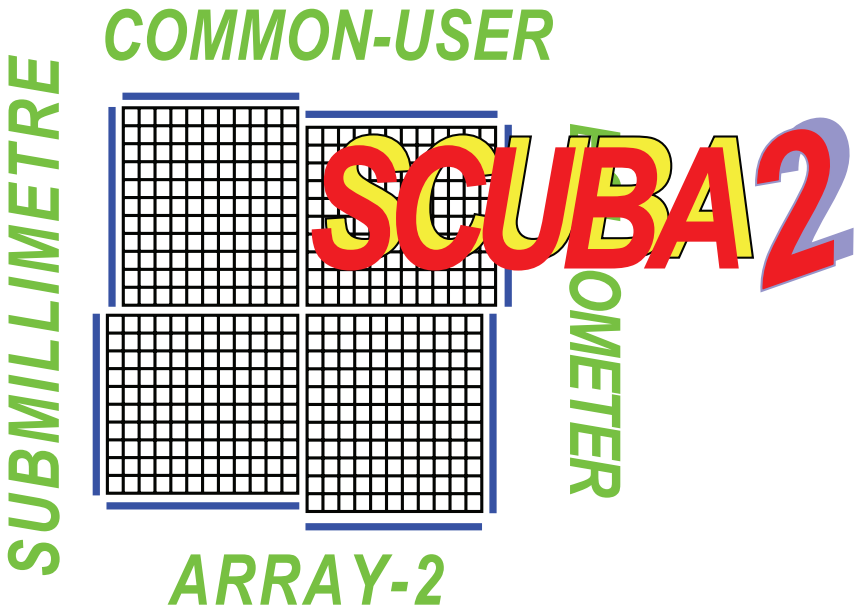
Starlink Project
Starlink Cookbook 21.2

H. S. Thomas, M. J. Currie, D. S. Berry, H. A. Parsons & S. Mairs

2021 June 23

The SCUBA-2 Data Reduction Cookbook 1.4



Abstract

This cookbook provides an introduction to Starlink facilities, especially SMURF, the Sub-Millimetre User Reduction Facility, and the ORAC-DR pipeline for reducing, displaying, and calibrating SCUBA-2 data. It describes some of the data artefacts present in SCUBA-2 time-series and methods to mitigate them. In particular, this cookbook illustrates the various steps required to reduce the data; and gives an overview of the Dynamic Iterative Map-Maker, which carries out all of these steps using a single command controlled by a configuration file. Specialised configuration files are presented.

Contents

List of Figures

Acronyms

CADC	Canadian Astronomy Data Centre
CSO	Caltech Submillimetre Observatory
DIMM	Dynamic Iterative Map-Maker
FCF	Flux Conversion Factor
FITS	Flexible Image Transport System
FWHM	Full-Width at Half-Maximum
GAIA	Graphical Astronomy and Image Analysis Tool
ITC	Integration Time Calculator
JCMT	James Clerk Maxwell Telescope
JSA	JCMT Science Archive
MSB	Minimum Schedulable Block
NDF	Extensible N-Dimensional Data Format
NEP	Noise Equivalent Power
NEFD	Noise Equivalent Flux Density
PCA	Principal Component Analysis
PSF	Point Spread Function
PWV	Precipitable Water Vapour
RMS	Root Mean Square
SCUBA-2	Submillimetre Common User Bolometer Array-2
SMURF	Sub-Millimetre User Reduction Facility
S/N	Signal-to-Noise ratio
SQUID	Superconducting QUantum Interference Device
STC-S	Space-Time Coordinate Metadata String Implementation
SUN	Starlink User Note
TES	Transition Edge Sensor
WVM	Water Vapour radioMeter

Chapter 1
Introduction

1.1 This cookbook

This guide is designed to instruct SCUBA-2 users on the best ways to reduce and visualise their data using Starlink packages: SMURF[4], KAPPA[7], GAIA[10], ORAC-DR [3] and PICARD[11].

This guide covers the following topics.

- Chapter 1 – Computer resources you'll need before getting started.
- Chapter 2 – A description of SCUBA-2 and its observing modes.
- Chapter 3 – An introduction to the Dynamic Iterative Map-Maker and a description of the configuration files.
- Chapter 4 – Instructions for using the ORAC-DR pipeline to reduce your data "the easy way", along with details on data retrieved from the JSA.
- Chapter 5 – Instructions for using individual SMURF commands to reduce your data—useful if you need extra flexibility.
- Chapter 6 – Options for tailoring the configuration parameters to improve your final map.
- Chapter 7 – Two worked examples covering a blank cosmology field and a galactic field.
- Chapter 8 – Post-processing reduction steps such as applying the FCF, co-adding multiple maps and estimating the noise.
- Chapter 9 – SCUBA-2 Diagnostic Tools.

Throughout this document, a percent sign (%) is used to represent the Unix shell prompt. What follows each % will be the text that you should type to initiate the described action.

1.2 Before you start: computing resources

Before reducing SCUBA-2 data using the Dynamic Iterative Map-Maker you should confirm you have sufficient computing resources for your type of map.

We recommend the following:

Reduction type	**Memory**
Large maps (PONG)	96 GB
Small maps (DAISY)	32–64 GB
Blank fields	32–64 GB

Why these recommendations?

For large-area maps it is important to process a full observation in a single chunk. See the text box on Page 24 for an explanation of chunking. For large maps, using normal map-maker parameters a machine having 96 GB is acceptable. It is important that the memory is as fast as can be afforded, as RAM speed has a direct linear effect on processing time given that the time-series data are continually being funnelled through the CPU.

For blank-field surveys or smaller regions of the sky you can usefully run the map-maker with less memory and 32 to 64 GB is reasonable depending on the specifics of your data set. SMURF is multi-threaded so multiple cores do help although above eight cores the price/performance gains tend to drop off.

If you have a very large machine (128 GB and 24 cores) you may be able to run two instances of the map-maker in parallel without chunking, depending on the nature of the data. Use the `SMURF_THREADS`[1] environment variable to restrict each map-maker to half the available cores.

1.3 Before you start: software

This manual uses software from the Starlink packages: SMURF [4], KAPPA [7], GAIA [10], ORAC-DR [3] and PICARD [11]. Starlink software must be installed on your system, and Starlink aliases and environment variables must be defined before attempting to reduce any SCUBA-2 data.

1.3.1 Data formats

Data files for SCUBA-2 use the Starlink N-dimensional Data Format (NDF, see Jenness et al. 2014[14]), a hierarchical format which allows additional data and metadata to be stored within a single file. KAPPA contains many commands for examining and manipulating NDF structures. The introductory sections of the KAPPA document (SUN/95) contain much useful information on the contents of an NDF structure and how to manipulate them.

A single NDF structure describes a single data array with associated metadata. NDFs are usually stored within files of type "`.sdf`". In most cases (but not all), a single `.sdf` file will contain just one top-level NDF structure, and the NDF can be referred to simply by giving the name of the file (with or without the "`.sdf`" prefix). In many cases, a top-level NDF containing JCMT data will contain other "extension" NDFs buried inside them at a lower level. For instance, raw files contain a number of NDF components which store observation-specific data necessary for subsequent processing. The contents of these (and other NDF) files may be listed with HDSTRACE. Each file holding raw JCMT data on disk is also known as a 'sub-scan'.

The main components of any NDF structure are:

- an array of numerical data (may have up to seven dimensions—usually three for JCMT data);
- an array of variance values corresponding to the numerical data values;
- an array holding up to eight boolean flags (known as "quality flags", see Section 8.6) for each pixel;
- World Coordinate System information;
- history;
- data units; and

[1] `SMURF_THREADS` should be set to an integer value indicating the number of threads to be used by each process.

- other extension items. These are defined by particular packages, but usually include a list of FITS-like headers together with provenance information that indicates how the NDF was created. Raw JCMT files also include extensions that define the state of the telescope and instrument at each time slice within the observation.

The CONVERT package contains commands fits2ndf and ndf2fits that allow interchange between FITS and NDF format.

1.3.2 Initialising Starlink

The commands and environment variables needed to start up the required Starlink packages (SMURF[4], KAPPA, *etc.*) must first be defined. For C shells (csh, tcsh), do:

```
% setenv STARLINK_DIR <path to the starlink installation>
% source $STARLINK_DIR/etc/login
% source $STARLINK_DIR/etc/cshrc
```

before using any Starlink commands. For Bourne shells (sh, bash, zsh), do:

```
% export STARLINK_DIR=<path to the starlink installation>
% source $STARLINK_DIR/etc/profile
```

1.3.3 KAPPA and SMURF for data processing

The Sub-Millimetre User Reduction Facility, or SMURF, contains the Dynamic Iterative Map-Maker, which will process raw SCUBA-2 data into images (see **SUN/258**). KAPPA meanwhile is an application package comprising general-purpose commands mostly for manipulating and visualising NDF data (see **SUN/95**). Before starting any data reduction you will want to initiate both SMURF and KAPPA.

```
% smurf
% kappa
```

After entering the above commands, you can access the help information for either package by typing `smurfhelp` or `kaphelp` respectively in a terminal, or by using the showme facility to access the hypertext documentation. See Section 1.3.7 for more information.

Tip

The .sdf extension on filenames need not be specified when running most Starlink commands (the exception is PICARD).

1.3.4 GAIA for viewing your map

Image visualisation can be done with GAIA (see **SUN/214**). GAIA is a GUI-driven image and data-cube display and analysis tool, which incorporates facilities such as source detection, three-dimensional visualisation, photometry and the ability to query and overlay on-line or local catalogues.

```
% gaia map.sdf
```

Alternatively, the KAPPA package includes many visualisation commands that can be run from the shell comand-line or incorporated easily into your own scripts—see Appendix "Classified KAPPA commands" in SUN/95. These tools are particularly useful for creating more complex composite plots including multiple images, line-plots, *etc*, such as the multi-image plots in Section 5.3.2.

1.3.5 ORAC-DR for running the pipeline

The ORAC-DR Data-Reduction Pipeline [3] (hereafter just ORAC-DR) is an automated reduction pipeline. ORAC-DR uses SMURF and KAPPA (along with other Starlink tools) to perform an automated reduction of the raw data following pre-defined recipes to produce calibrated maps. The following commands initialise ORAC-DR ready to process 850 μm and 450 μm data respectively.

```
% oracdr_scuba2_850
% oracdr_scuba2_450
```

For more information on available recipes and intructions for running the pipeline see Chapter 4.

1.3.6 PICARD for post-reduction processing

PICARD uses a pipeline system similar to ORAC-DR for post-processing and analysis of reduced data. PICARD documentation can be found at ORAC-DR web page, or at **SUN/265**. All PICARD recipes follow the same structure and are run like so:

```
% picard -recpars <recipe_params_file> RECIPE <input_files>
```

where `<recipe_param_file>` is a text file containing the relevant recipe parameters. `RECIPE` is the name of the recipe to be run (note the caps). The list of files to be processed is given by `<input_files>`. These must be in the current directory or a directory defined by the environment variable `ORAC_DATA_IN`. A number of PICARD recipes will be demonstrated in Chapter 8.

Other command-line options include `-log xsf` where the log file is written to any combination of the screen [`s`], a file [`f`] or an X-window [`x`]. Usage of `s` or `sf` is recommended as the recipes are often short. To ensure additional graphics X-windows do not open, you can use the command-line option: `-nodisplay`.

You do not specify an output filename for PICARD, instead the output is generated by adding a recipe depending suffix to the input filename. If there is more than one input file then the name of the last file is used.

You can create a file which lists the input files to be passed to PICARD for processing via the `-files` option. For example:

```
% picard -log s RECIPE_NAME -files myfilestoprocess.lis
```

Tip

Unlike other Starlink packages, the .sdf extension must be included when supplying the names of Starlink data files to PICARD.

Tip

If the environment variable `ORAC_DATA_OUT` is defined, any files created by PICARD will be written in that location. Check there if new files are expected but do not appear in your working directory.

1.3.7 How to get help

Help command	**Description**	**Usage**
showme	If you know the name of the Starlink document you want to view, use showme. When run, it launches a new webpage or tab displaying the hypertext version of the document.	`% showme sun95`
findme	findme searches Starlink documents for a keyword. When run, it launches a new webpage or tab listing the results.	`% findme kappa`
docfind	docfind searches the internal list files for keywords. It then searches the document titles. The result is displayed using the Unix more command.	`% docfind kappa`
Run routines with prompts	You can run any routine with the option `prompt` after the command. This will prompt for every parameter available. If you then want a further description of any parameter type `?` at the relevant prompt.	`% makemap prompt` `% REF - Ref. NDF /!/> ?`
Google	A simple Google search such as "`starlink kappa fitslist`" will usually return links to the appropriatre documents. However, be aware that the results may include links to out of date versions of the document hosted at non-Starlink sites. Always look for results in "`www.starlink.ac.uk/docs` (or "`www.starlink.ac.uk/devdocs` for the cutting-edge development version of the document).	

1.4 Processing options

You have two options for processing your data:

(1) running the automated pipeline (ORAC-DR), or

(2) performing each step manually.

The pipeline approach is simpler and works well if you have a lot of data to process. Performing each step by hand allows more fine-tuned control of certain processing and analysis steps, and is especially useful for refining the parameters used by the map-maker. However, once the optimal parameters have been determined, it is possible to pass them to the pipeline to process other observations using the same configuration. Chapter 3 and Chapter 5 discuss the manual approach; to use the science pipeline, skip straight to Chapter 4.

The JCMT will produce pipeline-reduced files for each observation and group of repeat observations for each night. These are reduced using the ORAC-DR pipeline with the recipe specified in the MSB. Chapter 4 gives instruction on retrieving reduced data from the JCMT Science Archive at CADC.

Chapter 2
SCUBA-2 Overview

2.1 The instrument

The Submillimetre Common User Bolometer Array-2 (SCUBA-2) is a 10,000-pixel bolometer camera. It has two arrays operating simultaneously to map the sky in the atmospheric windows of 450 and 850μm. Each array is made up of four sub-arrays as shown in Figure 2.1.

How it works

The SCUBA-2 bolometers are integrated arrays of superconducting transition edge sensors (TESs) with a characteristic transition temperature, T_c. In addition, each TES is ringed with a resistive heater which can compensate for changes in sky power. The SCUBA-2 focal plane is kept at a base temperature slightly below T_c, however a voltage is applied across each TES resistance to position the bolometer at the transition temperature. From this point, any increase of temperature on the bolometers (e.g. from an astronomical signal) will increase the TES resistance and heat it up. This causes a drop in current and therefore a drop in temperature making the system self-regulating.

For properly performing bolometers, the change in current through the TES is proportional to the change in resistance, with the response calibrated using flat-field observations (described below). This changing current generates a magnetic field which is amplified by a chain of superconducting quantum interference devices (SQUIDs). This induces a feedback current which is proportional to the current flowing through the TES, and it is this feedback current that is recorded during data acquisition.

Setups

Before science data can be taken the system must be optimised. These 'setups' are performed after slewing to the azimuth of the source, where the SQUID, TES and heater biases are set to pre-determined nominal values, in order to position the bolometers in the middle of the transition range.

Flat-field

The shutter then opens onto the sky, and as it does so the gradual increase in sky power hitting the array is compensated for by a decrease in the resistive heater power via a servo loop designed to keep the TES output constant. This acts to keep the bolometers positioned at the centre of the transition range and is known as **heater tracking**.

The responsivity of the bolometers will change slightly between the dark and the sky; therefore, once the shutter is fully open a fast **flat-field** observation is carried out to recalibrate them. **A flat-field measures the responsivity of each bolometer to changing sky power**. It does this by utilising the resistance heaters which are ramped up and down around the nominal value. The change in current through the TES is then recorded for each bolometer giving a measure of its responsivity. The flat field solution is then the inverse linear gradient of the current as a function of heater power.

At this point bolometers with responsivities above or below a threshold limit are rejected, along with bolometers that display a non-linear response or have a poor S/N. A second flat-field is performed at the end of an observation so bolometers whose responsivity has changed over the course of the observation can be flagged.

For full details of the array setup and operation see Holland et al. (2013) [12].

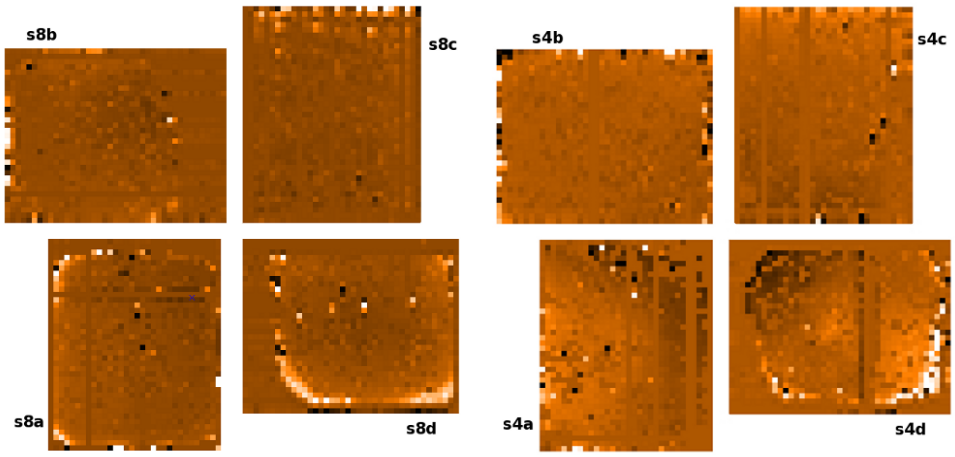


Figure 2.1: The layout of the arrays at 850μm (left) and 450μm (right). The labels denote the name assigned to each sub-array. Raw data files are generated separately for each sub-array and must be co-added. This figure was made by running wcsmosaic on a raw sub-scan from each sub-array.

2.2 Observing modes

Two observing modes are offered for SCUBA-2: DAISY and PONG. As the bulk of SCUBA-2 observing involves large area mapping, both observing modes are scan patterns. Your choice depends on the size of the area you wish to map, where you would like your integration time concentrated and the degree of extended emission you wish to recover.

PONG A PONG map is the scan strategy for covering a large area. The default options allow for three sizes—900 arcsec, 1800 arcsec and 3600 arcsec. A single PONG map is a square of these dimensions and the telescope fills in the square by bouncing off the edge of the area. To ensure an even sky background it is recommended a minimum of three, but preferably more than five, PONG maps are included in a single observation with a rotation introduced between each one. In this way a circular pattern is built up, (see the lower right-hand panel of Figure 2.3), with a diameter equal to your requested map size.

To recover large-scale extended structure you are advised to use larger PONG maps which scan at a higher rate. This option is preferable to tiling multiple smaller maps. Ultimately it is the size of the SCUBA-2 field-of-view that determines the sensitivity to large-scale structure.

DAISY DAISY maps are the option for point-like or compact sources (<3 arcmin) by maximising the exposure time on the centre of the image. The telescope moves at a constant velocity in a 'spirograph' pattern that has the advantage of keeping the source on the array throughout the observation. This is shown in the top panel of Figure 2.3. While the central <3 arcmin has a uniform background noise, DAISY maps cover a circular area of diameter of 12 arcmin.

Should I use a DAISY or a 15-arcmin PONG? A common issue is that a pong900 is used when possibly a DAISY would have been better, given that the latter is much faster and employs a significant exposure time out to a diameter of 12 arcmin. The numbers break down as follows:

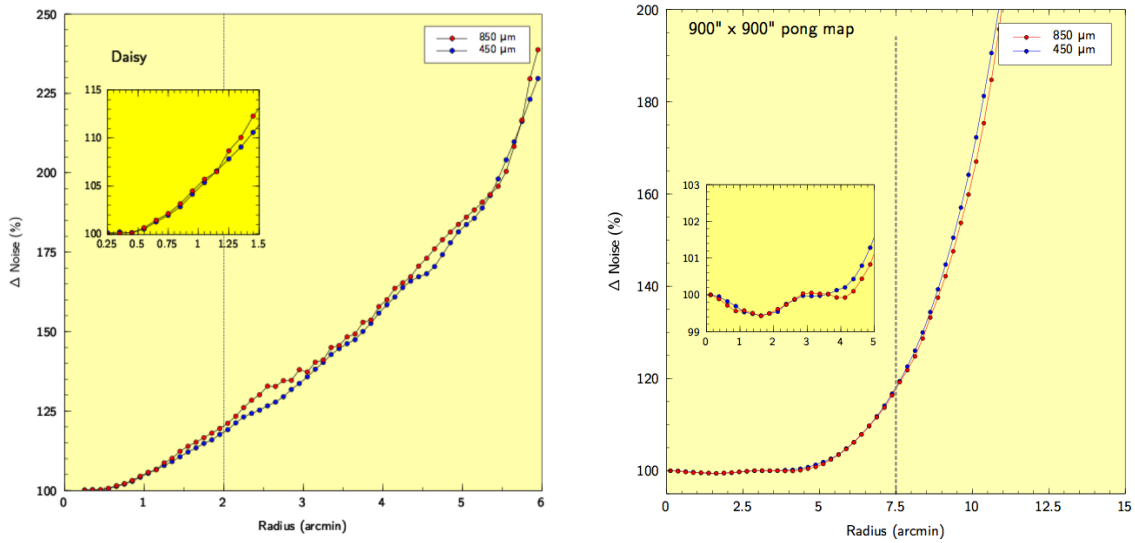


Figure 2.2: The radial noise profiles for a DAISY and PONG900 map. For the same integration time, the rms in the center (<3 arcmin) of a DAISY will be more than twice as good as in a PONG900. Out to a radius of ∼5.5 arcmin, the noise will still be below the PONG900 target noise.

For the same integration time, the rms in the center (<3 arcmin) of a Daisy will be more than twice as good as in a pong900. Out to a radius of ∼5.5 arcmin, the noise will still be below the pong900 target noise. Beyond this radius the noise will exceed the target noise, but that is also the case for the pong900 (see the radial profiles in Figure 2.2).

I.e. the trade-off is between a flatter and slightly larger map (pong900) and a somewhat smaller but much deeper map in the center with a distinct noise gradient across the field (DAISY).

Detection experiments may well be better off with Daisies, although statistical conclusions, such as number counts, may become more complicated. The same may be true for isolated (i.e. non-mosaicked) fields where one could ask if the negative impact of the noise gradient and a smaller field out-weigh the benefits of a deeper mapping across most of the image.

There are other possibilities, such as doing an initial exploratory DAISY to the required depth in a 3 arcmin field (this can be done in less that 25% of the time it takes for a PONG900) and then proceed with pongs on the most promising candidate(s). PONG and DAISY fields can be combined. It may also be beneficial to use a pattern of offset DAISIES to mitigate somewhat for the more pronounced gradient or to better match the source morphology in the field.

Why these patterns?

SCUBA-2 removes atmospheric noise in the data-processing stage (Holland et al. 2013) [12]. The power spectrum of data taken by SCUBA-2 has a $1/f$ noise curve at lower frequencies. To ensure astronomical signals are far away from this $1/f$ noise, fast scanning speeds are required.

In order to disentangle persistent source structure from other slowly varying signals (e.g. extinction, sky noise, $1/f$ noise), the scan pattern must pass across each region of the map from different directions (hour angles) and at different times. The scan patterns themselves, along with the associated parameters (velocity and scan-spacing), have been designed and optimised to meet both these criteria. DAISIES, PONG900, PONG1800, and PONG3600 have telescope velocities of 155″/s, 280″/s, 400″/s, and 600″/s, respectively.

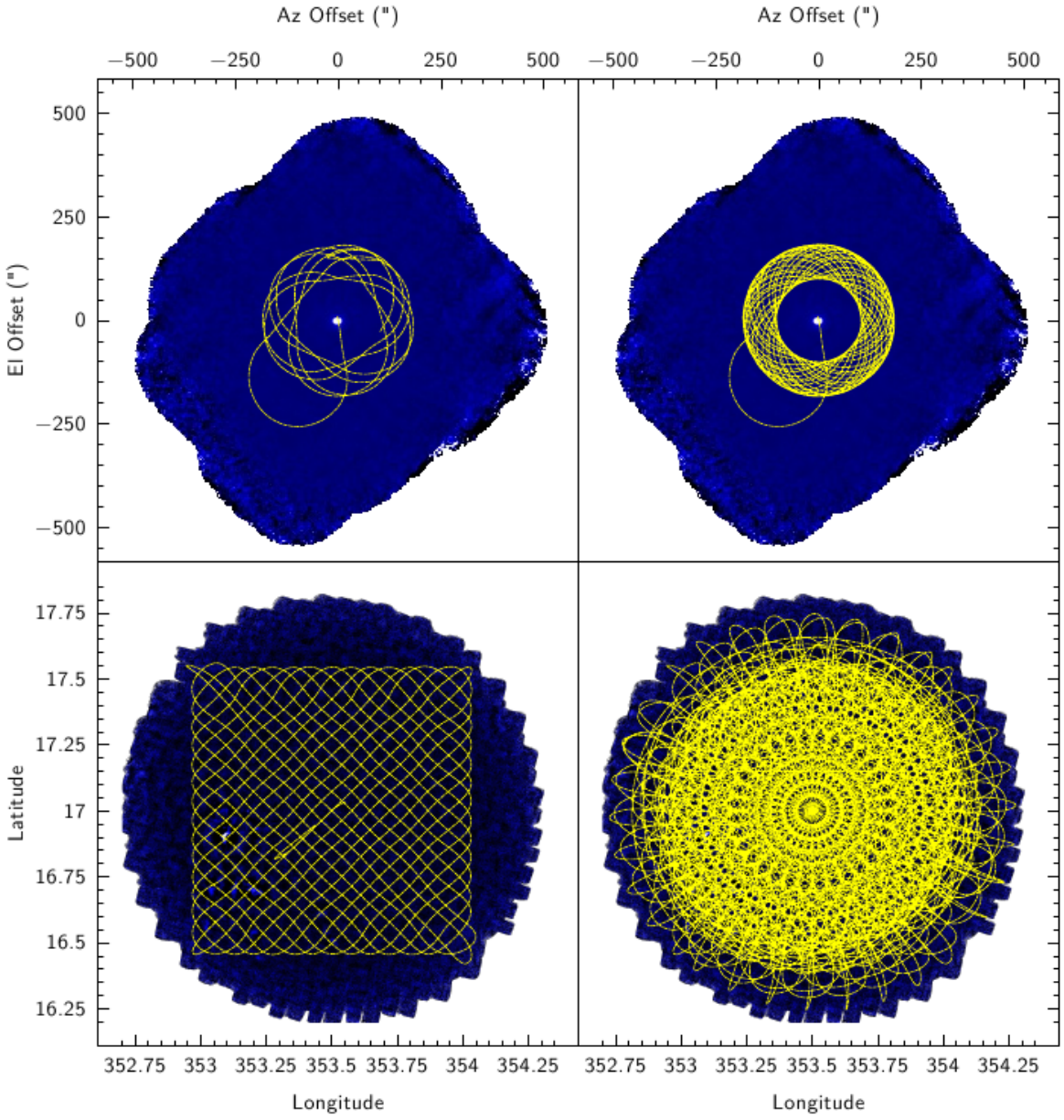


Figure 2.3:
The top row shows a DAISY and the bottom row shows a PONG. The left column shows the telescope track over a single rotation of the pattern. The right column shows the telescope track after multiple rotations of the pattern. The scan pattern for an observation can be visualised in this way with TOPCAT using the output from jcmtstate2cat. See section 9.3 for more details. Figure modified from Holland et al. (2013).

2.3 The raw data

A normal science observation will follow the following sequence.

(1) Flat-field

(2) Science scans
(3) Flat-field

The SEQ_TYPE keyword in the FITS header may be used to identify the nature of each scan (see Section 9.2). When you access raw from the Science Archive you will get all of the files listed above. Later when you reduce your data using the map-maker you must include all the science files *and* the first flat-field. The final flat-field is not currently used.

Shown below is a list of the raw files for a single sub-array (in this case s8a) for a short calibration observation. The first and last scans are the flat-field observations,which occur after the shutter opens to the sky at the start of the observation and closes at the end (note the identical file size); all of the scans in between are science.

```
% ls -lh /jcmtdata/raw/scuba2/s8a/20131227/00034


-rw-r--r-- 1 jcmtarch jcmt 8.0M Dec 27 03:00 s8a20131227_00034_0001.sdf
-rw-r--r-- 1 jcmtarch jcmt  22M Dec 27 03:00 s8a20131227_00034_0002.sdf
-rw-r--r-- 1 jcmtarch jcmt  22M Dec 27 03:01 s8a20131227_00034_0003.sdf
-rw-r--r-- 1 jcmtarch jcmt  22M Dec 27 03:02 s8a20131227_00034_0004.sdf
-rw-r--r-- 1 jcmtarch jcmt  22M Dec 27 03:02 s8a20131227_00034_0005.sdf
-rw-r--r-- 1 jcmtarch jcmt 6.8M Dec 27 03:02 s8a20131227_00034_0006.sdf
-rw-r--r-- 1 jcmtarch jcmt 8.0M Dec 27 03:03 s8a20131227_00034_0007.sdf
```

The SCUBA-2 data-acquisition (DA) system writes out a data file every 30 seconds; each of which contains 22 MB of data. The only exception is the final science scan which will usually be smaller (6.8 MB in the example above), typically requiring less than 30 seconds of data to complete the observation.

Note: All of these files are written out eight times, once for each of the eight sub-arrays.

The main data arrays of each file are cubes, with the first two dimensions enumerating bolometer columns and rows within a sub-array, and the third time slices (sampled at roughly 200 Hz).

A standardised file naming scheme is used in which each file name starts with the sub-array name, followed by the UT date of the observation in the format yyyymmdd, followed by a five-digit observation number, followed by the sub-scan number. The name ends with the standard suffix .sdf used by all Starlink data files. For instance, the files listed above hold data from the s8a sub-array for observation 34 taken on 27th December 2013.

Units

Raw SCUBA-2 data come in uncalibrated units. The first calibration step is to scale the raw data to units of picowatts (pW) by applying the flat-field solution. This step is performed internally by the map-maker but can be done manually when examining the raw data—see Section 9.1.

The second step is to scale the resulting map by the flux conversion factor (FCF) to give units of janskys. When running the ORAC-DR pipeline this is done automatically.

Chapter 3
The Dynamic Iterative Map-Maker Explained

The Dynamic Iterative Map-Maker, hereafter just referred to as the map-maker is the tool you will use to produce SCUBA-2 maps, and is implemented by the SMURF makemap command. It performs all pre-processing steps to clean the data, followed by solving for multiple signal components using an iterative algorithm, and binning the resulting time-series data to produce a final science map.

The makemap command can be invoked in two ways: 1) directly by typing "`makemap`" in response to a unix shell prompt (see Section 5), or 2) indirectly as part of an ORAC-DR recipe (see Chapter 4).

This chapter describes how the map-maker produces a science image from raw SCUBA-2 data. It should be considered essential reading as it provides an understanding of how your reduced image was produced. This is particularly true if you wish to modify the default map-maker parameters. If you prefer to jump straight in to the data reduction go to Chapter 4.

3.1 How it works

The time-varying signal recorded by a bolometer, $b(t)$, contains contributions from a variety of sources:

$$b(t) = e(t) \times a(t) + n^{\mathrm{w}}(t) + g n^{\mathrm{c}}(t) + n^{\mathrm{f}}(t), \tag{3.1}$$

where $e(t)$ is the atmospheric extinction, $a(t)$ is the astronomical signal (time varying as the telescope scans across the sky), $n^{\mathrm{w}}(t)$ is the uncorrelated white noise, g is an optional scalefactor which is unique to each bolometer, $n^{\mathrm{c}}(t)$ is the correlated (or, common mode) signal, and $n^{\mathrm{f}}(t)$ is the (predominantly low-frequency) noise which does not have a simple correlation relationship that would be included in the $n^{\mathrm{c}}(t)$ noise term.

The map-maker works by producing individual models of the various components that make up the signal recorded by each bolometer. It models and removes the extinction and noise components in order of decreasing magnitude, ultimately leaving just the astronomical signal plus residual noise. The modelled components are listed in Table 3.1 and described more fully in Section 3.3.

A *configuration file* should usually be supplied when running makemap directly. This file holds a set of parameter values that control all aspects of the map-maker, including details of the pre-processing steps, which model components to include, parameters that control the determination of each model, and the stopping (or convergence) criteria. If no configuration file is supplied when running makemap[1], a set of default parameter values will be used. There are a great number of these parameters, but fortunately not all of them are of interest to the typical user. Appendix I documents the parameters you are more likely to find of interest, including the default value that is assigned to each parameter if you do not give it a value in your configuration file. For a full list of all available parameters, see the appendix "Configuration Parameters" within SUN/258.

It is possible to create a map without supplying a configuration file to makemap (*i.e.* leaving all configuration parameters set to their default values—"`config=def`"—when running `makemap`) but it is not recommeneded since it will not give optimal results for your particular observation. For this reason, specialised configuration files have been developed which are tailored to different science goals, be

[1] *i.e.* you include "`config=def`" on the makemap command line.

they detecting faint galaxies or mapping large molecular clouds. A description of these specialised configuration files can be found in Section 3.7.

Note: When using the pipeline to create a map, rather than running makemap directly, the pipeline will always use a configuration file—one of the standard configuration files will be selected if none is specified by you.

3.2 The reduction step-by-step

This section describes the basic map-making process used by the default configuration. It may be modified in many ways by supplying a configuration file containing alternative parameter values.

Figure 3.1 shows the flow chart of the basic map-making process. It is divided into two sections: the pre-processing stage where the data are cleaned, then the iterative stage where the different models are subtracted, a sky map created and the convergence checked.

1 Initial cleaning and down-sampling
The separate raw data files are first concatenated into a single time series for each sub-array (if possible—see the description of *Data Chunking* on Page 24) and have the flat-field from the associated fast-flat scans applied to calibrate the bolometers. This resulting time-series are in units of pW.

These time-series are then re-sampled at a rate that matches the requested pixel size—the equivalent to applying a low-pass filter. Down-sampling saves time and memory usage when running the map-maker, without losing any significant information. The degree of down-sampling applied depends on how fast the telescope was moving during the observation, the requested pixel size. In general, slower scans will be more heavily down-sampled than faster scans, and fast scans may not be down-sampled at all.

A number of cleaning steps are then run: bolometers that have unusually high noise levels are identified and excluded from further processing, any sudden steps in the base-line of each bolometer are identified and corrected, and a polynomial estimate of the base-line is removed from each bolometer.

2 Iterative steps
Next comes the iterative stage. In each iteration, estimates are produced for each model component. These are removed from the cleaned time-series data and the remaining data values are binned into a map. However, in general, the presence of astronomical signal within the original time series will have upset the estimates of the COM, GAI and FLT models, causing the final map to be inaccurate. But now that we have a map (albeit an inaccurate map), we can sample the map at the position of each bolometer value to get an estimate of the astronomical component within the original time-series data. We then remove this astronomical signal from the original time-series data and re-estimate the models. These new model estimates should be more accurate since they are not so heavily influenced by the astronomical signal. In turn, this allows us to create a more accurate map. We repeat this process until the map does not change significantly with further iterations. Whilst not mathematically rigorous, we expect the process to converge because the astronomical signal is in general much smaller than any of the other components and so each iteration introduces a very small fractional change in the model estimates.

Within each iteration, the first components to be modelled and removed are COM and GAI, which work together to calculate and remove the average signal template of all bolometers, allowing each bolometer to have an arbitrary gain and offset. In addition, the GAI model will flag as unusable any bolometer in which the signal looks very different to the common-mode.

The next model (EXT) applies a multiplicative extinction correction to the data. Following this, the FLT model filters each bolometer time-stream independently to remove low frequencies that correspond to angular scales larger than 600 arcsec and 300 arcsec at 450μm and 850μm, respectively.

Model	Description
COM	Common-mode signal
GAI	Gains that scale each bolometer to the common-mode
EXT	Extinction correction
FLT	Filter that removes low frequencies
AST	Astronomical signal
NOI	Residual noise

Table 3.1: Table adopted from Chapin et al. (2013). A detailed explanation of each model is given in Section 3.3.

After these models have been removed, the AST model (*i.e.* the estimate of the astronomical signal) produced by the previous iteration, is added back onto the remaining time-series data[2], and the new values are binned up on the sky to produce a new estimate of the final science map. Since many samples typically contribute to the estimate of the signal in a given pixel, the noise is greatly reduced compared with the time-series data. The variance of each map pixel value is determined from the spread of sample values that contribute to the pixel. The value of this map at the position of each bolometer sample is then found. This forms the new AST model which is removed from the time-streams, leaving just the residual noise.

The NOI model then measures the noise in the residual signals for each bolometer to establish weights for the data as they are placed into the map in subsequent iterations. This is only done on the first iteration. Subsequent iterations re-use the weights established on the first iteration.

3 Checking convergence
Convergence is checked against the parameters detailed in the configuration file. Convergence is achieved either when the requested number of iterations has been completed, or when the mean change in the map pixel values is less than a specified fraction of a standard deviation (see parameter maptol). If more iterations are required or (in the latter case) the map is still changing significantly, all the model—*except for AST*—are added back onto the residuals, thus reconstructing the original time-series but without the astronomical signal. This is the signal upon which the model estimates will be based on the next iteration.

For full details of the map-maker see **Chapin et al. (2013)** [5].

3.3 The individual models

The list of models to be evaluated (and removed) by the map-maker, and the order in which they are used during the iterative stage, is given by the modelorder parameter in the configuration file. These models are modular however, so their order may be changed. The default model order is given below[3].

```
modelorder = (com,gai,ext,flt,ast,noi)
```

[2]This step is omitted on the first iteration since no estimate of AST has yet been created.
[3]If using STARLINK's PCA background removal, one can add "pca" to the list: (com,gai,pca,ext,flt,ast,noi). For more information, see https://www.eaobservatory.org/jcmt/2019/03/faster-pca/

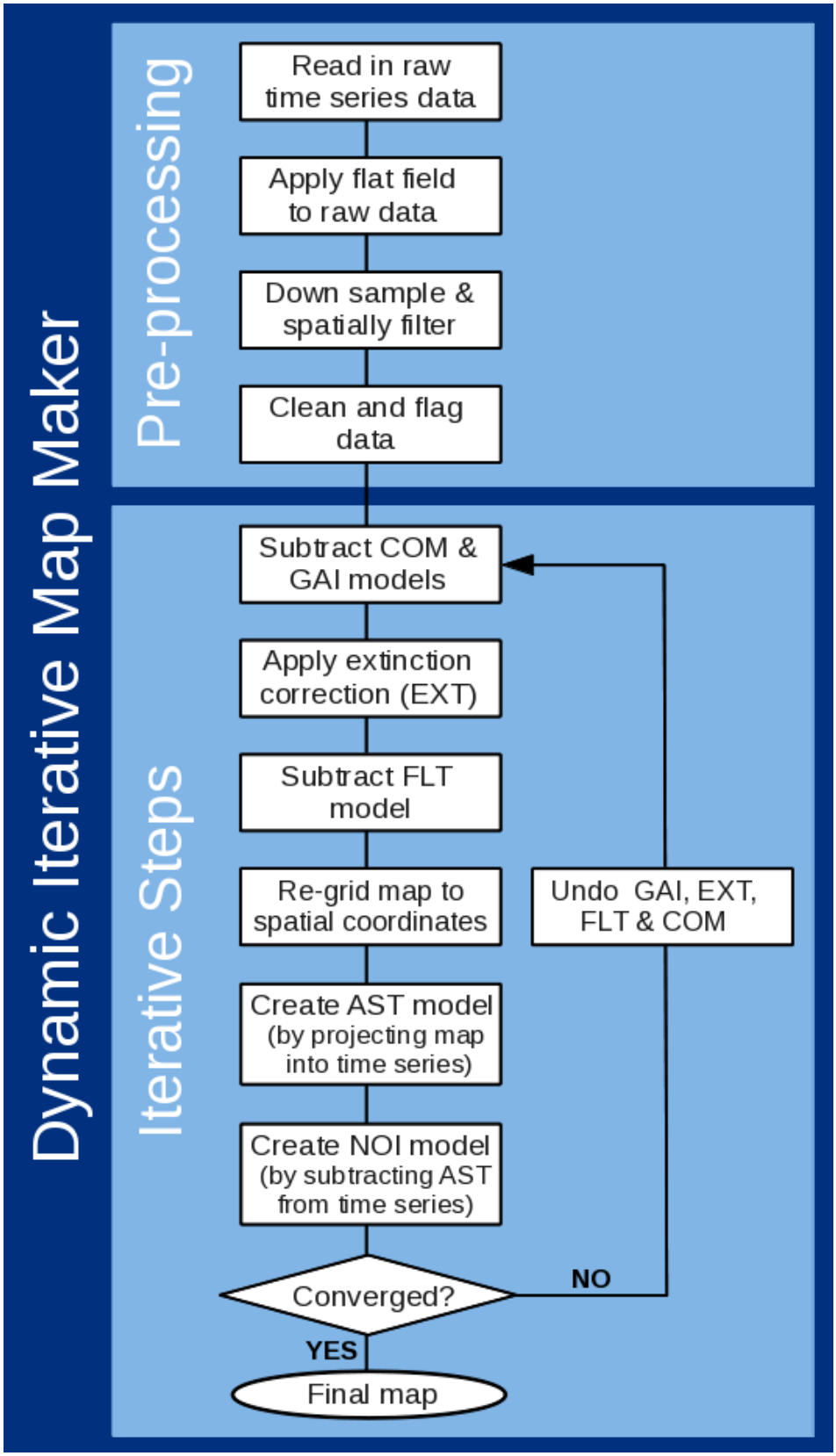


Figure 3.1: A flow chart illustrating the dynamic iterative map-maker. Note that for each iteration the `AST` model is subtracted from the time-series leaving only those contributions to be fitted and removed.

The only configuration file not following this model order is the one tailored for blank field maps which does not include a FLT model in the iterative stage but instead includes an equivalent high-pass filter as part of the cleaning stage (see Section 3.7).

Below is an introduction to each model. More complete descriptions of the models and all the associated caveats can be found in Chapin et al. (2013) [5]. Details of each model are controlled by a set of configuration parameters that begin with the name of the model. For instance, all parameters related to the COM model will be of the form "COM.xxx".

MODEL	DESCRIPTION
COM	The COM model removes the common-mode signal (the signal that is common to all bolometers), the dominant contributor to this signal being the sky noise. It determines this by simply averaging over all bolometers for each time slice. Bolometers are flagged as bad, (and thus omitted from the final map), if they do not resemble the COM model seen by the majority of the other bolometers. Extended emission on a scale larger than the array footprint on the sky will contribute a signal indistinguishable from a common-mode signal. This puts an upper limit on the spatial scale of astronomical emission that can be recovered[4].
GAI	GAI model works with COM in removing the common-mode signal. GAI consists of a time-varying scale and offset for each bolometer, which scales the COM model so that it resembles the original bolometer data as closely as possible. It is the scaled version of COM that is removed from the bolometer time-series. In addition, the GAI model identifies any bolometers that looks very different to the common-mode signal, and flags them as unusable.
EXT	EXT applies the extinction correction. This is a time-varying scaling factor that is derived from the JCMT water-vapour radiometer. As it deals with 30-second chunks of data, this accounts for varying conditions over a long observation. For more details see Appendix B.
FLT	The FLT model acts on the Fourier transform of the bolometer data. High-pass (only allows *higher* frequencies to pass) and low-pass (only allows *lower* frequencies to pass) filters can be specified. These cut-off frequencies can either be specified directly in Hz, or as an angular spatial scale in arcseconds that is subsequently converted into a frequency using the speed at which the telescope is moving. The most important role of this model is to apply a high-pass filter in order to remove the low frequency $1/f$ noise. Note that extended emission varies slowly over the array; it therefore appears at low frequencies and complicates the choice of a high-pass filter. A further discussion of this matter is given in Section 7.2.
AST	The AST model is generated in conjunction with a science map. Hence, the position of AST in the model order indicates at what stage the astronomical image should be estimated. When the AST model is calculated, the first step is bin the residual time-series data into a map (using nearest-neighbour sampling). Following this, the map is projected back into the time domain (*i.e.* sampled at the position of every bolometer sample) and removed as the AST model.

