

Welcome to the PIIC Pipelines

Pointing & Imaging In Continuum
a code by Robert Zylka

Stefano Berta and Robert Zylka

May 11, 2020

Acknowledgements: we thank A. Sievers and C. Kramer for their careful reading of the Oct. 31st 2019 version of this document and their precious comments.

Contents

1	Welcome to PIIC!	6
1.1	Acknowledging and citing PIIC	6
1.2	This release	7
1.3	The PIIC calibration files	7
1.4	Setting up PIIC	7
1.5	Starting PIIC	8
1.5.1	The graphic interface	8
1.5.2	Using external commands	8
2	Blind use of the PIIC science pipeline	9
2.1	Weak compact sources	9
2.2	Strong compact sources	9
2.3	Deep fields	9
2.4	Sunyaev-Zeldovich clusters	10
2.5	Complex sources	10
2.6	Important warning	10
3	The science data-reduction pipeline	11
3.1	Principles and challenges of NIKA2 science data reduction	11
3.2	Setup	12
3.2.1	The directory tree	12
3.2.2	The list of scans	13
3.2.3	The template script	14
3.2.4	Define your source	16
3.2.5	Specific cases	18
3.2.6	Order of baseline correction	20
3.2.7	Deep field and weak source options	20
3.2.8	Measuring the noise r.m.s.	20
3.2.9	Additional parameters	21
3.2.10	Correlating pixels	21
3.2.11	Correlating pixels: the case of extended sources	21
3.2.12	Pixel size	22
3.2.13	Exclude noisy timelines/maps	22

3.2.14	Pausing	22
3.2.15	Overwriting	23
3.3	Run it!	23
3.3.1	Information about re-gridding	24
3.4	Quick analysis of the 0-th iteration results	24
3.4.1	The intensity maps	27
3.4.2	The weight maps	27
3.4.3	Compute r.m.s. maps	28
3.4.4	Verify the source defined in the script	29
3.4.5	Define, verify and optimize the noise polygon	29
3.5	Iterative mode	32
3.5.1	Verify convergence	32
3.6	Products	32
4	Additional tips and advanced use	33
4.1	Non-azimuthal scans	33
4.2	Flux units	33
4.3	Effects of sky noise removal and of instabilities on extended and diffuse emission	34
4.3.1	Faint extended sources	34
4.3.2	The effect of NIKA2 instabilities	35
4.3.3	Uniform emission	36
4.4	Noise of individual scans	36
4.5	Number of KIDs effectively used	36
4.6	Scans of very different length	39
4.7	Cumulative signal and weights	39
4.7.1	Linking <code>rgw</code> and effective exposure time	39
4.7.2	Using <code>rgs</code> and <code>rgw</code> to produce individual scan maps	40
4.8	The source map or model	40
4.8.1	Continuing where PIIC was stopped (during the iterative mode)	40
4.8.2	Using a source model	41
4.8.3	Enhance detections by smoothing	41
4.9	Adding artificial sources to the timeline	42
4.9.1	Null maps (a.k.a. jackknife)	43
4.10	Saving maps of individual scans	43
4.11	Pointing correction between scans	44
5	The quick look monitor	46
5.1	On the fly data reduction at the telescope	46
5.2	Quick Look	46
5.3	Simple use of QL	47
5.4	Advanced example: use the QL on a refined list	48
5.5	Operations and results	49
5.6	Products	50

5.6.1	Save FITS files	51
5.7	Additional setup parameters	51
Appendix		53
A	History of PIIC releases	53

Chapter 1

Welcome to PIIC!

The *Pointing and Imaging In Continuum* software is developed by Robert Zylka at IRAM. 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.

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. **For 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 developer by adding a note in your related 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 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 paper¹ in your publication. Thank you.

¹<https://ui.adsabs.harvard.edu/abs/2013ascl.soft03011Z/abstract>

1.2 This release

The first release of PIIC was on October 2019. The software is distributed via the GILDAS pages². 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 A.

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 sweeps³; receiver pixels positions (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 τ_{225GHz} produced by the Observatory's tau-meter.

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.

At the moment, DAFS cover the period from October 2017 until March 2020. 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.

²www.iram.fr/~gildas/dist/index.html

³a *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 and the science pipeline open the graphical device automatically. 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.

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 PIIC, simply type the name of the command, preceded by the “\$” sign, e.g. `$ls`.

Chapter 2

Blind use of the PIIC science pipeline

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 follows.

Do not omit reading Section 2.6: it gives an important warning about map sizes *vs.* source sizes.

2.1 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).

2.2 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, switching the `weakSou` option from *yes* to *no*.

2.3 Deep fields

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

```
let souRmxAS 0
let souRmnAS 0
let blOrderOrig 3
let deepField yes
let nIterModel 5
```

Don't forget to define the r.m.s. polygon (`polRMSeq`) as described in Sect. 3.4.5).

2.4 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. 3.2.4).

If using the iterative mode, in order to detect the negative signal, 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 “+”.

Finally, take a look to Section 4.8.3 that describes how to use the “source map/model smoothing”: important in the case of SZ targets.

2.5 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
let souRmnAS 0
let nIterModel 10 ! for example
```

Note that — at odds with the deep fields case — now `blOrderOrig=3` and `deepField=no`.

If the dataset consists of scans of very different lengths (e.g. in different directions), please read Sect. 4.6. Moreover, Sect. 3.2.11 gives indications about advanced sky noise evaluation for this case.

2.6 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:

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

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.

Chapter 3

The science data-reduction pipeline

NIKA2 observations are carried out with a raster-scan scheme. 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. 3.2.3 and 3.2.4).

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 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 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 3.2.4). 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), also included in the data reduction script;
- a polygon defining a source-free region of sky, to be used for the computation of the noise r.m.s.

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:

```
cd
mkdir my_data_red_with_piic
cd my_data_red_with_piic/
```

```
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.

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
cd ql
cd imbfitsDir/
ls iram-30m*-2-* > ~/ql/all_ar2.LIST      ! lists all Ar2 IMBFITS files
cd ..

piic

indir imbfitsDir          ! define the input directory
inlis all_ar2              ! load the list
sel type m                 ! select scans with observation type = map
find obje /wri             ! find the names of all objects that were
                           ! observed with scans in the list
                           ! and writes one list of scans per objects
sel obje SUPERANTENNAE    ! selects only scans of object SUPERANTENNAE
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.

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, 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. 3.2.3). By default, it is “yourSource_a”’nikaBand’.LIST. In the above example, for the target called “Superantennae” and array 2, the default name is therefore `Superantennae_a2.LIST`.

