r.m.s. across maps of different sizes. *Bottom* panels: r.m.s. maps built as described in Sect. 3.4.4 and normalized to the maximum central depth. The contours are traced at 102%, 150%, 250% and 500% of the central minimum r.m.s. *Top* panels: distribution of the noise r.m.s. across the map, along the x-axis, as measured on the central lines of the maps.

If the map size would be smaller than $\sim 2\times$ the diameter of the field of view, the areas with r.m.s. larger than the minimum would start to overlap and the r.m.s. profile would become completely curved. In this case there would be no region of the final map that observed by all KIDs and therefore no region would reach the full nominal depth.

4.8.2 Trimmed edges

At the very edges of the map the r.m.s. tends to diverge. Moreover, because gridding is performed using the sinc function, the weights map at the edges become negative.

PIIC trims the edges of the final map, blanking out all those pixels for which the weight is $< 0.3\,|\texttt{rgMn}|$, where `rgMn` is the minimum weight value of the map (always negative if the size of final map is equal or larger than that of the area covered by the scans).

4.8.3 Excluding the outer regions from the computation of the source map

The source map is the result of a S/N thresholding of the signal map produced at the i-th iteration using the optimized r.m.s. polygon. This polygon is usually chosen automatically to minimize the noise r.m.s. itself. It is therefore possible that the enhanced noise in the outer regions of a map is identified as a source by the thresholding procedure.

Moreover, note that the r.m.s. noise measured on the map is dominated by high spatial-frequency noise, but the so-called "baseline instabilities" are low-frequency noise. In other words, the high-frequency noise determines the detection threshold in defining the source map, and the low-frequency noise may be detected as if it were a source. Also this effect is enhanced in the outer regions of scan maps.

In both cases, such sources are obviously spurious and shall be discarded. One way to do this is to define a polygon or a radius, outside of which the source map will be set to zero. These are addressed as *zeroing* polygon or radius, and are controlled by the parameters `polZmEq` and `rZmEq` defined in the main template script.

Some consequences of adopting different values of the zeroing radius are described in Appendix C.

4.9 Noise of individual scans

In addition to the final products and the PNG images of intermediate diagnostics, the PIIC data reduction script also saves numerous files containing data statistics, as measured during the data reduction flow and at the end of it. These data files are stored in the directory `stat/`.

As an example, one quantity of special interest is the noise of the individual scans before and after applying the atmospheric opacity correction. This can be plotted using the GREG script `plTODrms.greg`, included in the `pro/` folder under the PIIC installation directory. Proceed as follows:

```
@plTODrms Superantennae_..._ECend ! [continue when it pauses]
```

One example obtained with a large dataset of more than 100 scans is shown in Fig. 4.5 for Array 1. In green is depicted the noise of the scans, before applying the opacity correction (and after sky-noise subtraction). It is flat, denoting how the instrumental noise kept constant during the whole observations and across different runs. In red the noise after opacity corrections is plotted. The behaviour of noise now reflects the sky conditions, with peaks for those scans observed in poor conditions.

4.10 Number of KIDs effectively used

During the data reduction, some KIDs can be excluded on the basis of different criteria. Some examples are: noise along the timeline; cross-talking; sub-optimal tuning; etc. PIIC stores the number of KIDs rejected for different reasons in `stat/` directory.

The GREG script `plNurps.greg` — included in the `pro/` folder, as usual — can be used to draw a diagram of the number of KIDs left in each scan, and that were effectively used to produce the final map. Identify the files with extension `.nUrps` and use the one belonging to the latest iteration. Proceed as follows:

```
@plNurps Superantennae_..._i5n5 ! [e.g. for the 5th iteration if no input source
                                ! i.e. SbSource=" "]
@plNurps Superantennae_..._i5s5 ! [e.g. for the 5th iteration with input source]
```

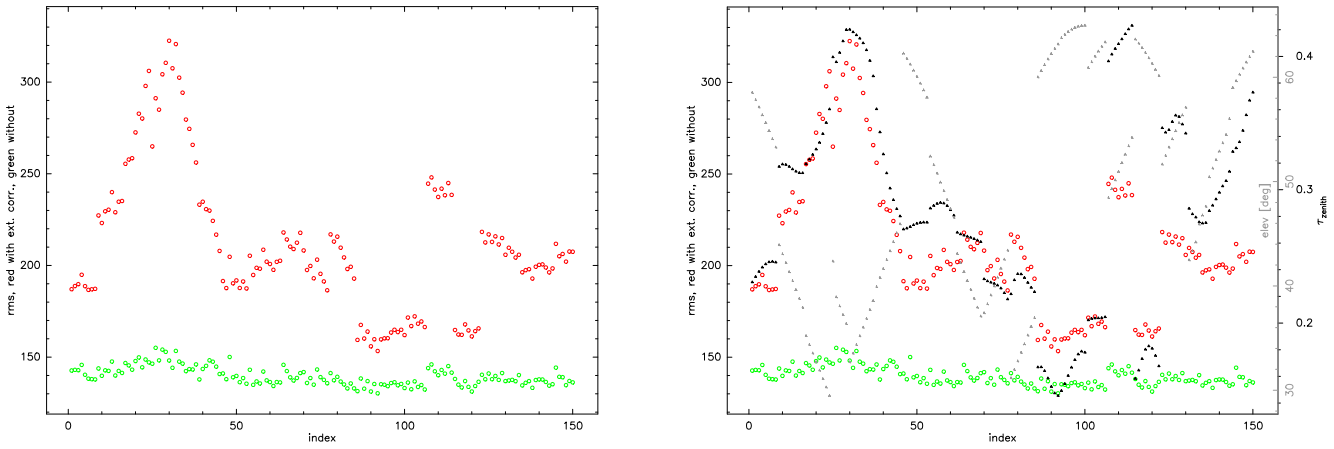


Figure 4.5: Array 1 noise of individual scans during observations. Green points depict the noise before applying the tau-correction. Red dots are the noise after the opacity correction. In the *right* panel, elevation and opacity information are plotted as grey and black symbols, respectively (see right-hand y-axes).

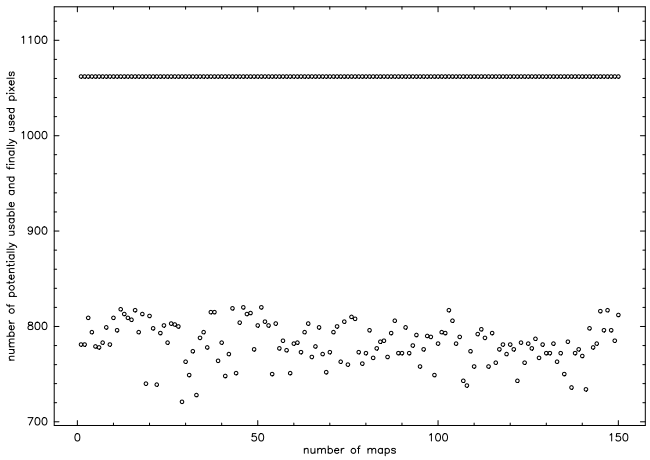


Figure 4.6: Number of KIDs that effectively made it to the final map, as part of a large dataset with more than 150 scans. For each scan, two quantities are shown: the total number of available KIDs defined by the sweep (note that this is not the number of hardware KIDs; top sequence); and the number of KIDs that survived the different selection criteria (bottom sequence).

```
! [in both cases omit the nUrps extension:
! it's added automatically]
```

One example, obtained with the same large dataset as before, is shown in Fig. 4.6 for Array 1. For each scan, two quantities are shown: the total number of available KIDs (named "potentially usable" on the y-axis label) and the number of KIDs effectively used to produce the final map.

4.11 Manually define, verify and optimize the noise r.m.s. polygon

The polygon, within which the r.m.s. of the map is measured, is by default defined automatically by PIIC (see Sect. 3.4.6). Alternatively, it can be also defined manually by expert users and given to PIIC as input. The parameter to be used in this case is found in the `mapTPoptionalSets.piic` macro (see Sect. 3.2.9) and is called `polRMSeq`.

The chosen area should ideally have deep enough coverage to be representative of the goal of the project and to allow excluding the presence of any source. In the case of large maps with very inhomogeneous coverage, this choice can turn out to be quite critical in properly defining the source area and in avoiding the creation of spurious signals.

Ideally one would like to define the r.m.s. polygon on the final map, in order to exclude possible sources from it, using the maximum possible S/N ratio. The result of the 0-th iteration can be used.

4.11.1 Produce the noise r.m.s. polygon

The polygon file is simply an ascii file, consisting of two columns: x and y of the polygon vertices (see the GREG commands help). The coordinates are in "world" units [arcsec], following GILDAS conventions[5] (see Fig. 4.1 and related text). Knowing these rules, users can define the polygon using their favorite piece of software. Nevertheless, we find convenient to use PIIC/GREG themselves, since here the coordinates definition is completely transparent.

Display the final map with PIIC and use the GREG command `polygon` in interactive mode (just type "polygon"). Click on the image where you wish to define vertices, taking care to proceed in clockwise or counterclockwise direction, and without crossing paths (see also Sect. 4.5). The coordinates are written on screen. Exit pressing "e" and save the coordinates on file or using the command `write pol filename` (with `.pol` default extension, added automatically).

To define the polygon in a more refined way, one might like to help the hand by visualizing the N% coverage levels on the map. In this way, one could define the polygon within an area of high coverage (or within a desired coverage range). All this, translated in PIIC, looks like the following:

```
read Superantennae.fits          ! reads the intensity map
plot sca lin -1 5                ! plots it with linear scale within the
                                 ! indicated range
store                            ! saves it in a backup array
read Superantennae_rgw.fits      ! load the weights map
store rgw                        ! store it in the rgw array
put                              ! transfer the rgw data from PIIC to GILDAS
levels 90 95 ...                 ! define the 90%, 95%, ... levels
rgmap /percent 1                 ! draw 90%, 95, ... coverage contours
polygon Superantennae /plot      ! load and plot an existing polygon (.pol)
polygon                          ! use pol command in interactive mode to
                                 ! define a new one in the desired area
```

[5] but *not* in [radians]!

```
greg\write pol Superantennae     ! write on (.pol) file the new polygon
                                 ! add the /overwrite option if the file
                                 ! already exists or use a different file name
```

The three bottom panels of Fig. 3.7 show the definition of the polygon as described here.

4.11.2 Verify the polygon

Let's verify that the area we have defined to be *inside* the polygon is effectively interpreted also by GILDAS to be *inside* the polygon. There could be different reasons for which this might not happen, for example if paths were crossed while defining vertices (easy to happen if complex geometries are involved), the direction was inverted, etc.

The procedure is simple: load the image; display the image; load and display the polygon; mask all pixels within the polygon; re-plot the image and visually verify that GILDAS actually masked those pixels that we meant to mask.

The same, translated into PIIC/GILDAS commands is:

```
read Superantennae.fits
plot sca lin -1 5
polygon Superantennae.pol /plot
mask in
plot sca lin -1 5
```

4.11.3 Optimize the polygon a posteriori

The basic principle on which we base our polygon verification is that the r.m.s. of the S/N map, in an empty area (i.e. without sources) in presence of white noise only, must be equal to 1.0.

In order to verify this, we need to produce the S/N map out of the intensity and weights map produced by PIIC (see Sects. 3.4.2 and 3.4.3).

The weight map is proportional to the inverse of the r.m.s. squared. In order to find the correct normalization and transform it into a r.m.s. map, it is necessary to measure the average value of the weight maps and the r.m.s. of the intensity map within the same area (polygon) free of sources. The rest is simple arithmetics.

Produce the S/N ratio map using the script `creaSNR.piic` (see Set. 3.4.4). Then measure its r.m.s. in different areas (polygons), defined where there are no sources. In this way we verify if the r.m.s. value is on average close to unity. If so, then the r.m.s. polygon is defined correctly. If not, the polygon needs to be re-defined. Let's do it in PIIC:

```
read Superantennae_snr.fits
plot sca lin -1 5
pol            ! yes, all commands can be abbreviated
               ! define a polygon somewhere appropriate
               ! press ``e''to exit
rms in
```

```
pol           ! do the same at different locations, several times
rms in        ! do the same at different locations, several times
```

4.12 Scans of very different length

For some particular source geometries (e.g. long filaments), the user might have defined scans and cross-scans with very different lengths. In this case it is necessary to use different orders of the baseline fitting polynomials, because the data instabilities might have different spatial frequencies.

The parameter `blScale`, found in the `mapTPoptionalSets.piic` script, is used to this aim. If its value is larger than 0.0, the baseline polynomial order is modified such that:

$$\text{blOrder} = nint\left(\text{blOrderOrig} \times \frac{(x_{\max} - x_{\min})}{\text{blScale}}\right) \tag{4.6}$$

where *nint* represents the closest integer, and $(x_{\max} - x_{\min})$ is the length of the given scan in units of arcsec. If `blScale` is set to 0.0 (i.e. its default value), no scaling of the baseline order is performed.

4.13 Cumulative signal and weights

For some project and science cases, it might be of interest to build — in addition to final maps containing all scans in the dataset — partial, cumulative maps. Given a list of M scans, partial cumulative intensity maps are obtained combining the first two, then three, then four, and so on scans of the list, up to M. Similarly, the associated cumulative weights can be produced.

We name these partial-depth maps cumulative `rgw` (regular grid weights) and `rgs` (regular grid signal). In practice, they represent the denominator and the nominator of Eq. 3.1, respectively.

Writing the "cumulative" `rgs` and `rgw` for each scan is switched on by setting the parameter `wrCumMaps` to "yes". This parameter is found in the `mapTPoptionalSets.piic` script. The cumulative `rgs` and `rgw` are written in the `red/` directory for each scan, i.e. up to that given scan.

This option cannot be combined with the iterative mode, because when `wrCumMaps` is switched on, the final maps are not produced and therefore also the corresponding source maps are not available. Understandably, one might want to write the cumulative `rgs` and `rgw` maps only for the last iteration. To do this, proceed as follows:

- perform the data reduction with the iterative mode as usual and produce the final signal, weight and source maps. At this stage keep `wrCumMaps`="no".
- switch `wrCumMaps` to "yes" and perform a second run of PIIC, limiting to the 0-th iteration only. Use the source map produced at the last iteration as `sbSource` input (see Sect. 4.14);

Once the `rgs` and `rgw` cumulative maps have been produced, the intensity map — in units of [mJy/beam] — corresponding to the m-th `rgs` is computed as follows:

```
read Superantennae_i_rgw.fits
calc extre          ! shows map min/max values
```

```
< '0.3*abs(rgMn)'   ! (for example, the limit should be ~0.1 to 0.3 of abs(min)
store rgw
read Superantennae_i_rgs.fits
div rgw

! save it to fits with your favorite filename
```

4.13.1 Saving only the last cumulative map

It is possible to save on disk only the "final" cumulative `rgs` and `rgw`, i.e. cumulating all scans in the input list, instead of all cumulative maps. To switch on this option, set the parameter `wrCumLast` to "yes", instead of `wrCumMaps`.

As in the case of `wrCumMap`, the `wrCumLast` option cannot be combined with `nIterSource`>0. The procedure to write the last cumulative maps after N iterations is similar to the one described for the case of `wrCumMap` and is not repeated here. See Sect. 4.13 for details.

One possible application of the `wrCumLast` option is when the user would like to split the input list of scans into sub-lists and process several sub-sets of data in parallel, using multiple CPU cores. For example split the input list of scans in as many sub-lists as the number of available CPU cores. Section 4.13.2 explains how to combine the results back into a final map. To run PIIC on several sub-lists in parallel, it is necessary to open several independent PIIC sessions.

If this is the aim of using `wrCumLast`, care must be taken that the maps obtained from the sub-lists have all the same dimension and are all centered and oriented in the same way in the sky. In other words, the final maps will be combined pixel-by-pixel, and each pixel in each map to be combined with the others must correspond to the same pixel of all other maps. Because scans can have different directions in the sky, this does not mean that all maps must have an identical data coverage, but they must have the same geometry (i.e. pixels can have no signal, but must be there). To assure that this happens, instead of letting PIIC define automatically the position and size of the maps, use the parameters `posSeq` and `eqExtrAS` (see Sect. 4.2) and set them to the same values for all sub-lists.

4.13.2 Combining cumulative maps of sub-sets of data

Once several cumulative maps are produced for different sub-sets of data (see Sect. 4.13.1), the script `creaAver.piic` (found in the `pro/` repository) shall be used to build the final map using the equation

$$\frac{\sum_j \mathtt{rgs}_j}{\sum_j \mathtt{rgw}_j}, \tag{4.7}$$

thus effectively combining together all scans in the parent full input list, after data processing (compare to Eq. 3.1).

The script can be copied in the working directory and can be edited. It needs the file name of the output combined map, coded in the parameter `outName` and two input lists: one of all cumulative `rgs` maps and one of the `rgw` maps produced for the partial data sets. These lists have to be prepared by

the user; their default names[6] are `'outName'_rgs.LIST` and `'outName'_rgw.LIST`. As mentioned in Sect. 4.13.1, the user must take care that all maps of the partial data sets to be combined together have the same geometry in the sky.

The combination of `wrCumLast` and `creaAver.piic` can save a significant amount of processing time, especially for long scan lists and if a large number of CPUs is available. It used to requires a lot of preparatory work and a good bookkeeping of all sub-lists and partial results, but now this approach is handled in a simpler way by the multi-core scripts of PIIC (Chapter 5).

Before opting for this solution, the user should evaluate the effective gain in execution and analysis time with respect to — for example — a classic one-core only processing or a parallel processing of the full scan list with different PIIC setups on different cores, aimed at testing which pipeline configuration is best for the given data set.

4.13.3 Linking `rgw` and effective exposure time

It would be interesting to compute the effective exposure time per pixel, or at least a quantity proportional to it. As said before (see Sect. 3.4.3) the current `rgw` is not a correct representation of effective time because it is weighted. Moreover the kernel convolution performed by distributing (in Fourier space) the signal of each KID onto the final grid, makes the proportionality constant not easy to define analytically.

The first issue can be solved by performing an alternative data reduction, which avoids weighting. The resulting intensity and r.m.s. maps will *not* be correct, but the `rgw` will be a proper description of the effective exposure time per pixel, modulo a scaling factor.

The second issue can be solved by performing a naive re-gridding that uses a box-convolution instead of re-distributing the signal of each KID adopting the real kernel. The result would then be used only to compute the scaling factor, and for no other science-related purpose.

These solutions are described in Sect. 4.17.

4.13.4 Using `rgs` and `rgw` to produce individual scan maps

Representing the cumulative signal at each i-th scan in the list, including all scans from 1 to i, the `rgs` maps can be used to produce reduced maps of each individual scan, defined on the final grid. While doing this, one needs to take into account the necessary weighting.

Produce the map of the j-th scan in the list by simply applying the following equation:

$$\frac{\mathtt{rgs}\,(j) - \mathtt{rgs}\,(j-1)}{\mathtt{rgw}\,(j) - \mathtt{rgw}\,(j-1)} \tag{4.8}$$

Note that, when dealing with 1 mm data, it is important to avoid mixing Array 1 and 3 maps in this equation.

[6] for example, if `outName=Superantennae_a2` then the lists are called `Superantennae_a2_rgs.LIST` and `Superantennae_a2_rgw.LIST`.

4.14 The source map

Each iteration step produces three files, just like the 0-th iteration. One of three is the source map, as defined detecting pixels above the S/N ratio given by the user (`snrLevel` parameter in the optional settings script). This is not a mask, but an actual source map in FITS format. At each iteration, this source map is subtracted from the data after calibration and before performing all computations (e.g. sky noise subtraction, baseline correction, etc., see Sect. 3.3), and then it's added back.

For example, the source map produced at iteration `nIter=20` is called `Superantennae...._20f21.fits` and will be used during the next iteration (`nIter=21`).

4.14.1 Continuing where PIIC was stopped (during the iterative mode)

For specific, complex sources or data sets, it might be necessary to use a large number of iterations to reach convergence. In this case the PIIC data processing takes a long time. Similarly, very large data sets or very large maps also require long data reduction sessions.

It is possible that the data reduction is interrupted and stopped, for example in case of loss of net connection, black outs, unplanned reboots, other accidental reasons, or on purpose.

If the stop was during the iterative process, and — for example — m iterations out of N were already completed, not everything is lost: it is possible to re-start the data reduction from the same m-th iteration, slightly editing the data reduction script and taking care of few details:

- the source map produced at the m-th iteration shall be now used as input source map (parameter `sbSource` in the optional settings script);
- in the new run, the 0-th iteration corresponds to the previous m-th iteration;
- in the iterative loop, the 1st iteration corresponds now to the $(m + 1)$-th iteration of the run that was stopped;
- filenames will now be different, because: *a)* now the `sbSource` option is used; *b)* the numbering of iterations starts again from zero;
- the number of iterations `nIterSource` of the new (restarted) run needs to be adjusted, otherwise the effective total number of loops will be $m + N$, instead of simply N.

Note: The input source map needs to have coordinates WCS including the Epoch. This excludes the forma ICRS, that does not have Epoch. Moreover, the Epoch shall be the same as in the data (typically FK5 J2000). Fast-moving targets have their own epoch (i.e. the time of the observations).

4.14.2 Using a pre-defined source map

It is possible to have a user-defined source map as input, to be adopted as initial source map. It will be used during the 0-th iteration and then it will be refined by PIIC during the iterative process, based on S/N as usual.

Such a user-defined map must simply be a two-dimensional map in FITS format, including an Equatorial coordinates system at the J2000 epoch. The actual size in pixels of the map is not important, because the WCS will be used.

Note that using a pre-defined source map is to be considered an *expert-mode* operation. Defining it properly, without knowing how the emission is at the NIKA2 frequencies, is complex and could involve a bit of gambling. For example, such a map might come from the analysis of shorter/longer wavelengths (e.g. *Herschel*) data, or from physical assumptions, or from a combination of the two. It is, though, not granted a priori that the underlying assumptions are fully correct.

The parameter defining the file name of this input pre-defined source map is again `sbSource` in the optional settings script.

4.14.3 Produce the source map a posteriori

In some cases, the user might like to produce new source maps a posteriori, for example to test how different noise r.m.s. polygons would influence the source detection and thresholding.

The script `creaSource.piic`, included in the PIIC `pro/` repository, is designed to produce a source map, starting from the final map, the `rgw` map and the noise r.m.s. polygon. Use it as follows:

```
@creaSource mainName polName snrLevel souSign smSouPar
```

where `mainName` is the main file name (i.e. the map produced by PIIC at any i-th iteration), `polName` is the polygon file name, `snrLevel` is the threshold above which pixels are selected (e.g. by default equal to 3.0 or 3.5 depending on the data reduction setup, see the optional settings script), `souSign` is the sign of the expected signal (positive, negative or both) and `smSouPar` is the desired smoothing value (see Sect. 4.14.4).

4.14.4 Enhance detections by smoothing

Excluding sources from sky noise is based on the source map generated during the iterative process. The latter is produced by identifying those pixels above a given S/N ratio on the final map. In some cases with low S/N ratio, the detection of source pixels likely will miss some of the signal. As mentioned in the previous Chapter, this might cause an important loss of information, and can even create spurious effects.

In order to improve on the detection of source pixels, it is possible to apply a smoothing algorithm to the S/N map and to the source map. The parameter `smSNRpar`, found in the main template script, controls this feature:

- if `smSNRpar`> 0, the S/N ratio is calculated using a smoothed version of the final map, but the source map is not smoothed. In this way one goes deeper in the detection.

- if `smSNRpar`< 0, also the source map is smoothed. This helps to cover more extension of the sources, i.e. it helps to go further than the contour defined by the mere S/N thresholding.

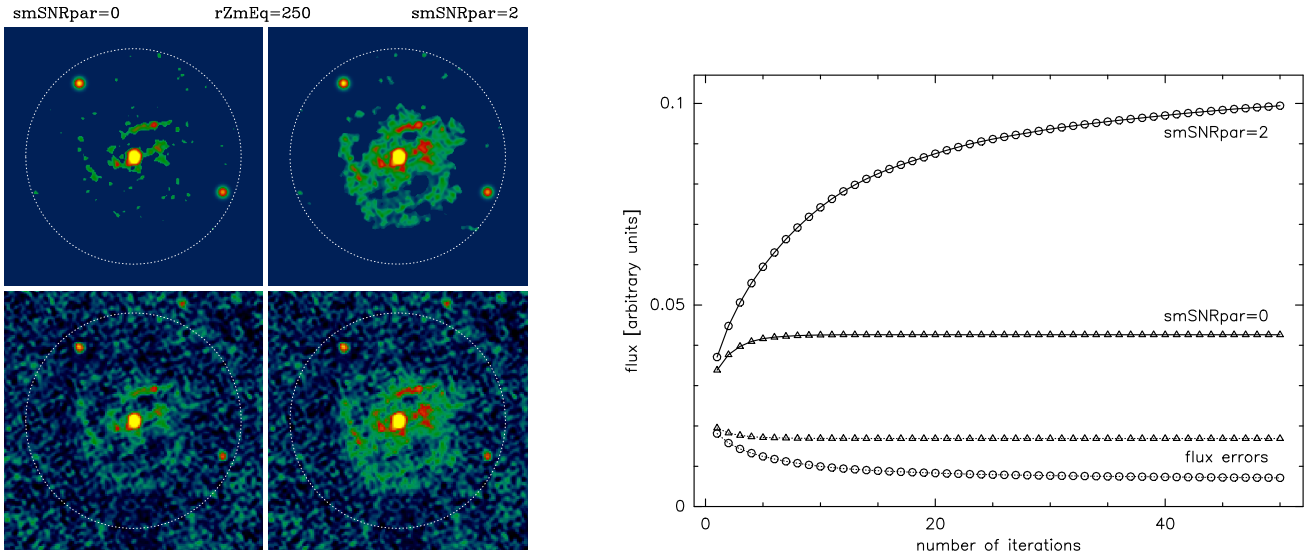


Figure 4.7: Effect of the `smSNRpar` parameter in the processing of faint, extended sources. *Left*: the source map (top panels) and the final map (bottom panels) obtained without the smoothing option (left) and with smoothing (right). Note again that the smoothing is employed only during the S/N thresholding process and the final map is *not*. smoothed. *Right*: the total recovered flux of the faint extended source, and its uncertainty, as a function of iteration, with (circles) and without (triangles) smoothing for the same case shown in the left panels.

The first case is useful for faint sources in general, including point-like and compact sources. The second is specifically more interesting for weak, extended objects (e.g. SZ signals).

The details of the improvement obtained are difficult to predict, because they depend very much on the data (geometry, strategy, instabilities, etc.). Therefore a bit of experimentation is required, to optimize the smoothing parameters. Figure 4.7 shows the net effect of using the `smSNRpar` to process an extended hypothetical source dataset.

Two different smoothing algorithms are possible at this time:

1. if $|$`smSNRpar`$| \leq$ HPBW[7] then a so-called "*conjugate gradient (CG) smoothing*" is adopted. The smoothed image is the equilibrium state of a thin flexible plate constrained to pass near each height data by a spring attached between it and the plate (see the `cgsm` help for more details). In this case, `smSNRpar` represents the stiffness of the spring and is dimensionless. The CG smoothing creates a very "clean" result, but it is not possible to tell what is the final resolution. It affects point-like sources less than a Gaussian smoothing, and is instead more effective in smoothing the noise.

2. if $|$`smSNRpar`$| >$ HPBW then a Gaussian smoothing to the given resolution is used. In this case, the value of `smSNRpar` given by the user is the final HPBW of the smoothed S/N map, in units of [arcsec].

[7]Values of 11.6 and 18.0 [arcsec] are adopted, i.e. slightly broader than the nominal NIKA2/30m HPBW at 1 mm and 2 mm.

If the smoothing is used for point-like sources, only small positive values shall be used: $0 <$ `smSNRpar` ≤ 0.5. When extended sources are analyzed, it is suggested to apply the smoothing also to the source map, i.e. `smSNRpar` < 0.

We stress that the final map produced by the data reduction is not smoothed. Smoothing is applied only to enhance the detection of source pixels during the thresholding process.

4.15 Adding artificial sources to the timeline

PIIC allows one to add artificial sources, models, or other other kind of signals to the data timeline, before undergoing the data reduction (and after the first calibration steps; see Sect. 3.3). This feature is useful in several different contexts.

One important application is the study of how the data reduction pipeline affects the data, computing a so-called "*transfer function*" (see Appendix C) for their typology or targets (e.g. point-like sources, galaxies with different profile shapes, SZ signal, Galactic clouds, etc.). Having a model that describes the expected or typical emission of the sources, one can inject it in the timeline and compare the result to the "intrinsic" model emission hitting the telescope and the KIDs, which is known a priori by definition.

Another possible use is the development of completeness and reliability simulations for (extragalactic) blank-field surveys. Once the flux distribution of the sources is known (or modeled), point-like objects are added to the timeline and processed. Comparing the output and input distributions, leads to the completeness characteristics of the data set and the incidence of spurious detections.

The parameter controlling this feature is called `addSource` and is found in the "optional sets" script. Its value shall be the file name of a FITS map that will be added to the timeline. This map can consist of a source model, or anything else. In the blank fields example, this would be a map with null background and no noise, containing a number of artificial sources, spread over the field, with a point-like profile, and having a given flux distribution. Other kind of observations, obviously require different kinds of artificial sources (model maps).

The model map can be produced with any tool, be it PIIC or any external software. The size, pixel scale and shape of the artificial map can be any and are left to the user's choice. It is paramount that the artificial map includes a *World Coordinates System* (WCS) in Equatorial coordinates at epoch J2000, that covers the position of the real scans in the sky. The signal on the artificial map is transferred to the timeline on the basis of the (RA,Dec) coordinates of each pixel. The units of artificial input map must be [*mJy/beam*], where the NIKA2 beam is intended (see Sect. 4.3).