[4] "Fake Sources" of known properties can be inserted into the raw SCUBA-2 time stream in order to evaluate the extent of robust emission recovery. For more information, see Mairs et al. 2015 ([15])

MODEL	DESCRIPTION
NOI	`NOI` should come last in the model order and calculates the RMS noise level in each bolometer. It is only calculated on the first iteration. The same noise levels are then used on all subsequent iterations to weight the bolometer values when binning them into a map. Each bolometer may have a single noise estimate that remains fixed over the whole observation, or may have multiple noise levels, each of which relate to a different part of the observation. See parameters `noi.box_size` and `noi.box_type`.

Examples of the time traces for a single bolometer from these models is shown in Figure 3.3. These traces cover a subset of an observation of the secondary calibrator CRL 2688. You can clearly see the dominance of the `COM` model which is removed first. The `FLT` model stores the data removed by the high-pass filter. In the `AST` model, CRL 2688 is clearly seen as positive spikes which appear when the bolometer passes over the source. Finally, the residual signal stored in `RES` is flat, indicating that most of the signal has been successfully accounted for by the other model components.

3.4 Stopping criteria

The map-maker will stop processing either when the requested number of iterations has been completed **OR** when the convergence criterion specified in the configuration file is reached.

Option 1: Fixed number of iterations

Specifying the number of iterations in the configuration file is done via the `numiter` parameter. This is shown below for `dimmconfig_jsa_generic.lis` (JSA stands for "JCMT Science Archive").

```
numiter = -25
```

A positive value for `numiter` means that the requested number of iterations will be executed. A negative value, as in the example above, indicates that no more than this number of iterations should be performed, *but* that it may stop at fewer if convergence (according to the noise criterion below) has been achieved.

Option 2: Convergence parameter

The convergence criterion is set by the `maptol` parameter.

When running the map-maker, `maptol` gives the average normalised change in the value of map pixels between subsequent iterations. Convergence is reached when the mean change across all pixels is less than the parameter `maptol`. It has units of the noise in the map, thus `maptol`=0.05 means a mean change in pixel value of $<0.05\,\sigma$. This option has the advantage of directly assessing the noise in the resulting map.

The map-maker displays the normalised change in pixel values between iterations at the end of each iteration. This value typically drops rapidly to begin with and then flattens out, decreasing increasingly slowly. The printed values can be inspected to check convergence is proceeding as expected. Be aware that setting `maptol` to much lower than 0.05 will dramatically increase the length of time needed to produce the final map, and may possibly result in convergence never being achieved.

If you wish to perform further checks on the progress of convergence while running the map-maker, you can use the `itermap` option. This causes the map created by each iteration to be dumped to disk for inspection (see Section 6).

3.5 Masking

The term *masking* refers to the inclusion or exclusion of selected bolometer samples from the estimate of specific models, based on the positions of those samples within the output map. Masking can be applied independently to the `AST`, `FLT` and `COM` models, as described in the later sub-sections within this section.

A *mask* is a selection of pixels within the output map. Normally these pixels form one or more contiguous regions that enclose the areas where significant sub-mm emission is expected (the "source" regions). Pixels outside this mask are referred to as "background pixels". The use to which such masks are put differs from model to model as described later in model-specific sub-sections.

Masks fall into two broad categories:

- **dynamic masks** – these are masks that change shape from one iteration to the next.
- **static masks** – these are masks that retain the same shape for all iterations.

Static masks are appropriate in cases where the areas containing significant sub-mm emission are known in advance. If this is not the case, then the mask needs to be based on evidence within the maps created at the end of each iteration, resulting in the masks evolving from iteration to iteration as the maps converge towards the final solution. Such *dynamic* masks have a potential problem in that they can upset the convergence process by introducing extra flexibility into the algorithm. For this reason, an option is provided to freeze these masks after a fixed number of iterations[5] (*e.g.* see `ast.zero_freeze`) and this is used within the specialist configuration file `dimmconfig_fix_convergence` described in Section 3.8.

Each of the three models—`AST`, `FLT` and `COM` —have an equivalent set of parameters to specify the masking (if any) that should be used. Each set starts with the model name, followed by a dot, followed by the parameter name. To enable masking, one or more of these parameters needs to be set to a suitable value to define a mask, as described below.

Static masks can be defined either as circles of specified radius, centred on the source position (suitable for compact sources—see for instance `ast.zero_circle`), or by an external image that uses special pixel values to indicate the source regions and the background regions (suitable for extended sources—see for instance `ast.zero_mask`). Within such "external masks" all background pixels should be set to the Starlink *bad* value (see "Bad-pixel Masking" in SUN/95). Pixels with any other value are treated as source pixels. Section 6.6 describes some ways in which external masks can be generated.

Dynamic masks can be defined in several different ways:

(1) By applying a simple threshold on the S/N ratio within the map created at the end of each iteration (*e.g.* see `ast.zero_snr`).

(2) By applying a simple threshold on the S/N ratio as above, and then extending the resulting source regions down to a lower S/N value (*e.g.* see `ast.zero_snrlo`). This allows a low S/N threshold to be used (which can sometimes help to avoid negative bowling around the edges of extended sources) without introducing lots of isolated pixels into the mask because of noise spikes in the map.

(3) By applying a threshold on the number of bolometer samples falling in each map pixel (*e.g.* see `ast.zero_lowhits`). Because of the scanning strategy used with SCUBA-2 (see Section 2.2), pixels near the edge of the map will receive fewer bolometer samples than those near the centre, and will therefore be noisier. Thus this scheme allows the noisy edges of the map to be masked out. Whilst this is technically a dynamic mask—because the number of samples falling in each pixel could potentially change from iteration to iteration—in fact a mask based on hits per pixel will usually only change very slightly from iteration to iteration.

[5] Although by default masks are never frozen.

(4) By using an algorithm that automatically identifies features of a given size in the map, similar to that used by the KAPPA ffclean command (*e.g.* see `ast.zero_snr_ffclean`). This can be a useful alternative to simple S/N thresholding in cases where bright artificial large-scale structures appear in the map. Such structures will result in artificially large source regions in a simple S/N mask, which may in turn cause the artificial structures to become even brighter and larger in subsequent iterations. Using an ffclean-based mask instead will restrict the mask to the brighter astronomical cores on top of such artificial large-scale structures.

Multiple masks can be used by selecting the appropriate configuration parameters as described above. Whether the combined mask represents the *union* or *intersection* of the individual masks is selected using the parameters `ast.zero_union`, `flt.zero_union` and `com.zero_union`. By default, the union is used.

The masks used on the last iteration are stored in the `Quality` component of the NDF containing the output map. A separate bit of the quality component is used for each model (`AST`, `FLT` or `COM`) that is being masked. These bits can be viewed in various ways—see Section 8.6 and Section 5.3.2.

3.5.1 AST masking

The iterative map-maker algorithm described earlier in this chapter divides each cleaned bolometer value into four additive parts:

(1) The common-mode signal (`COM`).

(2) The low frequency background drift in each bolometer (`FLT`).

(3) The astronomical signal (`AST`).

(4) The left-over residual noise (`RES`).

Whilst the value of each model is constrained to some extent by the manner in which the model is calculated, there remains a large degree of degeneracy in the algorithm. In other words, there are many different ways in which each bolometer value can be divided up into the four values listed above. For instance, any arbitrary change in the `AST` model can be accommodated provided that equal and opposite changes are introduced into the other model values so that the total value of all four components remains the same.

The degeneracy between `COM` and `AST` is particularly problematic, since it allows large scale features to appear in the map (on the scale of the sub-array footprint) that are corrected for by equal and opposite features in the `COM` model. Such features can appear as gradients or "saddles" across the final map.

The purpose of masking the `AST` model is to apply an extra constraint to the system to help break this degeneracy, and so avoid these large-scale features. It functions by forcing the `AST` model to zero at the end of each iteration for all samples that fall outside the masked area on the sky (*i.e.* samples that fall in the "background" areas of the map). Forcing the `AST` model to zero in this way means that the `COM` model is more closely constrained.

Whilst this constraint is applied only in the background regions, its effects are also felt part way into the source region because of the way in which the `COM` model averages over a sub-array footprint. Thus source regions that are much larger than a sub-array will still suffer from the degeneracy between `COM` and `AST`, allowing arbitrary large-scale background features to develop within the region, but these features will become weaker as the edges of the mask are approached.

In order to avoid the final map containing zero values in the background region, it is necessary to avoid using `AST`-masking on the final iteration. For this reason, by default one further iteration is performed—without `AST`-masking—once the basic iterative algorithm has converged (see `ast.zero_notlast`).

All the configuration files supplied with SMURF use `AST` masking, with the exception of `dimmconfig_blank_field`.

3.5.2 FLT masking

The FLT model estimates and removes a low frequency background from each bolometer time-stream. To do this it uses an FFT-based high-pass filter. Ringing around sudden transients—such as astronomical sources — is a natural consequence of all such filters, resulting in alternating bright and dark rings around sources in the map. In most cases, the iterative nature of the map-making algorithm eventually results in the rings reducing to a much lower level, as more of the astronomical signal is removed from the time stream on each iteration, causing the ring-inducing transients to become weaker. However, this is not always completely effective, and also slows down convergence.

For these reasons, it is often helpful to mask the FLT model. This causes each bolometer time stream to be blanked out as the bolometer passes over the source regions in the mask. Such blanked sections are replaced by linear interpolation between the data on either side of the blanked section. The effect of this is to remove the transients that cause the ringing in the FLT model, and thus get a better estimate of the low frequency background in each bolometer.

Such masking is most useful at the start of the iterative process, before the bulk of the astronomical signal has been removed from the time-streams. Indeed, if it is allowed to continue for too many iterations it can cause its own problems that tend to slow down convergence. For this reason, if FLT masking is used, it is (by default) limited to two iterations—see `flt.zero_niter`.

A difficulty with FLT masking is that the FLT model is evaluated *before* the map is made on each iteration. This means that when the FLT model is evaluated on the first iteration, there is no map available. This is not a problem if a static mask is used, but if a dynamic mask is used, then it is not possible to calculate the FLT mask on the first iteration (subsequent iterations will create the mask on the basis of the map created at the end of the previous iteration). This is a shame since FLT masking is most beneficial on the first few iterations, whilst the bulk of the astronomical signal is still present in the time streams. A scheme to get around this problem is described in Section 3.6.

All the configuration files supplied with SMURF use FLT masking, with the exception of `dimmconfig_blank_field`.

3.5.3 COM masking

The value of the COM model at a specific time-slice is the mean of all remaining bolometer values at that time slice. If the COM model is masked, bolometer samples that fall within the source regions defined by the mask are excluded from this mean, thus removing the bias introduced into the mean by bright astronomical sources. This results in the COM model more accurately representing the varying common-mode background value at each time slice. This can help to reduce negative bowling around sources in the map, and can help to reduce the amount of data that is rejected as being too poorly correlated with the common-mode signal.

COM masking has the same difficulty as FLT masking in that the COM model is evaluated *before* the map is made on each iteration, and so it is difficult to use dynamic masks with COM masking. However, the solution described in Section 3.6 can be used for COM masking as for FLT masking.

Currently, none of the configuration files supplied with SMURF use COM masking.

3.6 Skipping the AST model

As discussed in Section 3.5, FLT masking is most effective on the early iterations (particularly the first iteration). This is because the bulk of the astronomical signal is still present in the time streams and is likely to cause bad ringing if FLT masking is not used. However, since the FLT model is evaluated before the map is made within each iteration, there is no map available to use as the basis for a FLT mask on the very first iteration.

One way to get round this is simply to use a static mask that does not require a map to be available. However, this may not be possible, particularly for extended sources where no *a-priori* knowledge of the source areas in the sub-mm may be available.

Another scheme is provided by the `ast.skip` configuration parameter. If `ast.skip` is set to a positive integer N, the subtraction of the `AST` model from the time-series that normally occurs at the end of each iteration is not performed on the first N iterations. This means that for the first N iterations, the time-series data remain unchanged, and consequently the models and the resulting map are unchanged. Of itself, this does nothing useful—it merely delays the first useful iteration. However, when combined with `FLT` masking, it can be very beneficial because it allows a mask to be created *before* the first "real" iteration occurs (*i.e.* the first iteration that subtracts off the `AST` model). This means that the first "real" map will have far less ringing around bright sources than would otherwise be the case. This in turn means that far fewer iterations are required to correct this ringing.

It is also possible to use `ast.skip` with `COM` masking, but the benefits are not so noticeable as for `FLT` masking.

The default value for `ast.skip` is zero, but the `dimmconfig_bright_extended` and `dimmconfig_jsa_generic` configuration files supplied with SMURF both set `ast.skip = 5` (and also use `FLT` masking).

3.7 Specialised configuration files

Maps made by the pipeline using the REDUCE_SCAN recipe will use `dimmconfig_jsa_generic.lis` unless some alternative configuration has been specified. This configuration file is intended to provide a reasonably good map for all types of observations. However, compromises have been made to reach that balance, the main one being that some real extended structure is sacrificed in order to avoid introducing artificial large scale structures.

Whilst `dimmconfig_jsa_generic.lis` is always a good place to start, you will want to follow this up with a specialised configuration that will suit your observation. A few specialised configuration files are supplied with SMURF and can be found in `$STARLINK_DIR/share/smurf/`.

Note, any parameter for which no value is specified within the supplied configuration will take on the default value listed in Appendix I.

Below is a description of each of the specialised configuration files. The tables following each description list the parameters set by each file. The files themselves are stored in `$STARLINK_DIR/share/smurf/`—they are simple text files and so can be inspected easily.

3.7.1 dimmconfig_jsa_generic.lis

The `dimmconfig_jsa_generic` configuration file is designed to handle all science data. This is a good configuration file to start with if you are unsure how to proceed with your data reduction. You may notice that elements of this configuration are common to the other specialised configuration files.

This configuration is optimised to suppress the creation of artificial large scale structure within the final map. Consequently, some real extended emission may be lost if this configuration is used. This goal is achieved by:

(1) Setting `flt.filt_edge_largescale`) to 200 arcseconds so that the `FLT` model uses a heavier than normal filter.

(2) Setting `com.perarray` to 1, indicating that four separate `COM` models should be used, one for each sub-array[6].

[6]By default, a single `COM` model is used that represents the mean data value across all four sub-arrays.

(3) Using a mask to force the AST model to zero in the background regions after each iteration (see parameters ast.zero_snr and ast.zero_snrlo—also see Section 3.5.1).

In addition the S/N threshold for DC steps (dcthresh) is relaxed from 25 in the default file to 100 to avoid problems bright sources triggering the step detection algorithm.

The other major feature of this configuration is that the subtraction of the AST model is skipped on the first five iterations. In conjunction with the use of FLT masking, this helps to speed up convergence and minimise dark rings around bright source (see Section 3.6.

dimmconfig_jsa_generic.lis	
ast.skip = 5	ast.zero_snrlo = 3
ast.zero_snr = 5	com.perarray = 1
dcthresh = 100	flt.filt_edge_largescale = 200
flt.zero_snrlo = 3	flt.zero_snr = 5
maptol = 0.01	numiter = -25
noisecliphigh = 10.0	

3.7.2 dimmconfig_blank_field.lis

The dimmconfig_blank_field configuration is tuned for blank field surveys for which the goal is to detect extremely low signal-to-noise point sources within otherwise blank fields.

Applying a high-pass filter (FLT) on each iteration can result in convergence problems when there is little or no signal in the map. Instead, a single, harsher high-pass filter is applied as a pre-processing step (corresponding to 200-arcsec scales at both 450 μm and 850 μm). There are also more conservative cuts to remove noisy/problematic bolometers. Only 4 (positive) iterations are requested as there is no signal to confuse to models.

The option com.perarray = 1 requires the COM model to be fit to each sub-array independently. This improves the overall fit but with the loss of any structure on scales larger than a single sub-array—not an issue for blank fields.

Figure 3.4 shows the sharp contrast in the output map between reducing data with the default configuration file and using dimmconfig_blank_field.lis.

Blank-field maps commonly have a matched filter applied as part of the post-processing to aid source detection (see Section 8.7), however this is not applied by the map-maker.

dimmconfig_blank_field.lis	
numiter = 4	flt_edge_largescale = 200
spikethresh = 10	modelorder = (com,ext,ast,noi)
com.perarray = 1	

3.7.3 dimmconfig_bright_compact.lis

The dimmconfig_bright_compact configuration should be used for producing maps of bright, isolated compact sources that are positioned at the centre of the map (*e.g.* calibrators).

The addition of the ast.zero_circle parameter is used to constrain the map to zero beyond a radius of 1 arc-min from the source centre (see Section 3.5.1). This strategy helps with map convergence significantly, and can provide good maps of bright sources, even in cases where scan patterns failed to complete in full.

com.perarray is set to 1 indicating that a COM model should be fit separately for each sub-array. This is not advised for extended sources as signal on scales larger than a single sub-array is lost, but is fine for a compact central source. Likewise, the filtering is tighter. The S/N threshold for DC steps (dcthresh) is relaxed from 25 in the default file to 100 to avoid problems associated with bright sources.

The values of the parameters controlling the COM model are modified to relax the test that compares each bolometer to the common mode. This is because the algorithm that compares each bolometer to the common-mode can be fooled by a bright compact source passing across individual bolometers. Likewise the test for DC steps within each bolometer baseline is also relaxed (parameter dcthresh).

dimmconfig_bright_compact.lis	
numiter = -40	flt.filt_edge_largescale = 200
com.perarray = 1	flt.zero_circle = (0.016666)
ast.zero_circle = (0.0166666666)	
noisecliphigh = 10.0	dcthresh = 100
com.corr_tol = 7	com.gain_tol = 7
com.gain_abstol = 5	

3.7.4 dimmconfig_bright_extended.lis

The dimmconfig_bright_extended configuration is for reducing maps that contain emission that is extended to some degree and contains some bright regions. Here, the combination of ast.zero_snr and ast.zero_snrlo is used to identify components of the AST model wherever the S/N is higher than 3σ and then to expand this basic mask down to an SNR equal to 2σ without introducing any new isolated source areas (see Section 3.5.1). numiter has been raised to -40, as more iterations are required to maximise the sensitivity to large dynamic signal ranges in the map.

The other major feature of this configuration is that the subtraction of the AST model is skipped on the first five iterations. In conjunction with the use of FLT masking, this helps to speed up convergence and minimise dark bowls around bright source (see Section 3.6.

Figure 3.5 shows a comparison between maps reduced with the default configuration file and using dimmconfig_bright_extended.lis; the most noticeable difference is the improvement in the bowling around strong sources.

dimmconfig_bright_extended.lis	
numiter = -40	flt.filt_edge_largescale = 480
ast.zero_snr = 3	ast.zero_snrlo = 2
ast.skip = 5	flt.zero_snr = 5
flt.zero_snrlo = 3	

3.7.5 dimmconfig_pca.lis

The newest dimmconfig file (2019) is called: dimmconfig_pca.lis. This parameter file can be combined with other dimmconfig files to tell makemap to include a Principal Component Analysis (PCA) model in its iterative algorithm (the PCA model removes noise signals that are correlated between multiple bolometer time-streams). For instance, to use a PCA model when creating a map of a bright extended source, you could run makemap as follows:

```
% more conf
^$STARLINK_DIR/share/smurf/dimmconfig_bright_extended.lis
^$STARLINK_DIR/share/smurf/dimmconfig_pca.lis
% makemap config=^conf
```

To process compact sources, change "bright_extended" above to "bright_compact".

Using a PCA model can help to reduce the spurious extended structures that often appear in SCUBA-2 maps (although this benefit is bought at the cost of a much extended run time). For instance, four 850-μm maps of DR 21 are presented in Figure 3.6. Its top row shows the maps made with the basic "bright extendedd" dimmconfig, the middle row shows the results of adding in the new PCA dimmconfig, and the bottom row shows mosaics of the top and middle rows (left and right, respectively) with a difference map in between.

3.8 Configuration files for solving specific problems

Two additional configuration files are available that can be used to supplement your configuration file in order to solve specific problems:

- `dimmconfig_fix_convergence` to help your map converge. It does this by preventing masks from changing, and by freezing the flags that identify aberrant bolometers. Both of these restrictions are imposed only after 10 iterations have been performed.
- `dimmconfig_fix_blobs` to remove large "blooms" or smooth blobs of spurious emission in your map. These can be caused by ringing induced by unidentified steps or spikes in the bolometer time-streams. The fix is to look for oscillations within the time-streams following the `FLT` model, and flag such time-slices as unusable. In addition, time slices at which the four sub-arrays see significantly different common-mode values are also flagged as unusable.

These files provide a few additional settings that should be added to some other configuration in order to achieve the desired fix. For instance, a configuration file containing the following two lines would indicate that values from both `dimmconfig_bright_extended.lis` and `dimmconfig_fix_blobs.lis` should be used:

```
^/star/share/smurf/dimmconfig_bright_extended.lis
^/star/share/smurf/dimmconfig_fix_blobs.lis
```

This reads a set of configuration parameter from the `dimmconfig_bright_extended.lis` file, and then assigns new values for just those parameters defined in `dimmconfig_fix_blobs.lis`. These values override the values specified in `dimmconfig_bright_extended.lis`.

The parameters in these files are discussed further Chapter 6.

Data Chunking

Data Chunking :

Chunking occurs when there is insufficient computer memory available for the map-maker to process all the time-series data together, or when there is a gap in the data (e.g. from a missing sub-scan). In these cases, the map-maker divides up the time-series data into a number of "chunks" and produces a map from each chunk independently, before re-combining all the maps at a later stage to create the final map. Ideally you want your data reduced in a single chunk, however this can be unfeasible for large maps.

The more data the map-maker processes at once, the better chance it has of determining the difference between sky signal and background noise.

Information about chunking can be obtained in several ways:

- The screen output generated by `makemap` will include one or more statements about chunking, such as:

  ```
  smf_iteratemap: Continuous chunk 1 / 1 =========
  ```

  The final number indicates the number of chunks being used (one in this case).

- The MEMLOW FITS header in the final map can be examined to determine if chunking occurred due to insufficient memory. For instance:

  ```
  % fitsval mymap memlow
  FALSE
  ```

  indicates that the map in file `mymap.sdf` did not suffer from chunking as a result of low memory.

- The NCONTIG FITS header in the final map can be examined to determine if chunking occurred due to the supplied input data being discontiguous. For instance:

  ```
  % fitsval mymap ncontig
  2
  ```

  indicates that the time-series data from which `mymap.sdf` was created was broken into two contiguous groups of samples. This indicates a potential problem with the data—the NCONTIG header should normally be one.

In DAISY mode chunking is less of a concern as the entire map area is covered many times in the space of a single observation.

For PONG maps chunking is a bigger concern, with the maximum number of chunks that can be tolerated dependent on the number of map rotations. For example, a 40-minute PONG map with eight rotations may get divided into three or four chunks. Although not ideal, this will mean that each point is still covered by two or three passes. Fewer passes than this however and the map-maker become less effective.

Be aware that focus observations are intentionally dis-contiguous. If you attempt to make a map from a focus observation, it will be processed in several chunks no matter how much memory your computer has.

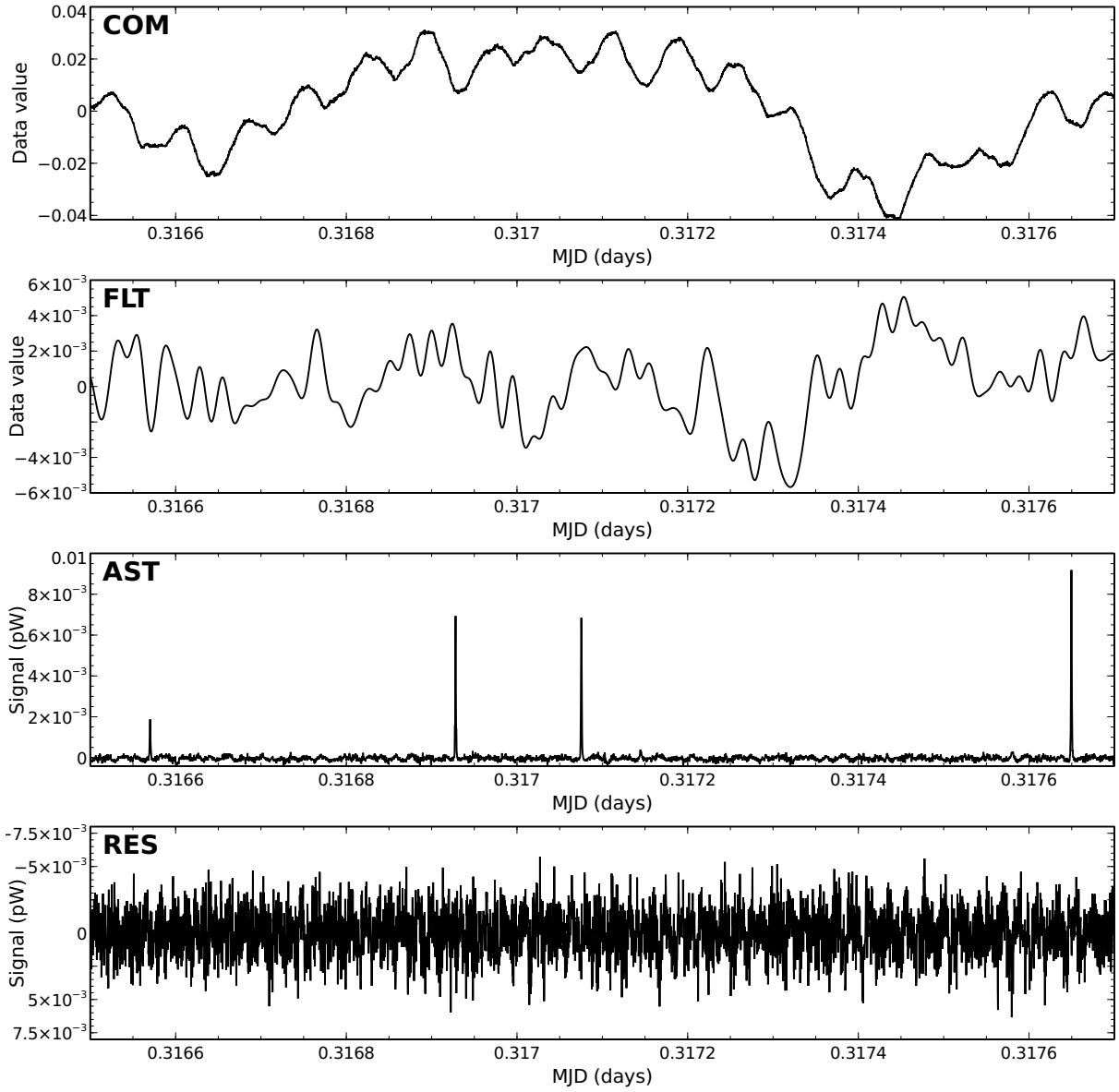


Figure 3.3: These plots show some of the models for a single bolometer, for part of an observation of CRL 2688. From top to bottom: the `COM` model containing signal common to all bolometers, the `FLT` model containing residual low-frequency noise missed by `COM`, the `AST` model with the astronomical signal showing as a positive spike when this bolometer passes over the source, and the `RES` model looking (as expected) like white noise.

Tip

Be aware that the S/N ratio within individual map pixels depends on pixel size—larger pixels will have higher S/N ratios than small pixels. Therefore masks based on a specific S/N threshold will grow or shrink in area as a function of pixel size. As a rough rule, if you double the pixel size, you should also double the S/N thresholds to get a mask that covers a similar area of the sky.

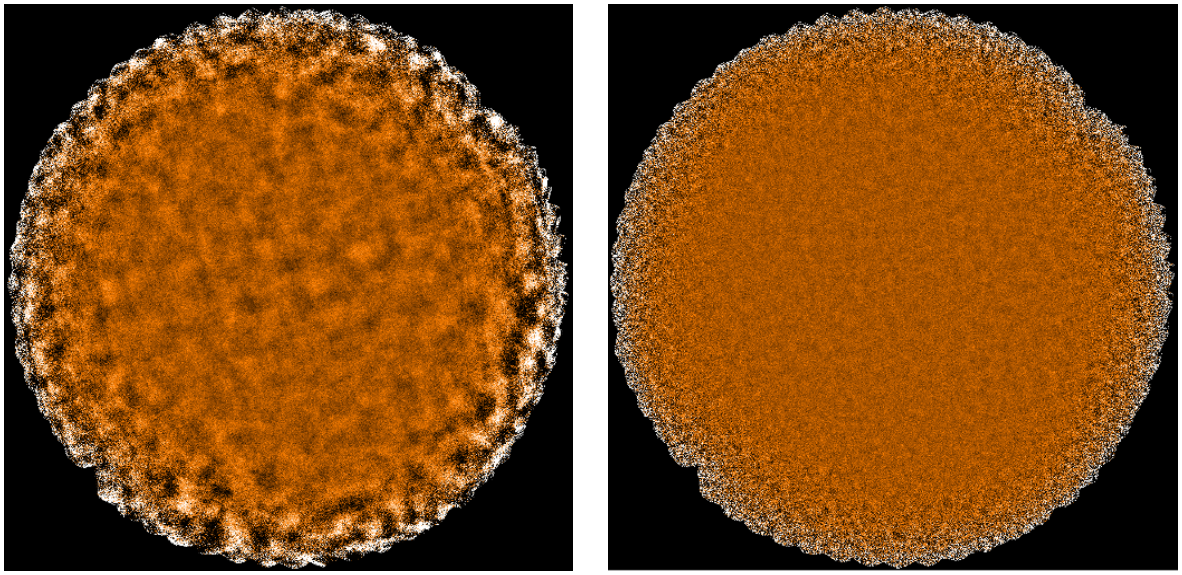


Figure 3.4: Maps of a deep cosmology field reduced with **(left)** default parameter values (*i.e.* no configuration file) and **(right)** `dimmconfig_blank_field.lis`
.

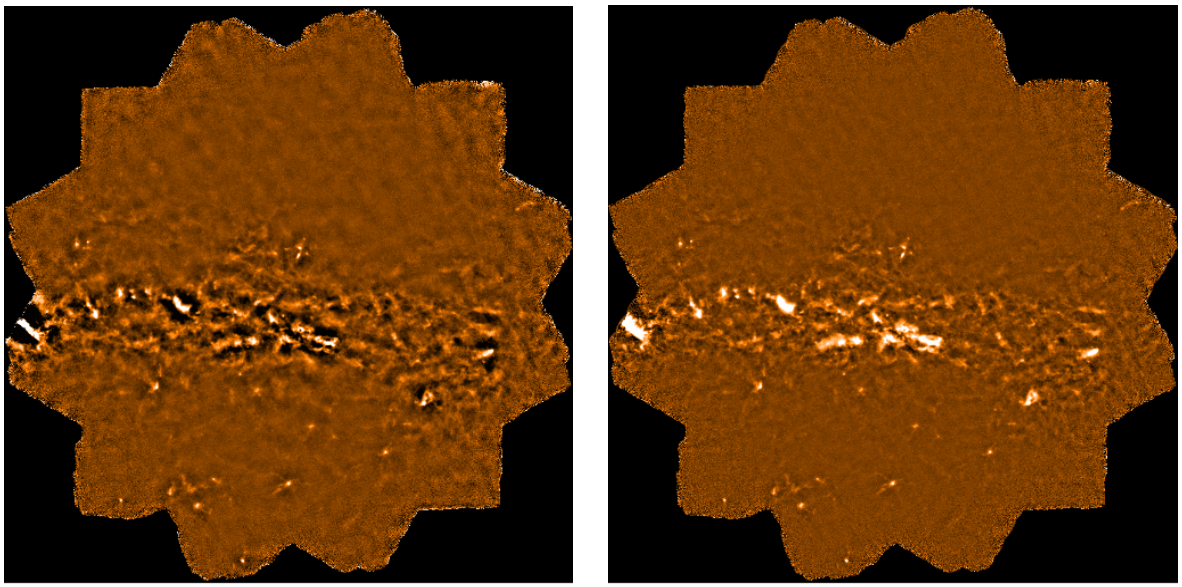


Figure 3.5: A region towards the Galactic Centre reduced with **(left)** default parameter values (*i.e.* no configuration file) **(right)** `dimmconfig_bright_extended.lis`.

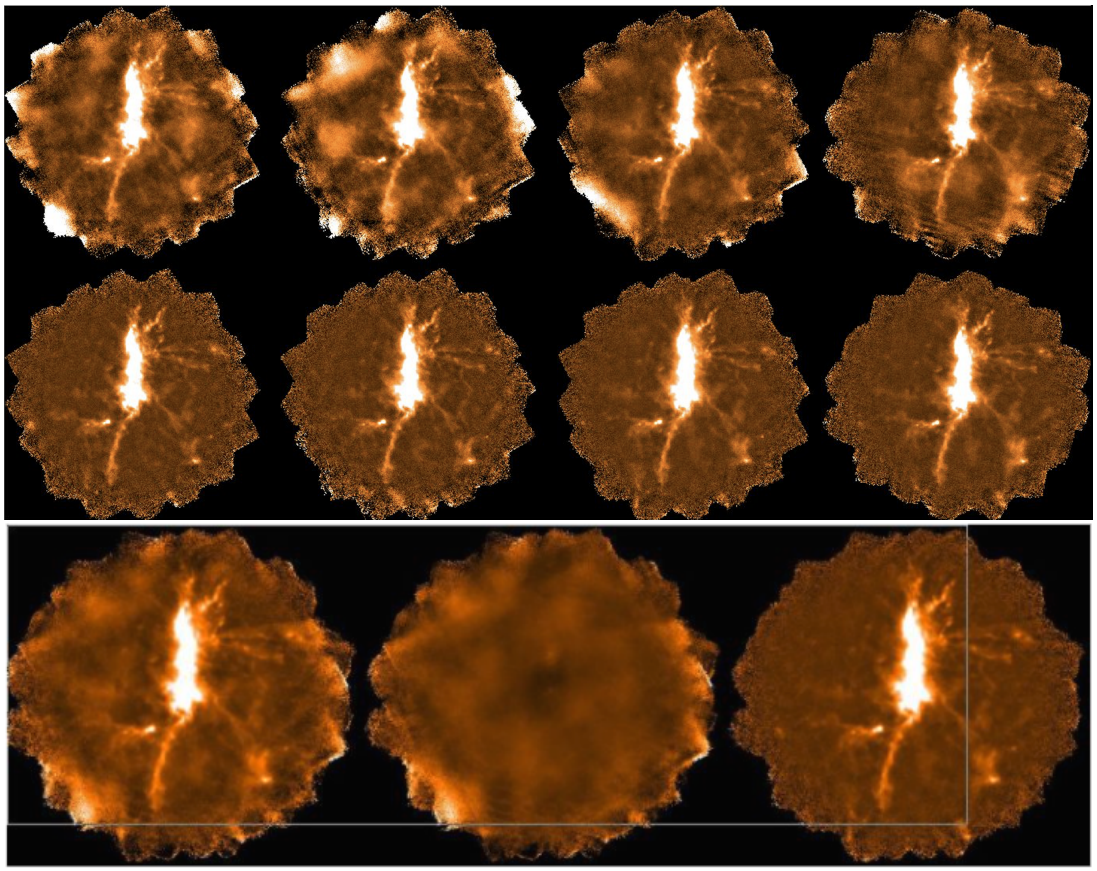


Figure 3.6: *Top*: Four 850-μm maps made with the basic "bright extended" dimmconfig. *Middle*: The results of adding in the new PCA dimmconfig file. *Bottom*: The mosaics of the top and middle rows (left and right, respectively) with a difference map in between.

Chapter 4
The SCUBA-2 Pipeline

4.1 Pipeline overview

SCUBA-2 data-reduction pipelines have been developed based on the existing ORAC-DR pipeline (Cavanagh et al., 2008[3]) used for ACSIS. There are three distinct pipelines currently utilised by SCUBA-2. Users will likely only need to run the science pipeline. The other two pipelines are designed to run at the JCMT—the quick-look (QL) and summit pipelines. The latter two are run in real time at the JCMT during data acquisition.

- The science pipeline has access to all the data observed for a given project and adopts a best-possible reduction approach. Images are made for each complete observation which are combined to create the final image. Users wishing to reduce their own data should use this pipeline. This pipeline is responsible for producing the reduced data that is accessible to users via CADC.
- The QL runs quality assurance checks on the data as they arrive. For science data, it calculates the noise between 2 Hz and 10 Hz, along with the NEP and effective NEP, for each 30-second scan. These values undergo quality-assurance checks to ensure SCUBA-2 is within an acceptable operating range.
- The summit pipeline is designed to provide a quick map of the data, it does this by running fewer iterations and chunking the data more. This is a useful guide to observers who wish to check the quality of their data.

The manual for the SCUBA-2 pipeline can be found at **SUN/264**, while the pipeline software comes as part of the Starlink suite. Data-reduction tutorials are available online[1]

4.2 The science pipeline

The science pipeline will perform the following:

- Run the iterative map-maker.
- Apply the FCF to calibrate to mJy/beam or mJy/arcsec^2.
- Co-add multiple observations of the same object.
- Apply the matched-filter (blank-field configuration file only)
- Run a source-finding algorithm.

[1] `https://www.eaobservatory.org/jcmt/science/reductionanalysis-tutorials/`

4.2.1 Pipeline recipes

When a project is initially created and MSBs (Minimum Scheduling Blocks) are constructed using the JCMT Observing Tool, the PI can select a pipeline recipe to assign to the data. When the data are run through the science pipeline this recipe is then called by default. This can be overridden on the command line—see Section 4.4. Described below are the six main ORAC-DR science recipes.

Note: the "dimmconfig*" files can be found in:

```
% ls $STARLINK_DIR/share/smurf
```

4.2.2 REDUCE_SCAN

Configuration file: `dimmconfig_jsa_generic.lis`

This recipe uses the configuration file `dimmconfig_jsa_generic` for makemap, unless the sources is identified as a calibrator in which case `dimmconfig_bright_compact.lis` is used and FCFs are derived from the map. After all observations have been processed the data are co-added and calibrated in mJy/beam using the default FCF. The noise and NEFD properties for the co-add are calculated and written to log files (`log.noise` and `log.nefd` respectively). Finally, the CUPID task findclumps is run using the FellWalker algorithm (Berry, 2015[2]) to create a source catalogue.

4.2.3 REDUCE_SCAN_CHECKRMS

Configuration file: `dimmconfig_jsa_generic.lis`

This recipe is the same as `REDUCE_SCAN`, but includes extra performance estimations determined by `SCUBA2_CHECK_RMS` (see PICARD's `SCUBA2_CHECK_RMS`). These extra metrics are written to a log file `log.checkrms`. Running `SCUBA2_CHECK_RMS` in the pipeline, rather than as a standalone PICARD recipe, allows it to calculate results for co-added maps.

4.2.4 REDUCE_SCAN_EXTENDED_SOURCES

Configuration file: `dimmconfig_bright_extended.lis`

This is the recipe for processing extended sources. Multiple observations are co-added and the output map is calibrated in units of mJy/arcsec^2. This recipe also performs a source-finder routine; the results are written as a FITS catalogue (with file extension `.FIT`) which can be read as a local catalogue into GAIA.

4.2.5 REDUCE_SCAN_FAINT_POINT_SOURCES

Configuration file: `dimmconfig_blank_field.lis`

This is the recipe for processing maps containing faint compact sources. This time the configuration file called by makemap is `dimmconfig_blank_field.lis` and the map calibrated in mJy/beam. The output map is further processed with a matched filter, then the S/N is taken to enhance point sources. A map is written out at each step. This recipe also performs a source finder routine; the results are written as a FITS catalogue (with file extension `.FIT`) which can be read as a local catalogue into GAIA.

4.2.6 REDUCE_SCAN_ISOLATED_SOURCE

Configuration file: `dimmconfig_bright_compact.lis`

This is the recipe used for processing calibrator data. It can also be used for any map of a single bright, isolated source at the tracking position.

This reduction constrains the map to zero beyond a radius of 1 arc-min from the source centre. See Section 3.7.3

4.2.7 FAINT_POINT_SOURCES_JACKKNIFE

Configuration file: `dimmconfig_blank_field.lis`

This recipe uses a jack-knife method to remove residual low-spatial frequency noise and create an optimal matched-filtered output map. The map-maker is run twice, first as a standard reduction using `dimmconfig_blank_field.lis` (and calibrated in mJy/beam), and the second time with a fake source added to the time series. This creates a signal map and an effective PSF map. A jack-knife map is generated from two halves of the dataset and the maps are 'whitened' by the removal of the residual $1/f$ noise. The whitened signal map is processed with the matched filter using the whitened PSF map as the PSF input. The data are calibrated in mJy/beam using a corrected FCF. See Section 7.1.2 for a more-detailed description of this recipe and the files produced.

4.3 Running the science pipeline

Note: Data-reduction tutorials are available online.

Step 1: **Initialise ORAC-DR**

For 850-μm data, this is done by:

```
% oracdr_scuba2_850 -cwd
```

For 450-μm data, this is done by:

```
% oracdr_scuba2_450 -cwd
```

Step 2: **Set environment variables**

These ensure the data are read from and written to the right places. Many are set automatically when the pipeline is initialised but others must be set manually. Details of the optional variables are given in **SUN/264** but the three main ones are:

- `STARLINK_DIR` – Location of your Starlink installation.
- `ORAC_DATA_IN` – The location where the data should be read from. If you are supplying a text file listing the raw data this should be the location of the files listed, unless they are given as full path names.
- `ORAC_DATA_OUT` – The location where the data products should be written. Also used as the location for a user-specified configuration file.

Example: Setting `ORAC_DATA_IN` to be the current directory for C shells (csh, tcsh):

```
% setenv ORAC_DATA_IN .
```

and for Bourne shells (sh, bash, zsh):

```
% export ORAC_DATA_IN=.
```

Step 3: **Run the pipeline**

This is done by:

```
% oracdr -files <list_of_files>
```

where the list of files that you wish to reduce can be an individual or multiple observations (one per line in a text file, with full path names).

Tip

If you run with -verbose on the command line then you will obtain all messages from the Starlink engines (rather than just ORAC-DR messages). This is particularly useful for understanding what is occurring during the map-maker stage of reduction. This is particularly recommended for new users.

When executing the ORAC-DR command, unless "-nodisplay" is specified, various graphical windows may appear showing the pipeline results. Also with the default "-log" option (including "x") a new Xwindow will appear which will contain the pipeline output, as shown in Figures 4.1– 4.4.