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 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_wc_template.piic**. This template script needs to be edited and optimized for the given science case and the give kind of observations (target type, 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:

- **inList**, **inDir**, ...: basic filenames;
- **souRmxAs**, **souRmnAs**, **souPA**, **souRAoffAS** **souDECoffAS**: size, position angle and possible offset of an elliptical source, to be used during the 0-th iteration to exclude the area covered by the source(s), during baseline computation;
- **polSeq**: in alternative to an elliptical source, it is possible to define a polygon that describes the source, stored in a separate ascii file;
- **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;
- **nIterModel**: the number of iterations to be performed;
- **maskLevel**: the S/N threshold to be used in iterative masking;
- **polRMSeq**: the polygon within which the r.m.s. of the map will be measured.

```

mapTP_a2_wc_template.piic

!!-----
!!
!!      usage: @macroName  (.piic is default extension)
!!
!!      Written by Robert W. Zylka, zylka@iram.fr
!!      created 02.10.2016, version 22.04.2020
!!
!! Pipeline to reduce maps of sources with sizeAboveTheNoise < FoVdiameter-2*HPBW, i.e.
!!      souRmxAS < recArRmxAS-HPBW
!! for NIKA2 recArRmxAS <180arcsec.
!! It is assumed that the source is centered, i.e. appears at (0,0) in (R.A.,DEC).
!! If not the map size has to be increased by the offset values.
!! Contrary to MAMBO2 for NIKA2 data the cascade CSF cannot be used, therefore for data of sources which
!! do not satisfy the above condition the source distribution has to be supplied with sbSource or -
!! if the S/N>4 - the iterative mode has to be used (nIterModel>3).
!! Depending on NIKA2 instabilities and source structure the best in the iterative mode might be to put
!! the source size to 0, i.e. souRmxAS=souRmnAS=0. However, with this setting smooth extended emission
!! below the chosen limit of S/N will be suppressed.
!! If the maps in the scanning direction are shorter than 2*(souRmxAS+recArRmxAS+HPBW) some crucial
!! processing steps cannot be performed. As a consequence, due to inevitable data instabilities (intrinsic
!! to NIKA2 or induced by the chosen mapping strategy), the results in such cases will be questionable.
!! The only way to make the results a bit more reliable, is to set blOrderOrig to <3 and minimize the size
!! of the BASE RANGE (polygon polBReq or brRmx, brRmn).
!! Defaults are set for A2 and weak (sigMax < 5*crmsOfNIKApixel) and compact (souRmxAS < 2.5*HPBW) sources.
!!-----

@ mapTPdefs

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

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

init inList
init outList
inDir  imbfitsDir
outDir red
inList "yourSource_a" 'nikaBand'
select type m      ! only maps
!select ...      ! your choice !
!dele scan 20160215s55 ... ! your choice !
sort inList
list

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

let polSeq " "      ! polygon in EQ system defining the source border
if polSeq.eq. " " then
  if nikaBand.eq."2" then
    let souRmxAS 40 ! [arcsec] max. source radius above the noise
    let souRmnAS 40 ! [arcsec] min. source radius above the noise
  else
    let souRmxAS 30
    let souRmnAS 30
  end if
  let souPA 0.0      ! [deg] source position angle in EQ system
end if
let posSeq " "      ! centre of the final (R.A.,DEC) map and of polSeq as "H M S D AM AS"
! necessary only if individual maps have different centres
let eqExtrAS 0.0      ! [arcsec] half map extent in EQ, 0 to calculate it (possible only if all maps 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 subscan by subscan

let weakSou yes      ! glitches(=signals) > 5*rmsOfNIKApixel masked after sky noise removal
!!! WARNING: for weakSou=yes signal above ~150mJy (a2), ~650mJy (a1) and ~550mJy (a3) will be removed

if weakSou then
  let deepField no
  ! let deepField yes ! e.g. GOODSNorth, COSMOS, HDF, DeepField1, ...
else
  let deepField no
end if

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

let mxA1toiAvRMS 250.0 ! [mJy/beam/smplRateHz] exlude A1 map if the effective A1 sensitivity (<rms> in TOIs) larger than this
let mxA2toiAvRMS 70.0 ! [mJy/beam/smplRateHz] exlude A2 map if the effective A2 sensitivity (<rms> in TOIs) larger than this
let mxA3toiAvRMS 190.0 ! [mJy/beam/smplRateHz] exlude A3 map if the effective A3 sensitivity (<rms> in TOIs) larger than this
! in the optimal case the final rms in the TOI is the NIKA2 sensitivity for the used sampling rate

let sbSource " "      ! no start model of the source distribution

let nIterModel 0      ! number of iterations

if nIterModel.le.0 then
  let polRMSeq " "      ! polygon defining the region in the model sbSource to calculate the rms
else
  let snrLevel 4.0      ! data with SNR in the iterative map above snrLevel assumed as source
  let polZmeq " "      ! polygon outside of which the data are neglected (set to 0) in the iterative model
  let polRMSeq " "      ! polygon defining the region in the iterative model to calculate the rms
end if

@ mapTPFurtherSets
pause main

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

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

!!----- go! -----

@ mapTPPrun
exit

```

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

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.

3.2.4 Define your source

During the data reduction, sources need to be excluded from baseline computation and subtraction for two reasons: 1) avoid the sources in the computation of sky (noise) removal; 2) do not cross the sources while performing the baseline subtraction (which would alter the flux and the profile of the sources).

At best, the data would have already been gone through the QL (see Chapter 5), 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]¹; 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 3.2 depicts these conventions.

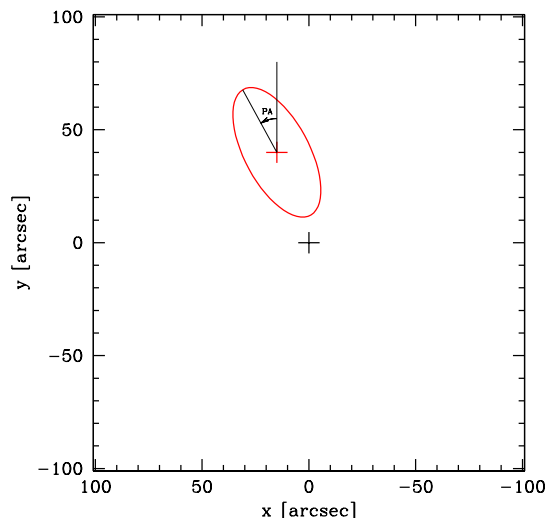


Figure 3.2: Scheme of GILDAS graphical conventions. The pixels coordinates origin (0,0) is at the center. Coordinates are defined in units of [arcsec] as in Equatorial coordinates (North is up, East is to the left). Position angles are defined starting from the y-axis, in the counter-clockwise direction. 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.

¹Note that some GILDAS component have [radians] units default. Therefore it is recommended to use the `polygon` command within PIIC.

When only a simple source is present in the maps, a simply 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 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 adaptive masking 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.