Note that no convolution with the 30m/NIKA2 PSF is performed at this stage, therefore the artificial map should already include the effects that light undergoes during its path from the source to the detector. Adding an artificial source and studying it a posteriori gives only a description of the effects of PIIC processing on the data.

PIIC does not provide the means to perform the post-processing analysis, e.g. the study of the "transfer function" or of the data "completeness/reliability". It only gives the tool to add the artificial sources to the timeline. The rest of the analysis is left to the users and their dedicated procedures.

Appendix B makes use of this feature to evaluate the effects of re-gridding on the profiles of

point-like sources.

4.15.1 Null maps (a.k.a. jackknife)

Artificial sources added to the time line will appear on the final combined map, that includes also the signal of the real sky. On the other hand, PIIC can produce *null maps*, setting the `nullMap` parameter to "yes". This parameter is found in the "optional sets" script.

When this option is used, after the rejection of problematic scans, the signal of each second scan among those that are left[8] is multiplied by **-1**, before processing. The final product is a null map, where real sources and background are cancelled.

In some astronomical contexts (see e.g. Herschel/SPIRE or SCUBA extragalactic surveys and Planck work), this method is also called "jackknife". Note that this has nothing to do with the statistical jackknife, originally developed by M. Quenouille, that consists in producing subsamples of $N-1$ scans and studying the statistics between the subsamples.

When using the `nullMap` and the `addSource` options combined, the artificial signal is added after multiplying by -1 each second map. The `nullMap` option can of course be used also without adding artificial sources, in case it serves for any other purpose.

When the null map is requested, the products of data processing are three files as for the normal data reduction. The file names are the same as in the non-jackknife case.

4.16 Saving maps of individual scans

It is possible to save the maps of individual scan, computed on the final grid, i.e. the same used to produce the final, combined map. Such individual maps can be used — for example — to compute pointing corrections between individual scans, identify and exclude scans of poor quality (for whatever reason), check the effects of scanning in different directions, etc.

Enable this feature by switching the parameter `wrIndMaps` to *yes*. This parameter is found in the "optional sets" script.

Because these individual maps serve for other analysis purposes and are not instrumental to create the final map, when using `wrIndMaps`, PIIC does not produce the combined map, nor the associated weights and source map. Similarly, it is not possible to associate the use of `wrIndMaps` with the iterative mode.

The ideal way to save individual maps and use them properly is then:

- perform the standard data reduction with PIIC, performing N iterations if needed, and producing the combined maps. At this stage keep `wrIndMaps`="no".
- set `wrIndMaps` to "yes" and perform a second run of PIIC, limiting it to the 0-th iteration only. Use the source map produced at the N-th iteration as `sbSource` input (in the optional settings script, see Sect. 4.14).

[8] the number of scans accepted by PIIC (out of the input list) might not necessarily be even. If it's an odd number, then the signal of one scan is left in the final "null" map. The user is invited to double check the actual number of used scans printed by PIIC at the beginning of the reduction.

- perform the analysis for which the individuals maps were intended.
- if necessary, as a consequence of this analysis, adjust the PIIC setup and perform again the standard (iterative; `wrIndMaps`="no") data reduction to now produce the final, optimal result.

4.16.1 Combine the maps of individual scans

The maps of individual scan, obtained with the `wrIndMaps` option, can be combined together using the script `combIndMaps.piic`, found in the `pro/` sub-directory of the PIIC distribution. The script combines both the signal and the `rgw` maps, provided that their filenames follow the PIIC standard convention (Sect. 3.6): the only difference between the two is the "rgw" string at the end of the filename, before the `.fits` extension. If not, then the script needs some editing.

It is important to stress that the `combIndMaps.piic` produces the correct result only for individual scan maps obtained with the `wrIndMaps` option and *not* with the `wrCumMaps` one. In the latter case, use the script `creaAver.piic`, as described in Sect. 4.13.2.

4.17 Hit map

Some science projects might have interests in recording how many times the pixels of the final map have been *hit* by all the scans, records, KIDs of the data set under analysis. One possible application is to evaluate the effective integration time per pixel. It is possible to compute a so called *hit map* with PIIC using the parameter `calcNhits` that is found in the optional settings macro. Three options are available:

`calcNhits=0` In this case, PIIC operates normally and no hit map is produced.

`calcNhits=1` The hit map is computed distributing the signals to the final grid according to the NIKA2 beam.

`calcNhits=2` The hit map is computed without distributing the signals, i.e. the hit is assigned on a 1:1 (KID:pixel) basis.

The resulting hit map is stored in the `rgw` file. The file name includes the value of `calcNhits` (first character after `sourcename_a*`).

It is important to point out that the signal map and `rgw` map produced by the standard data reduction (i.e. `calcNhits=0`) are obtained weighting the contribution of each KID by the inverse of the noise r.m.s. (along the timeline) squared. On the other hand, this weighting cannot be applied to produce the hit map, because otherwise one would simply obtain the `rgw` map, that has a different meaning (see Sect. 4.13.3).

The hit map gives — so to say — an estimate of the formal time per pixel that went into the observations, but it does not fully correspond to what goes in the final map r.m.s. because the latter takes into account the various instrumental and environmental effects (e.g. instabilities) via weighting. Moreover, the effect of the extinction correction is different in the two cases, because the noise properties are different.

For these reasons, a direct comparison of an effective integration time based on this (non-weighted) hit map to the properties (e.g. the r.m.s.) of a weighted map can give only qualitative results. In principle, to make a fully correct comparison, a non-weighted signal map should be used in combination to the hit map. Such a map is also produced when `calcNhits>0`, but it is much noisier than the proper signal map obtained with `calcNhits=0`.

Inverting the point of view: associating the non-weighted hit map to the weighted signal map can provide only a lower limit to the effective integration time per pixel, at a given map r.m.s.

With this caveats in mind, in order to produce the hit map in addition to the proper signal and weights map, it is necessary to perform a double PIIC pipeline run[9] (similarly to when maps of individual scans are saved, see Sect. 4.16). A simplified (quick) way to do it is as follows:

1. run the standard data `calcNhits=0`. Perform as many iterations as needed.

2. run PIIC with `calcNhits=1` or 2 and using the source map produced at the last iteration before (specifying it as `sbSource` input: Sect. 4.14.1, and Sect. 4.14 in general). One iteration should be then sufficient at this stage.

The file names of the maps produced by the two runs might seem similar but differ by the label linked to `calcNhits`.

It is important to note that the source map produced with the weighting setup (i.e., the standard data reduction) is different from the one produced without weighting (e.g., when computing the hit map) because of the different noise level resulting from the two methods.

4.18 Pointing and calibration correction between scans

The telescope pointing errors can have an impact on the products of NIKA2 observations. Since the final map is a weighted average of a number of scans/maps, if not corrected, pointing errors will broaden strong compact sources and will dilute their flux. Some flux fraction of weak wings might decrease below the detection limit and might be missed. Furthermore, in the unlikely case that some scans suffer from calibration errors, also a relative re-scaling of flux can be applied [10].

It is possible to correct the pointing and flux offsets between the different scans of a data set, by setting the parameter `nrCorrPoiCali` to 1, 2, or 3. This parameter is found in the "optional settings" script. A significant amount of extra work by users is required to use this feature.

The offsets to be applied to each scan are read from the file `NIKA2gfEqElli.pccIn`. The format of this file is:

```
arNr scanNr date   corrRA corrDEC corrCal
```

`e.g.:`

[9]The hit map and the weighted products are not produced in one single run because the operations and the execution time would be anyway doubled.

[10]the source map used in the iterative process comes from the weighted average of many scans. If some scan has a calibration problem (i.e. is an outlier in the list, in terms of calibration) then the source map will not be representative of that scan anymore. In such cases, the flux relative re-scaling is useful.

```
1    131 20190118  -0.727  3.114  0.94786
2    131 20190118  -0.507  2.896  1.04733
3    131 20190118  -0.635  3.308  1.07845
```

The adopted value of `nrCorrPoiCali` indicates the array to be used for the corrections. In this example, if `nrCorrPoiCali`=2 the pointing corrections will be $(-0.507, 2.896)$ in RA and Dec.

The pointing corrections are the difference between the offset position of the same source in the individual maps and the average map:

$$\mathrm{corr}\,(\mathrm{RA}) = \mathrm{off}\,(\mathrm{RA})_i - \mathrm{off}\,(\mathrm{RA})_{\mathrm{avg}}$$
$$\mathrm{corr}\,(\mathrm{Dec}) = \mathrm{off}\,(\mathrm{Dec})_i - \mathrm{off}\,(\mathrm{Dec})_{\mathrm{avg}}$$

Where by "offset" it is intended the offset of the source from the center of the final map in units of [arcsec] (in PIIC/GREG coordinates conventions).

The flux calibration correction is the ratio of the fluxes obtained for the same source in the individual maps and the average map:

$$\mathrm{corr}\,(\mathrm{Cal}) = \frac{\mathrm{flux}_i}{\mathrm{flux}_{\mathrm{avg}}}$$

These measurements shall be performed on compact sources, if not point-like, bright enough to be detected on individual scans. *Hence the correction of pointing offsets between scans is possible only for data sets that have such sources in the field.* There is no limit to the number of sources to use: the more the better.

To derive the fluxes of sources and the flux calibration correction, it is preferable not to use the peak fluxes; integrated fluxes are a better choice (e.g. the integral of a Gaussian fit to the sources profiles is an and easy solution). The reference source used for re-calibration must *not* be intrinsically variable.

The file `NIKA2gfEqElli.pccIn` must have an entry for each scan to be processed, for the array chosen as reference. Note that the reference array for the pointing corrections (i.e. `nrCorrPoiCali`) can be different from the array processed to derive the combined map. For example, when producing the Ar 1+3 map, one can use either Ar1 or Ar3 pointing offsets; or for array 2 (1) maps, one can use array 1 (2) as reference for the pointing corrections, depending on the desired angular accuracy of the corrections or on source brightness; etc. These are just examples and other cases might happen.

On the other hand, when applying a flux calibration correction, it should be avoided to mix Arrays, i.e. the calibration correction of Ar 1,2,3 must be done with Ar 1,2,3 data only.

The PIIC package does not include procedures to measure the position and fluxes of the sources and to build the `NIKA2gfEqElli.pccIn` *file. This choice is driven by the fact that every project and data set has different characteristics, different kind of sources, and therefore different techniques might be preferred by the users. This file has to be produced by users with their own tools.*

Finally before using the `nrCorrPoiCali` option, one additional directory `pcc/` needs to be created. The corrections applied to each map will be written there. The value of the `nrCorrPoiCali` parameter is reported in the output file namses, directly after the value of `calcNhits` (see Sect. 4.17).

4.19 What if reading errors occur?

When processing a long list of scans, it is possible that some of them contain corrupted data and cannot be read by PIIC. If this happens, PIIC will stop with an error message, waiting for the user to give the `continue` (`c`) command, or even exiting the processing in the worst cases. The same issue arises also if some polarimetry data, taken with the rotating HWP, are included in the data processing of total power scans, or vice versa.

In order to avoid such stops, it is possible to use the `pauseIfReadErr` variable. Its default value is `yes`, i.e. PIIC will pause if a reading error is encountered. Setting it to `no` will avoid that pause. The `pauseIfReadErr` variable is found in the `mapTPoptionalSets.piic` script and shall be added to the main `mapTP` data processing script, in the custom settings section, after the call to the optional settings script itself.

Importantly, `pauseIfReadErr` must not be set to `no` when using the multi-core processing scripts (Chap. 5), because one would risk to break the data reduction chain, thus ending in hanging inactive processes. In the multi-core case, all scans in the master list must be accepted by the data reduction (see Sect. 5.2.1 for details).

4.20 Saving time

Here we list some ways to reduce the PIIC processing time. Before opting for any of these solutions, the user should be aware of its counter-indications and should evaluate if it brings sufficient advantages for their case. So far, the following possibilities are available:

- test different setups of the PIIC data reduction pipeline in parallel on different CPU cores, using different PIIC sessions (mentioned in Sect. 4.13.2);
- split the data set in sub-sets and proceed as described in Sect. 4.13.1;
- reduce the time spent reading and pre-processing scans (see Sect. 4.20.1).

4.20.1 Reduce the read and pre-processing time

Reading IMBFITS data files and pre-processing scans takes up to $\sim$ 30% of the total PIIC data reduction time. By default, PIIC reads and performs the pre-processing of all KIDs.

It is possible to decrease the time spent on this phase of the data reduction by selecting a priori only those KIDs that are not flagged in the DAFs (see Appendix A) as unusable. If the `useDAFsFR` parameter (found in the optional settings script) is set to "yes", PIIC will use the information stored in the DAFs to exclude those flagged KIDs from the reading and pre-processing operations.

In this way, up to $\sim$ 30% of the reading and pre-processing time can be saved. This option comes, nevertheless, with a caveat: when using it, it can happen that PIIC won't identify some couples of KIDs tuned too close in (tuning) frequency, because one of the two might have already been excluded a priori from reading using the DAFs pre-selection. This implies that setting `useDAFsFR` to "yes" there might be more KIDs contributing to the final map than when setting it to "no". The information carried by these KIDs is affected by cross-talking.

The incidence of this effect and the number of extra KIDs left depends on the quality of tuning and can be up to $\sim 3\%$ of the total number of KIDs per Array.

4.21 Moving targets data processing

Observations of moving targets such as comets can be carried out with NIKA2 and the IRAM 30m telescope, once their ephemeris are known. In this case, the epoch stamp of the data will not be the standard J2000, but the actual epoch of the observations. In order to reduce these kind of data with PIIC, few ad-hoc operations need to be performed.

First of all, it is recommended to gather the data by scanning the sky in the Equatorial system (i.e., set the on-the-fly scan scripts in Ra-Dec), to avoid problems (e.g., source elongation/distortion) with the rotation of $(\Delta\mathrm{Az}, \Delta\mathrm{El})$ *to* $(\Delta\alpha, \Delta\delta)$ *with elevation changes, during the observations at a given epoch.*

Once the data are on your disk, follow this simple recipe:

1. In PIIC, load the file list and sort it by date (Sect. 3.2.2):

   ```
   indir imbfitsDir
   inlist my_list
   sort inl
   ```

2. List the centers of all scans:

   ```
   list pos
   ```

3. Choose the pivot position (scan): ideally the central scan in time. For example be it scan m. Let its central position in Equatorial coordinates be `HH MM SS.ss DD MM SS.ss` and its observation epoch be `EEEE.eeeeeee`.

4. List the offsets of all scans with respect to the pivot:

   ```
   list off /pos HH MM SS.ss  DD MM SS.ss
   ```

5. Obtain the epoch of the pivot scan (m):

   ```
   read m ! "m" being a number
   show epoch
   ```

6. Add the following lines at the end of a local copy of the `mapTPrea.mopsic` script, using the values determined with the previous commands:

```
!!!!!!! Custom Moving Targets Modification !!!!!!!
   set epoch EEEE.eeeeeee EEEE.eeeeeee
   if system.eq."HORIZONTAL" then
      set syst eq
      position HH MM SS.ss  DD MM SS.ss
      set syst hor
   else
      position HH MM SS.ss  DD MM SS.ss
   end if
!!!!!!! End of Custom Moving Targets Modif !!!!!!!
```

7. Prepare the main PIIC data reduction script in the usual way, taking care of setting the position of the target as determined above (Sect. 4.2):

   ```
   let posSeq "HH MM SS.ss  DD MM SS.ss"
   ```

8. Run PIIC as usual (Sects. 3.3 or 5.4).

Chapter 5

Speed-up: multi-core processing

Take your time... with some sweet meringues

Processing large data sets that consist of several hundreds of scans and/or very big maps, using a single CPU core does take a considerable amount of time. A couple of possible suggestion on how to save computation time have been described in Sect. 4.20, but a radical change in the way the computer resources are used is needed, in order to reduce drastically the CPU time in these cases.

Starting from June 2023, the PIIC package includes a set of scripts that allow to perform the science data reduction using several cores in parallel. This Chapter is dedicated to this parallel offer.

It is worth to note that a good familiarity with PIIC is needed to use this multi-core feature, i.e. the user should already be at ease with the single-core data processing and some advanced options (see Chapters 3 and 4). Moreover, obviously only having a multi-core machine available makes it worth to use this approach. And please... in the overwhelming enthusiasm of using this multi-core mode, be careful not to overload that machine...

5.1 Generalities of PIIC parallel processing

The principles of the multi-core PIIC data reduction are identical to those of the traditional single-core processing (see Sect. 3.1). But now several CPU cores are employed to process sub-sets of scans in parallel, instead of charging this task on one core only. For example, if the user choses to split the original list of scans in ten sub-sets, ten CPU cores will work simultaneously to process the scans at each data reduction iteration.

The parallelization is not achieved by parallelizing the PIIC Fortran core, but rather by streamlining and automatizing the use of multiple PIIC sessions in parallel, each one processing a sub-set of scans out of the full data set. The result of each PIIC parallel session are then combined together to produce the final products of the given data reduction iteration. This is done by writing on disk the results of each session and reading them again at the combining step.

To this aim, the list of all scans that need to be processed for the given NIKA2 array(s) needs to be split in sub-lists. This can be done with any technique. Section 5.2 describes how to do it with PIIC and points out important caveats to be kept in mind.

Two main scripts are involved in the parallel PIIC processing and have to be prepared by the user. The PIIC distribution `pro/` directory contains a template of both. The *master* script is named `source_a2_MP_template.piic`; the *slave* script is `source_setMPpar.piic`. Section 5.3 gives the necessary details for properly editing, setting up and naming these two scripts for the multi-core PIIC data reduction. Figures 5.1 and 5.2 show the template version of the two scripts.

5.2 Splitting the list of scans

The first step to prepare the PIIC multi-core processing is to organize the data set in multiple mists of scans, each one to be processed in a different PIIC session. As mentioned above, splitting the full list of scans into sub-lists can be done with any technique, as long as the produced sub-lists are one-column ascii files.

PIIC offers the possibility to split it in sub-lists with one simple command. For example, let's assume again that the observed target is called *Superantennae* and that the full list of scans to be processed for the NIKA2 2mm array is in hand, contains N files and is called `Superantennae_a2.LIST` (see Sect. 3.2.2, if needed). Proceed as follows:

```
inDir imbfitsDir
inList Superantennae_a2  ! omit the .LIST extension!
sel obs m                ! select scans of type "map"
sort inList              ! sort the list, if desired
wri inList Superantennae_a2 /perChunk 10
```

The option `perChunk` of the `write` command specifies that the output list will be split into sub-lists, each one containing n scans (in this example $n = 10$). Obviously, the last sub-list might include a smaller number of scans than all other lists if the total number of scans is not a integer multiple of n. It is important to keep in mind the actual number of sub-lists, because it is needed when editing the scripts (see Sect. 5.3).

The file names of the sub-lists produced in this way are `Superantennae_a2_1.LIST`, `..._2.LIST`, `..._3.LIST`, etc. If the user produces the sub-lists in some other way (e.g. not using PIIC), these are the file names that need to be adopted.

5.2.1 Important caveats

A few important details need to be kept in mind when producing the sub-lists of scans to be processed in parallel:

1. the sub-lists file name must begin with the same source name adopted in the scripts (see Sect. 5.3).

2. the scans included in the sub-lists need to be "accepted" for processing, i.e. the user shall down-select the list of all scans (before splitting) and exclude those that PIIC will not accept for any reason. This implies that one should already know which scans will fail during the data reduction, and therefore some preliminary processing might be needed to this aim. This

source_a2_MP_template.piic

```
!-------------------------------------------------------------------------------------------------
!
!  Written by Robert W. Zylka, zylka@iram.fr
!  created 31.03.2023, version 28.11.2023
!
!      execution: piic @scriptName   (.piic is the default extension)
!
! template script to run nProc parallel processes
!
!-------------------------------------------------------------------------------------------------

@ mapTPdefs

sic\defi integer finishedProc finishedProc0 /glob

let souName "??????"    ! the source, e.g. SuperAntennae
let nikaBand    "2"     ! process NIKA2 A2 data, 1=A1, 3=A3, 13=A1+A3; other names not accepted
let nProc       ?       ! number of processes, i.e. number of the sub-lists
let nIterSource ?       ! number of iterations; higher values might be necessary
let sbSourceIn "none"   ! "single beam source", FITS file name of the source distribution to
                        ! start the iteration (may include directory)

@ 'souName'_setMPpar 'sbSourceIn'

sic mkdir red
sic mkdir stat
sic dele ERROR* stat/'souName'*_a'nikaBand'*.excl
outDir red

for iterSource 0 to nIterSource
    sic delete finishedA'nikaBand'proc*

    !!! submit nProc processes
    @ mapTPsubmitMP 'iterSource' 'snrMx'

    sic mkdir msA'nikaBand'i'iterSource'
    sic wait 90
    system "mv PIIC.* *.log msA"'nikaBand'"i"'iterSource'

    !!! wait till all nProc processes are finished
    @ mapTPwaitMP 'iterSource'

    !!! create the weighted iterative average map 'outName'.fits
    !!! and the weights map 'outName'rgw.fits
    @ mapTPaverMP 'iterSource'

    !!! create the iterative source map 'outName'f'iterSource+1'.fits
    !?if iterSource.lt.nIterSource then    !! enable or disable, depending on needs
        @ mapTPcreaSource
    !?end if
    inDir imbfitsDir

    system "mv "'souName'"_a"'nikaBand'"_rgs.LIST "'souName'"_a"'nikaBand'"_rgw.LIST
finishedA"'nikaBand'"proc* msA"'nikaBand'"i"'iterSource'

next

exit
```

Figure 5.1: Master script for multi-core PIIC parallel processing.

exclusion is necessary because if any of the parallel sub-processes fails (pauses), then the whole multi-core parallel processing stops.

source_setMPpar.piic

```
!-------------------------------------------------------------------------------------------------------------
!
!  Written by Robert W. Zylka, zylka@iram.fr
!  created 09.03.2023, version 24.11.2023
!
! Internal subroutine; script defining the parameters for multi-core processing of NIKA2 maps.
!
! IMPORTANT WARNING!
! If the maps in the scanning direction are shorter than
! 2*(souRmxAS+recArRmxAS+HPBW+scanVelocity),
! where souRmxAS is the source semi-axis in this direction, recArRmxAS the NIKA2 FoV
! radius, scanVelocity is in (units of souRmxAS,recArRmxAS,HPBW)/s,
! some crucial processing steps cannot be performed. Consequently, due to inevitable
! data instabilities (intrinsic to NIKA2 or induced by the chosen mapping strategy),
! the results in such cases might be questionable.
!
! Further notes:
! - contrary to processing with just one processor the maps size must be defined
! - For simple structured sources the best might be to define the source region
!   (as an ellipse or using a polygon) but set the base range to 0:
!   brRmxAS=0 brRmnAS=0 and polBReq=" " (to be set after mapTPoptionalSets).
! - As a consequence of very low integration time and data instabilities, at edges
!   of the iterative maps S/N spikes and/or "instability" sources may appear.
!   To avoid their propagation while iterating towards the map centre, either polZmEq or
!   rZmEq should be used.
! - Defaults are set for Ar2 and not weak sources, i.e. max. signal > 5*rmsOfNIKA2pixel.
!   The average rmsOfNIKA2pixel is roughly ~33mJy/23.8Hz for Ar2, ~110mJy/23.8Hz for Ar1,
!   and ~87mJy/23.8Hz for Ar3, but many KIDs show smaller r.m.s. than average,
!   therefore sources stronger than ~100mJy/2mmBeam and ~350mJy/1mmBeam are "not weak".
!
!------------------------------------------- parameter setting ---------------------------------------

let weakSou   no      ! if yes signal >5*rmsOfNIKA2pixel (see above) masked after the sky noise subtraction

let deepField no      ! yes for e.g. GOODSNorth, COSMOS, HDF, DeepField1, ...

let souSign "+"       ! "-" if source with negative signal, e.q. SZ; "+-" if positive and negative sources

let posSeq " "        ! [H M S D AM AS] centre of the final (R.A.,DEC) map
                      ! necessary only if individual maps have different centres

let eqExtrAS    ?     ! [arcsec] half map extent in EQ, 0 to calculate it (all maps must have the same centre)
let souRAoffAS  0.0   ! [arcsec] R.A. offset of the source relative to posSeq
let souDECoffAS 0.0   ! [arcsec] DEC  offset of the source relative to posSeq

let blOrderOrig 2     ! order of instability corrections in subscans, for compact sources >2 might be better

if nIterSource.ge.1 then
   let rZmEq   0.0    ! [arcsec] if >0 outside of this radius the data of the iterative source are neglected
   let polZmEq " "    ! [arcsec] relative to posSeq, action as for rZmEq but using this polygon
   let smSNRpar 0.0   ! if >0 S/N is calculated using the smoothed map but the iterative source not smoothed,
                      ! if <0 also the iterative source smoothed; the ITERATIVE MAPS ARE NEVER SMOOTHED
end if

@ mapTPoptionalSets

!------------------------------------------ Don't change these -------------------------------------

let wrCumLast yes
if PRO%nArg.le.0 then
   let sbSource " "
 else
   let sbSource &1
end if

!------------------------------------------ Ste's custom parameters --------------------------------

!!let nBest 32          ! use data of nBest best correlating KIDs to calculate CN for the pixel to be filtered

!------------------------------------------------ Run it! ------------------------------------------

return
```

Figure 5.2: Slave script for multi-core PIIC parallel processing.

3. if using the `nullMap` option, the number of scans in each sub-list must be an even number.

4. with the `nullMap` option, special extra care might be necessary when sorting the scans. If different scans cover different regions of the sky (e.g. different orientation *and/or* different areas), the lists must be organized such that scans are sorted in pairs covering the same region. Note that this is not necessary if the scans were observed with different scanning directions but cover exactly the same patch of sky.

5.3 Preparation of the scripts

Once the lists of scans are split and properly organized in sub sets, it is time to prepare the scripts for the multi-core PIIC parallel processing. The two main scripts to be used are `source_a2_MP_template.piic` and `source_setMPpar.piic` and are included in the `pro/`. directory of the PIIC distribution..

5.3.1 The master script

It is convenient to rename the master script `source_a2_MP_template.piic` such to add some key information about the data reduction. Continuing our example, let's call it `Superantennae_a2_MP12i20.piic` to indicate that the target is called Superantennae, 12 parallel processes will be run, and 20 iterations are forseen.

Few parameters need to be set in the master script:

- the source name, `souName`, i.e. the same adopted for the slave script and the scan lists.
- the NIKA2 array to be processed, `nikaBand` (2, 13, 1 or 3, as usual).
- the actual number of data sub-sets involved in the parallel data reduction, `nProc`, as obtained when splitting the list of scans (see Sect. 5.2).
- the desired number of iterations, `nIterSource`.
- a possible starting source map, `sbSourceIn`, in case this is the continuation of a previously interrupted processing (see Sect. 4.14.1).

Very importantly, please *do not alter* how the definitions and commands are sorted in the script, so to avoid conflicts in the data processing flow.

5.3.2 The slave script

The slave script `source_setMPpar.piic` must be renamed using the proper source name, as adopted in the master script and in the lists of scans. In our example, its name is `Superantennae_setMPpar.piic`.

The content of the slave script is analogous to that of the `mapTP_template_a*.piic` script in the single-core traditional PIIC processing. It defines the main data reduction parameters to be used in the parallel processing. The detailed description and use of these parameters can be found in Chapters 3 and 4.

As in the case of the master script, the order of the definitions and commands in the slave script *must not be changed*, otherwise one risks to induce conflict in the data reduction pipeline. The user is invited to add any additional optional settings at the end of the script, where specifically indicated, and in any case before the `return` command.

5.3.3 Scans with different sizes

Ideally, the scans used to produce the final science map have all the same size and center. It may happen that, during the observations, the scan size has been adjusted and changed, so that different scans have different sizes. This change of size might occur with or without a change of the scan center.

In the case that the chosen scans have both different scan center and size, simply operate as in the single-core data reduction (see also Sect. 4.2): define a common `posSeq` center and the desired size, `eqExtrAS`, of the final map in the slave script.

In the case that only the scan size changes, but the scan center is the same for all scans, the multi-core processing acts differently with respect to the single-core processing. In fact, in this case the sub-maps produced for each sub-list will have different sizes and the process of combining them together won't work. Also in this case, define the half-extension of the final map, `eqExtrAS`, explicitly.

5.4 Run it!

When all the needed files are prepared, copy them all in a common working directory. At odds to the single-core data processing, this time the user does not have to set up the sub-directories tree. Only the data repository is needed, which can be set as a symbolic link as done previously. In summary, do the following:

```
cd
mkdir my_working_dir
cd  my_working_dir
cp -p .../Superantennae_setMPpar.piic .
cp -p .../Superantennae_a2_MP12i20.piic .
cp -p .../Superantennae_a2_*.LIST .
ln -s .../data imbfitsDir
```

It is important to point out that the processing of each different data set must be performed in a different working directory. This holds also for different NIKA2 arrays of the same set of scans. This is necessary mainly to avoid mixing the many PIIC log files produced during the multi-core parallel processing. PIIC will then take care of creating the necessary sub-directories of each working directory and organizing the output files accordingly.

Once the files have been copied in the working directory as described above, simply start the multi-core PIIC parallel processing by calling the master script:

```
piic
@ Superantennae_a2_MP12i20
```

5.4.1 Running the multi-core processing in *batch mode*

Even using the multi-core parallel processing, for long data reduction sessions of very big data sets it might be convenient to use the *batch mode*, so to be able to close the terminal from which PIIC is run (see also Sect. 3.3.3).

To do this, simply use the following command line from the usual shell:

```
piic -nl @ Superantennae_a2_MP12i20 > Superantennae_a2_MP12i20.log 2>&1 &
```

or

```
nohup piic -nl @ Superantennae_a2_MP12i20 > Superantennae_a2_MP12i20.log 2>&1 &
disown
```

The `-nl` option avoids the creation of GILDAS log files (not to be confused with the PIIC logs nor the batch mode logs).