4.4 Changing the defaults

4.4.1 Changing ORAC-DR's behaviour

ORAC-DR's behaviour can be changed on the command line. For help simply type

```
% oracdr -help
```

To run the pipeline and obtain all messages from the Starlink engines (rather than just ORAC-DR messages) you will need to run with verbose (*recommended*)

```
% oracdr -files <list_of_files> -verbose
```

To run the pipeline and have the results sent to the screen (`s`) and to a file (`f`—the file produced is usually called `.oracdr_NNNN.log` where NNNN is the current process ID. It is written to $ORAC_DATA_OUT and is a hidden file) is specified using the `-log` command.

```
% oracdr -files <list_of_files> -log sf -verbose
```

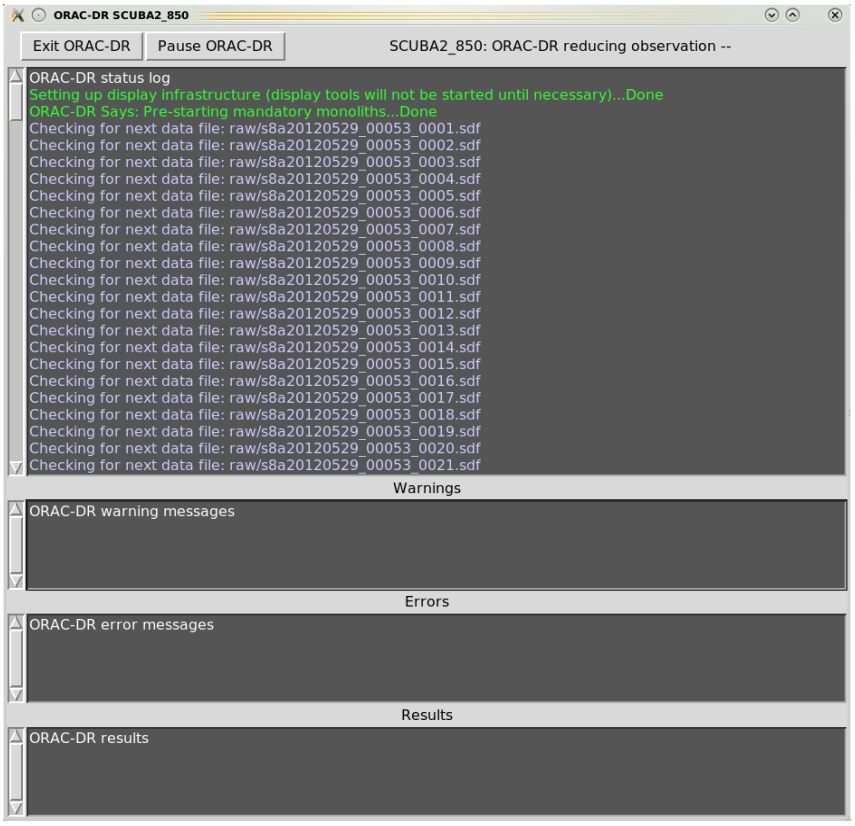


Figure 4.1: The Xwindows output from the ORAC-DR pipeline showing the initial log—here we see the pipeline is checking for the raw files.

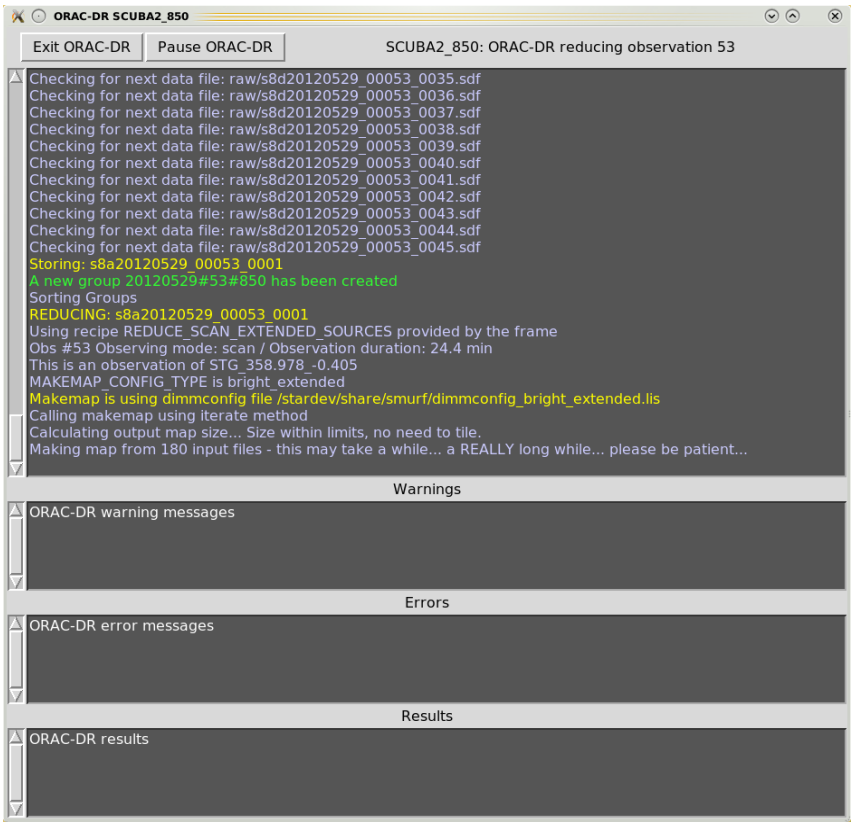


Figure 4.2: The Xwindows output from the ORAC-DR pipeline—here we see the data being reduced with the recipe REDUCE_SCAN_EXTENDED_SOURCES. The pipeline also reports the name of the observation being reduced and the duration of the observation.

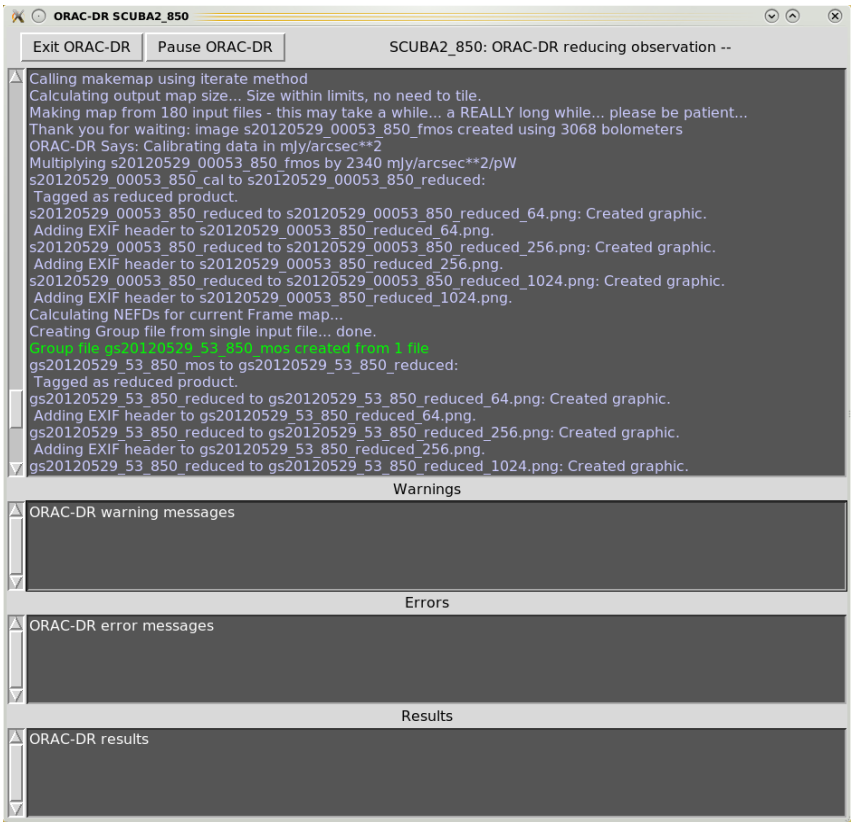


Figure 4.3: The Xwindows output from the ORAC-DR pipeline—here we we see the FCF being applied, the graphics being created along with a group file (file containing co-added observations from a single night, if provided).

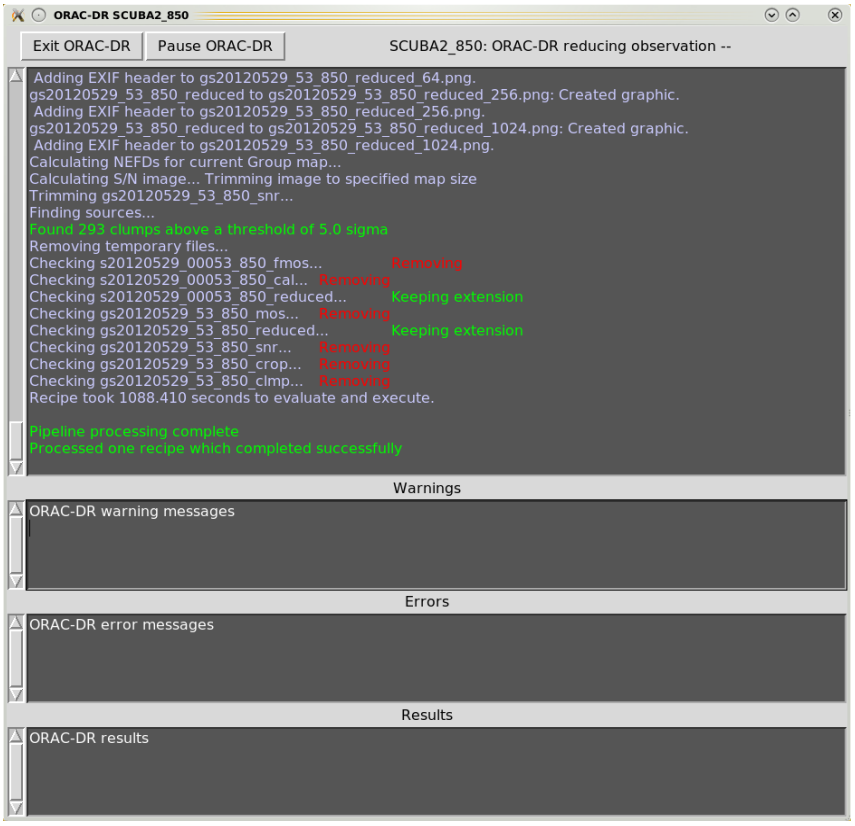


Figure 4.4: The Xwindows output from the ORAC-DR pipeline—here we see the pipeline process has completed.

4.4.2 Changing the pipeline recipe

You can override the recipe set in the header by listing any different one on the command line when starting ORAC-DR. For example

```
% oracdr -files <list_of_files> -log sf REDUCE_SCAN_CHECKRMS
```

You can find out which recipe is set in the data header via the FITS header `RECIPE` keyword in any of your raw files. For example both of these options will return the same result (ensure KAPPA commands are available before running):

```
% fitsval s8a20120725_00045_0003 RECIPE
% fitslist s8a20120725_00045_0003 | grep RECIPE
```

4.4.3 Changing the configuration file

Although each recipe calls one of the standard configuration files you can specify your own. You will need to create a recipe parameter file. This file will set the parameter `MAKEMAP_CONFIG` to be your new configuration file. The first line must be the name of the recipe used in the reduction.

For example, to run the pipeline with `REDUCE_SCAN_CHECKRMS` with a configuration file called `myconfig.lis`, the recipe parameter file (`mypars.ini`) will look like this.

```
[REDUCE_SCAN_CHECKRMS]
MAKEMAP_CONFIG = myconfig.lis
```

Then run the pipeline calling the parameter file via the `-recpars` option.

```
% oracdr -files <list_of_files> -log sf -recpars myparams.ini REDUCE_SCAN_CHECKRMS
```

4.4.4 Parameter-file options

To supply both a new configuration file and a different set of clump-finding parameters we would update the parameter file `mypars.ini` to look like:

```
[REDUCE_SCAN]
MAKEMAP_CONFIG = mynewconfig.lis
FINDCLUMPS_CFG = myfellwalkerparams.lis
```

Other options we can change in the parameter file include—changing the pixel size

```
[REDUCE_SCAN]
MAKEMAP_PIXSIZE = 2
```

changing output units to mJy/beam

```
[REDUCE_SCAN]
CALUNITS = beam
```

changing output units to mJy/arcsec

```
[REDUCE_SCAN]
CALUNITS = arcsec
```

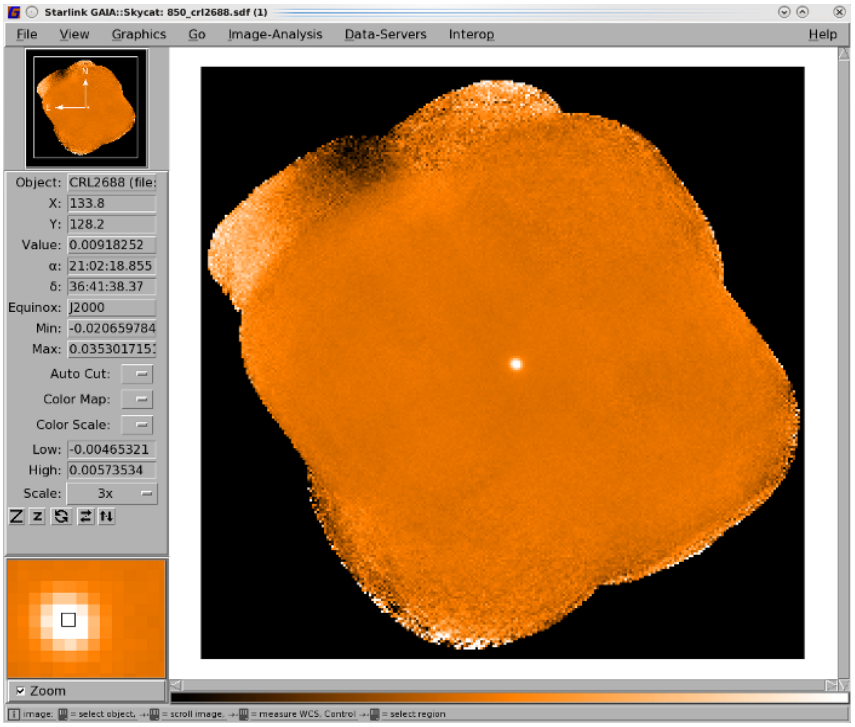


Figure 4.5: Map of CRL 2688 produced with the SMURF task makemap using the iterative algorithm with default parameters.

4.5 What to look out for

Once the map-maker has completed you can open your output map using GAIA (see Figure 4.5). The excerpt in Chapter 5 shows the output written to the terminal as you run the map-maker. There are a number of clues in this output that indicate the status of the reduction.

The number of input files The first to note is the number of input files; it is worth checking this matches your expected number. Also summarised are the source name, UT date and scan number.

Map dimension Next the basic dimensions of the data being processed are listed near the start of the first iteration. The example above has 4 arcsec pixels—the default at 850 μm.

Chunking The map-maker then determines if the raw data should be split and processed in more than one chunk. In this map the data are reduced in one continuous piece: `Continuous chunk 1 / 1`. Chunking is where the map-maker processes sub-sections of the time-series data independently and should be avoided if possible—see the text box on Page 24.

Quality statistics

At the beginning of the reduction, the main purpose of QUALITY flagging is to indicate how many bolometers are being used. In the example above you can see that from a total of 5120 bolometers, 1842 were turned off during data acquisition (`BADDA`). In addition, 136 bolometers exceeded the acceptable noise threshold (`NOISE`), while tiny fractions of the data were flagged because the telescope was moving too slowly (`STAT`) or the sample are adjacent to a step that was removed (`DCJUMP`).

The total number of bad bolometers (BADBOL) is 1984. Accounting for these, and the small numbers of additionally flagged samples, 3128.22 effective bolometers are available after initial cleaning[2].

After each subsequent iteration a new 'Quality' report is produced, indicating how the flags have changed. An important flag that appears in the 'Quality' report following the first iteration is COM: the DIMM rejects bolometers (or portions of their time series) if they differ significantly from the common-mode (average) of the remaining bolometers.

You may note that compared with the initial report, the total number of samples with good 'Quality' (`Total samples available for map`) has dropped from 18634826 to 18273302 (about a 2 per cent decrease) as additional samples were flagged in each iteration.

Be aware that some large reductions may take many iterations to reach convergence and you may find significantly fewer bolometers remaining resulting in higher noise than expected.

Convergence

The convergence criteria `maptol` is updated for each iteration. The convergence can be checked from the line reporting

```
smf_iteratemap:  *** NORMALIZED MAP CHANGE: 0.10559 (mean) 2.81081 (max)
```

The number to look out for is the mean value of the NORMALIZED MAP CHANGE. This will have to drop below your required `maptol` for convergence to be achieved.

The default configuration file used in this example executes a maximum of five iterations, but stops sooner if the change in `maptol` drops below 0.05 (i.e. `numiter =−5`). In this example it stops after five iterations.

Tip

You can interrupt the processing at any stage with a single `Ctrl-C`. The mapmaker will complete the iteration then write out a final science map. Entering `Ctrl-C` twice will kill the process immediately.

4.6 Pipeline output

The pipeline will produce a *group* file for each object being processed. If the pipeline is given data from multiple nights, all those data will be included in the group co-add using inverse variance weighting.

The final maps in your output directory will have the suffix `_reduced`. Maps will be made for individual observations, which will start with an `s` for "SCUBA-2" (e.g. `s20140620_00030_850_reduced.sdf`). Group maps, which may contain co-added observations from a single night, are also produced which have the prefix `gs` for "group SCUBA-2" and the date/scan of the *first input file* (e.g. `gs20140620_30_850_reduced.sdf`).

Note: A group file is *always* created, even if only a single observation is being processed.

Additionally, PNG images are made of the reduced files at a variety of resolutions.

Another useful feature is that the pipeline will generate log files to record various useful quantities. The standard log files from reducing science data are:

[2]The fractional number is due to time-slices being removed during cleaning. The number of bolometers is then reconstructed from the number of remaining time-slices.

- `log.noise`—noise in the map for each observation and the co-add (calculated from the median of the error array), and
- `log.nefd`—NEFD calculated for each observation and for the co-added map(s).
- `log.removedobs`—list of observations removed from each group (e.g. due to failing QA).

4.7 Getting your data from CADC

The JCMT Science Archive is hosted by The Canadian Astronomy Data Centre (CADC). Both raw data and data processed by the science pipeline are made available to PIs and co-Is through the CADC interface (`https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/jcmt/`).

To access proprietary data you will need to have your CADC username registered by the EAO and thereby associated with the project code. Please contact your friend of project or `helpdesk@eaobservatory.org` to register your account.

An important search option to be aware of is 'Group Type', where your options are Simple, Night, Project and Public. Simple (which becomes 'obs' on the result page) is an individual observation; night means the group file from the pipeline (these may or may not include more than one observation; the 'Group Members' value will tell you); and the project option is generated if an entire project has been run through the pipeline and identical sources across the project are co-added into master group files.

Chapter 5
Running makemap Outside the Pipeline

The previous chapter described how to make maps using the SCUBA-2 Pipeline. Users—particularly new users—are encouraged to use the pipeline for their map-making. However, greater control of the map-making process is available by running the makemap command directly, rather than from within the pipeline. This chapter describes how to do this, but should be seen as **"advanced usage"**.

Note, when makemap is run manually (rather than from the pipeline) the atmospheric extinction correction is applied automatically (see Appendix B), but the resulting image will be in units of pW, not Jy. Each pixel value in the map represents the weighted mean value of the bolometer samples (in pW) that fall within the pixel, removal of the noise components described in Section 3.3. Each weight is the reciprocal of the variance associated with the bolometer. Thus each pixel value can be thought of as the astronomical power received by a typical bolometer at each point on the sky. Section 8.1 describes how to convert a map from units of pW to Jy; see Appendices B, E, and F for more information.

5.1 Running makemap

Before running makemap directly, you need to ensure that the Starlink environment has been initialised and the SMURF package started (see Section 1.3.2 and Section 1.3.3).

To run makemap, you need to supply values for the following command-line parameters[1]:

`IN` A list of input NDFs containing raw SCUBA-2 data (see Section 2.3 and Section 1.3.1). There are many ways in which the list of files can be supplied, as described in Section "Specifying Groups of Objects" in SUN/95. The easiest is to create a simple text file containing the names of the raw data files – one per line—and then supply the name of the text file, preceded by an up-caret character (`^`), as the value for parameter `IN`. Note, the names of the raw data files can contain wild-cards such as "∗" and "?". See below for examples.

`OUT` The name of the NDF in which to store the final map. The supplied file name should either have a file type of "`.sdf`", or no file type at all (in which case `.sdf` will be appended to the supplied value). Any existing file with the same name will be over-written.

`CONFIG` A string that specifies new values for one or more configuration parameters. These new values are used in place of the default values described in SUN/258. Any configuration parameter not specified by the `CONFIG` string retains its default value. As with files names, there are many ways in which the group of parameter values can be specified, and the same documentation should be consulted for details (SUN/95). Again, the easiest way is to list the parameter values in a simple text file with one parameter setting (a "<name>=<value>" string) on each line. See Section 3.7 for a list of the pre-defined configuration files that come with SMURF. Note, `CONFIG` can be set to the

[1]Note the distinction between "command-line parameters" (also known as "ADAM" parameters) that are supplied on the `makemap` command line, and "configuration parameters" that are specified within a configuration file. Values for all *configuration* parameters are obtained using a single *command-line* parameter called `CONFIG`.

special value "def" to force makemap to use the default values for all parameters. It *is* possible to create a map using just these default values, but it will not usually be a very good map. Advice on which parameters to edit can be found in Section 6.

If any of the above parameters are *not* given on the makemap command line, prompts will be issued and you will be invited to supply values for the missing parameters.

So an example command line would be:

```
% makemap in=^myfiles.lis out=850_crl2688 \
          config=^$STARLINK_DIR/share/smurf/dimmconfig_bright_compact.lis
```

Here, the file myfiles.lis contains a list of the raw data files to be included in the map, and could for instance look like this:

```
% cat myfiles.lis
/jcmtdata/raw/scuba2/s8a/20120720/00030/*
/jcmtdata/raw/scuba2/s8b/20120720/00030/*
/jcmtdata/raw/scuba2/s8c/20120720/00030/*
/jcmtdata/raw/scuba2/s8d/20120720/00030/*
```

This uses all available data for all four 850 μm sub-arrays, for observation 30 taken on 20th July 2012[2].

The file dimmconfig_bright_compact is one of the pre-defined configuration files supplied with SMURF. It contains configuration parameter values that are optimised for creating maps from bright compact sources.

Tip

An up-caret (^) is required any time you are reading in a group text file in Starlink. For the map-maker this includes the configuration file (a group of configuration parameters) and the list of input files (a group of NDFs—*e.g.* in=^myfiles.lis).

After the makemap command completes, the final map will be left in file 850_crl2688.sdf and can be displayed using GAIA:

```
% gaia 850_crl2688
```

There are many other command-line parameters that can be supplied when running makemap, but only the three listed above are mandatory. All the others assume default values if not supplied on the command-line. For a full list of all available command-line parameters, their functions and default values, see the description of makemap within the SMURF document, SUN/258.

You can supply values for any of these extra parameters on the command line. For instance, one of the command-line parameters that is usually defaulted is PIXSIZE, which specifies the pixel size for the final map, in arcseconds. The default pixel sizes, defined as one quarter of the Airy disk rounded up to the nearest half arcsecond, are:

- 2 arcsec at 450μm

[2]The input files should all be for a single waveband and a single observation—do not mix files from different wavebands and/or observations.

- 4 arcsec at 850μm

So for instance, the above command-line could be modified as follows to produce a map with 3-arcsecond pixels:

```
% makemap in=^myfiles.lis out=850_crl2688 pixsize=3 \
          config=^$STARLINK_DIR/share/smurf_bright_compact.lis
```

Note, if parameters are specified by name on the command-line, as in these examples, then the order in which they are specified is insignificant[3]. Also, parameter names are case-insensitive.

Tip

Map-maker not finding your raw data files from a path? Check you have escaped or protected any shell meta-characters in your 'in' value, for instance by putting double quotes around it or escaping wild-cards using a backslash (*e.g.* "`in=\*.sdf`" or just "`in=\*`"). Note, this issue applies only to wild-cards included directly on the command line—there is no need to escape or protect wild cards within a text file.

5.2 Interpreting the screen output from makemap

Whilst makemap runs, it outputs information continuously to the screen about what it is doing. The amount of information displayed can be controlled using the `MSG_FILTER` command-line parameter. It defaults to "`normal`", but if you want more information you could set it to (say) "`verbose`":

```
% makemap in=^myfiles.lis out=850_crl2688 msg_filter=verb \
          config=^$STARLINK_DIR/share/smurf_bright_compact.lis
```

Note—unambiguous abbreviations may be used for many command line parameters—so "`verb`" is acceptable instead of "`verbose`". Setting `MSG_FILTER` to "`quiet`" will suppress all screen output.

Tip

Map-maker generates a lot of screen output! The main things to check are that the "`NORMALIZED MAP CHANGE`" value (the mean value, not the max value) decreases nicely towards your requested `maptol` value as each iteration is completed, and that the "`Total samples available for map:`" value does not fall too low (you should usually be looking for values above 50% of maximum). If neither of these two items look problematic, it is usually safe to pay less attention to the other screen output.

The following shows the screen output generated by a typical run of makemap if `MSG_FILTER` is left set to its default value of "`normal`". Explanatory comments, which are not actually part of the output generated by makemap, are included in *emphasised type*. The input data are a short observation of a bright compact source (CRL 2688):

[3]Some commands, for instance most KAPPA commands, allow the most important parameters to be specified by position rather than name, in which case their order *is* significant — see `Specifying Parameter Values on Command Lines` in SUN/95.

```
% makemap in=^myfiles.lis out=850_crl2688 \
          config=^$STARLINK_DIR/share/smurf_bright_compact.lis

Out of 32 input files, 4 were darks, 8 were fast flats and 20 were science
Processing data from instrument 'SCUBA-2' for object 'CRL2688' from the
following observation  :
  20120720 #30 scan
```

The output starts by reporting information about the input data files, including the number that contain on-sky bolometer values ("science" data), the astronomical object and the SCUBA-2 observation number.

```
MAKEMAP: Map-maker will use no more than 92586 MiB of memory

   Projection parameters used:
      CRPIX1 = 0
      CRPIX2 = 0
      CRVAL1 = 315.578333333333 ( RA = 21:02:18.800 )
      CRVAL2 = 36.6938055555556 ( Dec = 36:41:37.70 )
      CDELT1 = -0.00111111111111111 ( -4 arcsec )
      CDELT2 = 0.00111111111111111 ( 4 arcsec )
      CROTA2 = 0

   Output map pixel bounds: ( -132:122, -126:129 )

   Output map WCS bounds:
        Right ascension: 21:01:38.318 -> 21:03:03.280
        Declination: 36:33:07.19 -> 36:50:11.70
```

Next comes information about the output map. The world coordinate system is described by means of the equivalent FITS-WCS keywords[4]

The reported pixel bounds of the output map refer to a pixel coordinate system in which the source is centred at position (0,0). Note, the definition of pixel coordinates within the NDF format allows the origin of pixel coordinates to be at any nominated position within the array—it does not have to be at the bottom left corner as in FITS—and `makemap` *chooses to put the pixel origin at the specified source position.*

Finally, the bounds of the map are given in the celestial coordinate system specified by the `SYSTEM` *command-line parameter. This parameter defaults to "*`tracking`*", which causes the map to be created in the celestial coordinate system in which the observation parameters were originally defined. It may instead be set to a specific coordinate system (e.g. "galactic", "icrs", etc.) to force the map to be made in that system.*

```
smf_iteratemap: will down-sample data to match angular scale of 4 arcsec
smf_iteratemap: Iterate to convergence (max 40)
smf_iteratemap: stop when mean normalized map change < 0.05
```

By default, the data are down-sampled so that the on-sky distance between adjacent samples is roughly equal to the pixel size (4 arcseconds in this case). This saves memory and computing time without adversely affecting the final map. The degree of down-sampling can be controlled using the `downsampscale` *configuration parameter.*

Next comes information about the stopping criteria for the iterative map-making algorithm. In this case, iterations will stop when 40 iterations are completed, or the normalised change in the map between iterations reduces to less that 0.05. These values are specified by configuration parameters `numiter` *and* `maptol` *(see Section 3.4).*

[4] In fact, NDF data structures do not use FITS-WCS to describe WCS—instead they use the AST library, which provides a much more flexible scheme for handling WCS.

```
smf_iteratemap: provided data are in 1 continuous chunks, the largest of which
has 5957 samples (153.729 s)
smf_iteratemap: map-making requires 1626 MiB (map=28 MiB model calc=1598 MiB)
smf_iteratemap: Continuous chunk 1 / 1 =========
```

In almost all cases, the raw data files constituting a SCUBA-2 observation will correspond to a single continuous stream of data samples, taken at roughly 200 Hz. Sometimes however, this may not be the case[5]*, so you are now told how many chunks of data were found, and how long they were. Something may be wrong if the input data contains any breaks.*

Next comes information about the amount of memory needed to make the map. If insufficient memory is available to process all the input data together, it will be split into chunks, and a separate map made from each chunk. These maps are later co-added to form the final output map. This can often result in a poorer map—see the box describing Data Chunking *on Page 24) .*

Finally, a loop is entered to process each chunk in turn, and you are told which chunk is currently being processed. In this case there is only one chunk (which is good).

```
smf_iteratemap: Iteration 1 / 40 ---------------
```

We now start the first iteration of the iterative algorithm described in Section 3.2, to create a map from the current chunk of input data. We will be performing —- at most—40 iterations, as set by `numiter`*.*

```
--- Size of the entire data array -----------------------------------------
bolos  : 5120
tslices: 5957(2.6 min)
Total samples: 30499840
\begin{terminalv}
--- Quality flagging statistics -------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos
BADBOL:   10865568 (35.62%),        1824 bolos
DCJUMP:      19631 ( 0.06%),
  STAT:      71680 ( 0.24%),          14 tslices
 NOISE:      41699 ( 0.14%),           7 bolos
Total samples available for map:   19586411, 64.22% of max (3287.97 bolos)
```

Now we have a number of statistics describing the cleaned data prior to the first iteration[6]*.*

Each sub-array contains a grid of 32×40 *bolometers, making 5120 bolometers in total over all four sub-arrays. The number of time slices in the concatenated data after down-sampling is reported. In this case, 5957 time slices over 2.6 minutes, equating to a down-sampled frequency of about 38 Hz. The total number of samples is the product of the number of bolometers and the number of time slices. Each of the following lines indicates the percentage of the data that has been flagged as unusable for various reasons:*

- *BADDA – flagged as unusable during data acquisition.*
- *DCJUMP – flagged as unusable because of a sudden step change in the base-line.*
- *STAT – flagged as unusable because the telescope was stationary (or at least moving too slowly).*
- *NOISE – flagged as unusable because they were too noisy.*

[5] A common cause of this is if some sub-scans are omitted from the list of input files supplied to makemap.

[6] Setting `MSG_FILTER=verbose` on the `makemap` command line will generate more information about the cleaning process.

The BADBOL item gives the fraction of bolometers that have been flagged as entirely bad for any of these reasons, and from which no data will be used.

```
smf_iteratemap: Calculate time-stream model components
smf_iteratemap: Rebin residual to estimate MAP
smf_iteratemap: Calculate ast
```

Indicates the models that are being calculated. More information about each individual model is displayed if you set `MSG_FILTER=verbose` *on the* `makemap` *command line.*

```
--- Quality flagging statistics ---------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos  ,change          0 (+0.00%)
BADBOL:   11008536 (36.09%),        1848 bolos  ,change     142968 (+1.32%)
DCJUMP:      19631 ( 0.06%),                    ,change          0 (+0.00%)
  STAT:      71680 ( 0.24%),          14 tslices,change          0 (+0.00%)
   COM:     372786 ( 1.22%),                    ,change     372786 (+0.00%)
 NOISE:      41699 ( 0.14%),           7 bolos  ,change          0 (+0.00%)
Total samples available for map:   19214687, 63.00% of max (3225.56 bolos)
     Change from last report:    -371724, -1.90% of previous
smf_iteratemap: Will calculate chi^2 next iteration
smf_iteratemap: *** NORMALIZED MAP CHANGE: 2.22778 (mean) 60.5292 (max)
```

More statistics are displayed once the first iteration is completed and the first estimate of the science map has been generated. Data samples may be flagged as bad within the iterative stage for various reasons, and so these statistics may be different to those shown prior to the start of the iterative stage. Most significantly, the `COM:` *line shows the percentage of data that has been rejected because the time stream did not resemble the common-mode signal closely enough. The "*`NORMALIZED MAP CHANGE`*" values are of no real significance on the first iteration since there is no previous map with which to compare the new map (in fact the numerical values are generated by comparing the new map with a map full of zeros).*

```
smf_iteratemap: Iteration 2 / 40 ---------------
smf_iteratemap: Calculate time-stream model components
smf_calcmodel_noi: Calculating a NOI variance for each box of 581 samples
using variance of neighbouring residuals.
smf_iteratemap: Rebin residual to estimate MAP
smf_iteratemap: Calculate ast
--- Quality flagging statistics ---------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos  ,change          0 (+0.00%)
BADBOL:   11038321 (36.19%),        1853 bolos  ,change      29785 (+0.27%)
 SPIKE:         36 ( 0.00%),                    ,change         36 (+0.00%)
DCJUMP:      19631 ( 0.06%),                    ,change          0 (+0.00%)
  STAT:      71680 ( 0.24%),          14 tslices,change          0 (+0.00%)
   COM:     393125 ( 1.29%),                    ,change      20339 (+5.46%)
 NOISE:      41699 ( 0.14%),           7 bolos  ,change          0 (+0.00%)
Total samples available for map:   19194401, 62.93% of max (3222.16 bolos)
     Change from last report:     -20286, -0.11% of previous
smf_iteratemap: *** CHISQUARED = 0.997314089857974
smf_iteratemap: *** NORMALIZED MAP CHANGE: 1.78691 (mean) 8.09056 (max)
```

The second iteration starts, and proceeds in much the same way as the first iteration. The main difference is that the NOI model is generated at the start of the second iteration. This model measures the variance within each bolometer

time-stream. It is used to weight the bolometer samples when forming the mean sample value in each map pixel[7]*. NOI is calculated from the residuals left after removal of all other models, and is only calculated once—subsequent iterations re-use the same NOI values.*

The `SPIKE:` *item that has appeared in the quality flagging statistics records the number of samples that have been rejected as transient spikes in the time-series. This flagging is done by comparing the bolometer values that fall in each map pixel, and flagging any that appear to be statistical outliers—see configuration parameter* `ast.mapspike`*.*

The mean "`NORMALIZED MAP CHANGE`*" value should drop with each subsequent iteration (note, the "max" normalised map change value can usually be ignored as it records the maximum normalised change in any single map pixel and is thus highly subject to random variations).*

```
smf_iteratemap: Iteration 3 / 40 ---------------
smf_iteratemap: Calculate time-stream model components
smf_iteratemap: Rebin residual to estimate MAP
smf_iteratemap: Calculate ast
--- Quality flagging statistics --------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos  ,change          0 (+0.00%)
BADBOL:   11050235 (36.23%),        1855 bolos  ,change      11914 (+0.11%)
 SPIKE:         36 ( 0.00%),                    ,change          0 (+0.00%)
DCJUMP:      19631 ( 0.06%),                    ,change          0 (+0.00%)
  STAT:      71680 ( 0.24%),          14 tslices,change          0 (+0.00%)
   COM:     401600 ( 1.32%),                    ,change       8475 (+2.16%)
 NOISE:      41699 ( 0.14%),           7 bolos  ,change          0 (+0.00%)
Total samples available for map:   19185947, 62.91% of max (3220.74 bolos)
     Change from last report:      -8454, -0.04% of previous
smf_iteratemap: *** CHISQUARED = 0.976599086972009
smf_iteratemap: *** change: -0.0207150028859653
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.305 (mean) 1.24407 (max)
```

The mean normalised map change continues to drop with the third iteration, as expected. The total number of samples going into the map is dropping with each iteration, but only very slowly. This is mainly due to the increased number of samples being flagged by the COM model, but is at an acceptably small level.

```
smf_iteratemap: Iteration 4 / 40 ---------------
smf_iteratemap: Calculate time-stream model components
smf_iteratemap: Rebin residual to estimate MAP
smf_iteratemap: Calculate ast
--- Quality flagging statistics --------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos  ,change          0 (+0.00%)
BADBOL:   11056192 (36.25%),        1856 bolos  ,change       5957 (+0.05%)
 SPIKE:         36 ( 0.00%),                    ,change          0 (+0.00%)
DCJUMP:      19631 ( 0.06%),                    ,change          0 (+0.00%)
  STAT:      71680 ( 0.24%),          14 tslices,change          0 (+0.00%)
   COM:     409883 ( 1.34%),                    ,change       8283 (+2.06%)
 NOISE:      41699 ( 0.14%),           7 bolos  ,change          0 (+0.00%)
Total samples available for map:   19177684, 62.88% of max (3219.35 bolos)
     Change from last report:      -8263, -0.04% of previous
smf_iteratemap: *** CHISQUARED = 0.96404406922676
smf_iteratemap: *** change: -0.0125550177452489
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.0401588 (mean) 0.151877 (max)
```

[7] Bolometers are given equal weight in the map created at the end of the first iteration since the NOI model has not yet been calculated at that point.

After the fourth iteration the mean normalised map change has dropped to 0.0401588, which is below the value of 0.05 provided for the `maptol` *parameter by the* `dimmconfig_bright_compact` *configuration file*[8]*. Consequently,* `makemap` *considers the map to have converged. However, in view of the fact that the* `dimmconfig_bright_compact` *configuration file uses "AST masking" (see Section 3.5.1), it is necessary to do one final iteration in order to assign correct values to the pixels that lie outside the source mask. Note, when AST masking is being used, the reported normalised map change values only include pixels that are within the source mask.*

```
smf_iteratemap: Iteration 5 / 40 ---------------
smf_iteratemap: Calculate time-stream model components
smf_iteratemap: Rebin residual to estimate MAP
smf_iteratemap: Calculate ast
--- Quality flagging statistics --------------------------------------------
 BADDA:   10805998 (35.43%),        1814 bolos  ,change          0 (+0.00%)
BADBOL:   11068106 (36.29%),        1858 bolos  ,change      11914 (+0.11%)
 SPIKE:         36 ( 0.00%),                    ,change          0 (+0.00%)
DCJUMP:      19631 ( 0.06%),                    ,change          0 (+0.00%)
  STAT:      71680 ( 0.24%),          14 tslices,change          0 (+0.00%)
   COM:     414727 ( 1.36%),                    ,change       4844 (+1.18%)
 NOISE:      41699 ( 0.14%),           7 bolos  ,change          0 (+0.00%)
Total samples available for map:   19172853, 62.86% of max (3218.54 bolos)
     Change from last report:      -4831, -0.03% of previous
smf_iteratemap: *** CHISQUARED = 0.964032127175568
smf_iteratemap: *** change: -1.19420511923707e-05
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.0157131 (mean) 0.103135 (max)
smf_iteratemap: ****** Completed in 5 iterations
smf_iteratemap: ****** Solution CONVERGED
Setting 24282 map pixels bad because they contain fewer than 4 samples (=0.01
of the mean samples per pixel).
Total samples available from all chunks: 19172853 (3218.54 bolos)
```

After the extra iteration required by AST masking has been performed, the final output map is created. It is always advisable to check the final number of samples available for the map, as a low value will cause your map to have high noise levels. In this case, 62.86% of the samples are available for the map, which is quite acceptable—only a couple of percent of the samples have been rejected by the map-making, mostly flagged by the COM model. The bulk of the bad samples (35.43%) were rejected during data acquisition due to dead bolometers etc.

Note, the `numiter` *parameter is set to -40 by* `dimmconfig_bright_compact`*, meaning that no more than 40 iterations will be performed. In this particular case we only needed 4 iterations (plus a mandatory extra iteration) to achieve our requested* `maptol` *value. But it is possible for some observations—particularly observations of extended sources—to require more than 40 iterations to converge to a* `maptol` *of 0.05. In such cases the screen output will end with a message saying that the solution "FAILED TO CONVERGE"*[9]*. In such cases, you* could *simply increase* `numiter` *and re-run* `makemap`*, but you may also want to investigate the cause of the slow convergence using one or more of the techniques described in Chapter 9, as it may indicate some issue with the raw data.*

Map pixels that receive a very small number of samples are automatically set bad. These are usually the pixels around the periphery of the observation, and will have very unreliable variance estimates. The threshold is determined by the `hitslimit` *parameter, which defaults to 1 percent of the mean number of hits per pixel, corresponding to 4 samples per pixel in this case.*

The observation used in this particular case was a short observation of a calibrator, lasting only 2.6 minutes. Consequently there was no need to divide the data up into multiple chunks in order to fit it into memory. However, for very long observations, or for shorter observations when using a computer with less than the recommended amount of memory (see Section 1.2), it may be necessary to process the raw data in multiple chunks. If this happens, a map is created from each chunk in turn, and all these maps are then added together to form the final map. The

[8] In fact, 0.05 is the default `maptol` value, which is left unchanged by `dimmconfig_bright_compact.lis`.

[9] However a map will still be created, but should be used with caution.

final message "`Total samples available from all chunks: 19172853 (3218.54 bolos)`" indicates the total number of samples (and equivalent number of bolometers) used from all chunks. In this case there was only one chunk, so these values are equal to the numbers reported at the end of the last iteration. Chunking often results in a poorer map and should be avoided if possible—see the box describing Data Chunking *on Page 24) .*

5.3 Interacting with `makemap` during a long run

For long observations, makemap can take several hours to run, particularly on slow or low-memory computers. For this reason it is useful to be able to monitor progress so that potential problems can be detected early, in order to avoid wasting time.

5.3.1 Monitoring screen output

The most straight-forward way of monitoring progress is to check the values written to the screen by makemap at the end of each iteration (see the previous section). For instance, if you run makemap as follows:

```
% makemap in=^infiles out=outmap config=^conf | tee makemap.log
```

then the messages displayed by makemap on the screen will also be written to text file `makemap.log`. This makes it easy to search the log file whilst makemap is still running, for instance from another terminal window:

```
% grep NORMALIZED makemap.log
smf_iteratemap: *** NORMALIZED MAP CHANGE: 1.02829 (mean) 20.7922 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.739832 (mean) 9.81836 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.370128 (mean) 5.42933 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.258304 (mean) 3.2717 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.205415 (mean) 2.66815 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.177044 (mean) 2.28417 (max)
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.159569 (mean) 2.05803 (max)
```

This displays the normalised change in maps between successive iterations, for the iterations that have so far been completed. If the mean normalised map change is not decreasing smoothly, (for instance if it is oscillating around a fixed value), then there is a potential problem. In which case you may want to interrupt the makemap process, rather than waiting potentially for several hours just to end up with a bad map.

Likewise, it is useful to monitor the total number of samples that are being pasted into the map at the end of each iteration:

```
% grep "Total samples" makemap.log
Total samples: 299919360
Total samples available for map:  174105275, 58.05% of max (2972.2 bolos)
Total samples available for map:  171284110, 57.11% of max (2924.03 bolos)
Total samples available for map:  171242020, 57.10% of max (2923.32 bolos)
Total samples available for map:  171239147, 57.10% of max (2923.27 bolos)
```

The lower the percentage of samples included in the map, the greater will be the noise in the map. If this number drops much below 50% then you may want to think about aborting makemap and investigating why the number is so low (*e.g.* by looking at the "quality statistics" displayed at the end of each iteration, to determine the cause of the data loss).