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 were 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)
```

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 tutorials²).

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 `polMeq`. This is to be found in the script `mapTPfurtherSets.piic` (see 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.

²Note that some GILDAS components adopt [radians] units, while PIIC uses [arcsec]]. Therefore use the `polygon` command from within PIIC.

Simple sources, i.e. with simple and compact geometry are simple 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 ~ 200 [arcsec]. Hence the 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).

See Sect. 4.3 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 speeding up the 0-th iteration.

Very large objects

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

In such case, the diffuse emission extended over an area larger than the NIKA2 FoV is suppressed by definition, during sky noise subtraction (see also Sect. 4.3). 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.

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. 2.6 and 3.2.4, for example). Some experimentation might be needed in order o 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 should not be modified.

It is surely possible to experiment with larger values of `blOrderOrig`, but keep in mind that enough KIDs should be available to evaluate higher-order polynomials.

Note also 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.

If the map is produced with scans of very different lengths (e.g. in different directions), please read Sect. 4.6.

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` parameter on. This activates the `weakSou` option, which 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, pixels are masked 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. 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. To do this, a polygon needs to be defined in such an area and PIIC needs to be pointed to that polygon via the parameter `polRMSeq`.

The chosen area should ideally have deep enough coverage to be representative of the goal of the project or 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 proper masking and in avoiding the creation of spurious signals.

Also in this case — as for the source polygon definition — the vertices sequence must be clockwise or counterclockwise, but not mixed (see Sect. 3.2.4 and GILDAS tutorials).

In Section 3.4.5 further details and instructions on how to define, verify and optimize the noise polygon are presented.

3.2.9 Additional parameters

The main data reduction script calls the script `mapTPfurtherSets.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 “further sets” parameters in the main (template). script, after the call to the `mapTPfurtherSets.piic` macro, or they can make a copy of the `mapTPfurtherSets.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 “further 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`!

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 “further 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 ~ 11 arcsec at 1 mm and ~ 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.

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 main data reduction scripts and are called:

```
let mxA1toiAvrms 250.0      ! [mJy/beam/smplRateHz], A1 map will be excluded
                             ! if the TOI mean r.m.s. of all KIDs larger than this
let mxA2toiAvrms  70.0      ! [mJy/beam/smplRateHz], A2 map will be excluded
                             ! if the TOI mean r.m.s. of all KIDs larger than this
let mxA3toiAvrms 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 `.err` extension.

3.2.14 Pausing

By default, 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 `nIterModel` number of times, hence producing even a larger number of pauses, which one should then manually resume by pressing “c”.

It is possible to change when and how often the script pauses, by adding the parameter `nrPause` to the script or changing its value.

By default `nrPause=5`, which means that it stops at every frame and also at every other step after reducing all individual frames. Setting it to zero, the scripts will not make any pause, if not one at the very beginning after having loaded the list of scans to be processed. 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 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 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 `mapTPfurtherSets.piic` script (see Fig. 3.1).

3.2.15 Overwriting

The execution of the data reduction scripts halts before overwriting already existing PNG or FITS files. The user may continue running the script by using the write command written on the prompt, adding the `/overwrite` option to it, and typing continue.

