SUN/258.4

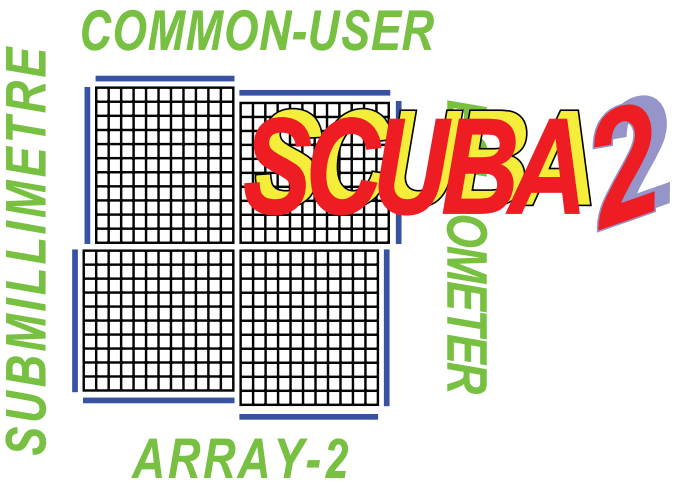
Starlink Project
Starlink User Note 258.4

Edward Chapin, Andrew G. Gibb, Tim Jenness, David S. Berry, Douglas Scott & Remo Tilanus

25th February 2015

SMURF – the Sub-Millimetre User Reduction Facility

Version 2.0.0
User's Guide



Abstract

The Sub-Millimetre User Reduction Facility (SMURF) is a software package for initial reduction of data produced by the ACSIS correlator and the SCUBA-2 bolometer array on the James Clerk Maxwell Telescope (JCMT). This document describes the SMURF tasks used to process raw ACSIS data into data cubes, and raw SCUBA-2 data into images. These low-level tasks are used within automated ORAC-DR pipelines (documented elsewhere) that produce calibrated data products. This document is intended for users who wish to run the low-level tasks themselves, to explore possibilities not provided by the automated pipelines.

Contents

List of Figures

1 Introduction

This document is aimed at users who wish to perform their own customised reductions of ACSIS or SCUBA-2 data. It is is expected that most users will normally prefer to use the higher level facilities provided by the appropriate ORAC-DR pipeline.

The main purpose of this document is to provided complete reference information for all facilities provided by SMURF. Thus, for instance, it contains details of *all* available command parameters and configuration parameters. It is not really intended to be read from start to finish as a complete document, but rather to be dipped into, as and when needed, for information about specific parameters or facilities. Most users will normally refer to this document during the course of reading the following higher-level documents:

- SC/21: SCUBA-2 data reduction cookbook (covers the use of both SMURF and ORAC-DR).
- SUN/264: More details on using the ORAC-DR SCUBA-2 pipeline.
- SC/19: SCUBA-2 SRO data reduction cookbook (focuses specifically on reduction of very early SCUBA-2 data but also contains some methods and information not yet available in SC/21).
- SC/20: Reducing ACSIS data using the ORAC-DR ACSIS pipeline.

After an introductory section covering the mechanics common to using all SMURF commands, the rest of this document is divided into two main parts; one dedicated to processing ACSIS data ([2], see Section 2) and the other for processing SCUBA-2 data ([4], see Section 3).

1.1 Document conventions

In an attempt to make this document clearer to read, different fonts are used for specific structures:

- Observing modes are denoted by all upper case body text (e.g. FLATFIELD).
- Starlink package names are shown in small capitals (e.g. SMURF); individual task names are shown in sans-serif (e.g. makemap).
- Content listings are shown in fixed-width type (sometimes called 'typewriter'). Extensions and components within NDF (NDF) data files are shown in upper case fixed-width type (e.g. `HISTORY`).
- Text relating to file names, key presses or entries typed at the command line are also denoted by fixed-width type (e.g. `% smurf`), as are command-line parameters for tasks (which are displayed in upper case - e.g. `METHOD`).
- References to Starlink documents, i.e., Starlink User Notes (SUN), Starlink General documents (SG) and Starlink Cookbooks (SC), are given in the text using the document type and the corresponding number (e.g. SUN/95). Non-Starlink documents are cited in the text and listed in the bibliography.

1.2 Using SMURF

SMURF is a suite of Starlink ADAM tasks (SUN/101 and SG/4) and therefore requires the Starlink environment to 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
```

1.2.1 Starting SMURF

Having set up Starlink as described in the previous paragraph, the SMURF commands are made available by typing `smurf` at the shell prompt. The welcome message will appear as shown below:

```
% smurf

        SMURF commands are now available -- (Version 1.6.1)

        Type smurfhelp for help on SMURF commands.
        Type 'showme sun258' to browse the hypertext documentation.
        Type 'showme sc21' to view the SCUBA-2 map-making cookbook
```

This defines aliases for each SMURF command, gives a reminder of the help command and shows the version number. You can now use SMURF routines or ask for help.

1.2.2 Getting help

Access the SMURF online help system as follows:

(1) At the prompt, type smurfhelp. The welcome message is displayed along with a list of available topics.

(2) To get information, type the name of an available topic at the help prompt. The next level of help lists information and further subtopics.

(3) To go to the next level, type the name of a subtopic.

(4) Type a question mark, ?, to re-display the available topics at the current level.

(5) To go back one level, press <Enter>.

(6) To exit the help system, press <Enter> until you return to the shell prompt.

Further help on the help system maybe obtained by accessing the topic `smurfhelp` from within smurfhelp. If you already know the topic for which you want help, you can access it directly by specifying it on the smurfhelp command line, as in the following example:

```
% smurfhelp makemap parameters
```

If an application prompts you for input and you do not know what the parameter means, you can use ? at the prompt for more information.

```
% calcflat
IN - Input flatfield files > ?

CALCFLAT

  Parameters

    IN

      IN = NDF (Read)
         Input files to be processed. Must all be from the same
         observation and the same sub-array.

IN - Input flatfield files >
```

1.2.3 SMURF parameters

SMURF uses named parameters to specify input and output files and other variables necessary for data processing. There are two types of named parameter which should not be confused as they are accessed and specified in very different ways:

"ADAM", or "command line" parameters: These can be specified on the command line when running a SMURF command, in just the same way as when running other Starlink commands. If no value is supplied on the command line for a parameter, a default value will be used. If no default value is available, or if use of a default is not appropriate, then the user is prompted for a value. The reference documentation for each SMURF command includes details of each ADAM parameter, and indicates if a default will be used or not. KAPPA (SUN/95) has a convenient overview of the Starlink parameter system. ADAM parameters are usually used to specify the main inputs and output for each command, and to select the main options to be used.

Maybe the most difficult aspect of giving ADAM parameter values on the command line is handling shell meta-characters. If the parameter value includes any characters that would normally be interpreted and replaced by the Unix shell before invoking the requested command, such as wild-cards, dollars, commas, *etc*, then they must be protected in some way so that the Starlink software receives them unchanged. This can be done either by escaping each meta-character (i.e. preceding each one with a back-slash character - "\") or by quoting the whole string. If all else fails, it may be necessary to enclose the parameter value in two layers of quotes, an inner layer of single quotes and an outer layer of double quotes. *Note, the above comments only apply for ADAM parameter values that are supplied on the command line - when supplying a value in response to a prompt, ths Unix shell is not involved and so shell meta-characters should not be escaped or enclosed in quotes.*

Option	Description
none	No messages
quiet	Limited messages
normal	Very few messages
verbose	Full messages
debug	Some debugging messages (useful for programmers)
all	All messages regardless of debug level

Configuration parameters: These are used to fine tune the details of specific algorithms, and are usually much more numerous than ADAM parameters. If required, a SMURF command will access an entire group of configuration parameter settings using a single ADAM parameter usually called "CONFIG". The configuration parameter settings can be specified directly as a comma separated list in response to a prompt for CONFIG, or may be stored in a text file, the name of which is then supplied (preceded by a caret - '^') when prompted for CONFIG. Each SMURF command that requires a group of configuration parameters will document what is needed, and how it can be supplied, in the reference documentation for CONFIG. Appendix F describes individual configuration parameters in detail. KAPPA (SUN/95) has a complete description of the various ways in which groups can be specified.

1.2.4 Message filter

All SMURF commands support the 'message filter' ADAM parameter (MSG_FILTER), which controls the number of messages SMURF writes to the screen when executing routines. The default setting for the message filter is normal. Table 1.2.4 lists the available values for MSG_FILTER. Be aware that specifying verbose or debug will slow down execution due to the (potentially vast) number of messages written to the terminal. It is also possible to control message output by setting the MSG_FILTER environment variable to one of the values listed in this table. To hide all messages, a quick option is to add QUIET to the command line.

1.2.5 Working with data files

SMURF does not itself enforce a naming scheme on files. However, raw data from ACSIS and SCUBA-2 obey a well-defined naming scheme. The convention is as follows: the name is composed of an instrument prefix, the UT date in the form YYYYMMDD, a zero-padded five-digit observation number, followed by a two-digit sub-system number (ACSIS only) and a zero-padded four-digit sub-scan number, all separated by underscore characters. The file has an extension of .sdf. The instrument prefix for ACSIS is simply "a". For SCUBA-2 it is a three-character string dependent on the particular sub-array from which the data were recorded. The SCUBA-2 sub-arrays are labelled a–d at each wavelength, which are coded by a single digit (either 4 or 8 for 450 and 850 μm data respectively); thus the SCUBA-2 prefix is s[4|8][a-d].

Example ACSIS file name: a20090620_00023_01_0002.sdf
Example SCUBA-2 file name: s8a20090620_00075_0001.sdf

Files can be processed either singly or in batches. It is more efficient to process multiple files at the same time. There are three ways to specify multiple files:

(1) store the file names in a text file and then supply the file name, preceded by a caret '^', as the value for ADAM parameter `IN`.

(2) include one or more wild-cards in the ADAM parameter value. Such wild-cards are expanded by Starlink itself, rather than the Unix shell, and so need to be protected from shell expansion using quotes or back-slashes as described earlier.

(3) list the file names explicitly, separated by commas (which need to be quoted).

For more information on specifying groups of objects for input and output, see the section *Specifying Groups of Objects* in the KAPPA documentation (SUN/95). Examples of valid inputs (including the back-slashes and quotes required to protect the shell meta-characters) are:

```
IN=s8a20090620_00075_0001.sdf
IN=s8a20090620_00075_\*
IN=s8a20090620_00075_00\?\?
IN="'file1,file2'"
IN=^myfile.lis
OUT=\*_out
```

Note that if you are providing a text file containing *output* file names, those should be listed in the same order as the input file names, otherwise the processed data will be written under the wrong file names.

1.3 Data file structure

Data files for both ACSIS and SCUBA-2 [6] use the Starlink N-dimensional Data Format (NDF, see NDF), a hierarchical format which allows additional data and metadata to be stored within a single file. The KAPPA KAPPA (SUN/95) contains many commands for examining and manipulating NDF structures. A single NDF structure describes a single data array with associated meta-data. 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 7 dimensions - usually 3 for JCMT data);
- An array of variance values corresponding to the numerical data values;
- World Coordinate System information;
- History;
- Raw data units ("K" for ACSIS, "adu" (possibly as compressed integers) for raw SCUBA-2 data, "pW" for flat-fielded SCUBA-2 data, *etc*).

For ACSIS, the raw data are stored as $N_{\rm chan} \times N_{\rm receptors} \times N_{\rm samp}$, while SCUBA-2 data are stored as $N_{\rm columns} \times N_{\rm rows} \times N_{\rm samp}$, where $N_{\rm samp}$ is the number of time samples in a file.

The files also contain additional NDF components common to both instruments:

- JCMT State structure (the telescope pointing record) and other metadata that potentially varies for every sample;
- JCMT Observatory Control System (OCS) configuration, with the contents of the XML file used to set up the observation;
- A "FITS extension" containing information in the form of a set of FITS (Flexible Image Transport System) header cards, that does not change during a sub-scan.

The jcmtstate2cat command can be used to extract the time varying metadata and store it in a tab-separated table (TST) format catalogue.[1] so that it can be visualised using TOPCAT. An example TOPCAT plot of the telescope motion for a particular observation can be seen in 1. The JCMTSTATE extension contains information from the telescope, secondary mirror and real-time sequencer (RTS). ACSIS observations include environmental weather information and SCUBA-2 observations include SCUBA-2 data (such as the mixing chamber temperature) and the water vapour monitor (WVM) raw data. jcmtstate2cat converts the telescope and SMU information to additional columns showing the tracking and AZEL offsets and also converts raw WVM data to a tau (CSO units). Finally, SCUBA-2 data has additional low-level MCE information that can be included in the output catalogue using the '`--with-mce`' option.

The original XML used to specify the details of the observation can be obtained from any data file using the dumpocscfg command.

The FITS extension is used to store information that either does not change or changes by a small amount during the course of an observation. Note that in the particular case of SCUBA-2 data some values in the FITS extension will change for each sub-scan (i.e. file) of a single observation. The values in the FITS extension may be viewed with the KAPPA fitslist command.

Each instrument has further specific components. SCUBA-2 files contain dark SQUID data, the current flatfield solution, an image of the bolometers used for heater tracking and possibly information indicating how to uncompress the raw data. ACSIS files contain information about the receptors used including their coordinates in the focal plane.

Output files created by SMURF may contain some or all of these, plus new components with information about the output data. These are noted in the description of specific applications. All output files contain a `PROVENANCE` extension which provides a detailed record of the data processing history. Use the KAPPA command provshow to list the contents.

1.4 Supported coordinate systems

SMURF uses AST for its astrometry and thus any coordinate system supported by AST may be used when creating images/cubes. The default behaviour is to use the system in which the observations were made (known as the `TRACKING` system within SMURF).

[1] This is a standard format historically supported by CURSA and ESO SkyCat

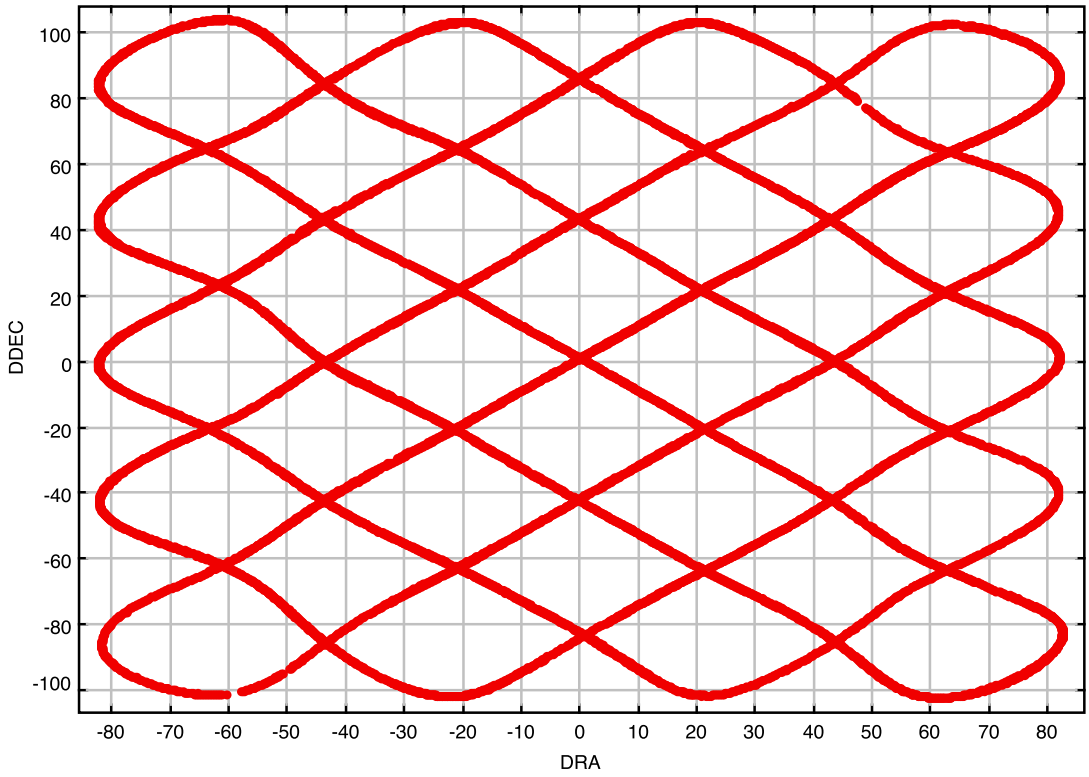


Figure 1: The telescope positions during observation number 7 on 20090107. The plot is created by TOPCAT from the output from jcmtstate2cat plotting the DRA columns against the DDEC column. The pong scan pattern is clearly visible.

1.4.1 Moving sources

The mapping tasks makemap and makecube automatically deal with moving sources. There is no need to deal with moving sources explicitly for any processing with SMURF. Maps and cubes made from moving sources will use a coordinate system that represents *offsets* from the source centre, rather than absolute celestial coordinates.

1.5 File sizes and disk space

Be aware that the raw data files from both instruments may be large (tens to hundreds of megabytes). Subsequent processing of raw SCUBA-2 time-series data produces output files which are even larger for two reasons:

(1) The archived raw integer data values are compressed using "delta" compression (see KAPPA task ndfcompress) and must therefore first be uncompressed before being processed[2]. The compression ratio varies from file to file but is usually in the range 2 to 3.

(2) The uncompressed data values are then converted from four byte integer to eight byte floating point.

Thus file size increases in the range 4 to 6 are to be expected. SMURF mapping tasks have the ability to restrict the size of output data files for manipulation on 32-bit operating systems. For further details, see the description of the `TILEDIMS` parameter in the sections on makemap and makecube. Processing SCUBA-2 data is faster on 64-bit systems due to its use of double precision for all calculations.

2 ACSIS Data Reduction

SMURF contains a small number of commands specifically directed towards the reduction of ACSIS data. This section contains an introduction to each of these commands. The reduction of ACSIS data is described more fully in *The Heterodyne Data Reduction Cookbook* (SC/20).

2.1 The makecube command

The central step in the reduction of ACSIS data is the conversion of one or more time-series cubes with `[spectral channel number, receptor index, time-slice index]` axes, into a spatial cube with `[RA, Dec, spectral channel number]` axes[3]. This step is accomplished using the makecube command.

This command first defines the extent (in both pixels and world co-ordinates) of the output cube, initialises the output cube to hold zero at every pixel, and then goes through each receptor sample value in the input data. For each such sample, it first works out its position on the sky and its spectral position, using the meta-data in the input files. It then determines the pixel in

[2]This uncompression is performed automatically when the data is read by any Starlink command - there is no need to uncompress the data as a separate step.

[3]The RA and Dec axes may be replaced by axes appropriate for other celestial co-ordinate systems.

the output cube that has the same spatial and spectral position, and then "pastes" the sample value into the output cube at this central pixel position. This pasting involves dividing the input sample value up between the output pixels in the neighbourhood of the central output pixel, and then incrementing each of these output pixel values by its assigned share of the input sample value. Various schemes can be used to determine the share assigned to each output pixel - the simplest and fastest is "nearest neighbour" in which the whole input sample value is assigned to the central output pixel. Whilst being fast, this scheme can produce small geometric shifts in feature positions. Other available schemes include a bi-linear scheme that spreads each input value out linearly between the four closest output pixels, Gaussian weighting with a given radius, Sinc weighting, *etc*.

There are many other aspects of the operation of makecube that can be controlled via various program parameters, some the more important of which are:

- The output data can be split up amongst multiple output cubes, thus reducing the size of each individual cube.
- The world co-ordinate system used by the output cubes can be specified. This includes both the nature of the WCS axes themselves and the details of the tangent-plane projection (pixel size, orientation, *etc*). If required, optimal values for the projection details can be determined automatically to produce a regular grid (or as close to a regular grid as is possible given the input data). Alternatively, the WCS in the output cube be copied from a specified reference NDF.
- Output variance values can be calculated either on the basis of the spread of the input data values that contribute to each output pixel, or on the basis of the system noise temperature values supplied in the input NDFs.
- Input data from specified receptors can be excluded.
- Input data from specified spectral channels can be excluded.
- A catalogue can be created holding the spatial position of every used input sample value.

2.2 The timesort command

The makecube commands turns time-series cubes into spatial cubes. Each plane in a time-series cube contains a set of spectra - one for each used receptor - observed simultaneously at a time given by the position of the plane on the third axis. It should be noted that the times associated with this third axis are not guaranteed to increase monotonically, although that will often be the case. For various reasons, the writing out of some times slices to the time-series cube may be delayed, resulting in backwards steps in time along the third axis. This in itself is not a problem for makecube which makes no assumptions about the order in which time-slices are stored in the time-series cube. However, other commands may fail - particularly commands that need to locate the time-slice index associated with a given time. Such commands assume that time is a monotonic function of time-slice index, and that therefore the equivalent of a simple binary chop can be used to locate the index associated with a given time. These commands will fail with a message indicating that "no inverse transformation can be found" if the time-series cube contains any out-of-order time slices. Operations that may produce such a failure include attempting to use KAPPA wcsalign to align two time-series cubes, or displaying a 2D slice of

a time-series cube using KAPPA display. In these cases, the time-series cube can be "rectified" using timesort. This produces a copy of the cube in which the time slices are in monotonic order.

In addition, timesort can also be used to assemble and re-order the time slices from multiple input cubes relating to the same observation, and then write them out again into a set of time-series cubes, each of a specified size. In this situation, if a single time slice is split across two input cubes (which can happen - spectra from some detectors being recorded in one file and those from the other detectors in another file), they will be merged together into a single plane in one of the output cubes.

2.3 The unmakecube command

The process of converting one or more time-series cubes into a spatial cube will usually produce some degree of smoothing in the output data due to multiple input samples contributing to each pixel in the output cube. This smoothing can sometimes be useful in terms of reducing systematic differences between the input time-series cubes, such as differing base-lines. In these cases it is informative to be able to quantify the effect of this smoothing. The unmakecube command provides this facility - it creates a set of "artificial" time-series cubes from a given spatial cube. In detail, for each sample position in a given set of input time-series cubes, a given spatial cube is interpolated to create a new spectrum, which is then stored at the corresponding position in an output time-series cube. Thus, any smoothing present in the spatial cube is transferred into the output time-series cube.

A typical scheme for using unmakecube is:

(1) use makecube to convert a set of "genuine" time-series cubes into a spatial cube.

(2) use unmakecube to generate a set of "artificial" time-series cubes from the makecube output spatial cube. The original "genuine" time-series cubes are used as templates for the new time-series cubes, so that they will be aligned sample-for-sample.

(3) use the facilities of KAPPA or GAIA to visualise or quantify the differences between the "genuine" and "artificial" time-series cubes.

2.4 The fit1d command

The fit1d command fits profiles along a particular axis of a NDF data file. It is a generic command that will work on hypercubes with up to 7 dimensions, but is here discussed in terms of a typical 3-D ACSIS data-file with axes RA, Dec, and LSR Velocity. Specifically, the fitting of spectra across the imaged section of the sky. The output of fit1d is a cube with the fitted profiles and "Component parameter files as NDF extensions". Be aware that the input cube is expected to be baseline subtracted and to have a zero level of 0.

What sets fit1d apart from most other fitting routines is that, by using "Component parameter files" as inputs, it gives individual control over the fitting at each position. For instance, it is possible to fit broad lines on the nucleus of a galaxy but narrow lines everywhere else in the disk. Or to fit multiple components in an outflow and single components everywhere else in the field. Still, these types of fits may require a considerable familiarity with handling, cutting, and pasting NDF files in order to "create" the desired parameter files for input.

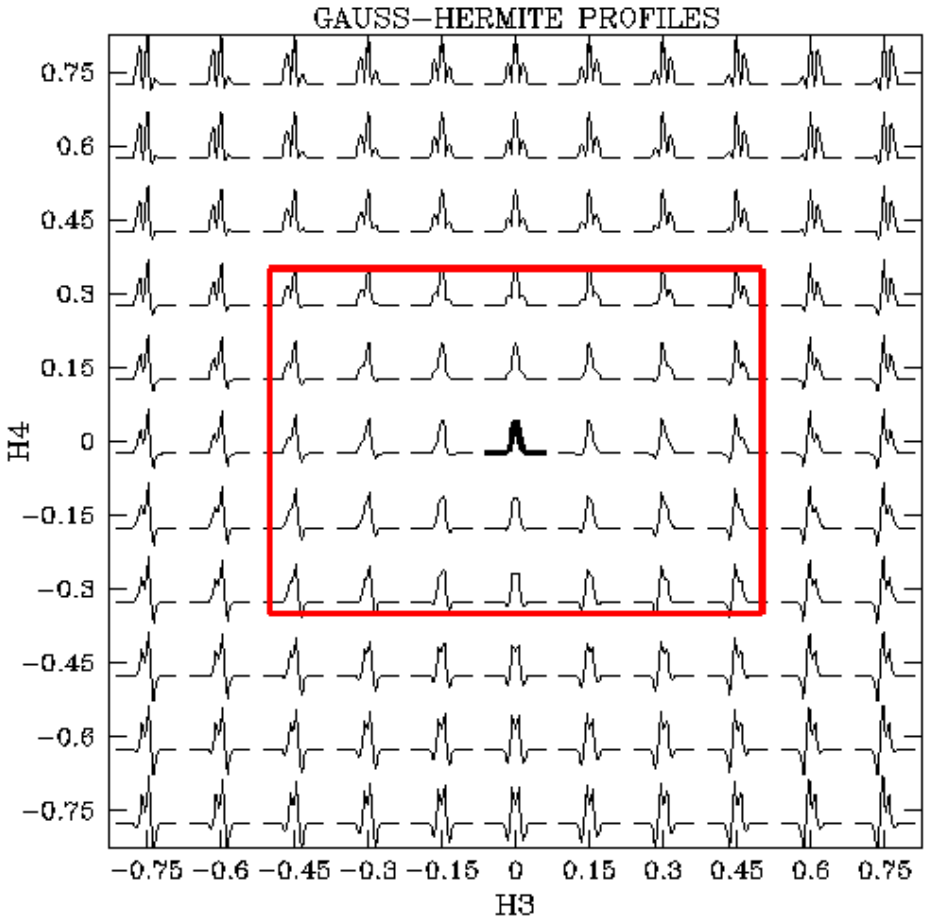


Figure 2: Fit1d − Gauss-Hermite shapes as a function of the 3rd-order "skewness" coefficient h3 and the 4th-order the "peakiness" coefficient h4. the 3rd-order, h3, and the 4th-order, h4, coefficients. The red box indicates the limits on acceptable values for h3 and h4 as defined in the defaults config file. Note that the fitted profile by default is restricted to positive values and this will omit the shown negative features (see the `POS_ONLY` configuration parameter).

fit1d can also fit more complicated shapes than Gaussians. In particular, Gauss-Hermite functions are a powerful extension when fitting profiles that are skewed, peaky, or only approximately Gaussian. Figure 2 shows Gauss-Hermite profiles as a function of the "skewness" coefficient h3 and the "peakiness" coefficient h4. The red box indicates the limits on acceptable values for h3 and h4 as defined in the defaults config file. The limits were chosen such as to exclude fits that look more like multiple components rather than a distorted single Gaussian, but, admittedly are fairly arbitrary.

Because of the ability to fit distorted shapes, Gauss-Hermites are particularly well suited to "capture" the maximum amount of emission from a cube. Figure 3 shows an example of the quality of the fits that can be obtained. For the shown case fit1d used a 3-component gauss-hermite2 (fitting h3 and h4) function with the range around the profiles and the remaining configuration parameters at their default setting. Collapsing the cube with the fitted profiles can

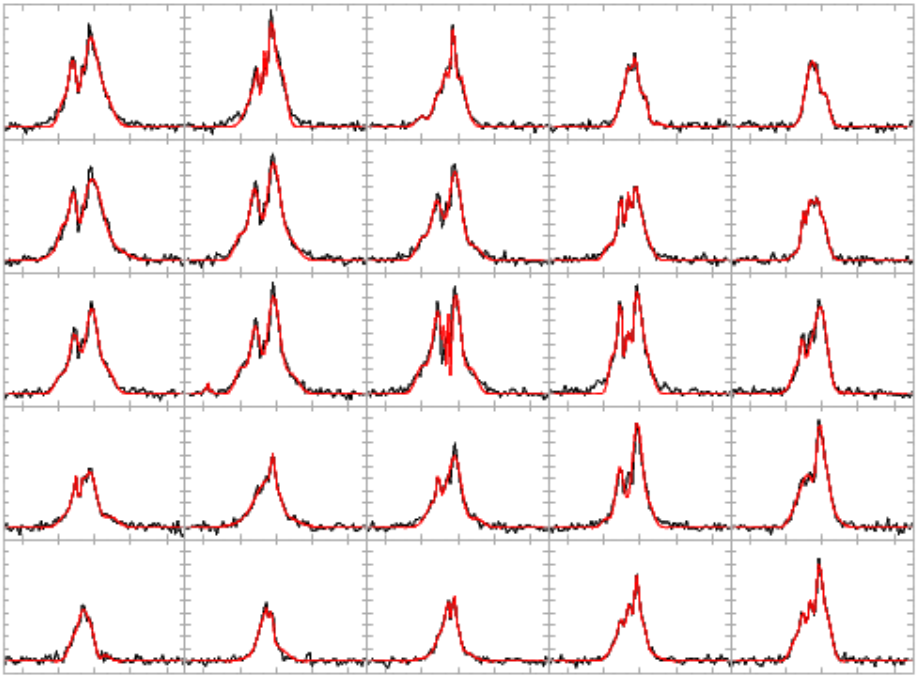


Figure 3: Fit1d — Black: original profiles; Red: results of a 3-component Gauss-Hermite2 fit (fitting both h3 and h4, see next section)

thus result in an accurate and almost noise-free white-light or total-emission map. Residuals from the fit can of course be studied by subtracting the fitted profiles from the original cube.

2.4.1 Fitting functions

fit1d does a one-dimensional fit along each "profile" (spectrum), fitting the number of requested "components" concurrently. Function shapes that can be fitted are "gaussians", "gausshermite1", "gausshermite2", and "voigt" functions, which are discussed in detail in:

```
% smurfhelp fit1d fitting_functions
```

Gauss-Hermite profiles are easiest visualised as the combination of a Gaussian and decaying asymmetric 3rd-order and/or symmetric 4th-order polynomials. The 3rd-order polynomial causes a positive bump on one side and a negative bump on the other side of the main Gaussian, resulting in asymmetric wings and a skewed shape. By contrast the 4th-order polynomial causes a bump in the centre and steeper slopes i.e. a peaky shape.

The Gauss-Hermite profiles in fit1d are called gausshermite1, fitting only h3; and gausshermite2, fitting both h3 and h4 (to fit only h4, use gausshermite2 and define h3 to be 0 and fixed). The default in the configuration file for `FUNCTION` is a gausshermite2.

To emphasise a number of issues:

(1) Be aware that the art of fitting profiles is not in the fits themselves, but rather the initial estimates provided to the fit. Supplying user-defined initial estimates for e.g. the width of the profiles can greatly influence and help the resulting fit. Also, if the initial estimate routine can only find 2 components, the fit will also be restricted to fitting that number of components even if the user is requesting more.

(2) Setting the configuration parameter `ESTIMATE_ONLY` to 1 will skip the fit and produce an output file with profiles based on the initial estimates, allowing the user to inspect those. The associated Component parameter files could be modified and used as initial estimates for a subsequent fit (see next section).

(3) Figure 2 shows that Gauss-Hermites can have prominent negative features. By default these are set to zero in the fitted spectra: see information in the configuration file for the parameter `POS_ONLY`.

(4) *ONLY for Gaussians* do the fitted parameters correspond *exactly* to amplitude, centre, and FWHM! For the other functions such correspondence does not exist: while they are related, the numerical values are not exact. Users are **strongly cautioned** to keep this in mind. The above-mentioned help for "fitting_functions" outlines the exact relations.

(5) The fitted "gauss*" functions can in principle be mixed along a profile: i.e. the first component can be fitted as a "gaussian", the second one as a "gausshermite2", etc. Use the `USERVAL` "User parameter values file" to accomplish this. It not possible to mix in Voigt profiles.

(6) However, since `fit1d` fits concurrently and does not do an iterative fit starting with the strongest or centre-most component, what is the first, second, etc. component is a fluid concept (see next item on sort). The initial estimates routine orders the estimates by decreasing amplitude, but estimates can be quite imprecise. The config file options `SORT` and `SORT_ESTIMATE` may help in minimising problems (see next point).

(7) Sorting of the resulting fits can be done based on the amplitude-like, position, or width-like parameter. This can be helpful, but be cautioned that it can also complicate things: if there are two components one at -10 km/s and one at 10 km/s sorting by amplitude or width can result in the parameter file for component 1 to be a mix of the -10 and 10 features depending on which one was relatively stronger or wider. Similarly, sorting by position can result in low-amplitude fits of noise spikes to be mixed with stronger components. For more precise control try to run the routine iteratively with e.g. a different restricted velocity range to try pick out the different components. Default sorting is by amplitude.

(8) In case multiple components are well separated each can be fitted separately using `RANGE`. The resulting Component parameter files can then be used to generate a combined profile using `fit1d` with `PARCOMP` and `MODEL_ONLY`. This is preferred over simply co-adding the output files with the fitted profiles since it will put all relevant parameters files in the header of the output file.

2.4.2 Component Parameter files

Besides a cube with the fitted profiles `fit1d` also outputs so-called "Component parameter files". These are added as NDF extensions in the header of the output file. They can be accessed there directly but also copied out as independent files:

```
% gaiadisp outfile.more.smurf_fit1d.comp_1 outfile.more.smurf_fit1d.comp_2
% ndfcopy outfile.more.smurf_fit1d.comp_0 comp_0
% ndfcopy outfile.more.smurf_fit1d.comp_1 comp_1
etc.
```

There is a parameter file for each component ("line") that was fitted along the profile up to the number of components requested by the user. These are labelled COMP_1 ... COMP_*n*. COMP_0 is a special diagnostics component that lists the number of components fitted at each position and a return code from the fitting routine.

The Component parameter files have 7 images along the axis that was fitted, with each representing a fitted parameter:

```
COMP_1..N fitted profiles, planes:
           (gaussian)          (gausshermite)        (voigt)
     1 =   amplitude                a                 area
     2 =   position                 b               position
     3 =     fwhm                   c             doppler hwhm
     4 =       -                    h3           lorenztian hwhm 'l'
     5 =       -                    h4           amp2area factor 'v'
     6 =   (empty)
     7 = function-id (1=gaussian), (2=gausshermite1), (3=gausshermite2)
                     (4=voigt)

COMP_0 diagnostics info, planes:
     1 = number of components found
     2 = fit error: (see help fit1d)
```

Thus, for Gaussian fits OUTFILE.MORE.SMURF_FIT1D.COMP_1(,,1) is an image of the fitted amplitudes of Component 1 of each profile, while OUTFILE.MORE.SMURF_FIT1D.COMP_1(,,3) shows the fitted FWHMs across the field-of-view.

Much of the (anticipated) use of fit1d derives from the fact that Component parameter files can be used as input as well: either to provide initial estimates or fixed values to the fitting routine. The hierarchy is as follows:

(1) The routine derives initial estimates up to the number of components requested by the user.

(2) These initial estimates are replaced by values from component parameter files in sequence given (values from the first file specified replace the initial estimate for the first component, etc.).

(3) Any parameter values for component(s) specified by the user in the "User parameter values file" (USERVAL) are then substituted.

(4) Finally, a "fitmask" is defined based on which parameters are free to be fitted and which are declared as "fixed" in the "User parameters values file". Any non-bad value for a free parameter is treated as an initial estimate.

The difference between values specified in the component parameter files and ones declared in the "User parameter values file" is that the former can vary across the field-of-view whereas the latter will result in the same value being used for all profiles.

In principle a Component parameter file can be created from "scratch":

```
# Copy a 7-image section from the cube and fill with blanks
% ndfcopy infile'(,,1~7)' temp

# Give the planes pixel coordinates 1..7. Keep the first two dimensions
# at the original value.
% setorigin temp '[-56,-56,1]'

# Fill planes with values: here just use a constant Gaussian (plane 7 = 1)
# with Ampl=10, Pos = 10 km/s, and FWHM = 5 km/s (assuming units of km/s).
% chpix in=temp   out=temp2  section='",,,"' newval=bad
% chpix in=temp2  out=par1   section='",,1"' newval=10
% chpix in=par1   out=par12  section='",,2"' newval=20
% chpix in=par12  out=par123 section='",,3"' newval=5
% chpix in=par123 out=comp_1 section='",,7"' newval=1

# Generate a cube with the model profiles using fit1d with model_only=1
% fit1d in=infile out=model rms=1 parcomp=comp_1 \
%       config='"^/star/bin/smurf/smurf_fit1d.def,model_only=1"'
```

Obviously, this is not a very practical example given that the profile is the same across the field: setting values in the user definition file for component 1 would achieve the same. However, with dedicated software or a tediously long script running chpix for every position a sophisticated model could, in principle, be created. For instance if a feature shifts in velocity across the field, it could be isolated and fitted by supplying a parameter file with a shifting initial estimate for the position of the peak.

Another situation where manipulation of parameter files can be useful is when parameter files from previous fits require corrections. For instance, in case it is possible to identify the troublesome locations by thresholding with KAPPA thresh, replacing those with bad values; and KAPPA fillbad can be used to interpolate from surrounding values.

Figure 4 shows an example. On the left is the a section of the Amplitude plane of a parameter file resulting from a 1-component fit of a Gaussian. In a few positions problems can be seen (actually due to a secondary component). These points were isolated using thresh on the Position plane of the parameters and be interpolated over using fillbad. The "fixed" parameter file was then used to provide initial estimates for a subsequent file, resulting in the fitted Amplitudes on the right.

```
% fit1d in=infile out=outfile rms=0.22 config=^fit1d.def
% ndfcopy outfile.more.smurf_fit1d.comp_1 comp_1
% thresh comp_1'(,,2)' out=pos thrlo=2 thrhi=5 newlo=bad newhi=bad
% fillbad pos out=pos_fix
% paste comp_1 p1=pos_fix out=comp_1_fix
% fit1d in=infile out=outfile rms=0.22 config=^fit1d.def parcomp=comp_1b_fix
```

These fairly simple examples are presented to illustrate the use of "Component parameter files", but more elaborate schemes are possible. For instance, a section of a parameter file that fits

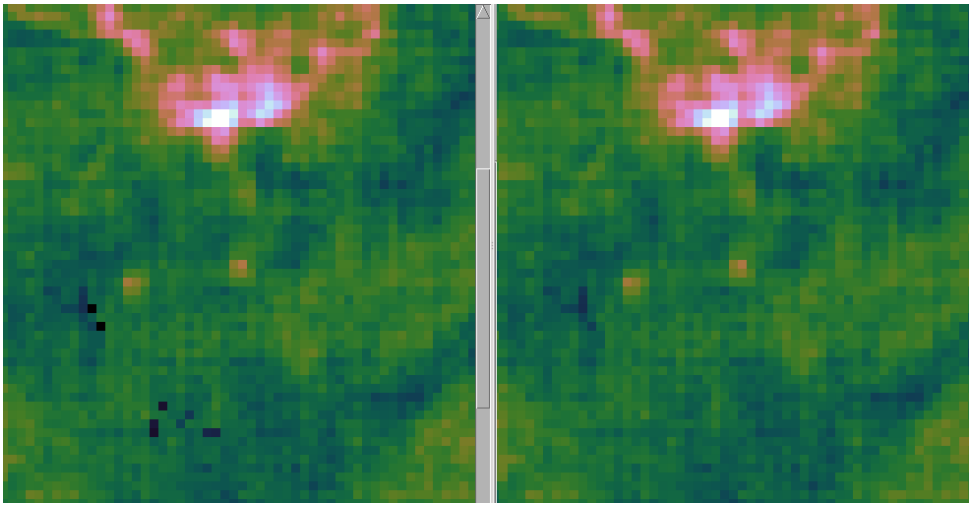


Figure 4: Fit1d — *Left:* Section of a parameter file showing originally fitted amplitudes; *Right:* Amplitudes after using a "fixed" parameter file from the original fit as initial estimates for a subsequent fit.

broad lines in the nucleus of a galaxy can be pasted into a parameter file with narrow fits in the remainder of the disk. We'll leave further examples to to the imagination of the reader.

3 SCUBA-2 Data Reduction

SCUBA-2 was designed to have two primary observing modes where the telescope is either stationary relative to the source (DREAM, STARE and typically when the FTS or polarimeter are in use), or moving (SCAN). In fact, the DREAM/STARE modes have never been commissioned, and so only the SCAN mode is functional at the time of writing. For this reason further information about DREAM/STARE modes has been relegated to appendix G.

Two general approaches may be taken to producing maps from SCUBA-2 data using SMURF commands. By far the easiest approach is to use makemap command to do everything. In this method, the entire job of creating a map, in units of pW, from raw uncalibrated data is handled in a single command. This includes all of the data pre-processing steps, as well as generating the final map.

The alternative approach is to use a range of individual calls to different SMURF task to perform each of the pre-processing steps, followed by using makemap to create the final map from the pre-processed data. This method may be useful if the user wishes to interact and view each step of the reduction, although the final maps are generally not of 'science grade' quality. The individual SMURF tasks used in this approach to perform the pre-processing are described in Section 3.2. Note, these tasks are not necessary in the usual case where makemap is used to perform equivalent pre-processing.

Regardless of how the pre-processing is done, the makemap command will be required to create the final map. Two methods are available when running makemap (selected using ADAM parameter METHOD):

- REBIN: A simple re-binning algorithm that just pastes the cleaned and flat-fielded bolometer values into the output map.
- ITERATE: An dynamic iterative method that divides each cleaned and flat-fielded data value into a number of distinct components, one being an estimate of the astronomical signal, and the others being various systematic noise components. The map is then created by pasting just the component representing the astronomical signal into the output map.

The systematic noise in SCUBA-2 data is usually so great that the REBIN method is inadequate for all but very bright compact sources (bright enough to be detected in one sample and smaller than the field of view of the array). In practice, the ITERATE method will give superior results, though it is more computationally intensive. For this reason, the REBIN method should never be used other than in exceptional circumstances, and further discussion of REBIN is relegated to appendix H.

In all normal circumstances, use the dynamic iterative map maker method described in Section 3.3.

3.1 Choosing input data files

In addition to on-sky data files, each observation usually includes other *calibration data files* such as flatfield ramps and noise fields[4]. It is usually safe to include all the files from an observation, whether on-sky or calibration, when running a SMURF command, as SMURF will sort them all out and only use the appropriate type of file for the operation being performed.

3.2 Array calibration and characterisation

This section describes the individual SMURF commands that can be used to perform the pre-processing steps required prior to making a map from SCUBA_2 data. It is usually simpler to supply the raw data direct to makemap without first using these commands, in which case makemap will automatically perform the required pre-processing.

Data from array calibration observations (such as FLATFIELD and NOISE) are processed with SMURF. NOISE observations are used to identify bolometers which are not in spec, from which a bad-bolometer mask can be created. There is usually no need to re-process flatfield data since the results have already been applied to subsequent observations, but the techniques are included here for completeness. Both NOISE and FLATFIELD observations can be done with the shutter closed or with the shutter open.[5]

3.2.1 FLATFIELD

FLATFIELD observations are processed with the calcflat command. The input data must be from the same observation and the same sub-array. The data are a series of frames in which the current supplied to the internal pixel heaters is varied about a nominal value (see the FITS keyword PIXHEAT). calcflat solves for the optimum heater setting given a list of resistances for each bolometer and a reference resistor value. The list of resistances is mandatory and

[4]The type of data in any individual file is recorded in the SEQ_TYPE FITS header.
[5]Skydip observations are not available at the time of writing

requires knowledge of the sub-array performance. However, a file with suitable default values is included with the installation of SMURF: `$STARLINK_DIR/share/smurf/resist.cfg`.

The output from processing a FLATFIELD observation is a data file (named automatically if not supplied) which contains the flatfield solution in the NDF extension `.MORE.SCUBA2.FLATCAL`, the same location as for the raw data. The main data array for this output file is a three-dimensional array containing the reference-subtracted measurements for each heater setting ($N_{\rm row} \times N_{\rm column} \times N_{\rm heat}$).

In addition to generating the flatfield solution, calcflat can also create a responsivity image for the current sub-array. The `RESP` parameter may be specified to store the responsivity. The responsivity has units of amperes per watt (A/W). Since each observation contains the flatfield, it is possible to extract the responsivity data from any file by using the calcresp command.

If for some reason this flatfield should be applied to existing data files the copyflat command can be used to copy from the file generated by calcflat. This is sometimes useful if a flatfield was accepted by mistake and it needs to be corrected.

3.2.2 NOISE

NOISE observations are designed to check that the bolometers are operating within specifications. This is achieved by calculating the power spectrum and looking at the mean power between two frequencies (usually 2 and 10 Hz). Excessively noisy bolometers are noted and a bad bolometer mask generated. The calcnoise command can be used to analyze any SCUBA-2 time series. It can be used to generate both a noise image (in pA/rt(Hz)) and a NEP image (in W/rt(Hz)) from raw data. The noise image will be the image in the main part of the data file and the NEP image will be in the .MORE.SMURF.NEP extension. If the data have been flat-fielded the NEP image extension will not be written since the primary image will already be in power units. A noise ratio image (.MORE.SMURF.NOISERATIO) is also created to give an indication of the mean power relative to the power at a particular frequency.

Bad bolometer masks are supported by the flatfield, extinction, remsky, and makemap tasks through the `BBM` parameter. A mask can be generated from the noise images using the KAPPA thresh command.

3.2.3 Comparing multiple noise or flatfield observations

Sometimes you will find that you have many noise or responsivity images from multiple observations and you would like to investigate the bolometer stability. The stackframes command can make this simple as it lets you convert a directory of 2-d images into one data cube where each frame is sorted by time. Once you have this cube you can load it into GAIA for easy visualization, use KAPPA collapse with different estimators, or plot variations using KAPPA clinplot or mlinplot.

3.3 Dynamic Iterative Map-Maker

The Dynamic Iterative Map-Maker (DIMM) method within the makemap command is the preferred tool for producing maps of SCUBA-2 total power data obtained in SCAN mode[9]. The DIMM is capable of performing all of the required data pre-processing steps, as well as solving iteratively for multiple signal components (including the image of the astronomical sky) using a single call to the makemap task.

As the telescope scans around the sky, each bolometer is sampled around 200 times a second to produce a stream of bolometer values[6]. The pixel values in the final map are in units of pW and represent the mean of the time-series bolometer values (after conversion into pW, down-sampling and removal of systematic noise) that fall in each pixel.

See Chapin *et. al.* [3] for a full description and analysis of the nature and behaviour of the Dynamic Iterative Map-Maker. Alternatively, see the SCUBA-2 cookbook (SC/21) for shorter and more introductory description.

When using makemap with the `Method=Iterate` option (which is the default and so does not need to be specified explicitly), it is first necessary to create a plain text file containing settings for any required configuration parameters. They can be listed explicitly in the form 'keyword=value', or they can be "inherited" from another file by including the file name preceded by a caret '^'. These two methods can also be combined, with later parameter settings over-riding earlier parameter settings. Any parameters that are not assigned a value using one of these two methods will revert to a default value, as listed in Appendix F.

Appendix F gives details of *all* the available configuration parameters - there are quite a lot! Many of these are of little interest to the majority of SMURF users as they control either experimental features of the map-maker, or features that should not normally be changed. The SCUBA-2 cookbook contains a much reduced list of configuration parameters that are more likely to be of interest to the majority of users.

Various pre-defined configuration files are included with SMURF. The file `dimmconfig_jsa_generic.lis` within directory `$STARLINK_DIR/share/smurf/` is a useful starting point, although better results can often be obtained by using a configuration file tuned to the specific sort of astronomical object being observed (see the other `dimmconfig...` files in the same directory). The SCUBA-2 cookbook (SC/21) describes other ways in which these "config files" can be customised.

The basic command to make a map is as follows:

```
% smurf
% makemap in=^input.lis out=mapname config=^config.lis
```

where `input.lis` is a text file containing the paths to all the input raw data files for a single observation, one per line, and `config.lis` is a text file containing your choice of configuration parameters. Note, all the input data files should relate to the same waveband - 450 or 850 - and the same observation.

Many other options are available - see the list of ADAM parameters for makemap for details. Maybe the most commonly used alternative option is to use a non-default pixel size. To do this assign the required pixel size, in arc-seconds, to the ADAM parameter `PIXESIZE` when running makemap:

```
% smurf
% makemap in=^input.lis out=mapname config=^config.lis pixsize=3
```

[6]This stream of values is usually down-sampled to a lower frequency so that individual samples are separated by a distance similar to the pixel size in the final map.

3.4 Speeding up map-making by re-using previously cleaned data

If you want to compare configurations by making several maps from a single observation, each using a different configuration for the iterative stages, you can save considerable time by using pre-cleaned data each time you run makemap.

Note, this is only appropriate if the differences between the various configurations affect only the iterative stage of the map-making process. If any changes affect the cleaning stage, then it is clearly inappropriate to re-use the cleaned data from a previous run.

There are two possible ways to generate the pre-cleaned data:

(1) Use the SMURF sc2clean task. You can run sc2clean in exactly the same way you would run makemap but instead of creating a map, it creates a set of files containing cleaned and concatenated time-series data that can then be supplied as input to makemap. Any parameters in the configuration file that do not relate to cleaning are ignored by sc2clean so you can conveniently use the same file that you would supply when running makemap. The ADAM parameter `OUT`, which specifies the names of the output data files, can conveniently be set to "`*_cln`", in which case the output files names will be copies of the input file names with the addition of "`_cln`" to the end. There will usually be fewer output files than input files because the data from multiple input files will be concatenated together into a single file. However, data for different wavebands (450 or 850 μm) will always be stored in separate files.

(2) Alternatively, you can ask makemap to generate the cleaned data. To do this, you should add `exportclean=1` to the makemap configuration file. In addition to the output map, makemap will then also create one or more files in the current directory with the suffix "`..._cln.sdf`" containing the cleaned and concatenated time-series data - the earlier part of these file-names will be derived from the names of the input raw data files. *Note, remember to remove `exportclean=1` from your configuration before running makemap again.*

Then for your subsequent runs of makemap you specify the cleaned data generated by one of the above two methods as the input, using ADAM parameter `IN`. In addition, you must add `doclean=0` to the configuration, which prevents makemap from attempting to clean the already cleaned input data.

4 Acknowledgments

The authors are grateful for the documentation written by Maria Kelly on which the SCUBA-2 part of this SUN was originally based, and Christa van Laerhoven for an earlier version. The SCUBA-2 simulator code was originally written as a standalone application by Dennis Kelly at the UK Astronomy Technology Centre and converted to a SMURF task by Jen Balfour.

References

[1] Archibald E. N., et al., 2002, MNRAS, 336, 1 G.0.3

[2] Buckle J. V., et al., 2009, MNRAS, 399, 1026 1

[3] Chapin E. L., et al., 2013, MNRAS, 430, 2545 doi:10.1093/mnras/stt052, arXiv:1301.3652 3.3

[4] Holland W. S., et al., 2006, Proc. SPIE, 6275, 45 1, G, G.0.7

[5] Jenness T., et al., 2002, MNRAS, 336, 14

[6] Jenness T., Gibb, A., 2007, *Data Acquisition/Data Reduction Pipeline Interface Control Document*, SCUBA-2 Data Reduction document SC2/SOF/IC210/01 1.3

[7] Le Poole R. S., van Someren Greve H. W., 1998, Proc. SPIE, 3357, 638 G.0.7

[8] Scott D., 2003, *Map-making in different noise regimes*, SCUBA-2 Data Reduction document SC2/ANA/S210/001

[9] Scott D., 2005, *Scan Mode Data Reduction Strategies for SCUBA-2*, SCUBA-2 Data Reduction document SC2/ANA/S210/006 3.3

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

[11] Van Engelen A., 2005, *Analysis of Atmospheric Emission using SHARC-II Data and Implications for the SCUBA-2 Simulator*, SCUBA-2 Data Reduction document SC2/ANA/S210/002 G.0.3, I.1.2

[12] Van Engelen A., Scott D., 2005, *A Correlation Study using SCUBA Data, with implications for SCUBA-2 data reduction*, SCUBA-2 Data Reduction document SC2/ANA/S210/002

5 Release Notes

5.1 2015A

- This document has been over-hauled, removing or relegating information that is no longer relevant, or is repeated in other documents.
- New commands: JSATILEMOC.
- An experimental scan synchronous noise (SSN) model has been added to MAKEMAP.
- The amount of multi-threading in MAKEMAP is reduced automatically if doing so allows chunking to be avoided.
- The number of chunks used to make a SCUBA-2 map is now recorded in the FITS extension of the map.
- MAKEMAP can now be told to abort if chunking would be used.
- The `dimmconfig.lis` configuration file has been deprecated. It should no longer be used either directly or as the basis for derived configuration files. Configurations that previously would have inherited from `dimmconfig.lis` should instead rely on the default parameter values listed in `$SMURF_DIR/smurf_makemap.def`.

5.2 2014A

- New commands: JSASPLIT, JSAJOIN, JSADICER, JSATILEINFO, JSATILELIST, CHECKCOORDS, FTS2MASKMAP and FITSMERGE.
- New MAKEMAP configuration parameters: AST.SKIP, CHUNKWEIGHT, COM.FREEZE_FLAGS, COM.SIG_LIMIT, FLT.FILT_ORDER, FLT.RING_BOX1, HITSLIMIT, xxx.ZERO_SNR_FFCLEAN, xxx.ZERO_SNR_HIPASS, xxx.ZERO_SNR_LOPASS ("xxx" = AST, FLT, COM).
- New MAKEMAP ADAM parameters: JSATILES, ITERMAPS, INTOPTION.
- New MAKECUBE ADAM parameters: JSATILES, POSERRFATAL.
- New configuration files: `dimmconfig_fix_blobs.lis` and `dimmconfig_fix_convergence.lis`.
- Several values have been changed in `dimmconfig_bright_extended.lis`.

5.3 Hikianalia

- New commands: SKYLOOP and CONFIGMELD.
- If multiple types of mask are requested for any of the AST, FLT and COM models (e.g. ZERO_LOWHITS and ZERO_MASK both specified), they will now be combined into a single mask. Previously, only one of the requested masks would be used.
- Up to three external masks can now be specified using ADAM parameters REF, MASK2 and MASK3. The ZERO_MASK configuration parameter can be set for the AST, FLT or COM model, to indicate which ADAM parameter to use.
- The dimmconfig.lis file now uses maptol to specify the convergence criterion rather than chitol.
- The `textttdimmconfig_bright_extended.lis` file now inherits from `dimmconfig.lis`, rather than `dimmconfig_bright.lis`. This means that the values used for COM.CORR_TOL, COM.GAIN_ABSTOL, COM.GAIN_TOL, DCTHRESH and NOISECLIPHIGH have changed. In addition, `dimmconfig_bright_extended.lis` now sets the value of FLT.FILT_EDGE_LARGESCALE to 600 arc-seconds for both 450 and 850 um. Previously, the defaults of 600 (at 450 um) and 300 (at 850 um) provided by `dimmconfig.lis` were accepted. **Thus, the filter size has changed from 300 to 600 at 450 um.**

5.4 Kapuahi

Many changes - the most significant were.

- New commands: UNMAKEMAP, SC2MAPFFT, SC2FILTERMAP, MCEHEAD2CAT, FIT1D.
- The QLMAKEMAP command has been removed.
- A new, more stable, common-mode estimation has been introduced into MAKEMAP.
- More code has been multi-threaded, making MAKEMAP significantly faster.

- Pointing corrections for the input raw data can now be specified via a text file, using new parameter POINTING.
- CONFIG parameters that specify a number of samples can now also be given in terms of seconds.
- New CONFIG parameter "COM.PERARRAY" allows a separate common mode signal to be estimated and used for each sub-array.
- The COM model can now be masked to omit source samples from the common mode estimation.
- The FLT model can now be masked to omit source samples when estimating the low frequency background of each bolometer time stream.
- SNR masks for the AST, COM and FLT models can now be taken down to much lower levels without introducing isolated noise spikes into the mask (using "xxx.ZERO_SNRLO" parameters).
- A bug has been fixed that caused the memory requirements to grow monotonically when running MAKEMAP repeatedly as a monolith (e.g. from ORAC-DR or ICL).
- Updated Water Vapor Monitor calculation. The WVM samples are now smoothed before being used.
- Setting config parameter "DOWNSAMPSCALE" to -1 will auto-scale to the pixel size. This is now the default.
- New config parameter "MAPTOL" provides a map-based scheme for testing convergence.
- New config parameter "NOI.BOX_SIZE" allows the estimate of the noise within a bolometer time stream to vary with time.
- JCMTSTATE2CAT now calculates telescope speed.
- CALCNOISE now accepts a map-maker config file for pre-cleaning. The default config includes common-mode removal.
- CALCNOISE now calculates the effective noise and NEP.

5.5 Kaulia

Supports compressed SCUBA-2 data files taken since February 2011.

- CALCQU - a new command to create Q and U values from SCUBA-2 fast-spinning polarimeter time-series data files.
- MAKECUBE
  - Now allows the spatial reference pixel in the output cube to be specified by the user.
  - Setting `TRIM=YES` now modifies any user-supplied `LBND/UBND` values to remove borders of bad pixels.

- MAKEMAP
  - The loading and flatfielding of raw data on multi-core systems is now faster.
  - The focal plane distortion model has been updated to cover the entire focal plane.
  - A new parameter (`TRIM`) has been added to allow borders of blank pixels to be removed from the output map.
  - A bug has been fixed so that a request to perform just one iteration (`numiter=1`) is honoured.

5.6 Namaka

Major updates to SMURF to support SCUBA-2 data processing.

- Polynomial fitting to flatfield data is now the default.
- Fast flatfield ramps are now supported. These are taken at the start of every observation.
- New scripts: smas analyzes "short maps" to investigate high frequency seeing/pointing variations

5.6.1 MAKEMAP (iterative)

- Maps from small chunks of the time series can now be written out. Specify the “shortmap” parameter.
- Typos in config parameters are now trapped and the defaults can be seen in file `smurf_makemap.def` in `$SMURF_DIR`. The actual parameters that are used (the merge of the supplied dimmconfig and the defaults) are now stored in the history of the output map.
- Enhanced common-mode removal algorithm which can now break the time series into smaller chunks.
- Enhanced spike removal using a rolling median calculation
- Enhanced step correction.
- Output map now includes QUALITY flags to indicate areas that have been set to 0 by the map-maker.
- The map-maker now reports the details of the flagging every iteration.
- Config files have been re-written to include the standard config with explicit overrides. This makes it easier to see what is being changed.
- Major improvement to the speed of calculating the world coordinates of every bolometer.
- Correctly switch to the CSO tau fits header in the absence of WVM data.

5.6.2 SC2CLEAN

- The parameter DCBOX has been renamed DCFITBOX.
- DCBAD and DCFLAGALL parameters have been removed.

5.6.3 SC2FFT

- Add NGOOD parameter.
- Can now calculate an average power spectrum.

5.6.4 STACKFRAMES

- Fix some bugs associated with maps of different sizes and with missing metadata.
- Propagate QUALITY properly.

5.7 Hawaiki

First official release supporting SCUBA-2 raw data (rather than simulated data).

New applications

- New command CALCNOISE for calculating the noise properties of a SCUBA-2 observation.
- New command COPYFLAT to copy a flatfield from one SCUBA-2 observation to another.
- New command STACKFRAMES for taking a collection of 2-D images (eg noise images or responsivities) and placing them into a time series cube.

Modified applications

- SC2FFT will now concatenate all related files before calculating the FFT.
- MAKECUBE can now avoid generating very thin tiles when tiling is enabled.
- MAKECUBE now uses offset sky co-ordinates in the output cube if, and only if, the input tracking system is (Az,El) or geocentric apparent (RA,Dec).
- JCMTSTATE2CAT now calculates DRA and DDEC columns to indicate arcsecond offsets from base position. Additionally DAZ and DEL are now given as offsets from BASE rather than offsets from the first base position. For SCUBA-2 files JCMTSTATE2CAT will now optionally include MCE header information and will calculate the tau dynamically from the raw WVM data.

5.8 Nanahope

Global changes

- Improve description of SCUBA-2 processing routines for Nanahope Starlink release.
- A summary of input observations is now presented.
- If an output file is derived from multiple observations the output FITS header will now include start and end information (such as date and airmass from the oldest observation and date and airmass from the newest observation).

New applications

- New command CALCRESP for calculating responsivities from flatfield solutions.

Modified applications

- MAKECUBE can now fix up most issues associated with older ACSIS data from 2006 and 2007.
- MAKECUBE output catalogues can now contain additional JCMTSTATE information by using the EXTRACOLS parameter.
- TIMESORT is now significantly faster.
- MAKECUBE, MAKEMAP and QLMAKEMAP are now multi-threaded by default on multi-processor machines.
- MAKECUBE now issues a warning message if WCS information implied by the RECEP-POS and FPLANEX/Y values in an input NDF is inconsistent.
- TIMESORT can now handle single time slice data cubes.
- Many enhancements to SCUBA-2 functionality including support for bad pixel masks and interpolated darks.
- jcmtstate2cat now reports AZ and EL as well as DAZ and DEL.

Bug Fixes

- MAKECUBE formerly mis-interpreted non-zero instrument aperture (INSTAP) values in recent ACSIS data.
- MAKECUBE now ignores input data values that have negative input Tsys values.
- TIMESORT now sets the correct pixel origin in the output NDFs.

A An Alphabetical Summary of SMURF Commands

ACSIS_INDEX Provide a listing of ACSIS data files in a directory.

BADBOLOS Generate a map of random dead bolometers and store in extension of the input file.

CALCDARK Calculate the 2d dark frame from a dark observation.

CALCFLAT Calculate a flatfield solution.

CALCNOISE Calculate the noise properties of a SCUBA-2 observation.

CALCQU Calculate Q and U images from a set of SCUBA-2 time-series data files.

CALCRESP Calculate bolometer responsivity from stored flatfield solution.

COPYFLAT Copy the flatfield solution from one SCUBA-2 observation to another.

DREAMSOLVE DREAM solver.

DREAMWEIGHTS DREAM weight matrix generation.

DSUTILS Utility functions for estimating focal plane distortions.

DUMPOCSCFG Retrieve OCS XML configuration used to generate the raw data.

EXTINCTION Extinction correct SCUBA-2 data.

FIXSTEPS Fix DC steps in a supplied SCUBA-2 time-series NDF.

FLATFIELD Flatfield SCUBA-2 data.

GETTSYS Retrieve TSYS (or TRX) from ACSIS data files.

GSD2ACSIS Convert a GSD format DAS data file to an ACSIS format NDF. (beta test)

GSDSHOW Display the contents of a GSD file's headers and arrays.

IMPAZTEC Import AzTEC NETCDF files and produce SCUBA-2 format data files. (untested)

JCMTSTATE2CAT Dump JCMTSTATE information to ASCII table.

MAKECUBE Regrid ACSIS spectra into a data cube.

MAKEMAP Regrid SCUBA-2 data into a map.

MCEHEAD2CAT Dump the MCE data header of SCUBA-2 files.

RAWFIXMETA Report metadata issues with ACSIS data files.

RAWPRESS Compress raw data that have not previously been compressed. (test routine)

RAWRECREATEWCS Attempt to fix corrupt spectral metadata in ACSIS data.

RAWREWRTSC2WCS Attempt to fix corrupt WCS in raw SCUBA-2 data.

RAWUNPRESS Uncompress raw SCUBA-2 data.

REMSKY Remove Sky signal from SCUBA-2 observations.

SC2CLEAN Clean-up SCUBA-2 time series.

SC2CONCAT Concatenate files from a single observation into a single file.

SC2EXPANDMODEL Expand a DIMM model component into a full time-series data cube.

SC2FFT Perform a forward or inverse FFT on SCUBA-2 data.

SC2FILTERMAP Filter a 2-D map.

SC2MAPFFT Fourier transform 2-D maps.

SC2PCA Use principal component analysis to identify correlated SCUBA-2 signals.

SC2SIM SCUBA-2 simulator.

SCUBA2_INDEX Provide a listing of SCUBA-2 data files in a directory.

SKYNOISE Generate a simulated sky background.

SMAS Analyse shortmaps from the map-maker.

SMURFCOPY Copy a 2d image out of a time series file.

SMURFHELP Give help about SMURF.

STACKFRAMES Stack multiple 2-d images into a 3d cube, optionally sorted by time.

STARECALC Calculate a map from a STARE-mode observation.

TIMESORT Re-order time slices in a raw data cube into increasing time.

UNMAKECUBE Produce simulated time series data from a regridded ACSIS data cube.

UNMAKEMAP Produce simulated time-series data from a SCUBA-2 map.

B Classified SMURF Commands

SMURF applications may be classified in terms of their functions as follows.

General Purpose

DUMPOCSCFG Retrieve OCS XML configuration used to generate the raw data.
JCMTSTATE2CAT Dump JCMTSTATE information to ASCII table.
SMURFHELP Give help about SMURF.
STACKFRAMES Stack multiple 2-d images into a 3d cube, optionally sorted by time.

SCUBA-2 Data Processing

CALCDARK Calculate the 2d dark frame from a dark observation.
CALCFLAT Calculate a flatfield solution.
CALCNOISE Calculate the noise properties of a SCUBA-2 observation.
CALCQU Calculate Q and U images from a set of SCUBA-2 time-series data files.
CALCRESP Calculate bolometer responsivity from stored flatfield solution.
COPYFLAT Copy the flatfield solution from one SCUBA-2 observation to another.
DREAMSOLVE DREAM solver.
DREAMWEIGHTS DREAM weight matrix generation.
DSUTILS Utility functions for estimating focal plane distortions.
EXTINCTION Extinction correct SCUBA-2 data.
FIXSTEPS Fix DC steps in a supplied SCUBA-2 time-series NDF.
FLATFIELD Flatfield SCUBA-2 data.
MAKEMAP Regrid SCUBA-2 data into a map.
MCEHEAD2CAT Dump the MCE data header of SCUBA-2 files.
RAWPRESS Compress raw data that have not previously been compressed. (test routine)
RAWRECREATEWCS Attempt to fix corrupt spectral metadata in ACSIS data.
RAWREWRTSC2WCS Attempt to fix corrupt WCS in raw SCUBA-2 data.
RAWUNPRESS Uncompress raw SCUBA-2 data.
REMSKY Remove Sky signal from SCUBA-2 observations.
SC2CLEAN Clean-up SCUBA-2 time series.
SC2CONCAT Concatenate files from a single observation into a single file.
SC2EXPANDMODEL Expand a DIMM model component into a full time-series data cube.
SC2FFT Perform a forward or inverse FFT on SCUBA-2 data.
SC2FILTERMAP Filter a 2-D map.

SC2MAPFFT Fourier transform 2-D maps.

SC2PCA Use principal component analysis to identify correlated SCUBA-2 signals.

SCUBA2_INDEX Provide a listing of SCUBA-2 data files in a directory.

SMAS Analyse shortmaps from the map-maker.

SMURFCOPY Copy a 2d image out of a time series file.

STARECALC Calculate a map from a STARE-mode observation.

UNMAKEMAP Produce simulated time-series data from a SCUBA-2 map.

ACSIS Data Processing

ACSIS_INDEX Provide a listing of ACSIS data files in a directory.

GETTSYS Retrieve TSYS (or TRX) from ACSIS data files.

MAKECUBE Regrid ACSIS spectra into a data cube.

RAWFIXMETA Report metadata issues with ACSIS data files.

TIMESORT Re-order time slices in a raw data cube into increasing time.

UNMAKECUBE Produce simulated time series data from a regridded ACSIS data cube.

Data Import

GSD2ACSIS Convert a GSD format DAS data file to an ACSIS format NDF. (beta test)

GSDSHOW Display the contents of a GSD file's headers and arrays.

IMPAZTEC Import AzTEC NETCDF files and produce SCUBA-2 format data files. (untested)

SCUBA-2 Simulations

BADBOLOS Generate a map of random dead bolometers and add it as an NDF extension to the input file.

SC2SIM SCUBA-2 simulator.

SKYNOISE Generate a simulated sky background.

C Full Specifications of all SMURF Commands

These pages describe all the SMURF commands in detail. Default values for parameters are quoted in square brackets. Empty brackets indicate a dynamic default will be calculated and inserted at run time. Parameters for which an array should be supplied (or are returned) are shown with parentheses after the parameter name. A number within those parentheses indicates the size of the array.

BADBOLOS
Generate a map of random dead bolometers and add it as an NDF extension to the input file

Description:

Given an input NDF file, retrieve the array size and randomly generate a user-specified number of bad bolometers. These bad bolometers are defined by bad rows, bad columns, or bad individual bolometers. Bad individual bolometers are created in excess of any bolometers already consider bad as part of a bad row or column. Alternatively the user can supply an ARD description for the bad bolometer mask.

Parameters:

ARD = ARD description (Read)

ARD description of bad bolometer mask. In the case that the user selects the ARD method of bad bolometer masking, a correctly formatted ARD description will need to be supplied. The ARD description is treated as a one-to-one correspondence between its values and the rows/columns of bolometers in a subarray.

BAD_BOLOS = _INTEGER (Read)

If the user selects the random generation of bad bolometers, this value indicates the desired number of dead bolometers in excess of those flagged as bad.

BAD_COLUMNS = _INTEGER (Read)

If the user selects the random generation of bad bolometers, this value indicates the desired number of dead columns of bolometers to be randomly generated.

BAD_ROWS = _INTEGER (Read)

If the user selects the random generation of bad bolometers, this value indicates the desired number of dead rows of bolometers to be randomly generated.

IN = NDF (Read)

Input NDF file. If the supplied file already has a bad bolometer mask, it will be overwritten by this routine. Only a single file can be given and it is modified in place.

METHOD = _CHAR (Read)

Bad bolometer generation method (either random, or from an ARD description) as part of the BAD_ROWS and BAD_COLUMNS.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

SEED = INTEGER (Read)

Seed for random number generator. If a seed is not specified, the clock time in milliseconds is used.

Notes:

This application is designed to be used in conjunction with the simulator to mask bolometers. It does not currently function correctly and is deprecated.

Related Applications :

SMURF: SC2SIM

CALCDARK
Calculate the 2d dark frame from dark observation

Description:

Given a set of dark observations, calculate a mean dark frame from each. A bad bolometer mask can be supplied to remove known bad bolometers. Does not flatfield.

Parameters:

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following one will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

IN = NDF (Read)

Input files to be processed. Non-darks will be filtered out.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

Output dark files. These can be used as bad bolometer masks in subsequent processing steps via the BBM parameter in other SCUBA-2 SMURF commands.

Notes:

Dark files will be subtracted from raw data during the flatfielding step. Commands that flatfield data can use either raw dark files or the output from CALCDARK.

Related Applications :

SMURF: FLATFIELD, MAKEMAP

CALCFLAT
Calculate a flatfield solution from a flatfield observation

Description:

This routine calculates a flatfield solution from a flatfield observation.

The flatfield observation consists of a series of measurements taken at various pixel heater settings. One standard SCUBA-2 raw data file is stored for each measurement.

An optimum pixel heater setting is chosen at the time of observation. The procedure is to record measurements at heater settings around this optimum value, continually returning to the optimum, which is used as a reference to subtract pixel zero-point drifts.

Parameters:

IN = NDF (Read)
Input files to be processed. Must all be from the same observation and the same subarray.

METHOD = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NGOOD = _INTEGER (Write)
Number of bolometers with good responsivities.

OUT = NDF (Write)
Output flatfield file. The primary data array contains the dark subtracted measurements for each heater setting. The flatfield itself is stored in the .MORE.SCUBA2.FLATCAL extension. A default output filename based on the date of observation number, subarray name and observation number will be suggested.

ORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

REFRES = _DOUBLE (Read)
Reference pixel heat resistance. Defines the mean power scale to be used. [2.0]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESP = NDF (Write)
Responsivity image with variance. No image is written if NULL. [!]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

SNRMIN = _DOUBLE (Read)

Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

Notes:

Works with Dark and Sky flatfields but not with black-body flatfields.

Related Applications :

SMURF: CALCRESP, FLATFIELD

CALCNOISE
Calculate noise image

Description:

This routine cleans the supplied data and then calculates the white noise on the array by performing an FFT to generate a power spectrum and then extracting the data between two frequency ranges. It additionally calculates an NEP image and an image of the ratio of the power at a specified frequency to the whitenoise.

Parameters:

CONFIG = GROUP (Read)

Specifies values for the cleaning parameters. If the string " def" (case-insensitive) is supplied, a set of default configuration parameter values will be used. CONFIG=! disables all cleaning and simply applies apodisation. This is generally not a recommended use of calcnoise.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The available parameters are identical to the cleaning parameters used by the iterative map-maker (method=ITER) and are are described in the " Configuration Parameters" appendix of SUN/258. Default values will be used for any unspecified parameters. Assigning the value " <def>" (case insensitive) to a keyword has the effect of resetting it to its default value. Options available to the map-maker but not understood by CALCNOISE will be ignored. Parameters not understood will trigger an error. Use the " cleandk." namespace for configuring cleaning parameters for the dark squids.

If a null value (!) is given all cleaning will be disabled and the full time series will be apodized with no padding. This differs to the behaviour of SC2CLEAN where the defaults will be read and used. [current value]

EFFNEP = _DOUBLE (Write)

The effective noise of the .MORE.SMURF.NEP image. See the EFFNOISE parameter for details of how it is calculated.

EFFNOISE = _DOUBLE (Write)

The effective noise of the primary output image. If this command was run on raw data it will be the current noise and if run on flatfielded data it will be the effective NEP. Calculated as the sqrt of 1/sum(1/sigma^2). See also the EFFNEP parameter.

FLATMETH = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

FLOW = _DOUBLE (Given)
Frequency to use when determining noise ratio image. The noise ratio image is determined by dividing the power at this frequency by the white noise [0.5]

FREQ = _DOUBLE (Given)
Frequency range (Hz) to use to calculate the white noise [2,10]

IN = NDF (Read)
Input files to be transformed. Files from the same sequence will be combined.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NEPCLIPHIGH = _DOUBLE (Given)
Flag NEP values this number of standard deviations above the median. If a null (!) value is supplied now high-outlier clipping. [!]

NEPCLIPLOW = _DOUBLE (Given)
Flag NEP values this number of standard deviations below the median. If a null (!) value is supplied now low-outlier clipping. [3]

NEPCLIPLOG = _LOGICAL (Given)
Clip based on the log of the NEP. [TRUE]

NEPGOODBOL = _INTEGER (Write)
The number of bolometers with good NEP measurements (see EFFNEP)

NOICLIPHIGH = _DOUBLE (Given)
Flag NOISE values this number of standard deviations above the median. If a null (!) value is supplied now high-outlier clipping. [!]

NOICLIPLOW = _DOUBLE (Given)
Flag NOISE values this number of standard deviations below the median. If a null (!) value is supplied now low-outlier clipping. [3]

NOICLIPLOG = _LOGICAL (Given)
Clip based on the log of the NOISE. [TRUE]

NOISEGOODBOL = _INTEGER (Write)
The number of bolometers with good NOISE measurements (see EFFNOISE)

OUT = NDF (Write)
Output files (either noise or NEP images depending on the NEP parameter). Number of output files may differ from the number of input files. These will be 2 dimensional.

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line) from the OUT parameter. If a null (!) value is supplied no file is created. [!]

POWER = NDF (Write)
Output files to contain the power spectra for each processed chunk. There will be the same number of output files as created for the OUT parameter. If a null (!) value is supplied no files will be created. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

TSERIES = NDF (Write)
Output files to contain the cleaned time-series for each processed chunk. There will be the same number of output files as created for the OUT parameter. If a null (!) value is supplied no files will be created. [!]

Notes:

- NEP and NOISERATIO images are stored in the .MORE.SMURF extension
- NEP image is only created for raw, unflatfielded data.
- If the data have flatfield information available the noise and NOISERATIO images will be masked by the flatfield bad bolometer mask. The mask can be removed using SETQUAL or SETBB (clear the bad bits mask).
- NOICLIP[LOW/HIGH] and NEPCLIP[LOW/HIGH] are done independently for the NOISE and NEP images (so a bolometer may be clipped in one, but not the other).

Related Applications :

SMURF: SC2CONCAT, SC2CLEAN, SC2FFT

CALCQU
Calculate Q and U images from a set of time-series data files

Description:

This application creates Q and U values from a set of POL-2 time series data files. It has two modes of operation, selected by the LSQFIT parameter. The supplied time series data files are first flat-fielded, cleaned and concatenated, before being used to create the Q and U values.

If LSQFIT is set TRUE, then the output NDFs associated with parameters OUTQ, OUTU and OUTI hold time series data in which the data values represent Q, U and I respectively, rather than raw bolometer values. These time-series are much shorter in length than the supplied input time-series files. Each input time series is split into blocks of adjacent time slices, and a single Q, U and I value is created for each bolometer for each such block. The size of these blocks is specified by the POLBOX configuration parameter, which is an integer giving the size of each block as a multiple of the time taken for a complete revolution of the half-waveplate. Each (Q,U,I) triplet is found by doing a least squares fit to the supplied input data (i.e. the analysed intensity data) within a single block of time slices. The fitted function includes first, second, fourth and eight harmonics of the half-waveplate, together with a linear background: (" w" is the angle of the half-waveplate, and " itime" is the zero-based offset of the time slice into the box):

y = A∗sin(4∗w) + B∗cos(4∗w) + C∗sin(2∗w) + D∗cos(2∗w) + E∗sin(w) + F∗cos(w) + G∗itime + H + J∗sin(8∗w) + K∗cos(8∗w)

The returned Q, U and I values are then:

U = 2∗A Q = 2∗B I = 2∗(G∗box/2 + H)

The Q and U values are specified with respect to either north or the focal plane Y axis (see parameter NORTH). Each single pair of corresponding Q and U values in the output NDFs are created from a single least-squares fit, and the residuals of each such fit are used to calculate a notional variance for the corresponding pair of Q and U values. These are not " real" variances, but are just a scaled form of the residuals variance using a scalaing factor that gives reasonable agreement to the visible noise in the Q and U values measured in ten test observations. These variances are intended for determining relative weights for the Q and U values, and should not be used as absolute variance values.

If LSQFIT is set FALSE, then the output NDFs associated with parameters OUTQ, OUTU and OUTI are each 2D and contain a single Q, U or I value for each bolometer. Multiple 2D images are created, as the telescope slowly moves across the sky. Each image is created from a block of time slices over which the sky position of each bolometer does not change significantly (see parameters ARCERROR, MAXSIZE and MINSIZE). The resulting set of Q images can be combined subsequently using KAPPA:WCSMOSAIC, to form a single Q image (normally, the " I" image should be used as the reference image when running WCSMOSIAC). Likewise, the set of U images can be combined in the same way. All the created Q and U images use the focal plane Y axis as the reference direction (positive polarisation angles are in the same sense as rotation from the focal plane X axis to the focal plane Y axis). Since this direction may vary from block to block due to sky rotation,

the idividual Q and U images should be processed using POLPACK:POLROTREF before combining them using KAPPA:WCSMOSAIC, to ensure that they all use the same reference direction. Q and U values are determined from the Fourier component of each time series corresponding to the frequency of the spinning half-waveplate (6 - 12 Hz), and should be largely unaffected by signal at other frequencies. For this reason, the cleaning specified by parameter CONFIG should usually not include any filtering that affects frequencies in the range 2 -16 Hz. There is an option (see configuration parameter SUBMEAN) to subtract the mean value from each time slice before using them to calculate Q and U.

Separate Q, U and I esimates are made for each half revolution of the half-wave plate. The Data values in the returned NDFs are the mean of these estimates. If there are four or more estimates per bolometer, the output will also contain Variance values for each bolometer (these variances represent the error on the final mean value, not the variance of the individual values). A warning message will be displayed if variances cannot be created due to insufficient input data.

The current WCS Frame within the generated I, Q and U images will be SKY (e.g. Right Ascension/Declination).

Parameters:

ARCERROR = _REAL (Read)

The maximum spatial drift allowed within a single Q or U image, in arc-seconds. The default value is wavelength dependant, and is equal to half the default pixel size used by smurf:makemap. Only used if LSQFIT is FALSE. []

CONFIG = GROUP (Read)

Specifies values for various configuration parameters. If the string " def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The available parameters include the cleaning parameters used by the SC2CLEAN and MAKEMAP commands, plus additional parameter related to the calculation of Q and U. Further information about all these parameters is available in the file $SMURF_DIR/smurf_calcqu.def. Default values will be used for any unspecified parameters. Assigning the value " <def>" (case insensitive) to a keyword has the effect of resetting it to its default value. Parameters not understood will trigger an error. [!]

FIX = _LOGICAL (Read)

If TRUE, then attempt to fix up the data to take account of the POL-2 triggering issue that causes extra POL_ANG values to be introduced into JCMTSTATE. [FALSE]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

HARMONIC = _INTEGER (Read)
The Q and U values are derived from the fourth harmonic of the half-wave plate rotation. However, to allow investigation of other instrumental effects, it is possible instead to derive equivalent quantities from any specified harmonic. These quantities are calculated in exactly the same way as Q and U, but use the harmonic specified by this parameter. They are stored in the output NDFs given by the OUTQ, OUTU and OUTI parameters, in place of the normal Q, U and I values. Only used if LSQFIT is FALSE. [4]

IN = NDF (Read)
Input file(s).

LSQFIT = _LOGICAL (Read)
Use least squares fitting method to generate I, Q and U time streams? If not, the the output I, Q and U values are found by convolving each input time stream with sine and cosine waves of the requested harmonic. Note, the reference direction for LSQFIT Stokes vectors is specified by parameter NORTH, whereas the reference direction for non-LSQFIT Stokes vectors is always focal plane Y. [FALSE]

MAXSIZE = _INTEGER (Read)
The maximum number of time slices to include in any block. No upper limit is imposed on block size if MAXSIZE is zero or negative. Only used if LSQFIT is FALSE. [0]

MINSIZE = _INTEGER (Read)
The minimum number of time slices that can be included in a block No Q or U values are created for blocks that are shorter than this value. No lower limit is imposed on block size if MINSIZE is zero or negative. Only used if LSQFIT is FALSE. [200]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NORTH = LITERAL (Read)
Only used if LSQFIT is TRUE. Specifies the celestial coordinate system to use as the reference direction for the returned Q and U values. For instance if NORTH=" AZEL" , then they use the elevation axis as the reference direction, and if " ICRS" is supplied, they use the ICRS Declination axis. If " TRACKING" is supplied, they use north in the tracking system - what ever that may be. If a null (!) value is supplied, the Y axis of the focal plane system is used as the reference direction. Note, Stokes parameters created with LSQFIT=FALSE always use focal plane Y as the reference direction. [" TRACKING"]

OUTF = LITERAL (Write)
The output files to contain the fitted analysed intensity. Only used if LSQFIT is TRUE. It should be a group of time series NDFs. No fit data files are created if a null (!) value is supplied. [!]

OUTFILESI = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line) that hold I data. If a null (!) value is supplied no file is created. [!]

OUTFILESQ = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line) that hold Q data. If a null (!) value is supplied no file is created. [!]

OUTFILESU = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line) that hold U data. If a null (!) value is supplied no file is created. [!]

OUTI = LITERAL (Write)
The output file to receive total intensity values. If LSQFIT is FALSE, this will be an HDS container file containing the I images. The NDFs within this container file are stored and named in the same way as those in the " OUTQ" container file, but using " U" insead of " Q" in the NDF names. If LSQFIT is TRUE, these will be a group of time series NDFs. No I data files are created if a null (!) value is supplied. [!]

OUTQ = LITERAL (Write)
The output file to receive Stokes Q values. If LSQFIT is FALSE, this will be an HDS container file containing the Q images. Each image is held in a separate 2D NDF within the container file. The NDF names will be " Q<i>_<s>_<c>" , where " <i>" is the integer one-based index of the time slice block from which the image was made, " <s>" is the name of the subarray (e.g. " s4a" , etc), and " <c>" is an integer one-based chunk index. If LSQFIT is TRUE, these will be a group of time series NDFs.

OUTU = LITERAL (Write)
The output file to receive Stokes U values. If LSQFIT is FALSE, this will be an HDS container file containing the U images. Each image is held in a separate 2D NDF within the container file. The NDF names will be " U<i>_<s>_<c>" , where " <i>" is the integer one-based index of the time slice block from which the image was made, " <s>" is the name of the subarray (e.g. " s4a" , etc), and " <c>" is an integer one-based chunk index. If LSQFIT is TRUE, these will be a group of time series NDFs.

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

CALCRESP
Calculate bolometer responsivity from stored flatfield solution

Description:

This routine calculates a bolometer responsivity image from a stored flatfield solution. To calculate a new flatfield solution, use the CALCFLAT command on a FLATFIELD observation.

Parameters:

IN = NDF (Read)

Input files to be processed. Each input file is processed independently so will work with all observation files including dark observations and a combination of sub-arrays.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NGOOD() = _INTEGER (Write)

Number of bolometers with good responsivities. Integer array, one entry for each input file.

RESIST = GROUP (Read)

A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

OUT = NDF (Write)

Output responsivity images. If the input files were each taken with the same stored FLATFIELD solution the output responsivity images will be identical.

Notes:

- Works on all raw data files. The responsivity image will be identical for all files in a single observation since it is only updated following a FLATFIELD observation.
- Provenance is not propagated to the output files, since the output files do not depend on the bolometer data.
- The responsivity data are filtered using a signal-to-noise ratio of 5.0. The number of bolometers passing this criterion is reported in the NGOOD parameter. The variance is stored in the output files so additional filtering is possible.
- For TABLE flatfields the CALCFLAT calculation of responsivity can use the variance of each measurement to calculate a weighted fit. The data files themselves do not store the variance in the flatfield solution so the answer from CALCRESP may differ slightly to the answer calculated with CALCFLAT.

Related Applications :

SMURF: FLATFIELD, CALCFLAT KAPPA: MAKESNR

CHECKCOORDS
Check for discrepancies between AZEL and TRACKING coordinates

Description:

This routine checks that the AZEL boresight positions stored in the TCS_AZ_AC1/2 arrays in the JCMTSTATE extension of the supplied input NDF are in agreement with the corresponding TRACKING positions given by the TCS_TR_AC1/2 arrays. It does this by using AST to convert each TCS_AZ_AC1/2 into the tracking system and then finding the arc-distance from this converted position to the corresponding TCS_TR_AC1/2 position. This is done for every time slice, and the statistics of the resulting separations are displayed, in arc-seconds. A warning message is reported if any separations larger than 3 arc-seconds are found.

Parameters:

IN = NDF (Read)

The time series to be checked.

COPYFLAT
Copy the flatfield information from a reference file

Description:

This routine copies the flatfield parameters from a reference data file (usually a raw SCUBA-2 observation or the output from the CALCFLAT command) to a group of files. The flatfield is updated in place.

Parameters:

REF = NDF (Read)

File from which to read the flatfield parameters. Can be a raw data file that contains the required flatfield or output from the CALCFLAT command. Only a single file for a single subarray can be provided.

IN = NDF (Read)

Input files to be updated. The subarray must match that used for the reference file.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

Related Applications :

SMURF: CALCFLAT

DREAMSOLVE
Solve DREAM observations and generate 2-D images

Description:

This command reconstructs a series of 2-D images from DREAM observations. The images are written as NDFs under the .MORE.SCU2RED extension.

Parameters:

IN = NDF (Read)
Name of input data files.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Name of output files containing DREAM images. The DREAM images will be written to extensions to match that in use by the SCUBA-2 data acquisition system.

Related Applications :

SMURF: DREAMWEIGHTS, STARECALC; KAPPA: WCSMOSAIC; CCDPACK: MAKEMOS

DREAMWEIGHTS
Generate DREAM weights matrix

Description:

This is the main routine for (re)calculating the DREAM weights array and inverse matrix.

Parameters:

CONFIG = Literal (Read)

Specifies values for the configuration parameters used to determine the grid properties. If the string " def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated list of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The parameters available for are listed in the " Configuration Parameters" sections below. Default values will be used for any unspecified parameters. Unrecognised options are ignored (that is, no error is reported). [current value]

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NDF = NDF (Read)

Raw DREAM data file(s)

OUT = NDF (Write)

Output weights file(s)

Notes:

- Raw data MUST be passed in at present (this is due to a limitation in smf_open_file)
- Should allow for a list of bad (dead) bolometers.
- This application interface is not finalised. Please do not rely on this command in scripts.

Configuration Parameters

GRIDSTEP = INTEGER

Scale size of the dream grid pattern. [6.28 arcsec]

GRIDMINMAX = INTEGER

Array of integers specify the extent of the DREAM pattern in pixels. Order is xmin, xmax, ymin, ymax [(-4,4,-4,4)]

Related Applications :

SMURF: DREAMSOLVE

DSUTILS
A collection of utilities for estimating focal plane distortion

Description:

This routine provides various functions needed by the scripts that determine a 2D polynomial describing the focal plane disortion produced by SCUBA-2 optics.

Parameters:

BMAP = FILENAME (Read)

Only accessed if a time series cube is specified via parameter " IN" . It gives the name of a text file in which to place an AST dump of a Mapping from a modified PIXEL Frame to the SKY offset frame for a time slice close to the middle of the data set The PIXEL frame is modified in the sense that its origin (i.e. pixel coords (0,0,0.0)) is shifted so that it co-incides with the sky reference point. No file is created if a null (!) value is supplied. [!]

BORDER = _INTEGER (Read)

Only accessed if a time series cube is specified via parameter " IN" , and a null(!) value sispecified for parameter BMAP. It gives the border width in pixels. Time slices with peak positions closer to any edge than this amount will not be included in the output catalogue. [4]

COLNAME = LITERAL (Read)

Only accessed if a value is supplied for " INCAT" . If supplied, COLNAME should be the name of a column in the INCAT catalogue. An output NDF holding these values will be created (see parameter COLNDF). [!]

COLNDF = NDF (Write)

Only accessed if a value is supplied for " COLNAME" . If supplied, an NDF is created holding the values from the catalogue column specified by COLNAME. The column value from each row in the INCAT catalgue is pasted into the output NDF at a position specified by the BD1/BF2 columns.

FORWARD = _LOGICAL (Read)

Only accessed if input NDFs are specified for parameter INFITX and INFITY, and if a non-null (!) value is supplied for parameter OUTCODE. It indices whether the code written to the OUTCODE file should describe the forward or inverse PolyMap transformation.

FPIXSIZE = _DOUBLE (Read)

Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the pixel size (in mm), of the NDFs created via parameters OUTANG and OUTMAG. [1.0]

FXHI = _DOUBLE (Read)

Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the upper bound (in mm) on the focal plane X axis, of the NDFs created via parameters OUTANG and OUTMAG. [50]

FXLO = _DOUBLE (Read)
Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the lower bound (in mm) on the focal plane X axis, of the NDFs created via parameters OUTANG and OUTMAG. [-50]

FXOFF = _DOUBLE (Read)
Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the focal plane X coord for a point that defines a global offset to be removed from OUTMAG and OUTANG. The point specified by FXOFF and FYOFF will have value zero in OUTANG. No offset is removed if a null (!) value is supplied. [!]

FYHI = _DOUBLE (Read)
Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the upper bound (in mm) on the focal plane Y axis, of the NDFs created via parameters OUTANG and OUTMAG. [50]

FYLO = _DOUBLE (Read)
Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the lower bound (in mm) on the focal plane Y axis, of the NDFs created via parameters OUTANG and OUTMAG. [-50]

FYOFF = _DOUBLE (Read)
Only acccessed if a non-null value is supplied for parameter OUTMAG. It gives the focal plane Y coord for a point that defines a global offset to be removed from OUTMAG and OUTANG. The point specified by FXOFF and FYOFF will have value zero in OUTANG. No offset is removed if a null (!) value is supplied. [!]

IN = NDF (Read)
Only accessed if null (!) values are supplied for parameter INFITX, INFITY, and INCAT. It should be a time series cube. If a non-null value is supplied, then the BMAP parameter can be used to get the WCS Mapping for a typical time slice, or the OUTCAT parameter can be used to create a catalogue holding the expected source position (in GRID coords) in every time slice. A single time slice from this cube can also be written to an output NDF (see parameter OUTSLICE). A list of time slices in which the reference point is close to a specified bolometer can also be produced (see parameter XBOL).

INFITX = NDF (Read)
A 2D NDF holding fitted focal plane X offsets (in mm) at every bolometer, or null (!). This NDF shoudl have been created by KAPPA:FITSURFACE, and should hold the coefficients of the fit in the SURFACEFIT extension. If NDFs are supplied for both INFITX and INFITY, then the OUTCODE parameter can be used to create C source code describing the coefficients in a form usable by the AST PolyMap constructor. In addition, the OUTDX and OUTDY parameters can be used to create output NDFs containing the values for the inverse quantities.

INFITY = NDF (Read)
A 2D NDF holding fitted focal plane Y offsets (in mm) at every bolometer, or null (!). See INFITX.

INCAT = FILENAME (Read)
Only access if a null (!) value is supplied for INFITX or INFITY. It is a text file holding a catalogue of corrected and uncorrected pixel positions for every bolometer in the subarray specified by SUBARRAY. These are used to create output NDFs holding the

X and Y focal plane offset (in mm) at every bolometer in the subarray (see OUTDX and OUTDY). An output catalogue can also be produced holding extra columns, and from which abberant rows have been rejected (see parameter OUTCAT).

ITIME = _INTEGER (Read)
The integer index of a time slice to be dumped to an NDF (see OUTSLICE). If supplied, the application terminates without further action once the NDF has been created. [!]

LOWFACTOR = _REAL (Read)
Only accessed if a value is supplied for parameter IN. It gives the lowest time slice data sum (as a fraction of the largest time slice data sum in the supplied timne series cube) for usable time slices. Any time slices that have total data sums less than this value are skipped.

NITER = _INTEGER (Read)
Only accessed if a value is supplied for parameter INCAT. It gives the number of sigma-clipping iterations to be performed whilst creating the output NDFs. [3]

OUTCAT = FILENAME (Write)
If a value was supplied for INCAT, then OUTCAT is the name of an output catalogue to create, containing a copy of the input catalogue form which abberant rows have been removed, and contaiing some extra informative columns (e.g. offsets in focal plane and pixel coordinates). No catalogue is created if a null (!) value is supplied. If a value is supplied for IN, then OUTCAT will hold the expected source position in each time slice.

OUTCODE = FILENAME (Write)
If a value was supplied for INFITX and INFITY, then OUTCODE is the name of an output text file in which to store the C code describing the coefficients of the forward or inverse distortion polynomial.

OUTDX = NDF (Write)
If a value was supplied for INFITX and INFITY, then OUTDX gives the name of the NDF in which to store the inverse X axis corrections at each bolometer in the subarray (in mm). If a value was supplied for INCAT, then OUTDX is the name of an NDF to recieve the forward X axis correctiosn at every bolometer in the subarray (in mm).

OUTDY = NDF (Write)
If a value was supplied for INFITX and INFITY, then OUTDX gives the name of the NDF in which to store the inverse Y axis corrections at each bolometer in the subarray (in mm). If a value was supplied for INCAT, then OUTDX is the name of an NDF to recieve the forward Y axis correction at every bolometer in the subarray (in mm).

OUTANG = NDF (Write)
Only acccessed if a non-null value is supplied for parameter OUTMAG. OUTANG specifies the output NDF to receive the orientation of the distortion (in degrees anti-clockwise from the positive Y axis) at each point in the focal plane.

OUTFX = NDF (Write)
The name of an NDF to recieve the focal plane X value (in arc-sec) at each bolometer in the subarray specified by SUBARRAY. Only produced if a non-null value is also supplied for OUTFY. [!]

OUTFY = NDF (Write)
The name of an NDF to recieve the focal plane Y value (in arc-sec) at each bolometer

in the subarray specified by SUBARRAY. Only produced if a non-null value is also supplied for OUTFX. [!]

OUTMAG = NDF (Write)

An output NDF to receive the magnitude of the distortion (in mm) at each point in the focal plane. If a null (!) value is supplied, no NDF will be created. The NDFs specified by OUTMAG and OUTANG can be displayed as a vector plot using KAPPA:VECPLOT. In addition, the outline of any sub-array can be over-plotted by changing the current coordinate Frame and then using KAPPA:ARDPLOT (for instance " wcsframe outmag s8a" followed by " ardplot s8a"). Note, for some distortions (e.g. NEW4) the distortion at 450 and 850 are different. The waveband to use is determined by the value supplied for the SUBARRAY parameter. [!]

OUTSLICE = NDF (Write)

If a value was supplied for IN and ITIME, then OUTSLICE gives the name of the NDF in which to store the bolometer data for the given time slice, including celestial WCS.

SUBARRAY = LITERAL (Write)

The name of the subarray being processed: one of " s8a" , " s8b" , " s8b" , " s8d" , " s4a" , " s4b" , " s4b" , " s4d" . If OUTMAG is not null, then the value supplied for SUBARRAY determines the waveband for which the distortion is returned.

XBOL = _INTEGER (Read)

The index of a test bolometer on the first GRID axis within the sub-array containing the bolometer. If values are supplied for all of IN, XBOL, YBOL and RADIUS, then a list of time slice indices are displayed. These are the indices of the time slice in which the reference point is close to the bolometer specified by (XBOL,YBOL). The RADIUS parameter spcified the distance limit. [!]

YBOL = _INTEGER (Read)

The index of a test bolometer on the second GRID axis within the sub-array containing the bolometer. See XBOL. [!]

RADIUS = _REAL (Given)

The radius of a test circle, in bolometers. See XBOL. [!]

Related Applications :

SMURF: DISTORTION, SHOWDISTORTION

EXTINCTION
Extinction correct SCUBA-2 data

Description:

This application can be used to extinction correct data in a number of ways.

Parameters:

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

CSOTAU = _REAL (Read)

Value of the 225 GHz zenith optical depth. Only used if TAUSRC equals 'CSOTAU' or ' AUTO' . If a NULL (!) value is given, the task will use the appropriate value from the FITS header of each file. Note that if a value is entered by the user, that value is used for all input files. In AUTO mode the value might not be used.

FILTERTAU = _REAL (Read)

Value of the zenith optical depth for the current wavelength. Only used if TAUSRC equals 'FILTERTAU' . Note that no check is made to ensure that all the input files share the same filter.

FLATMETH = _CHAR (Read)

Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)

The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)

Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)

If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

HASSKYREM = _LOGICAL (Read)

Indicate that the data have been sky removed even if the fact can not be verified. This is useful for the case where the sky background has been removed using an application other than SMURF REMSKY. [FALSE]

IN = NDF (Read)

Input file(s). The input data must have had the sky signal removed

METHOD = _CHAR (Read)
Method to use for airmass calculation. Options are:

- ADAPTIVE - Determine whether to use QUICK or FULL based on the elevation of the source and the opacity.
- FULL - Calculate the airmass of each bolometer.
- QUICK - Use a single airmass for each time slice.

[ADAPTIVE]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output file(s)

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

TAUREL = GROUP (Read)
Specifies values to be used for scaling the 225 GHz tau to the specific filter. These values will only be used if CSOTAU or WVMRAW methods are used to determine the tau. The group should have the form:

ext.taurelation.<FILT> = (a,b,c)

where <FILT> is the filter name and " a" , " b" and " c" are the coefficients for a relationship of the form

tau_filt = a (tau_cso + b + c sqrt(tau_cso))

A null value will use the default relations. [!]

TAUSRC = _CHAR (Read)
Source of optical depth data. Options are:

- WVMRAW - use the Water Vapour Monitor time series data
- WVMFIT - use a fit to the Water Vapor Monitor data
- CSOFIT - use a fit to the CSO 225 GHz tau data
- CSOTAU - use a single 225 GHz tau value
- FILTERTAU - use a single tau value for this wavelength
- AUTO - Use WVM if available and reliable, else a WVM or CSO fit.

[AUTO]

WVMLOG = FILENAME (Write)
Name of a text file into which to write raw WVM data with columns for each stage of

processing. Multiple blocks of data may be written to the file if the WVM data are processed in chunks. New data are appended to the file if it already exists. If a null (!) value is supplied, no file is written. [!]

Notes:

- The iterative map-maker will extinction correct the data itself and this command will not be necessary.
- QLMAKEMAP automatically applies an extinction correction.

Related Applications :

SMURF: REMSKY, MAKEMAP; SURF: EXTINCTION

FINDSLICES
Find time slices that are centred close to a given sky position

Description:

This application lists the time-slices within a given set of time-series files for which the telescope was close to a specified position on the sky. For each such position, the file, time slice index and distance (in arc-seconds) is displayed on the screen.

Each time the telescope scans past the supplied position, a continuous group of time slices will fall within the specified radius of the given position. By default, only the closest time slice within each such group is displayed. This can be changed using parameter CLOSEST.

Parameters:

CLOSEST = _LOGICAL (Read)
If TRUE, then only a single time slice (the closest) is displayed for each pass of the telescope past the specified position. If FALSE, then all time slices within the specified radius of the specified positiopn are displayed. [TRUE]

FRAME = NDF (Read)
The position specified by parameters XPOS and YPOS is assumed to be in the current WCS Frame of the NDF supplied for parameter FRAME. If a null (!) value is supplied for FRAME, the position is assumed to be in the tracking system of the first NDF supplied for parameter IN. [!]

IN = NDF (Read)
Input time-series file(s).

RADIUS = _DOUBLE (Read)
The radius of the search circle, in ars-seconds. [300]

XPOS = LITERAL (Read)
The first axis value at the search position. This should be a formatted value for the first axis of the Frame specified by parameter FRAME.

YPOS = LITERAL (Read)
The second axis value at the search position. This should be a formatted value for the second axis of the Frame specified by parameter FRAME.

FIT1D
Fit 1-D profiles to a data cube

Description:

This command fits 1-D profiles, such as gaussians, along an axis of a multi-dimensional data cube, that is expected to have spectral baselines or continua removed. The task creates a (hyper-)cube of the fitted profiles; this NDF also stores the fitted-profile parameters in an extension. Note that this is a preliminary release of FIT1D.

The routine can fit complex spectra with multiple components in a data cube (actually: fit along any axis of a hyper-cube). It is multi-threaded and capable of fitting a large number of (i.e. order a million) spectra in a few minutes, depending of course on the number of cores available. It borrows heavily from the " xgaufit" routine of the GIPSY package.

The type of profiles that can be fitted and have been tested are " gaussian" , " gausshermite1" (gaussian with asymmetric wings), and " gausshermite2" (peaky gaussians possibly with asymmetric wings). See the important " Fitting Functions" topic for more details.

The parameters for each fitted component reside in the SMURF_FIT1D extension as cube NDFs called COMP_1,..., COMP_N. For a gaussian the planes in these cubes correspond to: amplitude, position, and fwhm. Further details are under topic " Fitted Parameters" .

A default config file is in $SMURF_DIR/smurf_fit1d.def. A sample file for user specified values is $SMURF_DIR/smurf_fit1d_uval.def.

Usage:

```
fit1d in out rms config [userval] [pardir] [parndf] [parcomp]
```

Parameters:

CONFIG = GROUP (Read)
Specifies values for the configuration parameters used by the FIT1D. If the string " def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used from $SMURF_DIR/smurf_fit1d.def. See the " Configuration Parameters" topic for detailed information.

IN = NDF (Read)
Baselined input file(s).

OUT = NDF (Write)
Output file(s) with the fitted profiles.

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null value is supplied no file is created. [!]

PARCOMP = GROUP (Read)
Component parameter file(s) to use for initial estimates, fixed values, or to generate a model with (see " model_only"). Instead of a comma separated string of filenames a " ^list" file can be submitted with each filename on a separate line. Files will map

to components 1..N in the order as specified. See " Fitted Parameters" for more information on parameter files.

The full specification of a parameter file name consists of three parts: an optional directory path (" pardir"), an optional container ndf (" parndf") and plus the base name for the file (" parcomp"): " pardir" /" parndf" .MORE.SMURF_FIT1D." parcomp" For convenience " pardir" and " parndf" can be specified through their respective parameters, but full name(s) can also be specified through " parcomp" . In case " pardir" is specified, FIT1D will append a " /" . Similarly, if " parndf" is given the string " .MORE.SMURF_FIT1D." will be appended. Note that leaving " parndf" blank will result in a conventional " pardir" /" filename" path.

In case " parndf" is specified, but " parcomp" is not, all components in the container file will be read. Note that COMP_0 will be skipped since it hold diagnostics information about the fit.

To escape special characters (" ," , " ." , " @" , etc) in the string you may need to use a set of single+double quotes. A null value means no component parameter files will be used. [!]

PARDIR = LITERAL (Read)

Directory with component parameter files or the parameter NDF. For details see help on PARCOMP. To escape special characters (" ," , " ." , " @" , etc) in the string you may need to use a set of single+double quotes. A null value results in use of the current directory. [!]

PARNDF = LITERAL (Read)

NDF resulting from a previous execution of FIT1D and containing component parameter files as part of its meta-data to use in the current fit. The components are stored as " parndf" .MORE.SMURF_FIT1D.COMP_#. For further details see help on PARCOMP. To escape special characters (" ," , " ." , " @" , etc) in the string you may need to use a set of single+double quotes. [!]

RMS = _DOUBLE (Read)

RMS in input NDF(s) in data units.

USERVAL = GROUP (Read)

Input keymap file with user-supplied fixed values or initial estimates for the fit and a flag whether parameters are to be kept fixed in the fit. The sample/default keymap file is $SMURF_DIR/smurf_fit1d_uval.def. Entries are of the form: " comp" #." fid" = value, " comp" #.par = value, or " fix" #.par = [0,1] with ' par' being a parameter relevant for the function being fitted. ' #' (1..n) indicates the component profile being described. Fix indicates a parameter to be kept fixed or fitted.

If specified " comp" #.fid will override the default function selected in the config file. Parameter names are described in the help item " Fitting Functions" comp.fid comp.par Gauss: 1 a, b, c Gausshermite1: 2 a, b, c, h3 Gausshermite2: 3 a, b, c, h4 Voigt: 4 a, b, c, l

The " fix" #.par parameter can have a value of: 1 = fix parameter at given value, do not fit -or- 0 = use as initial estimate, but fit. As for comp, the ' #' (1..n) indicates the component.

A null value for USERVAL means that no user-supplied estimates or fixed values are to be used. [!]

Examples:

```
fit1d in=orion out=orion_gauh2 rms=0.22
```

Fits using the settings as defined in the default Configuration file " $SMURF_DIR/smurf_fit1d.def" : fitting a single gausshermite2 to each profile along the highest dimension of the file " orion.sdf" . The input file is expected to be baselined and the zerolevel of the profile to be 0. The fitted profiles are stored in the file " orion_gauh2.sdf" with a component parameter file with the gauss-hermite parameters a, b, c, h3, h4 as " orion_gauh2.more.smurf_fit1d.comp_1" (plus ...comp_0 for the diagnostics component).

```
fit1d in=orion out=orion_gauss rms=0.22 \
```

config=' " ^$SMURF_DIR/smurf_fit1d.def,function=gauss" ' Same as above, but fit a single gaussian instead. Alternatively, use config=' " ^myfits1d.def" ' having the following lines: ^$SMURF_DIR/smurf_fit1d.def function = gaussian

```
fit1d in=orion out=orion_gauh2 rms=0.22 parndf=orion_gauss
```

As in the first example fit a gausshermite2, but use the output from the gaussian fit of the second example for initial estimates. This can help or do harm: while the " internal" initial estimate from Fit1d may be in-accurate, in the first example a fit with a gausshermite2 will be attempted regardless. By contrast, the gaussian fit to a non-gaussian profile in example 2 may have been so poor that is was rejected: in that case current gausshermite2 fit will skip the profile because there won't be any initial estimates.