5.4.2 What to do if it fails on some scan

If the multi-core data processing stops without producing the desired output, it means that something prevented it to reach the end. One likely cause of failure is if some scan(s) are not accepted by the data reduction, or if they produce errors at some stage of the process.

There are a couple of ways to identify which scans failed and need to be removed from the original list of scans.

In case maps are not accepted, the PIIC multi-core scripts write an error file called `ERROR.NOTACCEPTEDMAPS` in the main working directory. This file lists the name and path of some `*.excl` files that list the not accepted scans. Simply exclude those scans from the parent list and start from scratch, i.e. split again the list into sub-lists and proceed further.

In case of other errors, it is possible to inspect the `PIIC.MES#` message files produces by PIIC for each sub-process. They are stored in the many `msA*i%` sub-directories produced during the multi-core processing. Seek for the "*fatal*" string in all `PIIC.MES#` files and identify which scans produced an error. For example, in the case of Arrays 1+3 and the 0th iteration:

```
cd msA13i0/
grep -i "fatal" PIIC.MES*
```

Once identified, simply exclude the affected scans from the original list and start from scratch, including splitting the main list of scans into sub-lists.

5.5 Products

Naturally, the multi-core PIIC parallel processing produces a very large number of output files. We describe here all the products, with no ambition to be detailed about their content. Several of these products have already been described in Chapters 3 and 4. We invite the interested users to explore the others or to contact the PIIC team, in case of doubts or further needs.

During the multi-core processing, PIIC automatically produces the sub-directories `red/`, `stat/` and `msA*i%/` and moves its products in there.

5.5.1 Content of the red/ sub-directory

For each iteration, and for each sub-process (i.e. for each sub-list of scans), PIIC stores in the red/ directory the rgs and rgw maps, i.e. *regular grig signal* and *weights* (see Sect. 4.13). Their filenames are

```
source_a*_#_....rgs.fits
source_a*_#_....rgw.fits
```

where * is the NIKA2 array number (2, 13, 1 or 3) and # is the number of the given sub-process (i.e. sub-list of scans).

When the processing of all sub-processes is completed for the given iteration, PIIC combines the results of all sub-processes together and produces the maps labeled MP:

```
sources_a*_MP#...%.fits          map
sources_a*_MP#...%rgw.fits       weight map
sources_a*_MP#...%.pol           rms polygon
sources_a*_MP#...%f%%.fits       source map produced at iteration %
                                  to be used in the next iteration %%
```

where now # is the total number of sub-processes (i.e. sub-lists of scans) used and % is the iteration number.

In principle all intermediate files, i.e. those without the MP label in their file name, can be simply deleted, unless the user needs them specifically for some of their science goals.

5.5.2 Content of the stat/ sub-directory

The content of the stat/ sub-directory produced by the multi-core PIIC parallel processing is equivalent to that of the single-core processing, with the solely difference that now the files are produced not only for each iteration, but also for each sub-process/list. The naming of the files is similar to the names used in single-core processing, but now it reflects also the sub-process number.

5.5.3 Content of the msA*i%/ sub-directories

During the multi-core processing, the numerous log and message files are moved to the msA*i%/ sub-directories. In this case, * is the NIKA2 array number and % is the number of the iteration to which the logs and messages belong. Few different file types are found here:

```
*rgs.LIST                 list of the intermediate rgs maps
*rgw.LIST                 list of the intermediate rgw maps
PIIC.LOG*                 the usual PIIC log files
PIIC.MES*                 the usual PIIC message files
*.log                     log of the screen messages redirected to file
                          by the batch mode used by the scripts
*.dat                     intermediate work files for each sub-list/process
```

These files contain useful information to track possible problems (e.g. if the processing of a map fails), but are not very useful for the science analysis. Therefore they can be deleted without harm, once the data reduction is completed.

5.6 Additional important notes

The PIIC multi-core processing presented in this Chapter has already been tested for: standard data reduction; use of null maps; adding artificial sources onto null maps; smoothing the S/N map; using polygons (r.m.s., source, zero-ing). Saving individual maps is not possible when using the multi-core processing (but see Sect. 5.6.1). As of 2024/03 it has not been tested yet in the case of: hit maps; cumulative maps[1].

5.6.1 Saving individual maps

As mentioned above, the option to save individual maps cannot be combined to the PIIC multi-core parallel processing. The reason is a conflict between `wrCumLast` (needed by the parallel processing) and `wrIndMaps`.

Nevertheless, if really necessary, the easy workaround is to perform the multi-core processing including only one scan per sub-list. In this way, each sub-process will produce a `.rgs` and `.rgw` map, to be used to generate the wished individual maps as ratio `rgs/rgw` (see also Sects. 4.13.1 and 4.13.4).

5.6.2 Comparison to single-core processing and advantages

The numerical accuracy of FITS files (used as i/o during the multi-processing approach) is of the order of 10^{-6} mJy. Within this accuracy, numerically there is no difference in the results of single-core and multi-core processing.

The only real advantage of this multi-core approach is the significant gain in execution time. This comes at the expenses of an increased complexity in the preparation of the scripts and scan lists, as well as of the bookkeeping of data products (see Sects. 5.3, 5.2 and 5.5). Splitting the scan lists in chunks needs special care in the case of null-maps, when different scans cover different regions of the sky.

Generally speaking, if the total time of the single-core processing of the given data set is of the order of hours, then the PIIC team thinks that there is no need to embark in the extra efforts of using this multi-core processing. On the other hand, if the total time needed to process the data set with one CPU core is of the order of days or weeks, then it is indeed worth it.

Since the multi-core processing can be so much faster than the single-core, the user can perform a larger number of iterations and check the real convergence of the total resulting flux of the source(s). In this way it's possible to better define the number of iterations necessary to obtain the best results. Similarly, a larger number of `blOrderOrig` values can be easily tested, in order to find the best

[1] for the cumulative maps option, an additional piece of code would be needed.

choice for the data set under analysis. See also notes in Sect. 3.2.6 for more considerations about `blOrderOrig`.

Chapter 6

PIIC-Pol: reduce NIKA2 polarimetry data

When the going gets tough,
puttanesca scomposta!

NIKA2 is equipped with a removable rotating half wave plate (HWP), in front of the light entrance window, and a cryogenic polarizer that splits the light into two perpendicularly-polarized beams that hit Array 1 and 3. In other words, the instrument is capable of Stokes polarimetry in the 1.2 mm band.

This short Chapter describes how to reduce NIKA2 polarimetry data using PIIC. The capability of producing Stokes I, Q, U maps is available starting from the 9th release of the software. From now on we will call this featured polarimetry pipeline *PIIC-Pol*. We also include some example scripts to naively compute and plot polarization degree maps and polarization angle vectors.

Starting from the 10th PIIC release, a script to correct the effect of instrumental polarization (IP, also known as instrumental leakage) is included in PIIC-Pol. To this aim, polarization observations of a non-polarized, point-like source (e.g. Uranus) need to be carried out during the NIKA2 polarimetry observing campaigns.

NOTE: in order to retrieve possible extended polarized signals, it is recommended to use PIIC-Pol with a *source polygon* (Sect. 4.5) and a *base range polygon* (Sect. 4.6). Section 6.11 describes an example of typical PIIC-Pol reduction.

6.1 Basic needed knowledge and suggested reads

In order to properly process and analyse NIKA2 polarimetry data, the user should be familiar with and know the basics of Stokes polarimetry carried out with a rotating HWP and a polarizer, the structure and principles of NIKA2 polarimetry setup, general millimeter ground-based astronomical observations, and possibly the advanced use of PIIC for total-power data.

As for the latter, it is recommended to be familiar at least with Chapters 1 to 4 of this PIIC tutorial. The principles of polarimetry with NIKA2 and of polarimetry in general will not be repeated here and can be found in the following publications:

- PhD thesis of Hamza Ajeddig (in english Ajeddig 2022)

- PhD thesis of Aina Andrianasolo (in french Andrianasolo 2019)
- PhD thesis of Alessia Ritacco (in english Ritacco 2016)
- PhD thesis of Nicolas Ponthieu (in french Ponthieu 2003)
- Results of the commissioning of NIKA2 polarimetry (Ajeddig et al. 2022)
- Results of the commissioning of NIKA polarimetry (Ritacco et al. 2017)
- HWP theory and construction: Savini et al. (2006); Pisano et al. (2016, and references therein)

The PhD theses are available on the NIKA2 web pages or on the `https://theses.hal.science` french theses portal.

6.2 Further details of NIKA2 polarimetry

The rotating HWP modulates the incoming signal, that is then split by the polariser in its two perpendicularly polarized components. Each 1.2 mm detector (Arrays 1 and 3) receive then one of the two beams produced in this way.

The sampling frequency of the NIKA2 acquisition electronics and software ($\nu_{\rm DAQ} \sim 47.7$ Hz) is designed to be an integer multiple of the HWP rotation frequency ($\nu_{\rm HWP} \sim 2.98$ Hz). In this way a full modulation cycle is sampled by 16 records along a scan's timeline. The corresponding 16 positions of the HWP are called *phases*. The rotation direction of the HWP is also fixed and defined by the acquisition software a priori.

If any of these instrumental properties (data sampling frequency, HWP rotation frequency, rotation direction) are changed, then PIIC-Pol might need to be adapted to the new setup, or at least briefly newly tested.

In the ideal case of perfect instrument and sky, the modulated incoming light consists of three main components:

$$S^k_{\rm Ar1}(t) = \frac{1}{2}\left(I + Q\cos\left(4\omega_t + 2\psi_t\right) + U\sin\left(4\omega_t + 2\psi_t\right)\right), \tag{6.1}$$

as seen by a KID k of Array 1. Here ω_t is the HWP angle at the given time t and ψ_t is the angle between the polarization zero-angle axis in the instrument and the one in the sky. On the other hand, a KID of Array 3 "sees":

$$S^k_{\rm Ar3}(t) = \frac{1}{2}\left(I - Q\cos\left(4\omega_t + 2\psi_t\right) - U\sin\left(4\omega_t + 2\psi_t\right)\right). \tag{6.2}$$

When working in Equatorial coordinates, the angle $\psi(t)$, between the instrument (Nasmyth) coordinates system and the sky, is given by:

$$\psi(t) = a(t) + \frac{\pi}{2} - \frac{76.2}{180}\pi - q(t), \tag{6.3}$$

where $a(t)$ is the elevation of the telescope at the time t, $q(t)$ is the parallactic angle, and the terms $\pi/2$ and $\frac{76.2}{180}\pi$ take into account the rotations induced by the optics (mirrors) of NIKA2. *PIIC-Pol*

allows to process the polarimetry data and produce Stokes maps only in the Equatorial coordinates system.

We remind the user that in the astronomical convention, angles are positive from North to East, and usually maps are represented with North up and East left. The same conventions apply here for PIIC-Pol.

6.3 Observing with NIKA2-Pol

A typical NIKA2-Pol observing campaign includes observations of the science targets, combined to observations of a "reference" point-like source of known polarization properties, to be used for the characterization of the instrumental polarization. At best, this reference source is a non-polarized target (e.g. Uranus), or a weakly polarised QSO if the former is not available in the sky.

A typical NIKA2-Pol session is as follows:

- pointing, focus scans, and pointing again, on the point-like "reference" source of known polarization properties.
- a map scan of the reference source (e.g. `calib_1scan` script);
- pointing on a point-like source close to the science target;
- a sequence of scans (maps) of the science targets, for a duration of roughly one hour;
- a new map scan of the reference source, taken with the same focus;
- a new focus sequence on the reference source, if needed, followed by a new pointing scan;
- a new map scan of the reference source, if the focus changed;
- back to the science target;
- and so on, continuing with the same cycle.

6.4 Operations performed by PIIC-Pol

The operations that PIIC-Pol performs on the NIKA2 polarimetry data are completely transparent to the users. By this it is meant that, from the user point of view, running PIIC-Pol does not require any special scripts or setup specific to polarimetry. The interface between pipeline and user is equivalent to the one seen for total power (i.e. not-polarized) data seen in the Chapters 2, 3 and 4.

Nevertheless, it is worth to briefly describe the operations performed by PIIC-Pol, specific to polarimetry data (see Sect. 3.3 for the case of total power). PIIC automatically detects if the scan under analysis was taken with the polarimetry setup and in such case switches to PIIC-Pol operations. *Importantly, all scans in the list must be either of polarimetry or total power. Mixing is not allowed and causes PIIC to stop.*

1. First of all, during the reading process, after computing the signal, some basic calculations are performed, such as deriving the position angle of the HWP at each record, $\omega(t)$, and the rotation angle to sky coordinates, $\psi(t)$.

2. Before flat-fielding, the parasitic signal, produced by the rotating HWP, is computed and subtracted from the timeline.

3. Then all operations follow in the same way as for total power.

4. In the case of the iterative mode, source maps for the three Stokes I, Q, U parameters are used, taking into account Eqs. 6.1 and 6.2.

5. Before gridding the timelines onto the final 2-D maps, the I, Q, U Stokes parameters are isolated via the process of de-modulation and lock-in. Additionally, 1-D re-gridding in time (analogous to a low-pass filtering in Fourier space) is possible.

6. Finally the I, Q, U maps and their respective weight maps are produced following the same gridding procedure as for total power (see Sect. 3.3.2).

6.4.1 Parasitic signal

The modulation of the light produced by the rotating HWP induces a so-called *parasitic signal* (known also as HWP *synchronous* signal) along the timeline. It was suggested that this might be due to impurities or imperfections in the HWP material (e.g. NIKA2-Pol commissioning team 2021; Andrianasolo 2019; Ritacco 2016). This parasite needs to be evaluated and subtracted from the timelines at the early stages of the data processing.

The computation and subtraction of the parasitic signal can be performed in three different ways:

a) compute a median of the signal for each of the 16 HWP phases over the length of whole scan (excluding turnovers between scans) and subtract it from each phase independently. This method is called BT0 and is the *default* choice in the PIIC-Pol pipeline.

b) a more sophisticated (but much slower) approach is to fit the periodic variation of the signal with a series of $\cos\omega_t$ and $\sin\omega_t$ harmonics up to the 7th order. This is done for each sub-scan independently. This method is called LFI, because it computes the coefficients of the terms of the harmonics series by means of a linear regression.

c) compute the series of $\cos\omega_t$ and $\sin\omega_t$ harmonics (see previous option) per sub-scan, also *excluding* the source, as "detected" in the *source map* (Sect. 4.14).

It is possible to switch between BT0 and LFI by setting the parameter `psLfi` from "no" (default) to "yes". The third method can be activated by setting the `psExSou` parameter to "yes" (regardless of `psLfi`). As usual, both parameters are found in the optional parameters script and shall be added to the main script in the custom setups section.

When performing the subtraction of the parasitic signal per sub-scan (2nd and 3rd methods), some important caveats need to be kept in mind:

- in the LFI case, the position of sources is not yet known, therefore they are not excluded from the computation of the parasitic signal. The impact of bright sources is therefore stronger than in the BT0 case (which operates over the whole scan at once).

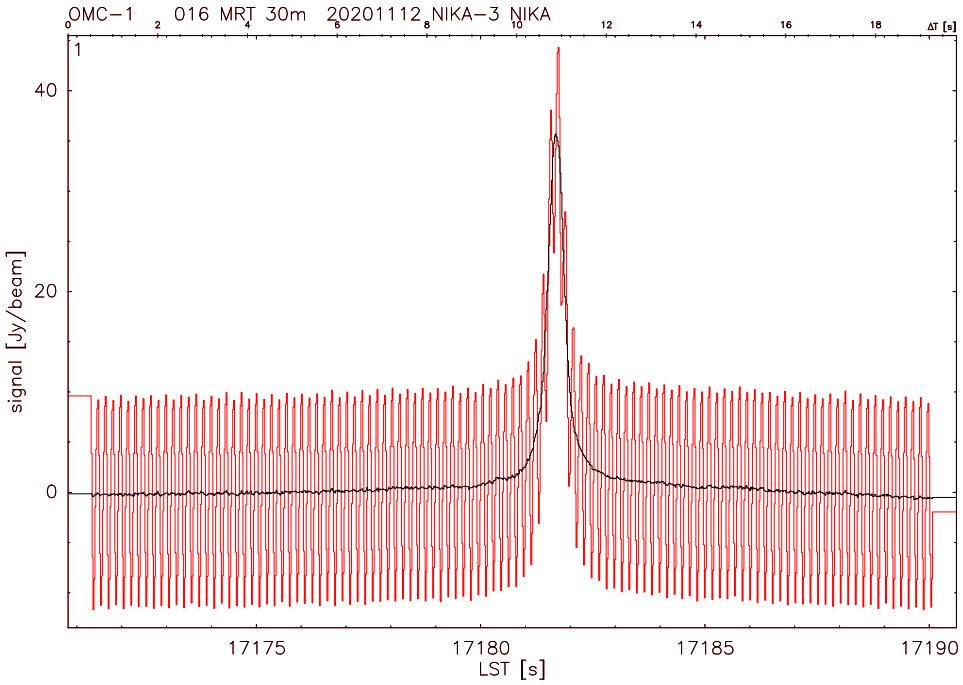


Figure 6.1: Subtraction of the HWP parasitic signal from the timeline of OMC-1. The red line represents the timeline with the oscillating parasite. The black line is the result of the subtraction.

- in the `psExSou` case, it is necessary that the maps are large enough in the scanning direction to allow for a sufficient number of source-free records (see Sect. 2.1), otherwise the baseline correction and even the source definition will be sub-optimal. The consequence is the appearance of the so-called "baseline effects" in the final maps, as well as the possible creation of spurious signals. Moreover, for this method to properly work, it is paramount that the data are taken with very stable atmospheric conditions and with very good behaviour of the telescope tracking (e.g. in low wind conditions).

Figure 6.1 shows a short portion of the timeline of a polarimetry scan targeting OMC-1, before and after removing the parasitic signal. In this case the amplitude of the parasitic signal is ~ 10 Jy.

6.4.2 Demodulation and lock-in

Starting from Eq. 6.1, the Stokes Q and U parameters are extracted from the signal using the process of *de-modulation* (see the references listed in Sect. 6.1). The timeline is multiplied once by $\cos(4\omega_t + 2\psi_t)$ and once by $\sin(4\omega_t + 2\psi_t)$. A small arithmetics (trigonometry) exercise shows that these operations produce one pure Q and one pure U term, among many others. For example, in the case of the multiplication by the cosine, keeping in mind that $\cos^2 x = (1 + \cos 2x)/2$ and $\sin 2x = 2 \cos x \sin x$, we obtain:

$$S^k_{\mathrm{Ar1}}(t) \times \cos(4\omega_t + 2\psi_y) = \frac{1}{2}\left(I \cos(4\omega_t + 2\psi_y) + \frac{Q}{2} + Q \cos(8\omega_t + 4\psi_y) + U \sin(8\omega_t + 4\psi_y)\right) \tag{6.4}$$

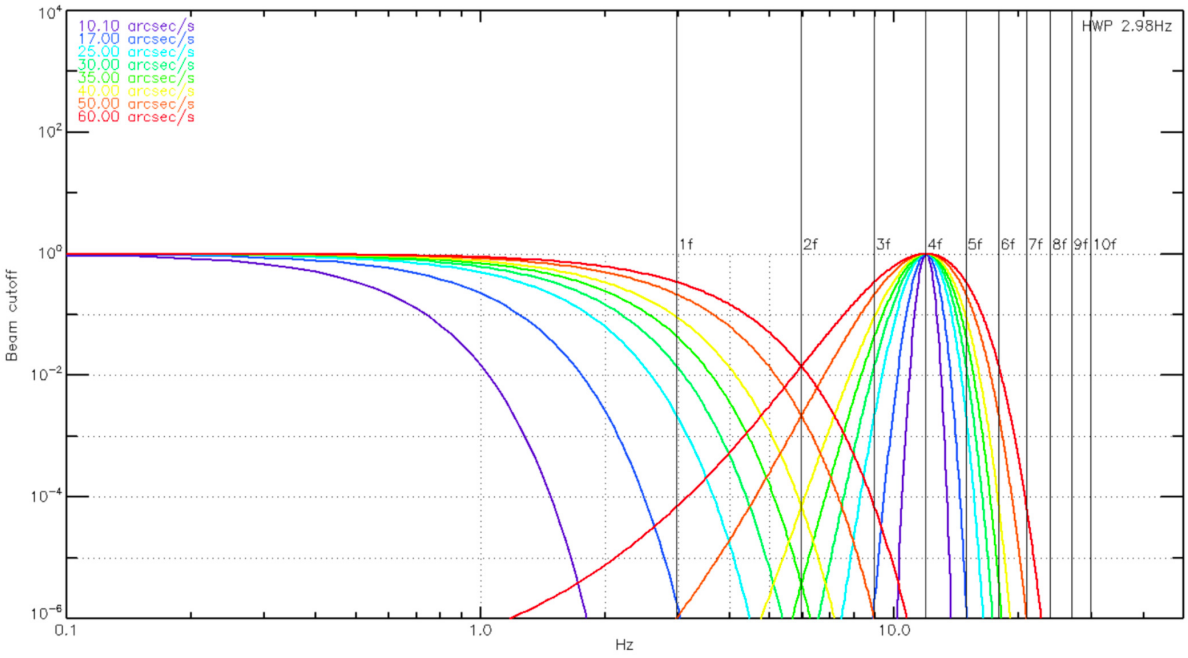


Figure 6.2: Lock-in procedure (courtesy of N. Ponthieu; see also Fig. 6.5 of Andrianasolo 2019). A Gaussian optical beam, coupled with a constant scanning speed, translates into a Gaussian filter for the timeline in Fourier space. For several typical simulated scanning speeds on the sky, the width of this effective bandpass varies. Considering a raw timeline, the band at low frequency contains the total intensity while the band around $4 \times \nu_{\rm HWP} \sim 12$ Hz contains the polarization information. After demodulation of the timeline, it is the opposite: Q or U lie at the lowest frequencies, while I is near 12 Hz. When applying a low-pass filter to the timeline, to isolate the low frequency component, it is paramount to minimise the overlap between the two bands, in order not to mix I to Q or U. This imposes stringent limits on the scanning speed. This Figure shows a simulation; for an analogous diagram based on real data, see Fig. 6.3 of Ritacco (2016).

A similar result (with some signs changed) is obtained when multiplying by sine. The original timeline, kept as is, gives instead the pure I Stokes parameter. In practice, three distinct timelines are now produced: $S^k(t)$, $S^k(t) \times \cos$, and $S^k(t) \times \sin$.

The pure Stokes terms must be isolated from all others. This can be done applying a low-pass filtering in Fourier space, with a cutoff frequency of $\sim 2 \times \nu_{\rm HWP}$ (see Fig. 6.2). Equivalently, the same operation can be performed with a one-dimentional re-gridding in time to a scale of $1/(2\nu_{\rm HWP})$. This holds because the pure I, Q, U signals are locked at the 4th harmonic of the HWP rotation, in these three distinct timelines, and are well separated at the 2nd harmonic (see next Sections).

6.4.3 Fourier low-pass filtering and scan speed limit

Figure 6.2 shows a simulation of the Fourier transform of a timeline, obtained varying the scan speed from 10 to 60 arcsec/s. In case of a timeline taken "as is", the low frequency component consists of the total intensity, while the one at $4 \times \nu_{\rm HWP}$ has the polarization information. When multiplying by cos

or sin, the opposite holds. A low-pass filtering in Fourier space therefore guarantees to produce a pure I, or Q or U timeline, avoiding any contamination by the other terms. Figure 6.2 demonstrates that $\sim 2\times\nu_{\rm HWP}$ is always the best cutoff frequency for filtering, *but* at faster scan speed the contamination of the pure I, Q, U by the other terms increases. Real data have larger contamination fractions with respect to this simulation.

There is one other important factor to take into account: the size of the beam. Adopting the usual 11.8 arcsec HPBW at 1.2 mm, and keeping in mind the current values of $\nu_{\rm HWP}$ and $\nu_{\rm DAQ}$, it turns out that $\sim 35-40$ arcsec/s is the limit scan speed to have a Nyquist sampling of the beam at each HWP phase.

Therefore, when observing in polarimetry, the adopted scan speed shall not exceed 35-40 arcsec/s, in order to properly sample the NIKA2 1.2 mm beam *and* to minimize the mutual contamination of the I,Q,U Stokes parameters.

6.4.4 One-dimensional gridding in time

PIIC-Pol does not work in Fourier space, but in real space. A Fourier low-pass filtering of the timelines removes the high frequency components of the signal. PIIC-Pol achieves this result by applying a one-dimensional re-gridding of the timeline along the time coordinate. It applies the same technique described for the two-dimensional gridding in Sect. 3.3.2. The ad-hoc kernel is designed so to re-grid to a time scale of $1/(2\nu_{\rm HWP})$.

It is worth to note, however, that after this 1-D time-gridding (or Fourier filtering) operation, the timelines are also gridded in 2-D onto the final maps, using a kernel designed on the NIKA2 beam. This 2-D gridding is also producing a similar effect of Fourier filtering. Because of the actual size of the beam and the scan speed (close to Nyquist), these two 1-D and 2-D gridding operations turn out to have a very similar frequency in time, with the consequence that the simple 2-D gridding in the map production already selects the pure I, Q, U timelines itself.

PIIC-Pol has been tested on NIKA2 polarimetry commissioning data with and without 1-D gridding in time and has produced equivalent results. Given the equivalence of effects of the 1-D gridding in time and the 2-D gridding to maps, by default the 1-D gridding in time is switched *off*. To activate it, change the value of the `filterInT` parameter, found in the optional sets script, from `no` to `yes`. Note, however, that the 1-D gridding in time alters significantly the weights properties and therefore the computation of noise and S/N maps might not be anymore straightforward when using it.

6.4.5 Iterative mode

When processing the NIKA2 polarimetry data with the iterative mode, by default PIIC-Pol applies the lock-in (demodulation) and, if enabled, the 1-D gridding in time at *all* iterations. Triple (IQU) source maps and r.m.s. polygons are computed using the Stokes I, Q and U maps and weight and are applied at the next iteration.

The final I, Q, U product cube is produced at each iteration. In previous implementations (up to PIIC v. 10.1), this was done by default only at the last iteration. Now the value of the `calcIQUeachIt` parameter is set to *yes* by default. Please, do not change it (otherwise the behavior of PIIC-Pol might be unpredictable).

6.5 Prepare your scripts

As mentioned before, the PIIC-Pol operations specific to the polarimetry data are transparent for the user. Therefore the data reduction script(s) to be used are the same as for the case of total power data (see Sects. 3.2.4 and 5.3).

We remind that the scrips are found in the `pro/` directory of the PIIC distribution and that the main script is called `mapTP_a2_template.piic`. When using the multi-core processing (parallel) approach, the main scripts to be used are two: `source_a2_MP_template.piic` and `source_setMPpar.piic`.

Optional parameters are to be found in `mapTPoptionalSets.piic`, as usual.

6.5.1 Extended faint signals and bright compact sources

The polarized emission of cosmic source can be rather faint, even in case of bright total power emission. Moreover one might expect an extended and smooth Stokes Q and U signal, difficult to detect.

Following the prescriptions described in Sects. 4.5 and 4.6, it is recommended to:

- In case of extended Q and U signal, define and adopt a *source polygon* encompassing the Stokes I emission, possibly generously, in order to avoid the subtraction of sky noise in the region and preserve the smooth faint polarized signal.

- In case of very bright Stokes I emission, define and adopt a *baserange* polygon, in order to avoid the rigion inside the polygon when computing the baseline correction, especially if the bright emission is compact.

6.5.2 Use of an input source map

In the case of PIIC-Pol, the *source map* is triple: each Stokes parameter (I, Q, and U) has its own source map (Sect. 6.6). In the iterative processing, at the end of each iteration three source maps are computed and are used in the next iteration, as in the case of total power.

The Q and U signal can be positive or negative, regardless of the nature of the Stokes I emission. The expected sign of the Stokes I (and total power) signal is specified by the user-defined parameter `souSign` (e.g., Sect. 2.5). During the computation of the Q and U source maps, both positive and negative signals are detected, regardless of the value of `souSign`.

If the processing stops and the user would like to continue later, starting from where it was interrupted, it is possible to use the last source maps produced as input (see Sect. 4.14.1). In PIIC-Pol, at odds with the total power case, the input `sbSource` name must be only the *root name* of the source maps and must *not* include the `.fits` extension.

For example, if the source maps are called `Crab_a13_...i50n50f51_I,Q,U.fits`, the value of `sbSource` shall be `Crab_a13_...i50n50f51`.

Section 4.14 describes the principles of the source maps for the case of total power data.

6.5.3 Use a custom r.m.s. polygon

To compute the source maps (Sect. 6.5.2), PIIC-Pol automatically defines an r.m.s. polygon for each Stokes parameter (IQU, see also Sects. 3.4.6 and 6.6).

Alternatively, the user can manually define their favorite r.m.s. polygon and specify it as input, by using the `polRMSeq` parameter (Sect. 4.11). While the automatically-defined r.m.s. polygon is triple, the user-defined is unique: in other words the same polygon will be used to compute the three source maps of the Stokes I, Q and U parameters.

6.6 Products

Running PIIC-Pol is straightforward, if the user has already experience with PIIC. Simply proceed as described in Sect. 3.3 or 5.4, for single- or multi-core, respectively. The few differences are described in Sect. 6.5.

PIIC-Pol has more final products than the total power case. PIIC-Pol produces two cubes as result, one consisting of the I, Q and U maps, the other containing their respective weight maps. *The cubes are structured so that the first plane is the I map, the second is the Stokes Q map and the third is U.* The same structure has the weights cube. It is important to note that these are *not* multi-extension FITS files. They are, instead, single extension FITS with three planes.