Alternatively, cleaning or backup-ing the `png/` and `red/` directories 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. Start PIIC in the working directory and run the script:

```
cd
cd my_data_red_with_piic/
piic dev i

@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. 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;
10. after the 0-th iteration (steps 1 to 9 above), the final map pixels with S/N ratio above the requested threshold are identified and a *source map* is produced;
11. in the iterative mode, the source map is subtracted from the data timeline (after calibration) and steps 2 to 8 are repeated `nIterModel` times.

Figures 3.3 to 3.6 show an example of the main steps for array 1, during a scan on the planet Uranus.

3.3.1 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.

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.

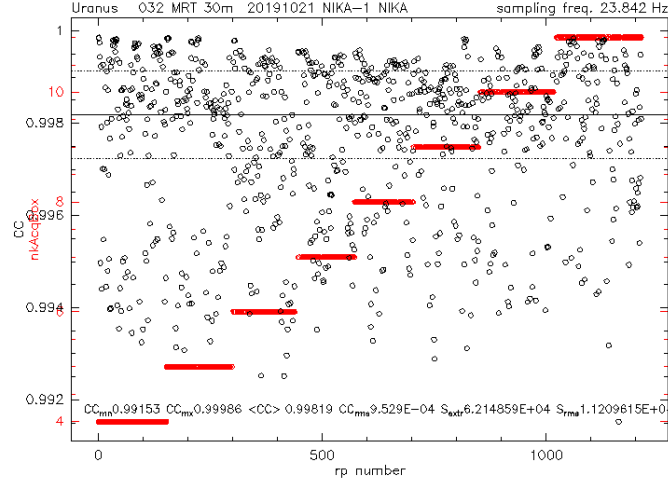


Figure 3.3: Average correlation coefficient of each receiver pixel (RP, i.e. KID) to all others, for array 1. In red are marked the RP id ranges belonging to NIKA2's 8 acquisition boxes of which array 1 consists (see also red y-axis label).

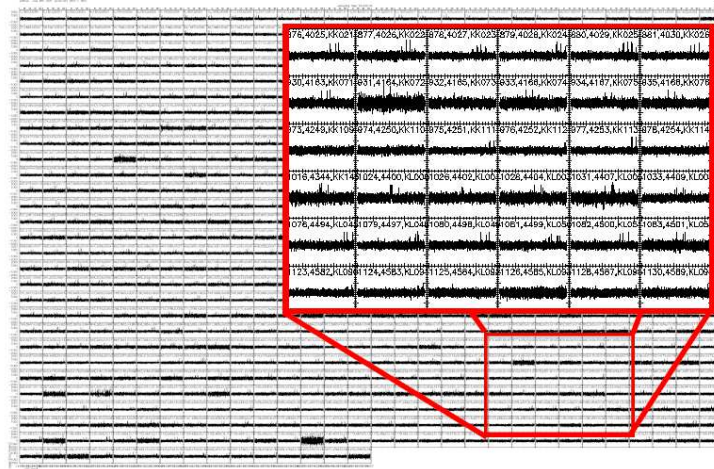


Figure 3.4: Time line of all KIDs of array 1, after the first baseline subtraction. The KIDs with the noisiest time lines are rejected (see also Fig. 3.6). The more pronounced spikes in the timelines are the bright source, crossed by the given KID at the given moment along the time line while scanning on sky.



Figure 3.5: Array-1 scan maps of each individual KID, during the Uranus observation already shown in Figs. 3.3 and 3.4. The ellipse defining the area covered by the source is depicted. Those KIDs “seeing” the source (ellipse) at the given record along the timeline are not used to compute baselines for that record.

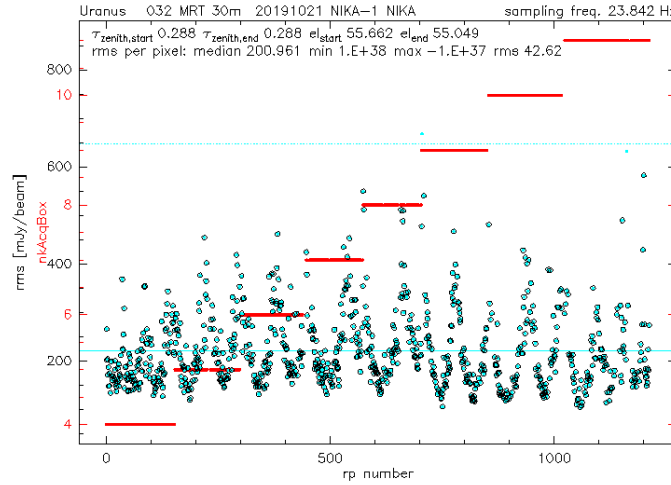


Figure 3.6: Timeline r.m.s. of all KIDs of array 1 (see also Fig. 3.4). The solid blue horizontal line marks the median value and the dot-dashed line marks a threshold set to $n0fRPrms \times$ the r.m.s. (corresponding to $n0fCCrms$ in Fig. 3.3). As in Fig. 3.3, in red are also marked the RP id ranges belonging to the 8 electronic boxes of array 1.

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

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

- 1) `Uranus_a1_OnwyyBrE60_60t113s60_60_0o0_0_0_10_50_medSb16_0_i2_0.fits`
- 2) `Uranus_a1_OnwyyBrE60_60t113s60_60_0o0_0_0_10_50_medSb16_0_i2_0_rgw.fits`
- 3) `Uranus_a1_OnwyyBrE60_60t113s60_60_0o0_0_0_10_50_medSb16_0_i2_0for1.fits`

These filenames also reflect 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.

3.4.1 The intensity maps

The intensity map combines the signal that hit 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

$$\frac{\sum_{i,j} s_{i,j} \times w_{i,j}}{\sum_{i,j} w_{i,j}} \quad (3.1)$$

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 MOPSIC data reduction software (Zylka 1998).

The WCS of the final maps is not based on any projection (e.g. TAN, Mercatore, or others). It is simply in spherical coordinates. Then, of course, it is displayed on a flat screen. This kind of WCS is called “Cartesian” in the header and in WCS jargon.

The top-left panel of Fig. 3.7 shows the 0-th iteration intensity map obtained for array 1 of our example Uranus case.

3.4.2 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, sky noise, 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

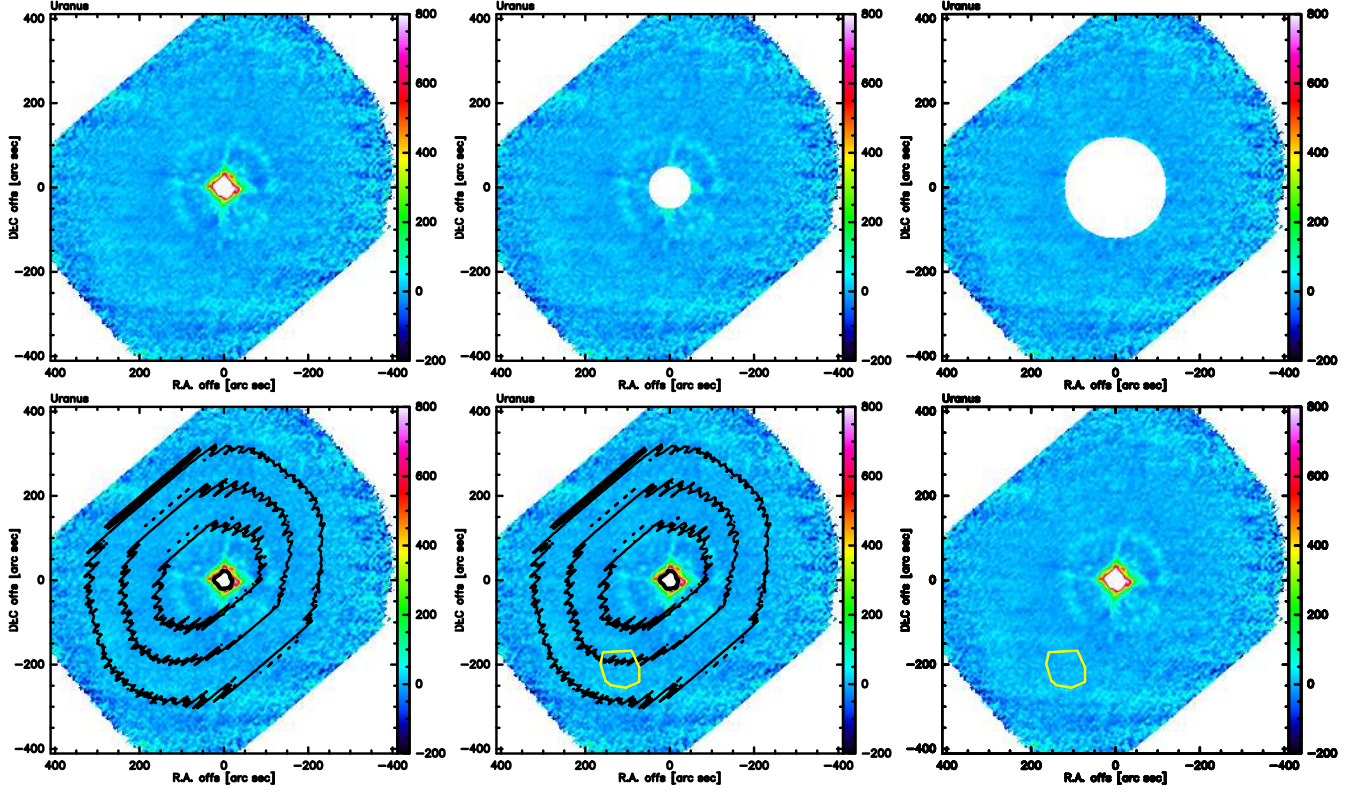


Figure 3.7: *Top-left*: array 1 intensity maps of Uranus. *Top-middle*: source area defined as a circle with radius 50 arcsec. *Top-right*: source definition with radius 120 arcsec. *Bottom-left*: 25%, 50% and 75% coverage levels (black lines). *Bottom-middle* and *right*: definition of the r.m.s. polygon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

weighting. Note, nevertheless, that even in such case, the proportionality constant between rgw and t_{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 4.7 for more on this subject.

3.4.3 Compute r.m.s. maps

As said earlier already, 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 has been previously called “the r.m.s. polygon”. The PIIC syntax to do it manually is as follows:

```
read Superantennae_rgw.fits    ! read the rgw map
init spik /all
calc ^0.5                      ! take the sqrt of the rgw
pol my_polygon                 ! 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

```

3.4.4 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.

3.4.5 Define, verify and optimize the noise polygon

The polygon, within which the r.m.s. of the map is to be measured, should be defined and given to PIIC as input. Ideally one would like to define it on the final map, in order to exclude possible sources from it, using the maximum possible S/N ratio.

For this reason, we said in Sect. 3.2.4 that ideally one would have already processed the scans with the QL, in order to have a first rough idea of where to position the r.m.s. polygon. The QL does not combine scans, therefore it can be preferable to use the results of the 0-th iteration (full data reduction, combining all scans) instead.

Produce the 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 (see Fig. 3.2 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. The coordinates are written on screen. Exit pressing “e” and save the coordinates on file or using the command **write pol filename.pol**.

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                    ! define the 90% level
rgmap /percent 1             ! draw 90% coverage contours
levels 95                    ! 95%
rgmap /percent 1
...                           ! etc
polygon my_polygon /plot     ! load and plot polygon
polygon                      ! use pol command in interactive mode to
                             ! define a new one in the desired area
```

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

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 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 my_polygon /plot
mask in
plot sca lin -1 5

```

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. 3.4.1 and 3.4.2).

The weight maps is proportional to the inverse 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.

PIIC includes a convenient command, in this case, which allows to directly compute the S/N map without the need to manually perform all operations one by one. In PIIC syntax, generating the S/N map translates in:

```

read Superantennae_rgw.fits
store rgw ! save it in the rgw array
read Superantennae.fits
polygon my_polygon
statistics in
calculate snr
plot sca lin -1 5
write Superantennae_snr.fits

```

The S/N map has been produced, plotted and saved. Now it is time to 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

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

```


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 faint and simple sources, 4-5 iterations can be enough; for extended emission and complex geometries, more than 10 iteration might be necessary.

3.5.1 Verify convergence

In order to verify if `nIterModel` 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 `nIterModel-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, then `nIterModel` 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.

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 directory `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. 4.8).
- if requested by the user, cumulative `rgs` and `rgw` maps (see Sect. 4.7) are stored in the directory `red/`.
- files containing statistical information about each scan are stored in the `stat/` directory (e.g. see Sect. 4.4).
- 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.
- all plots that are displayed on the graphical device of PIIC are also printed on file and stored in the `png/` directory.

Chapter 4

Additional tips and advanced use

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 masking 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 guess of masking at the very beginning is critical, in order to avoid producing spurious effects on the final map.

4.2 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. planets.

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 [arcsec²], is $2\pi\sigma_x\sigma_y$, where $\sigma_{x,y}$ is expressed in [arcsec] and is related to the FWHM of the Gaussian by the well known relation $\sigma = \text{FWHM} / (2\sqrt{2\ln(2)})$. In case of a circular Gaussian profile and point-like sources, we can write $\text{FWHM}_x = \text{FWHM}_y = \text{FWHM} = \text{HPBW}$.

Therefore, expressing the HPBW on units of [arcsec], we obtain:

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

Converting to steradians, one obtains:

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

The adopted HPBW values are of 11.3 and 17.0 [arcsec], at 1 mm and 2 mm, respectively.

4.3 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.

4.3.1 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 of 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`. 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 (`smModelPar`; see Sect. 4.8.3) to help detect more extension of the sources. This is highly recommended for faint extended targets.

Figure 4.1 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. 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

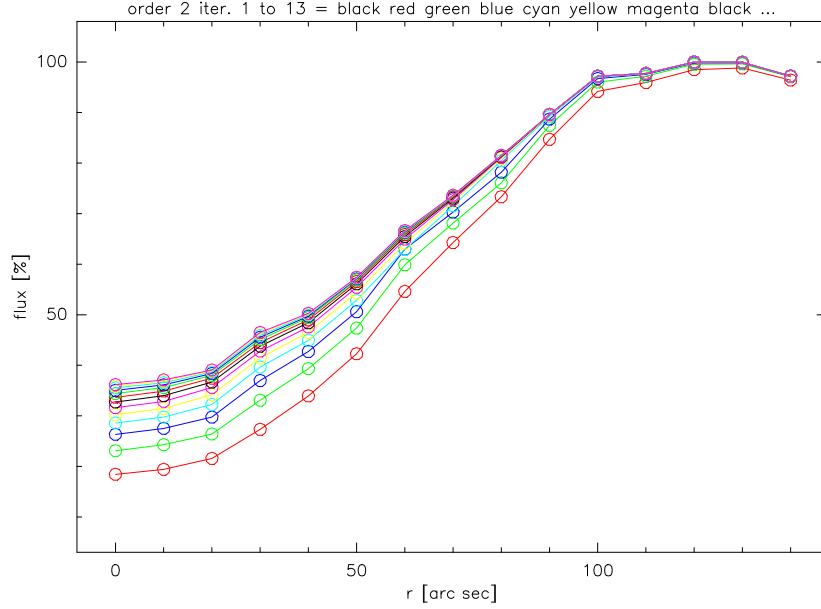


Figure 4.1: Fraction of flux recovered for an extended faint source, as a function of the source radius defined by `souRmxAS=souRmnAS`. Different lines (colours) refer to a different iteration, from 1 to 13. A `blOrderOrig=2` was used.

changes, the curves in Fig. 4.1 change. The users shall compute their own curves, if needed by their analysis (see also Sect. 4.9).

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.

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, $\sim 100\%$ of the flux is recovered, if the source is smaller than $r + x \times \text{HPBW}$. 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.

4.3.2 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. 2.6).

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.

4.3.3 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 source: 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 4.2 compares the NIKA2 beam obtained in this way to the one produced subtracting the sky-noise.

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.

4.4 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 `plRMS.greg`, included in the `pro/` folder under the PIIC installation directory. Proceed as follows:

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

One example obtained with a large dataset of more than 100 scans is shown in Fig. 4.3 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.

4.5 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:

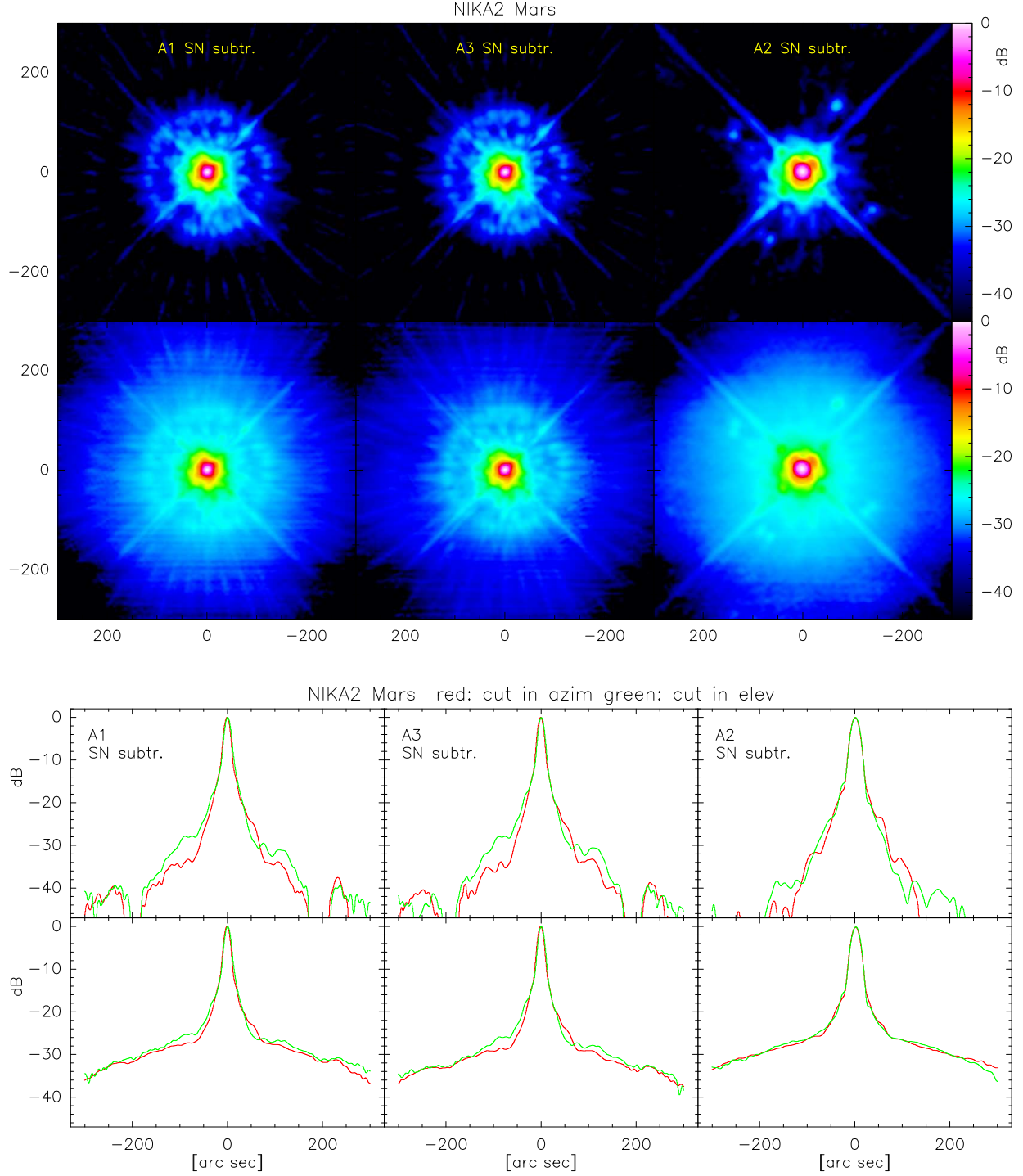


Figure 4.2: NIKA2 beam, based on Mars observations. The *top* maps and profiles show the beam as obtained with sky-noise (SN) removal. The *bottom* ones without it. The *left*, *middle*, *right* panels belong to arrays 1, 3, and 2, respectively. The profiles were obtained cutting the 2D maps along the central line and column (red/green lines).

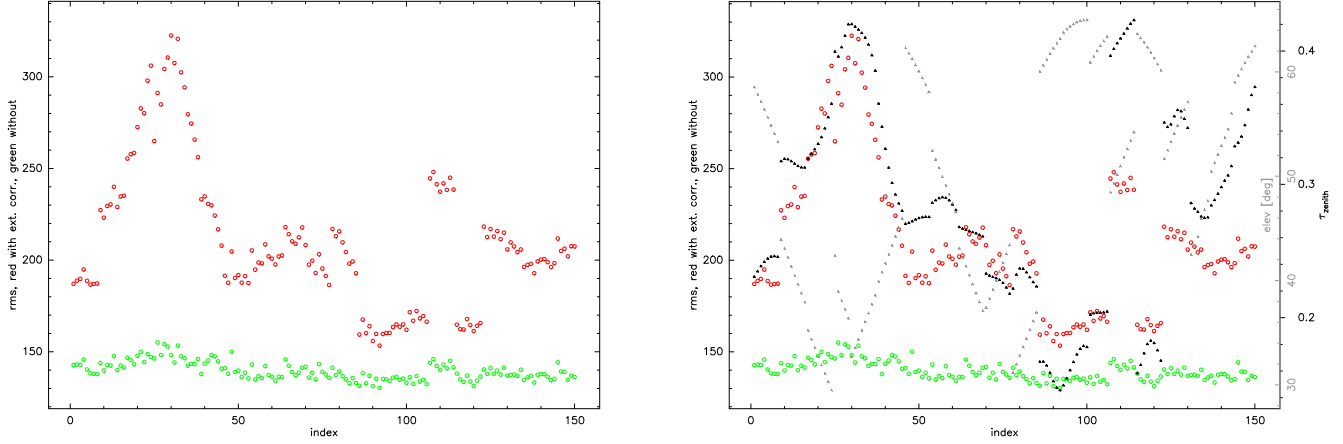


Figure 4.3: Array 1 noise of individual scans during observations. Green points depict the noise before applying the tau-correction. Red dots are the noise after the opacity correction. In the *right* panel, elevation and opacity information are plotted as grey and black symbols, respectively (see right-hand y-axes).

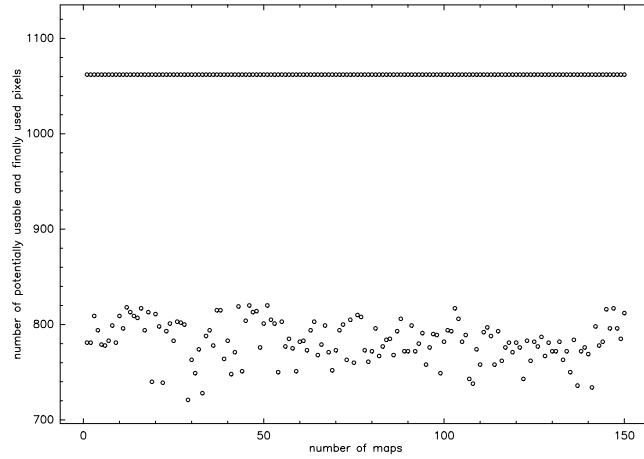


Figure 4.4: Number of KIDs that effectively made it to the final map, as part of a large dataset with more than 150 scans. For each scan, two quantities are shown: the total number of available KIDs (top sequence); and the number of KIDs that survived the different selection criteria (bottom sequence).

@plNurps Superantennae_..._i5_5 ! [e.g. for the 5th iteration]
! [omit the nUrps extension: it's added automatically]

One example, obtained with the same large dataset as before, is shown in Fig. 4.4 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.