Likewise, information can be gathered about chunking:

```
% grep chunk makemap.log
smf_iteratemap: provided data are in 7 continuous chunks, the largest of
which has 487 samples (10.1534 s)
smf_iteratemap: Continuous chunk 1 / 7 =========
smf_iteratemap: Adding map estimated from this continuous chunk to total
smf_iteratemap: Continuous chunk 2 / 7 =========
smf_iteratemap: Adding map estimated from this continuous chunk to total
smf_iteratemap: Continuous chunk 3 / 7 =========
smf_iteratemap: Adding map estimated from this continuous chunk to total
```

This indicates that the raw data has gaps in it, resulting in seven maps being made, one from each continuous chunk, before the final map is made by combining all the individual maps. Chunking can produce sub-optimal maps, and should normally be investigated—see the description of *Data Chunking* on Page 24) .

5.3.2 Monitoring the map at the end of each iteration

The map created at the end of each iteration is normally discarded, but can be saved by setting configuration parameter `itermap` to either 1 or 2. By default, these "itermaps" are written to an extension of the main output NDF as described in Section 6.2, and cannot be viewed until `makemap` has completed. However, an alternative destination for these itermaps can be specified on the `makemap` command-line as follows, allowing them to be viewed whilst `makemap` is still running:

```
% cat conf
^STARLINK_DIR/share/smurf/dimmconfig_jsa_generic.lis
itermap=1
%
% makemap in=^infiles out=outmap config=^conf itermaps=myitermaps
```

Note the distinction between the "`itermaps`" (plural) *command-line* option, and the "`itermap`" (singular) *configuration* parameter. As soon as the first iteration has finished, the above command will create a new file called `myitermaps.sdf` in which each itermap will be stored as soon as it is created. The file is closed after each itermap is written, and re-opened again before writing the next itermap. This is an example of a *container file*, where a single `.sdf` file contains several NDFs, each holding a different map (see Section 1.3.1). You can list the maps in such a file using KAPPA `ndfecho` as follows:

```
% ndfecho myitermaps
myitermaps.CH00I001
myitermaps.CH00I002
myitermaps.CH00I003
myitermaps.CH00I004
myitermaps.CH00I005
myitermaps.CH00I006
myitermaps.CH00I007
...
```

The name of each itermap is of the form "CHxxIyyy", where "xx" is the chunk number and "yyy" is the iteration number. In the common case where all raw data are processed in a single chunk, "xx" will be "00" for all itermaps.

You can view a single itermap using:

```
% gaia myitermaps.CH00I004
```

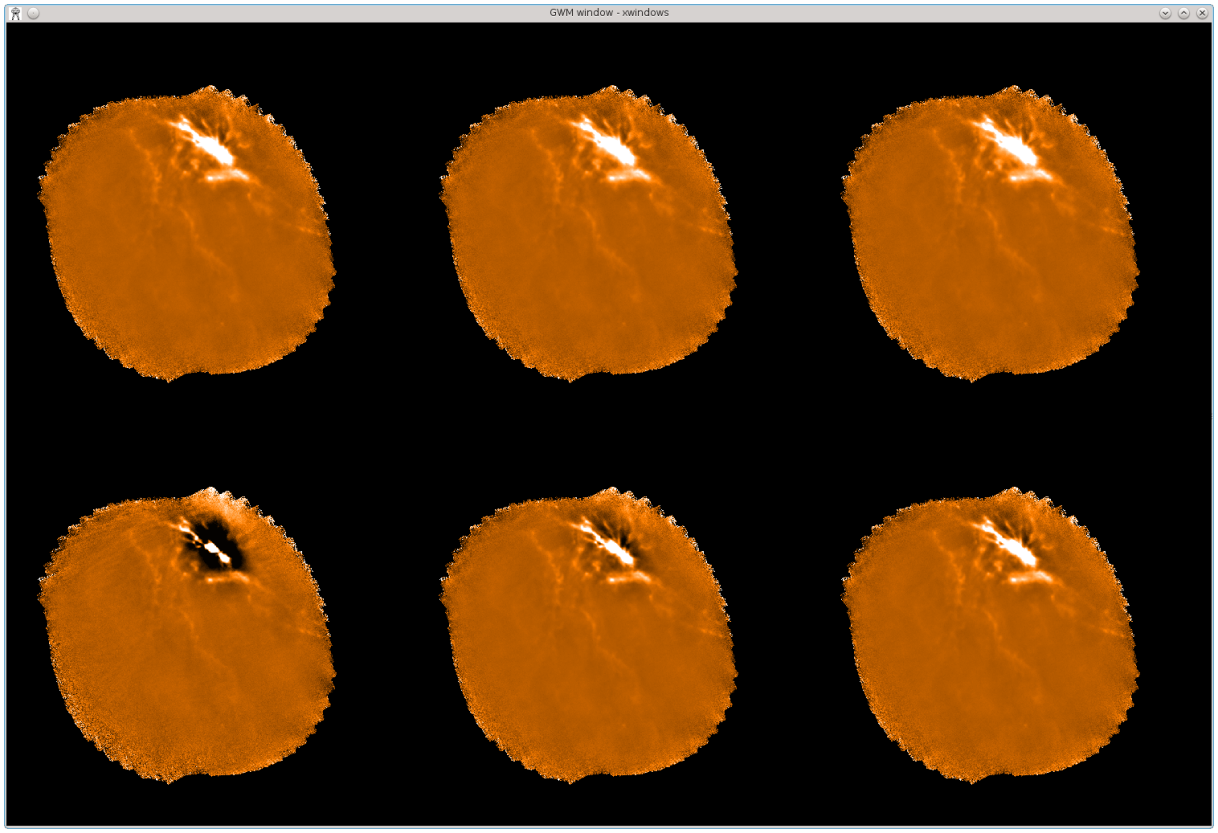


Figure 5.1: The maps created on the first six iterations of `makemap`, from an observation of Orion A, using the `dimmconfig_jsa_generic.lis` configuration file with the addition of "`itermap=1`". The first map is bottom left, and the sixth map is top right. This illustrates the initial fast improvement in the map over the first few iterations.

If you want to view several itermaps side-by-side, you can use KAPPA picgrid to divide the screen up into a grid of "pictures", then use picsel to pick each picture in turn, and use display to display an itermap in each one. For instance to display the first six itermaps side-by-side in a 3×2 grid, all with the same scaling, without any axes, do:

```
% gdclear
% picgrid 3 2
% picsel 1
% display myitermaps.CH00I001 axes=no mode=percentiles percentiles=\[2,98\]
% picsel 2
% display myitermaps.CH00I002 mode=current
% picsel 3
% display myitermaps.CH00I003 mode=current
% picsel 4
% display myitermaps.CH00I004 mode=current
% picsel 5
% display myitermaps.CH00I005 mode=current
% picsel 6
% display myitermaps.CH00I006 mode=current
```

This produces the display shown in Figure 5.1. KAPPA display has many options for controlling things like data scaling, axis annotation and style, colour table, graphics device, *etc.*.

Alternatively, you can combine the itermaps into a 3D cube, and then use the cube-viewing facilities of GAIA to scroll through the itermaps in the style of a movie (see Section 6.2)[10]:

```
% paste in=myitermaps out=itercube shift=\[0,0,1\]
% gaia itercube
```

It is sometimes informative to look at the *change* between itermaps, to see the incremental effect of each iteration. This is most easily done if the itermaps are first stacked into a cube as described above. The next step is to produce a copy of the cube in which the pixel origin is moved by one pixel along the third axis (*i.e.* the axis that enumerates iteration). Finally subtract one cube from the other, and display the resulting difference cube (this example shows how command-line parameters can often be specified by position instead of by name when running KAPPA commands—check the help information for each command to see the order in which options should be supplied):

```
% paste myitermaps out=itercube shift=\[0,0,1\]
% slide itercube itercube-shifted \[0,0,1\] near
% sub itercube-shifted itercube diffcube
%
% picsel 1
% display diffcube'(,,1)' axes=no mode=percentiles percentiles=\[2,98\]
% picsel 2
% display diffcube'(,,2)' mode=current
% picsel 3
% display diffcube'(,,3)' mode=current
% picsel 4
% display diffcube'(,,4)' mode=current
% picsel 5
% display diffcube'(,,5)' mode=current
% picsel 6
% display diffcube'(,,6)' mode=current
```

This produces the display shown in Figure 5.2.

Viewing the mask after each iteration:

The above plots are based on the itermaps that are created by adding "`itermaps=1`" to your `makemap` configuration. You can instead use "`itermaps=2`", which causes each itermap to be masked so that pixels outside the current source mask are set bad (*i.e.* blank)[11]. Figure 5.3 shows the resulting itermaps.

The main point of this section is to emphasise that all the above investigations can be performed *whilst makemap is still running*, if you specify an explicit file for the itermaps using the "`itermaps` command-line option. This allows you to decide if the reduction is progressing as expected, and whether to interrupt the reduction.

5.3.3 Interrupting `makemap`

If you interrupt makemap by pressing control-C on the keyboard (or equivalently by sending an `INT` signal to the `makemap` process), it will continue until the current iteration is completed and then will issue a prompt with options allowing you to save the current map before closing, as in the following example:

```
% makemap in=^infiles out=outmap config=^conf itermaps=myitermaps
Out of 29 input files, 1 was a dark, 1 was a fast flat and 27 were science
Processing data from instrument 'SCUBA-2' for object 'OMC1 tile10' from the
following observation  :
...
...
```

[10] Note, SMURF stackframes can be used instead of KAPPA paste if preferred.

[11] If no masking is being used, then `itermaps=2` has the same effect as `itermaps=1`.

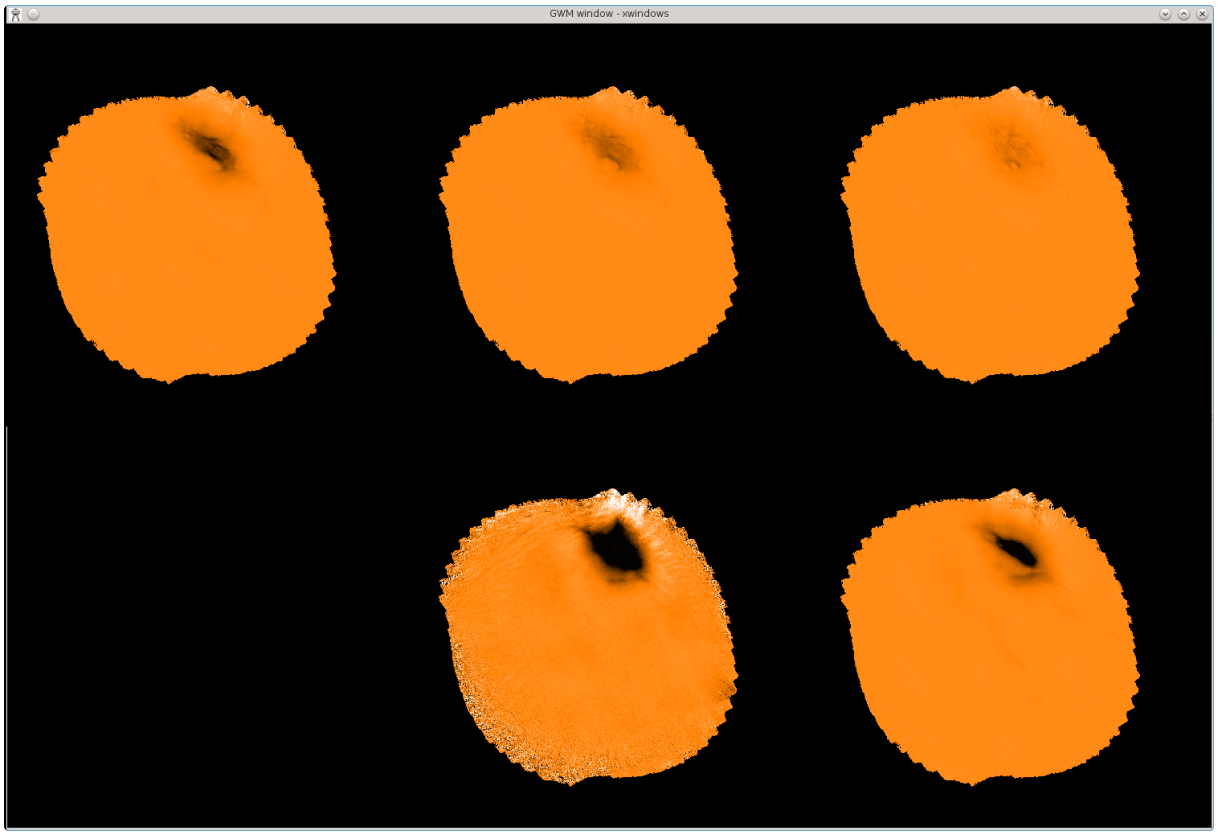


Figure 5.2: The difference between each pair of adjacent maps shown in Figure 5.1.

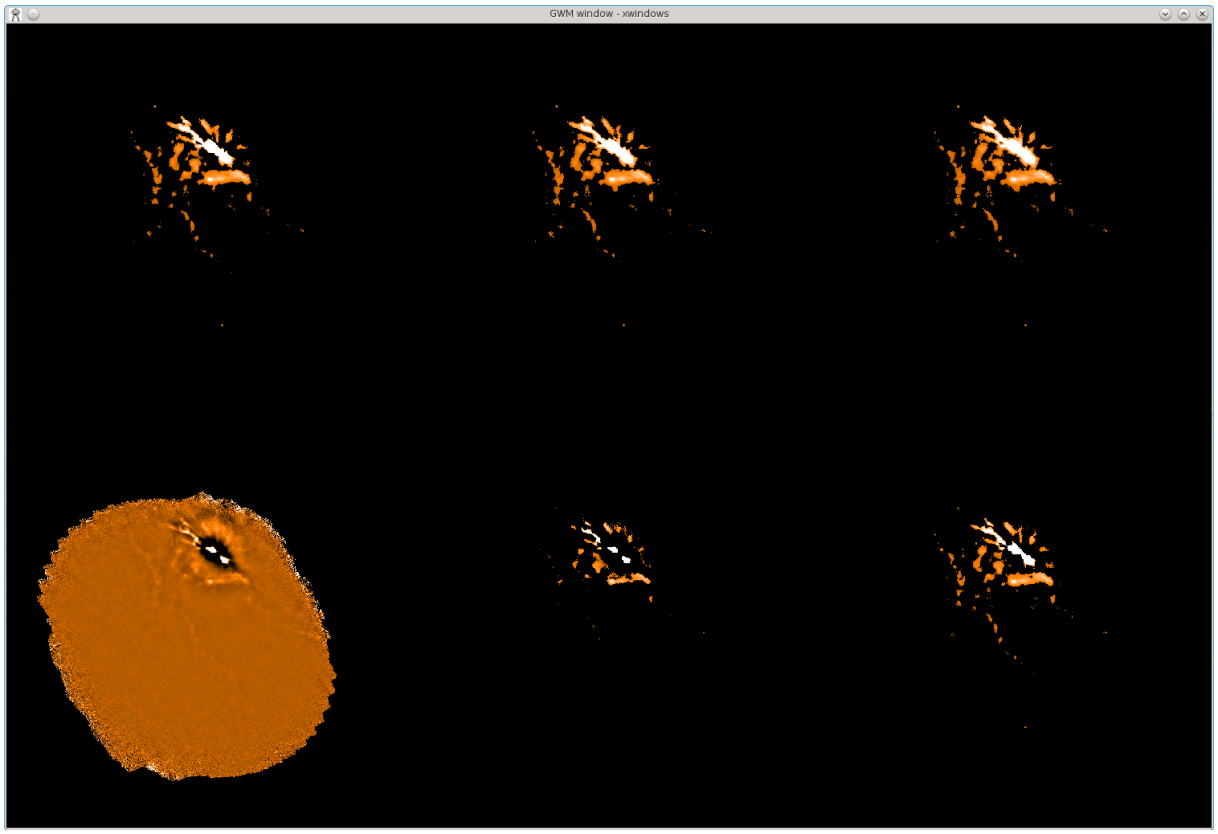


Figure 5.3: The first six itermaps, create by using `itermap=2` in the configuration. This causes pixels outside the source mask to be set blank, allowing the evolution off the mask to be seen (the first iteration is not masked). In this case, the mask includes all pixels above a signal-to-noise ratio of five.

```
<press control-C>
```

```
...
...
--- Quality flagging statistics
-------------------------------------------
 BADDA:   25525734 (20.86%),         267 bolos  ,change          0 (+0.00%)
BADBOL:   31261854 (25.55%),         327 bolos  ,change          0 (+0.00%)
 SPIKE:        339 ( 0.00%),                    ,change          4 (+1.19%)
DCJUMP:     146703 ( 0.12%),                    ,change          0 (+0.00%)
  STAT:     160000 ( 0.13%),         125 tslices,change          0 (+0.00%)
   COM:    1827454 ( 1.49%),                    ,change       3000 (+0.16%)
 NOISE:    5640518 ( 4.61%),          59 bolos  ,change          0 (+0.00%)
Total samples available for map:   89135553, 72.84% of max (932.361 bolos)
     Change from last report:      -3001, -0.00% of previous
smf_iteratemap: *** CHISQUARED = 0.998314983615372
smf_iteratemap: *** change: -0.0014711436109287
smf_iteratemap: *** NORMALIZED MAP CHANGE: 0.00941492 (mean) 2.9521 (max)


>>>> Interrupt detected!!! What should we do now? Options are:
1 - abort immediately with an error status
2 - close the application returning the current output map
3 - do one more iteration to finalise the map and then close

NOTE - another interrupt will abort the application, potentially leaving
files in an unclean state.

INTOPTION - What to do now (1-3) /3/ >
```

At this point you should respond to the prompt for parameter `INTOPTION` by typing 1, 2 or 3 followed by `<return>`. Option 1 aborts the process with an error status. Option 2 causes `makemap` to tidy up its internals and create a map from the current models just as if the iterative process had reached convergence. Option 3 causes one further iteration to be performed, without masking before creating a final map (you should use this option if you have been using AST masking—see Section 3.5.1).

Tip

Note if you are running `makemap` from within a script, then the handling of control-C interrupts will probably be quite different, because the shell process (or perl, python or whatever) will catch the interrupt before it gets to the `makemap` process. This usually causes the shell process to die but leaves the `makemap` process running in the background. Instead of pressing control-C on the keyboard, you should find the process ID for the `makemap` process itself, and then send an INT signal to that process explicitly:

```
% ps aux | grep makemap
dsb      25407  0.0  0.0  43716  1952 pts/5    0:00 makemap
% kill -s INT 25407
```

5.4 Tips and tricks

5.4.1 Aligning your map with a pre-existing image

If you want to compare SCUBA-2 data with a map of the same region taken with a different instrument, you will usually want to create the SCUBA-2 map using the same pixel grid as the other map so that you can compare pixel values directly in the two maps[12]. You can use the REF command-line parameter to do this. For instance, if your other map is in file herschel.sdf, you can do:

```
% makemap in=^infiles out=outmap config=^conf ref=herschel
```

This will cause the output map in file outmap.sdf to use the same pixel grid as herschel.sdf. This means that a given point on the sky will have the same *pixel coordinates* in both maps, and so for instance you could divide one by the other to get a ratio map without any further alignment step:

```
% div outmap herschel ratio_map
```

This will divide outmap.sdf by herschel.sdf and put the ratio in ratio_map.sdf[13].

However, by default makemap will trim the output map to exclude blank borders, thus the two maps may have different *dimensions*. NDF-based applications such as KAPPA div automatically take account of this when comparing corresponding pixels in two NDFs—see Figure 5.4.

If you want to force the output map to have certain bounds in pixel coordinates, you can do so by assigning values to the LBND and UBND parameters when running makemap. For instance, if you want to force the output map to have the same dimensions as the reference map, you could use KAPPA ndftrace to list the properties of the reference NDF, including its pixel bounds, and then assign these bounds to LBND and UBND parameters when running makemap:

```
% ndftrace herschel

   NDF structure /home/dberry/herschel1:
      Title:  Example Herschel data

   Shape:
      No. of dimensions:  2
      Dimension size(s):  351 x 400
      Pixel bounds     :  -50:300, 1:400
      Total pixels     :  140400
...
...
% makemap in=^infiles out=outmap config=^conf ref=herschel \
          lbnd=\[-50,1\] ubnd=\[300,400\]
```

The resulting map in outmap.sdf will have the same pixel bounds as hershel.sdf.

[12] Creating the map on the required pixel grid is better than creating it on a default grid and then re-sampling it onto the required grid later.

[13] In reality you would probably want to take account of differing resolutions and /or units in the two maps before dividing them.

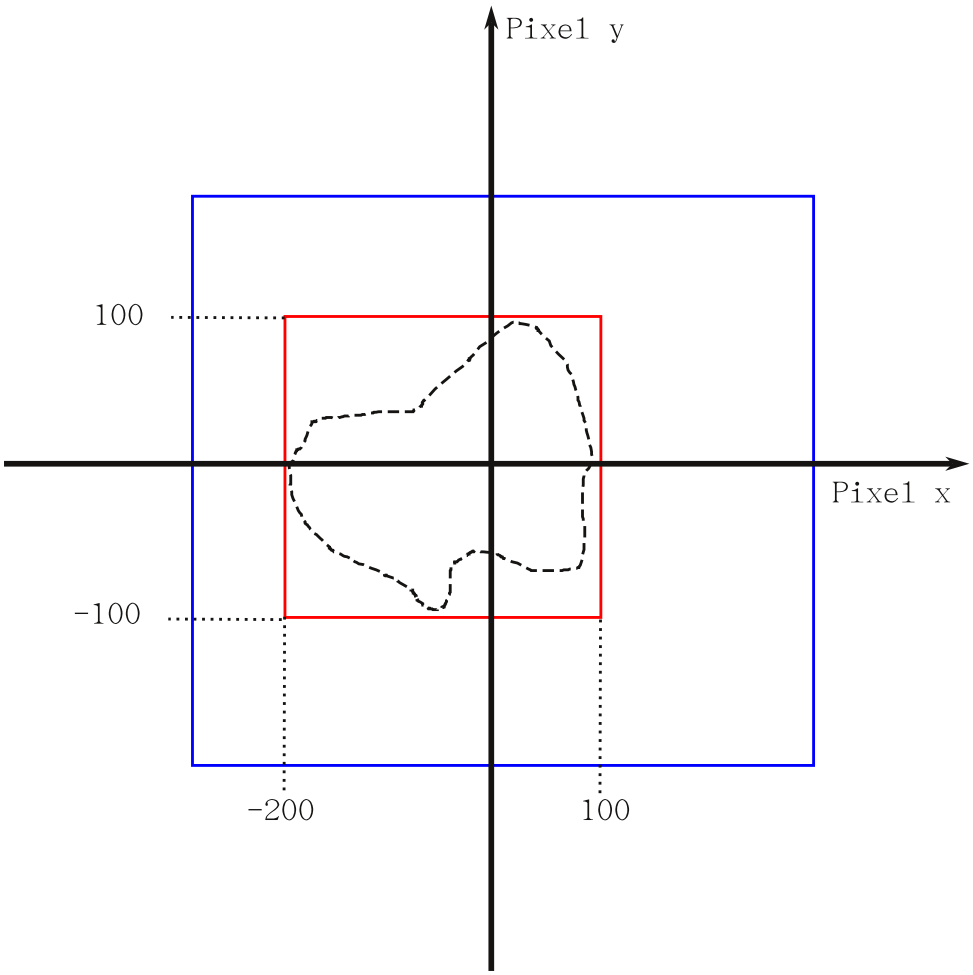


Figure 5.4: The curved dotted line represents a contour of an astronomical object. The heavy black lines represent the X and Y axes of the pixel coordinate system, crossing at the pixel origin (`pixel_x`,`pixel_y`)=(0,0). The numerical values represent pixel X and Y values. The blue box represents the bounds of the reference map supplied when running makemap. The red box represents the bounds of the map created by makemap. The reference image and the output image share the same pixel coordinate system, so a given point on the sky has the same pixel coordinates in both images. However, the output (red) image is cropped to remove blank borders and so has smaller dimensions than the reference (blue) image. All this is made possible by the fact that the NDF format, unlike FITS, does not require the origin of pixel coordinates to be at the bottom left corner of each map. NDF applications account for any difference in pixel origin when comparing pixels in different files.

5.4.2 Limiting the amount of memory used by `makemap`

There may be occasions when you need to limit the amount of memory used by makemap–for instance to leave enough memory for other processes to function correctly. For machines with more than 20 GB of memory, the default is to leave 4 GB free for other processes. For machines with less than than 20 GB of memory, the default is to leave 20% of the total memory free. This default can be over-ridden by indicating the maximum amount of memory that `makemap` should use by setting a value for the command-line parameter `MAXMEM`. For instance:

```
% makemap in=^infiles out=outmap config=^conf maxmem=50000
```

will ensure that `makemap` never uses more than 50000 MiB[14] of memory. Be aware though that for larger observations this may cause the data to be processed in multiple chunks rather than a single continuous chunk, resulting in a poorer map.

5.4.3 Re-using previously cleaned data to speed up map-making

Usually, you will create your SCUBA-2 maps using one of the standard configuration files distributed with SMURF (see Section 3.7). However, for certain problematic observations it may sometimes be necessary to experiment with several different configurations to find one that produces an acceptable map. In such cases a significant amount of time can be saved by re-using pre-cleaned raw data each time you make a map, rather than re-cleaning it each time.

By default makemap assumes that the supplied data are raw data that require cleaning before being used to make a map. However, if you include "`doclean = 0`" in your configuration, then `makemap` will skip the entire cleaning process and move directly on to the iterative map-making stage, thus saving significant time.

Obviously though, in this case you need to make sure that the data you supply to `makemap` have already been cleaned. There are two ways to get cleaned data:

(1) You can run `makemap` initially on the *un*-cleaned data (*i.e.* the original raw data), and force `makemap` to dump the cleaned data to disk before going onto the iterative map-making stage. You can then re-use this cleaned data in subsequent invocations of `makemap`. To dump the cleaned data, you need to add "`exportclean = 1`" to the configuration on your initial run of `makemap`—don't forget to remove it for subsequent runs! The cleaned data are put into files in the current directory, one for each sub-array, with names such as `s8c20120706_00037_0003_con_res_cln.sdf`. Note the `_cln` on the end of the name indicating cleaned data. The base name, "`s8c20120706_00037_0003`" in this case, is taken from the first raw data file that contributes to each chunk. If the entire observation is processed in one chunk (as it should usually be) then there will be one such `_cln` file for each sub-array. If the observation is split into multiple chunks, then there will be multiple `_cln` files for each sub-array (one for each chunk).

(2) You can use the separate SMURF task sc2clean to clean the data as described in Appendix A.

[14] One MiB is 1048576 bytes.

Chapter 6
Tailoring Your Reduction

6.1 Adding and amending parameters

The configuration file `dimmconfig_jsa_generic.lis` is a good configuration file to use as a first pass for reducing any data, and is the default configuration used by the pipeline's `REDUCE_SCAN` recipe.

You can create your own personalised configuration file from scratch or copy one of the provided ones to your local directory and edit it.

The first line of each specialised configuration file is often the path to another configuration file—known as the *parent* configuration file. The new configuration file inherits all the parameter values defined by the parent configuration file, and then goes on to specify additional parameter settings which may supplement or override those defined in the parent configuration file. Parameters that are not set to a specific value by *either* configuration file assume the default values listed in Appendix I[1].

Often you will want to use one of the pre-defined configuration files included within Starlink tree as the parent configuration. Remember that any parameters appearing in your configuration file automatically override the values supplied by the parent file. As an example, consider the following text file "`myconf`":

```
% cat myconf

#  This is an example configuration file.
^$STARLINK_DIR/share/smurf/dimmconfig_jsa_generic.lis

numiter=-100
maptol=0.01
```

The first thing to note is that blank lines and comment lines (*i.e.* lines beginning with the hash character "`#`") are ignored and can be used to document your configuration file. Next note that this example file uses `$STARLINK_DIR/share/smurf/dimmconfig_jsa_generic.lis` as its parent (the path to the parent file must be preceded by an up-caret (`^`) character). Thus, all the parameter values set by `$STARLINK_DIR/share/smurf/dimmconfig_jsa_generic.lis` are first read, before adding in other parameter settings. In this case, values are assigned to `maptol` and `numiter`, overriding the values provided by the parent file.

You can use more than one parent file if required (each one a separate line, and preceded by an up-caret). Each specified parent will be read in turn, with parameter settings read from later ones having priority over those read from earlier ones.

You can also add or amend parameters by listing them directly on the command line. They are appended to the configuration file name as a comma separated list as shown in the example below. Be sure to include all the necessary quotation marks.

```
% makemap in='s8*.sdf' out=850map \
       config='"^dimmconfig_jsa_generic.lis,numiter=-50,exportndf=(flt,noi),itermap=1"'
```

[1] These default values are defined in the file `$SMURF_DIR/smurf_makemap.def`.

For full details of all the possible ways of specifying groups of parameter values, see Section "Specifying Groups of Objects" in "Starlink User Note 95".

What parameters can be changed?
Appendix I lists the parameters that are more likely to be of interest to you when creating your own configuration files. You can also change any of the other more esoteric parameters not included in that list—see Appendix SUN/258 within Starlink User Note 258 for a full list—but we do not advise this.

> **Tip**
>
> If some feature is switched on either by default or within the parent configuration file, it can be switched off if required by assigning a suitable value to the corresponding parameter within your own configuration file. The value needed to do this will be given in the parameter description in Appendix I—for instance it may be `<undef>`, or `0`, or some other special value.

Note: any parameter can be made wavelength dependent by adding the prefix `450.` or `850.`, e.g. `flt_edge_largescale` applies to both 450 μm and 850 μm whilst `450.flt_edge_largescale` applies to 450 μm only. Be aware that if both are specified, unqualified values (no prefix) take priority over qualified values.

6.2 Writing out models & intermediate maps

itermaps
Setting the parameter `itermap = 1` writes out the map containing the astronomical signal after each iteration. Setting `itermap = 2` adds the QUALITY component (see Section 8.6. These can be visually inspected with

```
% gaia 850map.more.smurf.itermaps
```

to help determine an appropriate number of iterations. Alternatively, you can view several itermaps simultaneously side-by-side (*e.g.* Figure 5.1) using KAPPA as described in Section 5.3.2.

Viewing itermaps is useful when a fixed number of iterations have been requested (i.e. a positive value for `numiter`) and the map solution diverges before they have completed. See also Section 5.3.2 for how to view these maps whilst makemap is still running.

shortmaps
If the parameter `shortmaps` is non-zero, a map is made from every group of adjacent time-slices. When set to -1, a map is produced each time a full pass through the scan pattern has been completed. Any other negative value is interpreted as a duration in seconds, and is converted to time slices using the (possibly down-sampled) sample frequency of the data being mapped. These are stored as an NDF extension and can be viewed GAIA.

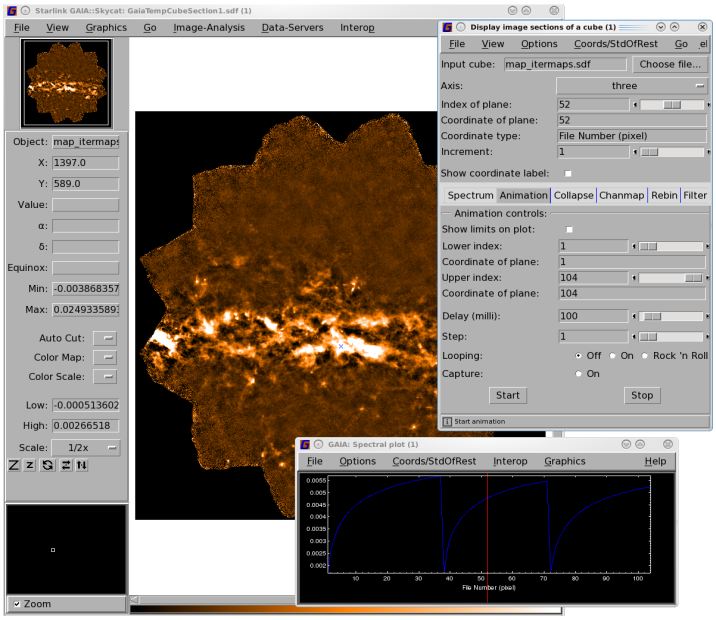
```
% gaia 850map.more.smurf.shortmaps
```

You can view the shortmaps and itermaps more conveniently by stacking them into a single cube using the SMURF command stackframes. This cube can then be viewed as a 'movie' with GAIA, using the animation option to loop through the itermaps. See Figure 6.1 for instructions.

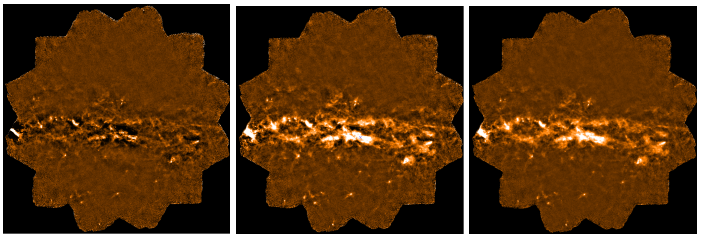
Viewing ITERMAPs

```
% stackframes map.more.smurf.itermaps \
sort=false map_itermaps
```

Stack the individual itermaps into a single cube (KAPPA PASTE can also be used).



The output map `map_itermaps` can be opened with GAIA. The data used in this example is the Galactic map reduced in Section 7.2. The Spectral plot window shows the value for a single pixel and the three chunks are easily identified. You can select the **Animation** tab in the **Display image sections** window and click **Start** to loop through the itermaps for each iteration. The 'movie' will appear in the main GAIA window.



These windows show the itermaps map at 1, 10, and 30 iterations. A specific iteration can be selected using the **Index of plane** slider on the **Display image sections** window.

Figure 6.1: Example using the SMURF command stackframes and GAIA to view the 'itermap' for each iteration.

exportmodel
This parameter has been discussed in Section 9.8 and allowed you to see the model that was fit for each component specified by the `modelorder` parameter.

6.3 Large-scale filtering

Some of the most important parameters to experiment with are the filtering options. By default, no filtering is applied during the pre-processing stage[2] (see `filt_edge_largescale` and `filt_edge_smallscale`).

[2]However, the `dimmconfig_blank_field` configuration overrides this default.

A high-pass filter is used during the iterative stage if `modelorder` includes `FLT`[3], and selected a suitable value for the filter size is crucial for maps containing extended emission. See Mairs et al. 2015 [15] for more information about extended emission recovery, wherein model sources with known parameters were added to the raw SCUBA-2 timestream using "`fakemap`" in order to investigate the effects of large-scale filtering.

The maximum spatial scale of structure that can be recovered by the map-maker is determined by the scanning speed and frequency cut applied to the data:

$$\frac{\text{speed}[arcsec/\text{s}]}{\text{frequency cut}[\text{Hz}]} = \text{scale size}[arcsec] \tag{6.1}$$

The default (*i.e.* if no configuration is supplied) filter sizes in arcseconds are:

```
450.flt.filt_edge_largescale = 200
850.flt.filt_edge_largescale = 480.
```

To make your life easier, these parameters allow you to specify the filter limits in terms of spatial scale in arcseconds—in this case 480 arcsec at 850 μm and 200 arcsec at 450 μm. For example, at 850 μm, recovering scales of 480 arcsec at a scan speed of 600 arcsec/sec (default for a 1 degree PONG) corresponds to a frequency of 1.25 Hz.

Choosing a high-pass filter is especially important for the recovery of extended emission.
The `dimmconfig_bright_extended` configuration file sets `flt.filt_edge_largescale = 480` for both 450 μm and 850 μm. Be aware that increasing filter sizes decreases the flatness of your background. A compromise must be made between extended structure and the flatness of your map. See Figure 6.2 for an illustration of the effect of `flt.filt_edge_largescale` on your map.

The scanning speeds are fixed for a given observing mode; you can find out the speed at which your data were taken from the `SCAN_VEL` keyword in the FITS header (see Section 9.2).

Flattening the background
There is an option to reduce the noise in your background introduced by setting a high value for the large-scale filter. The parameter `flt.filt_edge_largescale_last` filters the regions outside your `AST` mask (see Section 3.5.1) on a shorter scale for the last iteration only, thereby producing a much flatter background. Note that this is probably a bad idea if you intend to co-add several observations, as it removes potentially real structure in the background regions that could otherwise be recovered by co-adding several observations. In general, use of `flt.filt_edge_largescale_last` should be seen as a cosmetic enhancement, since it results in differing filter sizes being used inside and outside the `AST` mask.

Also note that when using `flt.filt_edge_largescale_last` the variances stored in the final map are from the penultimate iteration in order to avoid using the artificially reduced variances created on the last iteration.

6.4 Fitting COM for each sub-array

A useful option to improve the flatness of your maps is to fit the `COM` model independently for each sub-array. This is particularly effective if you find you have one sub-array noisier than the others.

This comes with the warning however that you will lose information on any scales larger than the area covered by a single sub-array ($\sim 200''$). It is therefore not recommended if you have very large-scale extended structure.

To initialise this option set `com.perarray = 1`.

[3] The `dimmconfig_blank_field` configuration omits `FLT` from `modelorder`.

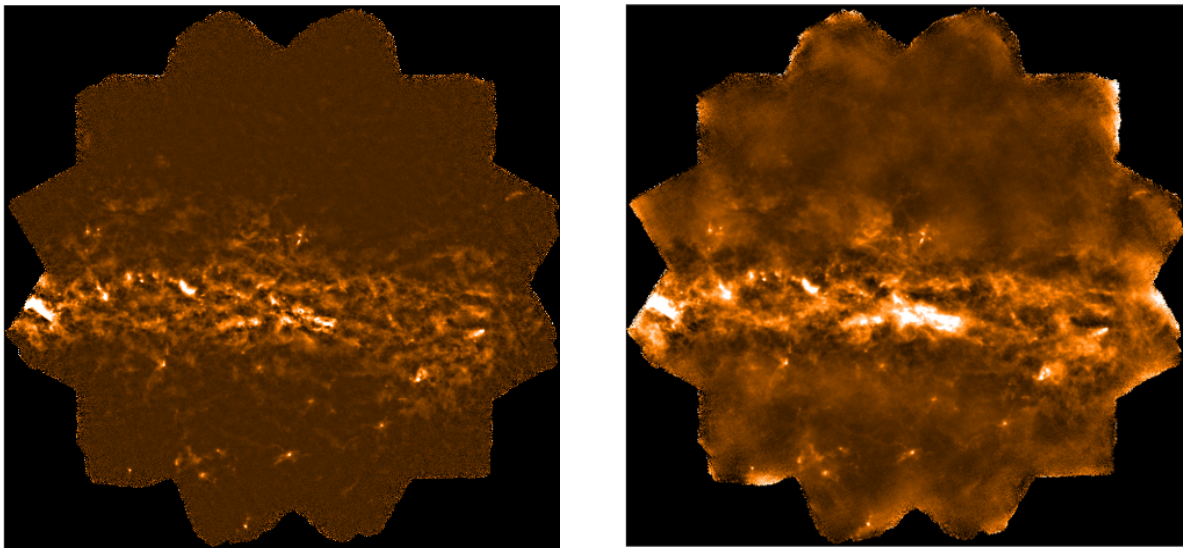


Figure 6.2: Highlighting the effects of high-pass filtering on your map. **(Left)** Map made with `850.flt.filt_edge_largescale` = 300. **(Right)** Map made with `850.flt.filt_edge_largescale` = 1000. All other configuration parameters remain the same.

6.5 Flagging bad data

Bolometers that have higher noise levels are down-weighted when forming the map. By default, a separate noise estimate is created for each 15 second section[4] of each bolometer time-stream, but an alternative length can be specified using parameter `noi.box_size`.

Dividing each bolometer up into sections helps to remove 'scuffs' or other noise artefacts you might see in your error map due to a sub-array (or arrays) temporarily jumping to a higher noise state. As the section length tends to 1 second we find some of the source signal being down-weighted. Higher than this 15 seconds and the map-maker becomes less sensitive to the higher noise states.

Other parameters you may want to try include `flagfast` and `flagslow`. You may find that setting `flagfast` to less than the default of 1000 arcsec/sec will help reduce the effect of any 'smearing' of sources (and of noise) in maps, while setting `flagslow` greater than the default of 300 arcsec/sec helps to flatten the edges of maps. To determine reasonable values for your data-set you should do jcmtstate2cat and view the scan speed using TOPCAT. See Section 9.3 for details.

6.6 Using external masks

For an introduction to the purpose and effects of masking, see Section 3.5 and Mairs et al. (2015) [15].

As an S/N mask is redetermined after each iteration it changes with the map which can sometimes cause convergence problems. The mask will also depend on the amount of data going into the map and the pixel size. A fixed externally supplied mask can get around these problems.

The sequence below is a summary of the procedure for generating and supplying an external mask. In this example the mask is generated from the map produced by an initial run through the map-maker.

[4] 15 seconds is half a sub-scan.

Alternatively, SNR maps or flux contours calculated from data obtained by other observatories can be used.

These steps are followed in the example in Section 7.2.

Step 1 Generate a map covering your region. This may be by simply running the map-maker on your data as shown below.

```
% makemap in='s8*.sdf' out=850map \
          config='"^dimmconfig_bright_extended.lis"'
```

The alternative is to access a map from a different data-set or even a different telescope, e.g. a map downloaded from the Herschel Science Archive. For instructions on converting from FITS to NDF see Appendix H.

Step 2 Make a signal-to-noise map using the KAPPA command makesnr.

```
% makesnr 850map 850map_snr
```

Step 3 Threshold this S/N map to set everything below 3σ to 0 and everything above to 1.

```
% thresh 850map_snr 850map_mask thrlo=3 newlo=0 thrhi=3 newhi=1
```

This generates a mask which has an unrealistic hard 3σ cut-off. Step 4 is performed to smooth the the edges of your mask.

Step 4 Smooth the thresholded map with a Gaussian filter of FWHM of 5 pixels (= 20 arcsec at 850 μm, using the default pixel size). Then it is again thresholded, this time keeping everything above 5 % of the 0 level as the mask and setting the rest to bad.

```
% gausmooth 850map_mask 850map_mask_sm fwhm=5
% thresh 850map_mask_sm 850map_mask_zm thrlo=0.05 newlo=bad \
  thrhi=0.05 newhi=1
```

Step 5 Finally the map is re-made with this mask supplied as an external file. Notice that the extra parameters required to pick up this external mask are being appended to the configuration file on the command line rather than editing the file itself.

```
% makemap in='s8*.sdf' out=850map_zm ref=850map_mask_zm \
          config='"^dimmconfig_bright_extended.lis,ast.zero_mask=1,\
                  ast.zero_snr=0"'
```

6.7 Skyloop

Traditionally, the map-maker divides a non-contiguous sequence of time-series data into chunks. It processes each chuck independently before co-adding them as a final step in the the reduction—see Figure 6.3.

This means for each chunk the map-maker has to start from scratch determining the AST model, and the benefit of long integration times spent building up the signal is lost. Configuration files that use signal-to-noise masks especially suffer from this approach as the signal-to-noise in each individual chunk can remain low and fainter extended structure is not recovered.

The skyloop command is a script that runs makemap multiple times performing just a single iteration on each occasion. It starts by performing a single iteration of makemap from which a co-added map is generated. This map is then supplied as an initial estimate of the sky for the next invocation of makemap.