Moreover, three source maps are produced, one per each Stokes parameter. The same holds for the r.m.s. polygons.

For example, in the case of the Crab Nebula (the Superantennae is only weakly polarized), reduced with 50 iterations, typical filenames could be:

```
1) Crab_a13_...i50n50.fits
2) Crab_a13_...i50n50rgw.fits
3) Crab_a13_...i50n50f51_I.fits
   Crab_a13_...i50n50f51_Q.fits
   Crab_a13_...i50n50f51_U.fits
4) Crab_a13_...i50n50_I.pol
   Crab_a13_...i50n50_Q.pol
   Crab_a13_...i50n50_U.pol
```

The three dots omit part of the filenames; their content reflects the setup chosen in the data reduction script. The four products are: *1)* the signal cube; *2)* the weights cube; *3)* the source maps; *4)* the associated r.m.s. polygons computed automatically (if not otherwise specified).

As previously mentioned, during the iterative mode the default product of the intermediate iterations are total power maps and weights. Only the last iteration produces the cubes, unless otherwise specified.

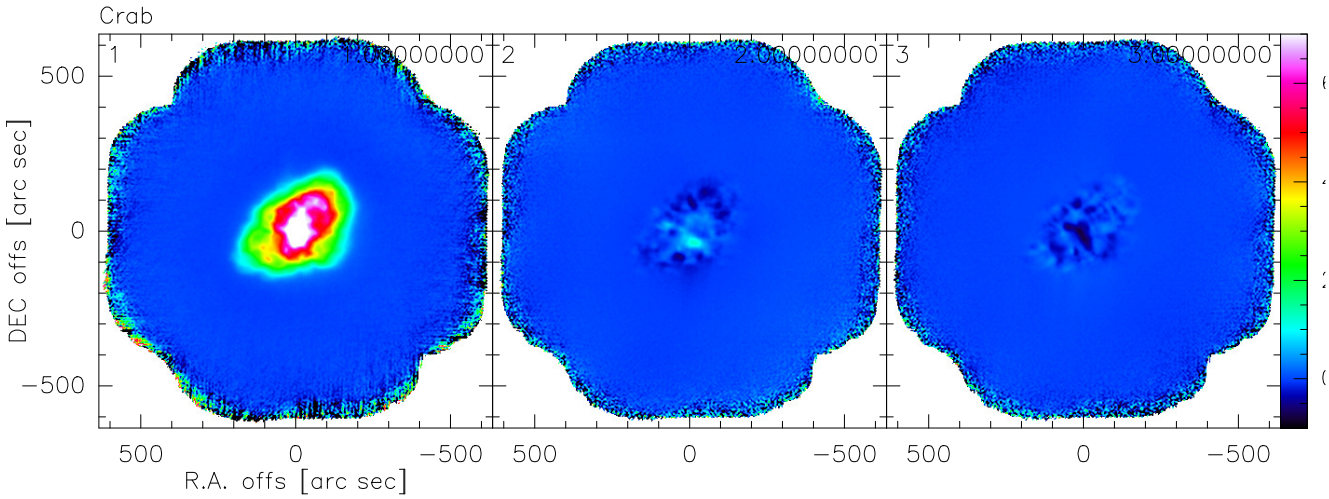


Figure 6.3: Displaying the I, Q, U (*left, center, right*) cube with the simple PIIC command `plot`.

6.7 Basic post-reduction operations

6.7.1 Read, plot, split FITS cubes with PIIC

PIIC reads and plots cubes in the same way as single-plane images (Fig. 6.3). If needed, it is possible to take only one plane of the cube. We invite the users to experiment with the following commands:

```
read Crab_a13_...i50n50.fits     ! read cube
plot sca lin -50 250             ! plot cube
take plane 1                     ! select only the 1st plane
write Crab_a13_...i50n50_I.fits  ! write plane 1 on a FITS file
plot sca lin -50 250             ! plot only plane 1
```

and similarly for the second and third planes (it is necessary to reload the cube).

6.7.2 Produce polarimetry source maps a posteriori

The script `creaSource_IQU.piic`, included in the PIIC `pro/` repository, is designed to produce a triple IQU source map, starting from the final cube, the rgw cube and a noise r.m.s. polygon (see also Sect. 4.14.3). Use it as follows:

```
@creaSource_IQU mainName polName snrLevel souSign smSouPar
```

As in the case of the user-define r.m.s. polygon, the same polygon is applied to all three Stokes I, Q and U maps.

6.7.3 Produce polarimetry r.m.s. and S/N maps

The PIIC `pro/` repository includes the scripts `creaRMS_IQU.piic` and `creaSNR_IQU.piic`, to be used for the computation of the polarimetry r.m.s. and S/N ratio maps (see also Sect. 3.4.4). Use them as follows:

```
@creaRMS_IQU mainName polNameI polNameQ polNameU
@creaSNR_IQU mainName polNameI polNameQ polNameU
```

In this cases, three r.m.s. polygon are needed, e.g. those produced automatically by PIIC-Pol during the iterative data reduction (or one can also use one single custom polygon three times). The result are the files `mainName_rms.fits` and `mainName_snr.fits`.

6.8 Plot polarization degree and angle

Given the Stokes parameters I, Q and U, the polarization intensity, polarization degree and polarization angle are given by:

$$I_{\rm pol} = \sqrt{Q^2+U^2} \tag{6.5}$$

$$P_{\rm pol} = \frac{\sqrt{Q^2+U^2}}{I} \tag{6.6}$$

$$\vartheta_{\rm pol} = \frac{1}{2}\arctan\left(\frac{U}{Q}\right) \tag{6.7}$$

taking care that $\vartheta_{\rm pol}$ is defined in the range $[-\frac{\pi}{2},+\frac{\pi}{2}]$, and therefore the arctangent function used in the computation needs to be defined in the range $[-\pi,+\pi]$ (i.e. use the function `atan2`, originally defined in Fortran in 1961, or an analogous one).

When computing the angles, one might want to re-bin the maps in order to have a higher S/N ratio per pixel. One simple way to do this, is to box-average the maps. The PIIC command `bsm` does this. For example:

```
read Crab_a13_...i50n50.fits          ! read cube
bsm 5 5                               ! apply a 5x5 pixels box avg
write Crab_a13_...i50n50_bsm5.fits    ! write result on disk
```

The resulting map (or cube) has bigger pixels than the original, the size being scaled by the chosen `bsm` values (5×5 pixels in the example above). The number of pixels in the output map is smaller than in the input by the same amount.

Three GREG scripts are included in the PIIC distribution, designed to plot the polarization quantities. They are mainly meant for display purposes only and need to be adapted to the given dataset under analysis. The first, `plPol1.greg` displays the I map and overplots polarization vectors on it. The vectors are represented as simple segments of length proportional to $P_{\rm pol}$ and orientation given by $\vartheta_{\rm pol}$. The second, `plPol4.greg` in addition displays also $I_{\rm pol}$, $P_{\rm pol}$ and $\vartheta_{\rm pol}$ maps. The third, `plPol3.greg`, displays the $I_{\rm pol}$, overlaying Stokes I contours and polarization vectors. As inputs, all three scripts need the I, Q, U cube produced by PIIC-Pol and a box-averaged version (see above) of the same cube. An example obtained with commissioning data of the Crab Nebula is shown in Fig. 6.4.

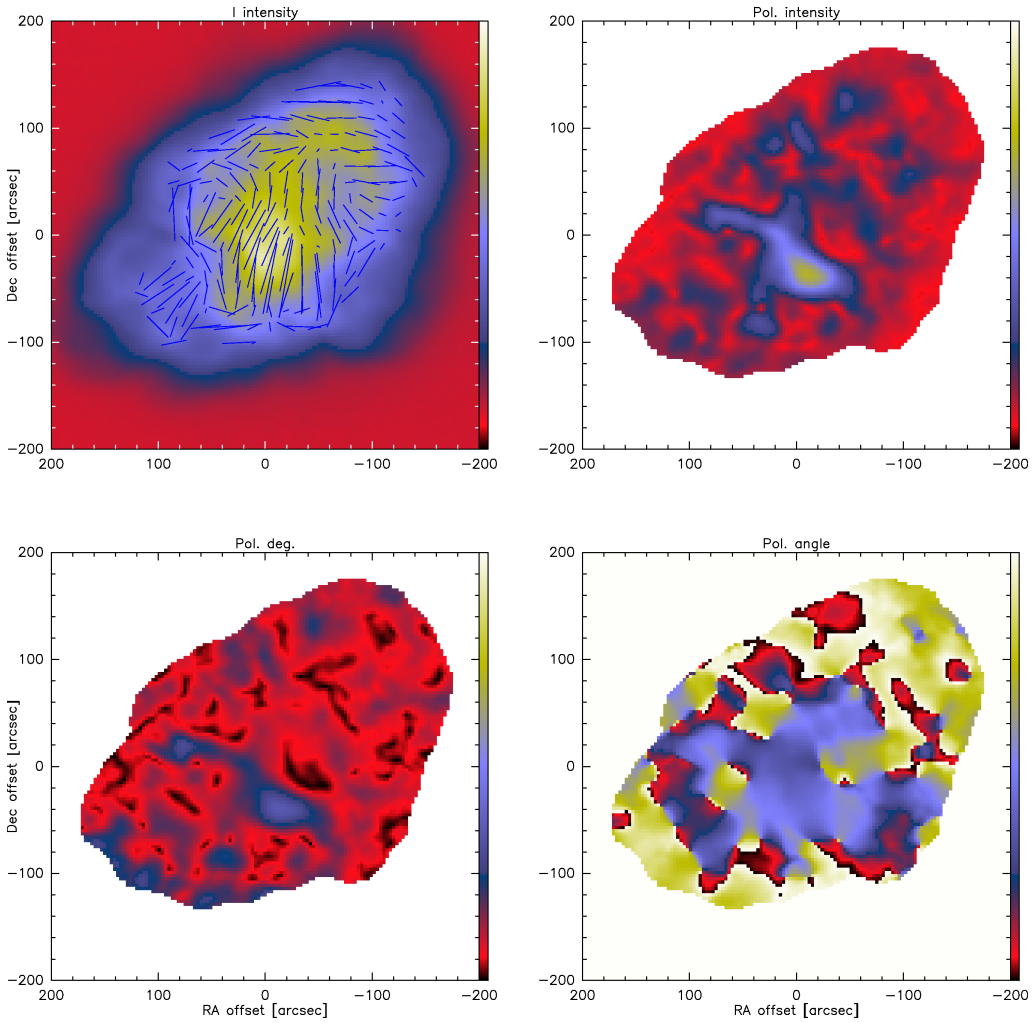


Figure 6.4: Example of the result of the script `plPol4.greg` with Crab Nebula NIKA2 polarimetry commissioning data. See Ritacco et al. (2018) for a comparison to the literature. From left to right and top to bottom, the four panels show: the I map with polarization vectors overlaid; the polarization intensity ($I_{\rm pol}$) map; the polarization degree ($P_{\rm pol}$) map; the polarization angle ($\vartheta_{\rm pol}$) map. Color scales values are omitted on purpose, for data proprietary reasons.

6.9 Correction of the instrumental polarization (a.k.a. leakage)

The observed I, Q, U signals received and recorded by NIKA2, can be expressed by:

$$\begin{aligned} I_{\rm obs} &= \mathcal{B}_I * I_{\rm src} \\ Q_{\rm obs} &= \mathcal{B}_I * Q_{\rm src} + \mathcal{L}_Q * I_{\rm src} \\ U_{\rm obs} &= \mathcal{B}_I * U_{\rm src} + \mathcal{L}_U * I_{\rm src} \end{aligned} \tag{6.8}$$

after correcting all the main atmospheric and instrumental effects described in the previous Chapters, and ignoring the noise terms. In these equations, $I_{\rm src}$, $Q_{\rm src}$ and $U_{\rm src}$ are the real Stokes I, Q, U signal

of the target, $\mathcal{B}_I$ is the telescope+NIKA2 beam, and $\mathcal{L}_Q$, $\mathcal{L}_U$ are the instrumental leakage terms (that implicitly include the beam too). The symbol $*$ denotes a convolution. The goal is to remove the additive instrumental contribution from the Stokes signal.

The polarization leakage can be characterized by observing a knowingly non-polarized, point-like source, in observing conditions as close as possible to those of the science target. We label this source "*k*". In our specific case, a natural choice is Uranus, that is a point-like source for NIKA2. The leakage terms in Eq. 6.8), to be subtracted from the observed $Q_{\rm obs}$, $U_{\rm obs}$ signals, can then be evaluated as:

$$\begin{aligned} \mathcal{L}_Q * I_{\rm src} &= \mathcal{F}^{-1}\left(\frac{\mathcal{F}(Q_{\rm k})}{\mathcal{F}(I_{\rm k})}\mathcal{F}\left(I_{\rm src}\right)\right), \\ \mathcal{L}_U * I_{\rm src} &= \mathcal{F}^{-1}\left(\frac{\mathcal{F}(U_{\rm k})}{\mathcal{F}(I_{\rm k})}\mathcal{F}\left(I_{\rm src}\right)\right), \end{aligned} \tag{6.9}$$

where $I_{\rm k}$ is nothing else than $\mathcal{B}_I$ in the previous Equation (because source *k* is point-like by definition), and the symbol $\mathcal{F}$ denotes the Fourier transform. See Andrianasolo (2019) and Ajeddig (2022) for more details.

The script `corr_instr_pol.piic` performs these calculations and produces the final I, Q, U cube, with the instrumental polarization subtracted. In what follows, we will call the non-polarized point-like source indifferently "*k*ernel", "lea*k*age pattern", or some combinations of these.

6.9.1 Preparing the data

A typical NIKA2-Pol data set consist of a series of scans, observed at different times and possibly during different days. In the case of big data sets, the target might have been observed across several months, if not years. Consequently, these numerous scans have hardly been observed at similar parallactic angles, elevations, atmospheric or instrumental conditions. Similarly, the observations of the non-polarized source – to be used to evaluate the instrumental leakage – must span a range of different conditions (Elevation, parallactic angle, focus, atmospheric opacity, season, etc.), ideally as close as possible to the conditions of the scientific target. Section 6.3 describes a typical NIKA2-Pol observing session.

Therefore the correction of the instrumental polarization needs to be performed per scan. The instrumental leakage has to be measured for each individual scan and subtracted from it, before building the final, combined map. To this aim, the standard procedure to follow is:

1. perform the iterative PIIC-Pol data reduction of the science target as usual (see previous Sections), using N iterations, and producing the final combined map, the `rgw` weights and the associated *source map*.

2. run again PIIC-Pol, with zero iterations, using the *source map* produced before, and writing individual maps (cubes) instead of the final map (see Sect. 4.16).

3. perform a standard PIIC-Pol data reduction of the non-polarized, point-like source (observed with HWP, i.e. with NIKA2-Pol), setting the parameter `compLeak` to *yes* (see below). This reduction must use one scan and one array only (because each scan has different observing conditions in general). It might be needed to reduce several scans of the non-polarized point-like source, individually and independently, depending on how large a spread of observing conditions is covered by the science target.

The products are naturally stored in the sub-directory `red/`.

The `compLeak` parameter is found in the `mapTPoptionalSets.piic` script and can be added to the main data reduction script after the call to the optional sets, as usual. Setting it to *yes* indicates that the result of the data reduction will be used as instrumental polarization "*kernel*". In this case, PIIC will not apply the rotation of the Q and U signal in the sky (Eq. 6.3). The map will still be in Equatorial coordinates, but not the signal. In this way, it is simpler to then rotate this kernel to the ψ angle of the science target. It is important to stress that `compLeak` *yes* produces a map suitable only for the aim of correcting the instrumental leakage from the science maps with the dedicated script, and *not* for scientific or any other purposes.

6.9.2 Preparation of the script

The script `corr_instr_pol.piic` is found in the `pro/` directory in the PIIC distribution and shall be copied to the working directory for ad-hoc editing. It needs four inputs:

a) the name of the scientific target (e.g. Crab);

b) a list of maps of the individual scans (I,Q,U cubes), as produced in item *2)* above and stored in the `red/` sub-directory;

c) a list of the `rgw` maps associated to the individual maps, stored in the same `red/` sub-directory;

d) the instrumental polarization pattern (I,Q,U cube, *one scan only*, i.e. the result of item *3)* above, also stored in the `red/` sub-directory.

Additionally, few parameters in the script might need to be adapted to the specific data set. They are easily found in the script and come with comments, therefore a detailed description is omitted here:

i) the inner and outer radii [arcsec] of the Tukey taper mask for the science target in real space.

ii) the inner and outer radii [arcsec] of the Tukey taper mask for the leakage kernel in real space; this is already optimized for a point-like source and won't require significant changes;

iii) the inner and outer radii [m] of the Tukey taper mask for the leakage kernel in Fourier space; already optimized for a point-like source and won't require significant changes;

If several kernels are needed, further adaptation of the script to the specific data set is needed (Sect. 6.9.5).

6.9.3 Execution of the script

The script is called specifying the four inputs in the command line:

```
@ corr_instr_pol source_name list_of_indiv_cubes list_of_rgw filename_kernel
```

For each scan in the list of individual science maps, the script rotates the leakage pattern to the same angle in the sky of the science map (Eq. 6.3), applying the rotation both to the geometry of the kernel map and to its signal. The result is transferred to a CUBE[1] script dedicated to perform the operations in Fourier space. The use of CUBE is fully transparent to the user.

Both the instrumental polarization kernel and the science maps are masked in real space to attenuate the discontinuities at the edge of the maps applying a circular Tukey window:

$$\begin{aligned} W &= 1.0 && \text{if } (x-x_0)^2+(y-y_0)^2 \le R_1^2, \\ W &= \tfrac{1}{2}\left[1-\cos\left(\pi\tfrac{R_2-\sqrt{(x-x_0)^2+(y-y_0)^2}}{R_2-R_1}\right)\right] && \text{if } R_1^2 < (x-x_0)^2+(y-y_0)^2 \le R_2^2, \\ W &= 0.0 && \text{if } (x-x_0)^2+(y-y_0)^2 > R_2^2, \end{aligned} \tag{6.10}$$

where R_1 and R_2 are the inner and outer radii of the taper mentioned before, and (x_0, y_0) is the center of the map. Finally Eq. 6.9 is evaluated, taking care to apply again a Tukey window taper to the de-convolved kernels[2] in Fourier space, so to avoid divergence in the outer regions.

6.9.4 Output

The final result of the `corr_instr_pol.piic` script is a cube that contains the Stokes I, Q, U maps of the scientific target, with Q and U corrected for instrumental polarization. It is stored in the `red/` sub-directory and is called `srcName_IQU_leakage_corrected.fits`.

6.9.5 Caveats

a) The `corr_instr_pol.piic` script works with one single kernel and applies it to all the individual maps in the input list. If several instrumental polarization patterns are needed – as is likely the case, when dealing with datasets observed over several days, at different elevations, in different environmental and instrumental conditions – then the list of input scans needs to be split in sub-lists, each containing all and only those individual maps to which a common kernel can be applied, and the script needs to be used on each sub-list independently. Moreover, the last part of the script (combining the corrected cubes back together) shall be used on all scans together, after correcting the sub-lists with their respective leakage patterns.

b) The script is designed to work on maps of individual maps obtained with the `wrIndMaps` option (see Sect. 4.16) and therefore uses Eq. 3.1. If the user would like to use cumulative maps (Sect. 4.13.1) of subsets of data (e.g. to increase the S/N ratio of the single maps), the last part of the `corr_instr_pol.piic` combining all maps together needs to be adapted. Compare to the script `creaAver.piic` (Sect. 4.13.2 and Eq. 4.7) to determine how to proceed[3].

[1] CUBE is a new GILDAS component, not to be confused with the data cubes we've been talking about so far. Special thanks to J. Pety for the help with this part of the procedure.

[2] $\frac{\mathcal{F}(Q_k)}{\mathcal{F}(I_k)}$ and $\frac{\mathcal{F}(U_k)}{\mathcal{F}(I_k)}$

[3] the user who reaches this stage has become an expert of PIIC data processing and thus easily understands the necessary changes

c) The `corr_instr_pol.piic` produces several intermediate products for each individual scan and might need a lot of disk space in the case of large datasets.

6.10 Correction of the instrumental polarization using QSOs

In absence of a non-polarized sources (e.g., a planet) in the sky at the same time as the science target, or at a similar Elevation, one might use a quasi stellar object (QSO) or another point-like source to characterize the instrumental polarization affecting the science data. This approach is subject to the caveats that QSOs are known to have their own intrinsic polarization *and* to be variable over relatively short timescales (few days/weeks).

In order to cope with these limitations, two options can be foreseen:

i) at the time of observation of the science targets, no planet (or non-polarized source) is available in the sky, therefore it is necessary to use a QSO, *but* at some other time both the same QSO and a planet are present in the observable sky at the same time. In this case, one can use an observation of the planet and the QSO taken at a different time to characterize the intrinsic polarization of the QSO, and then subtract it from the data of the same QSO taken together with the science target. Generally speaking, it is sufficient that such a parallel observation of QSO+planet is carried out within few days from the science+QSO observation, because of the QSO typical variability period of few days, but it is preferable to have them taken during the same day, or as close as possible in time.

ii) during the observing week(s) the chosen QSO is never in the sky at the same time of a planet (or non-polarized source). In this case, it is possible to retrieve the intrinsic polarization properties of the QSO from one of the ongoing monitoring programs carried out with the IRAM 30 m telescope (e.g., Agudo et al. 2018a) or by other facilities (e.g., the SMA), and use them to derive the instrumental polarization from the QSO scans. If this is the chosen approach, please ask the NIKA2 astronomer on duty and the NIKA2 pool manager which QSOs are regularly monitored.

Generally speaking, the procedure to characterize the instrumental polarization using a QSO instead of a non-polarized target is very similar to the one described in Sect. 6.9, with some natural differences related to the treatment of the QSO data. In the following sub-Sections we will describe them. It is taken for granted that the user is already familiar with the standard procedure and with the content of Sect. 6.9.

6.10.1 Case i) use a combination of a QSO and a non-polarized source

The general idea, in this case, is to have a set of QSO and non-polarized source data, taken consecutively with the same telescope focus, obtained within few hours/days from the science+QSO main data set. Their focus does not need to be the same as for the science+QSO data. In principle the Elevation can also be different with respect to the science+QSO scans, but it is suggested to avoid too big a difference in Elevation.

In this context, a typical NIKA2-Pol observing session would consist of:

- pointing, focus, (and pointing) on a bright QSO, possibly close to the science target, or at least at a similar elevation[4].
- a calibration scan of the QSO.
- pointing on a source close to the science target.
- the sequence of science scans (between one and one and a half hours, or longer if the focus and the sky conditions are very stable).
- pointing and calibration scan of the QSO (without changing focus).
- repeat.
- at a different time of the observing day, or even on a different day during the same week, a calibration scan of the same QSO and of a non-polarized source (e.g., a planet) have also been taken. These two scans shall be consecutive, have the same focus, and possibly a similar elevation.

It is important to bracket the science target observations between two IP calibrator scans (a QSO in this case). The first QSO calibration scan will be used to correct the IP from the first half of the science scans, following the first QSO scan; the second will be applied to the second half, preceding the second QSO scan. A similar observing and calibration strategy shall be adopted if the IP calibrator is a planet or a non-polarized source.

The data shall be prepared in the following way:

1. perform the iterative PIIC-Pol data reduction of the science target as usual, using N iterations, and producing the final combined map, the `rgw` weights and the associated *source map*.
2. run again PIIC-Pol, with zero iterations, using the *source map* produced before, and writing individual maps (cubes) instead of the final map (see Sect. 4.16).
3. perform a standard PIIC-Pol data reduction of the QSO scans observed together with the science target, setting the parameter `compLeak` to *yes*. This reduction must use one scan and one array only (because each scan has different observing conditions in general). Take care to fix the map half-size using the `eqExtrAS` parameter (e.g., to 400 arcsec). This will be called "QSO1" in what follows.
4. perform a standard PIIC-Pol data reduction of the QSO scans observed together with the non-polarized source (planet), setting the parameter `compLeak` to *no* (!). Also in this case, reduce one scan and one array at a time and take care to fix the map half-size to the same `eqExtrAS` value as in the previous item (e.g., to 400 arcsec). This will be called "QSO2" in what follows.
5. perform a standard PIIC-Pol data reduction of the non-polarized, point-like source (planet), setting the parameter `compLeak` to *yes*. Reduce one scan and one array at a time. This will be the IP (leakage) pattern used to correct the QSO2 scans.

[4]we keep stressing that the elevation ought to be similar, because the telescope's main dish is deformed by gravity and its shape is in a "neutral" state at El∼50 deg. Above and below 50 deg, the astigmatism of the telescope affects the beam shape and focus in two orthogonal directions. And the instrumental polarization pattern strongly depends on focus and beam shape.

All products shall naturally be stored in the sub-directory `red/`.

The correction of the IP using the QSO1, QSO2 and non-polarized source scans is performed by the script `corr_instr_pol_QSO_and_NONpol.piic`, stored in the `pro/` directory of the PIIC distribution. The script needs to be copied to the working directory for a possible ad-hoc editing. Six inputs are needed:

a) the name of the scientific target (e.g. Crab);

b) a list of maps of the individual scans (I,Q,U cubes), as produced in item *2)* above and stored in the `red/` sub-directory;

c) a list of the `rgw` maps associated to the individual maps, stored in the same `red/` sub-directory;

d) the QSO1 I,Q,U cube (one scan and one NIKA2 Array only, item 3) above) of the QSO taken together with the science scans, stored in the `red/` sub-directory;

e) the QSO2 I,Q,U cube (one scan and one NIKA2 Array only, item 4) above) of the *same* QSO taken together the non-polarized source, stored in the `red/` sub-directory;

f) the instrumental polarization pattern (I,Q,U cube, *one scan and one Array only*, obtained from the non-polarized source observed together with QSO2 (i.e., the result of item *5)* above), also stored in the `red/` sub-directory.

Additionally, few parameters (e.g., the Tukey taper mask radii) in the script might need to be adapted to the specific data set, as described in Sect. 6.9.

The script is called specifying the six inputs in the command line:

```
@ corr_instr_pol_QSO_and_NONpol a b c d e f    ! see above
```

Before performing the actual corrections (Sect. 6.9.3), the script uses the IP pattern obtained from the non-polarized source (items 3) and f) above) to correct the QSO2 scan and obtain a I,Q,U cube of the intrinsic polarization of the QSO. This intrinsic polarization cube is by construction in equatorial coordinates, that is the polarization angle and signal are correctly oriented in the sky. Subsequently, the script subtracts it from the QSO1 I,Q,U cube, thus obtaining the IP pattern associated to the science scans.

Once these preliminary steps are ready, the IP correction of the individual science I,Q,U cubes is performed in the same way as for the case when science and non-polarized point-like source (planet) are observable at the same time (Sect. 6.9.3), taking care to properly account for the different parallactic angles and elevations of each scan.

The final result of the `corr_instr_pol_QSO_and_NONpol.piic` script is a cube that contains the Stokes I, Q, U maps of the scientific target, with Q and U corrected for instrumental polarization. It is stored in the `red/` sub-directory and is called `srcName_IQU_leakage_corrected.fits`.

The same caveats described in Sect. 6.9.5 apply also to this QSO method. Moreover, it is important to point out that, if the science+QSO1 and the QSO2+planet data sets have been taken with a large gap in time (of the order of weeks), or with very different elevations or sky conditions, it is possible that the IP correction of the science scans will be less accurate than in the case with science+planet available simultaneously. Moral of the story: we do our best and we are grateful to Mother Nature for her gifts, but there are unfortunately some practical limits.

It is worth noting that if the non-polarized point-like source (planet) is available in the sky at the same time as the science target, at an elevation not too different with respect to the one of the science scans, it is preferable to use it for the direct IP correction of the science scans. The QSO-based method is preferred when at the time when the science scans are taken, no non-polarized source (planet) is available in the sky at a not-too-different elevation. It can happen that in the first part of the science target observations only the QSO is available, and in the second part both the QSO and the non-polarized source (planet) are. In this latter case, a mixed approach can be followed for different batches of science scans, and the non-polarized source can also be used to build the intrinsic polarization cube of the QSO.

6.10.2 Case ii) use an independent monitoring of the QSO polarization

When only QSO scans are available for the characterization of IP, the procedure is slightly different with respect to the one described in the previous Section. In this case, it is necessary to know the intrinsic polarization degree and angle of the QSO from an external source of information. This could be for example, a QSO monitoring campaign. It is important that the monitoring data are taken within few days from the NIKA2 observations and roughly at the same frequency (260 GHz, that a wavelength of is 1.2 mm). Several monitoring programs are ongoing at the IRAM 30m telescope using EMIR/XPOL[56] (Agudo et al. 2018a,b; Thum et al. 2018; Kam et al. 2025; Myserlis et al. 2025), NOEMA, the SMA[7], and ALMA[8]. In case more information is needed, feel free to contact I. Myserlis at IRAM.

Then, the intrinsic Stokes Q and U maps of the QSO can be obtained from the observed Stokes I using the relations:

$$Q = p\,I\,\cos 2\vartheta \tag{6.11}$$

$$U = p\,I\,\sin 2\vartheta \tag{6.12}$$

where the observed Stokes I is not affected by leakage, p is the intrinsic fractional polarization degree (1=100%) of the QSO, and ϑ is its intrinsic polarization angle in the sky.

In this context, a typical NIKA2-Pol observation session consists of:

- pointing, focus, (and pointing) on a bright QSO, possibly close to the science target, or at least at a similar elevation.
- a calibration scan of the QSO.
- pointing on a source close to the science target.
- the sequence of science scans (between one and one and a half hours, or longer if the focus and the sky conditions are very stable).
- pointing and calibration scan of the QSO (without changing focus).