4.6 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 `mapTPfurtherSets.piic` script, is used to this aim. If its value is larger than 0.0, the baseline polynomial order is modified such that:

$$\text{blOrder} = \text{nint} \left(\text{blOrderOrig} \times \frac{(x_{\max} - x_{\min})}{\text{blScale}} \right) \quad (4.3)$$

where *nint* represents the closest integer, and x_{\max}, x_{\min} are the maximum/minimum scan lengths. If `blScale` is set to 0.0 (i.e. its default value), no scaling of the baseline order is performed.

4.7 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 (which corresponds to the final complete map). 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 `mapTPfurtherSets.piic` script.

For each iteration, the script writes the cumulative `rgs` and `rgw` for each scan in the `red/` directory, obtained combining all scans in the list up the given one. This obviously can easily sum up to a lot of disk space, if many scans are used and several iterations are needed.

The intensity map — in units of [mJy/beam] — corresponding to the m -th `rgs` is computed from 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
```

4.7.1 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. 3.4.2) 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 not part of the standard pipeline and therefore not included in the current PIIC release. **Constructive discussions about this and other ideas of how to improve the data analysis tools are very welcome.**

4.7.2 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 i -th scan in the list by simply applying the following equation:

$$\frac{\text{rgs}(i) - \text{rgs}(i-1)}{\text{rgw}(i) - \text{rgw}(i-1)} \quad (4.4)$$

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

4.8 The source map or model

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). This is not a mask, but an actual source map in FITS format. At each iteration, this source map is subtracted from the data before performing all computations (e.g. baseline subtraction, etc., see Sect. 3.3), and then it's added back.

4.8.1 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 and to continue 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 model” (parameter `sbSource`);
- 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` model option is used; *b)* the numbering of iterations starts again from zero;
- the number of iterations `nIterModel` of the new (restarted) run needs to be adjusted, otherwise the effective total number of loops will be $m + N$, instead of simply N .

In practice, the m -th iteration is repeated, but it is a smaller loss than if one would need to repeat all the m iterations from scratch (unless $m = 1$).

4.8.2 Using a source model

It is possible to use a user-defined source model as input, to be adopted as initial source map. This model 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 model 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 model is to be considered an *expert-mode* operation. A proper definition of the model, without knowing how the emission is at the NIKA2 frequencies, is complex and could include a bit of gambling. For example, a model 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. By definition it’s just a model.

The parameter defining the file name of the input source model is again `sbSource`.

4.8.3 Enhance detections by smoothing

Excluding sources from sky noise is based on source boundaries defined by the user (elliptical or polygonal) and on the source map generated during the iterative process. The latter is based on the identification of 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 final map or to the source map. The parameter `smModelPar`, found in the `mapTPfurtherSets.piic` script, controls this feature:

- if `smModelPar` > 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 `smModelPar` < 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.

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.

Two different smoothing algorithms are possible at this time:

1. if $|\text{smModelPar}| \leq \text{HPBW}^1$ 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 `cgs` help for more details). 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 $|\text{smModelPar}| > \text{HPBW}$ then a Gaussian smoothing to the given resolution is used. In this case, the value of `smModelPar` is the final HPBW of the smoothed map, in units of [arcsec].

4.9 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 how the data reduction pipeline affects the data, computing a so-called “*transfer function*” for their typology or targets (e.g. point-like, galaxies with different profile shapes, SZ signal, 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 “further 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).

¹Values of 12 and 18 [arcsec] are adopted, i.e. slightly broader than the nominal NIKA2/30m HPBW at 1 mm and 2 mm.

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, 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.

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.

4.9.1 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 “further sets” script.

When this option is used, after the rejection of problematic scans, the signal of each second scan among those that are left² 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.

4.10 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 for 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 “further 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/model. Similarly, it is not possible to associate the use of `wrIndMaps` with the iterative mode.

²the 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.

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.
- 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.

4.11 Pointing 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. Further more some flux fraction of weak wings might decrease below the detection limit and might be missed.

It is possible to correct the pointing offsets between the different scans of a data set, by setting the parameter `nrCorrPoiCal` to 1, 2, or 3. This parameter is found in the “further 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.:

```
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 `nrCorrPoiCal` indicates the array to be used for the pointing correction. In this example, if `nrCorrPoiCal`=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:

$$\begin{aligned}\text{corr (RA)} &= \text{off (RA)}_i - \text{off (RA)}_{\text{avg}} \\ \text{corr (Dec)} &= \text{off (Dec)}_i - \text{off (Dec)}_{\text{avg}}\end{aligned}$$

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:

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

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; integrals of a Gaussian fit to the sources profiles are a better — and easy — solution.

The file `NIKA2gfEqE11i.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. `nrCorrPoiCal`) 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 offsets; or for array 2 (1) maps, one can use array 1 (2) as reference for the corrections, depending on the desired angular accuracy of the corrections or on source brightness; etc. These are just examples and other cases might happen.

The PIIC package does not include procedures to measure the position and fluxes of the sources and to build the `NIKA2gfEqE11i.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 `nrCorrPoiCal` option, one additional directory `pcc/` needs to be created. The corrections applied to each map will be written there.

Chapter 5

The quick look monitor

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.

5.1 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 2 and 3, if desired, by simply using *monitor2* and *monitor3*.

5.2 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. 5.1).

5.3 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 IMBFITS¹ 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:

```
cd
cd ql
cd imbfitsDir/
ls iram-30m*-1-* > ~/ql/ar1.LIST
cd ..
```

```
piic
```

```
@qlAr1list
```

¹IMBFITS means *IRAM Multi-Beam* FITS files; it is the standard file format of IRAM/30m and NIKA2 data.

5.4 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 iram-30m*-2-* > ~/ql/all_ar2.LIST      ! lists all Ar2 IMBFITS files
cd ..

piic

inlis all_ar2.                ! load the list
sel type m                    ! select scans with observation type = map
find obje /wri                ! find the names of all objects that were
                              ! observed writes with scans in the list
                              ! and writes one list of scans per objects
sel obje SUPERANTENNAE       ! selects only scans of object SUPERANTENNAE
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. 5.3).

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.

5.5 Operations and results

As it runs, the QL (or *monitor*) produces three 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, see left panel of Fig. 5.1) 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. 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). 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 *baselines* from the data timeline of each KID (see also Chapter 3) and produces a flattened timeline, in which the sky and instrumental “noise” is removed and only the source signal is left (right panel of Fig. 5.1).

After subtracting baselines and calibrating the data, the QL shows four different diagrams, as in Fig. 5.2. 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. 5.2 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.