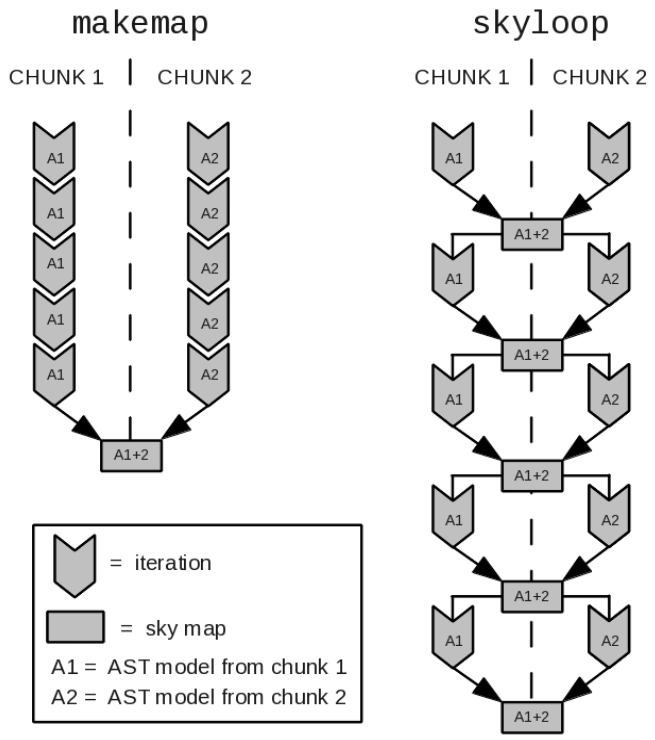


Figure 6.3: Illustration of the skyloop approach to map-making compared with the standard map-maker.

On this next invocation, the initial sky estimate is subtracted from the cleaned time-series data and the `COM`, `GAI`, `FLT`, `EXT` models are subtracted. This produces a new model of the sky (from the current iteration) to which the sky estimate (from the previous iteration) is then added. In this way the signal from all of the chunks is built up over the iterations and is all included in the final map estimate when convergence is reached.

Be aware that skyloop uses a lot of disk space. Setting environment variable `STAR_TEMP` to a suitable location before you start will prevent skyloop from crashing if you run out of temporary storage space.

```
% setenv STAR_TEMP /some/directory/with/a/lot/of/space
```

skyloop can then be called in a way very similar to makemap, with a configuration file specified on the command line.

```
% skyloop in=^myfiles.lis out=map_skyloop config=^dimmconfig_bright_extended.lis
```

6.8 Troubleshooting

PROBLEM	**POSSIBLE SOLUTION**
I have blobs in my map that look like big thumbprints.	Try adding `dimmconfig_fix_blobs` into your configuration file (see Section 3.8). Note that this sets `com.sig_limit = 5` which is somewhat conservative. You can experiment by lowering this, but we would not recommend lower than ∼2 because you may then lose too many samples. In addition, check you are not using `flt.notfirst = 1` as this can make blobs worse.
I want to recover more extended structure.	There is a trade off between extended emission and noise in your map. If you are willing to accept more low frequency noise you can increase the filter scale with `flt.filt_edge_largescale`. The default is 480 arcsec but you could try 600 arcsec. To reduce the increased background noise you can set `flt.filt_edge_largescale_last = 200`. This sets the background filtering to 200 arcsec for the final iteration only, though you can go as low as you want with it. Note, this should be seen as a mainly cosmetic effect as it causes the map to use different filter sizes in different regions, making interpretation of the map difficult.
I want a flatter background.	Try `com.perarray = 1`, although be aware this will lose structure on scales larger than a sub-array. If you are chasing extended emission see the point above. For a more uniform background set `flt.filt_edge_largescale_last` to a small value to get harsh filtering on your final iteration.
I have linear striations in my map making my background look scratchy.	Try setting `com.corr_abstol = 0.8` [default=0.2]. This rejects more bolometers with deviant common-mode signals. However, as more bolometers are removed there are fewer data available for your final map, resulting in higher noise.
My map will not converge.	Try adding `dimmconfig_fix_convergence` into your configuration file (see Section 3.8). This prevents the masks from changing after ten iterations.

Chapter 7
Examples of Different Reductions

7.1 Deep point-source maps

The science goal of many extra-galactic SCUBA-2 observations is to detect unresolved point sources. In the examples below we work through the reduction of just such an extra-galactic field, A1835.

Most extra-galactic objects are on average only slightly brighter than the confusion limit. The fluctuations of the background sky brightness due to multiple super-imposed, unresolved sources within the telescope beam, below which individual sources cannot be detected. It is likely that any sources in the map will be at best, only a few standard deviations brighter than the noise in the map (caused by a combination of instrumental noise and source confusion).

7.1.1 Example 1 – The simple reduction

The basic reduction method for maps like these follow two main steps: 1. running the data through the map-maker using the `dimmconfig_blank_field.lis` configuration file (see Section 3.7), and 2. applying the PICARD `SCUBA2_MATCHED_FILTER` recipe (see Section 8.7 and Appendix D).

Step 1: Run the map-maker
In this example the raw data are stored locally in a directory called `data`. We have three observations (#13, #18, #21) of the field, which we will reduce independently.

```
% makemap data/s8*00013_00\*.sdf cosmo1 \
          config=^$STARLINK_DIR/share/smurf/dimmconfig_blank_field.lis

% makemap data/s8*00018_00\*.sdf cosmo2 \
          config=^$STARLINK_DIR/share/smurf/dimmconfig_blank_field.lis

% makemap data/s8*00021_00\*.sdf cosmo3 \
          config=^$STARLINK_DIR/share/smurf/dimmconfig_blank_field.lis
```

Step 2: Combine the maps
These three maps are then combined using the PICARD recipe `MOSAIC_JCMT_IMAGES`. In this case we accept the default of wcsmosaic mosaicking and nearest-neighbour pixel spreading and so do not supply a parameter file.

```
% picard MOSAIC_JCMT_IMAGES cosmo*.sdf
```

The output map, `cosmo3_mos.sdf` (automatically named for the last input file appended by _mos), is shown in the left-hand panel of Figure 7.1. The advantage of using the PICARD recipe over standalone KAPPA commands is that the exposure time is also propagated correctly to the output mosaic (it is stored in the `MORE.SMURF.EXP_TIME` extension).

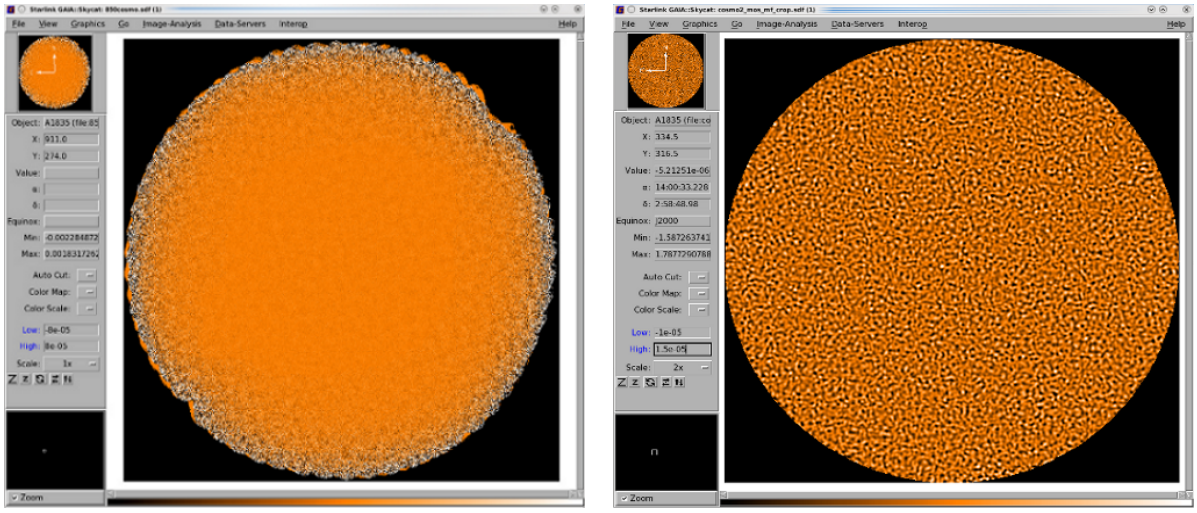


Figure 7.1: Reduced PONG maps of cosmology field A1835. **Left:** Map reduced with `dimmconfig_blank_field.lis`. **Right:** Map on the left after the matched filter has been applied and it has been cropped.

Step 3: Apply the matched filter
In order to optimally find sources that are the size of the telescope beam, we apply the matched filter recipe, namely `SCUBA2_MATCHED_FILTER`. We create a simple parameter file called `smooth.ini` containing the following lines:

```
[SCUBA2_MATCHED_FILTER]
SMOOTH_FWHM = 15
```

where `SMOOTH_FWHM = 15` indicates that the background should be estimated by first smoothing the map and PSF with a 15-arcsec FWHM Gaussian. Next, the recipe is executed as follows:

```
% picard -recpars smooth.ini SCUBA2_MATCHED_FILTER cosmo2_mos.sdf
```

The output of this operation is a smoothed image that is named by appending _mf to the input file (`cosmo3_mos_mf.sdf`) and a cropped version is shown in the right-hand panel of Figure 7.1. You can immediately see the contrast to the left-hand panel which is the output from the map-maker. A number of signal peaks now emerge as possible sources.

Step 4: Crop the map
Next we shall crop the map to remove the noisy edges, in this case to a 900-arcsec radius circle. The parameter file called `crop.ini` contains the following lines:

```
[CROP_SCUBA2_IMAGES]
CROP_METHOD = CIRCLE
MAP_RADIUS = 900
```

The output file from the following command will be named `cosmo2_mos_mf_crop.sdf`:

```
% picard -recpars crop.ini CROP_SCUBA2_IMAGES cosmo3_mos_mf.sdf
```

Step 5: Make an S/N map
Finally, we need to find sources. The filtered map contains a VARIANCE component, so it is easy to produce a S/N map using the KAPPA task makesnr:

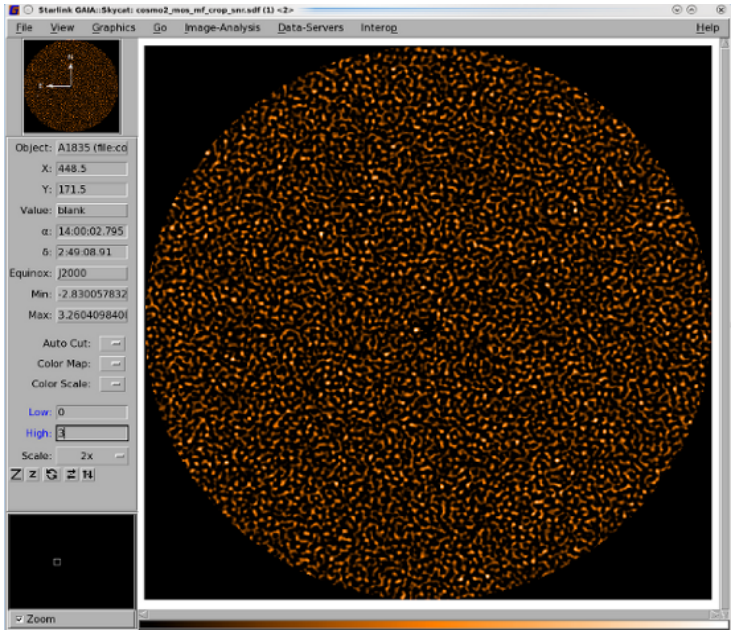


Figure 7.2: Signal-to-noise map made using the KAPPA command makesnr. The map has been scaled from 0 to +3.

```
% makesnr cosmo2_mos_mf_crop cosmo2_mos_mf_crop_snr
```

The resulting map, `cosmo2_mos_mf_snr`, is shown in Figure 7.2. Compared with the matched filter map the edges no longer appear as noisy because they have been down-weighted by the larger noise values where there were less data.

Step 6: Identify sources
The basic procedure for identifying sources would be to locate peaks above some threshold S/N. The S/N image above shows peaks that are likely to be real sources. For a start, a source appears where expected at the 0,0 position.

But how can we check if these sources are real?

- One option is to split your data into mutually exclusive subsets and produce independent maps. Are the highest S/N peaks detected in each of them?
- A second test is to compare the number of *negative* peaks above a given S/N with the number of *positive* peaks.

7.1.2 Example 2 – Advanced pipeline method (Recommended)

Although this method is considerably simpler to execute, the products have undergone more advanced processing than the manual method shown above. Due to these extra analysis steps, this pipeline method is particularly recommended.

Step 1: Create input file
Create a file with the full path names of all the files you wish to process, one per line (e.g. `myfiles.lis`)

Step 2: Run the pipeline
The pipeline must first be initiated for the wavelength you are working on. In the case below this is 850 μm. Note that the date does not *have* to be specified when initialising the pipeline. The pipeline is run using

the REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE recipe; this uses dimmconfig_blank_field.lis as the configuration file. If you wish to provide an alternative file you will need to put the name of the new configuration file in a recipe parameter file. See Chapter 4 for details.

```
% oracdr_scuba2_850 -cwd YYYYMMDD
% oracdr -files myfiles.lis -nodisplay -log sf FAINT_POINT_SOURCES_JACKKNIFE
```

You substitute the required date for YYYYMMDD. The pipeline will write out a large number of files with the following suffices.

sYYYYMMDD*_fmos	The map for each observation
sYYYYMMDD*_mappsf	The map for each observation with an artificial point source added at the map centre
gsYYYYMMD*_wmos	The co-add of all the _fmos files
gsYYYYMMD*_whiten	The whitened version of _wmos
gsYYYYMMD*_cal	The calibrated version of _whiten
gsYYYYMMD*_mf	The matched-filtered version of _cal

FAINT_POINT_SOURCES_JACKKNIFE is a recipe designed to process blank field/extra-galactic data. The recipe uses a jack-knife method to remove low-spatial frequency noise and generate a matched filter output map.

The recipe processes each observation twice, a standard reduction first, then a re-run with a fake point source added to the time series (see parameter "fakemap"). This produces a co-added signal map (_wmos) and a co-added PSF map (_mappsf).

Tip

The recipe name FAINT_POINT_SOURCES_JACKKNIFE can be used interchangably with REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE.

After the map-maker has completed, the recipe will call SCUBA2_JACKKNIFE. This routine divides the observations into two groups (odd and even) which are co-added and then subtracted to create a jack-knife map. This map contains only noise with no contribution from astronomical signal. The angular power spectrum of this map is then used to estimate and remove the residual $1/f$ noise from the signal map and the PSF map; this is the whitening step. The whitened jack-knife map is run through SCUBA2_MATCHED_FILTER using the whitened PSF map as the PSF input. It is this matched filter map which will be of most interest to users.

See **SUN/264** for more information on REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE and all other pipeline recipes.

Step 3: (Optional) Re-run SCUBA2_JACKKNIFE
You may wish to run the SCUBA2_JACKKNIFE step again independently from the pipeline. If your final map does not look as expected you might first examine the individual mosaics from the pipeline (_fmos), one of these observations might show visible artefacts that you wish to exclude from the co-add. The size of the region in the jack-knife image which is used to do the whitening step is determined automatically, but the method may fail if the box is too small.

If you decide to re-run this step you first co-add all the _mappsf files to create a co-added PSF using the PICARD recipe MOSAIC_JCMT_IMAGES.

```
% picard MOSAIC_JCMT_IMAGES *_mappsf
```

Next create a parameter file (`recpars.lis`) for the jack-knife recipe (`SCUBA2_JACKKNIFE`) containing the following lines.

```
[SCUBA2_JACKKNIFE]
PSF_MATCHFILTER = <name_of_above_co-added_PSF>.sdf
```

Another option for this parameter file is `WHITEN_BOX` to set the size of the region used to calculate the angular power spectrum. Finally run `SCUBA2_JACKKNIFE`.

```
% picard -log sf -nodisplay -recpars recpars.lis SCUBA2_JACKKNIFE *fmos.sdf
```

This will create files beginning with pg`YYYMMDD`... that should have the same suffices as above: `_wmos`, `_whiten`, `_cal`, and `_mf`.

7.2 Extended galactic sources

This example is concerned with recovering bright extended emission. The signal from extended emission varies slowly as seen by the array passing over it. It thus appears at lower frequencies in the power spectrum and complicates the high-pass filter selection. Too harsh a filter will make flat maps but any extended emission will have been removed in doing so (see Section 6.3 and Mairs et al. 2015 [15]).

Step 1: Running the map-maker

We run the map-maker using `dimmconfig_bright_extended.lis`; we have also specified a couple of overrides on the command line—`maptol` = 0.04 is slightly more stringent than default and `ast.zero_snr` = 3.5 constrains everything below 3.5 σ to zero (see also the `ast.zero_snrlo`) parameter.

In this example we give the map-maker a file containing a list of the input files (`filelist.txt`) and `dimmconfig_bright_extended.lis` is in the local directory.

```
% makemap in=^filelist.txt 850galactic \
          config='"^dimmconfig_bright_extended.lis,maptol=0.04,ast.zero_snr=3.5"'
```

The resulting map is shown in Figure 7.3.

Step 2: Generating an external mask

Next we create an external mask from the output of makemap. Here we follow the steps outlined in Section 6.6.

```
% makesnr 850map 850map_snr
```

This S/N map is thresholded to set everything below 3 σ to 0 and everything above to 1.

```
% thresh 850map_snr 850map_mask thrlo=3 newlo=0 thrhi=3 newhi=1
```

The thresholded map is shown in the left-hand panel of Figure 7.4. The next step is to smooth this map by convolving it with a Gaussian of 16 arcsec. For this we use a factor of 4 (pixels) for the FWHM parameter (since the default pixel size at 850 μm is 4 arcseconds).

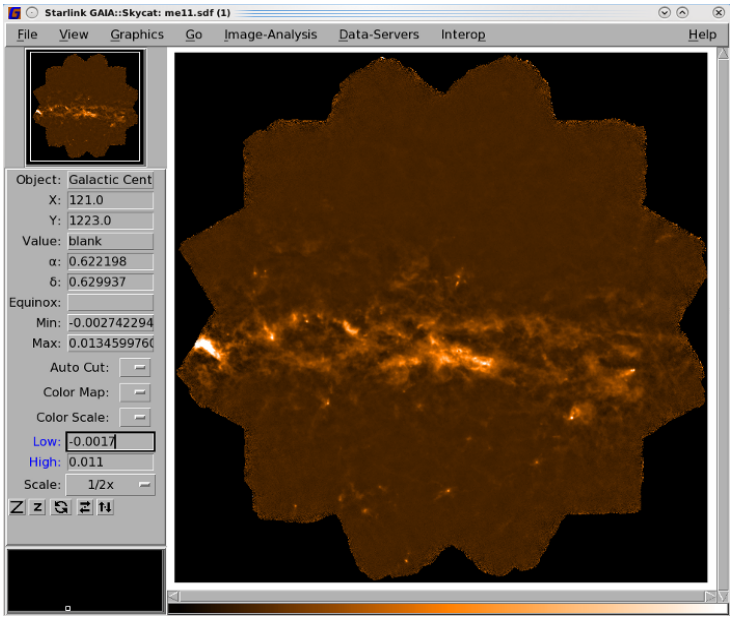


Figure 7.3: The output from the map-maker using `dimmconfig_bright_extended.lis`.

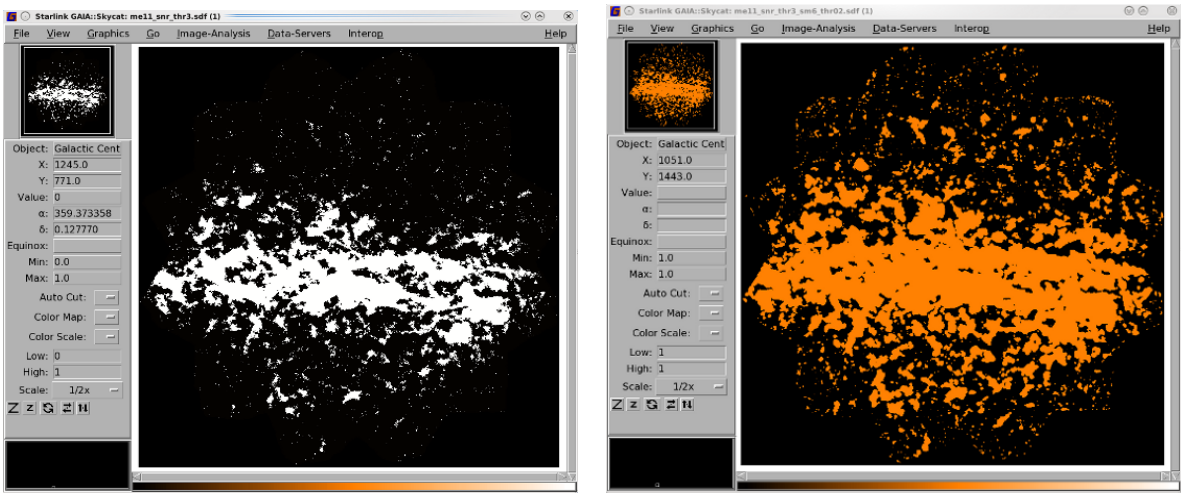


Figure 7.4: **(Left)** The initial mask created by thresholding 850map_snr to 3σ. **(Right)** Second mask made by thresholding the smoothed map to 0.02.

```
% gausmooth 850map_mask 850map_mask_sm fwhm=4
```

We threshold the map again to produce our mask. In this case all values below our threshold are set to 'bad'. The smoothed map now has values scaled between 0 and 1, we set our threshold at 0.02 to include more of the emission beyond the $3\,\sigma$ edge.

```
% thresh 850map_mask_sm 850map_mask_zm thrlo=0.02 newlo=bad thrhi=0.02 newhi=1
```

The final mask is shown in the right-panel of Figure 7.4. Note how it encompasses more emission and has softer edges than the first threshold map.

Step 3: Re-running the map-maker with an external mask supplied

As a last step the map is re-made with this mask supplied as an external file. For this run we apply the additional parameters in a personalised configuration file, `mydimmconfig.lis`.

```
% makemap in=^filelist.txt 850galactic \
          config=^mydimmconfig.lis ref=850map_mask_zm
```

The configuration file, `mydimmconfig.lis`, has the following format (note how it is based on `dimmconfig_bright_extended.lis`). It has decreased the convergence parameter to `maptol` = 0.03 but increased the number of iterations to compensate as 40 is unlikely to be sufficient.

```
^$STARLINK_DIR/share/smurf/dimmconfig_bright_extended.lis
numiter = -100
noisecliphigh = 10.0
maptol = 0.03
ast.zero_mask=1
ast.zero_snr = 0
```

Step 4: Cropping the map

We now crop the map to remove the noisy edges using the PICARD recipe `CROP_SCUBA2_IMAGES`. To determine what to trim we can look at the exposure-time image with GAIA. See Figure 7.5.

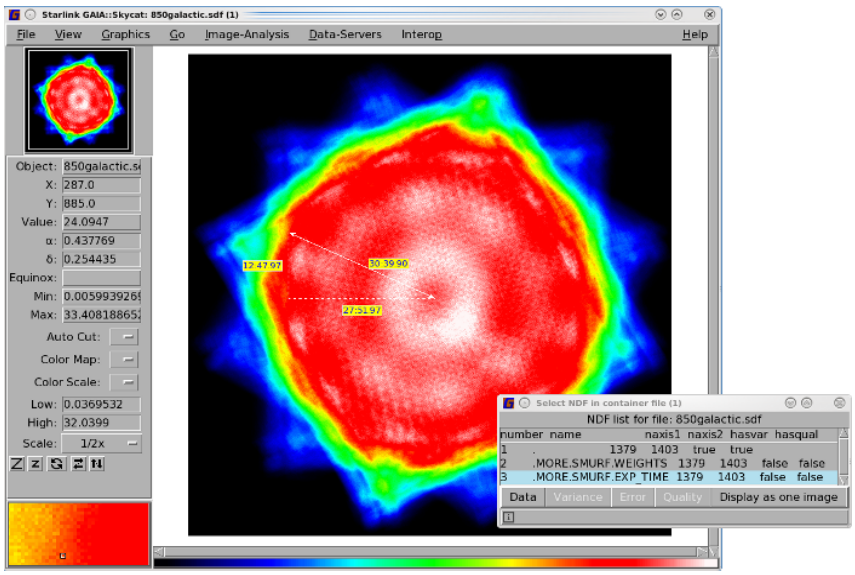


Figure 7.5: The exposure-time image of the science map from Figure 7.3. You can right-click and drag the mouse between two points to measure the distance. Here we see the exposure time dropping off sharply at a radius of 30 arcmin. A non-default colour scale has been chosen to illustrate the morphology.

The exposure-time map shows a sharp drop off at a radius of 30 arcmin. We can thus specify a parameter file called `crop1800.ini` like below:

```
[CROP_SCUBA2_IMAGES]
MAP_RADIUS = 1800
```

and then run:

```
% picard -recpars crop1800.ini CROP_SCUBA2_IMAGES 850galactic.sdf
```

The final cropped map is shown in Figure 7.6. Compared with the first map out of the map-maker (Figure 7.3), slightly more of the faint extended emission is apparent.

One of the challenges facing this type of reduction is the need to account for both faint extended structure and very bright sources in the same map. You may find some degree of bowling remains around the brightest sources.

There are areas you may wish to experiment with. One is to adjust the filtering. Another option is to supply an external mask from a different dataset, e.g. a Herschel map. See Chapter 6 for further discussion.

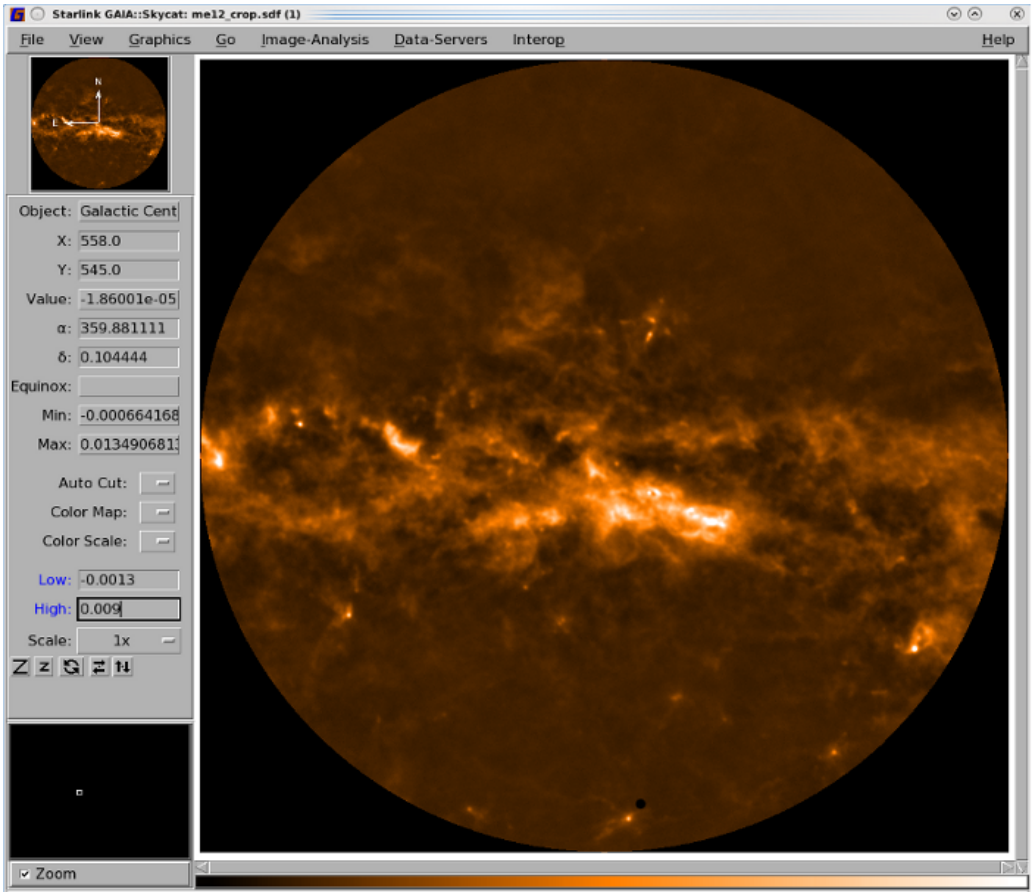


Figure 7.6: The final cropped, reduced map from the map-maker run with an external mask supplied.

Chapter 8
Post-processing Reduction Steps

8.1 Flux-conversion factors

If you are not using the ORAC-DR pipeline to reduce your data (i.e. you are manually employing the map-maker as described in Chapter 5), your data come out of the map-maker in units of picowatts (pW). A flux-conversion factor, or FCF, needs to be applied to scale your data from units of pW to janskys (Jy). For more information on calibrating SCUBA-2 data see Dempsey et al. (2013) [8], Mairs et al. (2021) [16], and Appendix B.

Below are the default FCF values as of Starlink 2021A and they apply to data obtained since 2018 June 30. The ORAC-DR pipeline automatically accounts for changes in the FCFs for earlier dates (see the important notes below) but manually running the pipeline requires consulting Appendix E for the appropriate values.

APERTURE		**PEAK**	
FCF_{arcsec} (Jy/pW/arcsec^2)		FCF_{beam} (Jy/pW/beam)	
450 μm	850 μm	450 μm	850 μm
3.87 ± 0.53	2.07 ± 0.12	472 ± 76	495 ± 32

IMPORTANT NOTES:

- The standard FCF to be applied depends on the observation date of your data. For data that have been *reduced prior* to 2018 June 30 you should consult Appendix E for alternative FCFs.
- Due to a glitch in the WVM, data reduced between 2012 September 19, and 2013 January 18 must be re-reduced using a recent version of Starlink (Hikianalia or later), or should have an FCF derived from a calibrator reduced at the same time (and not our standard FCF) applied to it. The most recent stable release of the Starlink software is always recommended.

8.1.1 Aperture flux

To measure the flux density of extended sources with aperture photometry you should apply the FCF_{arcsec}(also known as the arcsecond FCF). You can then sum the emission in an aperture. FCF_{arcsec} was determined using a 60-arcsec diameter aperture. If your aperture differs from this you should scale your

flux accordingly—the scaling factor can be read off the curve of growth (see Appendix G). This graph gives the ratio of aperture flux to total flux for a range of aperture diameters.

FCF_{arcsec} is determined using 1-arcsec pixels. For different pixel sizes you will also need to multiply by the pixel size squared for a total flux calibrated in units of Jy. For more information, see Appendix B.

8.1.2 Peak flux

If you want to read off the peak flux from your map or fit a function to estimate the peak brightness, you should apply the FCF_{beam} (also known as the peak FCF). When you open your map in GAIA the value of the brightest pixel will be the peak flux of your source. Applying FCF_{beam} will result in a map with units of Jy/beam. For point-like or compact sources smaller than the beam (with a Gaussian profile), this peak value will be the total flux density of your source. For more information, see Appendix B.

8.1.3 Manually Applying the FCF

You can apply an FCF value using the PICARD recipe `CALIBRATE_SCUBA2_DATA`. By default, this will multiply your map by $1000 \times FCF_{beam}$, where the default FCF_{beam}value is presented at the beginning of Section 8.1 or in Appendix E, depending on the observation date and Starlink version used. This produces a calibrated map with units of mJy/beam.

You can supply a parameter file if you wish to use a different value for the FCF or use FCF_{arcsec}. For example, the lines below are written in a parameter file called `cal.ini` to calibrate the data `mapinpW.sdf` with FCF_{arcsec}=2.07:

```
[CALIBRATE_SCUBA2_DATA]
FCF = 2.07
FCF_CALTYPE = ARCSEC
```

The default SCUBA-2 FCF will be used if not given and FCF_CALTYPE may either be BEAM (default) or ARCSEC. The recipe will write out the calibrated file with a `_cal` suffix and will change the units in the map header.

```
% picard -recpars cal.ini CALIBRATE_SCUBA2_DATA mapinpW.sdf
```

Note that the recipe will also take account of the pixel size when applying FCF_{arcsec}.

8.1.4 Determining your own flux-conversion factors

Calibration observations are obtained at various points throughout the night and each individual observation can be used to determine FCF values. As noted in Mairs et al. (2021) [16], the spread in FCF values observed over periods of months or years is comparable to the scatter that can be observed over an individual night.

There is no obvious parameter that is the singular dominant cause of the inherent scatter in the observed FCFs. The spread is undoubtedly affected by a combination of:

(1) The bright and variable atmosphere including subtle effects such as the submillimeter seeing, wind speed, ambient-temperature changes, humidity and low-lying variations in weather conditions with scale heights comparable to several times the height of the JCMT.

(2) The stability of the WVM calibration.

(3) Dish instability producing variations in the beam profile.

(4) Instrumental noise, non-cooled optical components, and uncertainties in the MAKEMAP routine's modeling and removal of noise components from the time stream.

We therefore recommend caution if you are calculating your own FCF values to apply to data on a given night and suggest using the default values supplied above or in Appendix E, depending on the observation date. Nevertheless, below are instructions on how to perform an investigation into the FCFs calculated from the calibrator observations performed close in time to your obtained science data:

(1) Reduce the calibrator with the map-maker using `dimmconfig_bright_compact.lis` as the configuration file.

(2) Determine the FCF value by passing the output map to the PICARD recipe `SCUBA2_CHECK_CAL`.

```
% picard SCUBA2_CHECK_CAL 850calibrator.sdf
```

This will produce a log file (`log.checkcal`) which records the both FCF_{beam} and FCF_{arcsec}. Check that these values closely approximate the standard FCF values given above (to a reasonable margin within the uncertainties).

8.2 Cropping your map

The nature of the scan patterns results in SCUBA-2 maps significantly larger than the requested size. The high noise towards the outer edges is a consequence of the scanning pattern. Although this excess data are down-weighted during reduction by the map-maker, you may wish to remove it before either combining maps (see Section 8.3) or publishing your map.

You can crop your map to the map-size set in the data header or to any requested size of box or circle using the PICARD recipe `CROP_SCUBA2_IMAGES`. The centre of the cropping area will always be the centre of your map.

```
% picard -recpars mypar.lis CROP_SCUBA2_IMAGES map_cal.sdf
```

The example above includes a parameter file specifying the radius of the circle to be extracted (in arcsecs). The format for the parameter file is shown below.

```
[CROP_SCUBA2_IMAGES]
MAP_RADIUS = 1800.0
CROP_METHOD = circle
```

If this parameter file is omitted it will default to a box of sides equal to the map size in the header (as requested in the MSB). The output from `CROP_SCUBA2_IMAGES` is a file with the suffix `_crop`. Full details of this recipe can be found in the PICARD website[1]

Tip

The default crop shape will be a square. Avoid losing good data by specifying a circle using the parameter file.

[1] http://www.oracdr.org/oracdr/PICARD

8.3 Co-adding multiple maps

You may have multiple maps of the same source which you would like to co-add. PICARD has a recipe called MOSAIC_JCMT_IMAGES that co-adds maps while correctly dealing with the exposure time and weights NDF extensions. The images are combined using inverse-variance weighting and the output variance is derived from the input variances.

```
% picard -recpars mypar.lis MOSAIC_JCMT_IMAGES 850map*_cal_crop.sdf
```

This creates a single output file based on the name of the last file in the list, and with a suffix _mos.

There are a number of options associated with MOSAIC_JCMT_IMAGES (see the PICARD manual for a full description). However, the main one is choosing between wcsmosaic (default) and the CCDPACK option makemos for the combination method. For more information on makemos and advice on choosing the best method see **SUN/139**.

The example parameter file below chooses makemos using a 3-σ clipping threshold.

```
[MOSAIC_JCMT_IMAGES]
MOSAIC_TASK = makemos
MAKEMOS_METHOD = sigmas
MAKEMOS_SIGMAS = 3
```

Currently there is no advantage in terms of data quality to reducing all observations simultaneously or separately. However, the latter does allow the option of assessing the individual maps before co-adding.

8.3.1 Registering maps

You can register a series of SCUBA-2 maps to a common reference position using the PICARD recipe SCUBA2_REGISTER_IMAGES. This is only possible if a there is a common, known source that is present in *all* of the input maps. This should be done before combining your maps.

```
% picard -recpars myparams.ini SCUBA2_REGISTER_IMAGES -files listoffiles.txt
```

Here the parameter file contains the equatorial position of the reference source as in the example below. See **SUN/265** for more details.

```
[SCUBA2_REGISTER_IMAGES]
REGISTER_IMAGES = 1
REGISTER_X  = HH:MM:SS.S
REGISTER_Y  = DD:MM:SS.S
```

REGISTER_X and REGISTER_Y may also be galactic longitude and latitude respectively, both measured in decimal degrees.

8.4 Sensitivity

8.4.1 Getting the noise

You can use the PICARD recipe SCUBA2_MAPSTATS to get the noise. This recipe estimates the RMS from both the map, the NEP, and the RMS predicted by the Integration Time Calculator. It then writes out a series of results in a log file called log.mapstats. The parameters written to this file are listed in Appendix C.

```
% picard SCUBA2_MAPSTATS map.sdf
```

This recipe will report the noise in the same input units provided.

Note: SCUBA2_MAPSTATS is only designed to work on reductions of single observations. On co-added observations it could produce misleading results, or even fail completely to work.

If multiple files are run through SCUBA2_MAPSTATS, either in a single call of PICARD or by repeatedly running PICARD in the same terminal on different files, the results will be appended to the existing log.mapstats file. The final columns — project, recipe and filename — are given to ensure it is clear to users which line of the logfile corresponds to which input file. This table can be viewed in an accessible way by using the TOPCAT software included in Starlink:

```
% topcat -f ascii log.mapstats
```

After running the command above, click on "Views" Then "Table Data" to display the table.

Steps for getting the noise in your co-added map

After applying any necessary FCF, SCUBA2_MAPSTATS simply executes the following steps to get the map noise.

(1) Crop the map to remove the noisy edges.

```
% picard CROP_SCUBA2_IMAGES map_cal.sdf
```

(2) Run the KAPPA command stats to extract the median value from the error array (the ndf extension that describes the uncertainty in each pixel, calculated from the square root of the variances in the raw bolometer data that contributed to the pixel).

```
% stats map_cal_crop comp=err order
```

Tip

Use the error array to avoid contamination of the noise distribution from bright sources.

8.4.2 Map statistics

The KAPPA commands histat and stats are very similar and both return a range of statistics describing any NDF. In addition to the main data array they can be passed the error (comp=err), variance (comp=var) or quality (comp=qua) arrays (if available).

The reported statistics include the pixel maximum and minimum, standard deviation, number of pixels used and omitted, along with pixel mode, and mean. If you supply the order keyword, the median is also shown.

```
% stats comp=err map_cal_crop order
```

Note: the standard deviation of the data array will give a similar result to the mean/median of the error array except with additional contamination from sources.

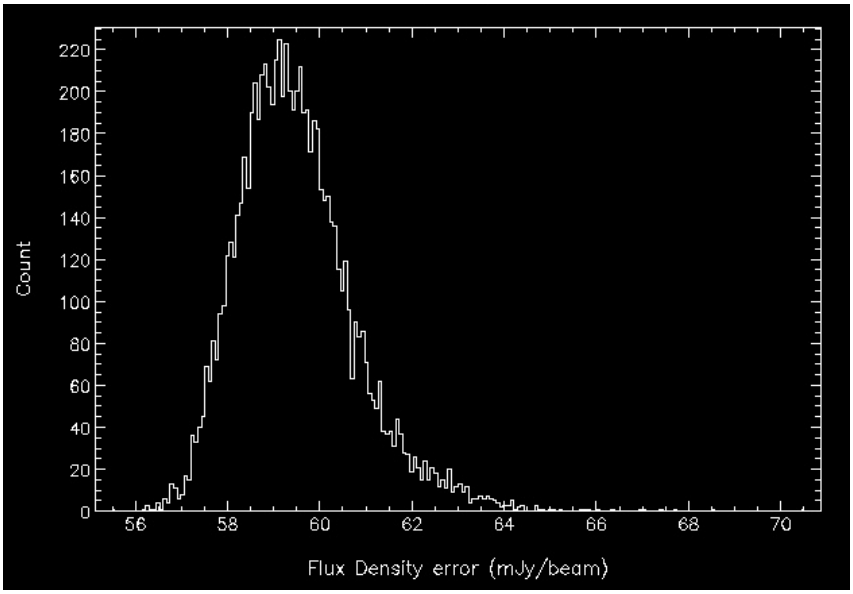


Figure 8.1: The error array viewed with KAPPA command histogram.

8.4.3 Viewing the noise histogram

You can view a histogram of the error array with the KAPPA command histogram. Again `comp=err` must be specified.

```
% histogram map_cal_crop comp=err numbin=200 style="color=white"
```

The output is shown in the Figure 8.1. For more information on the options for histogram see **SUN/95**.

8.4.4 Examining the error map with GAIA

It is also useful to view the error map itself. Open your reduced map in GAIA, then select the **Error** button on the **Select NDF in container file** window—see Figure 8.2. You will need to adjust the scaling to view the error map properly.

To assess the noise using GAIA, go to the toolbar on the main window and click on **Image-Analysis⇒Image regions**. Next select the region shape you would like to check and draw it on your map by clicking and dragging the mouse. Click the **Stats selected** button in the **Image regions** window to get a report of the statistics in the selected region.

Tip

You can write out the error array of your map into a new NDF using the KAPPA command ndfcopy. For example, `% ndfcopy map comp=err map_err`

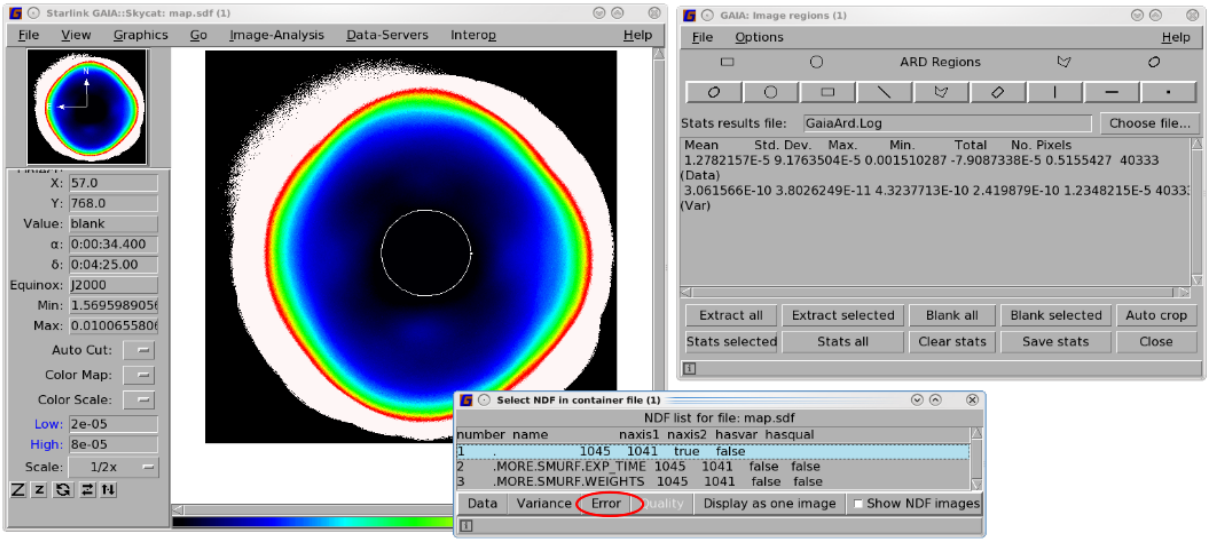


Figure 8.2: The error array displayed with GAIA. After clicking the **Error** button (circled in red) you will have to rescale your map.

8.5 Regridding your data

To change the pixel size use the KAPPA command compave. The following example increases the pixel size from 4 arcsec to 8 arcsec by using a compression factor of 2.

```
% compave map map_regrid 2
```

> **Tip**
>
> Remember you can use ndftrace if you are unsure of the pixel size.