[5]XPOL observations during NIKA2 pools: `https://publicwiki.iram.es/XPOLobsDuringNIKA2pools`

[6]`https://polami.iaa.es/`

[7]SMA list of calibrators: `http://sma1.sma.hawaii.edu/callist/callist.html`; see also page 6-9 of `https://lweb.cfa.harvard.edu/sma/Newsletters/pdfFiles/SMA_NewsJan2024.pdf`

[8]AMAPOLA, `https://www.alma.cl/~skameno/AMAPOLA/`

- repeat, changing focus if necessary.

As in the previous cases (using a non-polarized source, or a combination of a QSO and a non-polarized source), it is important to bracket the science target observations between two IP calibrator scans (a QSO in this case).

The data shall be prepared in the following way:

1. perform the iterative PIIC-Pol data reduction of the science target as usual, using N iterations, and producing the final combined map, the `rgw` weights and the associated *source map*.
2. run again PIIC-Pol, with zero iterations, using the *source map* produced before, and writing individual maps (cubes) instead of the final map (see Sect. 4.16).
3. perform a standard PIIC-Pol data reduction of the QSO scans observed together with the science target, setting the parameter `compLeak` to *yes*. This reduction must use one scan and one array only (because each scan has different observing conditions in general). This will be called "QSO1" in what follows.

All products shall naturally be stored in the sub-directory `red/`.

The correction of the IP using the QSO1 with known intrinsic properties is performed by the script `corr_instr_pol_QSO_known.piic`, stored in the `pro/` directory of the PIIC distribution. The script needs to be copied to the working directory for a possible ad-hoc editing. Six inputs are needed:

a) the name of the scientific target (e.g. Crab);
b) a list of maps of the individual scans (I,Q,U cubes), as produced in item *2)* above and stored in the `red/` sub-directory;
c) a list of the `rgw` maps associated to the individual maps, stored in the same `red/` sub-directory;
d) the QSO1 I,Q,U cube (one scan and one NIKA2 Array only, item 3) above) of the QSO taken together with the science scans, stored in the `red/` sub-directory;
e) the known intrinsic linear polarization degree of the QSO, in relative units (i.e., 1=100
f) the known intrinsic polarization angle in the sky, in degrees.

Additionally, few parameters (e.g., the Tukey taper mask radii) in the script might need to be adapted to the specific data set, as described in Sect. 6.9.

The script is called specifying the six inputs in the command line:

```
@ corr_instr_pol_QSO_and_NONpol a b c d e f    ! see above
```

The main difference with respect to the previous method (Sect. **??**) is that the intrinsic polarization cube of the QSO and the IP pattern are based on the known polarization degree and angel obtained externally, rather than on a parallel NIKA2 observation of the same QSO and a non-polarized source. All subsequent operations are equivalent.

The same caveats described for the previous methods apply also now. It is important to repeat that the known polarization degree and angle of the QSO must be temporally close to the NIKA2 observations and must be at a similar frequency.

6.11 Example of a typical PIIC-Pol data reduction

The PIIC-Pol data reduction is the most challenging task that the NIKA2 user might face when dealing with their data sets. We reiterate that familiarity with all the previous Chapters of this user manual is necessary to take on this challenge.

While the main steps of PIIC-Pol and NIKA-Pol are well established (see the previous Sections of this Chapter), some aspects are still in development and the PIIC and NIKA2 teams are happy to receive users' feedback, suggestions, ideas and doubts, to the aim of perfecting this data reduction and its final results.

On the data processing point of view, the case of extended polarized source is the most complex and requires particular care and patience in defining and finding the best PIIC-Pol setup. In this Section, an example flow of operations is schematically described, to be used as an initial guidance for the PIIC-Pol reduction. This sequence of items has no ambition to be exhaustive and is surely subject to future improvements. We invite the users to send us their inputs.

A typical example of PIIC-Pol data processing proceeds through the following steps:

1. perform a quick reduction of a limited number of scans:
   - use zero (0) iterations;
   - use a low base line order (e.g. `blOrderOrig` 2);
   - do not use any specific polygons (zeroing, source, base range), because no information about the structure of the source is available at this time.
2. analyse the result:
   - display the Stokes I map;
   - define a zeroing polygon or radius (Sect. 4.8.3);
   - define a source polygon, encompassing all of you source(s) (Sect. 4.5);
   - define a base range polygon, including the brightest regions of your source(s) (Sect. 4.6).
3. run a big reduction:
   - possibly use the multi-core mode;
   - possibly use the batch mode;
   - perform many iterations (20, 50 or more);
   - use the zeroing polygon;
   - use the base range polygon;
   - use the source polygon;
   - by default, save the I,Q,U cubes at each iteration;
   - modify the timelines r.m.s. threshold, if needed (Sect. 3.2.13).
4. run a second data reduction to save individual cubes
   - zero iterations;
   - use, as input, the last source map produced in the previous big reduction (keep in mind Sect. 6.5);

- use the same polygons as in the previous (big) run;
- save individual I,Q,U cubes.

5. reduce the scans to be used for the instrumental polarization, i.e. point-like sources with no polarization (e.g. Uranus) or weakly polarized (e.g. specific QSOs) (Sect. 6.9):

   - one by one;
   - use the `compLeak`.

6. correct the instrumental polarization (Sect. 6.9):

   - use the dedicated script, modifying it to the specific data set under analysis;
   - correct your dataset by sub-sets of scans: each subset will use the same leakage pattern, therefore the scans need to be grouped such that each given leakage scan was taken with the same focus (and other) conditions as these science scans;
   - combine all individual, corrected scans together (e.g. using the last part of the `corr_instr_pol.piic` script and adapting it to the specific case).

7. analyse the polarization

   - use either the dedicated `plPol4.greg` script
   - or the user's favorite tools.

6.12 Important, to keep in mind

One question that might arise is which accuracy it is possible to achieve in the polarization degree and angle of the source, applying the leakage correction. The answer depends on the actual procedure followed to apply such a correction and on the assumption and simplifications applied. Here we would like to highlight a couple of important caveats to keep in mind when planning the correction of the instrumental polarization, and when evaluating which accuracy the given science project needs to reach in order to pursue its goals.

The leakage pattern can be characterized by observing a known non-polarized point-like source with NIKA2-pol (e.g. Uranus). Consequently, it is well known that the instrumental polarization pattern depends on the beam of the system telescope+NIKA2, and therefore – among others – on the shape of the telescope's main dish and on the focus of the system.

6.12.1 Beam shape dependencies

Thermal expansion induced by the solar irradiation, either direct or indirect, produces variations of the shape of the main dish. Also the de-icing system is a cause of strong distortion , when active[9].

The focus surface of NIKA2 is not flat, but it is curved. Being the detector arrays flat, this curvature translates into a focus difference of up to 0.7-0.8 mm (in terms of the sub-reflector position

[9]usually it takes at least two hours for the dish to relax after de-icing, and this is taken into account if de-icing is necessary, i.e. NIKA2 observations are avoided during the first hours after de-icing.

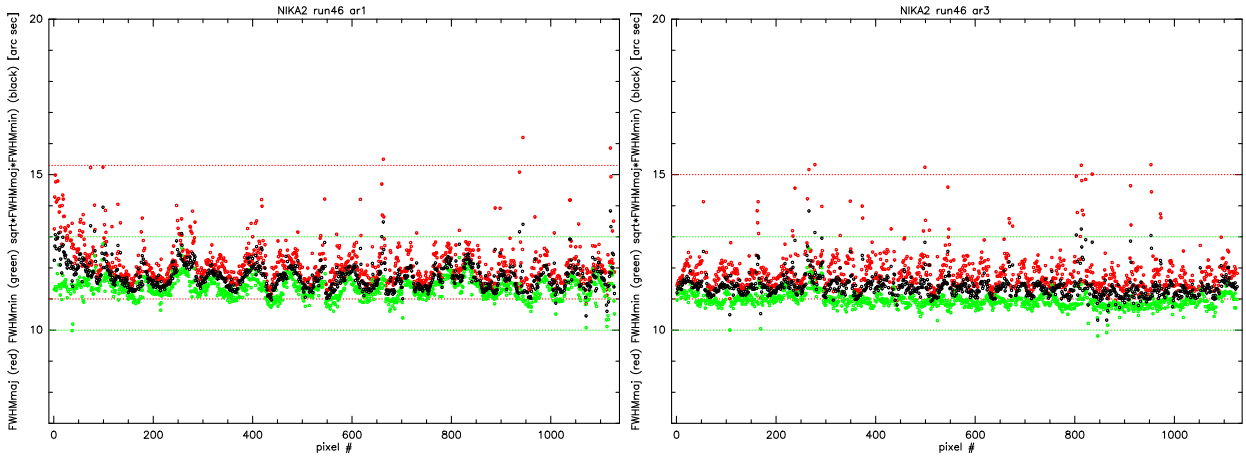


Figure 6.5: Variation of the 1 mm NIKA2 beams FWHM across the field of view and between Ar 1 (*left*) and Ar 3 (*right*), obtained for a beam map targeting Uranus, during the NIKA2 run 46. On the x-axis the KIDs ID number (from 1 to N) is used. The red, green and black symbols represent the major, minor axes FWHM of a Gaussian fit to the main beam, and their geometrical mean, respectively.

along the z axis) across the NIKA2 field of view. Moreover, the two 1 mm arrays (Ar 1 and 3) are not on focus with respect to each other. The consequences are that: *a)* the shape of the NIKA2 beam varies strongly across the field of view; *b)* because the procedure of focusing the telescope is usually based on Ar 2 (which lies between Ar1 and Ar 3 in terms of focus position), depending on the optimal focus at the time of the observations, it can happen that Ar 1 is better focused than Ar 3 or vice versa.

Figure 6.5 shows the variation of the FWHM of the NIKA2 1 mm beams for one beam map targeting Uranus. The left panel belongs to Ar 1 and the right panel to Ar 3. Each dot represents the beam of a different KID across the field of view. The KIDs are organized in rows: the wavy behaviour of FWHM is related to the variation of the beam along the rows of KIDs. The amplitude of the "wave" is at least 5% in FWHM ($\sim$ 10% peak to peak). Also the different focus of Ar1 *vs.* Ar 3 can be seen: the amplitude of the FWHM variations and its average value differ by at least 5%.

6.12.2 Consequences on the leakage pattern

These beam variations produce variations of the instrumental polarization pattern, depending on the main dish conditions in general (e.g. solar irradiation), focus, position in the field of view, etc. We can expect that a proper optimization of the focus prior to start a science polarimetry scan is critical, because the focus difference between Ar 1 and Ar3 can translate on the leakage patter in ways difficult to predict.

Therefore the accuracy reachable for the polarization degree and angle might strongly depend of the approach for the leakage correction. The NIKA2-pol user shall therefore evaluate which accuracy their science goal requires and find a compromise between required observing time (also to characterize the leakage), data reduction effort and foreseen results.

For example, adopting a single leakage pattern — obtained as the average over all KIDs, i.e. over the whole FoV, of Ar 1 and/or Ar 3, in typical observing conditions and Elevation — is a relative unexpensive approach in terms of exposure time and complexity, but the PIIC team guesses that it could reach an accuracy not better than 5% on polarization degree and angle. To do better, a more sophisticated approach is probably needed.

The ultimate solution would be a grid of leakage patterns, sampling the multi-dimensional space of (*focus, Elevation, position in the FoV, dish surface temperature, ...*) as best as possible, or a mathematical function mapping this parameter space into a leakage correction.

Chapter 7

The quick look monitor

Strawberries & lemon for the long NIKA2 shifts.

PIIC includes scripts for quick data reduction. They can be useful to the science user to check — for example — under which conditions the observations were done (e.g. pointing, tracking, etc.). These scripts adopt different names, depending on purpose. On-the-fly data reduction at the telescope, processing new data as they are produced by NIKA2, is called *monitor*. Quick off-line (i.e. not during the data flow, e.g. at home) data reduction is called *quick look* (QL). The name "quick look monitor" can also be found.

The setup of the QL is similar to the default setup of the science pipeline, i.e. it is optimized to process centered, compact sources.

7.1 On the fly data reduction at the telescope

Usually the on-the-fly data reduction at the telescope is activated by the astronomer on duty (AOD). In case not, prepare the needed directory tree as follows:

```
cd
mkdir ql
cd ql
mkdir dat plAr1 plAr2 plAr3 redAr1 redAr2 redAr3
ln -s directory_containing_data imbfitsDir
```

Then run the *monitor* in one terminal, typing:

```
piic @monitor1
```

for NIKA2 array 1 (first 1 mm array). Similarly, do the same in other terminals for Arrays 3 and 2, if desired, by simply using `monitor3`, `monitor2pf` (processing only pointing and focus scans) and `monitor2m` (processing all other maps).

7.2 Quick Look

The *quick look* analysis (QL) is the off-line version of the PIIC *monitor*. By "off-line" it is intended not on-the-fly as the data are produced and stored on disk at the telescope. It exists in two different incarnations:

- work on one scan at a time;
- work on all scans taken from a user-generated list.

In the first case the QL scripts for the three different arrays are called:

```
qlAr1
qlAr2
qlAr3
```

In the second case, they are called:

```
qlAr1list
qlAr2list
qlAr3list
```

To use the QL, setup on your local disk a directory tree organized as for the *monitor* (see Sect. 7.1).

7.3 Simple use of QL

To run the QL scripts on one single scan, follow this simple example:

```
@qlAr1 scanname
```

and similarly for arrays 2 and 3. For example:

```
@qlAr1 20150627s40
@qlAr2 20150627s40
@qlAr3 20150627s40
```

To make use of the QL for lists, first create a list of all scans to be examined and then run the script. The list shall be called `ar1.LIST`, `ar2.LIST` or `ar3.LIST`, and shall contain the IMBFITS[1] filenames (the IRAM NIKA2 FITS files provided to the users after pool observations, stored in the `imbfitsDir/` defined before) without path.

For example, to run QL on all array-1 scans present in the `imbfitsDir/` repository, follow this simple example:

[1] IMBFITS means *IRAM Multi-Beam* FITS *files*; it is the standard file format of IRAM/30m and NIKA2 data.

```
cd
cd ql
cd imbfitsDir
ls iram30m*-1-* > ~/ql/ar1.LIST
cd ..

piic

@qlAr1list
```

7.4 Advanced example: use the QL on a refined list

In some cases, the user might need to run the QL on restricted list of scans, for example those belonging *only* to the observations of a specific target. Such a list can be prepared using any of your favorite tools. One option is to use GILDAS or PIIC to select scans on the basis of different parameters (e.g. observation type) and/or on the basis of the observed object (see also Sect. 3.2.2). The list should contain only the IMBFITS filenames, without path, for a given array.

For example, using PIIC, the list of all array-2 scans belonging to observations of our favorite *hypothetical* source called "Superantennae" can be produced in the following way:

```
cd
cd ql
cd imbfitsDir
ls iram30m*-2-* > ~/ql/all_ar2.LIST    ! lists all Ar2 IMBFITS files
cd ..

piic

inlis all_ar2                       ! load the list
sel SUPERANTENNAE                   ! selects only scans of object SUPERANTENNAE
sel type m                          ! select scans with observation type = map
write inlist SUPERANTENNAE_a2.LIST ! write the list of selected scans on file
init inlist                         ! if needed, resets the starting list (in memory)
                                    ! the user will need to start from scratch,
                                    ! if they wish to produce a new list
```

In this way, one ascii list has been produced for array 2 containing the names of all Superantennae IMBFITS files to be processed by the QL. Do the same for arrays 1 and 3. The QL script accepts only lists with names `ar1.LIST`, `ar2.LIST` or `ar3.list`. Rename your list accordingly, before starting the QL, for example:

```
cp SUPERANTENNAE_a2.LIST ar2.LIST
```

Note that selecting scans is not strictly necessary: one can run the QL on all the scans of a given data set (see Sect. 7.3).

Run QL on all your favorite scans. Note that if the *monitor* is running at the same time, QL might interfere with it. To avoid interferences, the *monitor* and the QL should run in different working directories.

7.5 Operations and results

As it runs, the QL (or *monitor*) produces different diagrams, that are shown in sequence in two different graphic devices. The same diagrams are also saved as PNG files.

First of all, the QL shows the timeline of all KIDs (detector pixels) after different operations have been performed on the signal (see Fig. 7.1). The timeline of each KID is plotted as a line of different color in units of signal [mJy/beam] vs. LST [s]. The many lines overimpose to each other.

In the top-left panel, only forward beam calibration, flat fielding and the correction of first order drifts (over the whole sccan timeline) are applied. The low-frequency modulation is due to variations of sky signal (e.g. due to intrinsic changes of sky conditions or due to changes of airmass during the scan) and KIDs instabilities (e.g. light-blue line in the Figure). Possible short-time peaks represent the signal of bright sources at the time when the given KID crosses them on sky.

As the data reduction proceeds, the QL computes and subtracts the *sky noise* from the data timeline of each KID (see also Chapter 3) and produces a flattened timeline (*top-right panel*). Those KIDs that do not comply to a number of selection criteria are then excluded (*bottom-left panel*). Finally the baseline subtraction, the main beam calibration and the extinction correction are applied and only the corrected source signal is left (*bottom-right panel*).

After these operations, the QL shows four different diagrams, as in Fig. 7.2. Depending on which type of observations have been carried out, different diagrams might be displayed. As describing all of them here would be unfeasible, we invite the users to explore the possibilities of the QL.

The case shown in Fig. 7.2 is a pointing scan on source MWC349. The four panels show: the tracking accuracy of the telescope in Azimuth and Elevation; the reduced map of the scan, including information about the beam size, source flux, and pointing corrections; the noise r.m.s. in the field of view and the position of the KIDs in Nasmyth coordinates; the Gaussian fit in the Azimuth and Elevation axes of the source profile, aimed at determining pointing offsets.

The QL performs a simplified data reduction. Generally speaking, the results (even if written in FITS files) are to be used for display and checking purposes only.

Nevertheless, using the QL with the correct values of the atmospheric opacity τ for simple point-like, well-centered sources, the result does not differ too much from a proper thorough data reduction. The same does not hold if the source is not centered or is complex.

Note, finally, that the QL does not combine scans together; it reduces each scan separately.

7.6 Products

The QL script produces many PNG images, that are stored in the `plAr#/` directories.

It also produces many data files, that are stored in the `dat/` directory. It is important to keep in mind that the *.dat* files are appended. This means that each time the QL is run, the *.dat* files become bigger and bigger. One might want to check their size from time to time.

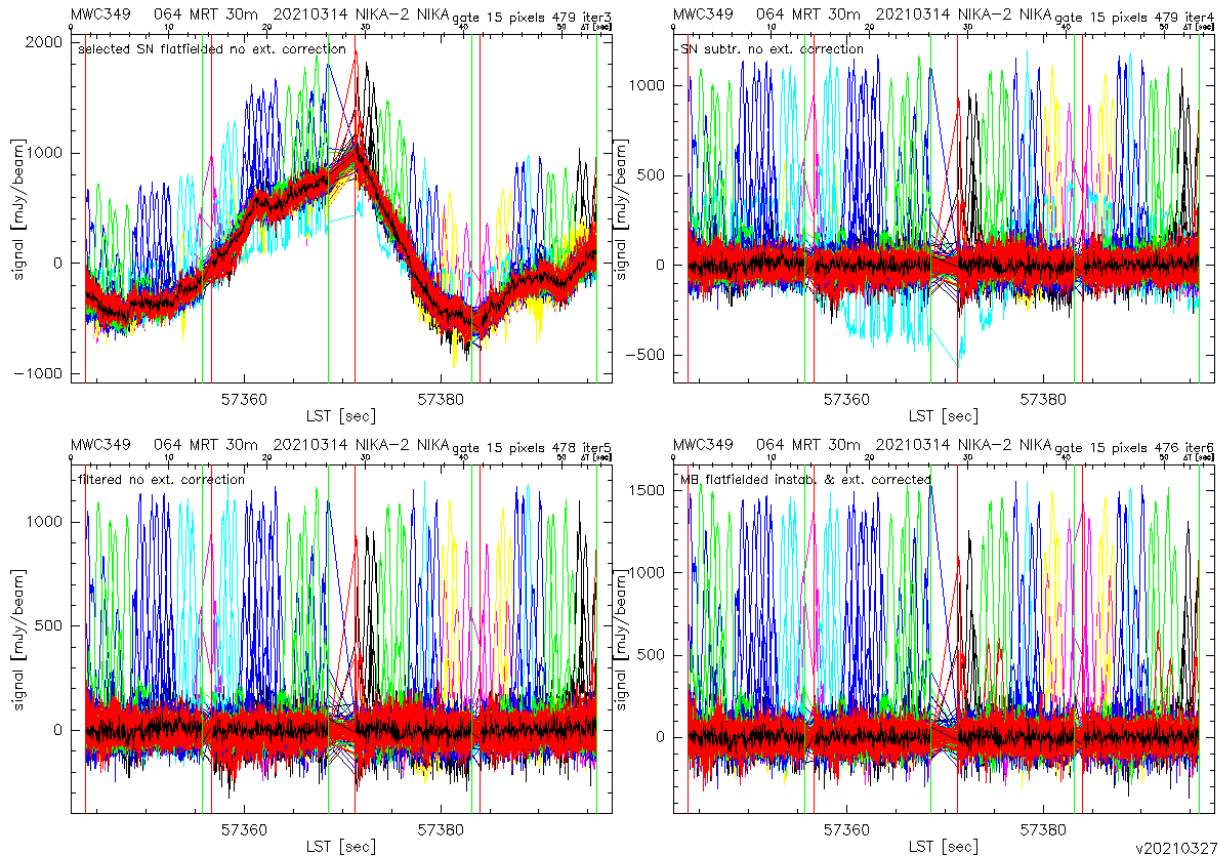


Figure 7.1: Example of timeline of a pointing scan on the source MWC349 (Array 2), after different operations have been performed on the signal: flat-field and forward beam calibration (*top-left*); sky-noise removal (*top-right*); exclusion of problematic KIDs (*bottom-left*); main beam, baselines and extinction corrections (*bottom-right*). Each line belongs to a different KID (detector pixel) of NIKA2. See main text for more details.

7.6.1 Save FITS files

The QL scripts, in their standard configuration, do *not* produce any FITS files of the *quickly reduced* data. If a FITS version of the result maps is needed, it is possible to override the standard setup of the QL scripts.

In order to do this, copy into the working directory, from the `pro/` folder in the installation directory, the scripts that define the QL parameters.

Generally speaking, PIIC finds the scripts in the `pro/` folder. Following GILDAS' conventions, if a script with the same name is in the working directory, then this has the priority. Therefore it is sufficient to copy the scripts of interest in the working directory and modify them there, in order to use a custom setup instead of the default setup, without the need to modify the installed version.

For the three arrays, these scripts are called:

```
qlDefsAr1.mopsic        for the individual-scan version
qlDefsAr2.mopsic
```

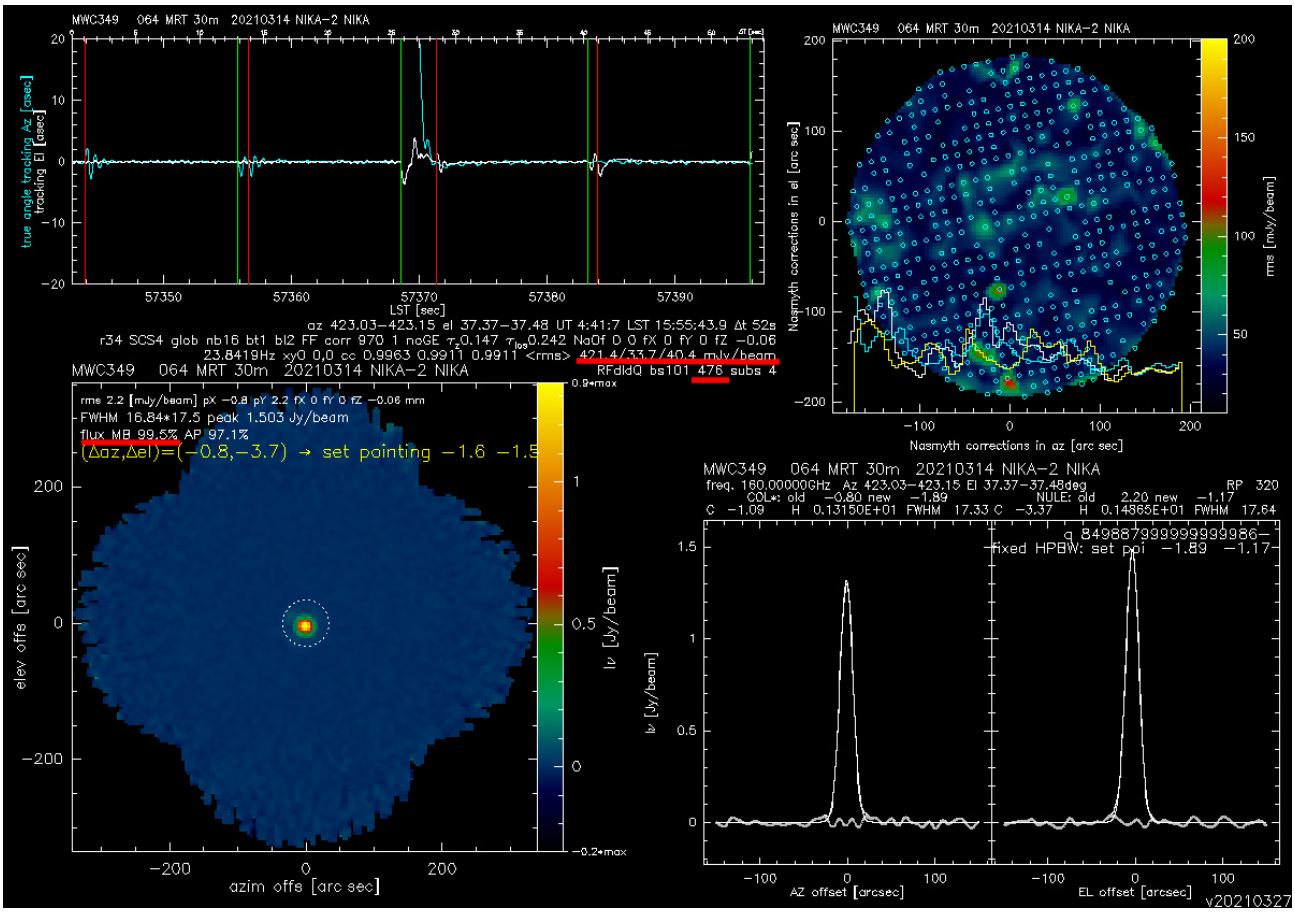


Figure 7.2: Result of QL analysis of a pointing scan on the source MWC349 (Array 2). Top-left: tracking accuracy in Azimuth (white) and Elevation (cyan), i.e. difference between commanded and effective coordinates as a function of time. *Bottom-left*: reduced map of the scan, including information about the beam size (HPBW), the peak flux, the main beam flux, the aperture flux (extracted within the aperture marked on the map), and pointing corrections (in yellow). *Top-right*: noise r.m.s. in the field of view and position of the KIDs in Nasmyth coordinates. Three histograms are also shown, representing a cut across the FoV at lines 0 (white), and ±100 (yellow and cyan). *Bottom-right*: Gaussian fit in the Azimuth and Elevation axes of the source profile, aimed at determining pointing offsets. Among the many pieces of information reported in this diagram, the timeline r.m.s., the number of KIDs left and the fraction of flux (of a known calibrator) recovered in the main beam are underlined in red. Three values of the noise r.m.s. are given; they correspond to the top-left, the bottom-left and the top-right panels of Fig. 7.1, respectively.

```
qlDefsAr3.mopsic

qlDefsAr1list.mopsic    for the list version
qlDefsAr2list.mopsic
qlDefsAr3list.mopsic
```

Edit the variable `writeMap` and change it from "*no*" to "*yes*":

```
let writeMap no          -> yes
```

Now the QL will write FITS files of the reduced data (maps) and will store them in the directory `redAr*/`.

7.7 Additional setup parameters

It is possible to optimize the value of some parameters in QL setup, to obtain a higher-quality QL products (both for simple sources and also more complex targets), or speed up the processing, etc.

The order of the baseline fit can be changed (e.g. from 1 to 3). This change needs to be taken with care, because — for example — if not enough data pixels (or not good enough data) are available, it might produce more damage than improvement. Nevertheless — assuming that you know what you are doing — the parameter to be changed is called `blOrderOrig`.

For increasing speed, just know that the parameter `nBest` defines how many best-correlating KIDs are considered in the computation of base-line corrections. A value of 32 does a good job; a value of 0 (zero) does a bad job, but is 10 time faster (because searching the `nBest` best correlating KIDs is not requested anymore).

In order to prevent the QL to stop after having processed each scan, set the parameter `usePause` to *no*.