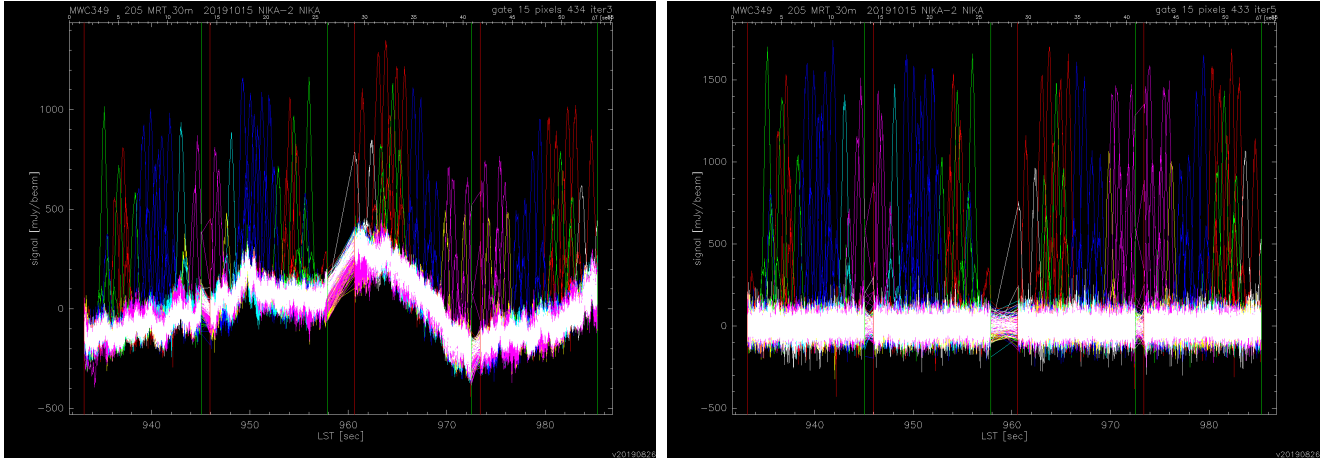


Figure 5.1: Example of timeline of a pointing scan on the source MWC349 (array 2). *Left*: Raw timeline; *right* baseline-subtracted timeline. Each coloured line belongs to a different KID (detector pixel) of NIKA2. See main text for details.

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.

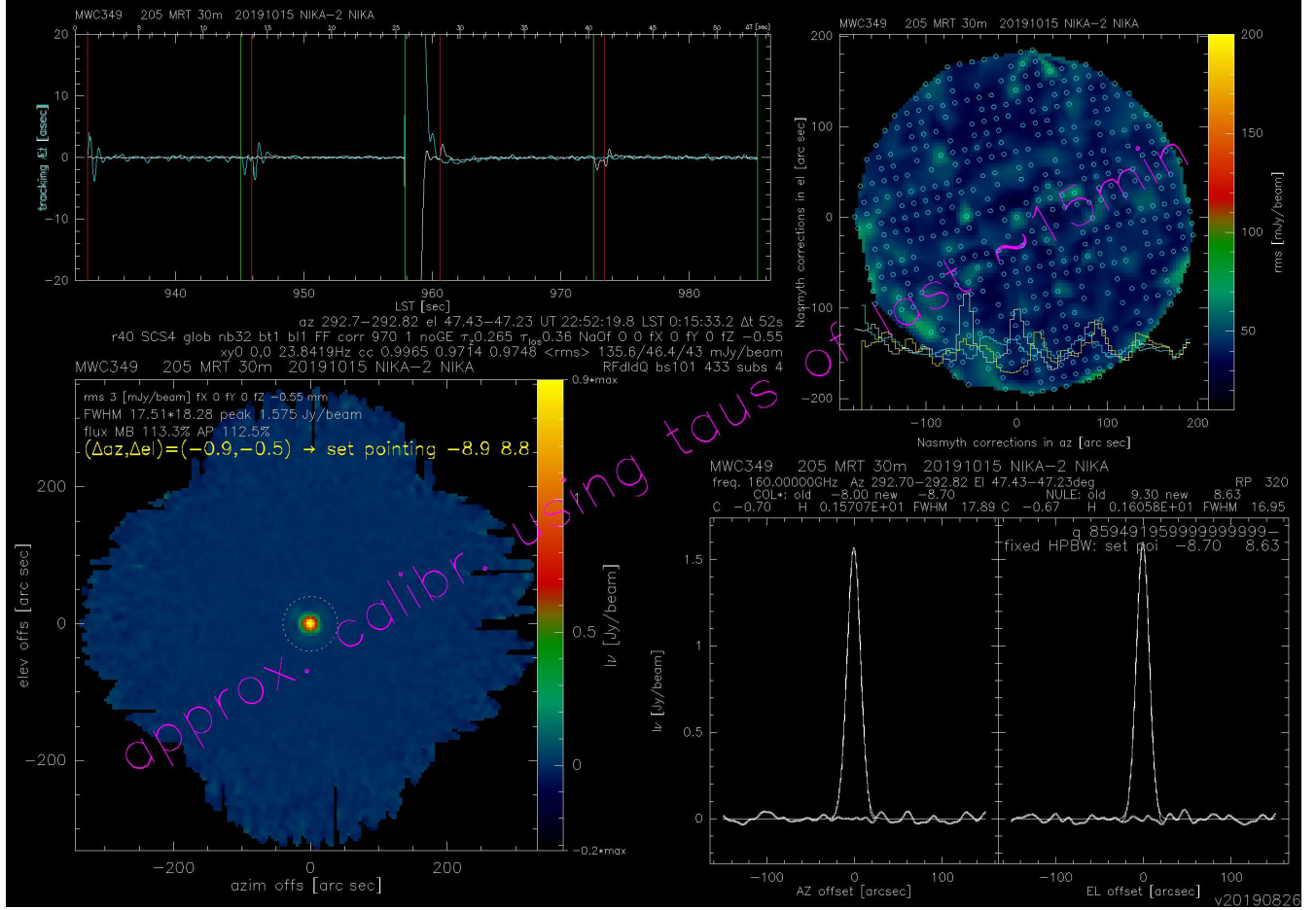


Figure 5.2: Result of QL analysis of a pointing scan on the source MWC349 (array 2). Top-left: tracking accuracy in Azimuth (white) and Elevation (cyan), i.e. difference between commanded and effective coordinates as a function of time. Bottom-left: reduced map of the scan, including information about 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.

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.

5.6 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.

5.6.1 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
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*/**.

5.7 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 times 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

Emerson, D. T., Klein, U., & Haslam, C. G. T. 1979, A&A, 76, 92

Haslam, C. G. T. 1973, Kleinheubacher Berichte, 16, 451

Zylka, R. 1998, MOPSIC Cookbook, ITA Heidelberg

Zylka, R. 2013, MOPSIC: Extended Version of MOPSI, Astrophysics Source Code Library, record
ascl:1303.011

Appendix

A 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 (run 30).
- **December 2019:** new DAFs, covering the period from June 2018 (run 19) to November 2019 (run 38).
- **January 2020:** new version 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 (run 12) to March 2020 (run 43).
- **May 2020:** new version of PIIC. The main changes are:
 - new version of the variable `blOrder`, now also depending on scan length (if needed);
 - smoothing of the source model/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.