8.6 Displaying masks

SCUBA-2 maps created using "AST-masking" (see Section 3.5) will contain a `Quality` array indicating the background pixels that were masked (i.e. forced to zero). In itself this is fairly simple—background pixels have a non-zero Quality value and source pixels have a Quality value of zero. However, it is also possible to have independent masks for the `FLT` and `COM` models, in addition to the `AST` mask. For instance, maps created using `dimmconfig_bright_extended` or `dimmconfig_jsa_generic` will have `Quality` arrays that contain both a `FLT` and an `AST` mask, and so some care needs to be used when interpreting the `Quality` array.

Each value in the `Quality` array is restricted to taking integer values between 0 and 255, and so can be thought of as 8 separate bits. Each "bit plane" within the `Quality` array holds a single mask—`AST`, `FLT` or `COM`. The showqual command can be used to find out which mask is held by which bit plane:

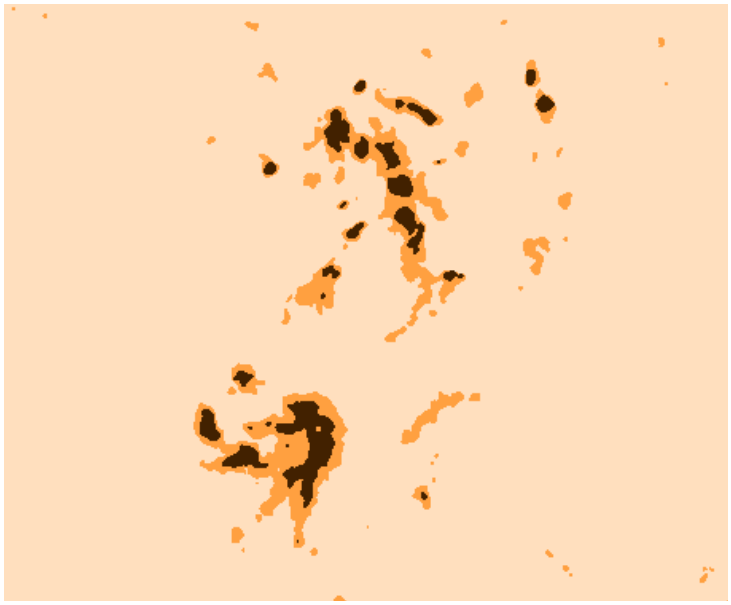


Figure 8.3: The quality component from a typical map.

```
% showqual fred.sdf
   AST (bit 1) - "Set iff AST model is zeroed at the output pixel"
   FLT (bit 2) - "Set iff FLT model is blanked at the output pixel"
```

This means that the AST mask is stored in bit 1 (the least significant bit), the FLT mask is stored in bit 2, and there is no COM mask. Note, if a map was produced using FLT masking but no AST masking, then the FLT mask would be stored in bit 1.

The decimal integer value of any element of the Quality array is equal to the binary value formed from the bits listed by showqual. So in the above case the maximum Quality value is 3 (the decimal equivalent of binary "11"-e–both bits set). Remembering that a bit is set (*i.e.* is 1) for background pixels and cleared (*i.e.* is 0) for source pixels, it follows that the four possible decimal Quality values in the above case (0-3) are:

(1) - neither bit set, so the pixel is inside both the AST and the FLT mask (a source pixel).

(2) – Bit 1 set, but not Bit 2, so the pixel is outside the AST mask but inside the FLT mask (a border-line pixel).

(3) – Bit 2 set, but not Bit 1, so the pixel is inside the AST mask but outside the FLT mask (a border-line pixel).

(4) – Both bits set, so the pixel is inside neither mask (a background pixel).

Figure 8.3 shows a simple map of the Quality array values—the black areas have value zero and are thus inside both masks, the dark brown areas have value 2 and are thus inside the AST mask but outside the FLT mask. The light-brown areas have value 3 and are inside neither mask. In this particular case, there are no areas with a quality value of 1, so the FLT mask is contained entirely within the AST mask.

To produce a plot like this, first open your map in GAIA. Select the top level NDF from list in the pop-up window then click the **Quality** button. You will need to rescale the map to bring out the masks–see Figure 8.4 for another example showing the full GAIA window (using rather brighter colours this time!).

There are several commands within KAPPA that manipulate Quality arrays in various ways. For instance, the setbb command allows pixel data values to be set bad if the associated Quality value has a specified collection of set bits. Thus:

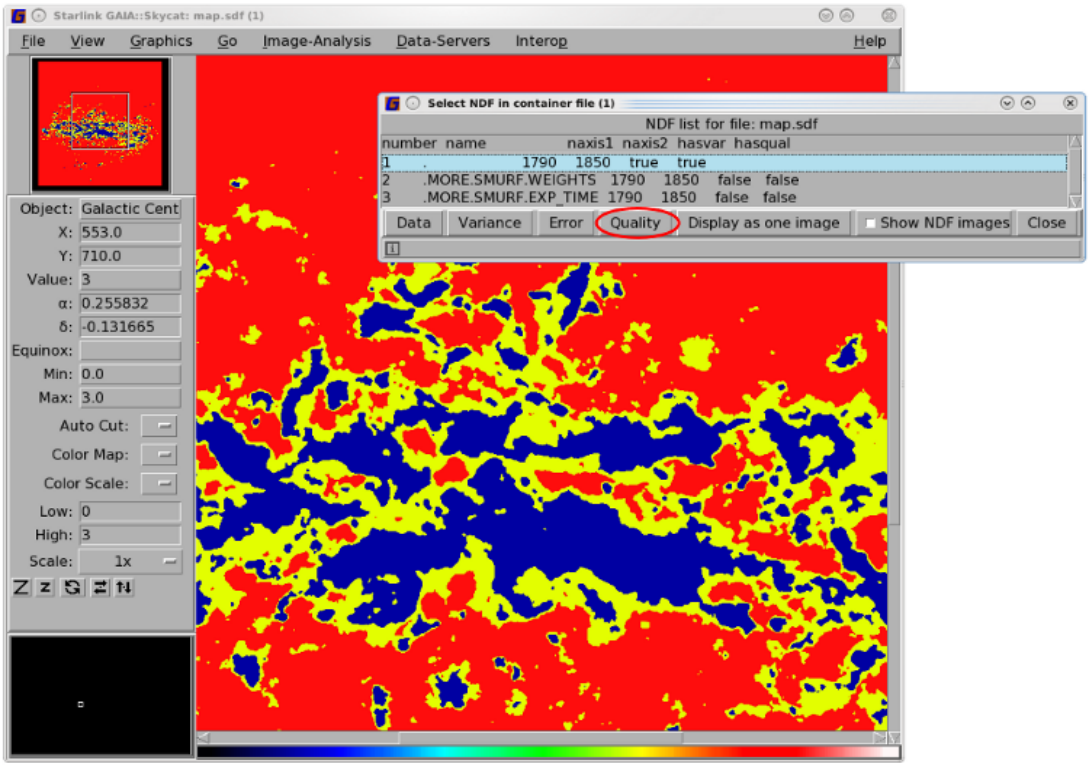


Figure 8.4: Using GAIA to display the mask applied by the map-maker. Select the QUALITY component of your map.

```
% setbb fred 1
```

will set all pixels bad in `fred.sdf` except for those inside the `AST` mask. Likewise,

```
% setbb fred 2
```

will set all pixels bad except for those inside the `FLT` mask. Note, the change made by setbb is temporary—it can be undone by doing:

```
% setbb fred 0
```

To display the `fred.sdf` map and then overlay the `AST` mask in blue and the `FLT` mask in red, do:

```
% gdclear
% display fred mode=perc percentiles=\[2,98\]
% setbb fred 1
% contour fred clear=no mode=good labpos=! style='colour=blue'
% setbb fred 2
% contour fred clear=no mode=good labpos=! style='colour=red'
% setbb fred 0
```

The resulting plot is shown in Figure 8.5.

Alternatively, GAIA can be used to contour the `Quality` array over an image, but you cannot then distinguish the `AST` mask from the `FLT` mask. First you will need to copy out the QUALITY component of the data to a separate file using ndfcopy:

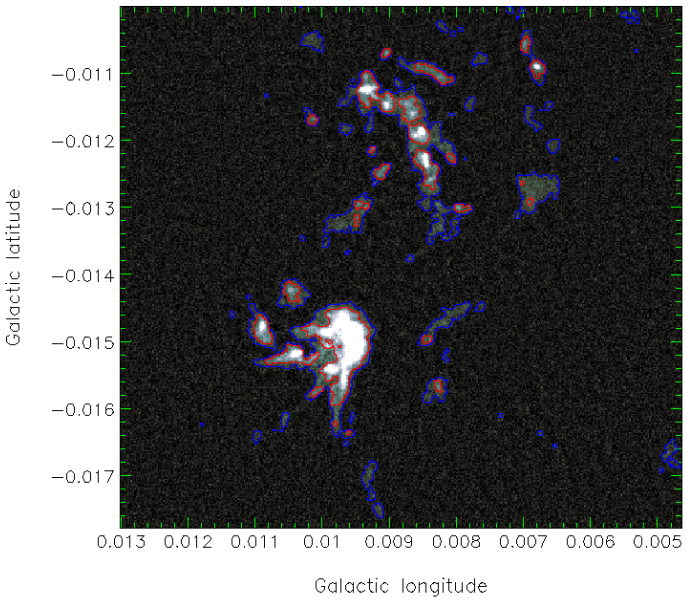


Figure 8.5: The AST and FLT masks contoured together over an image.

```
% ndfcopy comp=qua map map_mask
```

Then open your map in GAIA and contour the mask NDF on top. Select **Contouring...** from the **Image-Analysis** menu, and supply the name of the mask NDF either by entering its name in the **Other image:** box, or selecting with **Choose file...** file browser. See Figure 8.6.

For more information on the use of masks by makemap and the parameters that affect them see section 3.5.

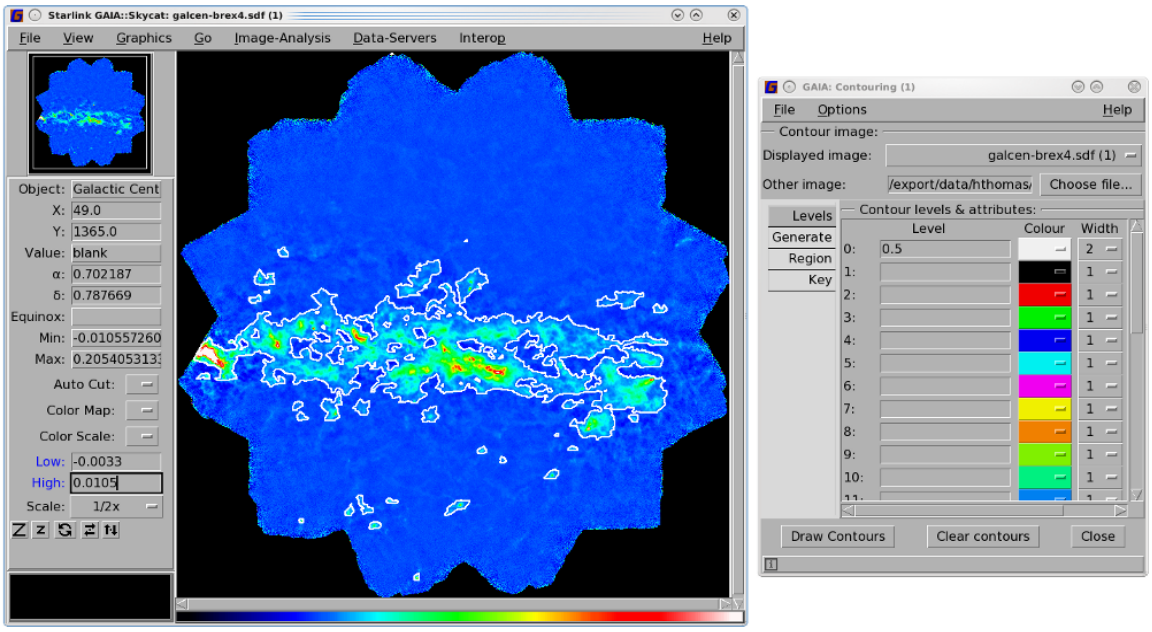


Figure 8.6: Using GAIA to display your map with the AST mask used by the map-maker contoured on top.

8.7 Point-source extraction: the matched filter

This effectively fits a single Gaussian point spread function (PSF), centered over every pixel in the map, and applies a background suppression filter to remove any residual large-scale noise.

Cosmology maps usually contain very faint sources that often need extra help extracting. The PICARD recipe SCUBA2_MATCHED_FILTER can be used to improve point-source detectability.

The matched filter works by smoothing the map and PSF with a broad Gaussian, and then subtracting from the originals. The images are then convolved with the modified PSF. The output map should be used primarily for source detection only. Although the output is normalised to preserve peak flux density, the accuracy of this depends on how closely the real PSF matches the telescope beam size. In the case of nearby sources, each ends up contributing flux to both peaks.

```
% picard -recpars mypar.lis SCUBA2_MATCHED_FILTER 850_map_cal_crop.sdf
```

As in the example parameter file below we have requested the background should be estimated by first smoothing the map and PSF with a 15-arcsec Gaussian.

```
[SCUBA2_MATCHED_FILTER]
SMOOTH_FWHM = 15
```

This is a fairly common technique used throughout the extra-galactic sub-millimetre community to identify potential sources. A full description of the matched filter principle is given in Appendix D, while the PICARD manual gives full details of all the available parameters.

8.8 Clump finding

The CUPID application findclumps can be used to generate a clump catalogue. It identifies clumps of emission in one-, two- or three-dimensional NDFs. You can select from the clump-finding algorithms "FellWalker"[2], "Gaussclumps", "ClumpFind" or "Reinhold" and must supply a configuration file specific to each method. See the CUPID manual for descriptions of the various algorithms.

The result is returned as a catalogue in a text file and as a NDF pixel mask showing the clump boundaries.

```
% findclumps in=S2map.sdf out=clumpmap.sdf outcat=clumps.FIT logfile=clumps.log \
  config=^config.dat method=fellwalker rms=25 shape=polygon
```

The shape option allows findclumps to create an STC-S description (polygonal or elliptical) for each clump that can be displayed over any image in GAIA(Figure 8.7). These are added as extra columns to the output catalogue.

Polygon Each polygon will have, at most, 15 vertices. For two-dimensional data the polygon is a fit to the clump's outer boundary (the region containing all good data values). For three-dimensional data the spatial footprint of each clump is determined by rejecting the least significant 10% of spatial pixels, where "significance" is measured by the number of spectral channels that contribute to the spatial pixel. The polygon is then a fit to the outer boundary of the remaining spatial pixels.

Ellipse All data values in the clump are projected onto the spatial plane and "size" of the collapsed clump at four different position angles—all separated by 45°—is found. The ellipse that generates the same sizes at the four position angles is then found and used as the clump shape.

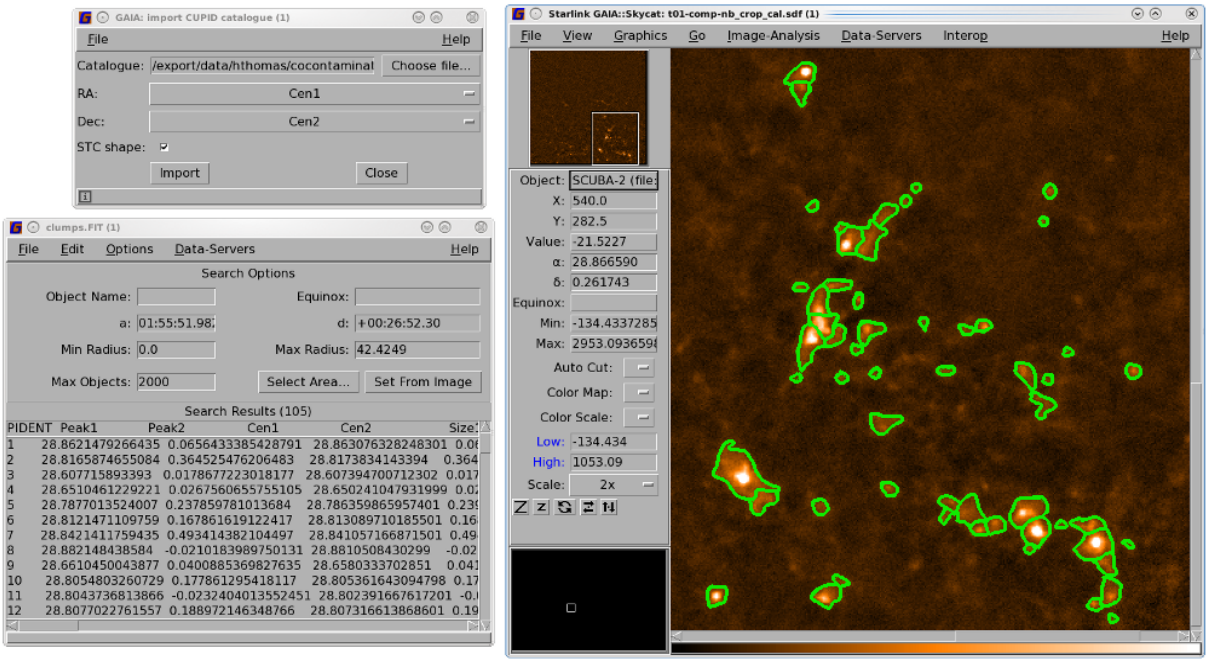


Figure 8.7: STC shapes displayed over the input map with GAIA. To overlay your clumps in this way display your map with GAIA. Under the **Image-Analysis** menu select **Positions...** followed by **Import CUPID catalogue...** In the pop-up window select the `.FIT` file generated by findclumps and ensure the **STC shape** box is checked before importing.

8.9 Map provenance & configuration parameters

You may want to check a reduced file to determine which raw files went into it and what configuration parameters were used with the map-maker. There are a number of useful KAPPA commands to help you with this:

Map provenance provshow will list all the NDFs and operations that were used in the creation of your map, while hislist will show a truncated list of input files and operations but will list every configuration parameter used by the map-maker. This includes all the hidden default values as well as those specified in your supplied configuration file. These commands are executed like this:

```
% provshow 850map
% hislist 850map
```

The output of provshow can be very long. If you merely want to know what the original files were, the ones with no parents, select the root ancestors.

```
% provshow 850map show=roots
```

Map-maker parameters The task configmeld will compare two maps and highlight any differences between the configuration parameters that were used to create the maps. A configuration file can be supplied in place of a map, in which case the parameters used to create the map are compared with those in the configuration file. configmeld requires a visual-differences tool to be installed on your machine; those currently recognised are: `meld` `opendiff`, `diffmerge`, `kdiff3`, `tkdiff`, and `diffuse`, and are searched for in that order. The follow example illustrates two ways to run configmeld:

```
% configmeld 850map1.sdf 850map2.sdf
% configmeld 850map1.sdf ^mydimmconfig2.lis
```

Another useful command is the KAPPA command configecho. This is very versatile and will display the name and value of one or all configuration parameters either from a configuration file or from the history of an NDF.

The first example below will return the value of `numiter` from the map `850map.sdf`. The second example will display the values of all the parameters in `850map.sdf` and will prefix them with a '+' if they differ from the the default parameter values defined in `$SMURF_DIR/smurf_makemap.def`.

```
% configecho name=numiter config=! ndf=850map.sdf
% configecho name=! config=^$SMURF_DIR/smurf_makemap.def ndf=850map.sdf
```

Chapter 9
SCUBA-2 Diagnostic Tools

This chapter describes a number of procedures to visualise and assess the quality of raw data files. These steps need not be part of your data reduction process and do not concern the iterative map-maker.

There are reasons, however, that you may wish to examine your raw data in greater depth. The most likely motivation is an unusual result from your reduction such as higher than expected noise, artefacts in your map, or inconsistent noise across multiple observations. This chapter will help you get to the bottom of many of these issues.

9.1 Concatenate & apply a flat-field

Since SCUBA-2 data for a given sub-array are broken into multiple 30-second scans by the data acquisition (DA) system, it is useful to concatenate the data into a single file. The SMURF task sc2concat can be used for this operation. The example below combines all of the files associated with Observation 8 for the s8a array into a single file called `s8a20120725_0008_con`.

```
% sc2concat s8a20120725_0008\*.sdf s8a20120725_0008_con
```

sc2concat will automatically filter out any dark or flat-field observations, so that the concatenated file contains only the science data. Be careful when concatenating a very long observation since the output file may be too large to reasonably handle. Fifteen-minute chunks (30 files) should be sufficient.

sc2concat applies the flat-field by default (although it can be disabled using the `noflat` option on the command-line).

The flat-field can also be applied manually using the flatfield command.

```
% flatfield 's8a20120701_0008*.sdf' '*_flat'
```

Here, the output will be a flat-fielded version of each science scan in Observation 8; the file names will be the original input names with _flat appended to them.

As a general rule, you should apply the flat-field to your data before examining it.

Tip

You do not need to apply the flat-field prior to reducing your data with the map-maker.

9.2 Headers and file structure

There are two KAPPA tasks which are extremely useful for examining your data: fitslist and ndftrace, which can be used to view the FITS headers and dimensions of the data.

fitslist: This lists the FITS header information for any file (raw or reduced). This extensive list includes dates & times, source name, scan type, pattern and velocity, size of the map, exposure time, start and end elevation, opacity, and the temperature of the instrument. An example, selecting any line in the FITS header which includes the string "SEQ_TYPE", is given below:

```
% fitslist s8a20120720_00030_000\*.sdf | grep SEQ_TYPE
```

If you already know the name of the parameter you want to view you can use the fitsval command instead, e.g.

```
% fitsval file.sdf TAU225ST
```

ndftrace: ndftrace displays the attributes of the data structure. This will tell you the units of the data, pixel bounds, dimensions, world coordinate bounds and attributes, and axis assignations.

```
% ndftrace file.sdf fullframe
```

Full details of these commands can be found in the KAPPA manual.

9.3 Displaying scan patterns

The movement of the telescope throughout a scan (as well as other state information) is stored in the MORE.SMURF.JCMTSTATE extension of a data file. The SMURF task jcmtstate2cat converts this information into a simple ASCII tab-separated table.

```
% jcmtstate2cat s8a20120701_00008_*.sdf > state.tst
```

Multiple files can be supplied to the command using standard shell wild cards. If you have already concatenated your data you can simply input the single concatenated file. It may be useful to view the scan pattern for your observation, particularly for maps taken at high elevations, to ensure the pattern completed successfully.

This catalogue can be loaded into TOPCAT for plotting, making sure to specify the TST format during loading.

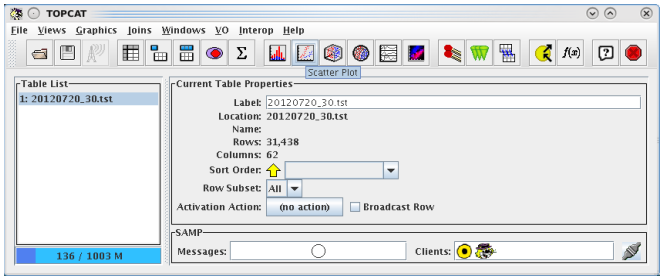
```
% topcat -f tst state.tst
```

Example of scan patterns displayed with TOPCAT can be seen in Figure 2.3. Detailed instructions on how to display the scan pattern for your observation are given in Figure 9.1. All of the time-varying header values are available for plotting. Other values include the azimuth and elevation offsets (DAZ & DEL), the WVM and 225 GHz opacity values, and the instrument temperatures (e.g. SC2_FPUTEMP gives the temperature of the focal plane).

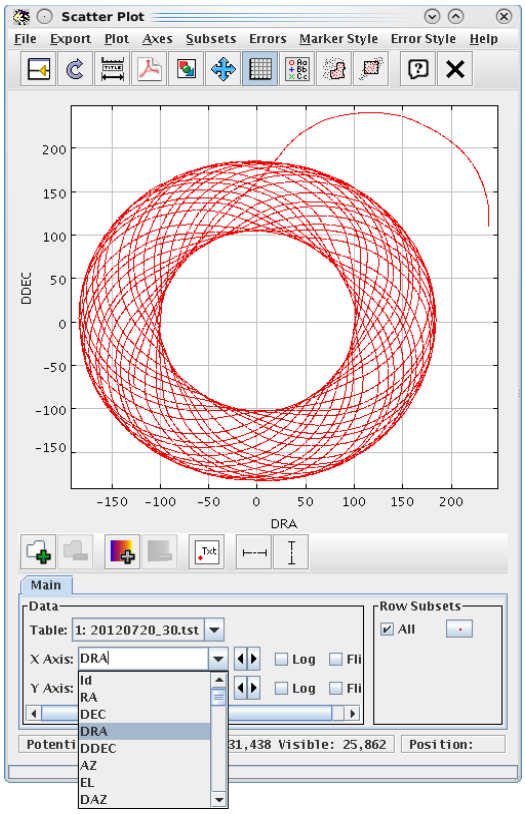
Topcat Example

```
% topcat -f tst 20120720_30.tst
```

Load the tst file generated by jcmtstate2cat into TOPCAT.



In TOPCAT select the scatter plot option from the menu bar across the top of the window.



With the scatter plot displayed you can adjust the *X*-axis and *Y*-axis values to DRA and DDEC respectively to display the scan pattern. If you are interested in seeing how any of the variables change over time, select the the *X* Axis to be either Id or RTS_NUM.

Figure 9.1: TOPCAT example demonstrating how to display the scan pattern for an observation.

Due to extreme accelerations at "turn-around" points of a scan pattern (especially for PONGs), the telescope finds it hard to follow the proscribed scan patterns at high elevations. To mitigate this we try to avoid observing any sources above 70° elevation. If the fitslist keywords `ELSTART` and `ELEND` indicate that your map was taken at high elevation you may consider checking the success of the scan pattern. If you find your observation has failed to follow the demanded scan pattern don't worry, the data are likely to still be useful. This is especially true for DAISY maps where the high exposure-time central region is usually unaffected.

> **Tip**
>
> Follow the jcmtstate2cat command with `-h` to find out more information.

9.4 Displaying time-series data

Use the Starlink application GAIA to visualise the bolometer time-series data (or indeed *any* SCUBA-2 data file). This is initiated simply typing `gaia` into a terminal.

```
% gaia s8a20120725_00058_con.sdf
```

Loading a file in GAIA produces two windows. The main window (see Figure 9.2) shows a map of bolometer values at a given point in time. The time slice displayed may be changed by scrolling through the time axis. This is done in the second window entitled **Display image sections of a cube**. The **Index of plane** slider towards the top of this window may be moved to display different time slices in the main window.

A third window will appear when you click on a bolometer—the **Spectral plot** (see Figure 9.3). This shows an automatically scaled plot of the raw time stream of data for that given bolometer. It will be overridden when you click on a different bolometer.

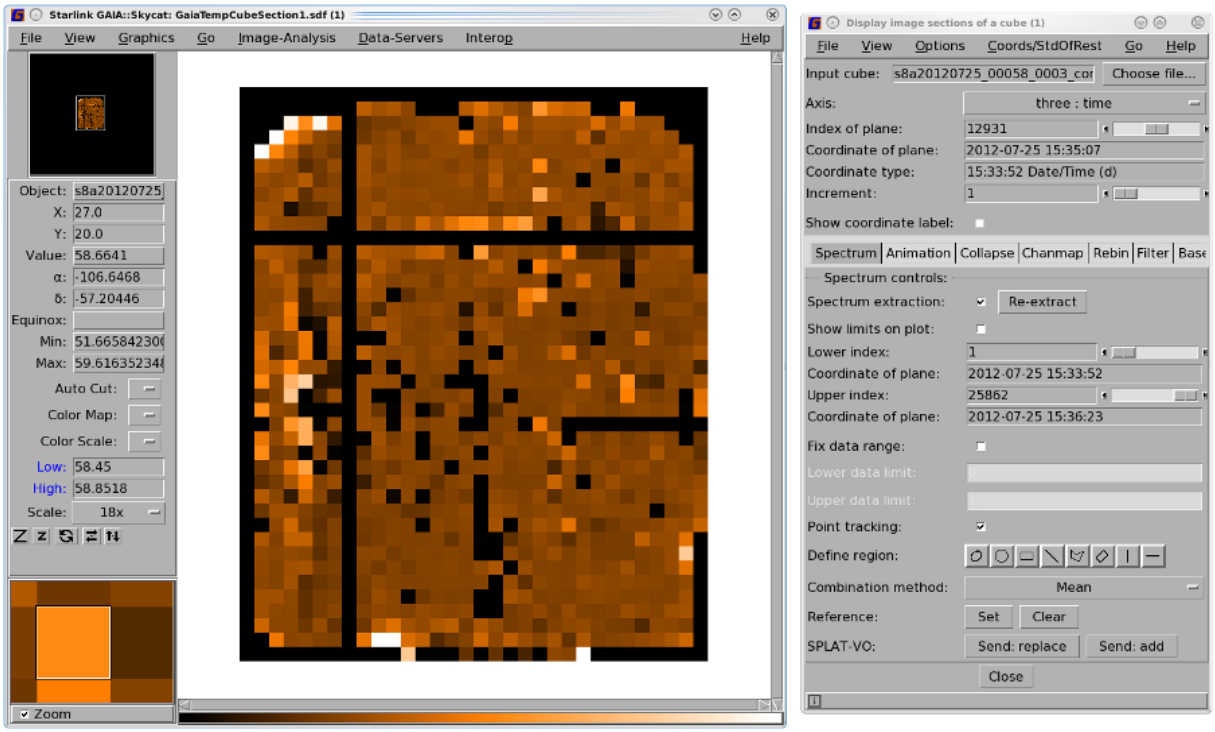


Figure 9.2: Initial GAIA windows displayed upon loading a data cube. The main window in the left shows a map of bolometer values at a fixed sample in time. You may have to zoom in multiple times by clicking the **Z** icon. On the right-hand side, the **Display image sections of a cube** dialogue enables you to navigate the time axis.

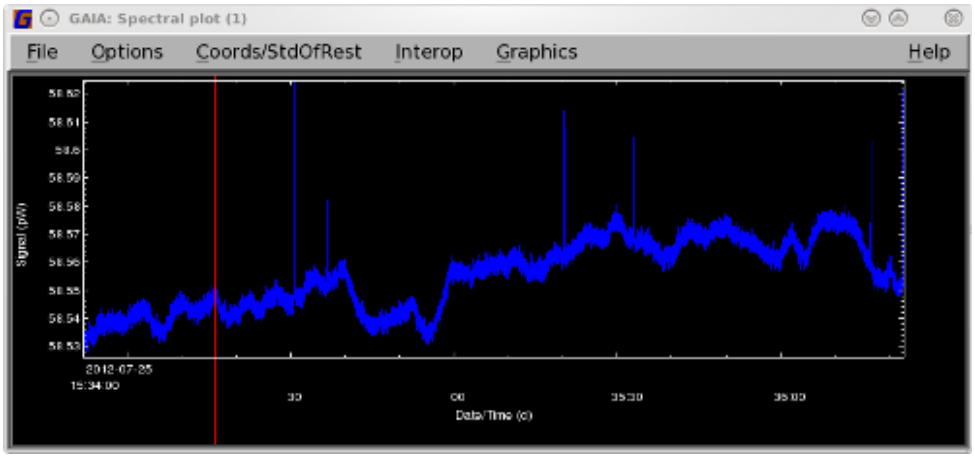


Figure 9.3: The **Spectral plot** window displays the time-varying signal. This window appears automatically when a bolometer is clicked in the main window. The vertical red line indicates the time slice that is currently selected in the **Display image sections of a cube** dialogue—this can be dragged across the spectrum to scroll through the time-slices.

A second way to scroll through the time axis is to click and drag the vertical red bar on the **Spectral plot** window. As you do so, the array shown in the main window will automatically update.

To highlight small variations between bolometers you will need to change the auto cut and (depending on your preference) the colour scheme—both are controlled by buttons on the sidebar.

See the GAIA manual for full details.[1]

9.5 Regridding data into a map

Any raw time-series data can be quickly regridded into sky frame coordinates using the SMURF makemap task in rebin mode. This involves no further processing of the data. The following command produces a map from the raw concatenated data; unlike the iterative mode of makemap described in the next chapter, no configuration file is required.

```
% makemap s8a20120725_00058_con.sdf crl2688_sky method=rebin spread=near
```

The output map here is called `crl2688_sky.sdf` and is shown in Figure 9.4. The pixel scale is left at the default values of 2 arcsec on a side at 450μm and 4 arcsec at 850μm (although this can be changed using the `pixsize=`x option on the command-line, where x is in arcsec).

9.6 Notes on cleaning your data

Cleaning raw data is an essential first step towards making a quality final map. The map-maker performs all of these cleaning steps during the pre-processing stage. The commands for manually cleaning your data are given in Appendix A. You can also check out the SMURF SRO Cookbook[2] which goes into great depth on the data cleaning options.

[1] `http://www.starlink.ac.uk/docs/sun214.htx/sun214.html`
[2] `http://www.starlink.ac.uk/docs/sc19.htx/sc19.html`

9.7 Checking the array performance

The on-sky performance of the array can be assessed using the SMURF command calcnoise. Rather than give an absolute measure, calcnoise should be used as an indicator of array performance and stability. calcnoise cleans the data then calculates the white noise on the array (between 2 and 10 Hz by default).

```
% calcnoise s8a20110720_00030\*.sdf s8a_noise power=!
```

It will prompt for a configuration file to describe the cleaning steps. The default is the file `$STARLINK_DIR/share/smurf/dimmconfig_calcnoise.lis` that is included in SMURF. Two noise measurements are reported in the terminal: the 'Effective noise' and the 'Effective NEP'.

An output file is created for each sub-array with the NEP map stored in the `.MORE.SMURF.NEP` extension.

If you have a bright source in the field this will contaminate the signal. In this case you should examine the `NOI` model from the map-maker instead—see Section 3.3 for a description and Section 9.8 for details on how to examine it.

9.8 Exporting individual models

By default, the final values of the models fitted by the map-maker are *not* written out. However, this can be changed by setting `exportndf` in the configuration file to the list of models that you wish to view.

```
exportndf = (com,gai,ast,flt,res,noi,qua)
```

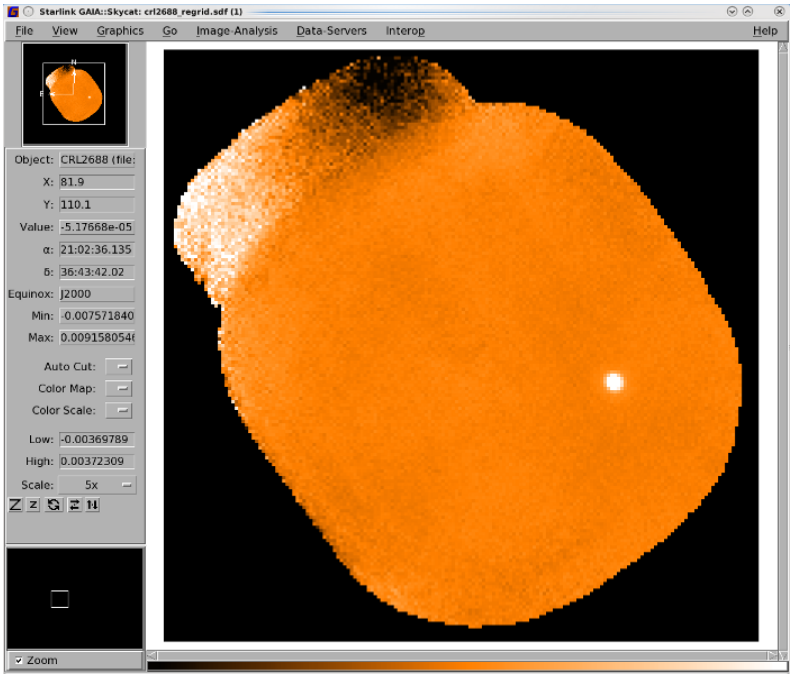


Figure 9.4: The regridded map of CRL 2688 with the s8a sub-array displayed with GAIA.

Tip

If you do not include `method=rebin`, the map-maker will default to `method=iterate`.

In addition to the models listed in Section 3.3, you can request `RES` in order to export the `RES` model (the residual signal remaining after the other models have been removed). If `NOI` (the estimate of the bolometer noise levels) is exported, it is stored as the VARIANCE component of the `RES` model; thus, export of `RES` is implied if `NOI` is specified.

The `exportndf` parameter will write out the requested models as NDF files with names based on the first input file that went into the maps for each sub-array. This is first suffixed by `con`, indicating that several data files may have been concatenated together. The three-letter code for each model is then appended to the filename (such as `s8a20120720_00030_0003_con_com.sdf`, `s8a20120720_00030_0003_con_flt.sdf`, `s8a20120720_00030_0003_con_res.sdf`)[3] The variance and quality for the data are stored as the VARIANCE and QUALITY components within the residual file NDF.

Tip

These exported model components are 3-dimensional arrays with axes `(bolometer column, bolometer row, time slice index)`, and so can be viewed using the cube visualisation facilities within GAIA (see **SUN/214**).

[3]The filename shows the third sub-scan of Observation 30 since this is the first science file that is encountered (see Section 2.3).

Bibliography

[1] Archibald, E. N., et al, 2002, *On the atmospheric limitations of ground-based submillimetre astronomy using array receivers*, MNRAS, 336, 1-13 (DOI:10.1046/j.1365-8711.2002.05582.x) B.2

[2] Berry D. S., 2015, *FellWalker - a Clump Identification Algorithm*, Ast. & Comp., 10, 22-31 (DOI:10.1016/j.ascom.2014.11.004) 4.2.2, 8.8

[3] Cavanagh B., Jenness T., Economou F., Currie M. J., 2008, *The ORAC-DR data reduction pipeline*, Astron. Nactr., 329, 295 (DOI:10.1002/asna.200710944) 1.1, 1.3, 1.3.5, 4.1

[4] Chapin E. L., et al., 2013, *SMURF – Sub-Millimetre User Reduction Facility*, Starlink User Note 258 1.1, 1.3, 1.3.2

[5] Chapin E. L., et al., 2013, *SCUBA-2: iterative map-making with the Sub-Millimetre User Reduction Facility*, MNRAS, 430, 2545 (DOI:10.1093/mnras/stt052) 3.2, 3.3

[6] Currie M. J., Wallace P. T., Warren-Smith R. F., 1989, *Starlink Standard Data Structures*, Starlink General Paper 38.2

[7] Currie M. J., Berry D. S, 2013, *KAPPA – Kernel Application Package*, Starlink User Note 95 1.1, 1.3

[8] Dempsey J. T. et al., 2013, *SCUBA-2: on-sky calibration using submillimetre standard sources*, MNRAS, 430, 2534 (DOI:10.1093/mnras/stt090) 8.1, B.2, B.2, B.2, B.1, E, E, E.1

[9] Dempsey J. T., Friberg P., Jenness T., Bintley D., Holland W. S., 2010 *Extinction correction and on-sky calibration of SCUBA-2*, Proc. SPIE, 7741 (DOI:10.1117/12.856476) B.2

[10] Draper P. W., Gray N., Berry D. S., Taylor M., 2012, *GAIA – Graphical Astronomy and Image Analysis Tool*, Starlink User Note 214 1.1, 1.3

[11] Gibb A. G., Jenness T., Economou F., 2012, *PICARD — a PIpeline for Combining and Analyzing Reduced Data* Starlink User Note 265 1.1, 1.3

[12] Holland, W. S., et al, 2013, *SCUBA-2: The 10,000 pixel bolometer camera on the James Clerk Maxwell Telescope*, MNRAS, 430, 2513 (DOI:10.1093/mnras/sts612) 2.1, 2.2

[13] Jenness T., et al, 2002, *Towards the automated reduction and calibration of SCUBA data from the James Clerk Maxwell Telescope*, MNRAS, 336, 14-21 (DOI:10.1046/j.1365-8711.2002.05604.x)

[14] Jenness T., et al, 2015, *Learning from 25 years of the extensible N-Dimensional Data Format*, Ast. & Comp., 12:146-161, (DOI:10.1016/j.ascom.2014.11.001) 1.3.1

[15] Mairs S., et al, 2015, *The JCMT Gould Belt Survey: a quantitative comparison between SCUBA-2 data reduction methods*, MNRAS, 454:2557, (DOI:10.1093/mnras/stv2192) 4, 6.3, 6.6, 7.2

[16] Mairs S., et al, 2021, *A Decade of SCUBA-2: A Comprehensive Guide to Calibrating 450 and 850 micron Data at the JCMT*, 2021, AJ, 162, 191, (DOI:10.3847/1538-3881/ac18bf). 8.1, 8.1.4, B.2, B.2, B.1, E, E, E.1, F

[17] Scott D., Van Engelen A., 2005, *Scan Mode Strategies for SCUBA-2*, SCUBA-2 Data Reduction document SC2/ANA/S210/005

Appendix A
Cleaning the Raw Data

There are two ways to clean time-series data:

(1) Run makemap and add `doclean = 1` to your configuration (see Section 5.4.3).

(2) Run sc2clean to clean the time-series data without making a map.

The rest of this appendix gives more details on using sc2clean.

sc2clean can be used to do two basic tasks in one go: concatenate data (with or without applying a flatfield); and cleaning (fix up steps and spikes, remove the means, filter, remove common-mode etc.). It uses the same configuration files as the iterative map-maker (though ignoring the map-making specific items).

In this first basic example, we just want to clean up some data enough to see whether the bolometers have been flat-fielded correctly, and more-or-less exhibit the same behaviour over time. The pre-processing or cleaning steps used by default (*i.e.* if "`config=def`" is included on the command line) are summarised in Section J.3. Note, whilst it is not recommended to run makemap in this way (*i.e.* without a configuration file), it is not so critical when running sc2clean.

```
% sc2clean ^files.lis clean config=def
```

Here `files.lis` can just contain a single file from a sub-array, or a subset, e.g. `s8a20110417_00051_0003.sdf` (the first file containing science data), `s8a20110417_00051_000"[1234]"` (File 1 is a noise observation with shutter closed that gets ignored, File 2 is a flatfield observation that will be used to override the flatfield stored in the subsequent Files 3 and 4 which are concatenated together, the `.sdf` is optional), `s8a20110417_00051_000\?` (Files 1 through 9), `s8a20110417_00051_\*` (the whole observation).

If you inspect the resulting `clean.sdf` in GAIA (Section 9.4) and flip through the data cube you should see all of the bolometers signals go up and down together with about the same amplitude: the hope is that for a well-behaved instrument you are mostly seeing sky noise variations that are seen with roughly the same amplitude by all bolometers.

Another common feature, if the scans are particularly long and/or fast (e.g. 1 degree across), is strong periodic signals that are correlated with the scan pattern. See Section 9.3—in particular you will want to plot `az` and `el` (the absolute azimuth and elevation), and also `daz` and `del` (the azimuth and elevation offsets from the map centre). This signal is usually azimuth-correlated due to magnetic-field pickup. It only shows up in azimuth, because the instrument is on a Nasmyth platform and therefore does not move in elevation.

Part of the reason the signals look the same is because they have been flatfielded. You can turn off flatfielding using the `noflat` option to sc2clean, and you should then see that all of the detector amplitudes vary.

Another very useful option is to remove the common signal observed by all of the bolometers. This may be accomplished by

```
% sc2clean ^files.lis clean config='"compreprocess=1"'
```

This `config` setting causes the default values to be used for all configuration parameters except `compreprocess`, which is set to 1 (the default is 0). The residual signal left by this command will exhibit second-order time-varying correlated signals across the focal plane. Usually these are not very large, but in some cases some very large localized signals have been detected, particularly in the 850 μm arrays in early 2011.