References

Adam, R., Adane, A., Ade, P. A. R., et al. 2018, A&A, 609, A115

Agudo, I., Thum, C., Molina, S. N., et al. 2018a, MNRAS, 474, 1427

Agudo, I., Thum, C., Ramakrishnan, V., et al. 2018b, MNRAS, 473, 1850

Ajeddig, H. 2022, PhD thesis, École doctorale n. 127 : astronomie et astrophysique d'Île-de-France (AAIF), https://theses.hal.science/tel-04078332

Ajeddig, H., Adam, R., Ade, P., et al. 2022, in European Physical Journal Web of Conferences, Vol. 257, mm Universe @ NIKA2 - Observing the mm Universe with the NIKA2 Camera, 00002

Andrianasolo, A. 2019, PhD thesis, Institut de Planétologie et d'Astrophysique de Grenoble (IPAG), https://theses.hal.science/tel-02936622

Berta, S. & Zylka, R. 2019-2025, Welcome to the PIIC Pipelines (Pointing and Imaging In Continuum, as code by Robert Zylka), https://www.iram.fr/~gildas/dist/piic.pdf

Calabretta, M. R. & Greisen, E. W. 2002, A&A, 395, 1077

Calvo, M., Roesch, M., Désert, F. X., et al. 2013, A&A, 551, L12

Catalano, A., Adam, R., Ade, P., et al. 2016, arXiv e-prints, arXiv:1605.08628

Emerson, D. T., Klein, U., & Haslam, C. G. T. 1979, A&A, 76, 92

Haslam, C. G. T. 1973, Kleinheubacher Berichte, 16, 451

Kam, M., Nagai, H., Kino, M., et al. 2025, arXiv e-prints, arXiv:2510.05540

Myserlis, I., Agudo, I., Thum, C., et al. 2025, in Highlights of Spanish Astrophysics XII, ed. M. Manteiga, F. González-Galindo, A. Labiano-Ortega, M. J. Martínez-González, N. Rea, M. Romero-Gómez, A. Ulla-Miguel, G. Yepes, C. Rodríguez-López, A. Gómez-García, & C. Dafonte, 121

NIKA2-Pol commissioning team. 2021, NIKA2-Pol: polarization commissioning report

Perotto, L., Ponthieu, N., Macías-Pérez, J. F., et al. 2020, A&A, 637, A71

Pisano, G., Maffei, B., Ade, P. A. R., et al. 2016, arXiv e-prints, arXiv:1610.00582

Ponthieu, N. 2003, PhD thesis, Laboratoire de Physique Subatomique et de Cosmologie, https://theses.hal.science/tel-00003465v1

Ritacco, A. 2016, PhD thesis, Laboratoire de Physique Subatomique et de Cosmologie, https://theses.hal.science/tel-01467862v2

Ritacco, A., Macías-Pérez, J. F., Ponthieu, N., et al. 2018, A&A, 616, A35

Ritacco, A., Ponthieu, N., Catalano, A., et al. 2017, A&A, 599, A34

Savini, G., Pisano, G., & Ade, P. A. R. 2006, Appl. Opt., 45, 8907

Thum, C., Agudo, I., Molina, S. N., et al. 2018, MNRAS, 473, 2506

Zylka, R. 1998, MOPSI Cookbook, ITA Heidelberg, http://www.iram.es/IRAMES/otherDocuments/manuals/Datared/pockcoo.ps

Zylka, R. 2013, MOPSIC: Extended Version of MOPSI, Astrophysics Source Code Library, record ascl:1303.011, https://ui.adsabs.harvard.edu/abs/2013ascl.soft03011Z

Appendix A

Content of PIIC NIKA2 DAFs

In order to process NIKA2 data, PIIC needs a set of calibration files, that describe the properties of the KIDs arrays and of the atmosphere: geometry, resonance frequencies, cross-talking, response to incoming light (i.e. flux calibration), atmospheric opacity and relation of sky load to line-of-sight extinction correction. These are called *data associated files* (DAFs) and can be retrieved from the GILDAS download pages.

At each new release of PIIC, the DAFs database is included in the PIIC package. After each observing pool or semester, the PIIC support team processes calibration data and produces new calibration files. The DAFs database is thus updated and a new version is available online, in a separate tar-ball, independent of the PIIC package. Therefore, when browsing the PIIC GILDAS pages for updates, always check for new PIIC *and* DAFs tar-balls.

The PIIC DAFs consist of six different main types of files. In alphabetical order:

- calibration files (CAL), defining the response of KIDs for flux calibration;
- deleted receiver pixels (DRP) files, listing those KIDs that are known to be defective;
- frequency files (NKFR), listing the natural resonance frequencies of all KIDs, for different sweeps[1];
- files defining the relation between the line-of-sight extinction correction and sky load (NKSLEX) for the given NIKA2 observing week(s).
- receiver pixels parameters (RPP), listing the flat field, flag and beam properties of each KID in the field of view of NIKA2, for different sweeps;
- atmospheric conditions over all observing runs (TAU files), containing the values of τ produced by the Observatory's tau-meter.

The DAFs are organized in the `dafs/` repository in the PIIC installation directory. This `dafs/` directory contains five sub-directories, one per each DAFs file type. The DAFs come with a PDF document[2] that summarizes the calibration properties and quality for all NIKA2 science runs and

[1] a *sweep* is practically a re-definition of KIDs resonance frequencies, performed regularly for each new season.
[2] file summary_PIIC_DAFs_and_figs_v*.pdf

includes more details and several explicatory diagrams. In this Appendix the content of each DAFs file is described.

A.1 CAL files

For each NIKA2 array, this directory contains one file, called `MRT_NIKA-*.CAL` (where `*` is the array number). Each file lists a number of entries, structured as three columns and one line, for example:

```
2019-05-16T20:00:00 162.84  1.0
```

The three pieces of information in each line are:

```
Column 1:       the expiration date of the given flux calibration factor
Column 2:       the flux conversion factor from instrumental units to Jy
Column 3:       a further correction factor.
```

Additional notes and comments are provided, e.g. specifying the start of a given NIKA2 run or sweep changes. Exclamation marks indicate comment lines.

The CAL value translates the instrumental units into physical units. The reference absolute fluxes of the primary calibrators (i.e. Planets) are taken from the models implemented in GILDAS/ASTRO. In the *current implementation (as of 2025/05/30)*, the reference 100 GHz brightness temperatures of Venus, Uranus and Neptune are 350, 132 and 125 K, respectively, with a 5% accuracy. A spectral index of -0.35 is used to derive their intrinsic fluxes at the NIKA2 frequencies (260 and 150 GHz).

A.2 DRP files

For each NIKA2 array, this directory contains one file, called `MRT_NIKA-*.DRP` (where `*` is the array number). These files list a number of entries, each one consisting of several lines. For example:

```
DATE 2019-09-18T23:13:04 1 12 20 21 32 37 38 71 72 73 74 77 78 91 92 93 98 ...
                         214 215 224 225 227 228 262 263 267 275 460 461 463 ...
                         344 350 351 352 361 362 377 393 403 426 441 442 453 ...
                         524 525 534 541 542 545 546 554 555 573 574 579 588 ...
```

These entries represent the identification numbers of the KIDs that are known to be affected by strong cross-talk or other problems and are therefore excluded a priori from the data analysis. The date and time of each entry indicate the expiration of the given entry. Usually, cross-talks are checked at least at each new sweep (because the logical order of KIDs changes) and whenever an intervention on the instrument is performed. Some times exceptional checks are also performed independently of these events.

A.3 NKFR files

For each NIKA2 array, this directory contains one file, called `MRT_NIKA-*.NKFR` (where `*` is the array number). Several other files are also included. The `MRT_NIKA-*.NKFR` files list a number of entries, for example:

```
2019-02-12T12:00:00 N2R12_NIKA-1-20171019s3.nkFr
```

For a given expiration date, these entries point to a file that contains the intrinsic resonance frequencies of all available KIDs, as defined by a sweep. Specific comments indicate when a new sweep was performed.

The `N2*.nkFr` files consist of two columns: identification number of the KID (from 1 to n) and their resonance frequency.

A.4 NKSLEX files

Introduced in the 7th release of PIIC, the NKSLEX files describe the relation between sky load and extinguished flux of the sources. This function is strongly dependent on the NIKA2 sweep, on seasonal atmospheric variations, and even on variabilities of weekly time scales. Therefore NKSLEX are re-defined at each new NIKA2 run.

For each NIKA2 array, the NKSLEX directory contains one file, called `MRT_NIKA-*.NKSLEX` (where `*` is the array number). Several other files are also included. The `MRT_NIKA-*.NKSLEX` files list a number of entries, for example:

```
2019-01-29T12:00:00 N2R27ar2_ynyna_0_1_57_0_1_6_NO.mfit2vsSL
```

For a given expiration date, these entries point to a file that gives the parameters describing the function valid until that date and since the previous one. These `.mfit2vsSL` files consist of 9 columns. The last four columns define the a and b parameters in the function $\log(F_{\rm ext}) = a \times \mathrm{SkyLoad} + b$, as well as their uncertainty.

A.5 RPP files

For each NIKA2 array, this directory contains one file, called `MRT_NIKA-*_MB.RPP` (where `*` is the array number, and MB means "main beam"). Several other files are also included. The `MRT_NIKA-*_MB.RPP` files list a number of entries, for example:

```
2019-01-29T12:00:00 NIKA2ar1allRPs_run21med_NO_YES_1_92.rpp
```

For a given expiration date, these entries point to a file that gives the parameters describing the NIKA2 Array. Additional comments specify when new sweeps have been performed, or when other changes happened. These `.rpp` files consist of 25 columns; for data processing, the essential columns are only the following:

```
Column  1       identification number of the given KID for the current sweep
Column  3       main beam flat field
Column  4       forward beam flat field
Column  5       delta(x,Nasm), Nasmyth x-offset at zero elevation
Column  6       delta(y,Nasm), Nasmyth y-offset at zero elevation
Column  7       quality flag; if =0 the given KID might be usable (but see DRP)
Column 14       FWHMmaj [arcsec], major axis  FWHM of beam, measured with beam-maps
Column 15       FWHMmin [arcsec], minor axis  FWHM of beam, measured with beam-maps
Column 16       beam PA [deg], measured with beam-maps
```

The ending line of each file defines the verse of the Nasmyth rotation and the pointing KID if the array is a pointing array; for NIKA2 it is Array 2.

Other parameters given in the `*.rpp` files might be used in PIIC internally for statistical purposes and are not essential.

For a given i-th NIKA2 data record, the offset position of each j-th KID, with respect to the reference (pointing) pixel, can be computed as:

$$\delta x_{ij} = \Delta x_i^{\rm tel} - \delta x^{\rm corr}, \tag{A.1}$$

$$\delta y_{ij} = \Delta y_i^{\rm tel} - \delta y^{\rm corr}, \tag{A.2}$$

where $\Delta x_i^{\rm tel}$ and $\Delta y_i^{\rm tel}$ are the telescope offsets defined by the observing mode for each record, and the $\delta x^{\rm corr}$ and $\delta y^{\rm corr}$ corrections are given by:

$$\delta x_{ij}^{\rm corr} = \delta x_j^{\rm Nasm} \cos(el_i) + \delta y_j^{\rm Nasm} \cos(el_i), \tag{A.3}$$

$$\delta y_{ij}^{\rm corr} = -\delta x_j^{\rm Nasm} \cos(el_i) + \delta y_j^{\rm Nasm} \cos(el_i). \tag{A.4}$$

In these Equations, $\delta x_j^{\rm Nasm}$ and $\delta y_j^{\rm Nasm}$ are read from the `.rpp` files, and el is the telescope elevation of the i-th data record.

A.6 TAU files

For each NIKA2 array, this directory contains several files, called `MRT_NIKA-*run**.TAUS` (where `*` is the array number, and `**` is the NIKA2 run number to which the file belongs).

The structure of each file is of two columns: the modified Julian date (MJD) at which the value of the atmospheric opacity, τ, was measured; and the value of τ at the frequency of the given NIKA2 array (i.e. 1 or 2 mm).

The atmospheric opacity was measured by the IRAM 30m tau-meter instrument at 225 GHz and converted to the NIKA2 frequencies using standard procedures.

Appendix B

Effects of re-gridding

As mentioned in Section 3.3.2, gridding is performed applying the algorithm by Haslam (1973) modified and improved by R. Zylka. The kernel used to re-distribute the flux of each KID onto the final grid is a sinc function in Fourier space. It is important to truncate it properly, in order to avoid artefacts (because it extends to infinity).

Using the `addSource` option (see Sect. 4.15), simulations have been produced and analyzed to study the effects of gridding on the width of point-like profiles and on the flux of point-like sources. For extended sources, the issue of broadening due to gridding is not important.

The artificial input map consists of 50 sources with peak signals between 1 and 90 mJy/beam. The sources are assumed to be point-like and their profiles are Gaussian with FWHM = 12 arcsec. Since these are point-like sources, peak fluxes correspond to total fluxes by definition. The sources were added onto a timeline of pure Gaussian noise with r.m.s. of 0.001 mJy. The noise r.m.s. was set so low in order to have a very high S/N ratio and be able to study the gridding effects on the point-like profiles without having to deal with the effects of poor S/N. The top-left panel of Fig B.1 shows the artificial map.

After processing with the default pipeline settings, the shape and fluxes of the artificial sources were measured by fitting elliptical Gaussian profiles, allowing re-centering. The profiles are described by their minor and major axes FWHM, their position angle (PA) and their peak flux (called "height" in the diagrams).

Figure B.1 presents the results. The profile geometrical parameters measured with the elliptical Gaussian fit (FWHM of minor and major axes and PA) are shown as a function of each other and as a function of the measured peak flux. The median measured $\mathrm{FWHM}_{\mathrm{min,maj}}$ are 12.059×12.055 arcsec, to be compared to the 12 arcsec inout FWHM. This demonstrates that the broadening of the point-like profiles due to the adopted gridding algorithm is $< 0.5\%$. The retrieved peak fluxes agree with the input ones within a $\sim 0.2\%$ difference.

We reiterate that the simulation is based on very high S/N artificial sources. In the real world, the limitations of the data play also an important role. By these are meant the limited exposure time (i.e. a higher intrinsic noise), the instrumental imperfections (including telescope deformations above all) and the sky and instrumental (KIDs and electronics) instabilities, and so on. Therefore the broadening inferred from real data may differ from the one measured under the ideal conditions shown here, but is not dominated by the effects of the gridding algorithm.

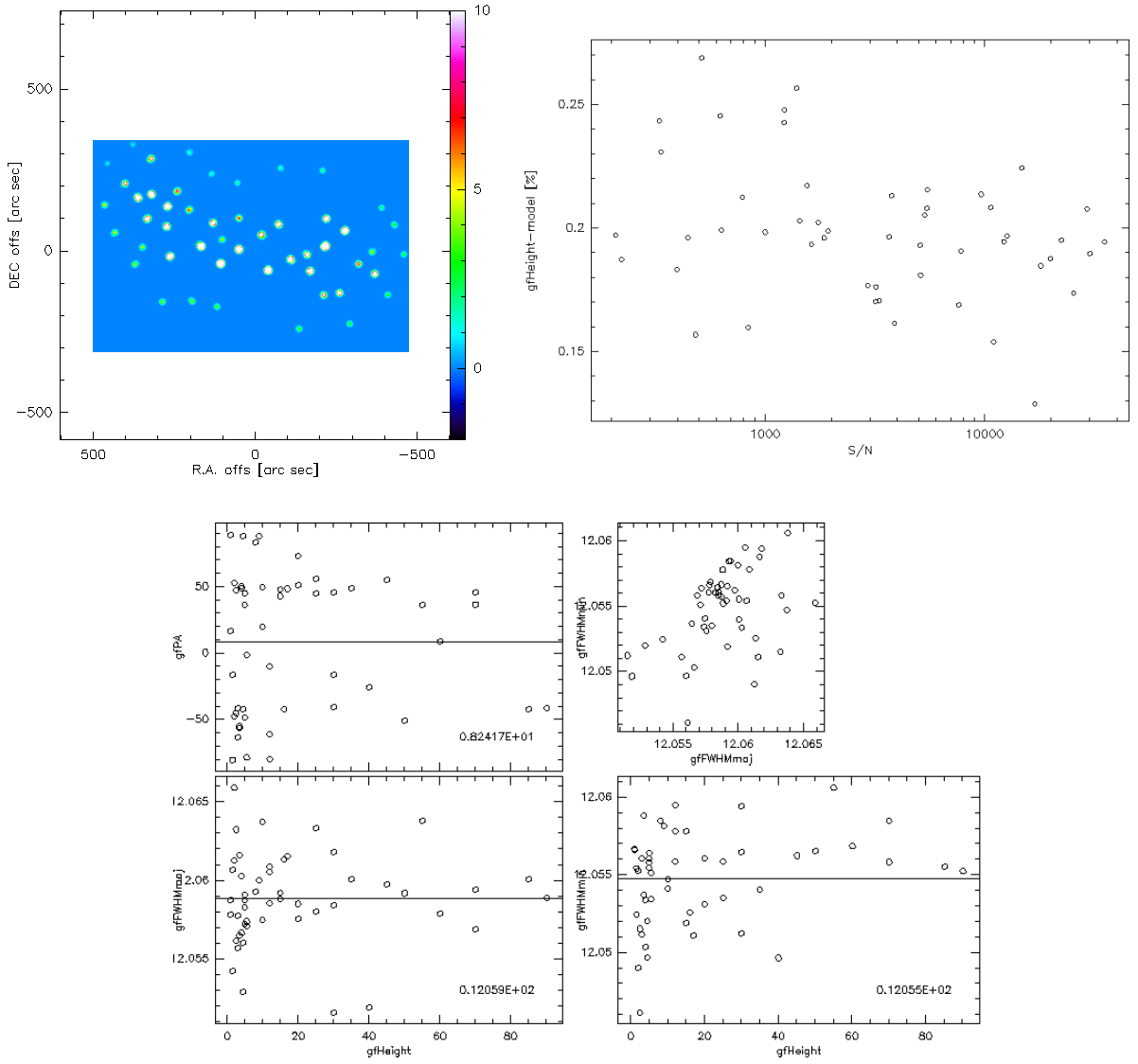


Figure B.1: Simulation to study the effects of gridding. *Top-left panel*: artificial map used in the simulation. *Top-right panel*: different between output (measured) and input peak fluxes, as a function of S/N ratio. *Bottom panels*: measured source geometry (elliptical Gaussian fit, `gf`, FWHM of minor and major axes and position angle as a function of each other and of peak flux, `Height`.

Appendix C

Transfer function examples and challenges

To the aim of performing an accurate and correct analysis of science data, it is necessary to handle the effects that are not related to the astronomical source itself, i.e. to study what happens to the light emitted by the target from the moment it leaves the object to the moment the final map is ready on one's hard disk.

The path starting from just outside the Earth's atmosphere and ending on the KIDs detectors has been commented before and is subject of the NIKA2 documentation. The possible effects induced by data processing can be studied by adding an artificial source to the KIDs timelines with the same characteristics as the original data (i.e. ideally *into the original timelines*), and by processing it in the same exact conditions as the original data (e.g. with the same PIIC setup).

This process is conveniently called "study of the *transfer function*" (TF), where it is meant the transfer from the raw data to the final map as operated by the data reduction process. During this process, possible idiosyncrasies of the data themselves might be highlighted, therefore the TF can also come in help to optimize the observing strategy for a given target.

Section 4.15 describes how to add artificial sources to the timelines. Appendix B discusses the effects of re-gridding on the profiles of point-like sources. Here we deal with the fundamental question: how does data processing affect the source flux distribution and total flux?

Some experiments with the TF of point-like and extended sources are presented in this Appendix, with the aim to show the principles and some caveats of the TF analysis. We do not have the pretension to be complete, nor exhaustive of all possible cases, hence each science project needs to perform its own specific TF analysis. Those presented here are only general examples, but can be still used as inspiration to define one's methodology to study the TF.

C.1 List of term and acronyms used in this Appendix

This *transfer function* Appendix makes use of a quite variegate and complex nomenclature of cases and of PIIC features. Here we list them, so that the reader can have an easy and direct reference to look at any time:

AS	AddSource or Artificial Source; i.e. the input artificial map/object used in the simulations.
FM	Final Map; i.e. the map produced by PIIC processing the real data, without the null map option.
FMAS	Final Map plus AS; the AS has been added to the KIDs timelines, without using the NM option.
L	in the figures it refers to `blOrderOrig`.
NM	Null Map; i.e. the map obtained multiplying each second scan by -1.
NMAS	Null Map plus AS; i.e. the result of PIIC processing, obtained adding the AS to timelines and using the NM method.
RZ	zeroing radius (`rZmEq` parameter; see Sect. 4.8.3).
SMAS	Simulated Map using AS; the AS is added to an empty, simulated timeline that includes only artificial Gaussian noise (linked to the NIKA2 sensitivity and sky noise).
S/N	Signal to Noise ratio.
TF	Transfer Function.

C.2 Point-like sources transfer function

The analysis of the *transfer function* for the case of point-like sources is based on the same artificial map presented in Appendix B. It includes 50 sources with peak signals between 1 and 90 mJy/beam, and it's added to the timelines of a large deep dataset reaching down to ~ 0.15 mJy/beam r.m.s. in the central regions. Different pieces of analysis and data processing approaches have been carried out and are presented here:

1. NMAS; the artificial map was combined with the null map technique to study the TF of point-like sources.

2. FMAS; to highlight the different noise properties between NM and FM, the AS was added to the timelines without using the NM technique, taking care to avoid any overlap to real sources.

3. SMAS; a completely simulated dataset was defined and studied: empty timelines containing only Gaussian noise, to which the AS is added.

In the FMAS and NMAS cases, the data processing was carried out with five different values of `blOrderOrig` = 1, 2, 3, 5, 7 and in iterative mode with up to 12 iterations.

In the SMAS case, the Gaussian noise in the timelines is made of two components: the NIKA2 sensitivity and the sky noise. This noise is symmetric around zero by construction and it is not affected by the "zig zag" due to crossing a range of airmasses. This experiment represents, so to say, the ideal case of observations without any "noise instability" induced by the observing strategy or the instrument. Six different realizations of the SMAS simulation have been performed: the positions and fluxes of the injected sources do not change; the r.m.s. of the Gaussian noise does not change either; only the seed of the random number generator used to produce the Gaussian noise does change and therefore the local noise in each position of the map is different between the 6 SMAS cases. The

SMAS has been processed only adopting `blOrderOrig` = 1 because it is not affected by baseline "instabilities".

For these different data sets, the study of TF consists in measuring the percentage of recovered flux for all sources, and study its dependence on S/N ratio and on iteration number. In this analysis we study also how the noise properties of the NM and FM differ to each other.

Only 1 mm results are shown. A similar reasoning can be applied also to 2 mm data.

3.2.1 Recovered point-like sources flux

Figure C.1 presents how the fraction of recovered flux varies for each source as a function of iteration number and S/N ratio. The cases of FMAS and NMAS obtained with `blOrderOrig` = 3 are shown. The experiment carried out only with pure Gaussian noise (SMAS) is presented in Section 3.2.4. The sources are grouped in bins of S/N ratio. The S/N ratio of a given source changes as the baseline fitting order and the iteration number vary. Therefore, to simplify the comparison between NMAS and FMAS, and to allow to build the full *recovered flux fraction vs. iteration number* curves, it is necessary to fix a reference S/N ratio for each source, to be used in all diagrams. Our choice is to use the S/N ratio measured in the FMAS experiment at the last (12th) iteration, obtained with `blOrderOrig` = 3 for each source.

The Figure demonstrates that for isolated point-like sources in a deep, empty field, 5 iterations are already sufficient to reach convergence on the source's flux. Below the 5σ detection limit, the flux loss can be as high as 50%. Even up to S/N=20, if not 30, there might be still a 10% scatter on the measured fluxes, even in the case of these very deep data.

The null map experiment (NMAS) shows a better recovered flux fraction than the results obtained without nulling the original signals (FMAS): smaller scatter and recovered flux fraction closer to unity even in the low S/N tail. This is clearly due to the fact that the null map approach cancels out some of the problems affecting the FMAS.

This effect suggests that a TF based on the NMAS experiment might be an optimistic description of the properties of the data set.

We stress again, however, that this is only an example aimed to present the methods and their limitations. Here we are operating in the small numbers limit: only few sources have been added to the timelines. Therefore this analysis does not claim to be complete. A publication-grade TF analysis would imply to add thousands of artificial point-like sources, possibly distributed in flux as the observed (or modeled) number counts, and obviously implying several batches of small numbers in order to avoid geometrical confusion.

3.2.2 Noise properties of Null Map *vs.* Final Map

Proceeding in the comparison of the different experiments performed for point-like sources, Figure C.2 shows the fraction of flux recovered at the last iteration, as a function of S/N ratio, again for `blOrderOrig` =3.

The distribution of the sources in the NMAS and FMAS is very different, especially at the low S/N end. As mentioned before and as will be shown again in Sect. C.3, the noise properties of NM are different with respect to those of the FM. The noise of the FM is not symmetric around zero

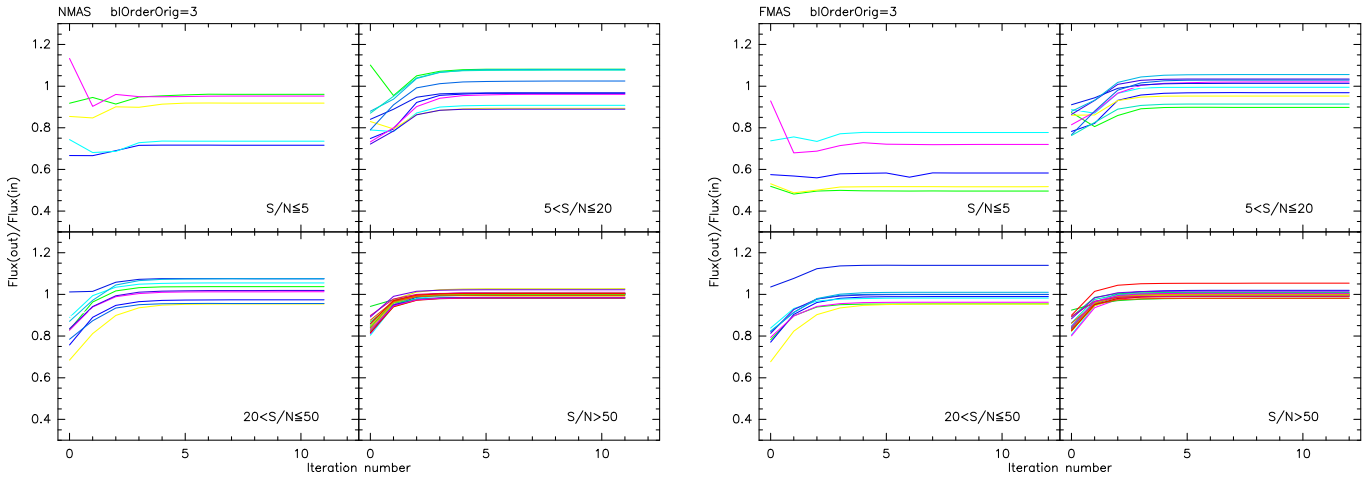


Figure C.1: Fraction of recovered flux in the point-like sources transfer function analysis. For the two experiments under analysis (NMAS and FMAS), this fraction is shown for each source as a function of iteration number, in four bins of S/N ratio. For clarity sake, only the results obtained with `blOrderOrig` = 3 are shown. Each colored line belongs to an individual artificial point-like source. For ease of identification, the colors of the lines are the same as the colors of the stars in Fig. C.2.

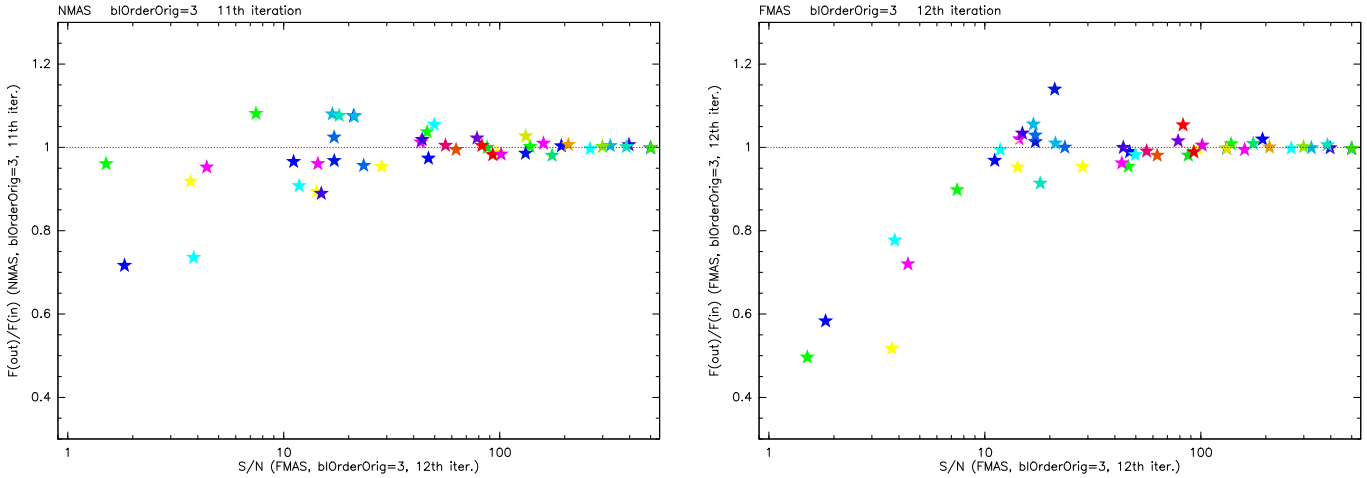


Figure C.2: Fraction of flux recovered at the last iteration, for the two experiments under analysis (NMAS and FMAS), in the case of `blOrderOrig` = 3, as a function of S/N ratio. Each colored star belongs to an individual artificial point-like source. The colors correspond to those of the lines in Fig. C.1.

and when multiplying every second scan by -1 to produce the NM, these asymmetries have the consequence that the NM noise is not equivalent to the FM noise.