Another variation on this is to accentuate the residual low-frequency noise by low-pass filtering the result. This can again be accomplished by simply adding a filter command in the `config` parameter, which in this case low-pass filters with a cutoff at 10 Hz:

```
% sc2clean ^files.lis clean config='"compreprocess=1,filt_edgelow=10"'
```

Finally, in some cases you might just want to fit and remove polynomial baselines from the bolometers (by default only the mean is removed). This example will remove a line, but you can increase the value of `order` to remove higher-order polynomials

```
% sc2clean ^files.lis clean config='"order=1"'
```

Non-default values for any of the cleaning parameter can be specified like so:

```
% sc2clean ^files.lis clean config='"order=1,dcfitbox=30,dcthresh=25,dcsmooth=50"'
```

Or you can create your own customised configuration file. For instance:

```
% cat myconf
order=1
dcfitbox=30
dcthresh=25
dcsmooth=50
% sc2clean ^files.lis clean config=^myconf
```

The more interesting pre-processing options that may be specified are listed and described in Appendix I.

Appendix B
SCUBA-2 Data Calibration

B.1 Flux-conversion factors (FCFs)

Primary and secondary calibrator observations have been reduced using the specifically designed `dimmconfig_bright_compact.lis`. The maps produced from this are then analysed using tailor-made PICARD recipes. For instructions on applying the FCFs to your map see Section 8.1 and Appendix E.

A map reduced by the map-maker has units of pW. To calibrate the data into units of janskys (Jy), a set of bright, point-source objects with well-known flux densities are observed regularly to provide a flux conversion factor (FCF). The data (in pW) can be multiplied by this FCF to obtain a calibrated map. The FCF can also be used to assess the relative performance of the instrument from night to night. The noise equivalent flux density (NEFD) is a measure of the instrument sensitivity, and while not discussed here, is also produced by the PICARD recipe shown here. For calibration of primary and secondary calibrators, the FCFs and NEFDs have been calculated as follows:

(1) The PICARD recipe `SCUBA2_FCFNEFD` takes the reduced map, crops it, and runs background removal. Surface-fitting parameters are changeable in the PICARD parameter file.

(2) It then runs the KAPPA beamfit task on the specified point source. The beamfit task will estimate the peak (uncalibrated) flux density and the FWHM. The integrated flux density within a given aperture (30-arcsec radius default) is calculated using PHOTOM autophotom. Flux densities for calibrators such as Uranus, Mars, CRL 618, CRL 2688 and HL Tau are already known to PICARD. To derive an FCF for other sources of known flux densities, the fluxes can be added to the parameter file with the source name (in upper case, spaces removed): `FLUX_450.MYSRC = 0.050` and `FLUX_850.MYSRC = 0.005` (where the values are in Jy), for example.

(3) Three different FCF values are calculated:

(a) **The Arcsecond (Aperture) FCF**

$$\mathrm{FCF}_{\mathrm{arcsec}} = \frac{S_{\mathrm{tot}}}{P_{\mathrm{int}} \times A_{\mathrm{pix}}} \tag{B.1}$$

where S_{tot} is the total flux density of the calibrator, P_{int} is the integrated sum of the source in the map (in pW) and A_{pix} is the pixel area in arcsec^2, producing an FCF in Jy/arcsec^2/pW.

(b) **The Beam (Peak) FCF**

$$\mathrm{FCF}_{\mathrm{beam}} = \frac{S_{\mathrm{peak}}}{P_{\mathrm{peak}}} \tag{B.2}$$

producing an FCF in units of Jy/beam/pW. The measured peak signal here is derived from the Gaussian fit of beamfit. The peak value is susceptible to pointing and focus errors, and we have found this number to be somewhat unreliable, particularly at 450 μm.

(c) **The Beammatch FCF** This FCF is calculated in the same way as the Beam FCF decribed above, but after a matched filter is applied to the data (see Appendix D). This FCF is less commonly used than the former two types.

For a true point source, the measured peak pixel in a map calibrated in units of Jy/beam (using the Beam FCF) is equivalent to the integrated total flux of the same source in a map calibrated in units of Jy/arcsec^2 (using the Arcsecond FCF). The ORAC-DR processing routine will automatically select the appropriate FCF based on the default data-reduction recipe that was linked to your data at the time of observations. Note that the data-reduction recipe can easily be changed when running ORAC-DR (see Section 4.4.2 for details).

B.2 Extinction correction

Starlink automatically applies the following multiplicative extinction correction to SCUBA-2 data:

$$\text{Extinction Correction} = \frac{1}{\exp[-\tau_\nu \times \text{Airmass}]} \tag{B.3}$$

where τ_ν is the opacity at the given frequency, ν.

The atmospheric opacity at SCUBA-2's operating frequencies are defined in terms of the opacity at 225 GHz. Optimizing the uniformity of the SCUBA-2 secondary calibrator fluxes as a function of atmospheric transmission has allowed calculation of the atmospheric opacity relationships for the SCUBA-2 450 μm and 850 μm pass-bands to be determined. Full details of the analysis and on-sky calibration methods of SCUBA-2 can be found in Dempsey et al. (2013) [8][9] with updated relations provided in Mairs et al. (2021) [16].

Archibald et al. (2002) [1] describes how the Caltech Submillimeter Observatory (CSO) 225 GHz opacity, τ_{225}, relates to SCUBA opacity terms in each band, τ_{450} and τ_{850}. The JCMT water-vapour radiometer (WVM) uses the 183 GHz water line to calculate the precipitable water vapour (PWV) along the line-of-sight of the telescope. This PWV is then input into an atmospheric model to calculate the zenith opacity at 225 GHz (τ_{225}). Historically, this has allowed for ease of comparison with the adjacent CSO 225 GHz tipping radiometer.

The updated opacity relationships (to be used in Equation B.3) derived by Mairs et al. (2021) [16] have been adopted as the default as of **Starlink Release 2021A**:

$$\tau_{450} = 23.3 \times (\tau_{225} - 0.018 + 0.05\sqrt{\tau_{225}}); \tag{B.4}$$

and

$$\tau_{850} = 3.71 \times (\tau_{225} - 0.040 + 0.202\sqrt{\tau_{225}}). \tag{B.5}$$

Previously, (**Starlink Versions 2018A and before**) adopted as defaults the opacity relationships found by Dempsey et al. (2013) [8]:

$$\tau_{450} = 26.0 \times (\tau_{225} - 0.012); \tag{B.6}$$

and

$$\tau_{850} = 4.6 \times (\tau_{225} - 0.0043). \tag{B.7}$$

The updated opacity relationships as of Starlink Release 2021A will affect 450-μm data obtained in very dry conditions and 850-μm data obtained in very dry or very wet conditions by up to 5% (see Figure B.1).

Note that the default extinction corrections are intrinsically connected to the default FCFs applied. If applying extinction corrections derived by Mairs et al. 2021 ([16]), the matching FCFs must also be applied (see Appendix E). The ORAC-DR software assumes the Mairs et al. (2021) [16] results beginning in Starlink Release 2021A. Starlink Release 2018A and previous versions assume the Dempsey et al. 2013 [8] values by default.

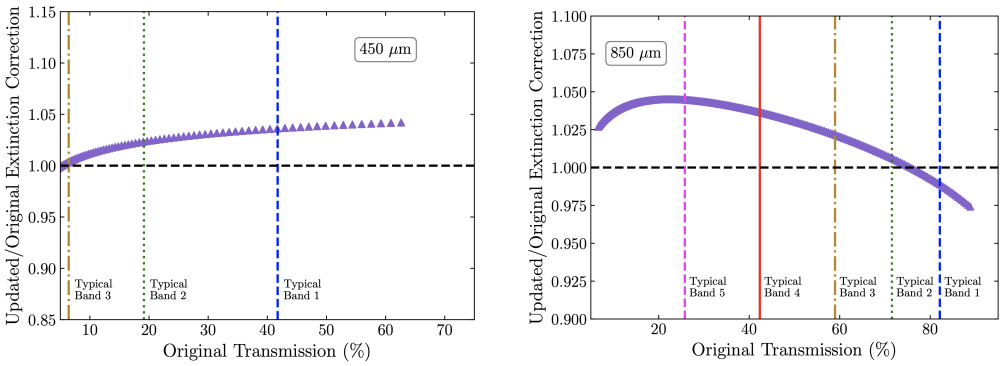


Figure B.1: The new extinction corrections derived by Mairs et al. (2021, [16]) divided by the original extinction corrections (Dempsey et al. 2013 [8]) as a function of atmospheric transmission. Vertical lines show the atmospheric transmission of the typical JCMT weather bands at each wavelength, assuming an airmass of 1.2. Left: 450 μm. Right: 850 μm. At 450 μm, the original correction is modified by a maximum of 5% in very dry weather while at 850 μm the original correction is modified by a maximum of 5% in very dry or very wet conditions. The majority of 850-μm observations, however, require no modification to the original correction and less than 4% of SCUBA-2 data are obtained in weather band 5.

The SCUBA-2 filter characteristics are described in detail on the JCMT website[1].

The extinction correction parameters that scale from τ_{225} to the relevant filter have been added to the map-maker code. You can override these values by setting `ext.taurelation.filtname` in your map-maker config files to the three coefficients '(a,b,c)' that you want to use (following the form $\tau_\nu = a \times (\tau_{225} + b + c\sqrt{\tau_{225}})$), where `filtname` is the name of the filter). The defaults are listed in `$SMURF_DIR/smurf_extinction.def`.

[1] https://www.eaobservatory.org/jcmt/instrumentation/continuum/scuba-2/filters/

Appendix C

SCUBA2_MAPSTATS

The PICARD recipe SCUBA2_MAPSTATS estimates the RMS and NEFD for a given observation.

The average NEP is calculated from the QL pipeline log file (log.nep) corresponding to the date and wavelength. The FCF (either from the file or the standard value) and mean transmission over the observation is used to convert this to a zenith NEFD (NEFD).

The input images are cropped to the given size (as specified in the FITS headers or via the MAP_HEIGHT and MAP_WIDTH recipe parameters) before the mean exposure time is derived, along with the mean/median noise and NEFD (RMS and NEFDs).

The parameters written out to log.mapstats are listed below:

Parameter	Description
UT	UT date including day fraction
HST	HST date and time stamp
Obs	observation number
Source	object name
Mode	observation mode (either daisy or pong)
FILTER	filter (wavelength)
El	mean elevation of observation
Airmass	mean airmass of observation
Trans	mean line-of-sight transmission
Tau225	mean zenith optical depth at 225 GHz, derived from WVM
Tau	mean zenith optical depth at observed frequency
telapsed	elapsed time of observation in seconds
texp	mean exposure time, derived from EXP_TIME NDF component (sec)
rms	RMS noise in map, obtained from median of error array
rms_units	RMS units
nefd	NEFD derived from combination of variance and exposure time images
nefd_units	NEFD units
RA	Right Ascension
Dec	Declination
mapsize	size (requested diameter) of input map (arcsec)
pixscale	pixel scale in arcsec
project	project ID
recipe	reduction recipe used
filename	name of input file

Although the output file log.mapstats can be viewed with any text editor, it is useful to read this file with TOPCAT:

```
topcat -f ascii log.mapstats
```

After running the command above, click on “Views” Then “Table Data” to display the table.

Appendix D
SCUBA-2 Matched Filter

In order to optimally find sources that are the size of the telescope beam, and suppress this residual large-scale noise, the PICARD recipe SCUBA2_MATCHED_FILTER may be used. If there were no large-scale noise in the map, the filtered signal map would be calculated as follows:

$$\mathcal{M} = \frac{[M(x,y)/\sigma^2(x,y)] \otimes P(x,y)}{[1/\sigma^2(x,y)] \otimes [P^2(x,y)]}, \tag{D.1}$$

where $M(x,y)$ and $\sigma(x,y)$ are the signal and RMS noise maps respectively produced by SMURF, and $P(x,y)$ is a map of the PSF. Here $\otimes$ denotes the 2-dimensional cross-correlation operator. Similarly, the variance map would be calculated as

$$\mathcal{N}^2 = \frac{1}{[1/\sigma^2(x,y)] \otimes [P^2(x,y)]}. \tag{D.2}$$

This operation is equivalent to calculating the maximum-likelihood fit of the PSF centered over every pixel in the map, taking into account the noise. Presently $P(x,y)$ is simply modelled as an ideal Gaussian with a FWHM set to the diffraction limit of the telescope.

However, since there is large-scale (and therefore correlated from pixel to pixel) noise, the recipe also has an additional step. It first smooths the map by cross-correlating with a larger Gaussian kernel to estimate the background, and then subtracts it from the image. The same operation is also applied to the PSF to estimate the effective shape of a point-source in this background-subtracted map.

Before running PICARD, a simple parameters file called `smooth.ini` may be created.

```
[SCUBA2_MATCHED_FILTER]
SMOOTH_FWHM = 15
```

where `SMOOTH_FWHM = 15` indicates that the background should be estimated by first smoothing the map and PSF with a 15 arcsec FWHM Gaussian. The recipe is then executed as follows:

```
% picard -recpars smooth.ini SCUBA2_MATCHED_FILTER map.sdf
```

The output of this operation is a smoothed image called `map_mf.sdf`. By default, the recipe automatically normalizes the output such that the peak flux densities of point sources are conserved. Note that the accuracy of this normalization depends on how closely the real PSF matches the 7.5 arcsec and 14 arcsec full-width at half-maximum (FWHM) Gaussian shapes assumed at 450μm and 850μm, respectively (an explicit PSF can also be supplied using the `PSF_MATCHFILTER` recipe parameter).

Appendix E
FCFs by Reduction Date

Ongoing development of the SCUBA-2 analysis has resulted in ongoing changes to the atmospheric opacity relationships and the FCFs. Depending on when your data were reduced you will need to apply different calibration values. ORAC-DR will automatically apply the appropriate values.

Note: As of the 2021A Starlink release, the opacity relationships and FCF values have been updated following the results presented by Mairs et al. 2021 [16]. Historical values derived by Dempsey et al. 2013 [8] are presented below the updated values for comparison.

Explanation of parameters (see also Appendix B):

Starlink currently applies the following multiplicative extinction correction to SCUBA-2 data:

$$\text{Extinction Correction} = \frac{1}{\exp[-\tau_\nu \times \text{Airmass}]} \tag{E.1}$$

where τ_ν is the atmospheric opacity at the given frequency, ν. "Opacity relationships" relate the measured τ_{225} to the atmospheric opacity at the operating frequencies of SCUBA-2, τ_{666} (450 μm) and τ_{345} (850 μm). Their form is:

$$\tau_\nu = a \times (\tau_{225,\text{zen}} + b + c \times \sqrt{\tau_{225,\text{zen}}}), \tag{E.2}$$

where a, b, and c are empirically derived coefficients (see Mairs et al. 2021 [16]). You can find out what opacity relationship was applied by using the KAPPA command hislist.

```
% hislist file | grep EXT.TAURELATION
```

This will return something like the following,

```
EXT.TAURELATION.450=(23.3,-0.018,0.05),
EXT.TAURELATION.850=(3.71,-0.040,0.202), EXT.TAUSRC=auto, FAKESCALE=1,
```

indicating, for example, that at 850 μm, a=3.71, b=-0.040, and c=0.202.

There are two commonly used FCF types:

- Peak FCF (Jy/pW/beam)—multiply your map by this when you wish to measure absolute peak fluxes of discrete sources.
- Arcsec FCF (Jy/pW/arcsec^2)—multiply your map by this if you wish to use the calibrated map to do aperture photometry/extended source flux recovery.

Note: The FCFs are applied after the extinction correction, so the values are intrinsically related to one another. The proper extinction correction must be applied before using the FCF values presented below.

Results from Mairs et al. 2021 [16]. These are employed by default from Starlink Version 2021A. Dates are inclusive. See Figure E.1.

Date	FCF – 450 μm		FCF – 850 μm	
	Jy/pW/beam	Jy/pW/arcsec^2	Jy/pW/beam	Jy/pW/arcsec^2
until 2011 April 30	383	4.9	1080	5.0
2011 May 1–2016 November 1	531 ± 93	4.61 ± 0.60	525 ± 37	2.25 ± 0.13
2016 November 2–2018 June 29	531 ± 93	4.61 ± 0.60	516 ± 42	2.13 ± 0.12
2018 June 30 onwards	472 ± 76	3.87 ± 0.53	495 ± 32	2.07 ± 0.12

From 2011 May 1 onwards, the a, b, and c coefficients presented in the table below correspond to those in Equation E.2.

Date	Opacity Relationship	
	450 μm	850 μm
until 2011 April 30	32 $\times$ (τ_{225} - 0.02)	5.2 $\times$ (τ_{225} - 0.013)
2011 May 1 onwards	a = 23.3$\pm$1.5 b = -0.018$\pm$0.006 c = 0.05$\pm$0.04	a = 3.71$\pm$0.18 b = -0.040$\pm$0.008 c = 0.202$\pm$0.044

Results from Dempsey et al. 2013 [8]: employed by default in Starlink Version 2018A and previous releases.

Date	FCF – 450 μm		FCF – 850 μm	
	Jy/pW/beam	Jy/pW/arcsec^2	Jy/pW/beam	Jy/pW/arcsec^2
until 2012 January	383	4.9	1080	5.0
2012 January–2012 July	606	6.06	556	2.42
2012 July onwards	491	4.71	537	2.34

Date	Opacity Relationship	
	450 μm	850 μm
until 2012 January	32 $\times$ (τ_{225} - 0.02)	5.2 $\times$ (τ_{225} - 0.013)
2012 January–2012 July	26 $\times$ (τ_{225} - 0.01923)	4.6 $\times$ (τ_{225} - 0.00435)
2012 July onwards	26 $\times$ (τ_{225} - 0.01196)	4.6 $\times$ (τ_{225} - 0.00435)

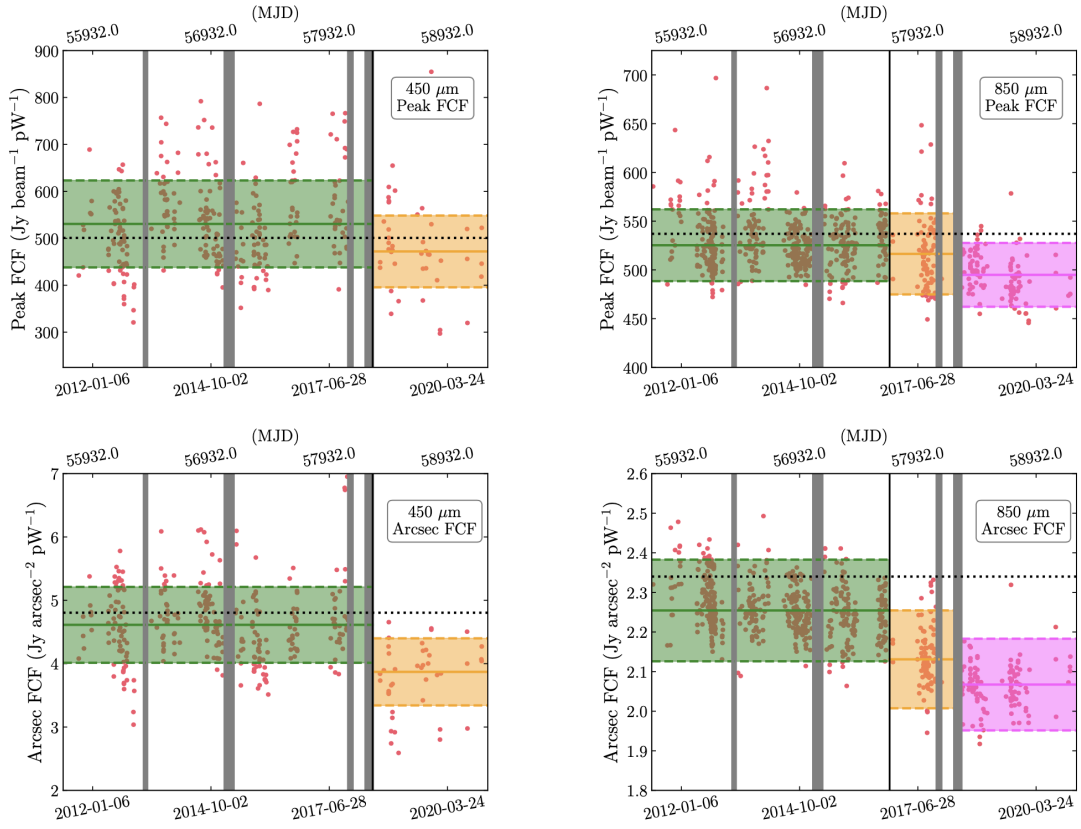


Figure E.1: From Mairs et al. (2021) [16]. FCFs derived using flux measurements of the primary calibrator Uranus during the stable part of the night (07:00-17:00 UTC) as a function of date. The grey shaded regions indicate epochs that are not included in the FCF determinations. The horizontal, shaded regions indicate the median FCF value over each span of time and the associated median absolute deviation added in quadrature with the 5% uncertainty in the Uranus flux model. The black (dotted) lines indicate the original FCF value derived by Dempsey et al. (2013) [8], adjusted for the newly derived opacity relationships, assuming the most-common atmospheric transmissions during observations (see Figure 2 of Mairs et al. (2021). Left: Peak (Top) and Arcsecond (Bottom) FCFs derived at 450 μm. The solid, vertical line at the right edge of the latest grey region marks 2018 June 30 when the secondary-mirror-unit maintenance was completed. Data wherein the atmospheric transmission are less than 10% are excluded. Right: Peak (Top) and Arcsecond (Bottom) FCFs derived at 850 μm. The solid, vertical line marks 2016 November, when the SCUBA-2 thermal-filter stack was updated. Data wherein the atmospheric transmission are less than 25% are excluded.

Appendix F
FCFs by Time of Night

Figure F.1 shows the Beam (Peak) FCFs at 450 and 850 μm as a function of UT time for observations of the primary calibrator, Uranus along with secondary calibrators CRL 2688, and CRL 618. The Peak FCF is larger in the evening and morning primarily because thermal deformations of the dish dilute the flux from the main beam into the secondary (error) component (see Mairs et al. 2021 [16]). There is no significant change to the Arcsecond FCFs in the early evening or late morning.

Figure F.2 show the Peak FCF trends in detail for evening and morning observations of Uranus and CRL 2688. The data are bootstrap-fitted with linear functions and "rs" indicates the Spearman rank correlation of the fit.

The Peak FCFs DECREASE in the evening as the ambient temperature cools and the dish settles, while the Peak FCFs INCREASE in the morning as the ambient temperature warms and the dish becomes unstable to thermal gradients. Table F summarises the rate of change. As of Starlink Release 2021A, ORAC-DR **does not** apply these corrections by default. If you wish to apply these corrections, the FCFs must be modified manually (see Section 8.1.3).

Wavelength	Time Range (UTC)	Peak FCF Correction (% hr^{-1})
450 μm	03:00–07:00	9.1 ± 0.5
450 μm	17:00–22:00	7.2 ± 0.6
850 μm	03:00–07:00	3.2 ± 0.1
850 μm	17:00–22:00	3.1 ± 0.1

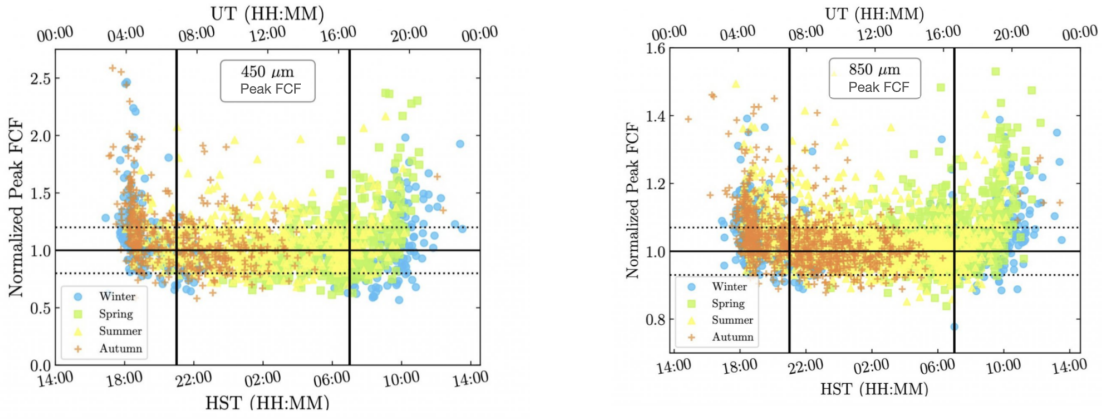


Figure F.1: The Normalized Peak FCFs at 450 (left) and 850 μm (right) as a function of observation time. All FCFs are derived using the primary calibrator Uranus and secondary calibrator CRL 2688. The vertical lines mark the beginning and end of the "stable" observation period from 07:00—17:00 (UTC). The horizontal (dotted) lines show the FCF uncertainties derived for the stable observation period around a value of 1.0 (horizontal, solid line). Data are coloured according to season.

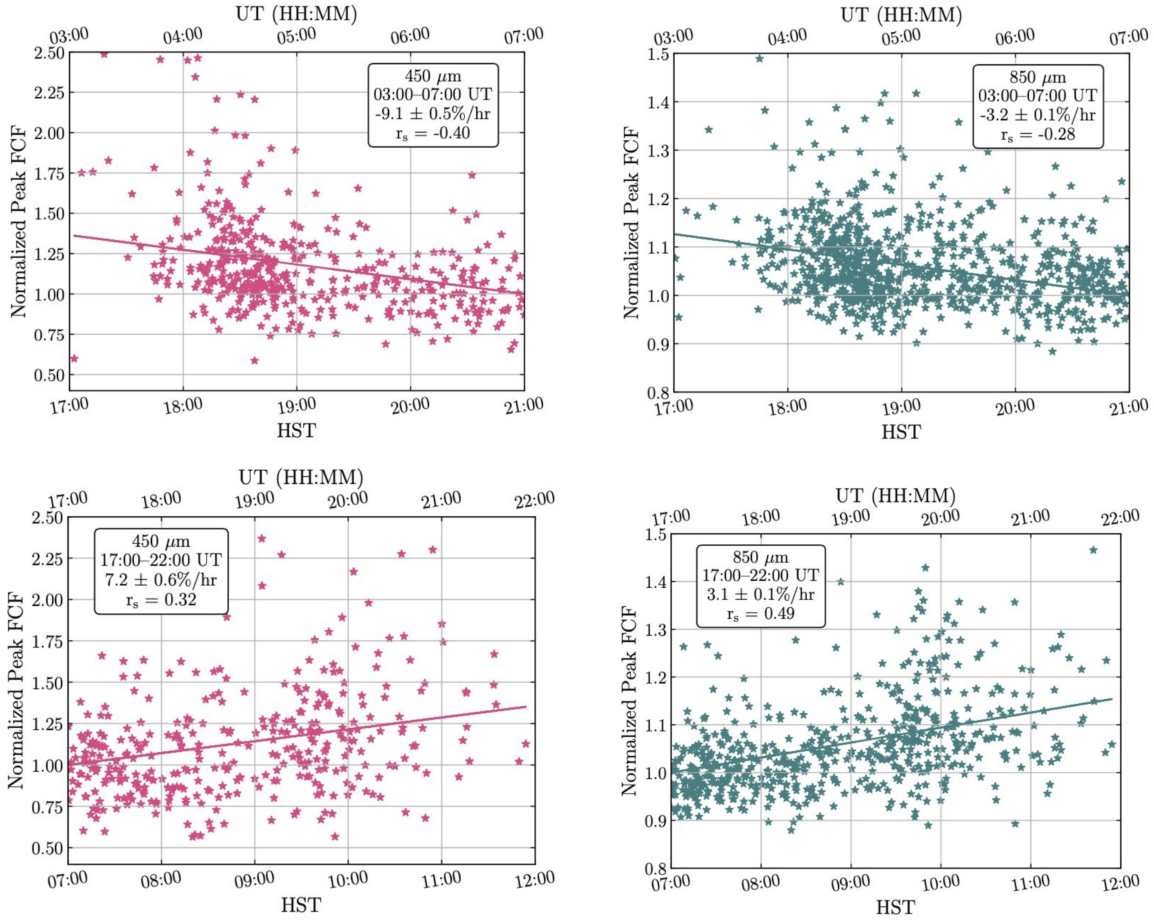


Figure F.2: Linear fits to the Peak FCFs at 450 μm (left) and 850 μm (right) during the evening (top) and morning (bottom) when dish distortions caused by thermal gradients affect the beam shape. The distributions during the stable part of the night (07:00–17:00 UTC) show no significant slope and therefore require no corrections to the default FCFs shown in Appendix E.

Appendix G
Aperture-photometry Curve of Growth

The SCUBA-2 beam has a broad error beam. As the size of the annulus changes, the contribution from the error beam scales according to the curve-of-growth—see Figure G.1. To correct for an aperture size differing from 60-arcsec diameter you should read off the appropriate scaling factor for your FCF from the graph below.

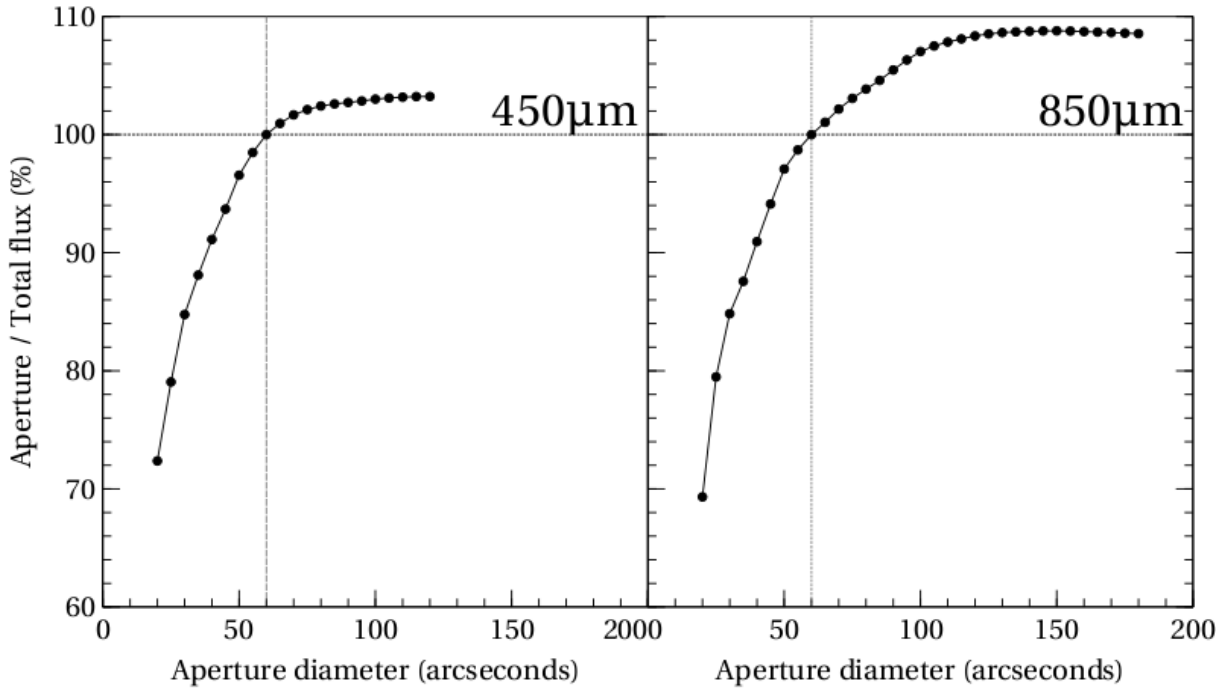


Figure G.1: Aperture photometry curve of growth normalised for a 60-arcsec aperture at 450μm **(left)** and 850μm **(right)**. Figure taken from Dempsey et al. (2013).

Appendix H
Convert Format from FITS to NDF

It is often useful to utilise data from other wavelengths or instruments (either for a comparison or for an external mask). In the following example, the FITS file called `file.fits` is converted to NDF format as `file.sdf` using the Starlink package CONVERT. Note that the `.sdf` file extension NDF may be omitted to save typing.

```
% convert
% fits2ndf file.fits file.sdf
```

FITS files from certain recognised sources have special rules applied when converting from FITS to NDF, as described in the documentation for fits2ndf. For FITS files from other sources, the primary array in the FITS file is stored as the main NDF in the output file. Any FITS extensions present in the FITS file will be placed into NDF extensions called FITS_EXT_$< n >$, where n counts from one for the first FITS extension. To see a list of the extension NDFs in `fred.sdf`, do:

```
% ndfecho fred.more
fred.MORE.FITS_EXT_1
fred.MORE.FITS_EXT_2
fred.MORE.FITS_EXT_3
```

When running a KAPPA ot SMURF command, you can refer to these extension just as they are listed above. So for instance:

```
% ndftrace fred.MORE.FITS_EXT_1

   NDF structure /home/dsb/fred.MORE.FITS_EXT_1:
      Units:  COUNTS/S

   Shape:
      No. of dimensions:  2
      Dimension size(s):  270 x 263
      Pixel bounds     :  1:270, 1:263
      Total pixels     :  71010
...
...
```

Alternatively, you can copy the NDF into its own separate file:

```
% ndfcopy in=fred.MORE.FITS_EXT_1 out=new_file
```

If one of the extensions contains a variance array that you would like to as the `Variance` component of the main NDF, a command like the following will do that:

```
% setvar ndf=new_file from=fred.MORE.FITS_EXT_2
```

The fits2ndf command offers a way of mapping FITS extensions to familiar NDF array components DATA, VARIANCE, and QUALITY through the `EXTABLE` file, avoiding the ndfcopy and possible setvar steps.

You can convert an NDF to a FITS file using the command ndf2fits:

```
% convert
% ndf2fits file.sdf file.fits
```

Appendix I
Configuration-parameter Descriptions

This appendix describes some of the makemap configuration parameters that are more likely to be of interest to a typical user. Thus parameters relating to experimental or deprecated features, or features that should not normally need to be changed, are not included here. For a complete list of all available configturation parameters, see SUN/258.

- The "SMURF Usage" section of each description lists the SMURFcommands that accept the parameter. This does not mean that all the listed commands actually *use* the parameter—it just means that any commands *not* in the list will report an error if the parameter is supplied.
- The default value for each parameter is shown in square brackets at the end of the description. These default values are defined in the file `$SMURF_DIR/smurf_makemap.def` and are the values that are used for any parameters that are not assigned a value within the configuration file supplied to makemap. In other words, any parameter values supplied within a configuration file will be used instead of these default values.

AST.MAPSPIKE
Removes spikes from the map

Description:

If ast.mapspike is non-zero, spikes in the time-series residuals will be identified by looking at the spread of residual values that contribute to each map pixel. Any residuals that are above ast.mapspike standard deviations from the mean value in the pixel are flagged as spikes. Parameter "ast.mapspike_freeze" can be used to control the number of iterations for which this de-spiking is applied. [10.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.SKIP
Skip subtraction of astronomical signal

Description:

If ast.skip is non-zero, it gives the number of initial iterations for which no AST model (astronomical signal) should be subtracted. A map is still formed at the end of these iterations but the astronomical signal implied by this map is not removed from the residuals (neither is it added back prior to forming the next map). This means that the residuals at the start of each of these iterations are unchanging and essentially equal to the cleaned raw data. The value supplied for parameter "numiter" should therefore be greater than the value of " ast.skip" .

This option is useful, for instance, when using SNR-based masking for the FLT and/or COM models, since it allows a reasonable mask to be formed before subtracting off the first estimate of the astronommical signal.

If ast.skip is set to a negative value, it gives the largest number of iterations to perform and indicates that the AST model should be skipped on all of them. In this case the value supplied for parameter "numiter" is ignored. [0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_CIRCLE
Reduces spurious large scale structure in the final map outside a circle of given radius

Description:

Using ast.zero_circle defines a circle on the map outside of which the map will be constrained to zero on each iteration (but see parameter "ast.zero_notlast"). If a value is supplied for this parameter, it can be a single real value, or a comma-separated list of three real values in parentheses. If one value is supplied, it should be the radius of the circle in decimal degrees (the centre of the circle defaults to the coordinates at the tangent point of the map). If three values are supplied they should be the central longitude, latitude and radius of the circle, in decimal degrees, in the coordinate system of the map (e.g., RA and Dec.). [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_FREEZE
Prevent the AST mask from changing after a given number of iterations. This can help convergence

Description:

If ast.zero_freeze is 1.0 or more, the AST mask will be frozen after the specified number of iterations (the nearest integer value is used). Note, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. If ast.zero_freeze is greater than zero but less than 1.0, the AST mask will be frozen when the normalized change in the map between iterations drops below the ast.zero_freeze value. A value less than zero means that the mask is frozen as soon as the initial iterations specified by parameter "ast.skip" have been done. A value equal to zero means that the mask is never frozen. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_LOWHITS
Reduces spurious large scale structure in the final map in regions containing few data samples

Description:

Using ast.zero_lowhits causes the map to be forced to zero in regions where the number of samples falling in each pixel is less than ast.zero_lowhits times the mean number of samples per pixel, averaged over the map. A value of zero means that no masking of low hits regions is performed. The mask is updated on each iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_MASK
Reduces spurious large scale structure in the final map within fixed regions specified by an external mask

Description:

If ast.zero_mask is set to one of " REF" , " MASK2" or " MASK3" then an NDF will be obtained using the specified ADAM parameter (REF, MASK2 or MASK3) and used as a user-defined mask. Setting ast.zero_mask to an integer value larger than zero has the same effect as setting it to " REF" . Setting it to an integer less than or equal to zero results in no external mask being used with the AST model. Note, using " REF" ensures that the mask and the output image of MAKEMAP are on the same pixel grid - using " MASK2" or " MASK3" does not provide this guarantee (it is then the users responsibility to ensure that the supplied masks are aligned with the output image in pixel coordinates). The pixels in the map that are to be constrained to 0 should be set to the bad value in the mask. All other pixels will be allowed to vary during map-making. The mask for the first iteration can be set separately using parameter "ast.zero_mask0" . [0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_NITER
Allows AST masking to be switched off after a given number of iterations

Description:

If ast.zero_niter is non-zero, it determines the number of iterations for which the AST model should be masked. Subsequent iterations are not masked. A value of zero means " mask on all iterations" . However, if parameter "ast.zero_notlast" is set, the mask will will not be applied on the last iteration, even if ast.zero_niter is zero. This feature will probably be useful for deep point-source observations for which the large-scale noise is not as important, but keeping as much data around the edges of the map is. Note, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. If ast.zero_niter is greater than zero but less than 1.0, the AST mask will be used until the normalized change in the map between iterations drops below the ast.zero_freeze value. A negative value for ast.zero_niter will disable all masking of the AST model. If a value is set for " ast.zero_niter" , the iterative map-making process will not terminate until the number of iterations required by the supplied " ast.zero_niter" value have occurred, regardless of whether the condition specified by parameter "maptol" is achieved earlier. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_NOTLAST
Allows flux to be present in masked areas in the final map

Description:

If ast.zero_notlast is 1, then the map is not masked on the final iteration. This means that data samples that fall outside the masked areas are allowed to remain in the final map. If ast.zero_notlast is 0, then the map is masked even on the final iteration, meaning that the masked areas will be zero in the final map. [1]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNR
Reduces spurious large scale structure in the final map within regions of low signal-to-noise

Description:

The ast.zero_snr parameter will mask the map after each iteration based on the the signal to noise ratio within each map pixel. For example, if it is set to 5, after each iteration all map pixels with an SNR below this threshold will be forced to zero. the mask is re-evaluated on each iteration. An ast.zero_snr value of zero means no SNR mask is used. See also parameter "ast.zero_snr_ffclean" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNR_FFCLEAN
Provides alternative method for SNR masking

Description:

Setting this parameter to a non-zero value causes the SNR mask requested by parameter "ast.zero_snr" to be created using an algorithm like that used by the KAPPA command " FFCLEAN" (see SUN/95), instead of using a simple thresholding of the SNR map. The parameter "ast.zero_snr" gives the clipping level of the ffclean algorithm, and the parameter "ast.zero_snr_hipass" gives the box size. Using an ffclean algorithm prevents the source regions within the mask being larger than the box size, and may thus produce faster convergence and avoid blobs developing. [0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNR_FWHM
Can help to remove bowls around sources

Description:

If ast.zero_snr_fwhm is non-zero, the map-maker produces two maps: the first is created normally using the mask specified by parameter "ast.zero_snr" . The final SNR-based mask associated with this map is then smoothed using a Gaussian with FWHM equal to the ast.zero_snr_fwhm value (in arcsec). The whole iterative map-making process is then run again from the start, using this smoothed mask on every iteration, to create the final map. Consequently, setting ast.zero_snr_fwhm causes the time taken to create the final map to nearly double. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNR_LOW
Can help to remove bowls around sources

Description:

The ast.zero_snr_low parameter gives the value (in the range 0.0 to 1.0) at which to threshold the smoothed mask specified by parameter "ast.zero_snr_fwhm" . If it is negative, the value is taken as the max smoothed value of a blob containing " ast.zero_snr_low" pixels. Thus a value of " -1.1" will cut at a height just sufficient to remove blobs of a single pixel form the mask. A value of " -2.1" would remove blobs of two pixels form the mask, etc. [-1.1]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNRLO
Can help to remove bowls around sources by increasing the size of the SNR mask without introducing noise

Description:

If values are supplied for ast.zero_snrlo and parameter "ast.zero_snr" , then the basic mask created by thresholding at the SNR value specified by ast.zero_snr is modified by expanding each unmasked " source" area down to an SNR equal to ast.zero_snrlo, without introducing any new isolated source areas. The ast.zero_snrlo should be lower than the ast.zero_snr value. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_UNION
Controls how multiple AST masks are combined

Description:

If more than one AST mask is specified (for instance, if values are supplied for both parameter "ast.zero_lowhits" and parameter "ast.zero_snr"), then they are combined into a single mask. If ast.zero_union is true (i.e. non-zero), then the source region in the combined mask is the union of the source regions in the individual masks. If ast.zero_union is false (i.e. zero), then the source region in the combined mask is the intersection of the source regions in the individual masks. [1]

SMURF Usage :

MAKEMAP, CALCQU

BADFRAC
Ensures that bad data from DA system are ignored

Description:

The fraction of samples to be bad to flag entire bolo as dead. [0.05]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

BOLOMAP
Create NDFs holding the map made form each bolometer

Description:

If non-zero, a separate map will be created from each individual bolometer. These maps are placed in the BOLOMAPS component of the SMURF extension in the main output map. [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.CORR_ABSTOL
Controls the rejection of bad samples from the COM estimate

Description:

Gives the absolute lower limit of acceptable correlation between a bolometer time-stream and the common-mode. This is the first of a set of " com.<xxx>" parameters that control the rejection of bad detectors based on the gain and correlation coefficients for the fit of the common-mode signal to each detector (good at identifying bolo signals with bizarre gains, or shapes if they have for example steps in them). These are basically sigma-clippers; outliers are removed at the given threshold and then new means and sample standard deviations are measured until convergence. The time axis is divided up into one or more equal sized boxes, and a separate fit is performed for each box. If you wish to completely disable the flagging of outlier bolometers compared with the common-mode, simply set com.noflag=1. The flags may be frozen after a specified number of iterations - see parameter "com.freeze_flags" . [0.2]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.CORR_TOL
Controls the rejection of bad samples from the COM estimate

Description:

The maximum number of standard deviations away from the mean correlation coefficient that a bolometer can be without being rejected. See parameter "com.corr_abstol" . [5.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.FREEZE_FLAGS
Controls flagging of samples that differ from the common-mode

Description:

If non-zero, the flags marking bolometers that differ from the common-mode are frozen after the specified number of iterations. this can help convergence. See parameter "com.corr_abstol" . Note - any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_ABSTOL
Controls the rejection of bad samples from the COM estimate

Description:

The maximum absolute ratio between a bolometer's gain coefficient, and the mean gain coefficient for the bolometer not to be rejected. See parameter "com.corr_abstol" . [3.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_BOX
Controls the rejection of bad samples from the COM estimate

Description:

The number of time slices (or seconds if negative) in a box. The gain, offset and correlation coefficient describing the relationship between a bolometer time stream and the common-mode is re-evauated for each such box of time slices. See parameter "com.corr_abstol" . [-30.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_FGOOD
Controls the rejection of bad samples from the COM estimate

Description:

The minimum fraction of good gain boxes for a usable bolometer. See parameter "com.corr_abstol" . [0.25]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_RAT
Controls the rejection of bad samples from the COM estimate

Description:

The ratio of the largest usable gain to the mean gain for a bolometer not to be rejected. See parameter "com.corr_abstol" . [4.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_TOL
Controls the rejection of bad samples from the COM estimate

Description:

The maximum number of standard deviations away from the mean gain coefficient that a bolometer can be without being rejected. See parameter "com.corr_abstol" . [5]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.NOFLAG
Controls the rejection of bad samples from the COM estimate

Description:

If non-zero, disable flagging of bad bolometers using the common-mode. See parameter "com.corr_abstol" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.PERARRAY
Controls the estimate of the common-mode signal

Description:

If non-zero, calculate a separate common-mode signal for each subarray. If zero, a single common-mode signal will be calculated from all subarrays at a given wavelength simultaneously. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.SIG_LIMIT
Controls the rejection of time slices with inconsistent common-modes

Description:

This value is only used if data from more than one sub-array is being included in the map, and parameter "com.perarray" is zero, causing a single mean COM model to be used for all sub-arrays. In such cases, if the common-mode signals estimated from the individual sub-arrays show significantly different structure, subtracting a mean COM model will leave large residuals in the bolometer time streams. If a FLT model is being used, the high pass filter will remove much of this residual signal, but any high frequency component will remain, often causing strong " blobs" in the map.

The COM model can identify and flag time slices where the individual common-mode signals appear to be significantly different to the mean common-mode signal. The individual common-mode signals are first high-pass filtered using the same filter as the FLT model. At each time slice, the mean and standard deviation of the remaining high frequency common-modes are found. The RMS of these standard deviations, taken over the whole time stream, is then found. All bolometers within individual time slices for which the standard deviation exceeds " com.sig_limit" times the RMS value are flagged in all sub-arrays, causing them to be excluded from the FLT model and the map. Note, this flagging process is performed from scratch on each iteration - that is, samples flagged on a previous iteration are tested again on each subsequent iteration and may be " unflagged" if the differences between individual common-mode signal become smaller.

Setting this parameter to 3 is a good starting point. Setting it to zero (the default) causes this consistency filter to be skipped. See also parameter "com.sig_wing" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.ZERO_CIRCLE
Improves common-mode estimation by excluding sources within a circle of given radius from the COM estimate

Description:

Using com.zero_circle causes any samples falling within a specified circle on the map to be excluded from the estimate of the mean signal at each time slice (the common mode, or " COM" , signal). [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_FREEZE
Prevent the COM mask from changing after a given number of iterations. This can help convergence

Description:

If com.zero_freeze is 1.0 or more, the COM mask will be frozen after the specified number of iterations (the nearest integer value is used). Note, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. If com.zero_freeze is greater than zero but less than 1.0, the COM mask will be frozen when the normalized change in the map between iterations drops below the com.zero_freeze value. A value less than zero means that the mask is frozen as soon as the initial iterations specified by parameter "ast.skip" have been done. A value equal to zero means that the mask is never frozen. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_LOWHITS
Improves common-mode estimation by excluding sources in regions containing many data samples

Description:

Using com.zero_lowhits causes samples to be excluded from the estimation of the common mode if they fall in regions of the map where the number of samples falling in each pixel is higher than com.zero_lowhits times the mean number of samples per pixel, averaged over the map. A value of zero means that no masking of low hits regions is performed. The mask is updated on each iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_MASK
Provides a better estimate of the common-mode (" COM") signal, by excluding samples that fall within fixed regions on the sky specified by an external mask

Description:

If com.zero_mask is set to one of " REF" , " MASK2" or " MASK3" then an NDF will be obtained using the specified ADAM parameter (REF, MASK2 or MASK3) and used as a user-defined mask. Setting com.zero_mask to an integer value larger than zero has the same effect as setting it to " REF" . Setting it to an integer less than or equal to zero results in no external mask being used with the COM model. Note, using " REF" ensures that the mask and the output image of MAKEMAP are on the same pixel grid - using " MASK2" or " MASK3" does not provide this guarantee (it is then the users responsibility to ensure that the supplied masks are aligned with the output image in pixel coordinates). The pixels in the map that are to be included in the common-mode estimation should be set to the bad value in the mask. All other pixels will be excluded from the COM estimation. The mask for the first iteration can be set separately using parameter "com.zero_mask0" . [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_SNR
Improve the estimate of the common-mode by excluding samples that correspond to high SNR pixels in the map

Description:

Setting the com.zero_snr parameter will exclude samples from the COM estimate that fall within map pixels with SNR values greater than com.zero_snr. A com.zero_snr value of zero means no SNR mask is used. See also parameter "com.zero_snr_ffclean" .

Note, the SNR values are only available once a map has been created, and so using this parameter results in no COM masking on the first iteration. Consequently the map at the end of the first iteration will have a bowl around any bright sources, since no COM masking was done. Normally, these rings would polute the AST model derived from the map, and thus polute the residuals on the next iteration, resulting in the bowls remaining in later maps. To avoid this, parameter "ast.skip" can be set to a positive value. This causes the AST model to be skipped (i.e. no AST signal is subtracted from the residuals) for the first " ast.skip" iterations. This means that a good COM mask can be formed from these initial iterations before any AST model is calculated and used. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_SNR_FFCLEAN
Provides alternative method for SNR masking

Description:

Setting this parameter to a non-zero value causes the SNR mask requested by parameter "com.zero_snr" to be created using an algorithm like that used by the KAPPA command " FFCLEAN" (see SUN/95), instead of using a simple thresholding of the SNR map. The parameter "com.zero_snr" gives the clipping level of the ffclean algorithm, and the parameter "com.zero_snr_hipass" gives the box size. Using an ffclean algorithm prevents the source regions within the mask being larger than the box size. [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_SNRLO
Improve estimate of the common-mode by increasing the size of the SNR mask without introducing noise

Description:

If values are supplied for com.zero_snrlo and parameter "com.zero_snr" , then the basic mask created by thresholding at the SNR value specified by com.zero_snr is modified by expanding each un-masked " source" area down to an SNR equal to com.zero_snrlo, without introducing any new isolated source areas. The com.zero_snrlo should be lower than the com.zero_snr value. [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_UNION
Controls how multiple COM masks are combined

Description:

If more than one COM mask is specified (for instance, if values are supplied for both parameter "com.zero_lowhits" and parameter "com.zero_snr"), then they are combined into a single mask. If com.zero_union is true (i.e. non-zero), then the source region in the combined mask is the union of the source regions in the individual masks. If com.zero_union is false (i.e. zero), then the source region in the combined mask is the intersection of the source regions in the individual masks. [1]

SMURF Usage :

MAKEMAP, CALCQU

COMPREPROCESS
Remove common-mode before the iterative algorithm begins

Description:

If non-zero, the common-mode will be estimated and removed additionally as a pre-processing step. All the " com.<xxx>" and " gai.<xxx>" parameters are parsed and used (e.g., to also flag bad data and optionally flatfield off the relative response to the common-mode signal). If this pre-processing step is chosen, it is still possible to specify COM/GAI as model components in the iterative solution. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DCFITBOX
Control the cleaning of DC steps in bolometer time streams

Description:

This gives the box size over which to fit data with a straight line on either side of a potential DC jump, prior to estimating the bolometer noise levels when doing initial data cleaning. If positive, in units of samples. If negative, in units of seconds. If zero, do not perform step correction during initial data cleaning. [30]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DCLIMCORR
Control the cleaning of DC steps in bolometer time streams

Description:

If more than DCLIMCORR bolometer have a step at a given time, then all bolometers are corrected for a step at that time, using lower thresholds. A value of zero switches off the correction of correlated steps within the initial data cleaning phase. Only used if parameter "dcfitbox" is non-zero. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DCMAXSTEPS
Control the cleaning of DC steps in bolometer time streams

Description:

The maximum number of steps that can be corrected in each minute of good data (taking any down-sampling into account) from a bolometer before the entire bolometer is flagged as bad. A value of zero will cause a bolometer to be rejected if any steps are found in the bolometer data stream. Only used if parameter "dcfitbox" is non-zero. [4]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DCSMOOTH
Control the cleaning of DC steps in bolometer time streams

Description:

The width of the median filter used to smooth a bolometer data stream prior to finding DC jumps. If positive, in units of samples. If negative, in units of seconds. Only used if parameter "dcfitbox" is non-zero. [50]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DCTHRESH
Control the cleaning of DC steps in bolometer time streams

Description:

The SNR threshold at which to detect DC steps. Note, this refers to the noise level in the bolometer data after it has been smoothed with a median filter of width given by parameter "dcsmooth" . In order to find the equivalent threshold in the unsmoothed data, multiply the dcthresh value by 1.25/sqrt(dcsmooth). For instance, the default values for dcsmooth (50) and dcthresh (25) correspond to a threshold of 25∗1.25/sqrt(50) = 4.4 sigma in the unsmoothed data. Only used if parameter "dcfitbox" is non-zero. [25.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DIAG.OUT
Switches on the dumping of various diagnostic information

Description:

The full path/name for the HDS container file in which to store the diagnostic info. This will contain components for each requested model (see parameter "diag.models"), with names like " COM" , " FLT" , etc. Each of these components will contain multiple NDFs with names in the following format: " <where>_<chunk>_<what>" , where <what> is " power" or " time" , <chunk> is the integer chunk index, and <where> is one of:

- " bef" : the NDF contains the residuals as they were before the model was subtracted.
- " mod" : the NDF contains the model values themselves.
- " aft" : the NDF contains the residuals as they were after the model was subtracted.

Each NDF will be 2-dimensional, with the first pixel axis representing time or frequency, and the second pixel axis representing iteration number. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

DOCLEAN
Allows pre-cleaned data to be used

Description:

Set this to 0 to turn off all data cleaning operations prior to the start of iterative map-making. [1]

SMURF Usage :

MAKEMAP, CALCQU

DOWNSAMPSCALE
Speeds up map-making, and reduces memory requirements

Description:

If the telescope is scanning slowly the data may be safely down-sampled to save memory and time. This parameter controls the minimum angular scale on the sky. The new sample frequency is chosen such that this scale will be preserved taking into account the average slew speed and the sample rate of the input files. If a positive value is selected, this gives the angular scale (in arcsec) to which the new sample rate will be matched. Alternatively, if a negative value is supplied, its magnitude will be multiplied by the PIXSIZE for the requested map. For example, the default here is to set it to -1 such that the time-series sample rate matches the pixel grid (in practice, a factor of 2 might make more sense as this would correspond to the Nyquist frequency of the map pixel grid). [-1]

SMURF Usage :

MAKEMAP, CALCQU

EXPORTCLEAN
Allows the initial cleaned data to examined or saved for later use

Description:

If non-zero, the data will be saved to an NDF immediately after data cleaning and before map-making. The NDF name will be the same as model components, except with the suffix " _cln" . Even if parameter "doclean" is set to zero, the data will be exported immediately before map-making. [0]

SMURF Usage :

MAKEMAP, CALCQU

EXPORTNDF
Create NDFs holding final model values

Description:

Specify a value of 1 or 0 to export all or none of the model components after the final iteration. You can also specify a comma-separated list of component names, enclosed in parentheses, to be exported. Note that you can specify additional components RES and QUA to what may be provided to parameter "modelorder" if you wish to export the residual model or quality arrays respectively. Exportation of RES is implied if NOI is specified as it becomes the variance component of the resulting NDF for RES. QUA will become the quality component of any full 3-dimensional model (e.g. RES, AST, FLT, EXT), but no quality will be written to model components with different dimensions. Note, since an NDF can only contain 8 quality flags, it may be necessary to compress quality information by combining flags with similar purposes together. To avoid this, it is possible to use parameter "exportqbits" to specify that only a subset of the quality flags be written out t the NDF. [0]

SMURF Usage :

MAKEMAP, CALCQU

EXT.CSOTAU
Controls the extinction values used in the EXT model

Description:

Specifies the CSO tau value to be used by the EXT model. If <undef>, the default value to use is derived from the FITS headers. See parameter "ext.tausrc" . [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

EXT.FILTERTAU
Controls the extinction values used in the EXT model

Description:

Used if parameter "ext.tausrc" is set to " filtertau" . If <undef>, the default value to use is derived from the FITS headers. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

EXT.TAUMETHOD
Controls the extinction values used in the EXT model

Description:

The method to use for determing tau. Can be " adaptive" , " full" or " quick" . See parameter "ext.tausrc" . [adaptive]

SMURF Usage :

MAKEMAP, CALCQU

EXT.TAURELATION.450
Controls the 450 um extinction values used in the EXT model and the EXTINCTION task

Description:

Each tau relation is parameterised in the form: " tau_filt = a (tau_cso + b + c sqrt(tau_cso))" , where " a" , " b" and " c" are the three values supplied for this parameter. See also parameter "ext.taurelation.850" . Here, the trailing " .450" in the parameter name is the filter name, not the sub-instrument name as would be the case if it appeared at the start of parameter name. [(23.3,-0.018,0.05)]

SMURF Usage :

EXTINCTION, MAKEMAP, CALCQU

EXT.TAURELATION.850
Controls the 850 um extinction values used in the EXT model and the EXTINCTION task

Description:

The 850 um equivalent to the parameter "ext.taurelation.450" . [(3.71,-0.040,0.202)]

SMURF Usage :

EXTINCTION, MAKEMAP, CALCQU

EXT.TAUSRC
Controls the extinction values used in the EXT model

Description:

Best is to use WVM, uses continuously varying measurements as a function of time stored with each observation. Allowed values are " auto" , " wvmraw" , " wvmfit" , " csofit" , " csotau" and " filtertau" . See EXTINCTION task for further information. [auto]

SMURF Usage :

MAKEMAP, CALCQU

FAKEMAP
Diagnostic tool to explore the effects of the map-making process on known sources

Description:

To test the response of the map-maker to different known astronomical sources, an external " fakemap" can be specified to provide an image of the sky that will produce additional astronomical signal to the time series. A rectangular region is first extracted from the supplied fake map with pixel bounds equal to those of the main output map. The WCS within this extracted region must match that of the main output map (i.e. each pixel must have the same sky coordinates in both maps). A typical procedure may involve: (i) produce a map with makemap; (ii) produce an image with simulated data with the same pixel dimensions and WCS; (iii) specify this new map for the " fakemap" parameter below. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FAKESCALE
Control the use of the supplied fake map

Description:

Each pixel in the supplied fake map (see paramater " fakemap") will be multiplied by a scaling factor before being added to the time stream data. If more than one contiguous chunk of data is being processed, each chunk may be given a different scale. They can be specified in two ways:

- A comma-separated list of explicit numerical values can be supplied for the " fakescale" parameter. The list must be enclosed in parentheses if it contains more than a single element. The last value in the list will be replicated if the number of values in the list is smaller than the number of chunks. Thus, the default value of 1.0 will cause a scale of 1.0 to be used for every chunk.
- The " fakescale" parameter can be set to an arbitrary algebraic expression in which the variable names are the names of FITS keyword. The keyword must be present in the FITS extension of the input data and must have numerical values. The expression should be enclosed in double quotes. The value of the expression using the FITS keywords values for a chunk will then be used as the scale for that chunk. [1]

SMURF Usage :

MAKEMAP, CALCQU

FILT_EDGE_LARGESCALE
Specifies the largest scale size to be retained by the initial data cleaning

Description:

Together with parameter "filt_edge_smallscale" , this specifies the frequencies which the initial data cleaning is to remove from the data streams, based on a range of requested spatial scales (in arcsec), and using internal measurements of the average slew speed. These will override parameter "filt_edgehigh" and parameter "filt_edgelow" . For example, suppose the slew speed is 100 arcsec/sec. We want to ensure that the beam is fully sampled, say 2 arcsec at 450um. That scale is crossed in 2/100 = 0.02 s, so we don't need frequencies in the data above 1/0.02 = 50 Hz in this case (i.e. internally it will set filt_edgelow to 50Hz if filt_edge_smallscale is set to 2 arcsec). Similarly, if we would like to attempt to preserve scales of 10 arcmin = 600 arcsec, we would want to keep frequencies that are greater than 1/(600/100.) = 0.17 Hz (i.e. setting filt_edge_largescale=600 would translate into filt_edgehigh = 0.17 Hz). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FILT_EDGE_SMALLSCALE
Specifies the smallest largest scale size to be retained by the initial data cleaning

Description:

If non-zero, features with spatial sizes less than this value (in arcsec) will be removed by the initial data cleaning. See parameter "filt_edge_largescale" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FILT_EDGELOW
Specifies the highest frequency to be retained by the initial data cleaning

Description:

If non-zero, this is the cut-off frequency of a low pass filter that is applied to the data stream as part of the initial data cleaning. See also parameter "filt_edge_largescale" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FILT_ORDER
Indicates the shape of the filter

Description:

If filt_order is zero or negative, a the edge filters defined by the other " filt_..." parameters are hard-edged. That is, they change from zero to one without any intermediate values. If filt_order is larger than zero, the edge filters are soft-edged Butterworth filters with order given by the value of this parameter. An order of 1 is the softest, and will thus produce least ringing, at the expense of poorer frequency response. Higher orders produce sharper filters that have better frequency response but at the expense of greater ringing. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FLAGFAST
Flag data when we' re moving too fast

Description:

Data taken when the telescope was moving too fast such that sources are smeared can be flagged using this parameter. The value is a threshold slew velocity (arcsec/sec) measured in tracking coordinates. Assuming a sample rate of 200 Hz, we want to be able to fully-sample the 450 and 850 beams. For now just set it to something that is bigger than we need, but be warned that point-sources may be smeared-out. [1000]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FLAGSLOW
Flag data when we' re moving too slowly

Description:

Data taken when the telescope was moving too slowly such that sources are buried in 1/f noise, can be flagged using this parameter. The value is a threshold slew velocity (arcsec/sec) measured in tracking coordinates. Assuming we would like to be able to sample scales of at least 30 arcsec (at least two 15 arcsec beams at 850), and assuming a typical 1/f knee of 1 Hz, the telescope needs to slew at least 30 arcsec/sec to place sources in the signal band above the knee. [30]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FLT.FILT_EDGE_LARGESCALE
Specifies the largest scale size to be retained by the FLT model

Description:

Together with parameter "flt.filt_edge_smallscale" , this specifies the frequencies which the FLT model is to remove from the data streams, based on a range of requested spatial scales (in arcsec), and using internal measurements of the average slew speed. These will override parameter "flt.filt_edgehigh" and parameter "flt.filt_edgelow" . For example, suppose the slew speed is 100 arcsec/sec. We want to ensure that the beam is fully sampled, say 2 arcsec at 450um. That scale is crossed in 2/100 = 0.02 s, so we don't need frequencies in the data above 1/0.02 = 50 Hz in this case (i.e. internally it will set flt.filt_edgelow to 50Hz if flt.filt_edge_smallscale is set to 2 arcsec). Similarly, if we would like to attempt to preserve scales of 10 arcmin = 600 arcsec, we would want to keep frequencies that are greater than 1/(600/100.) = 0.17 Hz (i.e. setting flt.filt_edge_largescale=600 would translate into flt.filt_edgehigh = 0.17 Hz). [600 (for 450 um), 300 (for 850 um)]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_EDGE_LARGESCALE_LAST
Specifies the largest scale size to be retained by the FLT model on the last iteration

Description:

This is the value to be used for parameter "flt.filt_edge_largescale" on the last iteration. If it is "<undef>" , then the same value will be used as for earlier iterations. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_ORDER
Indicates the shape of the filter

Description:

If flt.filt_order is zero or negative, a the edge filters defined by the other " flt.filt_..." parameters are hard-edged. That is, they change from zero to one without any intermediate values. If flt.filt_order is larger than zero, the edge filters are soft-edged Butterworth filters with order given by the value of this parameter. An order of 1 is the softest, and will thus produce least ringing, at the expense of poorer frequency response. Higher orders produce sharper filters that have better frequency response but at the expense of greater ringing. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.NOTFIRST
May improve convergence by avoiding the filtering of strong sources on the first iteration

Description:

If this is non-zero, then low frequencies will not be removed from the time streams on the first iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_BOX1
Controls the flagging of samples that suffer from ringing

Description:

If this is non-zero, then a ringing filter is applied to the residuals once the FLT model has been removed. This filter attempts to locate and flag residuals that suffer from ringing. It gives the size of the box used to smooth the residuals in order to determine the background. It is specified as a multiple of the filter size. A value of 0.5 could be a good starting point. Note, the ringing filter is not applied on any initial iterations specified by parameter "ast.skip" . [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_CIRCLE
Speeds up convergences and reduces ringing by excluding sources within a circle of given radius from the FLT estimate

Description:

Using flt.zero_circle causes any samples falling within a specified circle on the map to be excluded from the filtering performed by the FLT model. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_FREEZE
Prevent the FLT mask from changing after a given number of iterations. This can help convergence

Description:

If flt.zero_freeze is 1.0 or more, the FLT mask will be frozen after the specified number of iterations (the nearest integer value is used). Note, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. If flt.zero_freeze is greater than zero but less than 1.0, the FLT mask will be frozen when the normalized change in the map between iterations drops below the flt.zero_freeze value. A value less than zero means that the mask is frozen as soon as the initial iterations specified by parameter "ast.skip" have been done. A value equal to zero means that the mask is never frozen. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_LOWHITS
Experimental

Description:

Using flt.zero_lowhits causes samples to be excluded from the filtering performed by the FLT model if they fall in regions of the map where the number of samples falling in each pixel is higher than flt.zero_lowhits times the mean number of samples per pixel, averaged over the map. A value of zero means that no masking of low hits regions is performed. The mask is updated on each iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_MASK
Speeds up convergences and reduces ringing by excluding sources within a region specified by an external mask file from the filtering performed by the FLT model

Description:

If flt.zero_mask is set to one of " REF" , " MASK2" or " MASK3" then an NDF will be obtained using the specified ADAM parameter (REF, MASK2 or MASK3) and used as a user-defined mask. Setting flt.zero_mask to an integer value larger than zero has the same effect as setting it to " REF" . Setting it to an integer less than or equal to zero results in no external mask being used with the COM model. Note, using " REF" ensures that the mask and the output image of MAKEMAP are on the same pixel grid - using " MASK2" or " MASK3" does not provide this guarantee (it is then the users responsibility to ensure that the supplied masks are aligned with the output image in pixel coordinates). The pixels in the map that are to be included in filtering performed by the FLT model should be set to the bad value in the mask. All other pixels will be excluded from the filtering (i.e. they will be replaced by artifical data interpolated from the adjacent data). The mask for the first iteration can be set separately using parameter "flt.zero_mask0" . [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_NITER
Allows FLT masking to be switched off after a given number of iterations

Description:

If flt.zero_niter is non-zero, it gives the number of iterations for which the FLT model should be masked. Subsequent iterations are not masked. A value of zero means " mask on all iterations" . However, if parameter "flt.zero_notlast" is set, the mask will will not be applied on the last iteration, even if flt.zero_niter is zero. Note, using FLT masking on many iterations can inhibit convergence. Also, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. A negative value will disable all masking of the FLT model. This parameter controls the masking requested via parameter "flt.clip" , in addition to the masking requested via the " flt.zero_xxx" set of parameters. [2]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_SNR
Speeds up convergences and reduces ringing by excluding samples that correspond to high SNR pixels in the map

Description:

Setting the flt.zero_snr parameter will prevent samples contributing to the FLT model if they fall within map pixels that have SNR values greater than flt.zero_snr. A flt.zero_snr value of zero means no SNR mask is used. See also parameter "flt.zero_snr_ffclean" .

Note, the SNR values are only available once a map has been created, and so using this parameter results in no FLT masking on the first iteration. Consequently the map at the end of the first iteration will have deep rings around bright sources, since no FLT masking was done. Normally, these rings would polute the AST model derived from the map, and thus polute the residuals on the next iteration, resulting in the rings remaining in later maps. To avoid this, parameter "ast.skip" can be set to a positive value. This causes the AST model to be skipped (i.e. no AST signal is subtracted from the residuals) for the first " ast.skip" iterations. This means that a good FLT mask can be formed from these initial iterations before any AST model is calculated and used. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_SNR_FFCLEAN
Provides alternative method for SNR masking

Description:

Setting this parameter to a non-zero value causes the SNR mask requested by parameter "flt.zero_snr" to be created using an algorithm like that used by the KAPPA command " FFCLEAN" (see SUN/95), instead of using a simple thresholding of the SNR map. The parameter "flt.zero_snr" gives the clipping level of the ffclean algorithm, and the parameter "flt.zero_snr_hipass" gives the box size. Using an ffclean algorithm prevents the source regions within the mask being larger than the box size, and may thus produce faster convergence and avoid blobs developing. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_SNRLO
Speeds up convergences and reduces ringing by increasing the size of the SNR mask without introducing noise

Description:

If values are supplied for flt.zero_snrlo and parameter "flt.zero_snr" , then the basic mask created by thresholding at the SNR value specified by flt.zero_snr is modified by expanding each un-masked " source" area down to an SNR equal to flt.zero_snrlo, without introducing any new isolated source areas. The flt.zero_snrlo should be lower than the flt.zero_snr value. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_UNION
Controls how multiple FLT masks are combined

Description:

If more than one FLT mask is specified (for instance, if values are supplied for both parameter "flt.zero_lowhits" and parameter "flt.zero_snr"), then they are combined into a single mask. If flt.zero_union is true (i.e. non-zero), then the source region in the combined mask is the union of the source regions in the individual masks. If flt.zero_union is false (i.e. zero), then the source region in the combined mask is the intersection of the source regions in the individual masks. [1]

SMURF Usage :

MAKEMAP, CALCQU

HITSLIMIT
Rejects map pixels that receive very few samples

Description:

If non-zero, pixels that receive very few bolometer samples are set to bad in the final map. The limiting number of bolometer samples is equal to " hitslimt" times the mean number of hits per pixel, averaged over the map pixels that recieve at least one bolometer sample. [0.01]

SMURF Usage :

MAKEMAP, CALCQU

ITERMAP
Create NDFs holding the map created by each iteration

Description:

If itermap is set to a positive value, the map from each iteration of each chunk will be stored in an output NDF. If itermap is set to a negative value, only the final iteration will be written from each chunk. If its absolute value is larger than 1, then each itermap will include a quality component that reflects the AST mask in use.

By default, each itermap NDF will be stored in an extension called .MORE.SMURF.ITERMAPS in the main output NDF. However, an alternative location can be specified by supplying a value for ADAM parameter ITERMAPS. This is useful as it allows you to look at earlier itermaps whilst makemap is still running. [0]

SMURF Usage :

MAKEMAP, CALCQU

MAPTOL
Specifies when to stop iterating

Description:

If the normalised change (either the mean or maximum change - see parameter "maptol_mean") between the maps created on subsequent iterations falls below the value of maptol, then the map-maker performs one more iteration and then terminates. Only used if parameter "numiter" is negative. The normalised mean (or maximum) change between maps is defined as the mean (or maximum) of the absolute change in map pixel value, taken over all pixels within the region of the mask specified by parameter "maptol_mask" , and normalised by the RMS of the square root of the pixel variances. Compared to parameter "chitol" , this is much more like a " by eye" test, that will stop the solution when the map stops changing. [0.05]

SMURF Usage :

MAKEMAP, CALCQU

MAXLEN
Determines how the input time-series data is split into chunks

Description:

The maximum length (in seconds) for a single chunk of concatenated data. If 0 is supplied, attempt to concatenate entire continuous chunks. [0]

SMURF Usage :

MAKEMAP, CALCQU

MODELORDER
Specifies which models to include in the iterative process, and the order in which they are evaluated

Description:

This should be a comma-separated list, in parentheses, containing one or more of the following model names, in the order in which they should be evaluated. Note: components specified AFTER 'ast' will not be calculated for the first time until the second iteration:

- dks: fit and remove dark squid for the column
- com: remove common-mode signal
- gai: if com specified, fit gain/offset of common mode
- ext: apply extinction correction
- ast: estimate the map and astronomical signal
- flt: apply filter to time streams
- noi: estimate time-domain variance
- smo: time series smoothing using a median or mean boxcar filter
- ssn: scan-synchronous (i.e. azimuth dependent) noise removal
- pln: remove plane from each time slice
- tmp: remove externally define template such as azimuth [(com,gai,ext,flt,ast,noi)]

SMURF Usage :

MAKEMAP, CALCQU

NOI.BOX_SIZE
Allow finer estimation of the noise levels in the time-series data

Description:

Specifies the number of time slices used to determine the noise level in a section of a bolometer time stream. If zero, then the whole bolometer time stream is used, and each bolometer has only one variance value. If non-zero, each bolometer time stream is divided up into boxes containing the specified number of time slices, and a separate variance is found for each box. This variance is then used for each sample in the box, so each bolometer ends up with a variance for every time slice. Negative values are interpreted as number of seconds, and positive values as a number of down-sampled time slices. Note, very small box sizes may produce unrepresentative noise levels, and there is a hard minimum of 101 on the number of downsampled time slices in a noise box. Also, if the number of time slices in the data is smaller than two times the requested box size, then a single noise value is used for each bolometer. [-15]

SMURF Usage :

MAKEMAP, CALCQU

NOI.BOX_TYPE
Determines how the noise in each box is found

Description:

If this is zero, the noise in each box (see parameter "noi.box_size") is found by taking Fourier transform of the residuals in each box and then using the mean power in the range 2 to 10 Hz as the noise. If it is non-zero, the noise for each residual is set to the variance of the neighbouring residuals in a box centred on the residual. Using this scheme causes the noise values to vary continuously with time, whereas the FFT scheme produced blocks of equal noise values. When using a small box size, " noi.box_type=1" will often result in far fewer samples being flagged as unusable. Note, if " noi.box_size" is set to zero, then the value of " noi.box_type" is ignored and the noise is always calculated on the basis of the mean power in the 2 to 10 Hz band. [1]

SMURF Usage :

MAKEMAP, CALCQU

NOI.CALCFIRST
Determines when the noise in each bolometer is estimated

Description:

If a non-zero value is supplied, the bolometer noise levels are calculated immediately after pre-conditioning. Otherwise, they calculated at the end of the first iteration. The former can reduce execution time if parameter " noiseclip" is also set since both operations share a single FFT. [0]

SMURF Usage :

MAKEMAP, CALCQU

NOISECLIPHIGH
Reject bolometers based on their noise

Description:

This step will remove any bolometers noisier than noisecliphigh standard deviations above the median, or noisecliplow standard deviations below the median. Normally the noise clipping happens at the end of the cleaning stage, but if you set noiseclipprecom it will instead occur immediately prior to common-mode subtraction (see parameter " comppreprocess"). [4]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

NOISECLIPLOW
Reject bolometers based on their noise

Description:

See parameter "noisecliphigh" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

NOISECLIPPRECOM
Reject bolometers based on their noise

Description:

See parameter "noisecliphigh" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

NUMITER
Specifies when to stop iterating

Description:

If a positive number is supplied, the specified number of iterations will always be performed. If a negative number is supplied, the absolute value gives the maximum number of iterations to perform. Fewer iterations will be performed if the termination criteria specified by parameter "maptol" and parameter "chitol" are both met before " -numiter" iterations have been performed. [-5]

SMURF Usage :

MAKEMAP, CALCQU

ORDER
Baseline removal

Description:

Subtract a baseline polynomial of this order as part of the initial cleaning phase. Setting order to zero causes a constant value to be removed from each bolometer. Setting it negative causes no background to be removed. [1]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

SHORTMAP
Create NDFs holding the map made form a short chunk of data

Description:

If non-zero, then an extension called .MORE.SMURF.SHORTMAPS is added to the output map NDF, holding maps made from every group of " shortmap" adjacent time slices. Alternatively, set to -1 to produce a map each time the TCS_INDEX value within the JCMTSTATE extension is incremented (i.e., each time a full pass through the scan pattern has been completed). Any other negative value is interpreted as a duration in seconds, and is converted to time slices using the (possibly down-sampled) sample frequency of the data being mapped. [0]

SMURF Usage :

MAKEMAP, CALCQU

SPIKEBOX
Controls time-based spike detection within initial data cleaning

Description:

Size of filter box for the sigma-clipper within the initial data cleaning, in units of samples if positive and seconds if negative. For instance, setting spikebox to 50 will check for excursions from a rolling median filter in a box of length 50 samples. Also see parameter "spikethresh" . [50]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

SPIKETHRESH
Controls time-based spike detection within initial data cleaning

Description:

The SNR value at which to flag spikes within the sigma-clipper used within initial data cleaning. Also see parameter "spikebox" . No de-spiking is performed in the initial data cleaning if a value of zero is supplied. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

WHITEN
Experimental

Description:

If non-zero, then a whitening filter will be applied to each time stream prior to any filtering that is done as part of the initial data cleaning, as specified by parameter " flt_edge_largescale" , etc. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

Appendix J
Configuration Parameters Listed by Category

This appendix orders the configuration parameters described in Appendix I into functional categories.

J.1 General

Parameter	Purpose	Values
`numiter`	Specifies when to stop iterating.	default : -5 `dimmconfig_jsa_generic` : -25 `dimmconfig_bright_extended` : -40 `dimmconfig_bright_compact` : -40 `dimmconfig_blank_field` : 4
`maptol`	Specifies when to stop iterating.	default : 0.05 `dimmconfig_jsa_generic` : 0.01
`modelorder`	Specifies which models to include in the iterative process, and the order in which they are evaluated.	default : (com,gai,ext,flt,ast,noi) `dimmconfig_blank_field` : (com,ext,ast,noi)
`hitslimit`	Rejects map pixels that receive very few samples.	default : 0.01

J.2 Diagnostics

Parameter	Purpose	Values
`fakemap`	Diagnostic tool to explore the effects of the map-making process on known sources.	default : <undef>
`fakescale`	Control the use of the supplied fake map.	default : 1
`exportndf`	Create NDFs holding final model values.	default : 0
`itermap`	Create NDFs holding the map created by each iteration.	default : 0
`bolomap`	Create NDFs holding the map made form each bolometer.	default : 0

shortmap	Create NDFs holding the map made form a short chunk of data.	default : 0
diag.out	Switches on the dumping of various diagnostic information.	default : <undef>

J.3 Pre-processing

Parameter	Purpose	Values
downsampscale	Speeds up map-making, and reduces memory requirements.	default : -1
maxlen	Determines how the input time-series data is split into chunks	default : 0
doclean	Allows pre-cleaned data to be used.	default : 1
exportclean	Allows the initial cleaned data to examined or saved for later use.	default : 0
order	Baseline removal	default : 1
badfrac	Ensures that bad data from DA system are ignored	default : 0.05
compreprocess	Remove common-mode before the iterative algorithm begins.	default : 0
dcthresh	Control the cleaning of DC steps in bolometer time streams	default : 25.0 dimmconfig_jsa_generic : 100 dimmconfig_bright_compact : 100
dcfitbox	Control the cleaning of DC steps in bolometer time streams	default : 30
dcmaxsteps	Control the cleaning of DC steps in bolometer time streams	default : 4
dclimcorr	Control the cleaning of DC steps in bolometer time streams	default : 0
dcsmooth	Control the cleaning of DC steps in bolometer time streams	default : 50
spikethresh	Controls time-based spike detection within initial data cleaning.	default : 0 dimmconfig_blank_field : 10
spikebox	Controls time-based spike detection within initial data cleaning.	default : 50
whiten	Experimental.	default : 0
noisecliphigh	Reject bolometers based on their noise.	default : 4 dimmconfig_jsa_generic : 10.0 dimmconfig_bright_compact : 10.0

noisecliplow	Reject bolometers based on their noise.	default : 0
noiseclipprecom	Reject bolometers based on their noise.	default : 0
flagslow	Flag data when we're moving too slowly	default : 30
flagfast	Flag data when we're moving too fast	default : 1000
filt_edgelow	Specifies the highest frequency to be retained by the initial data cleaning.	default : 0
filt_edge_largescale	Specifies the largest scale size to be retained by the initial data cleaning.	default : 0 dimmconfig_blank_field : 200
filt_edge_smallscale	Specifies the smallest largest scale size to be retained by the initial data cleaning.	default : 0
filt_order	Indicates the shape of the filter.	default : 0

J.4 Iterative: COM model

Parameter	Purpose	Values
com.perarray	Controls the estimate of the common-mode signal	default : 0 dimmconfig_jsa_generic : 1 dimmconfig_bright_compact : 1 dimmconfig_blank_field : 1
com.sig_limit	Controls the rejection of time slices with inconsistent common-modes.	default : 0
com.noflag	Controls the rejection of bad samples from the COM estimate	default : 0
com.corr_tol	Controls the rejection of bad samples from the COM estimate	default : 5 dimmconfig_bright_compact : 7
com.corr_abstol	Controls the rejection of bad samples from the COM estimate	default : 0.2
com.freeze_flags	Controls flagging of samples that differ from the common-mode.	default : 0
com.gain_tol	Controls the rejection of bad samples from the COM estimate	default : 5 dimmconfig_bright_compact : 7
com.gain_abstol	Controls the rejection of bad samples from the COM estimate	default : 3 dimmconfig_bright_compact : 5
com.gain_box	Controls the rejection of bad samples from the COM estimate	default : -30.0
com.gain_fgood	Controls the rejection of bad samples from the COM estimate	default : 0.25

com.gain_rat	Controls the rejection of bad samples from the COM estimate	default : 4
com.zero_mask	Provides a better estimate of the common-mode ("COM") signal, by excluding samples that fall within fixed regions on the sky specified by an external mask.	default : 0
com.zero_circle	Improves common-mode estimation by excluding sources within a circle of given radius from the COM estimate.	default : <undef>
com.zero_lowhits	Improves common-mode estimation by excluding sources in regions containing many data samples.	default : 0
com.zero_snr	Improve the estimate of the common-mode by excluding samples that correspond to high SNR pixels in the map.	default : 0
com.zero_snr_ffclean	Provides alternative method for SNR masking.	default : 0
com.zero_snrlo	Improve estimate of the common-mode by increasing the size of the SNR mask without introducing noise.	default : <undef>
com.zero_union	Controls how multiple COM masks are combined.	default : 1
com.zero_freeze	Prevent the COM mask from changing after a given number of iterations. This can help convergence.	default : 0.0

J.5 Iterative: NOI model

Parameter	Purpose	Values
noi.calcfirst	Determines when the noise in each bolometer is estimated.	default : 0
noi.box_size	Allow finer estimation of the noise levels in the time-series data.	default : -15
noi.box_type	Determines how the noise in each box is found.	default : 1

J.6 Iterative: FLT model

Parameter	Purpose	Values
`flt.filt_edge_largescale`	Specifies the largest scale size to be retained by the FLT model.	default : `dimmconfig_jsa_generic` : 200 `dimmconfig_bright_extended` : 480 `dimmconfig_bright_compact` : 200
`flt.filt_edge_largescale_last`	Specifies the largest scale size to be retained by the FLT model on the last iteration.	default : <undef>
`flt.filt_order`	Indicates the shape of the filter.	default : 0
`flt.ring_box1`	Controls the flagging of samples that suffer from ringing.	default : 0
`flt.zero_mask`	Speeds up convergences and reduces ringing by excluding sources within a region specified by an external mask file from the filtering performed by the FLT model.	default : 0
`flt.zero_circle`	Speeds up convergences and reduces ringing by excluding sources within a circle of given radius from the FLT estimate.	default : <undef> `dimmconfig_bright_compact` : 0.016666
`flt.zero_lowhits`	Experimental.	default : 0
`flt.zero_snr`	Speeds up convergences and reduces ringing by excluding samples that correspond to high SNR pixels in the map.	default : 0 `dimmconfig_jsa_generic` : 5 `dimmconfig_bright_extended` : 5
`flt.zero_snr_ffclean`	Provides alternative method for SNR masking.	default : 0
`flt.zero_snrlo`	Speeds up convergences and reduces ringing by increasing the size of the SNR mask without introducing noise.	default : <undef> `dimmconfig_jsa_generic` : 3 `dimmconfig_bright_extended` : 3
`flt.zero_niter`	Allows FLT masking to be switched off after a given number of iterations.	default : 2
`flt.zero_union`	Controls how multiple FLT masks are combined.	default : 1
`flt.zero_freeze`	Prevent the FLT mask from changing after a given number of iterations. This can help convergence.	default : 0.0
`flt.notfirst`	May improve convergence by avoiding the filtering of strong sources on the first iteration.	default : 0

J.7 Iterative: EXT model

Parameter	Purpose	Values
`ext.tausrc`	Controls the extinction values used in the EXT model.	default : auto
`ext.taumethod`	Controls the extinction values used in the EXT model.	default : adaptive
`ext.taurelation.450`	Controls the 450 um extinction values used in the EXT model and the EXTINCTION task.	default : (23.3,-0.018,0.05)
`ext.taurelation.850`	Controls the 850 um extinction values used in the EXT model and the EXTINCTION task.	default : (3.71,-0.040,0.202)
`ext.csotau`	Controls the extinction values used in the EXT model.	default : <undef>
`ext.filtertau`	Controls the extinction values used in the EXT model.	default : <undef>

J.8 Iterative: AST model

Parameter	Purpose	Values
`ast.mapspike`	Removes spikes from the map.	default : 10
`ast.skip`	Skip subtraction of astronomical signal.	default : 0 `dimmconfig_jsa_generic` : 5 `dimmconfig_bright_extended` : 5
`ast.zero_mask`	Reduces spurious large scale structure in the final map within fixed regions specified by an external mask.	default : 0
`ast.zero_circle`	Reduces spurious large scale structure in the final map outside a circle of given radius.	default : <undef> `dimmconfig_bright_compact` : 0.016666
`ast.zero_lowhits`	Reduces spurious large scale structure in the final map in regions containing few data samples.	default : 0
`ast.zero_snr`	Reduces spurious large scale structure in the final map within regions of low signal-to-noise.	default : 0.0 `dimmconfig_jsa_generic` : 5 `dimmconfig_bright_extended` : 3
`ast.zero_snr_ffclean`	Provides alternative method for SNR masking.	default : 0

ast.zero_snrlo	Can help to remove bowls around sources by increasing the size of the SNR mask without introducing noise.	default : <undef> dimmconfig_jsa_generic : 3 dimmconfig_bright_extended : 2
ast.zero_snr_fwhm	Can help to remove bowls around sources.	default : 0
ast.zero_snr_low	Can help to remove bowls around sources.	default : -1.1
ast.zero_union	Controls how multiple AST masks are combined.	default : 1
ast.zero_freeze	Prevent the AST mask from changing after a given number of iterations. This can help convergence.	default : 0.0
ast.zero_niter	Allows AST masking to be switched off after a given number of iterations.	default : 0
ast.zero_notlast	Allows flux to be present in masked areas in the final map.	default : 1