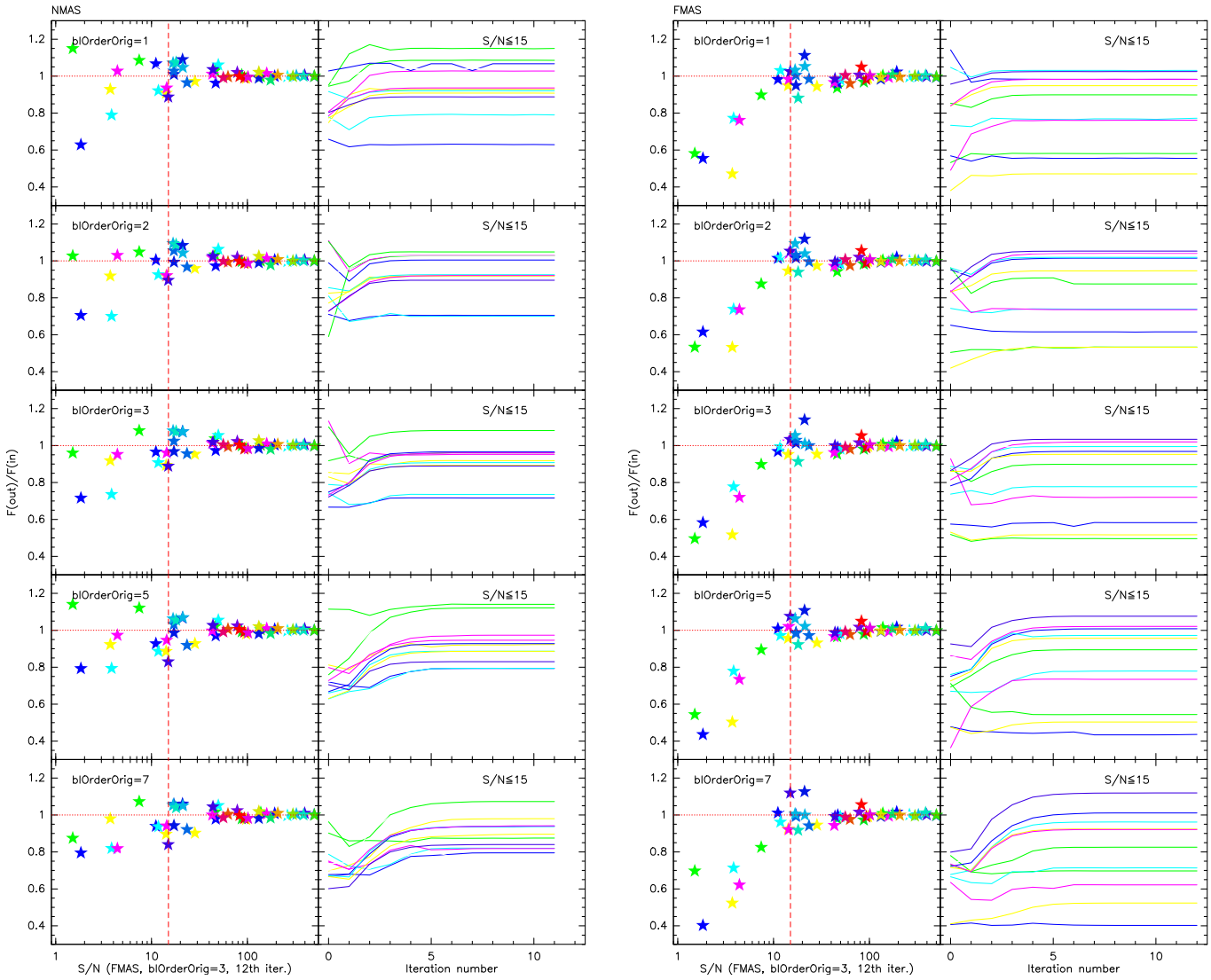


Figure C.3: Effect of changing the baseline fitting order on the point-like sources TF experiments: NMAS (*left*), and FMAS (*right*). For each of these cases, we show: *a)* the fraction of recovered flux at the last iteration, as a function of S/N ratio; *b)* the fraction of recovered flux as a function of iteration number, for sources with S/N $\leq$ 15. The colors of the stars and of the lines have a 1:1 correspondence. The vertical dashed line marks the S/N=15 level.

3.2.3 The effect of changing baseline fitting order

Each experiment has been performed with five different baseline fitting orders. Figure C.3 presents diagrams similar to Figs. C.1 and C.2 for the NMAS and FMAS, obtained varying the value of `blOrderOrig`.

As the baseline fitting order changes, the sources move in the diagram. At the low S/N side, this reshuffling is more evident. There is not a systematic rule regulating these movements, which testifies that the residuals of baseline fitting are here playing a role. The sources might lie on a crest or on

a depression produced by these residuals (see also Sect. C.3) and the recovered total flux therefore varies. This variation is larger, in relative terms, for fainter objects and/or sources positioned in noisier regions of the map.

The scatter at the low S/N end of the NMAS plot tends to improve for higher values of `blOrderOrig`. This is not necessarily the case for the FMAS, in which the residuals of the baseline subtraction can be amplified by the non-symmetric properties of the noise.

3.2.4 The ideal case of simulated pure Gaussian noise

The SMAS simulation has been conducted replacing the timelines with artificial pure Gaussian noise, aimed at representing the NIKA2 instrumental noise and the sky noise. To these timelines, the AS has been added during the processing in the usual way. As mentioned before, six different incarnations of the SMAS have been produced, each of them differing from the other only by the value of the random number generator seed. Only a `blOrderOrig` value of 1 has been used because the data do not have any baseline problems in this case.

Figure C.4 presents the fraction of recovered flux as a function of iteration number, for all the injected sources. As before, the Figure is split into four S/N bins and the S/N is again computed using the FMAS 12th iteration. The same diagram is repeated for all six different SMAS simulations. The panels in Fig. C.4 can be easily compared to those of Figs. C.1 and C.3.

The six different SMAS panels show the limitation of small numbers statistics especially at the low S/N end of the flux distribution of the artificial sources. The same sources, subject to Gaussian noise obtained with the same r.m.s. but with a different random seed, are recovered with very discrepant fluxes if their S/N is lower than 5. This is natura consequence of the fact that we are sampling only very sparsely the distribution of fluxes and noise, using only few sources.

Brighter sources are of course "better behaved", but even at S/N between 5 and 20 there is still at least a 10% scatter in the recovered total fluxes. This effect is even more evident in Fig. C.5, which depicts the flux recovered at the last SMAS iteration, for all 6 realizations in one single diagram. The classic opening of the flux distribution at low S/N is now clearly seen.

It is worth to note that, even taken together, these six SMAS simulations still are in the small numbers limit and a much larger number of objects should be used to study the flux vs S/N distribution in detail. Moreover, in order to make our point, here we have **not** varied the positions and fluxes of the input sources in each SMAS realization, at odds with what science projects will do.

C.3 Extended sources transfer function

Also in the case of extended sources, in the analysis of the *transfer function* care must be taken in examining possible spurious effects caused by the limitations of ground-based millimetric observations and by the limitations of the NIKA2 instrument installed on the IRAM 30m telescope.

The example analysis shown here has been carried out with an artificial galaxy including compact giant molecular clouds, added to the timelines of typical NIKA2 extended objects real scans. The null map switch was on, such that the objects originally observed cancel out and only the signal of the artificial object are kept (see Sect. 4.15.1). Tests similar to those presented here can also be

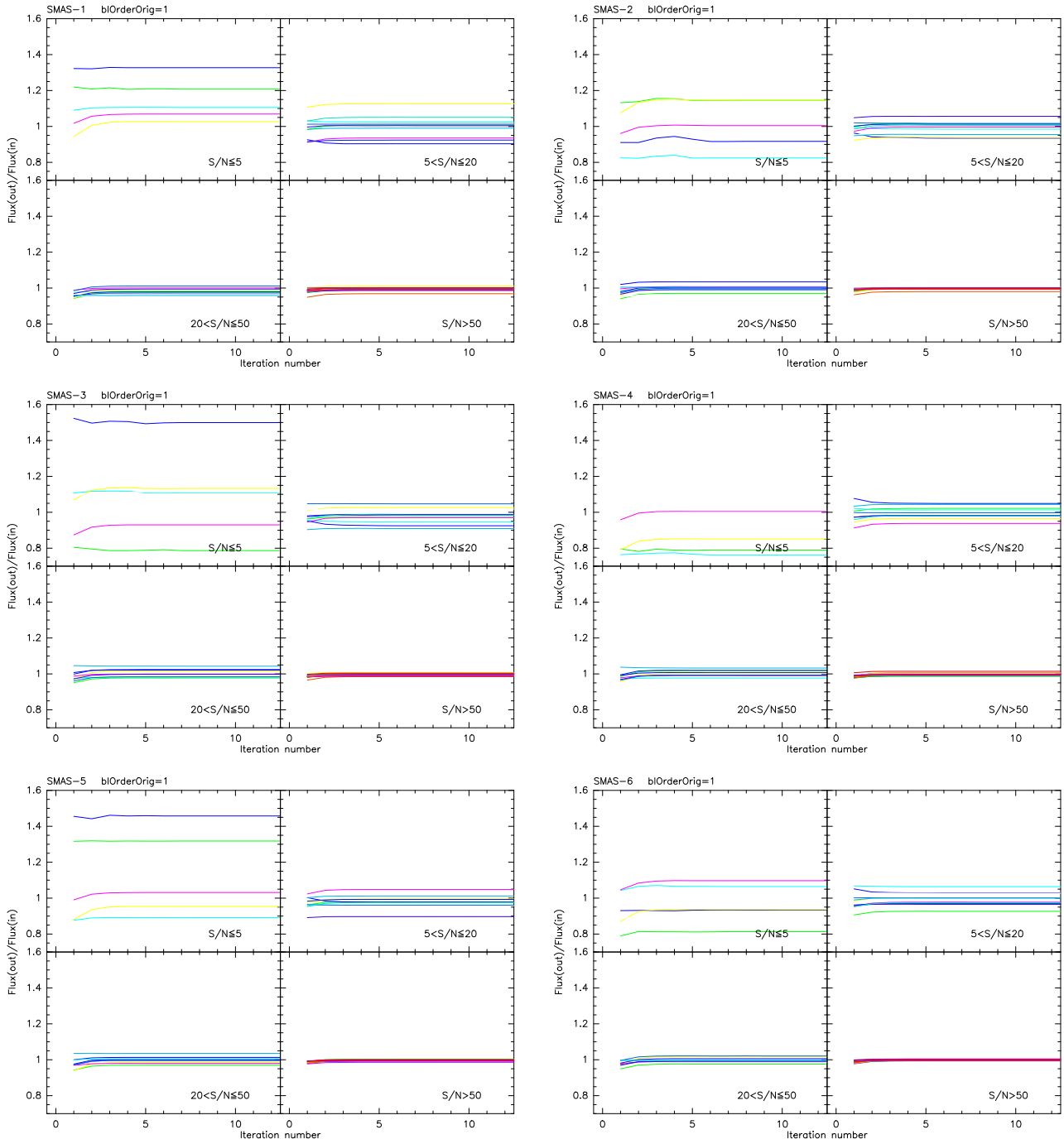


Figure C.4: Result of the six SMAS simulations. The fraction of recovered flux is shown for each artificial source as a function of iteration number, in four bins of S/N ratio. Each colored line belongs to an individual artificial point-like source. For ease of identification, the colors of the lines are the same for the 6 different SMAS and thus belong to the same point-like source. The sources (and the colors) are also the same as in Figs. C.1, C.2, C.5, etc.

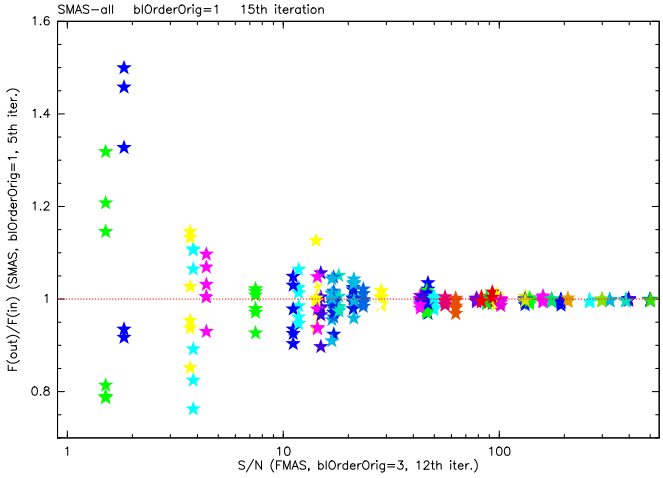


Figure C.5: Total recovered flux in the last iteration of the six SMAS as a function of S/N. See text and the captions of Figs. C.4, C.1 and C.2 for more details.

performed on other kind of extended sources (e.g. Galactic clouds or SZ negative signals); we invite each scientific project to carry out their specific TF analysis.

Because of the null map technique, the properties of noise on the resulting map are altered with respect to those of the noise of the final map that one would obtain without multiplying each second scan by -1. As shown before for the case of point-like sources, in fact, the noise on the scans is not perfectly symmetric, because of (sky and instrumental) instabilities, baseline subtraction residuals, etc. Therefore spurious effects can be produced by the null map processing. An analysis of possible caveats related to the NM approach is discussed in what follows. The effects of enhanced noise at the edges of the maps are also studied.

As brief overview, the following cases are analysed and synthetically described in this Section:

1. AS, the artificial map, without any Gaussian noise added, and without PIIC processing.

2. NMAS, i.e. null map plus artificial source, processed with two different RZ values (350 and 550 arcsec), sorting the list of scans only by observation time.

3. NMAS, processed with RZ=550 arcsec, sorting the list of scans by observation time *and* by scan direction.

4. NM, with no artificial source added, processed with two different RZ values (350 and 550 arcsec), sorting the list of scans only by observation time.

5. NM, with no artificial source added, processed with RZ=550 arcsec, sorting the list of scans by observation time *and* by scan direction.

6. NMAS, processed for each scan direction separately, with RZ=550 arcsec and smoothing parameter `smSNRpar`=-3.

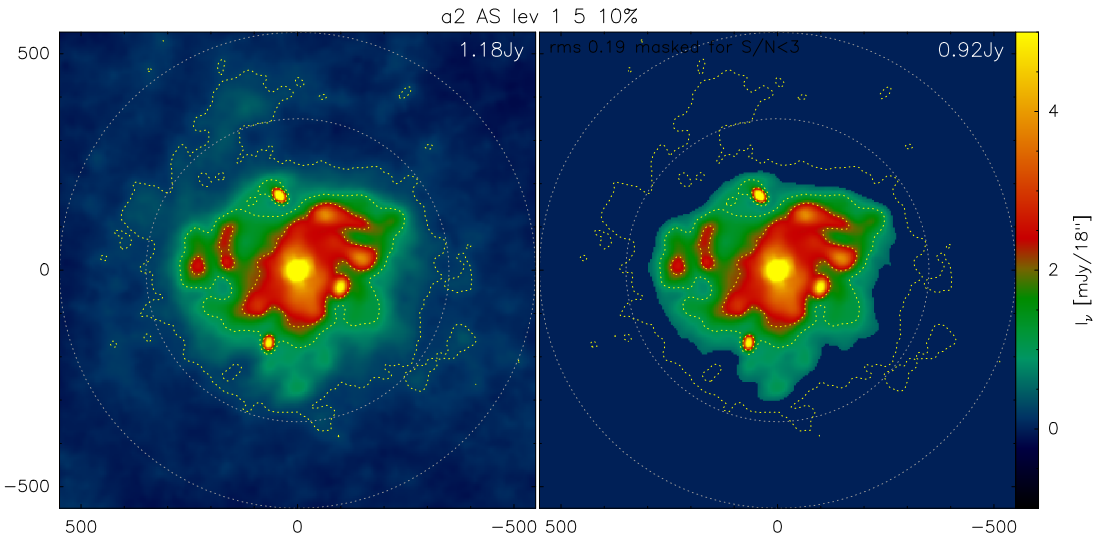


Figure C.6: The artificial source used in the *transfer function* simulation. *Left panel*: the AS as is. The two circles have radii of 350 and 550 arcsec and depict the two different RZ's used in the analysis. *Right panel*: the portion of the AS brighter than the 3σ detection limit of the PIIC final map (i.e. cut using the noise r.m.s. of the final processed map). The values reported in the top-right corner of each panel are the total flux of the source (in the right panel it's the flux above the 3σ threshold).

Unless otherwise stated, in the NMAS cases the following pieces of analysis are carried out:

- PIIC processing with four different values of the smoothing parameter (`smSNRpar` $= -1, -3, -9, -15$) and three different baseline fitting polynomial orders (`blOrderOrig` $= 2, 3, 4$).
- the resulting maps are shown and discussed.
- the total recovered source flux is analysed as a function of the processing parameters and of iteration number.
- the radial (azimuth averaged) power spectrum of the maps is computed and analysed as a function of the processing parameters.

For convenience and ease's sake, only 2 mm examples are discussed. Similar reasonings hold also for the 1 mm data.

3.3.1 Properties of the input artificial map

As mentioned above, the input AS consists of an artificial galaxy and compact sources meant to represent giant molecular clouds. The outskirts of the galaxy are extended and faint. Consequently, when considering the noise r.m.s. of the map produced by PIIC, the wings are by construction fainter than the 3σ detection limit.

Figure C.6 shows the artificial map (left panel). The two circles depict the two different zeroing radii used in the analysis (350 and 550 arcsec). In the right-hand panel, a threshold at the 3σ level of

the map produced by PIIC is applied, showing that the faint wings of this specific simulated galaxy are lost in the noise of the chosen set of real scans by definition. Above the 3σ level, one can expect to detect roughly 78% of the original total flux, in the case of this example. This percentage is therefore by construction the maximum fraction of total flux that one can retrieve after data processing.

3.3.2 The effect of the zeroing radius

The concept of zeroing radius (RZ; `rZmEq` parameter) was already described in Sect. 4.8.3: because of enhanced noise in the outer parts of the scan map, it is preferable to set to zero the source map beyond a radius of choice (or outside a user-defined polygon).

We study here the differences in the TF obtained with two different choices of the RZ: a large 550 arcsec radius circle and a smaller 350 arcsec radius one. While building the source map, the latter value allows for "pixel detections" above the chosen S/N threshold only over a smaller region centered in the scan map.

Figure C.7 shows the maps, the percentage of recovered total flux with respect to the known flux of the input artificial map, and the power spectrum of the maps, for all cases under study (i.e. the combination of the following parameters values: RZ=550 or 350 arcsec; `smSNRpar` = −1, −3, −9, −15; `blOrderOrig` = 2, 3, 4). Several effects can be identified.

First of all, when using a larger RZ, the system instabilities (atmosphere + telescope + NIKA2) can create spurious signals due to the enhanced noise in the outer regions (see a proper discussion of causes in Sect. 4.8.3), that are not fully excluded from the source map. In this situation, the fraction of recovered total flux[1] runs away and diverges, because of the spurious positive signal.

The power spectrum of the final maps seems closer to that of the input map when using a larger RZ, especially at the large angular scales (i.e. small k), which instead lose more power with smaller RZ. This might be misleading because the power spectrum analysis alone cannot tell that this large scale signal is spurious. On the other hand, note that at the medium scales the ratio of the final maps power spectrum to that of the input map is systematically larger than unity, indicating that the result is not trustworthy.

A smaller RZ guarantees that the outer "problematic" areas are fully excluded from the source map and these spurious positive signals are minimized. As byproduct, Fig. C.7 shows that 20 iterations are not enough to recover the total flux of this kind of extended source with faint wings, but a larger number of iterations is needed to converge to its asymptotic value.

The insurgence of the large scale spurious signal worsens as the smoothing of the source map becomes more important (i.e. large absolute value of `smSNRpar`) and for smaller baseline fitting order (`blOrderOrig` = 2 *vs.* 4).

Finally, the excess of power seen in the ratio of power spectra (inset of bottom plots in Fig. C.7) at ∼2.5 arcmin (k∼0.4 arcmin^{-1}) could be due to an enhancement of the source wings due to the thresholding method. When the signal in the wings is close to the adopted S/N threshold, some noise spikes can be detected above the threshold and can enhance the wings' signal. The scale at which this happens obviously depends on the shape of source's spatial profile. It is worth to note that, in the experiment presented here, this effect is minimised for RZ=350 arcsec and `blOrderOrig`=4.

[1] the percentage of recovered total flux is computed with respect to the input total flux, without taking into account that by construction only 78% is expected to be above the 3σ limit.

This indicates that, in combination to the thresholding effect, also the baseline fitting residuals play a non negligible role (see also Sect. 3.3.5).

3.3.3 Re-shuffling the list of scans

The analysis carried out here makes use of null maps. The artificial source is added on the timelines after multiplying each second scan by -1, so that the original signal for the real object is nullified, otherwise the artificial source would overlap to the real one.

Depending on the adopted observing strategy, it can happen that — in the input list of scans — odd-numbered scans are systematically taken in one scan direction in the sky (e.g. +45 deg in Equatorial coordinates) and the even-numbered scans are systematically in other direction (e.g. -45 deg). As a consequence, when producing the null-map, all scans taken in one direction are multiplied by -1 and all those taken in the other direction keep their original sign. This produces a null map that is indeed "nulled" in the central regions where all scans overlap (and where possibly the target lies) but that it is not in the areas covered by only one of the two scan directions.

This induces an additional difference in the noise properties of the outer regions with respect to the central area, and therefore a potential additional cause of spurious signals caused by baseline fitting residuals, not only in the outer regions but also in the central area itself.

Figure C.8 shows the results of processing obtained after re-shuffling the input scan list. In this experiment, the list is re-organized such that the first $\frac{N}{2}$ scans were all taken in the first scanning direction (e.g. +45 deg) and the following $\frac{N}{2}$ scans in the second scan direction (e.g. -45 deg). In this way the ± 1 multiplicative factor is applied evenly to both scan directions.

Let's compare these results to those in the *left* column of Fig. C.7. The spurious positive signal, that was produced before, has now mostly disappeared, suggesting that the outer regions have now similar noise properties as the central area of the map. On the other hand, one can notice that the percentage of recovered total flux is lower in this case with respect to the result obtained with RZ=350 arcsec and no re-shuffling of the list of scans. At the medium angular scales the ratio of the power spectrum of the final maps to that of the input artificial map is not systematically larger than unity but is not flat: it has a tendency to be < 1 at $k \sim 0,6$ arcmin^{-1} and to increase towards smaller angular scales. In other words, simply adopting a smaller RZ (when possible) seems to be more effective solution than re-sorting the list of scans.

Note, obviously, that more complex configuration can also happen, for example if three, four, or more different scanning directions were adopted.

3.3.4 Caveats, warnings and null maps

The sorting approach chosen and described above might be sub-optimal and different choices could be made. For example, one could organize the list of scans such that it lists scan directions in couples, i.e. two scans in direction 1, followed by two scans in direction 2, then again two scans in direction 1, and so on, taking care to always keep the scans taken close in time also close in the list.

Note, however, that this alternative approach might be complicated by the fact that scans can be rejected during the processing, because of different reasons (e.g. high noise along the timeline). If this happens then this kind of sorting can easily be altered during the processing. To be effective,

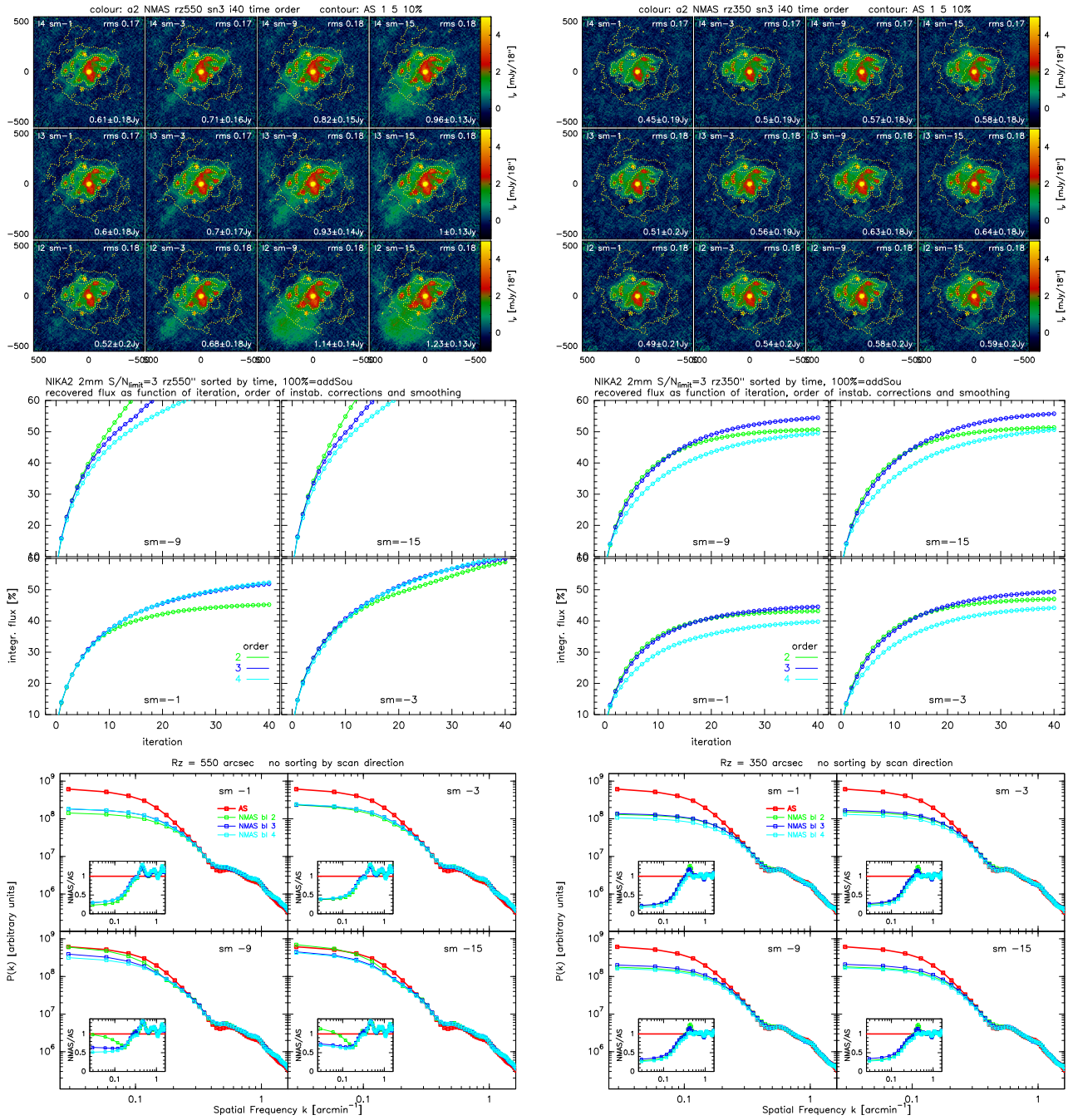


Figure C.7: The effect of different RZ values. *Left* panels show results obtained with RZ=550 arcsec; *right* panels with RZ=350 arcsec. *Top figures*: maps built with three different choices of `blOrderOrig`=4, 3, 2 (rows) and four different values of `smSNRpar` = −1, −3, −9, −15 (columns). *Middle figures*: percentage of recovered total flux for all studied cases. *Bottom figures*: power spectrum of the final map in all studied cases, and ratio to the power spectrum of the input artificial map (inset).

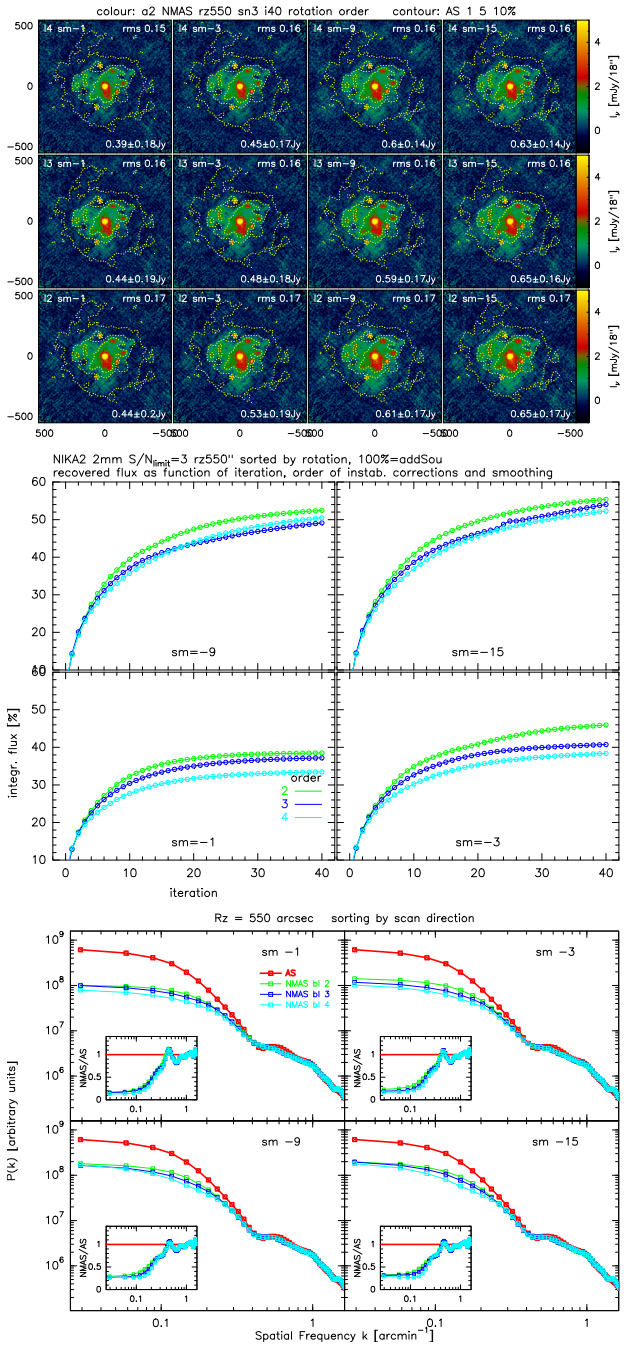


Figure C.8: The effect of sorting the input list of scans by scanning direction, to be compared to the *left* column of Fig. C.7. Here a value of RZ=550 arcsec is adopted. *Top*: maps built with three different choices of `blOrderOrig`=4, 3, 2 (rows) and four different values of `smSNRpar` = −1, −3, −9, −15 (columns). *Middle*: percentage of recovered total flux for all studied cases. *Bottom*: power spectrum of the final map in all studied cases, and ratio to the power spectrum of the input artificial map (inset).

one should first verify which scans will be excluded, by running a first "dummy" processing aimed only at identifying them.

The condition that, in order to properly produce the NM, the number of scans multiplied by -1 needs to be the same number of those "multiplied" by $+1$, is a simplification. To be fully correct, the condition is that the number of data records multiplied by ± 1 must be the same across the whole final map, per main beam angle. A compromise between the two conditions is that the number of maps multiplied by ± 1 must be the same across the whole final map, per main beam angle. Deviations from these conditions produce NM's with different degrees of approximation.

In any case, re-sorting the list of scans has some non negligible consequences on the properties of the final maps. Being the calibration of the data never perfect, re-ordering the list of scans necessarily produces different NM (for the reasons written few lines above).

The so-called "system instabilities" can be corrected only to some extent therefore there are residuals. Re-shuffling the list of scans, these residual instabilities are better suppressed when multiplying each second scan by -1 and the null map appears "cleaner". Figure C.9 shows the NM obtained with usual combinations of `smSNRpar` and `blOrderOrig`.

On the other hand, when dealing with real data to produce the final map (FM), i.e. not using the null map option, these residual instabilities are never suppressed because no scan is multiplied by -1. The final consequence is that the NM obtained by reshuffling the list of scans is no more representative of the real FM, and therefore the analysis of the TF might not be representative of the effects induced by the data processing on the final measurements (e.g. it could be too optimistic or pessimistic).

For example, if the effect of the residual instabilities is to produce a negative (spurious) signal, then this produces a dip in the FM and the measured flux of the real (positive) source will be lower. Re-shuffling the input scan list, instead, suppresses this effect and the TF calculated from the NM is different from the real TF associated to the FM.

The message here is therefore that re-sorting the input scan list must be done carefully and *only* after an in-depth thinking of the consequences. In the end it's a matter of purpose: some pieces of analysis might still benefit from re-shuffling the input list of scans, but some others (e.g. the TF described here) might be mislead by such choice.

3.3.5 Instabilities induced by the scanning direction

As known, scanning in the sky crossing different elevations produces large variations of "sky noise", because the sub-scans cover a wide range of airmasses. Scanning up and down across airmasses produces the "zig-zag" behaviour of the raw signal seen in the top-left panel of Fig 7.1. The baseline polynomial fit removes most of this zig-zag pattern, but after this process there are still KIDs that show the imprint of the total-power zig-zag on much lower level, hidden within the uncorrelated noise. This low-level zig-zag shows up in the final maps and depends on the scan direction in the sky. Maps obtained scanning at smaller angles in Alt-Azimuthal coordinates (i.e. with small variations of Elevation) show the smallest effect.

A thorough study of these instabilities carried out by R. Zylka in Summer 2020 to Winter 2021 has shown that this is just a base line problem and does not affect the flux calibration down to the

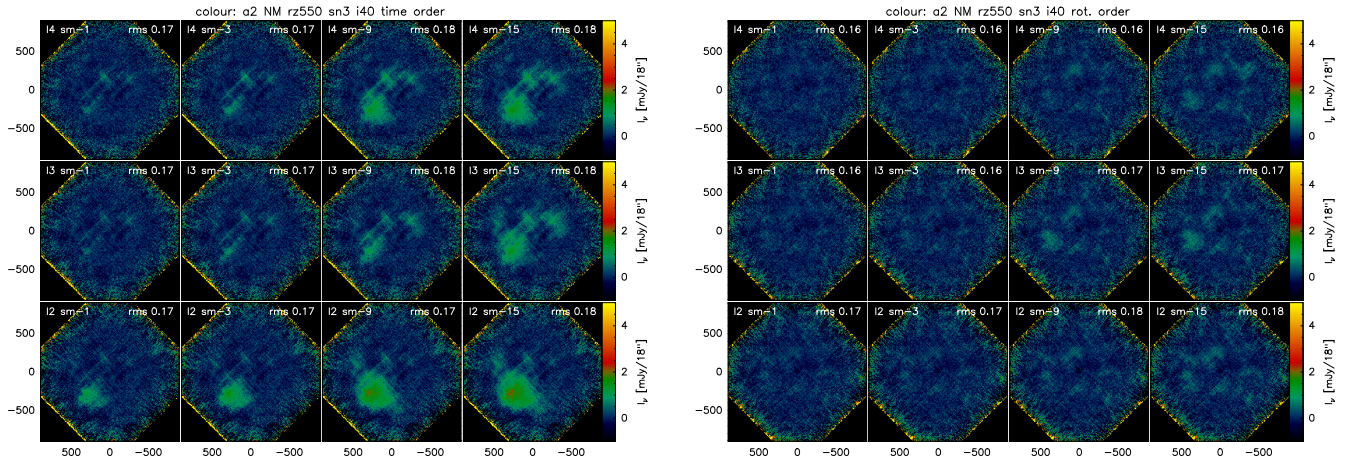


Figure C.9: Null maps obtained without re-sorting the input list of scans (*left*) and re-shuffling it (*right*). A zeroing radius RZ=550 arcsec is adopted. The rows belong to the three different choices of `blOrderOrig`=4, 3, 2; and the columns differ by the four different values of `smSNRpar` = −1, −3, −9, −15.

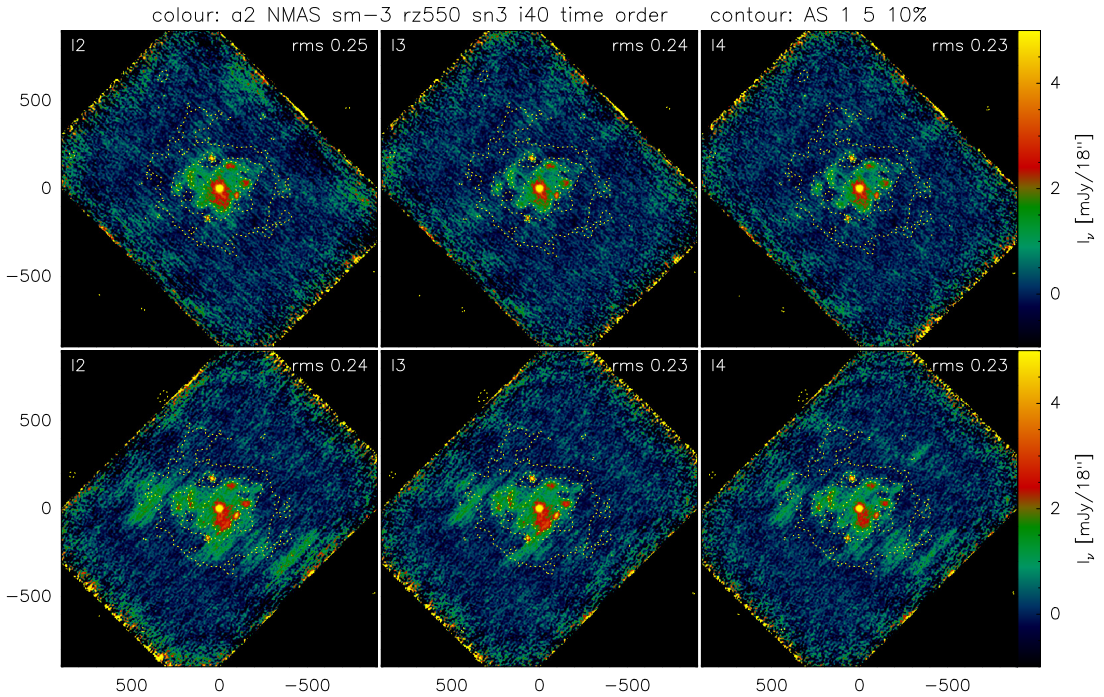


Figure C.10: Spurious signals due to system instabilities can depend on scanning direction in the sky. This example shows the result obtained with RZ=550 arcsec and `smSNRpar`=-3 for two different scanning directions (*top* and *bottom* rows) and three different values of `blOrderOrig` (from *left* to *right* = 2, 3 and 4).

mJy scale nor the size of sources on the final map. These instabilities visible in the final maps could be in principle minimized via base line fitting, provided that the sub-scans are sufficiently long, i.e.

for each KID the sub-scans have enough data on both sides of the source. This is not always the case, therefore in some cases this correction can be performed only approximatively.

The spurious signal induced by instabilities can also depend on scanning direction. The cause is similar to what described before: if the baseline correction cannot be well characterized, then the residuals can be identified by the S/N thresholding step and can produce "spurious sources". Since the amplitude of the baseline subtraction residuals depends on scanning direction, also the insurgence of spurious sources depends on it. Figure C.10 shows an example in the specific case of RZ=550 arcsec and `smSNRpar`$=-3$.

To this scanning effect, also possible intrinsic differences between NIKA2 KIDs and electronic boxes come into play and complicate the way the spurious signals appear. A discussion about these extra effects would be very technical and is beyond the scope of this Appendix.

The residual instabilities depend on the baseline fitting order and typically produce a "wavy" residual structure in the background of the final map, with regions lying on crests and valleys of the waves. The number and position of these crests and valleys depend on the adopted polynomial order (like harmonics). The left, middle and right panels in Fig. C.10 exemplify this effect clearly: the wavy pattern changes depending on baseline fitting order.

Appendix D

PIIC dictionary

Starting from version 5.5 of this PIIC tutorial, we include this dictionary that lists the commands available in PIIC and their use. Since compiling the full dictionary is a big project by itself, we start with the most useful (in our opinion) ones and we will fill this Appendix slowly with time, on a best effort basis.

The final goal is to have a full description of all available commands, but it will take time. A a start, the first version (PIIC tutorial 5.5) is rather short. We invite the users to write to *piic@iram.fr* and let us know what commands they find useful and they would like to see described here first.

?:
`? piic-command-name`
Calls the PIIC help/description of a given PIIC command.

help:
`help piic\`
GILDAS command listing all PIIC available commands.

bsm:
`bSmooth nX [nY] /whole`
Box smoothing: performs a resampling of the loaded data. If the RGDATA are a 2D map or a cube, the resampling is performed in the spatial dimensions (X, Y). The resulting pixel size in x is $nX \times xInc$, in y is $nY \times yInc$. If only nX is specified, the same box size is used in both spatial dimensions. If the loaded data are a time sequence, X is considered to be the scanning or the time coordinate.

ind:
`inDir [NAME]`
Shows or defines the input directory where the data are stored.

inl:
`inList NAME[.EXT] [off,phases,signals] /only names`

Loads and/or appends the list of observations. The lists are loaded additively.

init:
`initialize backUp, baseRange, daps, j, h, k, ffcIter, gFits, grid, grs, grw, history, lFits, inList, outList, spikePointer, subscansMasked, taus, LST, AZIM, ELEV, AZoffs, ELoffs,vm`
Initializes the buffer or data specified by the string in the command call, that is sets it back to no content.

sort:
`sort x|y|xysw [in x|y]`
Sorts the XYSW-data in X or Y.
`sort inList [hpbw, julian_date, lst, resolution, seeing, rms]`
Sorts the input list of observations on the basis of the specified parameter (default: modified Julian date).

store: stores the data structure (RGDATA) in the backup buffer.
`store [in][NAME]`
Saves/stores the current data in the backup buffer [in the buffer NAME] or the NAMEed parameters. Each data type has a specific storing buffer. Only few can be intechangeable, that is have the same format.
`NAME` can be: `#, backup, iqus, iquw, lst, rgs, rgw, signals, s, weights, badPixels, baseRange, flatField, blankPointer, spikePointer`. A simple number (#) indicated the buffer number, as listed above. IQUs and IQUw are polarimetry cubes (signal and weights) and have a similar format to BACKUP, RGS, RGW.
`store inList`: stores the input list
`store mask`: stores the information about masked pixels
`store centre`: stores the coordinates of the centre position
`store rpps`: stores the Receiver Pixel Parameters
`store rms`: stores the calculated rms as an input list parameter
`store weighting rms|rms2 in|out baseRange`
`store weighting rms|rms2 in|out [#1 #2] /db /hysteresis /offset #1 #2 /pa # /posi H M S.S D AM AS.S /system eq|hor`
`store weighting rms|rms2 /azoff < #1 > #2 /mode rp|subscan /pa # /posi H M S.S D AM AS.S`
`store weighting rms|rms2 /scancoo < #1 > #2 /mode rp|subscan /pa # /posi H M S.S D AM AS.S`

subCube: `subCube [cursor|BLC TRC]`
Used to extract a subsection of the loaded data (a 2D map or a cube). The selection is performed in the first two coordinates (e.g., the spatial (X, Y)). It can be used with the cursor to define the two diagonal corners that define the section to be extracted. Or it can be used specifying (X_1, Y_1) and (X_2, Y_2) of the bottom-left (BLC) and top-right (TRC) corners.

take: recovers a buffer or takes sections of data.
`take [from] [NAME] [notStored] [dataOnly] [doit]`:
Recovers/takes the whole data structure [only intensities] from the backup buffer [from buffer NAME] or the NAMEed parameters
`NAME can be (see store): #, backup, iqus, iquw, lst, elevation, rgs, rgw, signals, weights, weighting, x, y, badPixel, baseRange, flatField, blank, mask, spikePointer, inList, rpps`
`take plane #1 [#2 [signalsOnly]]`:
Selects the plane #1 [planes from #1 to #2]; i.e. deletes the other planes. If the users then wants to analyze another plane, the cube needs to be re-loaded (or it needs to be stored before in IQUs or IQUw).
`take medRP`:
Replaces the current signals by the median of RPs (receiver pizels / KIDs); see also `calculate + and - medRP`.
`take medSignal`:
Replaces the current signals by the median signal over all or selected receivers which are not on the source; see also `calculate + and - medSignal`.
`take rp #1 #2 ... #n`: selects the data of the defined Recever Pixels, that is it deletes the data of all other pixels
`take rp dark`: selects the data of the NIKA2 dark pixels
`take x from GreG`: loads the x buffer from GreG
`take y from GreG`: loads the y buffer from GreG
`take s from GreG`: loads the s buffer from GreG
`take xys from GreG`: loads the x, y and s buffers

Appendix E

History of PIIC releases

- **October-November 2019**: first public release of PIIC. It included DAFs valid from June 2018 (NIKA2 run 19) to March 2019 (NIKA2 run 30) and version 1.0 of the tutorial.
- **December 2019**: new DAFs, covering the period from June 2018 (NIKA2 run 19) to November 2019 (NIKA2 run 38).
- **January 2020**: 2nd release of PIIC, including the treatment of records affected by telescope tracking problems and the detection of negative sources, among other changes.
- **March 2020**: new DAFs, covering all NIKA2 science pools so far, from October 2017 (NIKA2 run 12) to March 2020 (NIKA2 run 43).
- **May 2020**: 3rd release of PIIC. The main changes are:
  - new version of the variable `blOrder`, now also depending on scan length (if needed);
  - smoothing of the source map;
  - implementation of `nBest2`;
  - add artificial sources to the timeline;
  - computation of a null-map (a.k.a. "jackknife");
  - save re-gridded maps of individual scans;
  - pointing correction between scans (incl. intensity corrections);
  - new, updated tutorial (v.2.2).
- **April 2021**: new DAFs, covering all NIKA2 science pools so far, from October 2017 (NIKA2 run 12) to March 2021 (NIKA2 run 51). The DAFs repository now includes also a document that summarizes the quality of calibration for all NIKA2 science runs.
- **May 2021**: 4th release of PIIC. The main changes are:
  - few parameters moved from the main template script to the optional settings script, including the r.m.s. polygon;

  - the new default setup implies that PIIC defines automatically the r.m.s. polygon, with no need of manual intervention by the user.
  - hit maps can now be produced;
  - new scripts available to produce r.m.s. and S/N maps, and to produce source maps;
  - new version of the PIIC tutorial (v.3.1), including a description of DAFs, and the analysis of the effects of gridding on point-like sources.

- **June 2021**: new version of the PIIC tutorial (v.3.5), now including the Appendix about the transfer function.

- **October 2021**: 5th release of PIIC. The main changes are:
  - new mode available, saving $\sim$ 30% of data reading and pre-processing time (parameter `useDAFsFR`);
  - new "final" cumulative maps available in output, allowing now to run PIIC on sub-sets of data in parallel (`wrCumLast` and `creaAver.piic`);
  - some variables changed name;
  - simplified template script: some variables have been moved to the optional settings;
  - now using the *gfortran* compiler to produce the released static executable for Linux, resulting in a 30% improvement of performance.
  - new, updated tutorial (v.3.8).

- **December 2021**: new DAFs, covering all NIKA2 science pools so far, from October 2017 (NIKA2 run 12) to November 2021 (NIKA2 run 56). The document that summarizes the quality of calibration has been updated accordingly.

- **March 2022**: 6th release of PIIC and new DAFs. The main changes include:
  - new improvements to the way instabilities are treated.
  - some variables changed name;
  - simplified template script: some variables have been moved to the optional settings.
  - the "further settings" script has changed name into "optional settings", so to stress that it is not strictly necessary to use/modify the parameters that it includes (i.e. they're optional and changes are best to be done by expert users).
  - updated DAFs and calibration documentation, covering all NIKA2 science pools so far: from October 2017 (NIKA2 run 12) to March 2022 (NIKA2 run 59).
  - new, updated tutorial (v.3.9.2).

- **December 2022**: 7th release of PIIC and new DAFs. The main changes include:
  - new flux calibration refinement method based on sky load, determining the atmospheric extinction along the line-of-sight.

  - new DAFs NKSLEX type, used in the new calibration method.
  - updated DAFs for all NIKA2 runs and up to NIKA2 run 62.
  - new, updated tutorial (v.4.0)

- **February 2023**: new DAFs, covering all NIKA2 science pools so far, from October 2017 (NIKA2 run 12) to January 2023 (NIKA2 run 63). The document that summarizes the calibration accuracy has been updated too.
- **June 2023**: 8th release of PIIC. The main changes include:
  - new multi-core parallel processing scripts.
  - new, updated tutorial (v.4.1).
- **July 2023**: fixed a small conflict between the multi-core and the single-core scripts.
- **December 2023**: 9th release of PIIC. The main changes include:
  - PIIC-Pol: reduction of polarimetry data, producing Stokes I, Q, U maps and weights.
  - PIIC-Pol: prototype scripts to naively display basic polarization information.
  - new, updated tutorial (v.5.0).
- **Ferbuary 2024**: updated tutorial (v.5.2).
- **May 2024**: new DAFs and calibration documentation, covering all NIKA2 science pools from October 2017 (NIKA2 run 12) to March 2024 (NIKA2 run 66).
- **July 2024**: 10th release of PIIC. The main changes include:
  - PIIC-Pol: new way of subtracting the parasitic signal, taking into account the source map, controlled by the parameter `psExSou`.
  - PIIC-Pol: new script to correct the instrumental polarization.
  - PIIC-Pol: few other minor upgrades.
  - new variable `pauseIfReadErr`.
  - new, updated tutorial (v.5.3).
- **December 2024**: 10.1 release of PIIC, including:
  - debugged script that performs the correction of instrumental polarization.
  - new DAFs up to the NIKA2 run 68 and updated calibration summary documentation.
- **January and February 2025**: two releases of DAFs, up to the NIKA2 runs 69 and 70, respectively, including the updated calibration summary document.
- **February 2025**: updated PIIC tutorial (v. 5.3.1), including new details about the PIIC-Pol data reduction and a warning about known effects on extended sources.

- **March 2025**: release of new DAFs, up to the NIKA2 run 71, including the updated calibration summary document (v. 3.6).
- **March-June 2025**: NIKA2 upgrade. New Ar1 and Ar3; new corrugated lenses, new IR-blocking filters; new (old) dichroic. Expect a nice improved performance!
- **June 2025**: 11th release of PIIC, including:
  - Production and use of triple IQU source maps in PIIC-Pol.
  - New IQU scripts `crea*_IQU.piic`.
  - Updated DAFs, covering up to the NIKA2 run 72.
  - Updated PIIC tutorial (v. 5.4).
  - Updated calibration documentation (v. 3.8).
- **November 2025**: 12th release of PIIC, including:
  - Updated PIIC-Pol processing, fixing some inconsistency.
  - New script to combine maps of individual scans.
  - Updated DAFs, covering up to the NIKA2 run 75.
  - Updated PIIC tutorial (v. 5.5), including (among others): how to reduce moving targets data; improved PIIC-Pol description; first version of the PIIC dictionary (limited to few commands).
  - Updated calibration documentation (v. 3.9.0).
- **December 2025**: release of new DAFs, up to the NIKA2 run 76, including the updated calibration summary document (v. 3.9.1).
- **January 2026**: release of new DAFs, up to the NIKA2 run 77, including the updated calibration summary document (v. 3.9.2).
- **February 2026**: release of new DAFs, up to the NIKA2 run 78, including the updated calibration summary document (v. 3.9.3).
- **March 2026**: 12th release of PIIC, including:
  - New PIIC-Pol scripts for IP correction using QSOs (see Sect. 6.10).
  - Updated PIIC tutorial (v. 5.5.2) with new sections on the IP correction with QSOs.
  - Updated DAFs, covering up to the NIKA2 run 79.
  - Updated calibration documentation (v. 3.9.4).
- **March 2026**: new intervention on the NIKA2 hardware: new Ar3 and focus difference between Ar1 and Ar3 solved. Expect a further improvement in the 1.2 mm performance!

- **April 2026**: release of new DAFs, up to the NIKA2 run 80, including the updated calibration summary document (v. 3.9.5).
- **July 2026**: 13th release of PIIC, including:
  - change to circular mean of the parallactic angle (instead of regular mean);
  - improvement of read/write routines to better preserve/transfer headers;
  - increased max number of PIIC sessions from 99 to 999 (now effectively working, but be careful not to fill your disk with too many LOG/MES files!);
  - more old bugs fixed;
  - new PIIC-Pol script `plPol3.greg`, plotting the $I_{\rm pol}$ map, Stokes I contours and polarization vectors;
  - updated DAFs, now including the x-talk analysis of the NIKA2 runs 72 to 80.
  - updated calibration summary document (v. 3.9.7)
  - updated PIIC tutorial (v. 5.5.3)

PIIC Cook Book

Stefano Berta

April 7th, 2026

Acknowledgements: this short, new cookbook was born from an idea of M. Beltran, A. Bongiovanni and SB, during some long winter NIKA2 run at the telescope. SB thanks them for the nice idea.

Night snacks

10 min
10 min
1 portion

While you're doing the NIKA2 observations of your favorite project, take a break and enjoy some local delicacies. Whether with Jamon or Queso, don't forget to toast your bread and to add some olive oil on top!

Preparation

1. Cut the bread in two parts and toast them in a little oven for few minutes, on both sides.
2. If your choice is Jamon, wear a steel glove, cut some and put it on top of the bread, when it's ready.
3. If you're vegetarian, go for the Queso: add a couple of slices while warming up the oven and let it melt while toasting the bread.
4. If you're vegan, enjoy the avocado variant. If not, consider adding some smashed Avocado to the Jamon.

Ingredients

Bread	1
Queso	3 slices
Jamon	N slices
Avocado	1/2
E.V.O. Oil	

Tips

Don't forget the bread in the oven, or it will burn!

Aglio Olio

15 min
4 portions

The classic midnight pasta of the good old Uni times... Awwww...

Preparation

Ingredients

Spaghetti	600 g
Water	boiling
Garlic	4 cloves
Chili	3
E.V.O. Oil	q.b.
Coarse salt	q.b.

1 Heat up 2-3 liters of water until it starts boiling. Remember that at an altitude water boils at lower temperature than at sea level, but takes forever nevertheless.

2 Peel the garlic and cut it in small pieces. Pour the E.V.O. oil in a pan, add the garlic and the chili peppers (cut or not, as you wish). Bring the pan to a mild temperature and stir from time to time with low heat. Neither the garlic nor the chili should burn.

3 When the water finally boils, add the salt and the pasta and cook *al dente*. Save some cooking water.

4 When ready, drain the pasta, place it in the pan, stir for a few moments, add some cooking water, if necessary (usually not), et voilà!

5 Buon appetito!

Tips

Avoid high temperatures, or the burned garlic will ruin this very simple recipe.

Pasta al pesto!

20 min
3 portions

The one and only pasta al pesto! Ask the PIIC developer!

Preparation

1 Heat up 2-3 liters of water until it boils. Remember that at an altitude water boils at lower temperature than at sea level but takes forever nevertheless.

2 Remove the basil leaves from the stems. At best, select only the small ones. The bigger the leaves, the more they taste like mint. Clean them, if need be, but keep them dry.

3 Peel and cut the garlic cloves in small pieces and place them together with the basil, the pine nuts and the salt in a marble mortar (LOL).

Ingredients

Pasta	500 g
Water	boiling
Basil	2 bunches
Pine nuts	30+
Garlic	1-2 cloves
Pecorino	Lots
E.V.O. Oil	q.b.
Coarse salt	q.b.

4 Nobody has a mortar these days... use a blender. Grind garlic, salt and pine nuts, blend them with the basil, until a wonderfully-smelling pesto is prepared. Let it have some darker and bigger leaf pieces (i.e. keep it coarsely ground).

5 Add the Queso and the E.V.O. oil. Be generous. Add also some more whole pine nuts. Stir with a spoon to obtain a homogeneous pesto.

6 When the water finally boils, add the salt q.b. (quanto basta – to taste) and the pasta, and cook *al dente*. Add half a cup of the cooking water to the pesto in a big bowl. When ready, drain the pasta, combine it with the pesto in the bowl and stir it.

7 Buon appetito!

Tips

Never toast the pine nuts! Keep the basil dry! Pecorino sardo, please!
Use the Pico Veleta Queso for a perfect fusion recipe.

Carbonara andalusa, a.k.a. Papalina fusion

30 min
8 portions

Born per request of Pope Pius XII, re-invented by chance, perfected over the years with the help of friends and guests from all over the world.

Ingredients

Pasta	1000 g
Water	boiling
Jamon	200+ g
Eggs	10
Pecorino	Lots
Coarse salt	q.b.
Black pepper	q.b.

Preparation

1 Boil 2-3 liters of water. Remember that at an altitude water boils at lower temperature but takes forever nevertheless.

2 Cut the Jamon in small cubes, taking care of including a more fat and less red meat. Place the cubes in a pan and brown them until the fat melts and the they get a gold-ish color without burning.

3 Crack the eggs, and separate the yolks from the albumen. Stir the yolks in a bowl. When the Jamon is golden and ready, extract it from the pan, let the melted fat cool down and mix it with the yolks.

4 When the water finally boils, add the salt q.b. (quanto basta – to taste) and the pasta, and cook *al dente*.

5 Grind the Queso and add the cooking water q.b., stir until you obtain a nice cream, not too liquid, not too dry. Add abundant black pepper.

6 When ready, drain the pasta, combine it with the Queso and pepper cream and stir it. Add the Jamon. After one minute or maybe two, add the eggs. The temperature must be right: the yolks shall not "become an omelette".

7 Serve and eat immediately. Buon appetito!

Tips

Pecorino romano, this time! But let's use the famous Pico Veleta Queso to obtain an astonishing result.

Puttanesca scomposta

30 min
5 portions

When you want something tasty, but you have enough of eggs, fat, meat, etc. etc.

Preparation

Ingredients

Spaghetti	700 g
Water	boiling
Garlic	4-5 cloves
Tomatoes	200+ g
Olives	lots
Capers	lots
Chili	3-4
E.V.O. Oil	q.b.
Coarse salt	q.b.

1 Heat 2-3 liters of water until it start boiling. Remember that at an altitude water boils at lower temperature but takes forever nevertheless.

2 Wash the tomatoes and cut them in not-too-little pieces. Peel the garlic and cut it. Cut the black olives in two halves or three rings.

3 Pour the E.V.O. oil in a pan, add all ingredients: garlic, tomatoes, olives, capers, and chili peppers (cut or not, as you wish). Stir fry at low temperature.

4 When the water finally boils, add the salt and the pasta. Drain the pasta *al dente*. Save some cooking water.

5 Place the pasta in the pan and mix it with all ingredients. Add some cooking water, if necessary, and serve hot.

6 Buon appetito!

Tips

Capern or not, use black olives and hot dried chili peppers for an explosive result!

Meringhe

30 min
120 min, 80°C
4 portions

Non si butta via niente – nothing goes to waste.

Ingredients

Eggs albumens	5
Powder Sugar	250 g

Preparation

1. After the Papalina, many egg albumens are left, and the question comes: what do we do with these?
2. Put the egg albumens in a bowl, add half of the sugar and start beating them. Stop when they're half whipped and add the remaining sugar.
3. Keep whipping until they're stiff. The mixture will be ready when it is frothy, and forms a so-called "beak" on the whisk. Try if a spoon stands unaided.
4. Form small shapes on a baking paper, leaving good free space between them. You can do this with a sac a poche or just a spoon.
5. Bake in the oven for 2 (yes, two!) hours at low temperature (80°C). Once baked, let cool down completely before removing them from the tray.
6. Buon appetito!

Tips

The night is long.

Strawberries & Lemon

10 min
2-4 portions

When you're on the mountain and you're missing love, give yourself this sweet treat. The sadness and all troubles will disappear!

Preparation

Ingredients

Strawberries	500 g
Lemon	1-2
Sugar	5 spoons

1. Wash and cut the strawberries in 2-4 pieces, depending on their size. Put them in a bowl.
2. Squeeze the lemon and extract its juice. Add it to the strawberries.
3. Cover strawberries and lemon juice with sugar, and stir thoroughly.
4. Resist the temptation to eat them immediately: put the bowl in the fridge and let the strawberries marinate for a couple of hours.
5. Serve cold and forget any lovesickness.

Tips

Try the season strawberries grown in January at Pico, and lemons from Sorrento. When the strawberries are finished, sip the lip-smacking juice!