```
fit1d in=orion out=orion_gauh2 rms=0.22 userval=' " ^myvalues.def" ' \
```

config=' " ^$SMURF_DIR/smurf_fit1d.def,function=gauss,ncomp=3" ' Fit three gaussian components to each profile using initial estimates and fixed values as defined in the file " myvalues.def" (template:" $SMURF_DIR/smurf_fit1d_uval' def"), e.g.: comp1.b = -5.2 fix1.b = 1 comp1.c = 6 comp2.b = 20.4 fix2.b = 1 comp2.c = 4 comp3.b = 35.3 That is: provide a user-defined fixed value for the position (parameter " b") of components 1 and 2, and an initial estimate for component 3. Also provide user-defined initial estimates for the FWHM (parameter " c") of components 1 and 2. Leave it to fit1d to find initial estimates for all other parameters.

Configuration Parameters

A default configuration file can be found at $SMURF_DIR/smurf_fit1d.def.

ABSH3MIN, ABSH3MAX = REAL

Min and Max allowable value for the H3 (∼skew) parameter in a Gausshermite# fit. If RETRY=1 has been set, a pure gaussian fit (H3=0) will be attempted in case the initial fit of H3 is out of bounds. Set to <undef> if no limit desired. [0.01, 0.5]

ABSH4MIN, ABSH4MAX = REAL
Min and Max allowable value for the H4 (∼peakiness) parameter in a Gausshermite4 fit. If RETRY=1 has been set, a gausshermite1 fit (H4=0) will be attempted in case the initial fit of H4 is out of bounds. Set to <undef> if no limit desired. [0.01, 0.35]

AXIS = INTEGER
Axis to fit along (starting at 1). A value of 0 translates as fit along highest dimension i.e. Vlsr in a Ra, Dec, Vlsr cube. [0]

CLIP(2) = REAL
Values in the input profiles outside the specified clip-range [min,max] will be not be used in the fit.

ESTIMATE_ONLY = LOGICAL
Set to 1: The output cube will have the results from the initial estimates routine instead of the the fit. Good initial estimates are critical for the fit and checking and/or fixing initial estimates may help solve problems. [0]

FUNCTION = STRING
Function to fit. Currently implemented are " gaussian" , " gausshermite1" , " gausshermite2" , " voigt" . See topic " Fitting Functions" for details. If your aim is to capture a much emission as possible e.g. in order to create a 2-D image from a 3-D cube, gausshermite2 profiles are recommmended. [" gausshermite2"]

MAXLORZ = REAL
Maximum value for the FHWM of the Lorentzian component (" L") in a Voigt fit in terms of ==PIXELS==(!). If RETRY=1 has been set, a pure gaussian fit (L=0) will be attempted in case the initial fit of H3 is out of bounds. [<undef>]

MINAMP = REAL
Minimum value for the Amplitude-like parameter to accept as a genuine fit in terms of the RMS(!). Based on this alone at 3-sigma ∼5% of the profiles selected for fitting can be expected to be noise spikes. This value drops to ∼2% for 5-sigma. All assuming gaussian statistics of course. [3]

MINWIDTH = REAL
Minimum value for the FHWM (∼2.35∗Dispersion) to accept as a genuine fit in terms of ==PIXELS==(!). [1.88]

MODEL_ONLY = LOGICAL
Set to 1: Bypass both the initial estimates and fitting routine and generate profiles directly from the supplied input parameter cube(s) and/or user supplied fixed values. Not supplying all parameters will generate an error. [0]

NCOMP = INTEGER
Maximum number of ' component' functions to fit to each profile, e.g. a multi-component spectrum of maximum three gaussians. [Note: The complete fit of the gaussians is done concurrently, not iteratively starting e.g. with the strongest component]. The routine will try to find and fit ncomp functions along each profile, but may settle for less. [3]

POS_ONLY = LOGICAL
FIT1D expected profiles to have been baselined i.e. fitted profiles typically should not

have a negative values. However, Gausshermite profiles naturally give rise to undesired negative features e.g. in fitting skewed profiles. This parameter simply causes the routine to set values in the output profiles to zero whereever they are negative. This generally gives better matching profiles, but of course means the fits are not pur gausshermites anymore. Make sure to set this parameter to NO of your profiles have genuine negative features e.g. as in p-cygni profiles. [YES]

RANGE(2) = REAL

Coordinate range along axis to find and fit profiles. The format is (x1, x2) including the (). For example, Vlsr -20 35 is " (-20,35)" . Default is to use the full extent of the axis: [<undef>]

RETRY = LOGICAL

Whenever the lorentzian (" L") or hermite parameters (" H3" , " H4") are out of the routine re-tries the fit with the out-of-bounds parameter(s) fixed at 0. This means that in effect the fit cascades to a simpler function: gausshermite2 -> gausshermite1 -> gaussian; voigt -> gaussian. The result is that there are valid fits for more profiles, but the function actually fitted may vary with position. Setting the retry value to 0 prevents this from happening and may cause the fit to fail or be (very) poor. [YES]

SORT = STRING

Sort the resulting fits: " amp" : sort by decreasing fitted value of the amp-like parameter " width" : sort by decreasing fitted fwhm of the width-like parameter " position" : sort by increasing position along the profile axis " distance" : sort by increasing fitted distance from the centre pixel in the profile. Sorting can be helpful, but be cautioned that it can also complicate things: if there are two components one at -10 km/s and one at 10 km/s sorting by amplitude or width can result in the parameter file for component 1 to be a mix of the -10 and 10 km/s features depending on which one was relatively stronger or wider. Similarly, sorting by position can result in low-amplitude fits of noise spikes to be mixed with stronger components. For more precise control try to run the routine iteratively with e.g. a different restricted velocity range to try pick out the different components. Default is to sort by amplitude. [" amp"]

SORT_ESTIMATE = LOGICAL

Sort initial estimates also with the sorting selected in ' sort' . Estimates can be very inaccurate plus are not checked against any boundary limits until after the fit. Thus this option may not be very helpful.

Fitting Functions :

The function menu provides the choice of four functions for which you can fit the parameters to the data in your profiles.

1) A standard GAUSSIAN. Parameters are a = maximum, b = centre, and c = FWHM.

NOTE that if one of h3 and h4 in a gauss-hermite function is non-zero, the mean of the distribution is not the position of the maximum (Reference; Marel, P. van der, Franx, M., A new method for the identification of non-gaussian line profiles in elliptical galaxies. A.J., 407 525-539, 1993 April 20):

2) GAUSS-HERMITE1 polynomial (h3). Parameters are a (amplitude), b (position),c (width), and h3 as mentioned these are ∗NOT∗ the same as maximum, centre, and fwhm of the distribution as for a gaussian: maximum ∼= [determine value and position of

max from fitted profiles using e.g. collapse] centre ∼= b + h3∗sqrt(3) FWHM ∼= abs(c∗(1-3h3^2)) ∼= c skewness ∼= 4∗sqrt(3)∗h3

3) GAUSS-HERMITE2 polynomial (h3, h4). Same as previous, but an extra parameter h4 is included: maximum ∼= [determine value and position of max from fitted profiles using e.g. collapse] centre ∼= b + h3∗sqrt(3) FWHM ∼= abs(c∗(1+h4∗sqrt(6))) skewness ∼= 4∗sqrt(3)∗h3 kurtosis ∼= 8∗sqrt(6)∗h4

4) VOIGT function. Parameters are a (area), b (centre), c (doppler FWHM), l (lorenztian FWHM), and v (area factor) with relations: maximum ∼= [determine value of max from fitted profiles using e.g. collapse] centre ∼= b doppler fwhm ∼= c lorentzian fwhm ∼= l amp = v (OUTPUT ONLY!) amplitude calculated from a (area) using the standard amp2area function for a voigt (based on the Faddeeva complex error function): amp = area / amp2area

Fitted Parameters

The fitted parameters are stored in the file header as

FILE.MORE.SMURF_FIT1D.COMP_0 to COMP_N, with N depending on how many

components are being fitted. These are regular data cubes that can be

inspected with e.g. Gaia or extracted using NDFCOPY. The ' planes' in

the cubes are:

COMP_0 diagnostics info, planes:
1 = number of components found 2 = fit error: (see below)

COMP_1..N fitted profiles, planes:
(gaussian) (gausshermite) (voigt) 1 = amplitude ' a' area 2 = position ' b' position 3 = fwhm ' c' doppler fwhm ' d' 4 = - ' h3' lorentzian fwhm ' l' 5 = - ' h4' amp2area ' v' last: function id 1 = gaussian; 2 = gausshermite1 (h3); 3 = gausshermite2 (h3,h4), 4 = voigt

FIT ERRORS:
>0 Number of iterations needed to achieve convergence according to TOL.

- 1 Too many free parameters, maximum is 32.
- 2 No free parameters.
- 3 Not enough degrees of freedom.
- 4 Maximum number of iterations too small to obtain a solution which satisfies TOL.
- 5 Diagonal of matrix contains elements which are zero.
- 6 Determinant of the coefficient matrix is zero.
- 7 Square root of negative number. <-10 All fitted components rejected due to minamp, minwidth, maxlorz, or range constraints.

FITSMERGE
Merge FITS headers

Description:

This routine reads the FITS headers of the files mentioned in the IN Parameter and merges them using the smf_fits_outhdr function. The merged headers are then written into the file specified by the NDF Parameter.

Parameters:

IN = NDF (Read)
Files containing input FITS headers.

NDF = NDF (Read and Write)
File to receive merged FITS headers.

Related Applications :

KAPPA: FITSLIST, FITSMOD

FIXSTEPS
Fix DC steps in a supplied SCUBA-2 time series NDF

Description:

This routine runs the DC step fixer on a supplied time series (see parameter IN), and optionally writes the corrected data to an output time series (see parameter OUT). It is primarily intended as a debugging tool for the step fixing code. A description of the step fixes performed can be written to a text file (see parameter NEWSTEPS). This can then be supplied as input to a later run of this application in order to check that the new results are the same as the old results (see parameter OLDSTEPS). A warning message will be displayed if any significant difference is found between the old step fixes and the new step fixes.

Configuration parameters for the step fixing algorithm can be supplied either within a " makemap" -style configuration file (see parameter CONFIG), or via command line environment parameters DCFITBOX, DCSMOOTH, etc.

Parameters:

CHANGED = _LOGICAL (Write)

An output parameter to which is written a flag indicating if any significant differences were found between the step fixes produced by the current invocation of this program, and the step fixes described in the file specified via parameter OLDSTEPS.

CONFIG = GROUP (Read)

Specifies default values for the configuration parameters used by the step fixing algorithm. This should be a configuration such as supplied for the MAKEMAP command. [!]

CONTINUE = _LOGICAL (Read)

This parameter is prompted for after each changed step is described. If TRUE is supplied, then the program continues to display details of further changed steps. If FALSE is supplied, the program aborts.

DCFITBOX = _REAL

Number of samples (box size) in which the signal RMS is measured for the DC step finder. The run time default value is obtained via the CONFIG parameter. [!]

DCLIMCORR = _INTEGER

The detection threshold for steps that occur at the same time in many bolometers. Set it to zero to suppress checks for correlated steps. If dclimcorr is greater than zero, and a step is found at the same time in more than " dclimcorr" bolometers, then all bolometers are assumed to have a step at that time, and the step is fixed no matter how small it is. The run time default value is obtained via the CONFIG parameter. [!]

DCMAXSTEPS = _INTEGER

The maximum number of steps that can be corrected in each minute of good data (i.e. per 12000 samples) 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. The run time default value is obtained via the CONFIG parameter. [!]

DCSMOOTH = _INTEGER
The width of the median filter used to smooth a bolometer data stream prior to finding DC jumps. The run time default value is obtained via the CONFIG parameter. [!]

DCTHRESH = _REAL
Threshold S/N to detect and flag DC (baseline) steps. The run time default value is obtained via the CONFIG parameter. [!]

FIRST = _INTEGER
The index of the first change to be display (the first change has index 1). Each change report starts with an index followed by a colon. [1]

IN = NDF (Read)
The time series cube to be fixed. Note, the data must be time ordered (like the original raw data), not bolometer ordered.

MEANSHIFT = _LOGICAL (Read)
Use a mean shift filter prior to step fixing? A mean-shift filter is an edge-preserving smooth. It can help to identify smaller steps, but does not work well if there are strong gradients in the bolometer time stream. Therefore, MEANSHIFT should only be used if the common-mode signal has been subtracted. The spatial width of the filter is given by DCSMOOTH, and the range of data values accepted by the filter is 5 times the local RMS in the original time stream. [FALSE]

NEWSTEPS = FILENAME (Write)
Name of a text file to create, holding a description of each step that was fixed by the step fixing algorithm. The created file can be re-used in a later run via the OLDSTEPS parameter. It can also be viewed using " topcat -f ascii" . If a null (!) value is supplied for NEWSTEPS, no file is created. [!]

NFIXED = _INTEGER (Write)
The number of steps fixed.

NREJECTED = _INTEGER (Write)
The number of bolometers rejected due to them containing too many steps.

OLDSTEPS = FILENAME (Read)
Name of a text file holding a description of each step that should be fixed, if the step fixing algorithm is working correctly. An error is reported if there is a difference between the steps fixes described in this text file, and the step fixes actually produced by runnning the step fixing algorithm. The text file should have been created by an earlier run of this command (see parameter NEWSTEPS). If a null (!) value is supplied for OLDSTEPS, no check is performed. [!]

OUT = NDF (Write)
The fixed time series cube. May be null (!), in which case no output NDF is created.

SIZETOL = _DOUBLE (Read)
Gives the fraction (i.e. relative error) by which two step sizes must differ for them to be considered different. In addition, the absolute difference between two step sizes must also differ by more than SIZETOL times the clipped RMS step size. [0.05]

FLATFIELD
Flatfield SCUBA-2 data

Description:

This routine flatfields SCUBA-2 data. Each input file is propagated to a corresponding output file which is identical but contains flatfielded data. If the input data are already flatfielded, the user will be informed and no action will be performed on the data.

Parameters:

BBM = NDF (Read)
Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

FLATMETH = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

IN = NDF (Read)
Input files to be uncompressed and flatfielded. Any darks provided will be ignored.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output file(s).

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a NULL (!) value is supplied no file is created. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually

specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

Notes:

- Darks will be ignored.
- At the moment an output file is propagated regardless of whether the input data are already flatfielded or not. This is clearly a waste of disk space.

Related Applications :

SMURF: CALCFLAT, RAWUNPRESS

FTS2DEGLITCH
Removes the glitches from the source

Description:

Removes the glitches from the source.

Parameters:

CCSIZE = _INTEGER (Read)
Core cluster size.

DEGLITCHMODE = _INTEGER (Read)
Deglitching mode, 1=CORE, 2=TAIL and any other value means ALL

DSHALFLENGTH = _INTEGER (Read)
Double-Sided interferogram half length.

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

TCSIGMA = _DOUBLE (Read)
Tail cluster standard deviation percentage.

TCSIGMAMUL = _DOUBLE (Read)
Tail cluster standard deviation multiplier.

TCSIZE = _INTEGER (Read)
Tail cluster size.

FTS2FLATFIELD
Corrects for the pixel-to-pixel variation in spectral responsivity, due to detector characteristics

Description:

Corrects for the pixel-to-pixel variation in spectral responsivity, due to detector characteristics.

Parameters:

IN = NDF (Read)
Input data files.

OUT = NDF (Write)
Output data files.

FTS2FREQCORR
Off-Axis frequency correction

Description:

Off-Axis Frequency Correction which scales the frequency grid according to the different path difference that off-axis rays travel through the interferometer.

Parameters:

IN = NDF (Read)
Input data files.

OUT = NDF (Write)
Output data files.

THETA = NDF (Read)
Theta file storing the off-axis angles.

FTS2INIT
Prepares the input to be processed by the FTS2 Data Reduction tasks

Description:

Prepares the input to be processed by the FTS2 Data Reduction tasks

Parameters:

CENTRE = REAL (Read)
Moving mirror position at centre of travel (mm).

FNYQUIST = DOUBLE (Read)
Override the default Nyquist frequency (default scalculated to be 50.0 at SCANVEL of 4.2 mm/sec) (Optional)

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

RTSDELAY = REAL (Read)
RTS delay (ms). [0.0]

ZPD = NDF (Read)
ZPD calibration file.

Notes:

- The CENTRE parameter should be the same value as was used when producing the supplied ZPD file in order to prevent the introduction of an unwanted offset.

FTS2MASKMAP
Mask time series data for a SCUBA-2 map with FTS-2 in the beam

Description:

This routine masks one or more SCUBA-2 time series cubes taken with FTS-2 in the beam.

Parameters:

FTSPORT = _CHAR (Read)

The FTS-2 port to use in calculating the mapping to sky coordinates. This parameter should be " tracking" or " image" . [TRACKING]

IN = NDF (Read)

A group of existing time series data cubes.

OUT = NDF (Write)

A group of output NDFs into which the masked time series data will be written.

Related Applications :

SMURF: MAKEMAP

FTS2OPCORR
Compansates for the effect of the FTS-2 optics on image distortion and field rotation

Description:

Compansates for the effect of the FTS-2 optics on image distortion and field rotation.

Parameters:

IN = NDF (Read)

Input files to be transformed.

OUT = NDF (Write)

Output files.

FTS2PHASECORR
Given a 3D data cube of single-sided interferograms, applies phase correction and outputs the corresponding 3D interferogram cube

Description:

Given a 3D data cube of single-sided interferograms, applies phase correction and outputs the corresponding 3D interferogram cube. Although double-sided 3D interferogram cubes can be ingested, it is recommended that the FTS2PHASECORRDS task is utilized to process them.

Parameters:

APODIZATION
Apodization Method

DEGREE = _INTEGER (Read)
Order of the fitting polynomial.

DSHALFLENGTH = _INTEGER (Read)
Double-Sided Interferogram half-length.

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

PCFHALFLENGTH = _INTEGER (Read)
Phase Correction Function half-length.

WNLBOUND = _DOUBLE (Read)
The lower bound of the wavenumber range

WNUBOUND = _DOUBLE (Read)
The upper bound of the wavenumber range

FTS2PHASECORRDS
Given a 3D data cube of double-sided interferograms, applies phase correction and outputs the corresponding 3D spectrum cube

Description:

Given a 3D data cube of double-sided interferograms, applies phase correction and outputs the corresponding 3D interferogram cube. Although single-sided 3D interferogram cubes can be ingested, it is recommended that the FTS2PHASECORR task is utilized to process them.

Parameters:

DEGREE = _INTEGER (Read)
Order of the fitting polynomial.

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

WNLBOUND = _DOUBLE (Read)
The lower bound of the wavenumber range

WNUBOUND = _DOUBLE (Read)
The upper bound of the wavenumber range

FTS2PORTIMBAL
Corrects for atmospheric transmission across the spectral dimension

Description:

Corrects for atmospheric transmission across the spectral dimension, given the current PWV and elevation. Transmission data is stored in the form of a wet and dry Tau vs frequency table in the calibration database.

Parameters:

IN = NDF (Read)
Input data files.

OUT = NDF (Write)
Output data files.

FTS2REMOVEBSE
Removes the Beam splitter Self Emission (BSE) from the source

Description:

Removes the Beam splitter Self Emission (BSE) from the source.

Parameters:

BSE = CHAR (Read)
Beam Splitter Self Emission filepath.

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

FTS2SPECTRUM
Computes the spectrum of the interferograms

Description:

Computes the spectrum of the interferograms.

Parameters:

IN = NDF (Read)

Input files to be transformed.

OUT = NDF (Write)

Output files.

RESOLUTION = _INTEGER (Read)

Spectral Grid Resolution. 0 : Full Resolution Any other value : Custom Resolution Default behaviour is the Full Resolution

SFP = NDF (Read)

Spectral Filter Profile (optional) Supply this sub-array specific NDF calibration file to fine tune each bolometer by factoring out its unique heat response deviation from the expected Planck curve response as measured through black body heated load scans

WNSFPFIRST = DOUBLE (Read)

Spectral filter profile wave number range starting from.

WNSFPLAST = DOUBLE (Read)

Spectral filter profile wave number range ending with.

ZEROPAD = LOGICAL (Read)

Determines whether to zeropad.

FTS2SPLIT
Convert multi scan NDFs for use in FTS-2 data reduction pipeline

Description:

Split NDF files containing multiple FTS-2 scans into separate files each containing a single (unidirectional) scan. The output file names are constructed by appending the number of the scan within each input file to the corresponding output base file name.

Parameters:

BANDPASS = REAL (Read)
Distance to extract on either side of center (mm). [0.0]

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Base names for output files. The actual files written will use these names plus a 3-digit integer identifier.

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a NULL (!) value is supplied no file is created. [!]

FTS2TRANSCORR
Corrects for atmospheric transmission across the spectral dimension

Description:

Corrects for atmospheric transmission across the spectral dimension, given the current PWV and elevation. Transmission data is stored in the form of a wet and dry Tau vs frequency table in the calibration database.

Parameters:

DEBUG = LOGICAL (Read)
If debug, does NOT include the dry component of the atnoshpheric transmission in the computation

IN = NDF (Read)
Input files to be transformed.

OUT = NDF (Write)
Output files.

TAU = NDF (Read)
TAU files.

GAU2FIT
Fit a two-component Gausian to a map of an extended source

Description:

This routine finds the parameters of a one- or two-component Gaussian beam by doing a least squares fit to a supplied 2D NDF containing an image of a compact source. The source need not be a point source - the source is assumed to be a sharp-edged circular disc of specified radius (i.e. a simple model of a planet). On each iteration of the fitting process, this source model is convolved with the candidate beam and the residuals with the supplied image are then found. The parameters of the beam are modified on each iteration to minimise the sum of the squared residuals. The beam consists of one or two (see parameter FITTWO) concentric Gaussians added together. The first Gaussian has a fixed amplitude of 1.0. Each candidate beam is normalised to a total data sum of unity before being used. The following parameters are varied during the minimisation process:

- The FWHM of the first Gaussian.
- The FWHM and amplitude of the second Gaussian (if used).
- The centre position of the beam within the supplied image.
- The peak value in the source.
- The background level (but only if parameter FITBACK is TRUE).

Note, if the two components have the same FWHM, they become degenerate (i.e. indistinguishable), which causes problems for the minimisation process. To avoid this, the cost function that is minimised (i.e. the sum of the squared residuals between the smoothed source model and the supplied map) is multiplied by a factor that increases the cost as the FWHM for the two beams becomes more similar. Consequently the second component will always have a larger FWHM than the first beam.

In addition, if parameter FITBACK is TRUE and the FWHM of the second component is similar in size to (or larger than) the data array, it becomes difficult for the minimisation process to distinguish between the second component and a constant background level, again making minimisation difficult. For this reason very wide second components are discouraged by increasing the cost if the second component FWHM is similar to (or larger than) the size of the array.

In addition large amplitude second components are discouraged by increasing the cost for large amplitude second components.

Parameters:

AMP2 = _DOUBLE (Write)

An output parameter holding the amplitude of the second fitted Gaussian relative to the first. For instance, if AMP2 is 0.15 it means that the amplitude of the second component is 15% of the amplitude of the first component.

BACK = _DOUBLE (Write)
An output parameter holding the fitted background level. This will be set to zero if FITBACK is FALSE.

CENTRE = LITERAL (Write)
An output parameter holding the fitted position of the beam centre, in the celestial co-ordinate frame of the NDF (which may or may not be the current co-ordinate system). The string consists of a space separated list of two formatted axis values.

FITBACK = _LOGICAL (Read)
If FALSE, the background level is fixed at zero throughout the minimisation. If TRUE, the background level is included as one of the free parameters in the fit. [TRUE]

FITTWO = _LOGICAL (Read)
If TRUE, the beam consists of two Gaussians. Otherwise only one Gaussian is used. [TRUE]

FWHM1 = _DOUBLE (Write)
An output parameter holding the FWHM of the first fitted Gaussian in arc-sec.

FWHM2 = _DOUBLE (Write)
An output parameter holding the FWHM of the second fitted Gaussian in arc-sec.

IN = NDF (Read)
Input image containing the source to be fitted. Note, if parameter FITBACK is FALSE, the source should be on a zero background. The WCS information must contain a pair of celestial axes, but they need not be the current coordinate system.

INITCENTRE = LITERAL (Read)
An initial guess at the position of the beam centre, in the celestial co-ordinate frame of the NDF (which may or may not be the current co-ordinate system). A space or comma separated list of formatted axis values should be supplied.

LOGFILE = LITERAL (Read)
Name of a log file in which to store a table holding the beam parameters and cost function at each iteration of the minimisation. This table is in TOPCAT " ascii" format. [!]

OUT = NDF (Write)
Output NDF containing the fitted model. This is the source model convolved with the final beam.

RADIUS = _DOUBLE (Read)
The radius of the sharp-edge circular source, in arc-seconds. Must be no smaller than 1.0 arcsec.

RMS = _DOUBLE (Write)
An output parameter holding the RMS residual between the fitted model and the supplied image.

GSD2ACSIS
Convert a GSD format DAS data file to an ACSIS format NDF

Description:

Opens a GSD file for reading, and checks the version (currently only supports GSD Version 5.3). The data are converted to ACSIS format and written to disk. Metadata are converted to appropriate FITS headers.

Parameters:

DIRECTORY = _CHAR (Read)
Directory for output ACSIS files. A NULL value will use the current working directory. This command will create a subdir in this directory named after the observation number.

IN = CHAR (Read)
Name of the input GSD file to be converted.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OBSNUM = INT (Read)
Observation number for files prior to Feb 03. For newer observations this parameter is not required. Default value will be the observation number read from the file but prior to Feb 03 this number was the number within the project rather than the number from the night and may lead to name clashes since ACSIS data are numbered for a UT date.

Notes:

- Whilst this command does a reasonable job of converting common data to ACSIS format it still has to undergo extensive testing to ensure that it is always doing the correct thing. Testing of this command and comparing its results with SPECX maps will be welcomed.
- The ORAC-DR recipe defaults to REDUCE_SCIENCE. The exceptions are as follows:
- REDUCE_SCIENCE_CONTINUUM for solar-system objects (Sun, Moon, planets, Titan);
- REDUCE_POINTING for a FIVEPOINT observation type and a DAS backend;
- REDUCE_FOCUS for a focus observation type; and
- REDUCE_SCIENCE_BROADLINE for objects with radial velocities above 120 km/s.

Related Applications :

SMURF: MAKECUBE, GSDSHOW; CONVERT: SPECX2NDF; SPECX; GSDPRINT; JCMTDR.

GSDSHOW
Display the contents of headers and arrays for GSD files

Description:

Opens a GSD file for reading, and checks the version (currently only supports GSD version 5.3). Then displays the contents of the headers and data arrays.

Parameters:

DESCRIPTIONS = _LOGICAL (Read)
Flag for showing header descriptions. [FALSE]

IN = CHAR (Read)
Name of the input GSD file to be listed.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

SHOWDATA = _LOGICAL (Read)
Flag for showing data array. [FALSE]

Related Applications :

SMURF: GSD2ACSIS; GSDPRINT; SPECX; JCMTDR

IMPAZTEC
Import AzTEC NetCDF files and produce SCUBA-2 ICD-compliant files

Description:

Uses the NetCDF library to import raw AzTEC data files and save to NDF files in a format approximating the SCUBA-2 ICD so that they may subsequently be read by other SMURF routines to make maps.

Parameters:

IN = _CHAR (Read)
Name of the input NetCDF file to be converted. This name should include the .nc extension.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = _CHAR (Read)
Output NDF file.

Notes:

- No base coordinates were stored in netcdf files.
- Time is presently inaccurate and requires an optimization routine to calculate the time that makes Az/El and RA/Dec consistent.
- This command is untested and probably does not work.

Related Applications :

SMURF: MAKEMAP

JSADICER
Dice an image or cube into JSA tiles

Description:

This routine creates multiple output NDFs by dicing a supplied 2D- or 3D- NDF up into JSA tiles (i.e. it is the inverse of JSAPASTER). The spatial WCS of the input NDF must matches the JSA all-sky grid.

Parameters:

IN = NDF (Read)

The input 2D or 3D NDF - it need not have been created by a smurf task. Currently, it should be gridded on the JSA all-sky pixel grid, which means that in practice it probably will have been created by MAKEMAP or MAKECUBE.

INSTRUMENT = LITERAL (Read)

The JCMT instrument (different instruments have different tiling schemes and pixel sizes). The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . The dynamic default is determined from the input NDF if possible. If this cannot be done, then no dynamic default is provided, and the user is prompted for a value if none was supplied on the command line. []

JSATILELIST() = _INTEGER (Write)

Returned holding the zero-based indices of the created JSA tiles. The number of such indices is given the " NTILE" parameter

NTILE = _INTEGER (Write)

AN output parameter which is returned holding the number of output NDFs created.

OUT = NDF (Write)

The base-name for the output NDFs. The names will be formed by appending the tile number to the basename, preceded by an underscore. A null(!) value causes the name of the input NDF to be used. [!]

OUTFILES = LITERAL (Write)

The name of a text file to create, in which to put the names of all the output NDFs created by this application via parameter OUT (one per line). If a null (!) value is supplied no file is created. [!]

PROJ = LITERAL (Read)

Determines the projection used by the output NDFs. The allowed values are " HPX" (HPX projection centred on RA=0h), " HPX12" (HPX projection centred on RA=12h), " XPHN" (XPH projection centred on the north pole) and " XPHS" (XPH projection centred on the south pole). A null (!) value causes " HPX" to be used. [" HPX"]

TRIM = _INTEGER (Read)

A zero or negative value results in each output NDF covering the full area of the corresponding JSAtile. A value of one results in each output NDF being cropped to the bounds of the supplied NDF. A value of two or more results in each output NDF being cropped to remove any blank borders. [2]

JSAPASTER
Paste several JSA tiles into a single mosaic

Description:

This routine creates a single output NDF by pasting together a supplied list of 2D- or 3D-JSA tiles (i.e. it is the inverse of JSADICER). The spatial WCS of each input NDF must matches the JSA all-sky pixel grid. The output NDF will usually be gridded using an HPX projection, but an XPH projection will be used if the mosaic would cross a discontinuity in the HPX projection.

Parameters:

IN = NDF (Read)

A group of input JSA tiles.

INSTRUMENT = LITERAL (Read)

The JCMT instrument (different instruments have different tiling schemes and pixel sizes). The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . The dynamic default is determined from the input NDF if possible. If this cannot be done, then no dynamic default is provided, and the user is prompted for a value if none was supplied on the command line. []

OUT = NDF (Write)

The mosaic.

JSATILEINFO
Return information about a specified sky tile

Description:

This routine returns information about a single standard sky tile for a named JCMT instrument (specified by parameter ITILE). This includes the RA and Dec. of the tile centre, the tile size, the tile WCS, and whether the NDF containing the accumulated co-added data for the tile currently exists. If not, there is an option to create it and fill it with zeros (see parameter CREATE). In addition, an NDF can be created containing the tile index at every point on the whole sky (see parameter ALLSKY).

Also, the bounds of the overlap in pixel indices between the tile and a specified NDF or Region can be found (see parameter TARGET).

The environment variable JSA_TILE_DIR should be defined prior to using this command, and should hold the path to the directory in which the NDFs containing the accumulated co-added data for each tile are stored. Tiles for a specified instrument will be stored within a sub-directory of this directory (see parameter INSTRUMENT). If JSA_TILE_DIR is undefined, the current directory is used.

Parameters:

ALLSKY = NDF (Write)

If not null (!), this is the name of a new 2D NDF to create holding the tile index at every point on the sky. The pixel size in this image is much larger than in an individual tile in order to keep the NDF size managable. Each pixel holds the integer index of a tile. Pixels that have no corresponding tile hold a bad value. [!]

CREATE = _LOGICAL (Read)

Indicates if the NDF containing co-added data for the tile should be created if it does not already exist. If TRUE, and if the tile' s NDF does not exist on entry to this command, a new NDF will be created for the tile and filled with zeros. The instrument subdirectory within the JSA_TILE_DIR directory will be created if it does not already exist. [FALSE]

DECCEN = _DOUBLE (Write)

An output parameter to which is written the ICRS Declination of the tile centre, in radians.

EXISTS = _LOGICAL (Write)

An output parameter that is set TRUE if the NDF containing co-added data for the tile existed on entry to this command (FALSE otherwise).

HEADER = LITERAL (Read)

The name of a new text file in which to store the WCS and extent of the tile in the form of a set of FITS-WCS headers. [!]

INSTRUMENT = LITERAL (Read)

The JCMT instrument (different instruments have different tiling schemes and pixel sizes). The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . NDFs

containing co-added data for the selected instrument reside within a corresponding sub-directory of the directory specified by environment variable JSA_TILE_DIR. These sub-directories are called " scuba2-450" , " scuba2-850" , " acsis" and " das" .

ITILE = _INTEGER (Read)
The index of the tile about which information is required. The first tile has index 0. The largest allowed tile index is always returned in output parameter MAXTILE. If a null (!) value is supplied for ITILE, the MAXTILE parameter is still written, but the command will then exit immediately without further action (and without error).

LBND(2) = _INTEGER (Write)
An output parameter to which are written the lower pixel bounds of the NDF containing the co-added data for the tile.

LOCAL = _LOGICAL (Read)
If TRUE, the FITS reference point is put at the centre of the tile. Otherwise, it is put at RA=0 Dec=0. [FALSE]

MAXTILE = _INTEGER (Write)
An output parameter to which is written the largest tile index associated with the instrument specified by parameter INSTRUMENT.

PROJ = LITERAL (Read)
Determines the JSA projection to use. The allowed values are " HPX" (HPX projection centred on RA=0h), " HPX12" (HPX projection centred on RA=12h), " XPHN" (XPH projection centred on the north pole) and " XPHS" (XPH projection centred on the south pole). A null (!) value causes " HPX" to be used. [" HPX"]

RACEN = LITERAL (Write)
An output parameter to which is written the ICRS Right Ascension of the tile centre, in radians.

REGION = LITERAL (Read)
The name of a new text file in which to store the WCS and extent of the tile in the form of an AST Region. [!]

SIZE = _DOUBLE (Write)
An output parameter to which is written the arc-length of each side of a square bounding box for the tile, in radians.

TARGET = LITERAL (Read)
Defines the region of interest. The pixel index bounds of the area of the specified tile that overlaps this region are found and reported (see parameter TLBND and TUBND). The value supplied for TARGET can be either the name of a text file containing an AST Region, or the path for an NDF, in which case a Region is created that covers the region of sky that maps onto the rectangular pixel grid of the NDF. [!]

TILENDF = LITERAL (Write)
An output parameter to which is written the full path to the NDF containing co-added data for the tile. Note, it is not guaranteed that this NDF exists on exit from this command (see parameters EXISTS and CREATE).

TLBND(2) = _INTEGER (Write)
An output parameter to which are written the lower pixel bounds of the area of the tile NDF that overlaps the region specified by parameter TARGET. If there is no overlap, the TLBND values are returned greater than the TUBND values.

TUBND(2) = _INTEGER (Write)
An output parameter to which are written the upper pixel bounds of the area of the tile NDF that overlaps the region specified by parameter TARGET. If there is no overlap, the TLBND values are returned greater than the TUBND values.

UBND(2) = _INTEGER (Write)
An output parameter to which are written the upper pixel bounds of the NDF containing the co-added data for the tile.

Related Applications :

SMURF: MAKECUBE, MAKEMAP, TILELIST.

JSATILELIST
List the sky tiles that overlap a given set of data files or an AST Region

Description:

This routine returns a list containing the indices of the sky tiles (for a named JCMT instrument) that overlap a supplied AST Region, map, cube or group of raw data files.

Parameters:

IN = LITERAL (Read)

Specifies the region of sky for which tiles should be listed. It may be:

- The name of a text file containing an AST Region. The Region can be either 2D or 3D but must include celestial axes.
- The path to a 2- or 3-D NDF holding a reduced map or cube. This need not necessarily hold JCMT data, but must have celestial axes in its current WCS Frame.
- A group of raw JCMT data files.
- A null (!) value, in which case a polygon region is used as defined by the parameters VERTEX_RA and VERTEX_DEC.

INSTRUMENT = LITERAL (Read)

The JCMT instrument (different instruments have different tiling schemes and pixel sizes). The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . If one or more NDFs are supplied for parameter IN, then a dynamic default is determined if possible from the first NDF. If this cannot be done, or if a Region is supplied for parameter IN, then no dynamic default is provided, and the user is prompted for a value if none was supplied on the command line. []

PROJ = LITERAL (Write)

The type of JSA projection that should be used to describe the area of sky covered by the returned lost of tiles. Will be one of " HPX" , " HPX12" , " XPHN" or " XPHS" . The choice is made to minimise the possibility of a projection discontinuity falling within the sky area covered by the tiles.

TILES(∗) = _INTEGER (Write)

An output parameter to which is written the list of integer tile indices.

VERTEX_DEC(∗) = _DOUBLE (Read)

The ICRA Dec value at each vertex of a polygon, in degrees. Only used if IN is null.

VERTEX_RA(∗) = _DOUBLE (Read)

The ICRA RA value at each vertex of a polygon, in degrees. Only used if IN is null.

Tile Definitions :

It should never be necessary to know the specific details of the tiling scheme used by SMURF. But for reference, it works as follows:

The whole sky is covered by an HPX (HEALPix) projection containing 12 basic square facets, the reference point of the projection is put at (RA,Dec)=(0,0) (except for facet six that has a reference point of (12h,0)). The projection plane is rotated by 45 degrees so that the edges of each facet are parallel to X and Y (as in Fig.3 of the A&A paper " Mapping on the HEALPix grid" by Calabretta and Roukema). Each facet is then divided up into NxN tiles, where N is 64 for SCUBA-2 and 128 for ACSIS. Each tile is then divided into PxP pixels, where P is 412 for ACSIS, 825 for SCUBA-2 850 um, 1650 for SCUBA-2 450 um. Facets are numbered from 0 to 11 as defined in the HEALPix paper (Gorsky et. al. 2005 ApJ 622, 759) (note that the facet six is split equally into two triangles, one at the bottom left and one at the top right of the projection plane). Within a facet, tiles are indexed using the " nested" scheme described in the HEALPix paper. This starts with pixel zero in the southern corner of the facet. The even bits number the position in the north-east direction and the odd bits number the position in the north-west direction. All the tiles in the first facet come first, followed by all the tiles in the second facet, etc.

This is a fairly complex scheme. To help understanding, the SMURF:TILEINFO command can create an all-sky map in which each pixel corresponds to a single tile, and has a pixel value equal to the corresponding tile index. Displaying this map can help to visualise the indexing scheme described above.

Related Applications :

SMURF: MAKECUBE, MAKEMAP, TILEINFO.

MAKECUBE
Regrid ACSIS spectra into a data cube

Description:

This routine converts one or more raw data cubes, spanned by (frequency, detector number, time) axes, into an output cube spanned by (celestial longitude, celestial latitude, frequency) axes.

Optionally, the output cube can be split up into several separate NDFs, each containing a spatial tile extracted from the full cube (see parameter JSATILES and TILEDIMS). These tiles abut exactly in pixel co-ordinates and can be combined (for example) using KAPPA:PASTE.

In addition, there is an option to divide the output up into separate polarisation angle bins (see parameter POLBINSIZE). If this option is selected, each tile is split up into several output NDFs (all within the same container file), each one containing the input data relating to a particular range of polarisation angle.

The full output cube can be either a regularly gridded tangent-plane projection of the sky, or a sparse array (see parameter SPARSE). If a tangent plane projection is selected, the parameters of the projection from sky to pixel grid co-ordinates can be specified using parameters CROTA, PIXSIZE, REFLAT, REFLON. Alternatively, parameter AUTOGRID can be set true, in which case projection parameters are determined automatically in a manner that favours projections that place samples centrally within pixels. Alternatively, a reference NDF can be supplied (see parameter REF), in which case the same pixel grid will be used for the output cube.

Variance values in the output can be calculated either on the basis of the spread of input dat avalues contributing to each output pixel, or on the basis of the system-noise temperature values supplied in the input NDFs (see parameter GENVAR).

Parameters:

ALIGNSYS = _LOGICAL (Read)

If TRUE, then the spatial positions of the input data are aligned in the co-ordinate system specified by parameter SYSTEM. Otherwise, they are aligned in the ICRS co-ordinate system. For instance, if the output co-ordinate system is AZEL, then setting ALIGNSYS to TRUE will result in the AZEL values of the input data positions being compared directly, disregarding the fact that a given AZEL will correspond to different positions on the sky at different times. [FALSE]

AUTOGRID = _LOGICAL (Read)

Only accessed if a null value is supplied for parameter REF. Determines how the dynamic default values should be determined for the projection parameters CROTA, PIXSIZE, REFLAT, REFLON, REFPIX1 and REFPIX2. If TRUE, then default projection parameters are determined by adjusting the grid until as many data samples as possible fall close to the centre of pixels in the output cube. If FALSE, REFLON/REFLAT are set to the first pointing BASE position, CROTA is set to the MAP_PA value in the FITS header (converted to the requested sky co-ordinate system), PIXSIZE is set to 6 arcseconds, and REFPIX1/REFPIX2 are both set to zero. [FALSE]

REFPIX1 = _DOUBLE (Read)

Controls the precise placement of the spatial tangent point on the first pixel axis of the output cube. The position of the tangent point on the sky is specified by REFLON/REFLAT, and this sky position is placed at grid coordinates specified by REFPIX1/REFPIX2. Note, these grid coordinates refer to an interim grid coordinate system that does not depend on the values supplied for LBND, rather than the final grid coordinate system of the output cube. Therefore, if values are supplied for REFPIX1/REFPIX2, they should be copies of the values written to output parameter PIXREF by a previous run of MAKECUBE. The REFPIX and PIXREF parameters allow an initial run of MAKECUBE with AUTOGRID=YES to generate projection parameters that can then be re-used in subsequent runs of MAKECUBE with AUTOGRID=NO in order to force MAKECUBE to use the same pixel grid. If a null (!) value is supplied, default values will be used for REFPIX1/2 - either the autogrid values (if AUTOGRID=YES) or (0,0) (if AUTOGRID=NO). [!]

REFPIX2 = _DOUBLE (Read)

Controls the precise placement of the spatial tangent point on the second pixel axis of the output cube. See REFPIX1. [!]

BADMASK = LITERAL (Read)

A string determining the way in which bad pixels are propagated from input to output. The " AND" scheme uses all input data (thus reducing the noise in the output) and also minimises the number of bad pixels in the output. However, the memory requirements of the " AND" scheme can be excessive. For this reason, two other schemes, " FIRST" and " OR" , are provided which greatly reduce the memory requirements, at the expense either of introducing more bad pixels into the output (" OR") or producing higher output noise levels (" FIRST"). The value supplied for this parameter is used only if SPREAD is set to " Nearest" (otherwise " AND" is always used):

- " FIRST" – The bad-pixel mask in each output spectrum is inherited from the first input spectrum that contributes to the output spectrum. Any subsequent input spectra that contribute to the same output spectrum but which have a different bad-pixel mask are ignored. So an output pixel will be bad if and only if the corresponding pixel in the first input NDF that contributes to it is bad. Since this scheme ignores entire input spectra if they do not conform to the expected bad-pixel mask, the noise in the output can be higher than using the other schemes. However, this scheme has the benefit of using much less memory than the " AND" scheme, and will in general produce fewer bad pixels in the output than the " OR" scheme.
- " OR" – The bad pixel mask in each output spectrum is the union (logical OR) of the bad pixel masks for all input spectra that contribute to the output spectrum. So an output pixel will be bad if any of the input pixels that contribute to it are bad. This scheme will in general produce more bad output pixels than the " FIRST" scheme, but the non-bad output pixels will have a lower noise because, unlike " FIRST" , all the contributing input data are coadded to produce the good output pixels. Like " FIRST" , this scheme uses much less memory than " AND" .
- " AND" – The bad pixel mask for each output spectrum is the intersection (logical

AND) of the bad pixel masks for all input spectra that contribute to the output spectrum. So an output pixel will be bad only if all the input pixels that contribute to it are bad. This scheme will produce fewer bad output pixels and will also give lower output noise levels than " FIRST" or " OR" , but at the expense of much greater memory requirements.

[" OR"]

CATFRAME = LITERAL (Read)

A string determining the co-ordinate Frame in which positions are to be stored in the output catalogue associated with parameter OUTCAT. The string supplied for CATFRAME can be one of the following:

- A Domain name such as SKY, AXIS, PIXEL, etc.
- An integer value giving the index of the required Frame.
- An IRAS90 Sky Co-ordinate System (SCS) values such as EQUAT(J2000) (see SUN/163).

If a null (!) value is supplied, the positions will be stored in the current Frame of the output NDF. [!]

CATEPOCH = _DOUBLE (Read)

The epoch at which the sky positions stored in the output catalogue were determined. It will only be accessed if an epoch value is needed to qualify the co-ordinate Frame specified by COLFRAME. If required, it should be given as a decimal years value, with or without decimal places (" 1996.8" for example). Such values are interpreted as a Besselian epoch if less than 1984.0 and as a Julian epoch otherwise.

CROTA = _REAL (Read)

Only accessed if a null value is supplied for parameter REF. The angle, in degrees, from north through east (in the co-ordinate system specified by the SYSTEM parameter) to the second pixel axis in the output cube. The dynamic default value is determined by the AUTOGRID parameter. []

DETECTORS = LITERAL (Read)

A group of detector names to include in, or exclude from, the output cube. If the first name starts with a minus sign, then the specified detectors are excluded from the output cube (all other detectors are included). Otherwise, the specified detectors are included in the output cube (all other detectors are excluded). If a null (!) value is supplied, data from all detectors will be used. [!]

EXTRACOLS = LITERAL (Read)

A group of names specifying extra columns to be added to the catalogue specified by parameter OUTCAT. Each name should be the name of a component in the JCMTState extension structure. For each name in the group, an extra column is added to the output catalogue containing the value of the named extension item for every table row (i.e. for each data sample). These extra columns can be viewed and manipulated with general-purpose FITS table tools such as TOPCAT, but will not be displayed by the KAPPA:LISTSHOW command. One use for these extra columns is to allow the catalogue to be filtered (e.g. by TOPCAT) to remove samples that meet (or do not meet) some specified requirement specified by the JCMTState contents. No extra columns are added if a null (!) value is supplied. [!]

FBL( ) = _DOUBLE (Write)
Sky co-ordinates (radians) of the bottom-left corner of the output cube (the corner with the smallest PIXEL dimension for Axis 1 and the smallest pixel dimension for Axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel. If SPARSE mode is enabled the positions reported will not be reliable.

FBR( ) = _DOUBLE (Write)
Sky co-ordinates (radians) of the bottom right corner of the output cube (the corner with the largest PIXEL dimension for Axis 1 and the smallest pixel dimension for Axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel. If SPARSE mode is enabled the positions reported will not be reliable.

FLBND( ) = _DOUBLE (Write)
The lower bounds of the bounding box enclosing the output cube in the selected output WCS Frame. The values are calculated even if no output cube is created. Celestial axis values will be in units of radians, spectral-axis units will be in the same units as the input frameset (matching those used in the SPECBOUNDS parameter). The parameter is named to be consistent with KAPPA:NDFTRACE output. Note, the stored values correspond to the outer edges of the first pixel, not to the pixel centre.

FUBND( ) = _DOUBLE (Write)
The upper bounds of the bounding box enclosing the output cube in the selected output WCS Frame. The values are calculated even if no output cube is created. Celestial axis values will be in units of radians, spectral-axis units will be in the same units of the input frameset (matching those used in the SPECBOUNDS parameter). The parameter is named to be consistent with KAPPA:NDFTRACE output. Note, the stored values correspond to the outer edges of the first pixel, not to the pixel centre.

FTL( ) = _DOUBLE (Write)
Sky co-ordinates (radians) of the top left corner of the output cube (the corner with the smallest PIXEL dimension for Axis 1 and the largest pixel dimension for Axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel. If SPARSE mode is enabled the positions reported will not be reliable.

FTR( ) = _DOUBLE (Write)
Sky co-ordinates (radians) of the top right corner of the output cube (the corner with the largest PIXEL dimension for Axis 1 and the largest pixel dimension for Axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel. If SPARSE mode is enabled the positions reported will not be reliable.

GENVAR = LITERAL (Read)
Indicates how the Variance values in the output NDF are to be calculated. It can take any of the following values:

- " Spread" – the output Variance values are based on the spread of input data values contributing to each output pixel. This option is not available if parameter SPARSE is set TRUE. If the BADMASK value is " OR" or " FIRST" , then a single variance value will be produced for each output spectrum (i.e. all channels in an

output spectrum will have the same variance value). If BADMASK is " AND" , then an independent variance value will be calculated for each channel in each output spectrum.

- " Tsys" – the output Variance values are based on the system noise temperature values supplied in the input NDFs. Since each input spectrum is characterised by a single Tsys value, each output spectrum will have a constant Variance value (i.e. all channels in an output spectrum will have the same variance value).
- " None" – no output Variance values are created.

[" Tsys"]

IN = NDF (Read)
Input raw data file(s)

INWEIGHT = _LOGICAL (Read)
Indicates if the input spectra should be weighted when combining two or more input spectra together to form an output spectrum. If TRUE, the weights used are the reciprocal of the variances associated with the input spectra, as determined from the Tsys values in the input. [TRUE]

JSATILES = _LOGICAL (Read)
If TRUE, the output cube is created on the JSA all-sky pixel grid, and is split up into individual JSA tiles. Thus multiple output NDFs may be created, one for each JSA tile that touches the cube. Each of these output NDFs will have the tile index number appended to the end of the path specified by parameter " OUT" . If " JSATILES" is TRUE, the " REF" parameter is ignored. [FALSE]

JSATILELIST() = _INTEGER (Write)
If parameter " JSATILES" is set TRUE, the zero-based indices of the created JSA tiles will be written to this output parameter. The number of such indices is given the " NTILE" parameter

LBND(2) = _INTEGER (Read)
An array of values giving the lower pixel-index bound on each spatial axis of the output NDF. The suggested default values encompass all the input spatial information. The supplied bounds may be modified if the parameter TRIM takes its default value of TRUE. []

LBOUND(3) = _INTEGER (Write)
The lower pixel bounds of the output NDF. Note, values will be written to this output parameter even if a null value is supplied for parameter OUT.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NTILE = _INTEGER (Write)
The number of output tiles used to hold the entire output array (see parameter JSATILES and TILEDIMS). If no input data falls within a specified tile, then no output NDF will be created for the tile, but (if JSATILES is FALSE) the tile will still be included in the tile numbering scheme.

NPOLBIN = _INTEGER (Write)
The number of polarisation angle bins used to hold the entire output data (see parameter POLBINSIZE).

OUT = NDF (Write)

Output file. If a null (!) value is supplied, the application will terminate early without creating an output cube, but without reporting an error. Note, the pixel bounds which the output cube would have had will still be written to output parameters LBOUND and UBOUND, even if a null value is supplied for OUT. If the output cube is split up into multiple output NDFs (e.g. an NDF for each tile – see parameter TILEDIMS – or for each polarisation angle bin – see parameter POLBINSIZE), then the value supplied for " OUT" will be used as the root name to which other strings are appended to create the name of each output NDF.

OUTCAT = FILENAME (Write)

An output catalogue in which to store all the spatial detector positions used to make the output cube (i.e. those selected using the DETECTORS parameter). By default, the stored positions are in the same sky co-ordinate system as the current Frame in the output NDF (but see parameter CATFRAME). The label associated with each row in the catalogue is the detector name. The detector positions in the catalogue are ordered as follows: all the positions for the first input NDF come first, followed by those for the second input NDF, etc. Within the group of positions associated with a single input NDF, the positions for the first time slice come first, followed by the positions for the second time slice, etc. If a null value (!) is supplied, no output catalogue is produced. See also parameter CATFRAME. [!]

OUTFILES = LITERAL (Write)

The name of text file to create, in which to put the names of all the output NDFs created by this application via parameter OUT (one per line). If a null (!) value is supplied no file is created. [!]

PARAMS(2) = _DOUBLE (Read)

An optional array which consists of additional parameters required by the Sinc, SincSinc, SincCos, SincGauss, Somb, SombCos, and Gauss spreading methods (see parameter SPREAD).

PARAMS(1) is required by all the above schemes. It is used to specify how many pixels on either side of the output position (that is, the output position corresponding to the centre of the input pixel) are to receive contributions from the input pixel. Typically, a value of 2 is appropriate and the minimum allowed value is 1 (i.e. one pixel on each side). A value of zero or fewer indicates that a suitable number of pixels should be calculated automatically. [0]

PARAMS(2) is required only by the SombCos, Gauss, SincSinc, SincCos, and SincGauss schemes. For the SombCos, SincSinc, and SincCos schemes, it specifies the number of pixels at which the envelope of the function goes to zero. The minimum value is 1.0, and the run-time default value is 2.0. For the Gauss and SincGauss scheme, it specifies the full-width at half-maximum (FWHM) of the Gaussian envelope. The minimum value is 0.1, and the run-time default is 1.0. On astronomical images and spectra, good results are often obtained by approximately matching the FWHM of the envelope function, given by PARAMS(2), to the point-spread function of the input data. []

PIXREF(2) = _DOUBLE (Write)

The grid coordinates used for the reference pixel, within the interim grid coordinate system. See REFPIX1.

PIXSIZE(2) = _REAL (Read)

Only accessed if a null value is supplied for parameter REF. Pixel dimensions in the output image, in arcseconds. If only one value is supplied, the same value will be used for both axes. The dynamic default value is determined by the AUTOGRID parameter. []

POINTING = LITERAL (Read)

The name of a text file containing corrections to the pointing read from the input data files. If null (!) is supplied, no corrections are used. Note - this parameter is ignored unless parameter USEDETPOS is set to FALSE. If a file is supplied, it should start with one or more lines containing " #" in column one. These are comment lines, but if any comment line has the form " # SYSTEM=AZEL" or " # SYSTEM=TRACKING" then it determines the system in which the pointing correction are specified (SYSTEM defaults to AZEL). The last comment line should be a space-separated list of column names, including " TAI" , " DLON" and " DLAT" . Each remaining line should contain numerical values for each column, separated by white space. The TAI column should contain the TAI time given as an MJD. The DLON and DLAT columns should give arc-distance offsets parallel to the longitude and latitude axes, in arc-seconds. The TAI values should be monotonic increasing with row number. The longitude and latitude axes are either AZEL or TRACKING as determined by the SYSTEM value in the header comments. Blank lines are ignored. The DLON and DLAT values are added onto the SMU jiggle positions stored in the JCMTSTATE extension of the input NDFs. DLON and DLAT values for non-tabulated times are determined by interpolation.

If you need to apply two sets of pointing corrections, one in TRACKING and one in AZEL, you can include two tables (one for each system) in a single text file. Both tables should use the format described above. The two tables must be separated by a line containing two or more minus signs with no leading spaces. [!]

POLBINSIZE = _REAL (Read)

This parameter is only prompted for if the input files contain polarisation data. The supplied value is used as the bin size (in degrees) for grouping polarisation analyser angles. The first bin is centred at the angle given by parameter POLBINZERO. The " analyser angle" is the anti-clockwise angle from celestial north (in the system chosen by parameter SYSTEM) to the axis of the " effective analyser" - a rotating analyser that would have the same effect as the combination of fixed analyser and half-wave plate actually present in the polarimeter. The supplied value for POLBINSIZE will be modified if required to ensure that a whole number of bins is used to cover the complete range of analyser angles (0 to 360 degrees). A separate output cube will be created for each bin that is not empty, and each output NDF will contain a POLPACK extension suitable for use with the POLPACK:POLCAL command. These NDFs are all stored in a single HDS container file (one per tile) with the name specified by parameter OUT. Within this container file, each cube will be held in a component with name of the form " P<N>" appended to the end, where " <N>" is an integer bin index. The largest value of N is written to output parameter NPOLBIN. If a null value (!) is supplied, then a single output NDF (without POLPACK extension) is created for each tile, containing all input data.

POLBINZERO = _REAL (Read)

This parameter is only prompted for if the input files contain polarisation data. It is

the analyser angle (in degrees) at the centre of the first analyser angle bin. A value of zero corresponds to north in the celestial co-ordinate system specified by parameter SYSTEM. [0]

POSERRFATAL = _LOGICAL (Read)
If a true value is supplied, then an error is reported and the application terminates if a significant difference is found between the detector positions array (RECEPPOS) and positions implied by the FPLANEX/Y arrays. If a false value is supplied, a warning is issued but the application proceeds. See also parameters POSERRMAX and USEDETPOS. [FALSE]

POSERRMAX = _REAL (Read)
Defines the maximum insignificant discrepancy between the detector positions array (RECEPPOS) and positions implied by the FPLANEX/Y arrays, in units of arcseconds. See parameter POSERRFATAL. [3.0]

REF = NDF (Read)
An existing NDF that is to be used to define the output grid, or the string " JSA" . If an NDF is supplied, the output grid will be aligned with the supplied reference NDF. The NDF need not be three-dimensional. For instance, a two-dimensional image can be supplied in which case the spatial axes of the output cube will be aligned with the reference image and the spectral axis will be inherited form the first input NDF. If " JSA" is supplied, the JSA all-sky pixel grid will be used (note, the cube will still be created as a single NDF - if multiple NDFs, one for each JSA tile, are required, the " JSATILES" parameter should beset TRUE instead of using the " REF" parameter). If a null (!) value is supplied then the output grid is determined by parameters AUTOGRID, REFLON, REFLAT, etc. [!]

REFLAT = LITERAL (Read)
Only accessed if a null value is supplied for parameter REF. The formatted celestial-latitude value at the tangent point of the spatial projection in the output cube. This should be provided in the system specified by parameter SYSTEM. The dynamic-default value is determined by the AUTOGRID parameter. []

REFLON = LITERAL (Read)
Only accessed if a null value is supplied for parameter REF. The formatted celestial-longitude value at the tangent point of the spatial projection in the output cube. This should be provided in the system specified by parameter SYSTEM. The dynamic-default value is determined by the AUTOGRID parameter. []

SPARSE = _LOGICAL (Read)
Indicates if the spectra in the output cube should be stored as a sparse array, or as a regularly gridded array. If FALSE, pixel Axes 1 and 2 of the output cube represent a regularly gridded tangent plane projection of the sky, with parameters determined by CROTA, PIXSIZE, REFLON and REFLAT. Each input spectrum is placed at the appropriate pixel position in this three-dimensional projection, as given by the celestial co-ordinates associated with the spectrum. If SPARSE is TRUE, then each input spectrum is given an associated index, starting from 1, and the spectrum with index " I" is stored at pixel position (I,1) in the output cube (pixel Axis 2 will always have the value 1 – that is, Axis 2 is a degenerate axis that spans only a single pixel).

In both cases, the third pixel axis in the output cube corresponds to spectral position (frequency, velocity, etc).

Whatever the setting of SPARSE, the output NDF's WCS component can be used to transform pixel position into the corresponding (celestial longitude, celestial latitude, spectral position) values. However, if SPARSE is TRUE, then the inverse transformation (i.e. from (long,lat,spec) to pixel co-ordinates) will not be defined. This means, for instance, that if a sparse array is displayed as a two-dimensional image, then it will not be possible to annotate the axes with WCS values. Also, whilst KAPPA:WCSMOSAIC will succesfully align the data in a sparse array with a regularly gridded cube, KAPPA:WCSALIGN will not, since WCSALIGN needs the inverse transformation to be defined.

The dynamic default value for SPARSE depends on the value supplied for parameter AUTOGRID. If AUTOGRID is set FALSE, then SPARSE defaults to FALSE. If AUTOGRID is set TRUE, then the default for SPARSE will be TRUE if the algorithm described under the AUTOGRID parameter fails to find useful default grid parameters. If the AUTOGRID algorithm succeeds, the default for SPARSE will be FALSE. []

SPECBOUNDS = LITERAL (Read)

The bounds of the output cube on the spectral axis. Input data that falls outside the supplied range will not be included in the output cube. The supplied parameter value should be a string containing a pair of axis values separated by white space or commas. The first should be the spectral value corresponding to the lower edge of the first spectral channel in the output cube, and the second should be the spectral value corresponding to the upper edge of the last spectral channel. The supplied values should refer to the spectral system described by the WCS FrameSet of the first input NDF. To see what this is, supply a single colon (" :") for the parameter value. This will display a description of the required spectral co-ordinate system, and then re-prompt for a new parameter value. The dynamic default is determined by the SPECUNION parameter. []

SPECUNION = _LOGICAL (Read)

Determines how the default spectral bounds for the output are chosen. If a TRUE value is supplied, then the defaults for the SPECBOUNDS parameter represent the union of the spectral ranges in the input data. Otherwise, they represent the intersection of the spectral ranges in the input data. This option is only available if parameter BADMASK is set to AND. For any other value of BADMASK, a value of FALSE is always used for SPECUNION. [FALSE]

SPREAD = LITERAL (Read)

The method to use when spreading each input pixel value out between a group of neighbouring output pixels. If SPARSE is set TRUE, then SPREAD is not accessed and a value of " Nearest" is always assumed. SPREAD can take the following values:

- " Linear" – The input pixel value is divided bi-linearly between the four nearest output pixels. Produces smoother output NDFs than the nearest-neighbour scheme.
- " Nearest" – The input pixel value is assigned completely to the single nearest output pixel. This scheme is much faster than any of the others.
- " Sinc" – Uses the sinc(pi∗x) kernel, where x is the pixel offset from the interpolation point (resampling) or transformed input pixel centre (rebinning), and sinc(z)=sin(z)/z. Use of this scheme is not recommended.

- " SincSinc" – Uses the sinc(pi∗x)sinc(k∗pi∗x) kernel. A valuable general-purpose scheme, intermediate in its visual effect on NDFs between the bi-linear and nearest-neighbour schemes.
- " SincCos" – Uses the sinc(pi∗x)cos(k∗pi∗x) kernel. Gives similar results to the " Sincsinc" scheme.
- " SincGauss" – Uses the sinc(pi∗x)exp(-k∗x∗x) kernel. Good results can be obtained by matching the FWHM of the envelope function to the point-spread function of the input data (see parameter PARAMS).
- " Somb" – Uses the somb(pi∗x) kernel, where x is the pixel offset from the transformed input pixel centre, and somb(z)=2∗J1(z)/z (J1 is the first-order Bessel function of the first kind). This scheme is similar to the " Sinc" scheme.
- " SombCos" – Uses the somb(pi∗x)cos(k∗pi∗x) kernel. This scheme is similar to the " SincCos" scheme.
- " Gauss" – Uses the exp(-k∗x∗x) kernel. The FWHM of the Gaussian is given by parameter PARAMS(2), and the point at which to truncate the Gaussian to zero is given by parameter PARAMS(1).

For further details of these schemes, see the descriptions of routine AST_REBINx in SUN/211. [" Nearest"]

SYSTEM = LITERAL (Read)

The celestial co-ordinate system for the output cube. One of ICRS, GAPPT, FK5, FK4, FK4-NO-E, AZEL, GALACTIC, ECLIPTIC. It can also be given the value " TRACKING" , in which case the system used will be which ever system was used as the tracking system during in the observation. The value supplied for the CROTA parameter should refer to the co-ordinate system specified by this parameter.

The choice of system also determines if the telescope is considered to be tracking a moving object such as a planet or asteroid. If system is GAPPT or AZEL, then each time slice in the input data will be shifted in order to put the base telescope position (given by TCS_AZ_BC1/2 in the JCMTSTATE extension of the input NDF) at the same pixel position that it had for the first time slice. For any other system, no such shifts are applied, even if the base telescope position is changing through the observation. [TRACKING]

TELPOSERRMAX = _DOUBLE (Read)

Maximum separation between the actual position (TCS_TR_AC1/2) and demand position (TCS_TR_DC1/2) in the JCMTSTATE extension (arcsec). If non-zero, time slices which exceed this separation will be excluded. [0.0]

TILEBORDER = _INTEGER (Read)

Only accessed if a non-null value is supplied for parameter TILEDIMS. It gives the width, in pixels, of a border to add to each output tile. These borders contain data from the adjacent tile. This results in an overlap between adjacent tiles equal to twice the supplied border width. If the default value of zero is accepted, then output tiles will abut each other in pixel space without any overlap. If a non-zero value is supplied, then each pair of adjacent tiles will overlap by twice the given number of pixels. Pixels within the overlap border will be given a quality name of " BORDER" (see KAPPA:SHOWQUAL). [0]

TILEDIMS(2) = _INTEGER (Read)

This parameter is ignored if parameter " JSATILES" is set TRUE.

For large data sets, it may sometimes be beneficial to break the output array up into a number of smaller rectangular tiles, each created separately and stored in a separate output NDF. This can be accomplished by supplying non-null values for the TILEDIMS parameter. If supplied, these values give the nominal spatial size of each output tile, in pixels. Edge tiles may be thinner if the TRIMTILES parameter is set TRUE. In order to avoid creating very thin tiles around the edges, the actual tile size used for the edge tiles may be up to 10 % larger than the supplied value. This creation of " fat" edge tiles may be prevented by supplying a negative value for the tile size, in which case edge tiles will never be wider than the supplied absolute value.

If only one value is supplied, the supplied value is duplicated to create square tiles. Tiles are created in a raster fashion, from bottom left to top right of the spatial extent. The NDF file name specified by " out" is modified for each tile by appending " _<N>" to the end of it, where <N> is the integer tile index (starting at 1). The number of tiles used to cover the entire output cube is written to output parameter NTILES. The tiles all share the same projection and so can be simply pasted together in pixel co-ordinates to reconstruct the full size output array. The tiles are centred so that the reference position (given by REFLON and REFLAT) falls at the centre of a tile. If a tile receives no input data, then no corresponding output NDF is created, but the tile is still included in the tile numbering scheme. If a null (!) value is supplied for TILEDIMS, then the entire output array is created as a single tile and stored in a single output NDF with the name given by parameter OUT (without any " _<N>" appendage). [!]

TRIM = _LOGICAL (Read)

If TRUE, then the output cube will be trimmed to exclude any borders filled with bad values. Such borders can be caused, for instance, by one or more detectors having been excluded (see parameter DETECTORS), or by the supplied LBND and/or UBND parameter values extending beyond the available data. [TRUE]

TRIMTILES = _LOGICAL (Read)

Only accessed if the output is being split up into more than one spatial tile (see parameter TILEDIMS and JSATILES). If TRUE, then the tiles around the border will be trimmed to exclude areas that fall outside the bounds of the full sized output array. This will result in the border tiles being smaller than the central tiles. [FALSE]

UBND(2) = _INTEGER (Read)

An array of values giving the upper pixel-index bound on each spatial axis of the output NDF. The suggested default values encompass all the input spatial information. The supplied bounds may be modified if the parameter TRIM takes its default value of TRUE. []

UBOUND(3) = _INTEGER (Write)

The upper pixel bounds of the output NDF. Note, values will be written to this output parameter even if a null value is supplied for parameter OUT.

USEDETPOS = _LOGICAL (Read)

If a true value is supplied, then the detector positions are read from the detector position arrays in each input NDF. Otherwise, the detector positions are calculated on the basis of the FPLANEX/Y arrays. Both methods should (in the absence of bugs) result in identical cubes. See also parameter POSERRFATAL. [TRUE]

WEIGHTS = _LOGICAL (Read)

If TRUE, then the weights associated with the array of output pixels are stored in

an extension named ACSISRED, within the output NDF. If FALSE the weights are discarded once they have been used. These weights record the relative weight of the input data associated with each output pixel. If SPARSE is set TRUE, then WEIGHTS is not accessed and a FALSE value is assumed. [FALSE]

Notes:

- A FITS extension is added to the output NDF containing any keywords that are common to all input NDFs. To be included in the output FITS extension, a FITS keyword must be present in the NDF extension of every input NDF, and it must have the same value in all input NDFs. In addition, certain headers that relate to start and end events are propagated from the oldest and newest files respectively.
- The output NDF will contain an extension named " SMURF" containing two NDFs named " EXP_TIME" and " EFF_TIME" . In addition, if parameter SPREAD is set to " Nearest" , a third NDF called " TSYS" will be created. Each of these NDFs is 2-dimensional, with the same pixel bounds as the spatial axes of the main output NDF, so that a pixel in one of these NDFs corresponds to a spectrum in the main output NDF. EXP_TIME holds the sum of the total exposure times (Ton + Toff) for the input spectra that contributed to each output spectrum. EFF_TIME holds the sum of the effective integration times (Teff) for the input spectra that contributed to each output spectrum, scaled up by a factor of 4 in order to normalise it to the reported exposure times in EXP_TIME. TSYS holds the effective system temperature for each output spectrum. The TSYS array is not created if GENVAR is " None" or if SPREAD is not " Nearest" .
- FITS keywords EXP_TIME, EFF_TIME and MEDTSYS are added to the output FITS extension. The EXP_TIME and EFF_TIME keywords hold the median values of the EXP_TIME and EFF_TIME arrays (stored in the SMURF extension of the output NDF). The MEDTSYS keyword holds the median value of the TSYS array (also stored in the SMURF extension of the output NDF). If any of these values cannot be calculated for any reason, the corresponding FITS keyword is assigned a blank value.
- FITS keywords NUMTILES and TILENUM are added to the output FITS header. These are the number of tiles used to hold the output data, and the index of the NDF containing the header, in the range 1 to NUMTILES. See parameter TILEDIMS.

Related Applications :

SMURF: TIMESORT

MAKEMAP
Make a map from SCUBA-2 data

Description:

This command is used to create a map from SCUBA-2 data. Two techniques are provided and can be selected using the METHOD parameter.

The " REBIN" method can be used to make a map by coadding all the samples in the correct location using a number of different convolution techniques. This is useful when the time series has been processed independently of the map-maker and the map should be made " as-is" . Raw data will be flatfielded but this method will not apply any extinction correction, sky removal or filtering. It is assumed that this has been handled by other tasks prior to making the map.

The default " ITERATE" method takes a more holistic approach to map making using an iterative technique to fit for a number of models for noise and instrumental behaviour, one of which is the underlying astronomical image. Details of the map making process can be controlled using the CONFIG parameter.

Parameters:

ABORTEDAT = _INTEGER (Write)

Set to a non-zero value on exit if the iterative process was aborted because of the ABORTSOON parameter being set TRUE. The specific non-zero value returned is the number of iterations that had been completed when the iterative process was aborted. Always set to zero if ABORTSOON is FALSE.

ABORTSOON = _LOGICAL (Read)

If TRUE, then the iterative process will exit as soon as it becomes likely that the convergence criterion (specified by configuration parameter MAPTOL) will not be reached within the number of iterations allowed by configuration parameter NUMITER. [FALSE]

ALIGNSYS = _LOGICAL (Read)

If TRUE, then the spatial positions of the input data are aligned in the co-ordinate system specified by parameter SYSTEM. Otherwise, they are aligned in the ICRS co-ordinate system. For instance, if the output co-ordinate system is AZEL, then setting ALIGNSYS to TRUE will result in the AZEL values of the input data positions being compared directly, disregarding the fact that a given AZEL will correspond to different positions on the sky at different times. [FALSE]

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

CHUNKCHANGE( ) = _DOUBLE (Write)

An output array holding the final normalised map change value for each chunk.

CONFIG = GROUP (Read)

Specifies values for the configuration parameters used by the iterative map maker (METHOD=ITERATE). If the string " def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used. A full list of the available configuration parameters is available in the appendix of SUN/258. A smaller list of the more commonly used configuration parameters is available in SC/21.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The parameters available are listed in the " Configuration Parameters" appendix of SUN/258. Default values will be used for any unspecified parameters. Assigning the value " <def>" (case insensitive) to a keyword has the effect of reseting it to its default value. Unrecognised options will result in an error condition. This is done to help find spelling mistakes. [current value]

CROTA = _REAL (Read)

The angle, in degrees, from north through east (in the coordinate system specified by the SYSTEM parameter) to the second pixel axis in the output cube. Only accessed if a null value is supplied for parameter REF.

FBL( ) = _DOUBLE (Write)

Sky coordinates (radians) of the bottom left corner of the output map (the corner with the smallest PIXEL dimension for axis 1 and the smallest PIXEL dimension for axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel.

FBR( ) = _DOUBLE (Write)

Sky coordinates (radians) of the bottom right corner of the output map (the corner with the largest PIXEL dimension for axis 1 and the smallest PIXEL dimension for axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel.

FLBND( ) = _DOUBLE (Write)

The lower bounds of the bounding box enclosing the output map in the selected output WCS Frame. The values are calculated even if no output cube is created. Celestial axis values will be in units of radians. The parameter is named to be consistent with KAPPA:NDFTRACE output.

FLATMETH = _CHAR (Read)

Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

FTL( ) = _DOUBLE (Write)
Sky coordinates (radians) of the top left corner of the output map (the corner with the smallest PIXEL dimension for axis 1 and the largest PIXEL dimension for axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel.

FTR( ) = _DOUBLE (Write)
Sky coordinates (radians) of the top right corner of the output map (the corner with the largest PIXEL dimension for axis 1 and the largest PIXEL dimension for axis 2). No check is made that the pixel corresponds to valid data. Note that the position is reported for the centre of the pixel.

FUBND( ) = _DOUBLE (Write)
The upper bounds of the bounding box enclosing the output map in the selected output WCS Frame. The values are calculated even if no output cube is created. Celestial axis values will be in units of radians. The parameter is named to be consistent with KAPPA:NDFTRACE output.

FTSPORT = _CHAR (Read)
The FTS-2 port to use in calculating the mapping to sky coordinates, or null if FTS-2 was not in the beam. If set, this parameter should be " tracking" or " image" . [!]

IN = NDF (Read)
Input file(s).

IPREF = NDF (Read)
An existing NDF that is to be used to define the correction to be made for instrumental polarisation (IP). It is only accessed if the input data contains POL2 Q or U time-series values, as created by SMURF:CALCQU. No IP correction is made if a null (!) value is supplied. If a non-null value is supplied, it should be an NDF that holds the total intensity (in pW) within the area of sky covered by the output map. The supplied NDF need not be pre-aligned with the output map - the WCS information in the NDF will be used to aligned them. For each Q or U value in the input time-streams, the corresponding total intensity (I) value is found by sampling the supplied IPREF map at the sky position of the Q/U value. This I value is multipled by a factor that depends on elevation and focal plane position, to get the IP correction. These Q and U corrections are rotated so that they use the same reference direction as the input Q/U data, corrected for extinction, and are then subtracted from the input Q or U value before going on to make a map from the corrected values. [!]

ITERMAPS = LITERAL (Read)
Specifies the name of a file in which to place a copy of the current map at the end of each iteration. If a null (!) value is supplied, they are placed in the MORE.SMURF.ITERMAPS component of the main output NDF (see parameter OUT). See configuration parameter " Itermap" . [!]

JSATILES = _LOGICAL (Read)
If TRUE, the output map is created on the JSA all-sky pixel grid, and is split up into individual JSA tiles. Thus multiple output NDFs may be created, one for each JSA tile that touches the map. Each of these output NDFs will have the tile index number appended to the end of the path specified by parameter " OUT" . If " JSATILES" is TRUE, the " REF" parameter is ignored. [FALSE]

JSATILELIST() = _INTEGER (Write)
If parameter " JSATILES" is set TRUE, the zero-based indicies of the created JSA tiles will be written to this output parameter. The number of such indices is given the " NTILE" parameter

LBND(2) = _INTEGER (Read)
An array of values giving the lower pixel index bound on each spatial axis of the output NDF. The suggested default values encompass all the input spatial information. The supplied values may be modified if TRIM is set TRUE. []

LBOUND(2) = _INTEGER (Write)
The lower pixel bounds of the output NDF. Note, values will be written to this output parameter even if a null value is supplied for parameter OUT.

MASK2 = NDF (Read)
An existing NDF that can be used to specify a second external mask for use with either the AST, FLT or COM model. See configuration parameters AST.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK. Note, it is assumed that this image is aligned in pixel coordinate with the output map. [!]

MASK3 = NDF (Read)
An existing NDF that can be used to specify a third external mask for use with either the AST, FLT or COM model. See configuration parameters AST.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK. Note, it is assumed that this image is aligned in pixel coordinate with the output map. [!]

MAXMEM = _INTEGER (Read)
Maximum memory available for map-making in MiB (mebibytes). For machines with more than 20 GB or memory, the default is to leave 4 GB free for other processes. For machines with less than than 20 GB or memory, the default is to leave 20% of the total memory free for other processes. []

METHOD = LITERAL (Read)
Specify which map-maker should be used to construct the map. The parameter can take the following values:

- " REBIN" – Use a single pass rebinning algorithm. This technique assumes that the data have previously had atmosphere and instrument signatures removed. It makes use of the standard AST library rebinning algorithms (see also KAPPA:WCSMOSAIC). It is an excellent choice for obtaining an image quickly, especially of a bright source.

- " ITERATE" – Use the iterative map maker. This map maker is much slower than the REBIN algorithm because it continually makes a map, constructs models for different data components (common-mode, spikes etc.). See CONFIG for parameters controlling the iterative map maker. [ITERATE]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NBOLOEFF = _DOUBLE (Write)
Effective number of bolometers in the output map when METHOD=iterate. [!]

NCONTCHUNK = _INTEGER (Write)
Total number of continuous data chunks processed by makemap when METHOD=iterate. [!]

NMCNVG = _INTEGER (Write)
Total number of continuous data chunks processed by makemap when METHOD=iterate that failed to converge. [!]

NMINSMP = _INTEGER (Write)
Total number of continuous data chunks processed by makemap when METHOD=iterate that failed due to insufficient samples. [!]

NOMFCF = _DOUBLE (Write)
The nominal beam FCF for the output map (Jy/beam/pW). This will depend on the observation dates of the input time-seroes files and the filter (450 or 850). If the input data spans a critical date at which the FCF changed, the stored value will be weighted by the amount of data included in the map that was taken before and after the critical date.

NTILE = _INTEGER (Write)
The number of output tiles used to hold the entire output array (see parameters JSATILES and TILEDIMS). If no input data fall within a specified tile, then no output NDF will be created for the tile, but (if JSATILES is FALSE) the tile will still be included in the tile numbering.

OUT = NDF (Write)
Output file.

OUTFILES = LITERAL (Write)
The name of a text file to create, in which to put the names of all the output NDFs created by this application via parameter OUT (one per line). If a null (!) value is supplied no file is created. [!]

PARAMS(2) = _DOUBLE (Read)
An optional array which consists of additional parameters required by the Sinc, SincSinc, SincCos, SincGauss, Somb, SombCos, and Gauss spreading methods (see parameter SPREAD).

PARAMS(1) is required by all the above schemes. It is used to specify how many pixels on either side of the output position (that is, the output position corresponding to the centre of the input pixel) are to receive contributions from the input pixel. Typically, a value of 2 is appropriate and the minimum allowed value is 1 (i.e. one pixel on each side). A value of zero or fewer indicates that a suitable number of pixels should be calculated automatically. [0]

PARAMS(2) is required only by the SombCos, Gauss, SincSinc, SincCos, and SincGauss schemes. For the SombCos, SincSinc, and SincCos schemes, it specifies the number of pixels at which the envelope of the function goes to zero. The minimum value is 1.0, and the run-time default value is 2.0. For the Gauss and SincGauss scheme, it specifies the full-width at half-maximum (FWHM) of the Gaussian envelope. The minimum value is 0.1, and the run-time default is 1.0. On astronomical images and spectra, good results are often obtained by approximately matching the FWHM of the envelope function, given by PARAMS(2), to the point-spread function of the input data.

PIXSIZE(2) = _REAL (Read)

Pixel dimensions in the output image, in arcsec. If only one value is supplied, the same value will be used for both axes. The default depends on the wavelength of the input data.

POINTING = LITERAL (Read)

The name of a text file containing corrections to the pointing read from the input data files. If null (!) is supplied, no corrections are used. If a file is supplied, it should start with one or more lines containing " #" in column one. These are comment lines, but if any comment line has the form " # SYSTEM=AZEL" or " # SYSTEM=TRACKING" then it determines the system in which the pointing correction are specified (SYSTEM defaults to AZEL). The last comment line should be a space-separated list of column names, including " TAI" , " DLON" and " DLAT" . Each remaining line should contain numerical values for each column, separated by white space. The TAI column should contain the TAI time given as an MJD. The DLON and DLAT columns should give arc-distance offsets parallel to the longitude and latitude axes, in arc-seconds. The TAI values should be monotonic increasing with row number. The longitude and latitude axes are either AZEL or TRACKING as determined by the SYSTEM value in the header comments. Blank lines are ignored. The DLON and DLAT values are added onto the SMU jiggle positions stored in the JCMTSTATE extension of the input NDFs. DLON and DLAT values for non-tabulated times are determined by interpolation.

If you need to apply two sets of pointing corrections, one in TRACKING and one in AZEL, you can include two tables (one for each system) in a single text file. Both tables should use the format described above. The two tables must be separated by a line containing two or more minus signs with no leading spaces. [!]

RATE_LIMITED = _LOGICAL (Write)

Set TRUE on exit if the iterative loop was terminated because the mean normalised change in the map does not seem to be falling (see config parameter " MAPTOL_RATE").

REF = NDF (Read)

An existing NDF that is to be used to define the output grid, or the string " JSA" . If an NDF is supplied, the output grid will be aligned with the supplied reference NDF. The reference can be either 2D or 3D and the spatial frame will be extracted. If " JSA" is supplied, the JSA all-sky pixel grid will be used (note, the map will still be created as a single NDF - if multiple NDFs, one for each JSA tile, are required, the " JSATILES" parameter should beset TRUE instead of using the " REF" parameter). If a null (!) value is supplied then the output grid is determined by parameters REFLON, REFLAT, etc. In addition, this NDF can be used to mask the AST, FLT or

COM model. See configuration parameters AST.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK. [!]

REFLAT = LITERAL (Read)
The formatted celestial latitude value at the tangent point of the spatial projection in the output cube. This should be provided in the coordinate system specified by parameter SYSTEM.

REFLON = LITERAL (Read)
The formatted celestial longitude value at the tangent point of the spatial projection in the output cube. This should be provided in the system specified by parameter SYSTEM.

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

SPREAD = LITERAL (Read)
The method to use when spreading each input pixel value out between a group of neighbouring output pixels if using METHOD=REBIN (for METHOD=ITERATE nearest-neighbour resampling is always used). If SPARSE is set TRUE, then SPREAD is not accessed and a value of " Nearest" is always assumed. SPREAD can take the following values:

- " Linear" – The input pixel value is divided bi-linearly between the four nearest output pixels. Produces smoother output NDFs than the nearest-neighbour scheme.
- " Nearest" – The input pixel value is assigned completely to the single nearest output pixel. This scheme is much faster than any of the others.
- " Sinc" – Uses the sinc(pi∗x) kernel, where x is the pixel offset from the interpolation point (resampling) or transformed input pixel centre (rebinning), and sinc(z)=sin(z)/z. Use of this scheme is not recommended.
- " SincSinc" – Uses the sinc(pi∗x)sinc(k∗pi∗x) kernel. A valuable general-purpose scheme, intermediate in its visual effect on NDFs between the bi-linear and nearest-neighbour schemes.
- " SincCos" – Uses the sinc(pi∗x)cos(k∗pi∗x) kernel. Gives similar results to the " Sincsinc" scheme.
- " SincGauss" – Uses the sinc(pi∗x)exp(-k∗x∗x) kernel. Good results can be obtained by matching the FWHM of the envelope function to the point-spread function of the input data (see parameter PARAMS).
- " Somb" – Uses the somb(pi∗x) kernel, where x is the pixel offset from the transformed input pixel centre, and somb(z)=2∗J1(z)/z (J1 is the first-order Bessel function of the first kind). This scheme is similar to the " Sinc" scheme.
- " SombCos" – Uses the somb(pi∗x)cos(k∗pi∗x) kernel. This scheme is similar to the " SincCos" scheme.

- " Gauss" – Uses the exp(-k∗x∗x) kernel. The FWHM of the Gaussian is given by parameter PARAMS(2), and the point at which to truncate the Gaussian to zero is given by parameter PARAMS(1).

For further details of these schemes, see the descriptions of routine AST_REBINx in SUN/211. [" Nearest"]

SYSTEM = LITERAL (Read)

The celestial coordinate system for the output cube. One of ICRS, GAPPT, FK5, FK4, FK4-NO-E, AZEL, GALACTIC, ECLIPTIC. It can also be given the value " TRACKING" , in which case the system used will be which ever system was used as the tracking system during the observation. The supplied value is ignored if a value is supplied for parameter " REF" .

The choice of system also determines if the telescope is considered to be tracking a moving object such as a planet or asteroid. If the system is GAPPT or AZEL, then each time slice in the input data will be shifted in order to put the base telescope position (given by TCS_AZ_BC1/2 in the JCMTSTATE extension of the input NDF) at the same pixel position that it had for the first time slice. For any other system, no such shifts are applied, even if the base telescope position is changing through the observation. [TRACKING]

TILEBORDER = _INTEGER (Read)

Only accessed if a non-null value is supplied for parameter TILEDIMS. It gives the width, in pixels, of a border to add to each output tile. These borders contain data from the adjacent tile. This results in an overlap between adjacent tiles equal to twice the supplied border width. If the default value of zero is accepted, then output tiles will abut each other in pixel space without any overlap. If a non-zero value is supplied, then each pair of adjacent tiles will overlap by twice the given number of pixels. Pixels within the overlap border will be given a quality name of " BORDER" (see KAPPA:SHOWQUAL). [0]

TILEDIMS(2) = _INTEGER (Read)

This parameter is ignored if parameter " JSATILES" is set TRUE.

For large data sets, it may sometimes be beneficial to break the output array up into a number of smaller rectangular tiles, each created separately and stored in a separate output NDF. This can be accomplished by supplying non-null values for the TILEDIMS parameter. If supplied, these values give the nominal spatial size of each output tile, in pixels. Edge tiles may be thinner if the TRIMTILES parameter is set TRUE. In order to avoid creating very thin tiles around the edges, the actual tile size used for the edge tiles may be up to 10 % larger than the supplied value. This creation of " fat" edge tiles may be prevented by supplying a negative value for the tile size, in which case edge tiles will never be wider than the supplied absolute value.

If only one value is supplied, the supplied value is duplicated to create square tiles. Tiles are created in a raster fashion, from bottom left to top right of the spatial extent. The NDF file name specified by " out" is modified for each tile by appending " _<N>" to the end of it, where <N> is the integer tile index (starting at 1). The number of tiles used to cover the entire output array is written to output parameter NTILES. The tiles all share the same projection and so can be simply pasted together in pixel coordinates to reconstruct the full size output array. The tiles are centred so that the reference position (given by REFLON and REFLAT) falls at the centre of a tile. If a tile receives no input data, then no corresponding output NDF is created, but

the tile is still included in the tile numbering scheme. If a null (!) value is supplied for TILEDIMS, then the entire output array is created as a single tile and stored in a single output NDF with the name given by parameter OUT (without any " _<N>" appendix). [!]

TRIM = _LOGICAL (Read)
If TRUE, then the output image is trimmed to remove any border of bad pixels. [FALSE]

TRIMTILES = _LOGICAL (Read)
Only accessed if the output is being split up into more than one spatial tile (see parameter TILEDIMS and JSATILES). If TRUE, then the tiles around the border will be trimmed to exclude areas that fall outside the bounds of the full sized output array. This will result in the border tiles being smaller than the central tiles. [FALSE]

UBND(2) = _INTEGER (Read)
An array of values giving the upper pixel index bound on each spatial axis of the output NDF. The suggested default values encompass all the input spatial information. The supplied values may be modified if TRIM is set TRUE. []

UBOUND(2) = _INTEGER (Write)
The upper pixel bounds of the output NDF. Note, values will be written to this output parameter even if a null value is supplied for parameter OUT.

WVMLOG = FILENAME (Write)
Name of a text file into which to write raw WVM data with columns for each stage of processing. Multiple blocks of data may be written to the file if the WVM data are processed in chunks. New data are appended to the file if it already exists. If a null (!) value is supplied, no file is written. [!]

Notes:

- If multiple masks are specified for a single model component, then the source areas of the individual masks are combined together to form the total mask. For instance, if values are supplied for both AST.ZERO_MASK and AST.ZERO_LOWHITS, then a pixel in the total mask will be considered to be a " source" pixel if it is a source pixel in either the external mask specified by AST.ZERO_MASK, or in the " low hits" mask.
- The iterative algorithm can be terminated prematurely by pressing control-C at any time. If this is done, the current iteration will complete and the user will then be asked how to continue. Options include: 1) abort immediately without an output map, 2) close retaining the current unfinalised output map, and 3) perform one more iteration to finalise the map and then close. Note, if control-C is pressed a second time, the application will abort immediately, potentially leaving files in an unclean state.
- A FITS extension is added to the output NDF containing any keywords that are common to all input NDFs. To be included in the output FITS extension, a FITS keyword must be present in the NDF extension of every input NDF, and it must have the same value in all input NDFs. In addition, certain headers that relate to start and end events are propogated from the oldest and newest files respectively.

- The output NDF will contain an extension named " SMURF" containing an NDF named " EXP_TIME" , which contains the exposure time associated with each pixel.
- The FITS keyword EXP_TIME is added to the output FITS extension. This header contains the median value of the EXP_TIME array (stored in the SMURF extension of the output NDF).If this value cannot be calculated for any reason, the corresponding FITS keyword is assigned a blank value.
- If parameter TILEDIMS is assigned a value, FITS keywords NUMTILES and TILENUM are added to the output FITS header. These are the number of tiles used to hold the output data, and the index of the NDF containing the header, in the range 1 to NUMTILES, but if JSATILES is TRUE then FITS keyword TILENUM is also added but is instead used for the JSA tile number in the range 0 to 12 $*$ Nside ^ 2 - 1.
- The model configuration parameters can be sub-instrument dependent. For example, 850.flt.edgelow will copy the edgelow value into the flt section only for 850 micron data. Similarly for 450.flt.edgelow.
- Default values can be read from the $SMURF_DIR/smurf_makemap.def file.

Related Applications :

SMURF: QLMAKEMAP

NANTEN2ACSIS
Convert NANTEN2 FITS format data files to an ACSIS format NDF

Description:

Opens NANTEN2 SDFITS-style files for reading, and writes out the spectra in ACSIS format. Metadata are converted to appropriate FITS headers. The NANTEN2 spectra must have been calibrated.

Parameters:

DIRECTORY = _CHAR (Read)
Directory for output ACSIS files. A NULL value will use the current working directory. This command will create a subdir in this directory named after the observation number.

IN = GROUP (Read)
Name of the input NANTEN2 FITS files to be converted.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

Notes:

- Whilst this command does a reasonable job of converting common data to ACSIS format it still has to undergo extensive testing to ensure that it is always doing the correct thing.
- The ORAC-DR recipe defaults to REDUCE_SCIENCE.
- This routine works with NANTEN data exported from CLASS as a single SDFITS-style file per observation with the data in a binary table.

Related Applications :

SMURF: MAKECUBE, GSD2ACSIS, SUPERCAM2ACSIS;

POL2CHECK
Check if specified NDFs probably hold POL-2 data

Description:

This application checks each supplied file to see if it looks like it probably holds POL-2 data in a recognised form. If it does, it is categorised as either:

- raw analysed intensity time-series data
- Q, U or I time-series data created by CALCQU
- Q, U or I maps created by MAKEMAP.

If requested, output text files are created each holding a list of the paths for the NDFs in each category.

The checks are based on NDF meta-data and FITS headers. It is possible that an NDF could pass these checks and yet fail to open in other smurf task if any of the additional meta-data required by those tasks has been corrupted or is otherwise inappropriate.

An error is reported if POL2 data from more than one waveband (450 or 850) is present in the list of supplied data files.

By default, an error is also reported if POL2 data for more than one object is present in the list of supplied data files (this check can be disabled by setting parameter MULTIOBJECT to TRUE). The object is given by FITS header " OBJECT" .

Parameters:

IN = NDF (Read)
Input NDFs(s).

JUNKFILE = LITERAL (Read)
The name of a text file to create containing the paths to the input NDFs that do not hold POL-2 data in any recognised form. Only accessed if one or more such NDFs are found within the group of NDFs specified by parameter IN. Supplying null (!) results in no file being created. [!]

JUNKFOUND = _LOGICAL (Write)
Returned TRUE if one or more of the input NDFs is not a recognised POL-2 file.

MAPFILE = LITERAL (Read)
The name of a text file to create containing the paths to the input NDFs that hold 2-dimensional maps of Q, U or I from POL-2 data. Only accessed if one or more such NDFs are found within the group of NDFs specified by parameter IN. Supplying null (!) results in no file being created. [!]

MAPFOUND = _LOGICAL (Write)
Returned TRUE if one or more of the input NDFs holds 2-dimensonal maps of Q, U or I from POL-2 data.

MAPINFO = LITERAL (Read)
The name of a text file to create containing a line of information for each input file listed in the MAPFILE file (in the same order). Each line contains two space-sparated items: the first is a single letter Q, U or I indicating the Stokes parameter, and the second is an identifier of the form " <UT>_<OBS>_<SUBSCAN>" , where <UT> is the 8 digit UT date, <OBS> is the 5 digit observation number and <SUBSCAN> is the four digit number for the first subscan in the chunk (usually " 0003" except for observations made up of more than one discontiguous chunks). No file is created if null (!) is supplied. [!]

MISSING = LITERAL (Read)
The name of a text file to create identifying any missing raw data sub-scans. No file is created if no sub-scans are missing or if no raw data is supplied. The largest expected sub-scan number for all sub-arrays is the largest sub-scan number for which any raw data was found for any sub-array. The text file will contain a line for each sub-array that has any missing sub-scans. Each line will start with the sub-array name and be followed by a space spearated list of sub-scan identifiers. For instance, " S8A: _0012 _0034" .

MULTIOBJECT = _LOGICAL (Read)
Indicates if it is acceptable for the list of input files to include data for multiple objects. If FALSE, an error is reported if data for more than one object is specified by parameter IN. Otherwise, no error is reported if multiple objects are found. [FALSE]

RAWFILE = LITERAL (Read)
The name of a text file to create containing the paths to the input NDFs that hold raw analysed intensity POL-2 time-series data. Only accessed if one or more such NDFs are found within the group of NDFs specified by parameter IN. Supplying null (!) results in no file being created. [!]

RAWFOUND = _LOGICAL (Write)
Returned TRUE if one or more of the input NDFs holds raw analysed intensity POL-2 time-series data.

RAWINFO = LITERAL (Read)
The name of a text file to create containing a line of information for each input file listed in the RAWFILE file (in the same order). Each line contains a key for the raw data file of the form " " <UT>_<OBS>" , where <UT> is the 8 digit UT date, and <OBS> is the 5 digit observation number. No file is created if null (!) is supplied. [!]

STOKESFILE = LITERAL (Read)
The name of a text file to create containing the paths to the input NDFs that hold Q, U or I POL-2 time-series data. Only accessed if one or more such NDFs are found within the group of NDFs specified by parameter IN. Supplying null (!) results in no file being created. [!]

STOKESFOUND = _LOGICAL (Write)
Returned TRUE if one or more of the input NDFs holds Q, U or I POL-2 time-series data.

STOKESINFO = LITERAL (Read)
The name of a text file to create containing a line of information for each input file listed in the STOKESFILE file (in the same order). Each line contains two space-sparated items: the first is a single letter Q, U or I indicating the Stokes parameter, and the second is an identifier of the form " <UT>_<OBS>_<SUBSCAN>" ,

where <UT> is the 8 digit UT date, <OBS> is the 5 digit observation number and <SUBSCAN> is the four digit number for the first subscan in the chunk (usually "0003" except for observations made up of more than one discontiguous chunks). No file is created if null (!) is supplied. [!]

STOKES = LITERAL (Read)
The name of a text file to create containing the identifiers

Notes:

- This application was written originally for use within the pol2scan.py script, as a means of speeding up operations that are very slow when imlemented via multiple calls to KAPPA commands such as "fitsval", etc.

POL2IPCOR
Create an IP model from a set of POL2 observations of a bright extended source

Description:

This routine determines a correction for Instrumental Polarisation (IP) from a set of I, Q and U maps for a single field. At least nine POL2 observations of the same field at the same wavelength must be supplied, covering as wide a range in elevation as possible and all using the same pixel grid. The maps must NOT have been produced using smurf:skyloop. The correction may then be applied to the supplied Q and U maps to create a set of IP-corrected output maps. The parameters of the IP correction are reported.

IP correction may already have been applied to the supplied Q and U map (for instance, within makemap or pol2map), in which case this routine will in effect determine a secondary IP correction that aims to remove any residual elevation-dependence left over by the earlier primary IP correction.

The IP model measured and applied by this routine can take one of two forms (selected using the MODELTYPE parameter). Firstly, the " single component" form:

p = A + B∗el + C∗el∗el Qn_fp = p∗cos(-2∗(el - D)) Un_fp = p∗sin(-2∗(el - D))

where " el" is elevation (in radians), " p" is the fractional polarisation due to IP, (Qn_fp,Un_fp) are normalised Q and U corrections with respect to the focal plane Y axis, and (A,B,C,D) are the parameters of the model determined from the supplied input maps as described below.

Secondly, the " double component" form:

Qn_fp = A + B∗cos(-2∗(el - C)) + D∗cos(-2∗(el - E)) Un_fp = F + B∗sin(-2∗(el - C)) + D∗sin(-2∗(el - E))

This form has two extra free parameters (E and F) compared to the single component model.

For both model forms, the corrected output Q and U values are then given by:

Q_out = Q_in - Qn_tr∗I_in U_out = U_in - Un_tr∗I_in

where (I_in,Q_in,U_in) are the input I, Q and U values with respect to tracking north (e.g. Declination) and (Qn_tr,Un_tr) are the normalised Q and U corrections with respect to tracking north. These are determined by rotating the (Qn_fp,Un_fp) vector as follows:

Qn_tr = cos(2∗alpha)∗Qn_fp - sin(2∗alpha)∗Un_fp Un_tr = sin(2∗alpha)∗Qn_fp + cos(2∗alpha)∗Un_fp

where alpha is the angle from tracking north to the focal plane Y axis, in sense of North through East, at the central epoch of the observation.

The (Q,U) values in the input maps are given with respect to tracking north. So if these maps had already been correct for IP with a pefect IP model, then in the absence of noise the (Q,U) of an astronomical source should be constant for all observations regardless of elevation (assuming the source is not variable). If the IP correction is not perfect, the (Q,U)

measured in each observation will be offset away from the 'true' values by offsets that vary with elevation and azimuth. Thus if the (Q,U) values at a single point on the sky are plotted as a scatter plot, any imperfection in the IP model will be revealed by the points for different observations being distributed along an arc of some centro-symetric shape centred on the true (Q,U), with azimuth varying with distance along the arc. The above argument relies on the input (Q,U) values using tracking north as the reference direction. For instance, if they were instead to use the focal plane Y axis as the reference direction, then the true astronomical (Q,U) would not be constant (due to sky rotation) but would itself form some arc of a circle centred on the origin of the (Q,U) plane.

In practice, a separate circle is fitted to the input data at every point on the sky that is within both the AST and PCA masks. At a single point, each observation defines one position in the (Q,U) plane, and the best fitting circle passing through the (Q,U) positions for all observations is found. The centre of the circle defines the best estimate of the true astronomical (Q,U), and the offsets from the centre to each observation's (Q,U) position is a measure of the IP. Note, all reference to (Q,U) above refer to normalised (Q,U). The IP defined by each measured (Q,U) position is then rotated to use focal plane Y axis as the polarimetric reference direction instead of tracking north.

The above process, taken over all pixels in the AST and PCA masks, will typically produce many estimates of (Qn_fp,Un_fp) over a range of elevations. The parameters of the IP model (A,B,C,D) are then found by doing a weighted least squares fit to these (Qn_fp,Un_fp) values. The weighting function gives greater weight to pixels for which the residuals of the actual (Q,U) values from the fitted circle are smaller. It also favours circles in which the azimuth of each (Q,U) position is more closely correlated with its position around the circle.

Parameters:

IN = NDF (Read)

A group of input NDFs, each holding a 2-D map of I, Q or U for a single observation. All maps must hold data for the same object at the same wavelength (850 or 450), and must be aligned in pixel coordinates. A complete trio of I, Q and U maps must be supplied for each observation. The maps must NOT have been created by smurf:skyloop. This parameter is only used if a null (!) value is supplied for parameter MAPDIR. If null (!) is also supplied for IN, then the fit is based solely on any data supplied via parameter INLOGS. [!]

INLOGS = LITERAL (Read)

A group of existing log files created by previous runs of this application. If supplied, the data in these log files is included in the IP model fit. This allows the fit to be based on data from multiple objects. Note, this data is NOT included in the output log file specified by parameter LOGFILE. [!]

LOGFILE = LITERAL (Write)

The name of an output text file to create holding a table of the values that were generated from the data supplied by parameter IN or MAPDIR. These values, together with any specified by parameter INLOGS, is used in the fit to determine the values of model parameters (A,B,C,D). The values are stored in the form of a TOPCAT " ascii" table. So if LOGFILE is set to " table.asc" , you can view the table using the command " topcat -f ascii table.asc" . [!]

MAPDIR = LITERAL (Write)
The path to a directory holding existing observation maps created by the POL2MAP command (i.e. via the MAPDIR parameter). If supplied, the maps in this directory are used as the input maps and parameter IN is ignored. The naming convenions of the POL2MAP command are assumed (i.e. auto-masked maps in " *_imap.sdf" , externally masked maps in " *_Imap.sdf" , *_Qmap.sdf" and " *_Umap.sdf"). The externally masked maps are used as the input maps (see parameter IN). If a null (!) value is supplied for MAPDIR, all input maps are instead obtained using parameter IN.

MODELFILE = LITERAL (Write)
The name of an output text file to create holding a table of values evaluated from the fitted IP model. The file uses the TOPCAT " ascii" format, and holds the fitted model parameters in the header. [!]

MODELTYPE = LITERAL (Write)
Selects the mathematical form of the IP model. Can be " SINGLE" (for the single component model) or " DOUBLE" (for the double component model). [" SINGLE"]

OUT = NDF (Write)
An optional group of output NDFs. If null (!) is supplied, no IP corrected output maps are created. If a non-null value is supplied, the number of NDFs supplied must be equal to the number of input NDFs supplied for parameter " IN" . Each output NDF corresponding to an input Q or U map receives an IP-corrected copy of the corresponding input map. Output NDFs corresponding to input I maps are ignored (i.e. no total intensity output NDFs are created). Thus, if the value " *_C" is supplied for parameter OUT, output Q and U maps will be created with names of the form " <in>_C.sdf" , where <in> is the name of the corresponding input Q or U map, but no output I maps will be formed (the output maps will be placed in the same directory as the input maps). [!]

Notes:

- Single observation maps produced by skyloop can sometimes show IP that seems to vary with total intensity. In such cases, using a modified single component model in which:

p = A + B*el + C*el*el + E*total_intensity q = p*cos(-2(el - D)) u = p*sin(-2(el - D))

results in corrected maps that show lower variation with elevation.

QUCOVAR
Find co-variance of Q and U from a POL2 observation

Description:

Calculates the co-variance of Q and U in each map pixel for a POL2 observation. For a single pixel, the returned co-variance is:

sum(W_i∗RQ_i∗RU_i)/(N∗sum(W_i))

where:

- RQ_I is the ith residual from the Q map, and RU_I is the ith residual from the U map. A residual is the bolometer Q or U sample minus the Q or U map pixel value (the pixel value is the weighted mean of the Q or U sample values that fall in the pixel). These residuals can be dumped by makemap when the Q and U maps are created.
- W_i is the weight for the ith sample. It is the geometric mean of the weights associated with RQ_i and RU_i. These can also be dumped by makemap, but are created initially by calcuq as the residual of the fitting process for each sample.

The input NDFs should be created by running makemap twice on the CALCQU output for a single POL2 observation - once to create a Q map and once to create a U map. On each invocation of makemap, the config should include " exportndf=(res,qua,noi,lut)" . This will causes two extra NDFs to be created by each invocation of makemap with suffixes " QT_con_lut" , " QT_con_res" , " UT_con_lut" and " UT_con_res" . These should be supplied as input to this command.

Parameters:

QLUT = NDF (Read)
3-D NDF holding Q LUT model (suffix " QT_con_lut").

QRES = NDF (Read)
3-D NDF holding Q residuals time-series (suffix " QT_con_res").

ULUT = NDF (Read)
3-D NDF holding U LUT model (suffix " UT_con_lut").

URES = NDF (Read)
3-D NDF holding U residuals time-series (suffix " UT_con_res").

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
The output NDF - a 2-dimensional map holding the QU co-variance at each pixel.

REF = NDF (Read)
2-D NDF holding a Q or U map made from the same POL2 observation. This acts as a template to define the shape, size and WCS of the output NDF.

RAWPRESS
Compress raw data

Description:

Compress the raw time series data. Currently two compression schemes are available - see parameter " METHOD" . This task is intended to be a test bed of compression algorithms.

Parameters:

IN = NDF (Read)

Input files to be compressed.

METHOD = _CHAR (Read)

The compression scheme to use:

" OLD" - converts 32-bit integers to 16-bit integers by calculating a common mode signal at each time slice and a multiplicative value (BZERO and BSCALE) after removing the first measurement (STACKZERO).

" DELTA" - stores the differences between adjacent bolometer samples as 16-bit integers. Any values for which the differences are too big to be stored in 16 bits are stored explicitly in 32 bit integers (see SUN/11 for full details).

[DELTA]

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

Output file(s).

Notes:

- Data will be uncompressed automatically by any SMURF routine.
- Files may well be larger when compressed.

Related Applications :

SMURF: RAWUNPRESS

RAWRECREATEWCS
Fix broken raw ACSIS files by enabling the spectral WCS to be reconstructed

Description:

Some ACSIS faults result in a file being written with the frequency axis unable to be written because of an error in the FITS WCS stored in the FITS header. This command lets the spectral WCS be recreated once the FITS header has been fixed following a manual intervention. This command will fail on files that already have had the WCS information stripped from the FITS header.

Parameters:

NDF = NDF (Read)

Input files to be fixed.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

Notes:

- Supports ACSIS raw data files

RAWREWRTSC2WCS
Fix broken WCS in raw SCUBA-2 file by re-writing it

Description:

Some SCUBA-2 files end up with corrupt WCS extensions. It is possible to recreate the WCS by looking at the JCMTSTATE and FITS header. The new WCS is written to the file, replacing the original.

Parameters:

NDF = NDF (Read)

Input files to be fixed.

Notes:

- Supports SCUBA-2 raw data files
- Currently it will be necessary to remove the corrupt .WCS data using KAPPA ERASE before running this command.

RAWUNPRESS
Uncompress raw data

Description:

Uncompress the raw time series data from 16-bit to 32-bit integers. Does not flatfield and so the data are still in integer format. If the data are not compressed they will be copied without change.

Parameters:

IN = NDF (Read)

Input files to be uncompressed.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

Output file.

Related Applications :

SMURF: FLATFIELD

RAWFIXMETA
Fix metadata associated with a raw data file

Description:

Report any issues associated the metadata of a particular file. In most cases SMURF applications will automatically apply these corrections but the command can be used to investigate issues prior to making a map.

Parameters:

IN = NDF (Read)
Input files to be checked.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

STEPTIME = _DOUBLE (Write)
Average steptime for the given file in seconds. Only written if a single file is given.

Notes:

- Supports ACSIS raw data files
- In the future this command may gain the ability to fix the data files.

REMSKY
Remove sky background from SCUBA-2 data

Description:

This command can be used to fit and remove the sky signal from a SCUBA-2 time series file.

Parameters:

BBM = NDF (Read)
Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

FIT = _CHAR (Read)
Type of fit to be carried out for the PLANE sky removal method. Choices are Mean, Slope (to fit in elevation only) or Plane. No default.

FLATMETH = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

GROUP = _LOGICAL (Read)
If true, group related files together for processing as a single data set, else process each file independently. [FALSE]

IN = NDF (Read)
Input file(s).

METHOD = _CHAR (Read)
Sky removal method, either POLY or PLANE.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output file(s)

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

Notes:

- SC2CLEAN can calculate the common-mode signal much more accurately than the naive algorithm implemented in this routine.
- The iterative map-maker will calculate the sky signal itself and this command should not be used if that variant of the map-maker is to be used.

Related Applications :

SMURF: EXTINCTION, MAKEMAP, SC2CLEAN; SURF: REMSKY

SC2CLEAN
Clean SCUBA-2 time-series data

Description:

This command is a stand-alone task for cleaning SCUBA-2 time-series data. Cleaning operations include:

- flag entire bolometer data streams as bad based on a threshold fraction of bad samples;
- removing large-scale detector drifts by fitting and removing low-order polynomial baselines;
- identifying and repairing DC steps;
- flagging spikes;
- replacing spikes and other gaps in the data with a constrained realization of noise; and
- applying other frequency-domain filters, such as a high-pass or correction of the DA system response.

All the above operations can be performed on the dark squid data. These take the same parameters used for cleaning the primary bolometer data but use the " cleandk" namespace. For example, " dcthresh" would become " cleandk.dcthresh" .

Parameters:

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

COM = NDF (Write)

If COMPREPROCESS is set in the configuration file, the common mode is calculated and removed from the bolometer data. The COM adam parameter can then be used to specify an NDF to store the common mode. See also GAI. [!]

CONFIG = GROUP (Read)

Specifies values for the cleaning parameters. If the string " def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The available parameters are identical to the cleaning parameters used by the iterative map-maker (method=ITER) and the available parameters are listed in the " Configuration Parameters" appendix of SUN/258. Default values will be used for any unspecified parameters. Assigning the value " <def>" (case insensitive) to a keyword has the effect of resetting it to its default value. Options available to the map-maker but not understood by SC2CLEAN will be ignored. Parameters not understood will trigger an error. Use the " cleandk." namespace for configuring cleaning parameters for the dark squids. [current value]

FLAT = _LOGICAL (Read)
If set ensure data are flatfielded. If not set do not scale the data in any way (but convert to DOUBLE). [TRUE]

FLATMETH = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

GAI = NDF (Write)
If COMPREPROCESS is set in the configuration file, the common mode is calculated and removed from the bolometer data. The GAI adam parameter can then be used to specify an NDF to store the gain/offset/correlation coefficients of the common-mode template for each bolometer. See also COM. [!]

IN = NDF (Read)
Input files to be cleaned

MAXLEN = _DOUBLE (Read)
Maximum length (in seconds) for concatenated file. The default is to use all data if possible (subject to available memory). [!]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output file(s).

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs

created by this application (one per line). If a null (!) value is supplied no file is created. [!]

PADEND = _INTEGER (Read)
Number of samples to pad at end. Default is no padding. [!]

PADSTART = _INTEGER (Read)
Number of samples to pad at start. Default is no padding. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

USEDARKS = _LOGICAL (Read)
Use darks to mask data. [TRUE]

Notes:

- The default values and allowed parameters can be found in $SMURF_DIR/smurf_sc2clean.def
- An iterative map-maker config file can be used.

Related Applications :

SMURF: MAKEMAP, SC2CONCAT, SC2FFT

SC2CONCAT
Concatenate files into a larger file

Description:

Given a list of input files this task concatenates them into larger files. The rules it follows are:

- data files are grouped by subarray;
- files are only concatenated if they are continuous in time;
- the longest a concatenated file may be is given by MAXLEN (in sec.);
- for each continuous chunk of data, shorter than MAXLEN, a file is generated on disk for each subarray. The file name is determined as the name of the first input file for the chunk, with a suffix " _con" . The can be modified using the parameter OUT.

Parameters:

FLAT = _LOGICAL (Read)
If set ensure data are flatfielded. If not set do not scale the data in any way (but convert to DOUBLE). [TRUE]

FLATMETH = _CHAR (Read)
Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)
The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)
Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)
If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

IN = NDF (Read)
Input file(s).

MAXLEN = _DOUBLE (Read)
Maximum length (in seconds) for concatenated file. The default is to use all data if possible (subject to available memory). [!]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output concatenated files. Only used if OUTBASE is null (!). Note, the correct number of output files must be specified for OUT. If this number is not known, use parameter OUTBASE instead.

OUTBASE = LITERAL (Write)
The base name for the output NDFs. Each output NDF has a name equal to " base_<n>" where <n> is an integer greater than or equal to 1. If a null (!) value is supplied, the output NDFs are instead specified by parameter OUT. [!]

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

PADEND = _INTEGER (Read)
Number of samples to pad at end. Default is no padding. [!]

PADSTART = _INTEGER (Read)
Number of samples to pad at start. Default is no padding. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

USEDARKS = _LOGICAL (Read)
Use darks to mask data. [TRUE]

Related Applications :

SMURF: SC2FFT, SC2CLEAN

SC2EXPANDMODEL
Expand a DIMM model component into a full time-series data cube

Description:

This command is a stand-alone task for converting a DIMM model component (which may be stored as a series of model parameters) into a full time-series representation (a data cube with time along the third axis).

Parameters:

IN = NDF (Read)
Input files to be uncompressed and flatfielded.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output file(s).

Related Applications :

SMURF: makemap

SC2FFT
Fourier Transform SCUBA-2 time-series data

Description:

This routine performs the forward or inverse FFT of SCUBA-2 time-series data. The FFT of the data are stored in a 4-dimensional array with dimensions frequency, xbolo, ybolo, component (where component is a dimension of length 2 holding the real and imaginary parts). The inverse flag is used to transform back to the time domain from the frequency domain. If the data are already in the requested domain, the ouput file is simply a copy of the input file.

Parameters:

AVPSPEC = _LOGICAL (Read)

Calculate average power spectral density over " good" bolometers. By default a 1/noise^2 weight is applued (see WEIGHTAVPSPEC). [FALSE]

AVPSPECTHRESH = _DOUBLE (Read)

N-sigma noise threshold to define " good" bolometers for AVPSPEC [5]

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

FLAT = _LOGICAL (Read)

If set ensure data are flatfielded. If not set do not scale the data in any way (but convert to DOUBLE). [TRUE]

FLATMETH = _CHAR (Read)

Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)

The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)

Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)

If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

IN = NDF (Read)

Input files to be transformed.

INVERSE = _LOGICAL (Read)
Perform inverse transform. [FALSE]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NGOOD = _INTEGER (Write)
Number of good bolometers which contribute to average power spectrum.

OUT = NDF (Write)
Output files. The number of output files can differ from the number of input files due to darks being filtered out and also to files from the same sequence being concatenated before aplying the FFT.

OUTFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

POLAR = _LOGICAL (Read)
Use polar representation (amplitude, argument) of FFT. [FALSE]

POWER = _LOGICAL (Read)
Use polar representation of FFT with squared amplitudes divided by the frequency bin spacing (gives a power spectral density, PSD). [FALSE]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

WEIGHTAVPSPEC = _LOGICAL (Read)
If set, weight power spectrum of each bolo by 1/noise^2 when calculating the average (estimated from 2–20 Hz spectrum). [TRUE]

ZEROBAD = _LOGICAL (Read)
Zero any bad values in the data before taking FFT. [TRUE]

Notes:

Transforming data loses the VARIANCE and QUALITY components.

Related Applications :

SMURF: SC2CONCAT, SC2CLEAN, CALCNOISE

SC2FILTERMAP
Filter a 2-d map

Description:

This routine takes the FFT of a 2D map, applies a Fourier-space filter, and then transforms back to real-space before writing out. Currently the only available filter is a whitening filter which is measured using a supplied reference image.

Parameters:

FILT_EDGEHIGH = _REAL (Write)
High-pass filter frequency (1/arcsec)

FILT_EDGELOW = _REAL (Write)
Low-pass filter frequency (1/arcsec)

IN = NDF (Read)
Input files to be transformed.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output transformed files.

OUTFILTER = NDF (Write)
Optional NDF for the filter.

WHITEN = _LOGICAL (Read)
If selected, measure azimuthally-averaged angular power spectrum in WHITEREFMAP, fit a model A/F^B + W, and apply its complement to the FFT of the data (normalized to W). AZAVSPEC is set implicitly. [FALSE]

WHITEREFMAP = NDF (Read)
Reference map in which to measure whitening filter. [FALSE]

ZEROBAD = _LOGICAL (Read)
Zero any bad values in the data before taking FFT. [TRUE]

Notes:

Transforming data loses the VARIANCE and QUALITY components.

Related Applications :

SMURF: SC2MAPFFT

SC2MAPFFT
Fourier Transform 2D maps

Description:

This routine performs the forward or inverse FFT of a 2D map. The FFT of the data are stored in a 3-dimensional array with dimensions xfrequency, yfrequency, component (where component is a dimension of length 2 holding the real and imaginary parts, or ampliude and phase if in polar form). The inverse flag is used to transform back to the spatial domain from the frequency domain. If the data are already in the requested domain, the ouput file is simply a copy of the input file.

Parameters:

AZAVPSPEC = _LOGICAL (Read)
If true, calculate the azimuthally-averaged angular power power spectrum. POLAR and POWER are set implicitly. [FALSE]

IN = NDF (Read)
Input files to be transformed.

INVERSE = _LOGICAL (Read)
Perform inverse transform. [FALSE]

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
Output transformed files.

POLAR = _LOGICAL (Update)
Use polar representation (amplitude, argument) of FFT. [FALSE]

POWER = _LOGICAL (Update)
Use polar representation of FFT with squared amplitudes divided by the frequency bin spacing (gives a power spectral density, PSD). [FALSE]

ZEROBAD = _LOGICAL (Read)
Zero any bad values in the data before taking FFT. [TRUE]

Notes:

Transforming data loses the VARIANCE and QUALITY components.

Related Applications :

SMURF: SC2FFT

SC2PCA
Use principal component analysis to identify correlated SCUBA-2 signals

Description:

This routine calculates a new set of N statistically independent basis vectors (i.e. with a diagonal covariance matrix) for the N bolometer time series, and calculates the projection of the bolometers along this new basis. This " Principal Component Analysis" is useful for identifying time-correlated noise signals. The ouput array of components contains the new basis vectors, normalized by their RMS, though ordered by decreasing significance. The output amplitudes data cube gives the amplitude of each component for each bolometer across the focal plane. Generally speaking the component time series illustrate the time-varying shape of the correlated signals, and the amplitudes show how strong they are, and which bolometers are affected.

Parameters:

BBM = NDF (Read)

Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

FLAT = _LOGICAL (Read)

If set ensure data are flatfielded. If not set do not scale the data in any way (but convert to DOUBLE). [TRUE]

FLATMETH = _CHAR (Read)

Method to use to calculate the flatfield solution. Options are POLYNOMIAL and TABLE. Polynomial fits a polynomial to the measured signal. Table uses an interpolation scheme between the measurements to determine the power. [POLYNOMIAL]

FLATORDER = _INTEGER (Read)

The order of polynomial to use when choosing POLYNOMIAL method. [1]

FLATSNR = _DOUBLE (Read)

Signal-to-noise ratio threshold to use when filtering the responsivity data to determine valid bolometers for the flatfield. [3.0]

FLATUSENEXT = _LOGICAL (Read)

If true the previous and following flatfield will be used to determine the overall flatfield to apply to a sequence. If false only the previous flatfield will be used. A null default will use both flatfields for data when we did not heater track at the end, and will use a single flatfield when we did heater track. The parameter value is not sticky and will revert to the default unless explicitly over-ridden. [!]

IN = NDF (Read)

Input files to be uncompressed and flatfielded. Any darks provided will be subtracted prior to flatfielding.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUTAMP = NDF (Write)
Amplitude data cube. The first two coordinates are bolometer location, and the third enumerates component.

OUTAMPFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output amplitude NDFs created by this application (one per line). If a NULL (!) value is supplied no file is created. [!]

OUTCOMP = NDF (Write)
Component vector data cubes (N components $*$ 1 $*$ M time slices). A cube is created so that it may be run through SC2FFT if desired.

OUTCOMPFILES = LITERAL (Write)
The name of text file to create, in which to put the names of all the output component NDFs created by this application (one per line). If a NULL (!) value is supplied no file is created. [!]

RESIST = GROUP (Read)
A group expression containing the resistor settings for each bolometer. Usually specified as a text file using " ^" syntax. An example can be found in $STARLINK_DIR/share/smurf/resist.cfg [$STARLINK_DIR/share/smurf/resist.cfg]

RESPMASK = _LOGICAL (Read)
If true, responsivity data will be used to mask bolometer data when calculating the flatfield. [TRUE]

Related Applications :

SMURF: SC2CLEAN, SC2FFT

SC2SIM
SCUBA-2 Simulator

Description:

This command attempts to simulate the data taken by a SCUBA-2 subarray when observing an astronomical image plus atmospheric background while driving the JCMT. The simulation includes photon and 1/f noise from the atmosphere, and nonlinear response which varies for different bolometers. It also includes SCUBA-2 field distortion. Sc2sim combines the functionality of a number of executables built in earlier versions of the simulator: staresim, dreamsim, pongsim. The output file from a 'heat' observation is named after the subarray, date, and filenumber, for example: s8a20060301_00001_0001.sdf. For each subarray used in a simulation, a corresponding 'heat' (flatfield) output file must be present in the working directory.

Sc2sim also performs the heatrun task when supplied an input file with ' heat' observation mode specified. This generates a heater flat-field measurement from simulated data for each of range of heater settings. The output file from a 'heat' observation is named after the subarray, date, and filenumber, for example: s8aheat20060301_00001.sdf.

Parameters:

BADBOL = _CHAR (Read)

ARD Description File for Bad Bolometer mask. [!]

MAXWRITE = INTEGER (Read)

Number of samples to write in output file.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OBSPAR = GROUP (Read)

Specifies values for the observation parameters used by the simulation.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated (or line-break separated) lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The parameter names and their default values are listed below. The default values will be used for any unspecified parameters. Unrecognized parameters are ignored (i.e. no error is reported).

OVERWRITE = LOGICAL (Read)
Flag to specify whether existing files are overwritten. Setting this to FALSE increments the 'group' counter in the output file names. Default is TRUE.

SEED = INTEGER (Read)
Seed for random number generator. If a seed is not specified, the clock time in milliseconds is used.

SIMPAR = GROUP (Read)
Specifies values for the simulation parameters. See the description for OBSPAR for the file format.

The parameter names and their default values are listed below. The default values will be used for any unspecified parameters. Unrecognized parameters are ignored (i.e. no error is reported).

SIMSTATS = LOGICAL (Read)
Flag to specify whether to report the properties of the current simulation given the parameters specified in SIMPAR and OBSPAR. The simulation is not carried out.

SIMTYPE = CHAR (Read)
Simulation type : In a ' full' simulation the flux for each bolometer at each time slice is calculated and stored in the output files, along with the pointing information. In a ' weights' simulation, the flux is set to zero for each bolometer, but the pointing information is written to the output files. These ' weights' files can be generated in less time than for a ' full' simulation and may be used to predict the rate of sampling across a mapped area for a given observation.

Observation Parameters

bol_distx (DOUBLE)
Average bolometer distance in the x-direction in arcseconds. [6.28]

bol_disty (DOUBLE)
Average bolometer distance in the y-direction in arcseconds [6.28].

bous_angle (DOUBLE)
For the BOUS obsmode, this parameter specifies the angle of the pattern relative to the telescope axes in radians anticlockwise. [0.0]

bous_height (DOUBLE)
For the BOUS obsmode, this parameter specifies the height of the boustrophedon pattern in arcseconds. This height is the height of the stack of back-and-forth sweeps across the sky, and is measured from the centre of the array. [2000.0]

bous_spacing (DOUBLE)
For the BOUS obsmode, this parameter specifies the spacing of the boustrophedon pattern in arcseconds. This spacing is the distance between the parallel horizontal sweeps across the sky, and is measured from the centre of the array. [240.0]

bous_vmax (DOUBLE)
For the BOUS obsmode, this parameter specifies the maximum telescope velocity in arcseconds/second. [200.0]

bous_width (DOUBLE)
For the BOUS obsmode, this parameter specifies the width of the boustrophedon pattern in arcseconds. This width is the width of each of back-and-forth sweeps across the sky, and is measured from the centre of the array. [2000.0]

colsize (INTEGER)
This is the number of bolometers in a " column" , and this is the total number of " rows" . [40]

conv_shape (INTEGER)
Flag for the possible convolution functions where

- 0 - Gaussian
- 1 - sinc(dx).sinc(dy)
- 2 - sinc(dx).sinc(dy) tapered
- 3 - sinc(dx).sinc(dy) after first 1.0
- 4 - bessel tapered

[1]

conv_sig (DOUBLE)
Convolution function parameter. [1.0]

coordframe (CHAR)
Map coordinate frame, ether NASMYTH, AZEL, or RADEC. [RADEC]

dec (CHAR)
Sexagesimal string representation of the Declination of the observation. [00:00:00.0]

distfac (DOUBLE)
Distortion factor, where 0 = no distortion. [0.0]

dut1 (DOUBLE)
Value of UT1 - UTC for current date in seconds. [0.0]

externobs (CHAR)
File name of external SCUBA-2 observation from which to generate scan pattern. [none]

flatname (CHAR)
Name of flatfield correction technique, either TABLE or POLYNOMIAL.

focstart (DOUBLE)
Starting focus position in mm. [-3.0]

focstep (DOUBLE)
Interval between focus positions in mm. [1.0]

grid_max_x (INTEGER)
DREAM reconstruction grid max x. [1]

grid_max_y (INTEGER)
DREAM reconstruction grid max y. [1]

grid_min_x (INTEGER)
DREAM reconstruction grid min x. [1]

grid_min_y (INTEGER)
DREAM reconstruction grid min y. [1]

grid_step_x (DOUBLE)
DREAM grid step in X direction in arcseconds. [6.28]

grid_step_y (DOUBLE)
DREAM grid step in Y direction in arcseconds. [6.28]

heatnum (INTEGER)
Number of heater settings. [1]

heatstart (DOUBLE)
Initial heater setting in pW. [0.0]

heatstep (DOUBLE)
Increment of heater setting in pW. [0.0]

height (DOUBLE)
Minimum height of pattern in arcseconds. [2000.0]

instap (CHAR)
Name of instrument aperture. []

instap_x (DOUBLE)
X focal plane offset in arcseconds. [0.0]

instap_y (DOUBLE)
Y focal plane offset in arcseconds. [0.0]

jig_step_x (DOUBLE) : 6.28 (arcseconds)
The DREAM step size in the x-direction between jiggle positions.

jig_step_y (DOUBLE) : 6.28 (arcseconds)
The DREAM step size in the y-direction between jiggle positions.

jig_pos.x (CHAR)
Array with relative DREAM vertex coordinates for the x-direction, in units of pixel distance. This parameter value should be a comma-separated list of integer values surrounded by parentheses. The number of values must be the same as the number of values in the jig_pos.y array. [(0,-1,1,-1,0,1,-1,1)]

jig_pos.y (CHAR)
Array with relative vertex coordinates for the y-direction, in units of pixel distance. This parameter value should be a comma-separated list of integer values surrounded by parentheses. The number of values must be the same as the number of values in the jig_pos.x array. [(1,-1,0,1,-1,1,0,-1)]

lambda (DOUBLE)
Wavelength of observation in m. [0.85e-3]

liss_angle (DOUBLE)
For the LISS obsmode, this parameter specifies the angle of the pattern relative to the telescope axes in radians anticlockwise. [0.0]

liss_height (DOUBLE)
For the LISS obsmode, this parameter specifies the height of the Lissajous pattern in arcseconds. [2000.0]

liss_nmaps (INTEGER)
The number of times the Lissajous pattern should repeat. [1]

liss_spacing (DOUBLE)
For the LISS obsmode, this parameter specifies the spacing of the Lissajous pattern in arcseconds. This spacing is the distance between the parallel sweeps across the sky, and is measured from the centre of the array. [240.0]

liss_vmax (DOUBLE)
For the LISS obsmode, this parameter specifies the maximum telescope velocity in arcseconds/second. [200.0]

liss_width (DOUBLE)
For the LISS obsmode, this parameter specifies the width of the Lissajous pattern in arcseconds. [2000.0]

mjdaystart (DOUBLE)
Modified Julian date at start of observation. [53795.0]

mspat_x (DOUBLE)
Array of microstep X-offsets in the focal plane. Units are arcseconds. Multiple values can be supplied as comma-separated list of offsets surrounded by parentheses, e.g " (10,20)" . [0.0]

mspat_y (DOUBLE)
Array of microstep Y-offsets in the focal plane. Units are arcseconds. Multiple values can be supplied as comma-separated list of offsets surrounded by parentheses, e.g " (10,20)" . [0.0]

nfocstep (INTEGER)
Number of focus positions. [7]

nmaps (DOUBLE)
Number of times to repeat pattern. [1]

nmicstep (INTEGER)
Number of microsteps. [1]

numsamples (INTEGER)
For the STARE obsmode, this is the number of samples. [128]

nvert (INTEGER)
The number of vertices in the DREAM jiggle pattern. [8]

obsmode (CHAR)
The observation mode, which can be any of the following :

- STARE : Simulates a simple point-and-shoot observation mode in which the camera stares at at a specified area of sky for a period of time.
- DREAM : (Dutch REal-time Acquisition Mode) Simulates an observition in which the Secondary Mirror unit moves rapidly in a star-like pattern such that each bolometer observes multiple points on the sky.
- SINGLESCAN : Simulates moving the array in a straight path across the sky.
- BOUS : Simulates mapping a rectangular area of sky using a simple Boustrophedon or raster pattern.
- PONG : Simulates mapping a rectangular area of sky by filling the box with a orthogonally cross-linked pattern by " bouncing" off the sides of the box. There are two subcategories of PONG, either Straight pong (straight lines between vertices) or Curve pong (slightly wiggly lines between vertices - pattern is a Fourier-expanded Lissajous).
- LISS : Simulates mapping a rectangular area of sky by filling the box with a Lissajous pattern.
- EXTERN : Recreates the scanning pattern of a real SCUBA-2 observation.

[PONG]

obstype (CHAR)
Observation type (POINT, FOCUS or SCIENCE). [SCIENCE]

planet (CHAR)
Planet (MARS, URANUS, VENUS, JUPITER, MOON, SATURN or NEPTUNE). []

platenum (INTEGER)
The number of waveplate rotations. [1]

platerev (DOUBLE)
The waveplate rotation in revolutions/second. [2.0]

pong_angle (DOUBLE)
For the PONG obsmode, this parameter specifies the angle of the pattern relative to the telescope axes in radians anticlockwise. [0.0]

pong_height (DOUBLE)
For the PONG obsmode, this parameter specifies the height of the pong pattern in arcseconds. [2000.0]

pong_nmaps (INTEGER)
The number of times the Pong pattern should repeat. [1]

pong_spacing (DOUBLE)
For the PONG obsmode, this parameter specifies the spacing of the Pong pattern in arcseconds. This spacing is the distance between the parallel sweeps across the sky, and is measured from the centre of the array. [240.0]

pong_type (CHAR)
Specifies the type of pong simulation (STRAIGHT or CURVE). [STRAIGHT]

pong_vmax (DOUBLE)
For the PONG obsmode, this parameter specifies the maximum telescope velocity in arcseconds/second. [200.0]

pong_width (DOUBLE)
For the PONG obsmode, this parameter specifies the width of the Pong pattern in arcseconds. [2000.0]

ra (CHAR)
Sexagesimal string representation of the Right Ascension of the observation. [00:00:00.0]

rowsize (INTEGER)
This is the number of bolometers in a " row" , and this is the total number of " columns" . [32]

scan_angle (DOUBLE)
For the SINGLESCAN obsmode, this parameter specifies the angle of the pattern relative to the telescope axes in radians anticlockwise. [0.0]

scan_pathlength (DOUBLE)
For the SINGLESCAN obsmode, this parameter specifies the width of the scan path in arcseconds. [2000.0]

scan_vmax (DOUBLE)
For the SINGLESCAN obsmode, this parameter specifies the maximum telescope velocity in arcseconds/second.

smu_move (INTEGER)
Code for the SMU move algorithm. The possible codes are :

- 0 : Block wave.
- 1 : 2 term not damped.
- 2 : 3 term not damped.
- 3 : 4 term not damped.
- 4 : 2 term flat end.
- 5 : 3 term flat end.
- 6 : 4 term flat end.
- 7 : ScubaWave - After 1 ms 0.098. After 8 ms 0.913. After 9 ms 1.000. popepi points is equivalent with 64 ms.
- 8 : This is an experimental wave form, which may change often. Now it is a cosine waveform from 0 to 1 in the full time.

[8]

smu_offset (DOUBLE)
SMU phase shift. [0.0]

smu_samples (INTEGER)
Number of samples per jiggle vertex. [1]

spacing (DOUBLE)
Grid spacing in arcseconds. [240.0]

steptime (DOUBLE)
Sample interval time in seconds. [0.005]

subsysnr (INTEGER)
Subsystem number. [1]

targetpow (DOUBLE)
Target bolometer power input in pW. [25.0]

vmax (DOUBLE)
Telescope maximum velocity in arcseconds/second. [200.0]

width (DOUBLE)
Minimum width of pattern in arcseconds. [2000.0]

Simulation Parameters

add_atm (INTEGER)
Flag for adding atmospheric emission. [0]

add_fnoise (INTEGER)
Flag for adding 1/f noise. [0]

add_hnoise (INTEGER)
Flag for adding heater noise. [0]

add_pns (INTEGER)
Flag for adding photon noise. [0]

airmass (DOUBLE)
Airmass of simulated observation. [1.2]

anang (DOUBLE)
Polarisation angle of analyser in degrees. [0.0]

anpol (DOUBLE)
Polarisation of analyser in percent. [100.0]

antrans (DOUBLE)
Transmission of analyser in percent. [100.0]

aomega (DOUBLE)
Coupling factor (0.179 for 850 microns, 0.721 for 450 microns). [0.179]

astname (CHAR)
Name of the input file containing astronomical sky image.

astpol (DOUBLE)
Polarisation of source in percent. [10.0]

atend (DOUBLE)
Ambient temperature at end in degrees celsius. [5.0]

atmname (CHAR)
Name of the input file containing atmospheric sky image.

atmrefnu (DOUBLE)
ATM reference corner frequency in Hz. [0.5]

atmrefvel (DOUBLE)
ATM reference velocity in meters/second. [15.0]

atmxvel (DOUBLE)
Atmospheric background velocity in arcseconds/second in the X direction. [5000.0]

atmyvel (DOUBLE)
Atmospheric background velocity in arcseconds/second in the Y direction. [0.0]

atmzerox (DOUBLE)
Atmospheric background offset in arcseconds in the X direction. [5000.0]

atmzeroy (DOUBLE)
Atmospheric background offset in arcseconds in the Y direction. [50000.0]

atstart (DOUBLE)
Ambient temperature at start in degrees celsius. [5.0]

bandGHz (DOUBLE)
Bandwidth in GHz. [35.0]

blindang (DOUBLE)
Polarisation angle of JCMT windblind in degrees. [10.0]

blindpol (DOUBLE)
Polarisation of blind in percent. [1.0]

blindtrans (DOUBLE)
Transmission of JCMT windblind in percent. [93.0]

cassang (DOUBLE)
Polarisation angle of Cassegrain optics in degrees. [135.0]

casspol (DOUBLE)
Polarisation of Cassegrain optics in percent. [1.0]

casstrans (DOUBLE)
Transmission of Cassegrain optics in percent. [98.0]

flux2cur (INTEGER)
Flag to indicate conversion of power from flux to current. [1]

interp (CHAR)
Name of the interpolation scheme to use when sampling the sky image. See docs for astResample in SUN/210. [NEAREST]

meanatm (DOUBLE)
Mean expeected atmospheric signal in pW. [7.0]

jy2pw (DOUBLE)
Jy to pW conversion modulo atmospheric transmission. [2.3e-3]

nasang (DOUBLE)
Polarisation angle of Nasmyth optics in degrees. [90.0]

naspol (DOUBLE)
Polarisation of Nasmyth optics in percent. [1.0]

nastrans (DOUBLE)
Transmission of Nasmyth optics in percent. [98.0]

ncycle (INTEGER)
Number of cycles through the DREAM pattern. [1]

param1 (DOUBLE)
Name of the first parameter for the sky interpolation scheme specified by " interp" . See docs for astResample in SUN/210. [2.0]

param2 (DOUBLE)
Name of the second parameter for the sky interpolation scheme specified by " interp" . See docs for astResample in SUN/210. [2.0]

refload (DOUBLE)
Reference load in pW. [7.4]

refnoise (DOUBLE)
Reference NEP in pW/sqrt(Hertz). [6.5e-5]

smu_terr (DOUBLE)
SMU timing error in seconds. [0.0]

spike_alpha (DOUBLE)
Index of spike p-law distribution. [-1.5]

spike_p0 (DOUBLE)
Minimum spike power in Jy. [1.0]

spike_p1 (DOUBLE)
Peak spike power in Jy. [1000.0]

spike_t0 (DOUBLE)
Mean time between spikes in seconds. [20.0]

subname (CHAR)
Subarray names for the simulation. Any number of subarrays can be selected in any order. A single subarray can be named as a simple string (e.g. " s8a"), multiple subarrays can be given using commas and parentheses (e.g. " (s8a,s4a)" . [s8a]

tauzen (DOUBLE)
Optical depth at 225 GHz at the zenith. [0.052583]

telemission (DOUBLE)
Telescope background per pixel in pW. [4.0]

xpoint (DOUBLE)
X pointing offset on the sky in arcseconds. [20.0]

ypoint (DOUBLE)
Y pointing offset on the sky in arcseconds. [20.0]

Related Applications :

SMURF: SKYNOISE

SC2THREADTEST
Task for testing speeds of different threading schemes

Description:

This routine tests schemes for visiting large quantities of SCUBA-2 data using multiple threads. This is a developer tool.

Parameters:

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NCHUNKS = _INTEGER (Read)
Number of time chunks. [2]

NSUB = _INTEGER (Read)
Number of subarrays. [4]

NTHREAD = _INTEGER (Read)
Number of threads to use. [2]

TSTEPS = _INTEGER (Read)
Number of time samples in simulated data chunk. [6000]

SKYNOISE
Generate a simulated sky background with spatial noise

Description:

This generates a simulated sky background with spatial noise following a power law spectrum.

Parameters:

FILENAME = _CHAR (Write)

Name of the output file containing the sky noise image.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OBSPAR = GROUP (Read)

Specifies values for the observation parameters used for skynoise generation.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character " ^" , containing one or more comma-separated (or line-break separated) lists of strings. Each string is either a " keyword=value" setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with " #" are ignored). Within a text file, newlines can be used as delimiters, as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

The parameter names and their default values are listed below. The default values will be used for any unspecified parameters. Unrecognized parameters are ignored (i.e. no error is reported).

SEED = INTEGER (Read)

Seed for random number generator. If a seed is not specified, the clock time in milliseconds is used.

SIMPAR = GROUP (Read)

Specifies values for the simulation parameters. See the description for OBSFILE for the file format.

The parameter names and their default values are listed below. The default values will be used for any unspecified parameters. Unrecognized parameters are ignored (i.e. no error is reported).

Observation Parameters

lambda (DOUBLE)

Wavelength of observation in m. [0.85e-3]

Simulation Parameters

aomega (DOUBLE)
Coupling factor (0.179 for 850 microns, 0.721 for 450 microns). [0.179]

atmname (CHAR)
Name of the file containing the atmospheric sky image.

atmrefnu (DOUBLE)
Atmospheric reference corner frequency in Hz. [0.5]

atmrefvel (DOUBLE)
Atmospheric reference velocity in m/s. [15.0]

bandGHz (DOUBLE)
Bandwidth in GHz. [35.0]

tauzen (DOUBLE)
Optical depth at 225 GHz at the zenith. [0.052583]

Related Applications :
SMURF: SC2SIM

SMURFCOPY
Copy a 2d image out of a time series file

Description:

This task can be used to extract data from a file for a particular time slice. The world coordinates will be valid on this slice so the data can be used for display or image overlay (e.g. when using the KAPPA OUTLINE command to determine where this slice lies in relation to the reconstructed map).

KAPPA NDFCOPY will not add the specific astrometry information when used to extract a slice and so cannot be used when WCS is required.

Parameters:

FTSPORT = _CHAR (Read)

The FTS-2 port to use in calculating the WCS for the output NDF, or null if FTS-2 was not in the beam. If set, this parameter should be " tracking" or " image" . [!]

IN = NDF (Read)

Input file. Cannot be a DARK frame. If the input file is raw data it will be flatfielded before writing out. This allows a reasonable bad pixel mask to be applied.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

Output file. Extensions are not propagated.

POINTING = LITERAL (Read)

The name of a text file containing corrections to the pointing read from the input data files. If null (!) is supplied, no corrections are used. If a file is supplied, it should start with one or more lines containing " #" in column one. These are comment lines, but if any comment line has the form " # SYSTEM=AZEL" or " # SYSTEM=TRACKING" then it determines the system in which the pointing correction are specified (SYSTEM defaults to AZEL). The last comment line should be a space-separated list of column names, including " TAI" , " DLON" and " DLAT" . Each remaining line should contain numerical values for each column, separated by white space. The TAI column should contain the TAI time given as an MJD. The DLON and DLAT columns should give arc-distance offsets parallel to the longitude and latitude axes, in arc-seconds. The TAI values should be monotonic increasing with row number. The longitude and latitude axes are either AXEL or TRACKING as determined by the SYSTEM value in the header comments. Blank lines are ignored. The DLON and DLAT values are added onto the SMU jiggle positions stored in the JCMTSTATE extension of the input NDFs. DLON and DLAT values for non-tabulated times are determined by interpolation. [!]

SLICE = _INTEGER (Read)

Index of time axis (GRID coordinates). 0 can be used to specify the last slice in the file without having to know how many slices are in the file.

Notes:

- Currently, this routine cannot support multiple input files or multiple indices from a single input file. Once extracted the output file can no longer be processed by SMURF routines.
- Currently only understands SCUBA-2 data.
- SCUBA-2 data will be flatfielded in the output slice if the input file is raw.

Related Applications :

KAPPA: CONTOUR, OUTLINE; SMURF: jcmtstate2cat

SMURFHELP
Gives help about SMURF

Description:

Displays help about SMURF. The help information has classified and alphabetical lists of commands, general information about SMURF and related material; it describes individual commands in detail.

Here are some of the main options:

- smurfhelp No parameter is given so the introduction and the top-level help index is displayed.
- smurfhelp application/topic This gives help about the specified application or topic.
- smurfhelp application/topic subtopic This lists help about a subtopic of the specified application or topic. The hierarchy of topics has a maximum of four levels.
- smurfhelp Hints This gives hints for new and intermediate users.
- smurfhelp summary This shows a one-line summary of each application.
- smurfhelp classified classification This lists a one-line summary of each application in the given functionality classification.

See the Section " Navigating the Help Library" below for details how to move around the help information, and to select the topics you want to view.

Usage:

```
smurfhelp [topic] [subtopic] [subsubtopic] [subsubsubtopic]
```

Parameters:

SUBTOPIC = LITERAL (Read)
Subtopic for which help is to be given. [" "]

SUBSUBTOPIC = LITERAL (Read)
Subsubtopic for which help is to be given. [" "]

SUBSUBSUBTOPIC = LITERAL (Read)
Subsubsubtopic for which help is to be given. [" "]

TOPIC = LITERAL (Read)
Topic for which help is to be given. [" "]

Navigating The Help Library :

The help information is arranged hierarchically. You can move around the help information whenever SMURFHELP prompts. This occurs when it has either presented a screen's worth of text or has completed displaying the previously requested help. The information displayed by SMURFHELP on a particular topic includes a description of the topic and a list of subtopics that further describe the topic.

At a prompt you may enter:

- a topic and/or subtopic name(s) to display the help for that topic or subtopic, so for example, " block parameters box" gives help on BOX, which is a subtopic of Parameters, which in turn is a subtopic of BLOCK;
- a <CR> to see more text at a " Press RETURN to continue ..." request;
- a <CR>} at topic and subtopic prompts to move up one level in the hierarchy, and if you are at the top level it will terminate the help session;
- a CTRL/D (pressing the CTRL and D keys simultaneously) in response to any prompt will terminate the help session;
- a question mark " ?" to redisplay the text for the current topic, including the list of topic or subtopic names; or
- an ellipsis " ..." to display all the text below the current point in the hierarchy. For example, " BLOCK..." displays information on the BLOCK topic as well as information on all the subtopics under BLOCK.

You can abbreviate any topic or subtopic using the following rules.

- Just give the first few characters, e.g. " PARA" for Parameters.
- Some topics are composed of several words separated by underscores. Each word of the keyword may be abbreviated, e.g. " Colour_Set" can be shortened to " C_S" .
- The characters " %" and " $*$" act as wildcards, where the percent sign matches any single character, and asterisk matches any sequence of characters. Thus to display information on all available topics, type an asterisk in reply to a prompt.
- If a word contains, but does end with an asterisk wildcard, it must not be truncated.
- The entered string must not contain leading or embedded spaces.

Ambiguous abbreviations result in all matches being displayed.

Implementation Status:

- Uses the portable help system.

STACKFRAMES
Stack 2d processed frames into time series cube

Description:

Takes a stack of 2d frames of bolometer data, usually noise images or responsivity images, and combines them into a single cube with, if sort is enabled, an annotated time axis or an axis based on a numeric FITS header item. This makes it easy to look at the behaviour of a single detector as it varies with time. Not all observations include time information or should be sorted at all and for those set SORT to false. The 3rd axis will not be a sorted axis in that case. This can be useful for examining bolometer maps created by MAKEMAP.

Parameters:

IN = NDF (Read)

Input file(s). Files must all be 2-d and have the same dimensions. For the SORT option to be available they must have a DATE-OBS FITS header.

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

Single output file with all the 2d images stacked into a single observation.

SORT = _LOGICAL (Read)

Should the data be sorted into time order (true) or left in the order given in IN (false). If the first file in IN has no date information sorting will be disabled automatically. Default is true if date information is available.

SORTBY = _CHAR (Read)

If the data are sorted (SORT=TRUE) this parameter controls which header item should be used to do the sorting. Options are MJD (the date) or the name of a numeric FITS header. MJD is not allowed if no date items are present. [MJD]

Notes:

- No special SCUBA-2 processing is applied. The assumption is simply that you have some images that are all the same size and you want to put them into a single cube with a time axis.
- Variations in pixel origin are ignored. Make sure images are aligned and are the same size.
- Useful for looking at the variations in bolometer parameters such as images created by CALCNOISE or CALCFLAT.

Related Applications :

SMURF: CALCNOISE, CALCFLAT, MAKEMAP

STARECALC
Calculate image for SCUBA-2 STARE observations

Description:

This command is used for reconstructing 2-D images from STARE observations. Files containing data not taken in STARE mode are ignored. The user has the option to specify the number of frames to be averaged together, but the default is to automatically calculate that number to give 1-second averages.

Parameters:

BBM = NDF (Read)
Group of files to be used as bad bolometer masks. Each data file specified with the IN parameter will be masked. The corresponding previous mask for a subarray will be used. If there is no previous mask the closest following will be used. It is not an error for no mask to match. A NULL parameter indicates no mask files to be supplied. [!]

IN = NDF (Read)
Name of input data file(s).

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NAVER = _INTEGER (Read)
Number of frames to average together in output images. If a NULL value is given, NAVER is calculated dynamically for each input file to give output images which are 1-second averages. [!]

OUT = NDF (Write)
Name of output file containing STARE images.

OUTFILES = LITERAL (Write)
The name of a text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

Related Applications :

SMURF: DREAMSOLVE; KAPPA: WCSMOSAIC; CCDPACK: MAKEMOS

SUPERCAM2ACSIS
Convert a Supercam SDFITS format data file to an ACSIS format NDF

Description:

Opens Supercam SDFITS files for reading, and writes out the spectra in ACSIS format. Metadata are converted to appropriate FITS headers. The Supercam spectra must have been calibrated.

Parameters:

DIRECTORY = _CHAR (Read)
Directory for output ACSIS files. A NULL value will use the current working directory. This command will create a subdir in this directory named after the observation number.

IN = GROUP (Read)
Name of the input SDFITS files to be converted.

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

Notes:

- Whilst this command does a reasonable job of converting common data to ACSIS format it still has to undergo extensive testing to ensure that it is always doing the correct thing.
- The ORAC-DR recipe defaults to REDUCE_SCIENCE.
- SUPERCAM data are written as one SDFITS file for every on-the-fly spectrum. Each file has one spectrum from each of the 64 receptors. A full observation can therefore consist of many files.

Related Applications :

SMURF: MAKECUBE, GSD2ACSIS;

TIMESORT
Re-order the time slices in a raw ACSIS data cube into increasing time

Description:

This routine accepts as input one or more raw ACSIS data cubes, spanned by (frequency, detector number, time) axes. It sorts the time slices into monotonically increasing time value and writes the resulting data to one or more new output NDFs. The ACSIS and JCMTSTATE extensions and the WCS component are modified along with the main Data array, so that the resulting cube remains internally consistent.

The main reason for using this routine is to ensure that data have a defined transformation from WCS coordinates to pixel coordinates. It can also be used to reduce the size of data files by excluding dead detectors (see parameter DETPURGE). This command should be run before attempting to merge multi-subsystem data.

There are two main modes, selected by parameter MERGE. If MERGE is FALSE, then the time slices in each input NDF are sorted independently of the other NDFs, and each output NDF contains data only from the corresponding input NDF. If MERGE is TRUE, then the input NDFs are sorted into groups that contain NDFs from the same observation and sub-system (that is, all NDFs in a group have the same value for the OBSIDSS FITS keyword). For each group, the time slices in all NDFs in the group are sorted into a single list. This list is then divided up into chunks (in a manner selected by parameter SIZELIMIT), and the time slices are written out sequentially to a number of output NDFs. If any time slice is present in more than one input NDF, then the data values for the two or more input time slices are merged into a single time slice.

MERGE = TRUE should be used to sort the time slices contained in a set of sub-scans from a sub-system.

Parameters:

DETECTORS = LITERAL (Read)

A group of detector names to include in, or exclude from, the output cube. If the first name starts with a minus sign, then the specified detectors are excluded from the output cube (all other detectors are included). Otherwise, the specified detectors are included from the output cube (all other detectors are excluded). Information in the ACSIS extension that is associated with excluded detectors is also excluded from the output NDFs. If a null (!) value is supplied, data from all detectors will be used. See also DETPURGE. [!]

DETPURGE = _LOGICAL (Read)

If TRUE, then any detectors that have no good data values in the input NDFs will be excluded from the output NDFs. This is in addition to any excluded detectors specified by the DETECTORS parameter. Information in the ACSIS extension that is associated with such detectors is also excluded from the output NDFs. Note, DETPURGE takes precedence over DETECTORS. That is, if DETPURGE is set TRUE, then bad detectors will be excluded even if the DETECTORS parameter indicates that they should be included. [FALSE]

GENVAR = _LOGICAL (Read)
If TRUE, then the Variance component in each output file is filled with values determined from the Tsys values in the input files. These variances values replace any inherited from the Variance component of the input NDFs. [FALSE]

IN = NDF (Read)
A group of input NDFs, each holding raw time series data.

LIMITTYPE = LITERAL (Read)
Only accessed if parameter MERGE is set TRUE and a positive value is supplied for SIZELIMIT. Specifies the units of the SIZELIMIT value. It must be one of:

- " SPECTRA" : SIZELIMIT is the maximum number of spectra in each output NDF.
- " SLICES" : SIZELIMIT is the maximum number of time slices in each output NDF.
- " FILESIZE" : SIZELIMIT is the maximum number of megabytes of data in each output NDF. Here, the SI definition of megabytes is used in which 1 MB = 1,000,000 bytes.

Note, when using the FILESIZE option the specified file size only includes the size of the Data and Variance components in the NDF. Consequently, the actual file size may be a little larger than the requested size because of the extra information held in NDF extensions. [" FILESIZE"]

MERGE = _LOGICAL (Read)
If FALSE, then each input NDF is sorted independently of the other input NDFs, and the sorted data for each input NDF is written to a separate output NDF. If TRUE, then the input NDFs are divided up into groups relating to different observations. Each such group is then further sub-divided up into groups relating to sub-systems within the observation. Each group then holds sub-scans from a single observation and sub-system. All the time slices from every NDF in each such group are read into a single list, which is then sorted. The sorted data can be written out to a single large output file (one for each observation sub-system), or can be split up into several smaller output files, as specified by the SIZELIMIT parameter. The dynamic default for this parameter is TRUE if two or more of the input NDFs refer to the same observation and sub-system number, and FALSE otherwise. []

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

NOUT = _INTEGER (Write)
An output parameter in which is stored the total number of output NDFs created.

OUT = NDF (Write)
A group of output NDFs. If parameter MERGE is FALSE, then a separate output NDF is created for each input NDF and so the size of the supplied group should equal the number of input NDF. If parameter MERGE is TRUE, then the number of output NDFs is determined by the SIZELIMIT parameter. In this case, the number of values in the supplied group should equal the number of sub-systems represented in the input data. If a GRP modification element is used to specify the names, then the specified modifiation will be applied to a set of names containing the first input

NDF for each sub-system. If all the input file names conform to the usual naming convention of ACSIS raw time series files (" ayyyymmdd_nnnnn_nn_nnnn" with an optional arbitrary trailing suffix that must begin with an underscore) then each output NDF for a given sub-system will have an appropriately incremented value for the trailing " _nnnn" field. If any of the input NDFs do not conform to the ACSIS file naming convention, the strings " _1" , " _2" , etc will be appended to the end of the supplied group of names to form the output NDF names.

OUTFILES = LITERAL (Write)

The name of text file to create, in which to put the names of all the output NDFs created by this application (one per line). If a null (!) value is supplied no file is created. [!]

SIZELIMIT = _INTEGER (Read)

Only accessed if parameter MERGE is set TRUE. It is a number that specifies the maximum size of each output NDF when merging data from several input NDFs (see parameter MERGE). The minimum number of output NDFs needed to hold all the input data will be used. The final output NDF may be smaller than the specified maximum size. The value given is either the file size in SI megabytes (1,000,000 bytes), the number of time slices, or the number of spectra, as specified by parameter LIMITTYPE. If a null (!) value is supplied, then the number of output NDFs will be the same as the number of input NDFs, and all output NDFs will have the same size. If a negative or zero value is supplied, then a single output NDF will be created holding all the input data. [!]

SPECBND = LITERAL (Read)

Indicates what to do if the input NDFs have differing pixel bounds on the spectral axis.

- " FIRST" : The spectral axis in each output NDF will have the same pixel bounds as the spectral axis in the first input NDF.
- " UNION" : The pixel bounds of the spectral axis in each output NDF will be the union of the pixel bounds of the spectral axis in all input NDFs.
- " INTERSECTION" : The pixel bounds of the spectral axis in each output NDF will be the intersection of the pixel bounds of the spectral axis in all input NDFs.

[" FIRST"]

Notes:

- This command runs on ACSIS raw data files. SCUBA-2 files are guaranteed to be in time order.

Related Applications :

SMURF: MAKECUBE

UNMAKECUBE
Produce simulated time series data from a regrided ACSIS data cube

Description:

This routine creates one or more time series cubes, spanned by (frequency, detector number, time) axes, from one or more input sky cubes spanned by (celestial longitude, celestial latitude, spectrum) axes. Thus, it performs a sort of inverse to the MAKECUBE application.

The output time series detector samples are created by interpolating the supplied input sky cubes at the position of the reference time series sample centre. Various interpolation methods can be used (see parameter INTERP).

The output time series cubes inherit all meta-data from the corresponding input reference time series. The only thing modified is the values in the NDF " Data" array.

Parameters:

DETECTORS = LITERAL (Read)

A group of detector names. Only data for the named detectors will be included in the output time series cubes. If a null (!) value is supplied, data for all detectors will be created. [!]

IN = NDF (Read)

A group of input (ra,dec,spectrum) sky cubes (for instance, a set of tiles produced by MAKECUBE). If these sky cubes have any spatial overlap, then the output time series data will be derived from the last supplied sky cube that covers the overlap region. That is, sky cubes near the end of the supplied group take precedence over those near the start.

INTERP = LITERAL (Read)

The method to use when resampling the input sky cube pixel values. For details of these schemes, see the descriptions of routines AST_RESAMPLEx in SUN/210. INTERP can take the following values:

- " Linear" – The output sample values are calculated by bi-linear interpolation among the four nearest pixels values in the input sky cube. Produces smoother output NDFs than the nearest-neighbour scheme, but is marginally slower.
- " Nearest" – The output sample values are assigned the value of the single nearest input pixel. A very fast method.
- " Sinc" – Uses the sinc(pi∗x) kernel, where x is the pixel offset from the interpolation point and sinc(z)=sin(z)/z. Use of this scheme is not recommended.
- " SincSinc" – Uses the sinc(pi∗x)sinc(k∗pi∗x) kernel. A valuable general-purpose scheme, intermediate in its visual effect on NDFs between the bi-linear and nearest-neighbour schemes.
- " SincCos" – Uses the sinc(pi∗x)cos(k∗pi∗x) kernel. Gives similar results to the " Sincsinc" scheme.

- " SincGauss" – Uses the sinc(pi∗x)exp(-k∗x∗x) kernel. Good results can be obtained by matching the FWHM of the envelope function to the point-spread function of the input data (see parameter PARAMS).
- " Somb" – Uses the somb(pi∗x) kernel, where x is the pixel offset from the interpolation point and somb(z)=2∗J1(z)/z (J1 is the first-order Bessel function of the first kind). This scheme is similar to the " Sinc" scheme.
- " SombCos" – Uses the somb(pi∗x)cos(k∗pi∗x) kernel. This scheme is similar to the " SincCos" scheme.

[current value]

MSG_FILTER = _CHAR (Read)

Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)

A group of output NDFs into which the simulated time series data will be written.

PARAMS(2) = _DOUBLE (Read)

An optional array which consists of additional parameters required by the Sinc, SincSinc, SincCos, SincGauss, Somb and SombCos interpolation schemes (see parameter INTERP).

PARAMS(1) is required by all the above schemes. It is used to specify how many pixels are to contribute to the interpolated result on either side of the interpolation point in each dimension. Typically, a value of 2 is appropriate and the minimum allowed value is 1 (i.e. one pixel on each side). A value of zero or fewer indicates that a suitable number of pixels should be calculated automatically. [0]

PARAMS(2) is required only by the SombCos, SincSinc, SincCos, and SincGauss schemes. For the SombCos, SincSinc, and SincCos schemes, it specifies the number of pixels at which the envelope of the function goes to zero. The minimum value is 1.0, and the run-time default value is 2.0. For the SincGauss scheme, it specifies the full-width at half-maximum (FWHM) of the Gaussian envelope. The minimum value is 0.1, and the run-time default is 1.0. Good results are often obtained by approximately matching the FWHM of the envelope function, given by PARAMS(2), to the point-spread function of the input data. []

REF = NDF (Read)

A group of existing time series data cubes. These act as templates for the new time series cubes created by this application, and specified via parameter OUT.

USEDETPOS = _LOGICAL (Read)

If a true value is supplied, then the detector positions are read from the detector position arrays in each template NDF. Otherwise, the detector positions are calculated on the basis of the FPLANEX/Y arrays. Both methods should (in the absence of bugs) result in identical cubes. [TRUE]

Related Applications :

SMURF: MAKECUBE

UNMAKEMAP
Produce simulated time series data from a SCUBA-2 map

Description:

This routine creates one or more simulated SCUBA-2 time series cubes, from a supplied 2D image of the sky. Thus, it performs a sort of inverse to the MAKEMAP application.

The output time series bolometer samples are created by interpolating the supplied input sky image at the position of the reference time series sample centre. Various interpolation methods can be used (see parameter INTERP). Gaussian noise and pointing errors may also be added (see parameters SIGMA and PERROR).

The output time series cubes inherit all meta-data from the corresponding input reference time series. The only thing modified is the values in the NDF " Data" array.

Parameters:

ALIGNSYS = _LOGICAL (Read)

If TRUE, then the spatial positions of the template time series data are aligned with the supplied sky map in the current co-ordinate system of the map, Otherwise, they are aligned in the ICRS co-ordinate system. For instance, if the current co-ordinate system in the sky map is AZEL, then setting ALIGNSYS to TRUE will result in the template data being aligned in AZEL directly, disregarding the fact that a given AZEL will correspond to different positions on the sky at different times. [FALSE]

AMP2 = _DOUBLE (Read)

Controls the amplitude of the 2 Hz signal in the analysed intensity streams created in polarimetry mode (see " QIN" and " UIN"). This parameter is only used if HARMONIC is set to its default value of 4 (a value of zero is assumed otherwise). It gives the amplitude of the 2 Hz signal as a fraction of the total intensity. See also " PHASE2" . [0.0]

AMP4 = _DOUBLE (Read)

Controls the amplitude of the 4 Hz signal in the analysed intensity streams created in polarimetry mode (see " QIN" and " UIN"). This parameter is only used if HARMONIC is set to its default value of 4 (a value of zero is assumed otherwise). It gives the amplitude of the 4 Hz signal as a fraction of the total intensity. See also " PHASE4" . [0.0]

AMP16 = _DOUBLE (Read)

Controls the amplitude of the 16 Hz signal in the analysed intensity streams created in polarimetry mode (see " QIN" and " UIN"). This parameter is only used if HARMONIC is set to its default value of 4 (a value of zero is assumed otherwise). It gives the amplitude of the 16 Hz signal as a fraction of the total intensity. See also " PHASE16" . [0.0]

ANGROT = _DOUBLE (Read)

The angle from the focal plane X axis to the POL2 fixed analyser, in degrees. Measured positive in the same sense as rotation from focal plane X to focal plane Y. [90.0]

COM = NDF (Read)
A group of existing time series NDFs that supply the common-mode signal to be added to the output time series data. The number of NDFs supplied should match the number of NDFs supplied for parameter REF. Each supplied NDF should be one-dimensional, with length at least equal to the length of the time axis of the corresponding REF cube. If null (!) is supplied, the common mode is set to a constant value given by parameter COMVAL. [!]

COMVAL(2) = _DOUBLE (Read)
One or two values in pW that determine the common mode signal for all time slices. Only accessed if parameter " COM" is set to null (!). If two values are supplied, the first is taken to be the emission from the sky and the second is taken to be an offset caused by the electronics. The total common-mode signal is the sum of the two. If only one value is supplied, it is assumed that the second value is zero (i.e. the entire common-mode is caused by sky signal). Note - when simulating POL-2 data, Instrumental Polarisation is based on just the sky emission. Supplying zero or null results in no common mode being included in the output time series data. [!]

GAI = NDF (Read)
A group of existing 2D NDFs that specify the gain of each bolometer for the corresponding IN file. If null (!) is supplied, all bolometer gains are set to unity. Otherwise, each of the supplied 2D NDFs must have dimensions of (32,40). The number of NDFs in the group must equal the number of NDFs supplied for IN. [!]

HARMONIC = _INTEGER (Read)
The Q and U values are derived from the fourth harmonic of the half-wave plate rotation. However, to allow investigation of other instrumental effects, it is possible instead to derive equivalent quantities from any specified harmonic. These quantities are calculated in exactly the same way as Q and U, but use the harmonic specified by this parameter. They are stored in the output NDFs given by OUT, in place of the normal fourth harmonic signal. [4]

IN = NDF (Read)
The input 2D image of the sky. If NDFs are supplied for the QIN and UIN parameters, then IN should hold I values. For POL2 data, the input I, Q and U pixel values are assumed to incorporate the effect of the 1.35 loss caused by placing POL2 in the beam.

INSTQ = NDF (Read)
An optional 2D input NDF holding the instrumental normalised Q value for each bolometer, with respect to fixed analyser Ths parameter is only used if parameter IPFORM is set to " USER" . The NDF should have dimensions of (32,40). The total intensity falling on each bolometer is multiplied by the corresponding value in this file, to get the instrumental Q value that is added onto the value read from the QIN parameter. Bad values are treated as zero values. Note, currently there is no facility to use different INSTQ values for different sub-arrays - all data supplied via IN will use the same INSTQ values regardless of sub-array. To overcome this restriction, run unmakemap separately for each sub-array supplying a different INSTQ each time.

INSTU = NDF (Read)
An optional 2D input NDF holding the instrumental normalised U value for each bolometer, with respect to fixed analyser Ths parameter is only used if parameter IPFORM is set to " USER" . The NDF should have dimensions of (32,40). The total intensity falling on each bolometer is multiplied by the corresponding value in this

file, to get the instrumental U value that is added onto the value read from the UIN parameter. Bad values are treated as zero values. Note, currently there is no facility to use different INSTU values for different sub-arrays - all data supplied via IN will use the same INSTU values regardless of sub-array. To overcome this restriction, run unmakemap separately for each sub-array supplying a different INSTU each time.

INTERP = LITERAL (Read)

The method to use when resampling the input sky image pixel values. For details of these schemes, see the descriptions of routines AST_RESAMPLEx in SUN/210. INTERP can take the following values:

- " Linear" – The output sample values are calculated by bi-linear interpolation among the four nearest pixels values in the input sky cube. Produces smoother output NDFs than the nearest-neighbour scheme, but is marginally slower.
- " Nearest" – The output sample values are assigned the value of the single nearest input pixel. A very fast method.
- " Sinc" – Uses the sinc(pi∗x) kernel, where x is the pixel offset from the interpolation point and sinc(z)=sin(z)/z. Use of this scheme is not recommended.
- " SincSinc" – Uses the sinc(pi∗x)sinc(k∗pi∗x) kernel. A valuable general-purpose scheme, intermediate in its visual effect on NDFs between the bi-linear and nearest-neighbour schemes.
- " SincCos" – Uses the sinc(pi∗x)cos(k∗pi∗x) kernel. Gives similar results to the " Sincsinc" scheme.
- " SincGauss" – Uses the sinc(pi∗x)exp(-k∗x∗x) kernel. Good results can be obtained by matching the FWHM of the envelope function to the point-spread function of the input data (see parameter PARAMS).
- " Somb" – Uses the somb(pi∗x) kernel, where x is the pixel offset from the interpolation point and somb(z)=2∗J1(z)/z (J1 is the first-order Bessel function of the first kind). This scheme is similar to the " Sinc" scheme.
- " SombCos" – Uses the somb(pi∗x)cos(k∗pi∗x) kernel. This scheme is similar to the " SincCos" scheme.

[current value]

IPFORM = LITERAL (Read)

Indicates the nature the instrumental polarisation (IP) to be added to the returned time stream data if the template is a POL2 observation. It can be any of the following (case-insensitive):

- " JK" : The Johnstone-Kennedy model based on analysis of skydip data.
- " PL1" : A simpler model based on analysis of planetary data.
- " PL2" : A simpler model based on analysis of planetary data.
- " PL3" : A simpler model based on analysis of planetary data.
- " USER" : IP is based on the values supplied for parameters INSTQ and INSTU.
- " NONE" : No IP is added.

Note, if the PL1 or PL2 model is used, suitable values also need to be supplied for parameter PLDATA (the default values for PLDATA are appropriate for PL3).

Supplying a null value (!) value is equivalent to " NONE" . [" PL3"]

JKDATA = LITERAL (Read)
The path to an HDS container file holding data defining the parameters of the Johnstone/Kennedy model of POL2 instrumental polarisation. This parameter is only used if parameter IPFORM is set to " JK" . [' $STARLINK_DIR/share/smurf/ipdata.sdf']

MSG_FILTER = _CHAR (Read)
Control the verbosity of the application. Values can be NONE (no messages), QUIET (minimal messages), NORMAL, VERBOSE, DEBUG or ALL. [NORMAL]

OUT = NDF (Write)
A group of output NDFs into which the simulated time series data will be written. These will hold _DOUBLE data values. For POL2 data, the values should be considered to incorporate the 1.35 loss caused by POL2 .

PARAMS(2) = _DOUBLE (Read)
An optional array which consists of additional parameters required by the Sinc, SincSinc, SincCos, SincGauss, Somb and SombCos interpolation schemes (see parameter INTERP).

PARAMS(1) is required by all the above schemes. It is used to specify how many pixels are to contribute to the interpolated result on either side of the interpolation point in each dimension. Typically, a value of 2 is appropriate and the minimum allowed value is 1 (i.e. one pixel on each side). A value of zero or fewer indicates that a suitable number of pixels should be calculated automatically. [0]

PARAMS(2) is required only by the SombCos, SincSinc, SincCos, and SincGauss schemes. For the SombCos, SincSinc, and SincCos schemes, it specifies the number of pixels at which the envelope of the function goes to zero. The minimum value is 1.0, and the run-time default value is 2.0. For the SincGauss scheme, it specifies the full-width at half-maximum (FWHM) of the Gaussian envelope. The minimum value is 0.1, and the run-time default is 1.0. Good results are often obtained by approximately matching the FWHM of the envelope function, given by PARAMS(2), to the point-spread function of the input data. []

PAOFF = _DOUBLE (Read)
The angle from the fixed analyser to the have-wave plate for a POL_ANG value of zero, in degrees. Measured positive in the same sense as rotation from focal plane X to focal plane Y. [18.65]

PASIGN = _LOGICAL (Read)
Indicates the sense of rotation of the spinning half-wave plate. If TRUE, it is assumed that a positive POL_ANG value corresponds to rotation from focal plane X to focal plane Y axis. If FALSE, it is assumed that a positive POL_ANG value corresponds to rotation from focal plane Y to focal plane X axis. [FALSE]

PHASE2 = _DOUBLE (Read)
The phase offset to apply to the 2 Hz signal specified via parameter AMP2, in degrees. [0.0]

PHASE4 = _DOUBLE (Read)
The phase offset to apply to the 4 Hz signal specified via parameter AMP4, in degrees. [0.0]

PHASE16 = _DOUBLE (Read)
The phase offset to apply to the 16 Hz signal specified via parameter AMP16, in

degrees. [0.0]

PLDATA() = DOUBLE (Read)

The numerical parameters of the PL1, PL2 or PL3 IP model for POL2 data. This parameter is only used if parameter IPFORM is set to " PL1" , " PL2" or " PL3" . This should be a vector of three (PL1) or four (PL2 and PL3) values, being the coefficients of a quadratic polynomial that gives the fractional polarisation produced by instrumental polarisation, as a function of elevation (in radians):

fractional IP = A + B∗elev + C∗elev∗elev

where the vector (A,B,C) are given by the first three elements of parameter PLDATA. The PL1 model assumes that the IP is parallel to the elevation axis at all elevations. The PL2 and PL3 require a fourth value to indicate the offset between the IP and the elevation axis.

The default values are appropriate for PL3. [2.624E-3,4.216E-2,-2.410E-2,-3.400E-2]

POINTING = LITERAL (Read)

The name of a text file containing corrections to the pointing read from the reference data files. If null (!) is supplied, no corrections are used. If a file is supplied, it should start with one or more lines containing " #" in column one. These are comment lines, but if any comment line has the form " # SYSTEM=AZEL" or " # SYSTEM=TRACKING" then it determines the system in which the pointing correction are specified (SYSTEM defaults to AZEL). The last comment line should be a space-separated list of column names, including " TAI" , " DLON" and " DLAT" . Each remaining line should contain numerical values for each column, separated by white space. The TAI column should contain the TAI time given as an MJD. The DLON and DLAT columns should give arc-distance offsets parallel to the longitude and latitude axes, in arc-seconds. The TAI values should be monotonic increasing with row number. The longitude and latitude axes are either AXEL or TRACKING as determined by the SYSTEM value in the header comments. Blank lines are ignored. The DLON and DLAT values are added onto the SMU jiggle positions stored in the JCMTSTATE extension of the reference NDFs. DLON and DLAT values for non-tabulated times are determined by interpolation. [!]

PERROR = _DOUBLE (Read)

The standard deviation of the pointing errors to include in the output data, in arc-seconds. [0.0]

QIN = NDF (Read)

The input 2D image of the sky Q values, with respect to the second pixel axis (i.e. the pixel Y axis). Positive polarisation angles are in the same sense as rotation from the pixel X axis to the pixel Y axis. If QIN and UIN are both supplied, then the time series specified by the REF parameter should contain flat-fielded POL2 data. These values are assumed to incorporate the effect of the 1.35 loss caused by placing POL2 in the beam. [!]

REF = NDF (Read)

A group of existing time series data cubes. These act as templates for the new time series cubes created by this application, and specified via parameter OUT. They may contain either raw (_INTEGER) or flat-fielded (_DOUBLE) data values.

SIGMA = _DOUBLE (Read)

The standard deviation of the Gaussian noise to add to the output data. [0.0]

UIN = NDF (Read)

The input 2D image of the sky U values, with respect to the second pixel axis (i.e. the pixel Y axis). Positive polarisation angles are in the same sense as rotation from the pixel X axis to the pixel Y axis. If QIN and UIN are both supplied, then the time series specified by the REF parameter should contain flat-fielded POL2 data. These values are assumed to incorporate the effect of the 1.35 loss caused by placing POL2 in the beam. [!]

USEAXIS = LITERAL (Read)

A set of 2 axes to be selected from the Current Frame in the sky map. Each axis can be specified either by giving its index within the Current Frame in the range 1 to the number of axes in the Frame, or by giving its symbol. This parameter is only accessed if the Current Frame in the supplied NDF has more than 2 axes. The dynamic default selects the axes with the same indices as the significant NDF axes.

Related Applications :

SMURF: MAKEMAP

D Python Scripts

These scripts use KAPPA and SMURF commands to do the number crunching, but use Python rather than shell script to glue these commands together. The parameter system used by these scripts has been designed to be similar to the ADAM parameter system used by most other Starlink commands. So for instance, values can be specified by keyword or position on the command line, and the user will be prompted for any required parameter values that were not specified on the command line. However, there are some significant differences, including:

- The used parameter values are *not* stored in a parameter file for later use. So for instance, the KAPPA parget command cannot be used to recover parameter values.
- Parameters that take logical parameters must always specify their value using an equal sign (*e.g.* `retain=yes` rather than `retain`, and `retain=no` rather than `noretain`).
- The documentation for the command may be displayed by running the command with the `--help` option on the command line.

CONFIGMELD
Compare two configs using a visual file comparison tool

Description:

This script uses a visual file comparison tool such as " meld" , to display two sets of configuration parameters, highlighting the differences between them. Each config may be supplied directly, as is done when running a SMURF command such as MAKEMAP, or can be read from the History component of an NDF that was created by a command that has a CONFIG parameter (such as SMURF:MAKEMAP).

Maps created using SKYLOOP contain configs for both SKYLOOP and MAKEMAP. By defaut, the MAKEMAP config will be displayed. To See the SKYLOOP config instead, run this script with " app=skyloop" .

Usage:

```
configmeld config1 config2 app waveband defaults tool
```

Parameters:

APP = LITERAL (Read)
The application for which configuration parameters should be displayed. Only used if CONFIG1 or CONFIG2 is an NDF [" MAKEMAP"]

CONFIG1 = LITERAL (Read)
The first configuration. This can be a normal config such as is supplied for the CONFIG parameter of a SMURF application, or an NDF created by the application specified by parameter APP.

CONFIG2 = LITERAL (Read)
The second configuration. This can be a normal config such as is supplied for the CONFIG parameter of a SMURF application, or an NDF created by the application specified by parameter APP. If a value is supplied for PARAM, then CONFIG2 defaults to null (!). []

WAVEBAND = LITERAL (Read)
This parameter is not used if either CONFIG1 or CONFIG2 is an NDF created by a SMURF command. It should be one of " 450" or " 850" . It specifies which value should be displayed for configuration parameters that have separate values for 450 and 850 um. If either CONFIG1 or CONFIG2 is an NDF, then the wavebands to use are determined from the headers in the NDFs.

DEFAULTS = _LOGICAL (Read)
If TRUE, then each supplied configuration (CONFIG1 and CONFIG2) is extended to include default values for any config parameters that it does not specify. These defaults are read from the application's defaults file (e.g. " $SMURF_DIR/smurf_makemap.def"). The dynamic default is TRUE if a defaults file can be found and FALSE otherwise. []

PARAM = LITERAL (Read)
If supplied, then the value used for the specified parameter is displayed on standard

output, and no visual comparison is displayed. Separate values are displayed for CONFIG1 and (if supplied) CONFIG2. [!]

TOOL = LITERAL (Read)

Gives the name of the file comparison tool to use. The named command should be available on the current PATH. It should take the names of two files to compare as command line arguments. If null (!) is supplied, the first available tool in the following list is used:

- meld: www.meldmerge.org
- opendiff: developer.apple.com
- diffmerge: www.sourcegear.com/diffmerge
- kdiff3: kdiff3.sourceforge.net
- tkdiff: sourceforge.net/projects/tkdiff
- diffuse: diffuse.sourceforge.net

[!]

FTS2GAIA
Display in-band FTS-2 spectrum in gaia

Description:

Display those frames of an FTS-2 spectrum which fall within the band-pass. For 850 um, this band is between 11.2 and 12.2 cm^-1 wave numbers, and for 450 um, it is between 22.1 and 23.3 cm^-1 wave numbers. This translates to the peak of the spectrum of the target, found near the middle of the sensor array plus and minus some number of frames corresponding to the desired portion of the band pass.

Usage:

```
fts2gaia in
```

Parameters:

IN = NDF (Read)

The NDF FTS-2 spectrum file to be displayed.

JSAJOIN
Create a single tangent-plane NDF from a set of JSA tiles

Description:

This script pastes together one or more JSA tiles and then resamples the resulting montage onto a tangent-plane projection with square pixels and celestial north upwards. It can also restrict the output NDF to a specified sub-section of this montage.

By default, the output NDF contains all the data from all the tiles specified by parameter TILES. Alternatively, an existing NDF can be specified for parameter REGION, in which case the output NDF will contain only the data that falls within the supplied NDF. Another way to specify the bounds of the output NDF is to give the centre and radius of a circle on the sky using parameters CENTRE1, CENTRE2, RADIUS and SYSTEM.

The pixel size of the output may be specified by parameter PIXSIZE, but defaults to the nominal pixel size of the JSA tiles.

Usage:

```
jsajoin tiles out [centre1] [centre2] [radius] [system] [region] [pixsize] [retain]
```

Parameters:

CENTRE1 = LITERAL (Read)

The formatted RA or Galactic longitude at the centre of a circle that defines the required extent of the output NDF. See also CENTRE2 and RADIUS. The coordinate system is specified by parameter SYSTEM. It is only accessed if a null (!) value is supplied for parameter REGION. If a null value is supplied for both CENTRE1 and REGION, the output NDF will encompass all the data specified by parameter TILES. [!]

CENTRE2 = LITERAL (Read)

The formatted Dec or Galactic latitude at the centre of a circle that defines the required extent of the output NDF. See also CENTRE1 and RADIUS. The coordinate system is specified by parameter SYSTEM. It is only accessed if a null (!) value is supplied for parameter REGION and a non-null value is supplied for parameter CENTRE1.

INSTRUMENT = LITERAL (Read)

Selects the tiling scheme to be used. The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . If the first input NDF contains JCMT data, the default value for this parameter is determined from the FITS headers in the input NDF. Otherwise, there is no default and an explicit value must be supplied. []

OUT = NDF (Read)

The output NDF.

PIXSIZE = _REAL (Read)

The pixel size to use for the output NDF, in arc-seconds. Not used if an NDF is supplied for parameter REGION. If a null (!) value is supplied, a default pixel size is used equal to the geometric mean of the pixel dimensions in the middle tile specified by parameter TILES. [!]

RADIUS = _DOUBLE (Read)

The radius of the circle that defines the required extent of the output NDF, in arcminutes. See also CENTRE1 and CENTRE2. It is only accessed if a null (!) value is supplied for parameter REGION and a non-null value is supplied for parameter CENTRE1.

REGION = LITERAL (Read)

Specifies the required extent of the output NDF. It can be either a text file holding an AST Region description, or an NDF. If it is an NDF, it also defines the WCS and pixel grid of the output NDF. If a null (!) value is supplied, the region is specified using parameter CENTRE1, CENTRE2 and RADIUS. [!]

RETAIN = _LOGICAL (Read)

Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

SYSTEM = LITERAL (Read)

The celestial coordinate system used by the CENTRE1 and CENTRE2 parameters. It can be either " ICRS" or " Galactic" . The output NDF inherits this same system as its current WCS Frame. [" ICRS"]

TILES = NDF (Read)

A group of NDFs each of which corresponds to a JSA tile. They should all relate to a single instrument.

JSASPLIT
Split a single NDF up into a set of JSA tiles

Description:

This script re-projects a supplied NDF onto the JSA all-sky pixel grid and then splits the resulting NDF into more JSA tiles.

Usage:

```
jsasplit in out [trim] [retain] [msg_filter] [ilevel] [glevel] [logfile]
```

Parameters:

GLEVEL = LITERAL (Read)

Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE" . [" ATASK"]

ILEVEL = LITERAL (Read)

Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information. In addition scatter plots showing how each Q and U image compares to the mean Q and U image are displayed at this ILEVEL.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"]

IN = NDF (Read)

The input NDF.

INSTRUMENT = LITERAL (Read)

Selects the tiling scheme to be used. The following instrument names are recognised (unambiguous abbreviations may be supplied): " SCUBA-2(450)" , " SCUBA-2(850)" , " ACSIS" , " DAS" . If the input NDF contains JCMT data, the default value for this parameter is determined from the FITS headers in the input NDF. Otherwise, there is no default and an explicit value must be supplied. []

LOGFILE = LITERAL (Read)

The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. []

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

OUT = LITERAL (Read)
The basename for the output NDFs. Each output NDF will have a name formed by appending the integer tile index to the supplied basename, preceeded by and underscore. A null (!) value causes the name of the input NDF to be used.

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

TRIM = _INTEGER (Read)
A zero or negative value results in each output NDF covering the full area of the corresponding JSAtile. A value of one results in each output NDF being cropped to the bounds of the supplied NDF. A value of two or more results in each output NDF being cropped to remove any blank borders. [2]

JSATILEMOC
Create an image MOC based on a JSA tile

Description:

This script takes a JSA tile and produces a MOC representation of the area covered by good pixels.

Usage:

```
jsatilemoc in out [maxorder]
```

Parameters:

IN = NDF (Read)

An NDF (or FITS) file containing a JSA tile. It must have a TILENUM header indicating the JSA tile number accompanied by a comment including the HEALPix Nside value. The JSA tile must not have been trimmed. It could either be created with the trimming options disabled, or be untrimmed using the UNTRIM_JSA_TILES PICARD recipe.

MAXORDER = _INTEGER (Read)

The maximum HEALPix order to be included in the MOC. [29]

OUT = FITS (Read)

The output MOC FITS file name.

SKYLOOP
Create a map using the " inside-out" algorithm

Description:

This script makes a map from specified raw time-series data using the algorithm described at http://pipelinesandarchives.blogspot.co.uk/2012/10/inside-out-map-making.html. It runs SMURF:MAKEMAP multiple times, performing a single iteration of the Dynamic Iterative Map-Maker algorithm on each invocation, including data from all chunks. Each map created by MAKEMAP is used as the initial sky estimate for the next invocation. MAKEMAP subtracts this initial sky estimate from the time-series data before starting the first (and only) iteration, and then adds the initial sky estimate back on at the end prior to creating the output map.

The script produces several intermediate files: a set of cleaned time-series files that may be 2 to 3 times the size of the entire set of raw data files included in the map, and a 2D map for every iteration. These files are placed in a newly created directory that is normally deleted before the script exits. The files can be retained for debugging purposes if required by running the script with " retain=yes" on the command line.

The temporary files are placed in a directory name " NDG_xxxxx" , located within the directory specified by environment variable STAR_TEMP. If STAR_TEMP is not defined, they are placed in the system' s temporary directory (e.g. " /tmp").

Usage:

```
skyloop in out niter pixsize config [itermap] [ref] [mask2] [mask3] [extra] [ipref]
[retain] [msg_filter] [ilevel] [glevel] [logfile] [restart]
```

Parameters:

CHUNKWGT = _LOGICAL (Read)

Controls the weight used for each chunk. If False, then the weights are determined by the value of the CHUNKWGT FITS header in each input time-stream data file (unit weight is used for any data file that has no CHUNKWGT FITS header). If True, the weight for each chunk is based on the normalised map change for each chunk calculated on the previous iteration - the weight for a chunk is the ratio of the mean of all the chunk map changes to the chunk' s own map change. Weights above 1.0 are limited to 1.0. [False]

CONFIG = LITERAL (Read)

The MAKEMAP configuration parameter values to use. Additions will be made as follows:

- First iteration:

  numiter=1
  noi.export=1
  exportNDF=(lut,ext,res,qua)
  noexportsetbad=1

exportclean=1
ast.zero_notlast = 0
flt.zero_notlast = 0
com.zero_notlast = 0
pca.zero_notlast = 0
itermap=0
shortmap=0
bolomap=0
flagmap=<undef>
sampcube=0
diag.append=0

- Subsequent iterations:

numiter=1
noi.import=1
exportNDF=(res,qua)
doclean=0
importsky=ref
importlut=1
ext.import=1
ast.zero_notlast = 0
flt.zero_notlast = 0
com.zero_notlast = 0
pca.zero_notlast = 0
ast.zero_mask0 = <undef>
flt.zero_mask0 = <undef>
com.zero_mask0 = <undef>
pca.zero_mask0 = <undef>
flt.notfirst = 0
pca.notfirst = 0
pln.notfirst = 0
smo.notfirst = 0
itermap=0
shortmap=0
bolomap=0
flagmap=<undef>
sampcube=0
diag.append=1
downsampscale=0
downsampfreq=0
fakemap=<undef>

- Last iteration:

numiter=1

noi.import=1
doclean=0
importsky=ref
importlut=1
ext.import=1
ast.zero_notlast = 1
flt.zero_notlast = 1
com.zero_notlast = 1
pca.zero_notlast = 1
flt.notfirst = 0
pca.notfirst = 0
pln.notfirst = 0
smo.notfirst = 0
itermap=0
shortmap=<value specified by parameter CONFIG>
bolomap=<value specified by parameter CONFIG>
flagmap=<undef>
sampcube=0
diag.append=1
downsampscale=0
downsampfreq=0
fakemap=<undef>

EXTRA = LITERAL (Read)
A string holding any extra command line options to be passed to MAKEMAP (all invocations). [!]

GLEVEL = LITERAL (Read)
Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. [" ATASK"]

ILEVEL = LITERAL (Read)
Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information.

[" PROGRESS"]

IN = NDF (Read)
The group of time series NDFs to include in the output map.

INITIALSKY = NDF (Read)
An NDF holding an initial guess at the final map. This should contain any a priori expectations of what the final map should look like. It is used to define the starting point for the iterative map-making algorithm, in place of the usual flat map full of zeros. The data units in the supplied NDF must be " pW" . If an NDF is supplied, it is also used to define the WCS and pixel bounds of the output map (any NDF supplied for parameter REF is then ignored). [!]

IPREF = NDF (Read)
An existing NDF that is to be used to define the correction to be made for instrumental polarisation (IP). It is only accessed if the input data contains POL2 Q or U time-series values, as created by SMURF:CALCQU. No IP correction is made if a null (!) value is supplied. If a non-null value is supplied, it should be an NDF that holds the total intensity (in pW) within the area of sky covered by the output map. The supplied NDF need not be pre-aligned with the output map - the WCS information in the NDF will be used to aligned them. For each Q or U value in the input time-streams, the corresponding total intensity (I) value is found by sampling the supplied IPREF map at the sky position of the Q/U value. This I value is multipled by a factor that depends on elevation and focal plane position, to get the IP correction. These Q and U corrections are rotated so that they use the same reference direction as the input Q/U data, corrected for extinction, and are then subtracted from the input Q or U value before going on to make a map from the corrected values. [!]

ITERMAP = NDF (Write)
A 3D NDF to create holding the maps from all iterations. [!]

LOGFILE = LITERAL (Read)
The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. []

NITER = _INTEGER (Read)
The number of iterations to perform. A positive value specifies a fixed number of iterations to perform. A negative value indicates that iterations should continue until the normalized change in the map between iterations is less than the value of the " maptol" parameter in the configuration supplied by parameter CONFIG (a maptol value of 0.05 is used if CONFIG does not specify maptol). If a value of zero is supplied for NITER, the value used will be read from the " numiter" parameter in the configuration. [0]

MASK2 = NDF (Read)
An existing NDF that can be used to specify a second external mask for use with either the AST, PCA, FLT or COM model. See configuration parameters AST.ZERO_MASK, PCA.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK. Note, it is assumed that this image is aligned in pixel coordinate with the output map. [!]

MASK3 = NDF (Read)
An existing NDF that can be used to specify a third external mask for use with either the AST, PCA, FLT or COM model. See configuration parameters AST.ZERO_MASK, PCA.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK. Note, it is assumed that this image is aligned in pixel coordinate with the output map. [!]

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

OBSDIR = LITERAL (Read)
The name of a directory in which to put maps made from the individual observations. These are generated on the final iteration. If null is supplied, individual observation maps will not be created. Each map is stored in a file with name <UT>_<OBS>.sdf. If a single observation is split into multiple chunks, the first chunk will use the above naming scheme but the second and subsequent chunks will have names of the form <UT>_<OBS>_<CHUNK>.sdf. [!]

OUT = NDF (Write)
The NDF holding the output map.

PIXSIZE = _REAL (Read)
Pixel dimensions in the output image, in arcsec. The same value will be used for both axes. The default depends on the wavelength of the input data. []

REF = NDF (Read)
An existing NDF that is to be used to define the output grid. If supplied, the output grid will be aligned with the supplied reference NDF. The reference can be either 2D or 3D and the spatial frame will be extracted. If a null (!) value is supplied then the output grid is determined by parameters REFLON, REFLAT, etc. In addition, this NDF can be used to mask the AST, PCA, FLT or COM model. See configuration parameters AST.ZERO_MASK, PCA.ZERO_MASK, FLT.ZERO_MASK and COM.ZERO_MASK.

This parameter is only accessed if a null value is supplied for INITIALSKY. Otherwise, the NDF supplied as the initial sky is used to define the output grid.

On the second and subsequent invocations of MAKEMAP, any supplied REF image is replaced by the map created by the previous invocation of MAKEMAP. [!]

RESTART = LITERAL (Read)
If a value is assigned to this parameter, it should be the path to a directory containing the intermediate files created by a previous run of SKYLOOP. If supplied, execution of skyloop will restart from the point where the previous run finished. This is useful for continuing runs that have been interupted accidentally. The path to the intermediate files can be found by examining the log file created by the previous run. [!]

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

POL2MAP
Create Q, U and I maps from a group of POL-2 " spin&scan" data files

Description:

This script creates maps (Q, U and I) and a vector catalogue from a set of POL-2 observation. New observations can be added into the map without the need to re-process previously processed observations. The output maps are all in units of pW.

Note, with the default configuration this script can take up to an hour to run for each observation on a typical SCUBA-2-capabale computer.

Masking of models within makemap (AST, etc) can be based either on the SNR of the map created as the end of each iteration, or on an external map, or on a fixed circle centred on the origin - see parameter MASK.

By default, the Q, U, I and PI catalogue values are in units of mJy/beam (see parameter Jy).

Usage:

```
pol2map in iout qout uout [cat] [config] [pixsize] [qudir] [mapdir] [mask] [masktype]
[ipcor] [ipref] [reuse] [ref] [north] [debias] [retain] [maskout1] [maskout2]
[msg_filter] [ilevel] [glevel] [logfile]
```

Parameters:

BINSIZE = _REAL (Read)

The bin size in the output vector catalogue, in arcsec. The value supplied for parameter PIXSIZE is used as the default for BINSIZE. An error is reported if BINSIZE is smaller than PIXSIZE. []

CAT = LITERAL (Read)

The output FITS vector catalogue. No catalogue is created if null (!) is supplied. The Q, U and PI values in this catalogue will be in units of pW or mJy/beam, as selected using parameter JY . The bin size is specified by parameter BINSIZE. An extra column named " AST" is added to this catalogue in addition to those created by the polpack:polvec command. The AST column that holds a non-zero integer for each row that corresponds to a point inside the AST mask (i.e. a source point), and zero for all other rows. [!]

CONFIG = LITERAL (Read)

Extra parameter values to include in the MAKEMAP configuration used to create both the I maps and the Q/U maps.

In general, it is important that the I, Q and U maps are all created using the same configuration so that they can be compared directly. However, if it is necessary to use a different configuration for I and Q/U maps, the differences may be specified using the ADAM parameters " ICONFIG" and " QUCONFIG" . The ADAM parameter " CONFIG" specifies the configuration parameters that are always used, whether an I map or a Q/U map is being created. In all cases the configuration parameters specified by " CONFIG" are applied first, followed by the configuration parameters

specified by " ICONFIG" (if creating an I map) or " QUCONFIG" (if creating a Q or U map). Thus values supplied in " ICONFIG" or " QUCONFIG" over-ride values for the same parameters specified in " CONFIG" .

The configurations specified by CONFIG, ICONFIG and QUCONFIG are applied on top of the following set of default parameters:

```
^$STARLINK_DIR/share/smurf/.dimmconfig_pol2.lis
numiter = -200
modelorder=(com,gai,pca,ext,flt,ast,noi)

maptol = 0.05
maptol_mask = <undef>
maptol_mean = 0
maptol_box = 60
maptol_hits = 1

ast.mapspike_freeze = 5
pca.pcathresh = -150
pca.zero_niter = 0.5
com.zero_niter = 0.5
flt.zero_niter = 0.5
com.freeze_flags = 30
```

Additional parameters are also set, depending on the value of parameter MASK. If MASK is set to " AUTO" , the following parameters are added to the above default config:

```
ast.skip = 10
ast.zero_snr = 3
ast.zero_snrlo = 2
ast.zero_freeze = 0.2

pca.pcathresh = -50
pca.zero_snr = 5
pca.zero_snrlo = 3
pca.zero_freeze = -1

com.zero_snr = 5
com.zero_snrlo = 3
com.zero_freeze = -1

flt.zero_snr = 5
flt.zero_snrlo = 3
flt.zero_freeze = -1
```

If MASK is set to " CIRCLE" , the following parameters are added to the above default config:

ast.zero_circle = 0.0083 (degrees, i.e. 30 arc-seconds)
pca.zero_circle = 0.0038
com.zero_circle = 0.0083
flt.zero_circle = 0.0083

The default value for pca.pcathresh indicated above will be changed if it is too high to allow convergence of the I maps within the number of iterations allowed by numiter (this change only occurs if parameter SKYLOOP is FALSE).

If MASK is set to the name of an NDF, this script creates fixed masks from the NDF, and the following parameters are added to the above default config:

ast.zero_mask = mask2
pca.zero_mask = mask3
com.zero_mask = mask3
flt.zero_mask = mask3

The above " mask2" mask consists of clumps of pixel with SNR greater than 3, extended down to an SNR level of 2. The " mask3" mask consists of clumps of pixel with SNR greater than 5, extended down to an SNR level of 3. However, the above SNR levels are raised if necessary to ensure that the source occupies no more than 20% of the pixels within the " ref" mask, and 10% of the pixels within the " mask3" mask.

The same configuration is used for all three Stokes parameters - I, Q and U with the exception that " com.noflag=1" is added to the configuration when creating maps for Q and U.

If a configuration is supplied using parameter CONFIG, values supplied for any of the above parameters will over-write the values specified above. In addition, the following mandatory values are always appended to the end of the used configuration:

flagslow = 0.01
downsampscale = 0
noi.usevar=1

If null (!) or " def" is supplied, the above set of default configuration parameters are used without change. [" def"]

CALCQUCONFIG = LITERAL (Read)

Extra parameter values to include in the CALCQU configuration used to create the I, Q and U time-streams. [!]

DEBIAS = LOGICAL (Given)

TRUE if a correction for statistical bias is to be made to percentage polarization and polarized intensity in the output vector catalogue specified by parameter CAT. The bias estimator to use is given by parameter DEBIASTYPE. [FALSE]

DEBIASTYPE = LOGICAL (Given)

Only used if DEBIAS is TRUE. It gives the type of bias estimator to use, using the nomeclature of Montier at al " Polarization measurements analysis II. Best estimators of polarization fraction and angle" (A&A, 2018):

- " AS" : The asymptotic estimator. See section 2.3 of Montier et al. This estimator

produces bad P and PI values if the squared PI value is less than the variance in PI.

- " MAS" : The modified asymptotic estimator. See section 2.5 of Montier et al. This estimator does not produces bad P and PI values, even if the squared PI value is less than the variance in PI. [" AS"]

FCF = _REAL (Read)
The FCF value that is used to convert I, Q and U values from pW to Jy/Beam. If a null (!) value is supplied a default value is used that depends on the waveband in use and the dates of the observations being processed. [!]

GLEVEL = LITERAL (Read)
Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" ATASK"]

ICONFIG = LITERAL (Read)
Extra parameter values to include in the MAKEMAP configuration used to create I maps. The values specified by " ICONFIG" are applied after those specified by " CONFIG" . [!]

ILEVEL = LITERAL (Read)
Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"]

IN = NDF (Read)
A group of input files. Each specified file must be one of the following types:

- a raw POL-2 data file. Any supplied raw POL-2 data files will be converted into time-series Q,U and I files using SMURF:CALCQU and placed in the directory specified by parameter QUDIR. These will then be converted into maps using SMURF:MAKEMAP, and placed in the directory specified by parameter MAPDIR.
- a time-series file holding Stokes Q, U or I values. Any supplied time-series files will be converted into individual maps (one for each file) using SMURF:MAKEMAP,

and placed in the directory specified by parameter MAPDIR. These maps are created only for the required Stokes parameters - as indicated by parameters IOUT, QOUT and UOUT.

- a two-dimensional map holding Stokes Q, U or I values. Any maps must be in units of pW. The final output I map is created by coadding any supplied I maps with the I maps created by this script. These coadded maps are created only for the required Stokes parameters - as indiciated by parameters IOUT, QOUT and UOUT. Note, an error is reported if the pixel size in any supplied map is not equal to the value of parameter PIXSIZE.

Any combination of the above types can be supplied. Note, if parameter REUSE is TRUE, then any required output files that already exist in the directory specified by parameter MAPDIR are re-used rather than being re-created from the corresponding input data.

INITSKYI = NDF (Read)

An NDF holding an initial guess at the final I map. This should contain any a priori expectations of what the final I map should look like. It is used to define the starting point for the iterative map-making algorithm, in place of the usual flat map full of zeros. The data units in the supplied NDF must be " pW" . See also parameter REF. Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE. [!]

INITSKYQ = NDF (Read)

An NDF holding an initial guess at the final Q map. This should contain any a priori expectations of what the final Q map should look like. It is used to define the starting point for the iterative map-making algorithm, in place of the usual flat map full of zeros. The data units in the supplied NDF must be " pW" . See also parameter REF. Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE. [!]

INITSKYU = NDF (Read)

An NDF holding an initial guess at the final U map. This should contain any a priori expectations of what the final U map should look like. It is used to define the starting point for the iterative map-making algorithm, in place of the usual flat map full of zeros. The data units in the supplied NDF must be " pW" . See also parameter REF. Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE. [!]

IOUT = NDF (Write)

The output NDF in which to return the total intensity (I) map including all supplied observations. This will be in units of pW. Supply null (!) if the I map is not to be retained on exit. In this case, the I map will only be created if it is needed to create the output vector catalogue (see parameter CAT) and will be deleted on exit.

IPCOR = _LOGICAL (Read)

If TRUE, then IP correction is used when creating Q and U maps, based on the values in the total intensity map specified by parameter IPREF. If FALSE, then no IP correction is performed. The default is TRUE if any Q or U output maps are being created, and FALSE otherwise. []

IPREF = NDF (Read)

The total intensity map to be used for IP correction. Only accessed if parameter

IPCOR is set TRUE. If null (!) is supplied for IPREF, the map supplied for parameter REF is used. The map must be in units of pW. If the same value is supplied for both IOUT and IPREF, the output I map will be used for IP correction. Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE. [!]

JY = _LOGICAL (Read)

If TRUE, the I, Q and U values in the output catalogue will be in units of mJy/beam. Otherwise they will be in units of pW. Note, the Q, U and I maps are always in units of pW. The same FCF value is used to convert all three Stokes parameters from pW to mJy/beam, derived from the value supplied for parameter FCF. [TRUE]

LOGFILE = LITERAL (Read)

The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. The script can change the logfile if necessary by assign the new log file path to the module variable " starutil.logfile" . Any old log file will be closed befopre the new one is opened. []

MAPDIR = LITERAL (Read)

The name of a directory in which to put the Q, U an I maps made from each individual observation supplied via " IN" , before coadding them. If null is supplied, the new maps are placed in the same temporary directory as all the other intermediate files and so will be deleted when the script exists (unless parameter RETAIN is set TRUE). Note, these maps are always in units of pW. Each one will contain FITS headers specifying the pointing corrections needed to align the map with the reference map. [!]

MAPVAR = _LOGICAL (Read)

Determines how the variance information in the final I, Q and U coadded maps (parameters IOUT, QOUT and UOUT) are derived.

If MAPVAR is FALSE, the variances in the coadded maps are calculated by propagating the variance information from the individual observation maps. These variances are determined by makemap and are based on the spread of bolometer I, Q or U values that fall in each pixel of the individual observation map.

If MAPVAR is TRUE, the variances in the coadded maps are determined from the spread of input values (i.e. the pixel values from the individual observation maps) that fall in each pixel of the coadd.

The two methods produce similar variance estimates in the background regions, but MAPDIR=TRUE usually creates much higher on-source errors than MAPDIR=FALSE. Only use MAPDIR=TRUE if you have enough input observations to make the variance between the individual observation maps statistically meaningful. [FALSE]

MASK = LITERAL (Read)

Specifies the type of masking to be used within makemap (the same type of masking is used to create all three maps - I, Q and U):

- " AUTO" : makemap uses automatically generated masks based on the SNR map at the end of each iteration. The SNR levels used are specified by the " xxx.ZERO_SNR" and " xxx.ZERO_SNRLO" configuration parameters (see parameter CONFIG).

- " CIRCLE" : makemap uses a fixed circular mask of radius 60 arc-seconds centred on the expected source position.
- Any other value is assumed to be a group of one or two NDFs that specify the " external" AST and PCA masks to be used. The way in which these NDFs are used depends on the value of parameter MASKTYPE. These NDFs must be aligned in pixel coordinates with the reference map (parameter REF) Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE.

[" AUTO"]

MASKOUT1 = LITERAL (Write)
If a non-null value is supplied for MASKOUT, it specifies the NDF in which to store the AST mask created from the NDF specified by parameter MASK. Only used if an NDF is supplied for parameter MASK. [!]

MASKOUT2 = LITERAL (Write)
If a non-null value is supplied for MASKOUT, it specifies the NDF in which to store the PCA mask created from the NDF specified by parameter MASK. Only used if an NDF is supplied for parameter MASK. [!]

MASKTYPE = LITERAL (Read)
Specifies the way in which NDFs supplied for parameter MASK are to be used. This parameter can be set to either of the following values:

- " Signal" : A single NDF should be supplied for parameter MASK holding the astronomical signal level at each pixel within the astronomical field being mapped. It can be in any units, but must have a Variance component. The AST and PCA masks are created from this map by finding all clumps of contiguous pixels above a fixed SNR limit, and then extending these clumps down to a lower SNR limit. For the AST model, the upper and lower SNR limits default to 3.0 and 2.0. For the PCA mask, the defaults are 5.0 and 3.0. These defaults can be over-ridden by supplying values for AST.ZERO_SNR, AST.ZERO_SNRLO, PCA.ZERO_SNR and PCA.ZERO_SNRLO within the configuration specified by parameter CONFIG. The AST and PCA masks created in this way can be saved using parameters MASKOUT1 and MASKOUT2.
- " Mask" : A pair of NDFs should be supplied for parameter MASK, each holding a mask in which background pixels have bad values and source pixels have good values. The first supplied NDF is used directly as the AST mask, and the second is used as the PCA mask.

[" Signal"]

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

MULTIOBJECT = _LOGICAL (Read)
Indicates if it is acceptable for the list of input files to include data for multiple

objects. If FALSE, an error is reported if data for more than one object is specified by parameter IN. Otherwise, no error is reported if multiple objects are found. [FALSE]

NEWMAPS = LITERAL (Read)

The name of a text file to create, in which to put the paths of all the new maps written to the directory specified by parameter MAPDIR (one per line). If a null (!) value is supplied no file is created. [!]

NORMALISE = _LOGICAL (Read)

If TRUE, scale corrections for individual observations found in any pre-existing auto-masked maps (e.g. made on a previous run of this script) are applied when creating new maps. If False, no scale corrections are applied. Scale correction factors are created and stored at the same time as the pointing corrections. The correction factor for a single observation is found by comparing the data values in the map made from the single observation with those in the coadd map made from all observation. The comparison is limited to the areas inside the AST mask, and a factor of unity is used for any observation that is not well correlated to the coadd). The factor found in this way is stored in the FITS extension of the map made from the observation (header " CHUNKFAC"). [FALSE]

NORTH = LITERAL (Read)

Specifies the celestial coordinate system to use as the reference direction in any newly created Q and U time series files. For instance if NORTH=" AZEL" , then they use the elevation axis as the reference direction, and if " ICRS" is supplied, they use the ICRS Declination axis. If " TRACKING" is supplied, they use north in the tracking system - what ever that may be. [" TRACKING"]

OBSWEIGHT = _LOGICAL (Write)

This parameter affects how maps from separate observations are weighted when they are combined together to form a coadd. If it is FALSE, each pixel in each map is weighted simply using the reciprocal of the Variance value stored in the map. If it is TRUE, an extra factor is included in the pixel weights that is constant for all pixels in a map but varies from observation to observation. In other words, each observation is assigned a weight, which is used to factor the pixel weights derived from the Variance values. The purpose of this per-observation weight is to down-weight observations that are very different to the other observations and which would therefore contribute to a high Variance if parameter MAPVAR is set TRUE. These weights are proportional to 1/(RMS∗RMS), where " RMS" is the RMS residual between an individual observation map and the coadd of all observation maps, after they have been aligned spatially to take account of any pointing error in the individual observation. See also parameter WEIGHTLIM.

If skyloop is used (see parameter SKYLOOP) with OBSWEIGHT=YES, then a set of observation maps must already exist in the MAPDIR directory that were also created with OBSWEIGHT=YES. The weights to be used by skyloop are read from these maps, which would normally have been created by a prior run of this script.

WARNING: This option should only be used if the number of observation being processed is sufficiently large to allow aberrant observations to be identified with a reasonable degree of confidence. [FALSE]

PIXSIZE = _REAL (Read)

Pixel dimensions in the output I, Q and U maps, in arcsec. The default is 4 arc-sec for both 450 and 850 um data. The bin size for the output catalogue can be specified

separately - see parameter BINSIZE and CAT. Note the pixel size within any maps supplied as input to this script should equal the value of parameter PIXSIZE. An error will be reported if this is not the case. [4]

QOUT = NDF (Write)
The output NDF in which to return the Q map including all supplied observations. This will be in units of pW. Supply null (!) if no Q map is required.

QUCONFIG = LITERAL (Read)
Extra parameter values to include in the MAKEMAP configuration used to create Q and U maps. The values specified by " QUCONFIG" are applied after those specified by " CONFIG" . [!]

QUDIR = LITTERAL (Read)
The name of a directory in which to put the Q, U and I time series generated by SMURF:CALCQU, prior to generating maps from them. If null (!) is supplied, they are placed in the same temporary directory as all the other intermediate files and so will be deleted when the script exists (unless parameter RETAIN is set TRUE). [!]

REF = NDF (Read)
An optional map defining the pixel grid for the output maps, and which is used to determine pointing corrections. If null (!) is supplied, then the map (if any) specified by parameter MASK is used. Note, an error is reported if the pixel size in the supplied map is not equal to the value of parameter PIXSIZE.

The value of the REF parameter is ignored if an NDF is supplied for one or more of parameters INITSKYI, INITSKYQ or INITSKYU. In such cases the first such NDF is used to define the pixel grid for all output maps. [!]

REUSE = _LOGICAL (Read)
If TRUE, then any output maps or time-treams that already exist (for instance, created by a previous run of this script) are re-used rather than being re-created from the corresponding input files. If FALSE, any previously created output maps or time-streams are ignored and new ones are created from the corresponding input files. [TRUE]

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

SKYLOOP = _LOGICAL (Read)
Should the skyloop script be used in place of makemap to create the maps from the I, Q and U time-series data? Note, when using skyloop it is not possible to add in new observations to an existing collection of I, Q and U maps - all observations must be processed together. Therefore the value supplied for parameter REUSE will be ignored and a value of FALSE assumed if the MAPDIR directory is missing maps for any of the supplied observations. [FALSE]

SMOOTH450 = _LOGICAL (Read)
This parameter is only accessed if the input files specified by parameter IN contain 450 um data. If SMOOTH450 is TRUE, the 450 um I, Q and U coadd maps will be smoothed prior to creating any output vector catalogue so that the smoothed maps have the 850 um beam shape. This changes the resolution of the maps from (approximately) 8 arc-sec (the 450 um beam width) to (approximately) 13 arc-seconds

(the 850 um beam width), and also reduces the noise. The smoothing kernel is derived from the two-component beam models described in the paper " A Decade of SCUBA-2: A Comprehensive Guide to Calibrating 450 um and 850 um Continuum Data at the JCMT" (Mairs et al, 2021). If parameter MAPVAR is TRUE, the individual observation maps will be smoothed prior to determining the variances, thus ensuring that the resulting variances are still accurate. [FALSE]

TRIM = _REAL (Read)

This indicates how the edges of the final I, Q and U coadds should be trimmed to remove the noisey edges. If a null (!) value is supplied (the default), no trimming is performed. Otherwise, the supplied value indicates the fraction of the mean expsoure time at which the coadd should be trimmed. For instance, a value of 0.2 causes pixels to be set bad if the number of usable bolometer samples that fall within the pixel is less than 0.2 times the mean number of samples per pixel, taken over the whole coadd (excluding parts that receive no samples at all). [!]

UOUT = NDF (Write)

The output NDF in which to return the U map including all supplied observations. This will be in units of pW. Supply null (!) if no U map is required.

WEIGHTLIM = _REAL (Read)

The lowest usable weight (see parameter OBSWEIGHT). Any observation that has a weight below this value will not be included in the final coadded I, Q or U maps or in the vector catalogue. This can be useful since observations with very low weight can sometimes cause makemap to crash. [0.05]

POL2SIM
Create simulated POL2 data from known I, Q and U maps

Description:

This script creates simulated POL2 time-series data, using a supplied real POL2 observation as a template. The maps containing the artificial I, Q and U values at each point on the sky can be created automatically, or can be supplied by the user (see parameters ARTI, ARTQ and ARTU).

There are two basic modes of operation, selected by the parameter ADDON:

- If ADDON is True, noise-free artificial POL2 time-stream data is generated by sampling supplied I, Q and U maps at the position of each bolometer value in the supplied real POL2 observation, and the artificial time-stream data is then added onto the real time-stream data to generate the output time-streams. In this mode, no artificial noise components are added into the artificial time-stream data. Instead, the noise in the output time-streams is inherited from the noise in the input real POL2 time-streams
- If ADDON is False, artificial POL2 time-stream data is generated in the same way, but is then used directly as the output time-sreams without first adding on the real time-stream data. In this mode, artificial noise components are added into the artificial time-stream data as follows (NOTE - this mode takes a very long time to run !!):

o A time-varying unpolarised sky brightness is added onto the I values amapled from the supplied ARTI map. The sky brightness is common to all bolometers, and is derived from the time-series data for a a real scan map (see parameter INCOM). For some unknown reason, the common-mode for real POL2 data seems to be much flatter than the common-mode for real non-POL2 data. For this reason, there is an option to flatten the common-mode before using it in the simulation (see parameter CFACTOR). o 2, 4 and 16 Hz signals proportional to the total intensity (including sky brightness) in included in the simulated POL2 data. o Each bolometer has a separate gain, which is allowed to vary over time (like the GAI model used by smurf:makemap). The GAI values are derived from the template POL2 data supplied for parameter IN. The extent to which the GAI values vary with time is controlled by parameter GFACTOR o Random Gaussian noise is added to the returned time-stream data.

In both modes, instrumental polarisation is included in the artificial time-stream data (see parameter IPFORM). Three forms of instrumental polarisation can be used: the Johnstone/Kennedy IPDATA model, the simplified planetary data " PL1" model, or an arbitrary user-define IP model (see parameters IPMAX, IPMIN and IPTHETA).

Usage:

```
pol2sim in out newart arti artq artu
```

Parameters:

ADDON = _LOGICAL (Read)

If True, the output time-stream data consists of the sum of the artificial time-stream

data (generated by sampling the maps given by parameters ARTI, ARTQ and ARTU) and the real time-stream data (given by parameter IN). If False, the output time-stream data consists just of the artificial data. [False]

AMP2 = _DOUBLE (Read)

Controls the amplitude of the 2 Hz signal. It gives the amplitude of the 2 Hz signal as a fraction of the total intensity. See also " PHASE2" . Only used if ADDON is False. [0.0003]

AMP4 = _DOUBLE (Read)

Controls the amplitude of the 4 Hz signal. It gives the amplitude of the 4 Hz signal as a fraction of the total intensity. See also " PHASE4" . Only used if ADDON is False. [0.009]

AMP16 = _DOUBLE (Read)

Controls the amplitude of the 16 Hz signal. It gives the amplitude of the 16 Hz signal as a fraction of the total intensity. See also " PHASE16" . Only used if ADDON is False. [0.0008]

ARTFORM = _INTEGER (Read)

Indicates the form of polarisation pattern created if parameter NEWART is set True:

0 - A single Gaussian source centred at pixel coordinates given by parameters XC and YC (default is (0,0)), with peak total intensity given by parameter IPEAK and width given by parameter IFWHM. The polarisation vectors are tangential, centred on the source. The fractional polarisation is constant at the value given by POL.

1 - Exactly like " 1" except that the vectors are parallel across the whole Gaussian blob, rather than being tangential. The vectors are parallel to the Y pixel axis.

[0]

ARTI = NDF (Read or write)

A 2D NDF holding the artificial total intensity map from which the returned time-stream data is derived. If the NEWART parameter is True, then a new artificial I map is created and stored in a new NDF with name specified by ARTI. If NEWART is False, then ARTI should specify an existing NDF on entry, which is used as the artificial I map. The values in this NDF include the effects of the degradation caused by placing POL2 into the beam, (a factor of 1.35).

ARTQ = NDF (Read or write)

A 2D NDF holding the artificial Q map from which the returned time-stream data is derived. If the NEWART parameter is True, then a new artificial Q map is created and stored in a new NDF with name specified by ARTQ. If NEWART is False, then ARTQ should specify an existing NDF on entry, which is used as the artificial Q map. The values in this NDF include the effects of the degradation caused by placing POL2 into the beam, (a factor of 1.35).

ARTU = NDF (Read or write)

A 2D NDF holding the artificial U map from which the returned time-stream data is derived. If the NEWART parameter is True, then a new artificial U map is created and stored in a new NDF with name specified by ARTU. If NEWART is False, then ARTU should specify an existing NDF on entry, which is used as the artificial U map. The values in this NDF include the effects of the degradation caused by placing POL2 into the beam, (a factor of 1.35).

CFACTOR = _DOUBLE (Read)

A factor by which to expand the COM model values derived from the supplied INCOM data. The expansion is centred on the mean value. Real POL2 data seems to have a much flatter common-mode than real non-POL2 data, so the default flattens the common-mode to some extent. Only used if ADDON is False. [0.2]

COMVAL1 = _DOUBLE (Read)

Only used if ADDON is False and INCOM is null (!). If supplied,

COMVAL1 should be a constant sky emission value in pW seen by all :

bolometers and time slices. The total common-mode signal added to the simulated data is the sum of COMVAL1 and COMVAL2, but only COMVAL1 is considered when determing the Q and U values caused by Instrumental Polarisation. Supplying null (!) is equivalent to supplying zero. [0.0] COMVAL2 = _DOUBLE (Read) Only used if ADDON is False and INCOM is null (!). If supplied,

COMVAL2 should be a constant offset to add to all bolometers and :

time slices representing an offset in the electronics (i.e. not caused by sky emission). The total common-mode signal added to the simulated data is the sum of COMVAL1 and COMVAL2, but only COMVAL1 is considered when determing the Q and U values caused by Instrumental Polarisation. Supplying null (!) is equivalent to supplying zero. [0.0] GFACTOR = _DOUBLE (Read) A factor by which to expand the GAI model values derived from the supplied time-series data. The expansion is centred on the value 1.0. No GAI model is used if GFACTOR is zero. Only used if ADDON is False and INCOM is not null (!). [1.0] GLEVEL = LITERAL (Read) Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" ATASK"] IFWHM = _DOUBLE (Read) FWHM of Gaussian source for new artificial total instensity map, in pixels. [8] ILEVEL = LITERAL (Read) Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information. In addition scatter plots showing how each Q and U image compares to the mean Q and U image are displayed at this ILEVEL.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"] IN = NDF (Read) A group of POL2

time series NDFs. INCOM = NDF (Read) A group of non-POL2 time series NDFs that are used to define the common-mode (i.e. the sky brightness) to include in the simulated POL2 data. The number of NDFs supplied for INCOM must equal the number supplied for IN. Each INCOM file must be at least as long (in time) as the corresponding IN file. If null (!) is supplied, the common-mode is defined by parameters COMVAL1 and COMVAL2 instead. Only used if ADDON is False. [!] IPFORM = LITERAL (Read) The form of instrumental polarisation to use. Can be " JK" for the Johnstone/Kennedy model, " PL1" , " PL2" or " PL3" for one of the simplified planetary data models, " User" for a user-defined model (see parameters IPMAX, IPMIN and IPTHETA), or " None" if no instrumental polarisation is to be added to the simulated data. [" PL3"] IPEAK = _DOUBLE (Read) Peak intensity for new artificial total instensity map, in pW. [0.08] IPMIN = _DOUBLE (Read) The minimum instrumental polarisation within the focal plane, expressed as a fraction. The IP varies linearly across each array from IPMIN to IPMAX. The IP is fixed in focal plane coordinates over all stare positions. This parameter is only accessed if IPFORM is set to " User" . [0.0004] IPMAX = _DOUBLE (Read) The maximum instrumental polarisation within the focal plane, expressed as a fraction. The IP varies linearly across each array from IPMIN to IPMAX. The IP is fixed in focal plane coordinates over all stare positions. This parameter is only accessed if IPFORM is set to " User" . [0.0008] IPTHETA = _DOUBLE (Read) The angle from the focal plane Y axis to the instrumental polarisation vectors, in degrees. This parameter is only accessed if IPFORM is set to " User" . [15] LOGFILE = LITERAL (Read) The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. The script can change the logfile if necessary by assign the new log file path to the module variable " starutil.logfile" . Any old log file will be closed befopre the new one is opened. [] MSG_FILTER = LITERAL (Read) Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"] NEWART = _LOGICAL (Read) Indicates if new artificial I, Q and U maps should be created. These are the maps from which the returned time-stream data are derived.

If NEWART is False, then existing 2D NDFs holding the artificial I, Q and U data to be used should be specified via parameter ARTI, ARTQ and ARTU. These maps should have WCS that is consistent with the supplied template time-stream data (parameter IN). The data values are assumed to be in units of " pW" . The Y pixel axis is assumed to be the polarimetric reference direction.

If NEWART is True, then new artificial I, Q and U data is created containing data of the form indicated by parameter ARTFORM. The Y pixel axis is reference direction (suitable POLANAL Frames are included in the WCS to indicate this, as required by POLPACK). OUT = NDF (Write) A group of output NDFs to hold the simulated POL2 time series data. Equal in number to the files in " IN" . PERROR = _DOUBLE (Read) Standard deviation of pointing errors to include in the simulated data, in arc-seconds. [0.0] PHASE2 = _DOUBLE (Read) The phase offset to apply to the 2 Hz signal specified via parameter AMP2, in degrees. Only used if ADDON is False. [0.0] PHASE4 = _DOUBLE (Read) The phase offset to apply to the 4 Hz signal specified via parameter AMP4, in degrees. Only used if ADDON is False. [-30.0] PHASE16 = _DOUBLE (Read) The phase offset to apply to

the 16 Hz signal specified via parameter AMP16, in degrees. Only used if ADDON is False. [0.0] POL = _DOUBLE (Read) The fractional polarisation for new artificial Q and U maps. [0.05] RESTART = LITERAL (Read) If a value is assigned to this parameter, it should be the path to a directory containing the intermediate files created by a previous run of POL2SIM (it is necessry to run POL2SIM with RETAIN=YES otherwise the directory is deleted after POL2SIM terminates). If supplied, any files which can be re-used from the supplied directory are re-used, thus speeding things up. The path to the intermediate files can be found by examining the log file created by the previous run. [!] RETAIN = _LOGICAL (Read) Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE] SIGMA = _DOUBLE (Read) Gaussian noise level (in pW) to add to the final data. Only used if ADDON is False. [0.004] XC = _DOUBLE (Read) The X pixel coordinate at which to place the artificial blob if NEWART is YES. [0.0] YC = _DOUBLE (Read) The Y pixel coordinate at which to place the artificial blob if NEWART is YES. [0.0]

POL2IP
Create an Instrumental Polarisation (IP) model from a set of POL2 observations

Description:

This script produces Q and U maps from a supplied list of POL2 planet observations (this list should include observations over a wide range of elevations). It then estimates the parameters of an IP model that gives good estimates of the resulting Q and U, based on a supplied total intensity map of the planet.

It is assumed that the source is centred at the reference point of the supplied observations.

An IP model gives the normalised Q and U values (Qn and Un) with respect to focal plane Y axis, at any point on the sky, as functions of elevation. The correction is applied as follows:

Q_corrected = Q_original - I∗Qn U_corrected = U_original - I∗Un

where " I" is the total intensity at the same point on the sky as Q_original and U_original. All (Q,U) values use the focal plane Y axis as the reference direction.

The " PL2" IP model is as follows (" el" = elevation in radians):

p1 = A + B∗el + C∗el∗el Qn = I∗p1∗cos(-2∗(el-D)) Un = I∗p1∗sin(-2∗(el-D))

It is parameterised by four constants A, B, C and D, which are calculated by this script. It represents an instrumental polarisation that varies in size with elevation but is always at a fixed angle (D radians) from the elevation axis.

The PL2 model replaces the earlier PL1 model. The difference is that the D constant was fixed at zero in the PL1 model.

Usage:

```
pol2ip obslist iref [diam] [pixsize]
```

Parameters:

DIAM = _REAL (Read)

The diameter of the circle (in arc-seconds), centred on the source, over which the mean Q, U and I values are found. If zero, or a negative value, is supplied, the fit is based on the weighted mean values within the source, where the spatial weiging function is a fitted beam shape determined using kappa:beamfit. In this case, a text file called " beamfit.asc" is created in the current directory. This is a table containg columns of the geometric properties of the polarised intensity beam in each observation. The header for this file contains the parameters of quadratic fits to these properties, which are used within pol2scan. [40]

ILEVEL = LITERAL (Read)

Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information. In addition scatter plots showing how each Q and U image compares to the mean Q and U image are displayed at this ILEVEL.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"]

IREF = NDF (Read)

A 2D NDF holding a map of total intensity (in pW) for the object covered by the observations in OBSLIST. It is assumed that the object is centred at the reference point in the map. The supplied map is resampled to to give it the pixel size specified by parameter PIXSIZE. If a null value(!) is supplied, the total intensity is determined from the POL2 data itself.

LOGFILE = LITERAL (Read)

The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. The script can change the logfile if necessary by assign the new log file path to the module variable " starutil.logfile" . Any old log file will be closed befopre the new one is opened. []

MAPDIR = LITERAL (Read)

Path to a directory containing any pre-existing Q/U/I maps. Each UT date should have a separate subdirectory within " mapdir" , and each observation should have a separate subdirectory within its <UT> date subdirectory. Any new Q/U/I maps created by this script are placed in this directory. If null (!) is supplied, the root directory containing the Q/U maps is placed within the temporary directory used to store all other intermediate files. [!]

MSG_FILTER = LITERAL (Read)

Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

OBSLIST = LITERAL (Read)

The path to a text file listing the POL2 observations to use. Each line should contain a string of the form " <ut>/<obs>" , where <ut> is the 8 digit UT date (e.g. " 20151009") and <obs> is the 5 digit observation number (e.g. " 00034"). Blank lines and lines starting with " #" are ignored. The raw data for all observations is expected to reside in a directory given by environment variable " SC2" , within subdirectories with paths of the form: $SC2/[s4a|s8a]/20150918/00056/ etc. The choice of " s8" or " s4" is made on the basis of parameter WAVEBAND.

PIXSIZE = _REAL (Read)
Pixel dimensions in the Q and U maps, in arcsec. The default is 4 arc-sec for 850 um data and 2 arc-sec for 450 um data. []

RESTART = LITERAL (Read)
If a value is assigned to this parameter, it should be the path to a directory containing the intermediate files created by a previous run of POL2IP (it is necessry to run POL2IP with RETAIN=YES otherwise the directory is deleted after POL2IP terminates). If supplied, any files which can be re-used from the supplied directory are re-used, thus speeding things up. The path to the intermediate files can be found by examining the log file created by the previous run. [!]

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

QUDIR = LITERAL (Read)
Path to a directory containing any pre-existing Q/U/I time streams. Each UT date should have a separate subdirectory within " qudir" , and each observation should have a separate subdirectory within its <UT> date subdirectory. Any new Q/U/I time streams created by this script are placed in this directory. If null (!) is supplied, the root directory containing the Q/U time streams is placed within the temporary directory used to store all other intermediate files. [!]

TABLE = LITERAL (Read)
The path to a new text file to create in which to place a table holding columns of elevation, Q, U, Qfit and Ufit (and various other useful things), in TOPCAT ASCII format. [!]

TABLEIN = LITERAL (Read)
The path to an existing text file containing a table created by a previous run of this script, using the TABLE parameter. If supplied, none of the other parameters are accessed, and a fit is performed to the values in the supplied table. [!]

USEMEANI = _LOGICAL (Read)
If TRUE, the Q and U values for all observations are normalised using the same total intensity value - the mean of the total intensity values averaged over all observations. If FALSE, the Q and U values for each observation are normalised using the total intensity value for that observation. [TRUE]

WAVEBAND = LITERAL (Read)
Indicates the waveband - " 450" or " 850" .

POL2STACK
Combine multiple Q, U and I images and create a vector catalogue from them

Description:

This script combines multiple Q, U and I images and creates a vector catalogue from them.

By default, the Q, U, I and PI catalogue values, together with the maps specified by parameters " QUI" and " PI" , are in units of Jy/beam (see parameter Jy).

Usage:

```
pol2stack inq inu ini cat pi [retain] [qui] [in] [msg_filter] [ilevel] [glevel]
[logfile]
```

Parameters:

CAT = LITERAL (Read)
The output FITS vector catalogue.

DEBIAS = LOGICAL (Given)
TRUE if a correction for statistical bias is to be made to percentage polarization and polarized intensity. [FALSE]

GLEVEL = LITERAL (Read)
Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" ATASK"]

ILEVEL = LITERAL (Read)
Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information. In addition scatter plots showing how each Q and U image compares to the mean Q and U image are displayed at this ILEVEL.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"]

IN = Literal (Read)
A group of container files, each containing three 2D NDFs in components Q, U and I, as created using the QUI parameter of the pol2cat script. Parameters INQ, INU and INI are used if a null (!) value is supplied for IN. [!]

INI = Literal (Read)
A group of input I maps in units of pW. Only used if a null value is supplied for parameter IN.

INQ = Literal (Read)
A group of input Q maps in unts of pW. Only used if a null value is supplied for parameter IN.

INU = Literal (Read)
A group of input U maps in units of pW. Only used if a null value is supplied for parameter IN.

JY = _LOGICAL (Read)
If TRUE, the output catalogue, and the output Q, U, PI and I maps will be in units of Jy/beam. Otherwise they will be in units of pW (in this case, the I values will be scaled to take account of any difference in FCFs for POL-2 and non-POL-2 observations). [True]

LOGFILE = LITERAL (Read)
The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. The script can change the logfile if necessary by assign the new log file path to the module variable " starutil.logfile" . Any old log file will be closed befopre the new one is opened. []

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

PI = NDF (Read)
The output NDF in which to return the polarised intensity map. No polarised intensity map will be created if null (!) is supplied. If a value is supplied for parameter IREF, then PI defaults to null. Otherwise, the user is prompted for a value if none was supplied on the command line. []

QUI = NDF (Read)
If a value is supplied for QUI, the total Q, U and I images that go into the final polarisation vector catalogue will be saved to disk as a set of three 2D NDFs. The three NDFs are stored in a single container file, with path given by QUI. So for instance if QUI is set to " stokes.sdf" , the Q, U and I images can be accessed as " stokes.q" , " stokes.u" and " stokes.i" . [!]

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

POL2NOISE
Analyse the noise in a POL2 vector catalogue

Description:

This script manipulates the noise estimates contained within a POL2 vector catalogue. Currently two options are available, selected by parameter MODE:

- " Display" - this displays the noise values stored in a specified column of the catalogue (e.g. " DQ") and compares them to the noise values estimated from the variations of the corresponding value column (e.g. " Q"). This allows the noise values (" DQ" etc) to be verified.
- " Remodel" - This replaces all the noise values stored in a specified catalogue with values based on smoother models of the background and source noise. This helps to remove outlying noise estimates that are anomalously low or high because of the random nature of the MAPVAR errors created by the pol2map script.

Usage:

```
pol2noise cat column [mode] [perc] [presmooth] [style] [device]
```

Parameters:

CAT = LITERAL (Read)

The input vector catalogue. This should have been created by POL2MAP.

COLUMN = LITERAL (Read)

The name of the catalogue column to be used if parameter MODE is " Display" . Both the named column and the associated error column (" <X>" and " D<X>") must exist in the catalogue. The name must be one of " Q" , " U" , " I" or " PI" .

DEBIASTYPE = LOGICAL (Given)

Gives the type of bias estimator to use if paremeter MODE is " Remodel" , using the nomeclature of Montier at al " Polarization measurements analysis II. Best estimators of polarization fraction and angle" (A&A, 2018):

- " AS" : The asymptotic estimator. See section 2.3 of Montier et al. This estimator produces bad P and PI values if the squared PI value is less than the variance in PI.
- " MAS" : The modified asymptotic estimator. See section 2.5 of Montier et al. This estimator does not produces bad P and PI values, even if the squared PI value is less than the variance in PI.
- " None" : No de-biasing.

DEVICE = DEVICE (Read)

The graphics workstation to use. The default is the current graphics device as previously set using KAPPA:GDSET. If GDSET has not been used to establish a current graphics device, the user is prompted for a device. Use KAPPA:GDNAMES to see a list of available device names. []

EXPTIME = NDF (Read)
An NDF that contains an exposure time map for the data in the supplied catalogue. Only used if parameter MODE is " Remodel" . For instance, the " iext" , " qext" or " uext" map that was created by POL2MAP at the same time as the catalogue could be supplied.

GLEVEL = LITERAL (Read)
Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. Note, the default value is " NONE" , which caused no log file to be created. Setting this parameter to another value (e.g. " ATASK") causes the log file to be produced. [" NONE"]

ILEVEL = LITERAL (Read)
Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information.

[" PROGRESS"]

LOGFILE = LITERAL (Read)
The name of the log file to create if GLEVEL is not NONE (which it is by default). The default log file name is " pol2noise.log" and is created in the current directory. Any file with the same name is over-written. Any old log file will be closed before the new one is opened. []

MODE = LITERAL (Read)
The operation to be performed on the input catalogue specified by parameter CAT:

- " DISPLAY" : Verify the noise estimates on a single quantity by comparing them to the local variations of the quantity. See parameters COLUMN, DEVICE, PER, PRESMOOTH.
- " REMODEL" : Replace the noise estimates in the catalogue using a smoother model. See parameters OUT, EXPTIME, DEBIASTYPE.

[" Display"]

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

OUT = LITERAL (Read)
The output FITS vector catalogue. Only used if parameter MODE is " Remodel" .

PERC = _REAL (Read)

The percentile corresponding to the highest " D<X>" value to include in the scatter plot. Only used if parameter MODE is " Display" . In the range 20 to 100. A value below 100 causes the edge pixels, which usually have very high variances, to be excluded from the plot. A red contour is displayed over the " D<X>" map indicating the noise level corresponding to the value of PERC. [90]

PRESMOOTH = _REAL (Read)

Controls initial smoothing of the " D<X>" map in " Method 2" . If a value is supplied for PRESMOOTH, then the " D<X>" map read from the catalogue is first smoothed using a Gaussian kernel before being used. The value of PRESMOOTH gives the FWHM of the Gaussian kernel, in pixels. If a null (!) value is supplied for PRESMOOTH (which is the default value), the " D<X>" values read from the catalogue are used directly as the second noise estimate, without any smoothing. [!]

RETAIN = _LOGICAL (Read)

Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

STYLE = LITERAL (Read)

A group of attribute settings describing the plotting style to use for the annotated axes, etc.

A comma-separated list of strings should be given in which each string is either an attribute setting, or the name of a text file preceded by an up-arrow character " ^" . Such text files should contain further comma-separated lists which will be read and interpreted in the same manner. Attribute settings are applied in the order in which they occur within the list, with later settings overriding any earlier settings given for the same attribute.

Each individual attribute setting should be of the form:

<name>=<value>

where <name> is the name of a plotting attribute, and <value> is the value to assign to the attribute. Default values will be used for any unspecified attributes. All attributes will be defaulted if a null value (!)—the initial default—is supplied.

See section " Plotting Attributes" in SUN/95 for a description of the available attributes. Any unrecognised attributes are ignored (no error is reported). The appearance of the markers in the scatter plot is controlled by the attributes " Colour(Markers)" , " Width(Markers)" , etc. Likewise the appearance of the best fit line (and the contour lines) is controlled using " Colour(Curves)" , " Width(Curves)" , etc. [current value]

Display Mode :

The " display" mode compares the error estimates stored in the error columns of the catalogue (e.g. the values in the DI column) with estimates of the errors derived from the value columns (e.g. the I column). This comparison is limited to regions where the true astronomical signal can be assumed to be zero (i.e. the background regions). This mode can only be used with the intensity-like values in the catalogue (e.g. I, Q, U and PI).

Two different methods are used for comparing the noise values. Ideally they should both show that the error columns stored in the catalogue accurately describe the noise in the data columns. However, in reality they will both give reports that depart from this ideal by

differing amounts. Results for both methods are displayed on the graphics device specified by parameter DEVICE. The results from " Method 1" are displayed in the lower row of three pictures on the graphics device, and the results from " Method 2" are displayed in the upper row of three pictures. For convenience, the scatter plot produced by method 1 (top right picture) is overlayed in blue on top of the scatter plot produced by method 1 (lower right plot).

Method 1:

Firstly, a mask is created that is later used to identify background regions. This is based on the total intensity, I, regardless of the column being checked, since I is usually much brighter than Q, U or PI. The values from catalogue columns " I" and " DI" are extracted into a pair of 2-dimensional maps. A basic SNR map is then formed (I/DI) and the FellWalker algorithm within the Starlink CUPID package (see SUN/255) is used to identify contiguous clumps of pixels with SNR higher than 5 and to extend each such clump down to an SNR of 2.

Next, the values from the requested catalogue columns, " <X>" and " D<X>" , are extracted into a pair of 2-dimensional maps and masked to remove source regions (i.e. the clumps of high SNR identified by FellWalker).

The full range of " D<X>" values in the remaining background is divided into a set of bins, and each " <X>" value is then placed into a bin on the basis of its corresponding " D<X>" value. The " <X>" values in each bin should in principle have a mean value of zero since they are all background pixels. The standard deviation of the " <X>" values in each bin is found and plotted against the " D<X>" value associated with the bin (which varies across the map, being larger nearer the edges). Ideally the resulting best fit line should have a slope of unity and an offset of zero, indicating that the noise estimate " D<X>" associated with each bin is a good measure of the standard deviation of the " <X>" values in the bin.

The masked " <X>" and " D<X>" maps are displayed using a common scaling on the graphics device specified by parameter DEVICE. A scatter plot showing the standard deviation of the " <X>" values in each bin on the vertical axis and the RMS " D<X>" value in each bin on the horizontal axis is also displayed. The slope and offset of the best fitting straight line are displayed on standard output, together with the RMS residual of the fit. The upper data limit included in the scatter plot is displayed as a red contour on the two map.

The " D<X>" map may optionally be smoothed using a Gaussian kernel before being used - see parameter PRESMOOTH.

The size of each " D<X>" bin and the data included in the scatter plot can make a significant difference to the final slope and offset. The first bin (lowest D<X>) is centred on the peak of the D<X> histogram. This histogram is usually heavily skewed with a very rapid rise at low D<X> values followed by a long tail to higher D<X> values. The bin width is 1.5 times the FWHM of the histogram peak, as determined solely from the D<X> values below the peak. All bins have equal width, and the highest bin includes the D<X> value corresponding to the value of parameter PERC. Any points below the first bin or above the last bin are excluded from the scatter plot. This binning scheme aims to reduce statistical bias at the low D<X> end, which tends to cause the lowest D<X> points in the scatter plot to be slightly higher than they should be. This bias is caused by there being few points at lower D<X> to balance those with higher D<X> value.

Method 2:

Firstly, the values from catalogue columns " I" and " DI" are extracted into a pair of 2-dimensional maps. A basic SNR map is then formed (I/DI) and significant spatial structures are identified and blanked out using the KAPPA:FFCLEAN command on the SNR map. The SNR map is used here, instead of simply using " I" , in order to flatten the noise level across the map, which helps FFLCEAN. Each blanked out region in this mask (i.e. each source area) is then enlarged slightly to remove any remaining nearby low-level source pixels.

Next, the values from catalogue columns " <X>" and " D<X>" are extracted into a pair of 2-dimensional maps and masked (using the mask described above) to remove source regions.

The first noise estimate measures the spatial variation in pixel value in the neighbourhood of each pixel in the masked " <X>" map. It is formed by first squaring the masked " <X>" map, then smoothing the squared map using a Gaussian smoothing kernel, then taking the square root of the smoothed map. Thus each pixel value in the final map represents the RMS of the nearby pixels in masked " <X>" map. The FWHM of the Gaussian smoothing kernel is chosen in order to maximise the correlation between the first and second estimates of the noise.

The " D<X>" map, which holds the second noise estimate, may optionally be smoothed using a Gaussian kernel before being used - see parameter PRESMOOTH.

The maps holding the two masked noise estimates are displayed using a common scaling on the graphics device specified by parameter DEVICE. A scatter plot of the values in these two maps is also displayed. The slope and offset of the best fitting straight line, based on the visible points in the scatter plot, are displayed on standard output, together with the RMS residual of the fit. The upper data limits to be included in the scatter plot can be controlled by parameter PERC, and are displayed as red contours on the two maps.

Remodel Mode :

The " remodel" mode creates an output catalogue holding a copy of the input catalogue, and then calculates new values for all the error columns in the output catalogue. The new I, Q and U error values are first derived from a three component model of the noise in each quantity:

The " background component" is derived from the exposure time map (obtained using parameter EXPTIME). The background component is equal to " A∗(exptime^B)" where A and B are constants determined by doing a linear fit between the log of the noise estimate in the catalogue (DQ, DU or DI) and the log of the exposure time (in practice, B is usually close to -0.5). The fit is limited to background areas in the signal map, but also excludes a thin rim around the edge of the map where the original noise estimates are subject to large inaccuracies. Since the exposure time map is usually very much smoother than the original noise estimates, the background component is also much smoother.

The " source component" represents the extra noise found in and around compact sources and caused by pointing errors, calibration errors, etc. The background component is first subtracted from the catalogue noise estimates and the residual noise values are then modelled using a collection of Gaussians. This modeling is done using the GaussClumps algorithm provided by the findclumps command in the Starlink CUPID package. The noise residuals are first divided into a number of " islands" , each island being a collection of

contiguous pixels with noise residual significantly higher than zero (this is done using the FellWalker algorithm in CUPID). The GaussClumps algorithm is then used to model the noise residuals in each island. The resulting model is smoothed lightly using a Gaussian kernel of FWHM 1.2 pixels.

The " residual component" represents any noise not accounted for by the other two models. The noise residuals are first found by subtracting the other two components from the original catalogue noise estimates. Any strong outlier values are removed and the results are smoothed more heavily using a Gaussian kernel of FWHM 4 pixels.

The final model is the sum of the above three components. The new DI, DQ and DU values are found independently using the above method. The errors for the derived quantities (DPI, DP and DANG) are then found from DQ, DU and DI using the usual error popagation formulae. Finally new P and PI values are found using a specified form of de-biasing (see parameter DEBIASTYPE).

The results of the re-modelling are displayed on the graphics device specified by parameter DEVICE. A row of four pictures is created for each Stokes parametyer (I, Q and U). From left to right, these are:

- An image of the original error estimates in the supplied catalogue.
- An image of the re-modelled error estimates in the output catalogue.
- An image of the residuals between original and re-modelled error estimates.
- A scatter plot of re-modelled against original error estimates.

The images of the original and re-modelled error estimates use the same scaling. The image of the residuals is scaled between the 2nd and 98th percentiles.

POL2SCAN
Create Q and U maps from a set of POL-2 " spin&scan" data files

Description:

This script runs SMURF:CALCQU on any raw POL-2 data files specified by parameter IN, to create a set of down-sampled time series files holding Q and U values in each bolometer. These are placed in the directory specified by parameter QUDIR. These time series, together with any Q,U time series files specified directly by parameter IN, are then converted into Q and U maps using SMURF:MAKEMAP. A separate pair of Q and U maps is created for each observation present in the supplied data. These maps, together with any Q,U maps specified by parameter IN, are then coadded to form a final pair of Q and U maps. In addition, a range of information derived from each pair of maps included in the coadd is displayed on the screen, and may also be written to an output text file (see parameter OBSTABLE).

The final Q and U maps, together with the supplied total intensity reference map (see parameter IPREF) are then used to create a vector catalogue and polarised intensity map (see parameters CAT, PI and DEBIAS).

Correction for instrumental polarisation is made only if a value is supplied for parameter IPREF.

By default, the Q, U, I and PI catalogue values, together with the maps specified by parameters " Q" , " U" and " PI" , are in units of Jy/beam (see parameter Jy).

Usage:

```
pol2scan in q u [cat] [ipref] [config] [pixsize] [qudir] [mapdir] [obstable]
[retain] [msg_filter] [ilevel] [glevel] [logfile]
```

Parameters:

ALIGN = LOGICAL (Read)

If TRUE, and if a non-null value is supplied for parameter REF, then corrections to the telescope pointing are determined and applied when creating the Q and U maps for each observation. To determine this pointing correction for an observation, a total intensity map is created from the raw POL2 data for the observation using the supplied reference map (see parameter REF) to specify the pixel grid. Due to pointing errors, this total intensity map may not be aligned accurately with the reference map (errors of up to 8 arc-seconds have been seen). The positional shift required to minimise the residuals between the shifted total intensity map and the reference map is found, and used to correct the pointing when creating the Q and U map. Note, this option slows down the operation of pol2scan as it requires an extra invocation of smurf:makemap for each observation, to create the total intensity maps (which is not required if ALIGN is FALSE). [TRUE]

CAT = LITERAL (Read)

The output FITS vector catalogue. No catalogue is created if null (!) is supplied. [!]

CONFIG = LITERAL (Read)

The MAKEMAP configuration parameter values to use. If a null value (!) or " def" is

supplied, the following defaults will be used:

```
ast.zero_snr=3
ast.zero_snrlo=2
maptol=0.05
modelorder=(pca,ext,ast,noi)
noisecliphigh=3
numiter=-20
pca.pcathresh=4
spikebox=10
spikethresh=5
```

If a configuration is supplied, it is used in place of the above default configurations. In either case, the following values are always appended to the end of the used config (whether external or defaulted):

```
flagslow = 0.01
downsampscale = 0
noi.usevar=1
```

DEBIAS = LOGICAL (Given)
TRUE if a correction for statistical bias is to be made to percentage polarization and polarized intensity in the output vector catalogue specified by parameter CAT and the polarised intensity map specified by parameter PI. [FALSE]

GLEVEL = LITERAL (Read)
Controls the level of information to write to a text log file. Allowed values are as for " ILEVEL" . The log file to create is specified via parameter " LOGFILE. In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" ATASK"]

ILEVEL = LITERAL (Read)
Controls the level of information displayed on the screen by the script. It can take any of the following values (note, these values are purposefully different to the SUN/104 values to avoid confusion in their effects):

- " NONE" : No screen output is created
- " CRITICAL" : Only critical messages are displayed such as warnings.
- " PROGRESS" : Extra messages indicating script progress are also displayed.
- " ATASK" : Extra messages are also displayed describing each atask invocation. Lines starting with " >>>" indicate the command name and parameter values, and subsequent lines hold the screen output generated by the command.
- " DEBUG" : Extra messages are also displayed containing unspecified debugging information.

In adition, the glevel value can be changed by assigning a new integer value (one of starutil.NONE, starutil.CRITICAL, starutil.PROGRESS, starutil.ATASK or starutil.DEBUG) to the module variable starutil.glevel. [" PROGRESS"]

IN = NDF (Read)

A group of input files. Each specified file must be one of the following types:

- a raw POL-2 data file
- a time-series file holding Stokes Q, U or I values
- a two-dimensional map holding Stokes Q or U values (Q and U maps must be in units of pW). Any combination of the above types can be supplied.

IPBEAMFIX = _LOGICAL (Read)

Should the supplied total intensity reference image (parameter IPREF) be modified so that its beam shape matches the expected IP beam shape at the elevation of each supplied POL2 observation, before doing IP correction? This is currently an experimental feature. [FALSE]

IPFCF = _REAL (Read)

The FCF that should be used to convert the supplied IP REF map to pW. This parameter is only used if the supplied IPREF map is not already in units of pW, and if the FCF is not stored in the FITS extension of the map. The suggested default is the standard FCF for the band concerned (450 or 840). Just press return at the prompt to use this default, or enter a new value if the suggested value is not the FCF that was actually used to create the map. []

IPREF = NDF (Read)

A 2D NDF holding a map of total intensity within the sky area covered by the input POL2 data, in units of pW, mJy/beam, Jy/beam, mJy/arcsec**2, Jy/arcsec**2 (" ^" may be used in place of " **"). If supplied, the returned Q and U maps will be corrected for instrumental polarisation, based on the total intensity values in IPREF. The supplied IPREF map need not be pre-aligned with the output Q and U maps - it will be resampled as necessary using a transformation derived from its WCS information. The total intensity values in this map are also used to calculate the percentage polarisation values stored in the output vector catalogue specified by parameter CAT. [!]

JY = _LOGICAL (Read)

If TRUE, the output catalogue, and the output maps specified by parameters " Q" , " U" and " PI" , will be in units of Jy/beam. Otherwise they will be in units of pW (in this case, the I values in the output catalogue will be scaled to take account of the different FCFs for POL-2 and non-POL-2 observations). Note, the Q and U maps made from individual observations (see parameter MAPDIR) are always in units of pW. [True]

LOGFILE = LITERAL (Read)

The name of the log file to create if GLEVEL is not NONE. The default is " <command>.log" , where <command> is the name of the executing script (minus any trailing " .py" suffix), and will be created in the current directory. Any file with the same name is over-written. The script can change the logfile if necessary by assign the new log file path to the module variable " starutil.logfile" . Any old log file will be closed befopre the new one is opened. []

MAPDIR = LITTERAL (Read)

The name of a directory in which to put the Q and U maps made from each individual observation supplied via " IN" , before coadding them (the QMAP and UMAP parameters specify the final coadded Q and U maps). If null is supplied, they are

placed in the same temporary directory as all the other intermediate files and so will be deleted when the scrip exists (unless parameter RETAIN is set TRUE). Note, these maps are always in units of pW. [!]

MSG_FILTER = LITERAL (Read)
Controls the default level of information reported by Starlink atasks invoked within the executing script. This default can be over-ridden by including a value for the msg_filter parameter within the command string passed to the " invoke" function. The accepted values are the list defined in SUN/104 (" None" , " Quiet" , " Normal" , " Verbose" , etc). [" Normal"]

NORTH = LITERAL (Read)
Specifies the celestial coordinate system to use as the reference direction in any newly created Q and U time series files. For instance if NORTH=" AZEL" , then they use the elevation axis as the reference direction, and if " ICRS" is supplied, they use the ICRS Declination axis. If " TRACKING" is supplied, they use north in the tracking system - what ever that may be. [" TRACKING"]

OBSTABLE = LITERAL (Read)
The path of a new text file to create, to which will be written statistics describined the Q and U maps for each individual observation present in the list of files specified by parameter IN. No file is created if null (!) is supplied. The values are written in the form of a TOPCAT " ascii" table, with one row for each observation. The columns are:

- UT: UT date of observation
- OBS: Observation number
- SUBSCAN: The first subscan included in the map
- WVM: The mean of the starting and ending WVM tau values
- NEFD_Q: The measured NEFD in the Q map (mJy.sec^(0.5))
- NEFD_U: The measured NEFD in the U map (mJy.sec^(0.5))
- NEFD_EXP: The expected NEFD based on WVM and elevation (mJy.sec^(0.5))
- TIME: The elapsed time of the data included in the maps (s)
- SIZE_Q: The total area of the source regions in the Q map (square arc-mins)
- SIZE_U: The total area of the source regions in the U map (square arc-mins)
- RMS_Q: The RMS Q value within the source regions (pW)
- RMS_U: The RMS U value within the source regions (pW)
- NBOLO_Q: Number of bolometers contributing to Q map
- NBOLO_U: Number of bolometers contributing to U map
- DX: Pointing correction in azimuth (arc-sec)
- DY: Pointing correction in elevation (arc-sec) The last two columns (DX and DY) are only created if parameter ALIGN is TRUE. [!]

PI = NDF (Read)
An output NDF in which to return the polarised intensity map. No polarised intensity map will be created if null (!) is supplied. [!]

PIXSIZE = _REAL (Read)
Pixel dimensions in the output Q and U maps, in arcsec. The default is 4 arc-sec for 850 um data and 2 arc-sec for 450 um data. []

Q = NDF (Read)
The output NDF in which to return the total Q intensity map including all supplied observations.

QUDIR = LITTERAL (Read)
The name of a directory in which to put the Q, U and I time series generated by SMURF:CALCQU. If null (!) is supplied, they are placed in the same temporary direcory as all the other intermediate files. [!]

U = NDF (Read)
The output NDF in which to return the total U intensity map including all supplied observations.

REF = NDF (Read)
An optional map defining the pixel grid for the output maps. If no value is specified for REF on the command line, it defaults to the value supplied for parameter IPREF. See also parameter ALIGN. []

RETAIN = _LOGICAL (Read)
Should the temporary directory containing the intermediate files created by this script be retained? If not, it will be deleted before the script exits. If retained, a message will be displayed at the end specifying the path to the directory. [FALSE]

219

E Other Scripts

ACSIS_INDEX
Provide a listing of ACSIS data files in a directory

Description:

Reads ACSIS observation files in a directory and creates an index. Standard JCMT directory layout is supported.

Usage:

```
acsis_index [-v | -h | -d | -o] [-a] [-c] [-e] [-f] [-t/T -r/R -s/S] [datadir]
```

Parameters:

−**all**
Print information on all files even if the headers are identical. For example, include multiple subbands (subsystems) and sub-scans.

−**debug**
Print debugging information.

−**extended**
Print extended lines. Can add AZ, EL, pointing offsets and bandwidth.

−**force**
Force a subdirectory search. If no files are found in the directory itself, as would be the case for a raw data tree, the program will automatically search subdirecties. This option forces the subdirectory search.

−**help**
Print help information.

−**version**
Print version information.

−**man**
Print the full documentation to STDOUT.

−**ocscfg**
Print the OCS configuration XML file.

−**cal**
Only print calibration observations.

−**skip**
Skip observation information when printing Tsys or Trx

−**stdev**
Include the standard deviation when printing Tsys or Trx.

−**trx**
Print median receiver temperature.

−**tsys**
Print median system temperature.

Notes:

The primary argument is the data directory to index. If a `yyyymmdd` string is given it will default to the ACSIS data for the specified data unless there is a local subdirectory named `yyyymmdd`. Will default to the value of the $DATADIR environment variable if set.

Related Applications :

SMURF: SCUBA2_INDEX, GETTSYS

DUMPOCSCFG
Retrieve OCS configuration XML from data file.

Description:

Searches for the OCS configuration in the file (if present) and writes it to standard output. If the configuration is not present in the file (older data will not contain it) the OCS configuration name is retrieved and an attempt made to locate the file on the file system (only valid at JAC).

Usage:

```
dumpocscfg a20070105_00050_01_0001.sdf
```

Parameters:

−**help**
: Print help information.

−**version**
: Print version information.

−**man**
: Print the full documentation to STDOUT.

GETTSYS
Get system temperature information from ACSIS data

Description:

Simple program to list TSYS or TRX information for receptors from an ACSIS data file. Optionally, calculates statistics.

Usage:

```
gettsys -statistics a20070105_00050_01_0001.sdf

gettsys -trx -receptor h10 a20070105_00050_01_0001.sdf
```

Parameters:

−**help**
Print help information.

−**version**
Print version information.

−**man**
Print the full documentation to STDOUT.

−**trx**
Lists all TRX values for each receptor instead of TSYS.

−**receptor**
Only report information for the current receptor.

−**statistics**
Report statistics in addition to the values. For tabular form, the statistics are included at the end of the table as additional lines of median, mean and standard deviation.

JCMTSTATE2CAT
convert JCMT state structure into TST format

Description:

Reads a set of SCUBA-2 or ACSIS files and writes a catalogue of the state information to standard out. The output file is in TST format and can be read into the TOPCAT application (but may require that TOPCAT is told explicitly that the catalogue is in TST format, e.g. with the "-f tst" command line option).

This information includes the telescope pointing position (Actual, Demand and Base) in both the tracking system and AZEL coordinate frames, jiggle patterns, telescope row/offset index amongst others.

Usage:

```
jcmtstate2cat *.sdf > catalogue.tst

topcat -f tst catalogue.tst
```

Parameters:

−**help**
Print help information.

−**version**
Print version information.

−**man**
Print the full documentation to STDOUT.

−−**with−mce**
Include SCUBA-2 MCE information in output table. This adds a lot of data to the output files and much of it is constant for the observation. Default is not to add this option. This option only works for data prior to summer of 2010 (which essentially means S2SRO data). After that date MCE data is written in a different form and is not supported. For modern data use the separate MCEHEAD2CAT command.

Notes:

Additional derived columns are included in addition to those stored directly in the JCMTSTATE extension. All telescope values include a correction for motion of the secondary mirror.

- `RA/DEC`
  Tracking coordinates in degrees. The columns will have the same name even if the telescope was tracking in GALACTIC.
- `DRA/DDEC`
  Tracking offsets from the base position in arcsec.
- `AZ/EL`
  Azimuth and elevation in degrees. Calculated directly from JCMTSTATE without being converted to RA/Dec.

- `DAZ/DEL`
  Azimuth and elevation offsets from the base position in arcsec.
- `TELSPEED`
  Instantaneous telescope speed in arcsec/sec.
- `WVMTAU`
  Zenith 225 GHz opacity derived from the water vapor radiometer.
- `PWVZEN`
  Precipitable water vapor at the zenith in mm.
- `PWVLOS`
  Precipitable water vaport in the line of sight in mm.

Note that information from the ACSIS extension is not included at this time. This is partly because this extension can change in shape between observations. MCE header information is included for SCUBA-2 data files.

MCEHEAD2CAT
Convert SCUBA-2 MCE header information to TST format

Description:

Reads a set of SCUBA-2 files and writes catalogues of the mcehead information to files. The output files are in TST format and can be read into the TOPCAT application (but may require that TOPCAT is told explicitly that the catalogue is in TST format, e.g. with the `-f tst` command line option).

Multiple output files are created since the data are stored with different dimensionality. The *root* name for the output files can be specified using the `−out` parameter (default: uses the `s??ddmmyyyy_nnnnn` part of the input file name or the basename of the filename if different).

Output files (suffix: `tst`) are as follows:

- `_gen`:
  parameters that have a single value (includes the 10 values for `psc_status_psc` which don't fit the scheme)
- `_crd`:
  parameters that have 4 values
- `_col`:
  parameters that have 32 values (includes parameters that have 8 values since they always come in sets of 4).
- `_row`:
  parameters that have 41 values

If `−out` is used to specify the root output file name, out from all input files will be concatinated into a single set of output files.

Usage:

```
mcehead2cat *.sdf

mcehead2cat *.sdf -out="outroot" -item=par1,par2

mcehead2cat *.sdf -item=l̂ist.in
```

Parameters:

−help
Print help information.

−version
Print version information.

−man
Print the full documentation to STDOUT.

−out
Optional root name for the output files.

−items
Optional comma-separated list of header items to print. Instead a filename listing the item names can be given as `-item=f̂ile`. The format of the file is the desired list of item names with one item per line.

Notes:

For SCUBA-2 raw files, the program will only process the first file of any observation and skip all others. This is because the MCE configuration is only updated at the start of an observation. Assuming there are 10 observations for the night:

```
mcehead2cat /jcmtdata/raw/scuba2/s4a/20110524/*/*.sdf
```

will only process 10 files, not all x-nr sdf files that may be there. It will produce 10 sets of output files unless `−out` has been specified, in which case the mceheaders from the 10 files will be written to a single set of output files.

SCUBA2_INDEX
Provide a listing of SCUBA-2 data files in a directory

Description:

Reads ACSIS observation files in a directory and creates an index. Standard JCMT directory layout is supported.

Usage:

```
scuba2_index [-v | -h | -d | -o] [-a] [-c] [-e] [-f] [datadir] [scuba2_array]
```

Parameters:

−**all**
Print information on all files even if the headers are identical. For example, include multiple subbands (subsystems) and sub-scans.

−**debug**
Print debugging information.

−**extended**
Print extended lines. Can add AZ, EL, pointing offsets and bandwidth.

−**force**
Force a subdirectory search. If no files are found in the directory itself, as would be the case for a raw data tree, the program will automatically search subdirecties. This option forces the subdirectory search.

−**help**
Print help information.

−**version**
Print version information.

−**man**
Print the full documentation to STDOUT.

−**ocscfg**
Print the OCS configuration XML file.

−**cal**
Only print calibration observations.

Notes:

The primary argument is the data directory to index. If a `yyyymmdd` string is given it will default to the SCUBA-2 data for the specified data unless there is a local subdirectory named `yyyymmdd`. Will default to the value of the $DATADIR environment variable if set.

This command can take a specific sub-array directory to index. Defaults to s8d.

Related Applications :

SMURF: ACSIS_INDEX, GETTSYS

SMAS
Short Map Analysis Script

Description:

With the exception of Moon maps, each NDF is processed in the same way. The KAPPA:PSF command is used to locate the accurate source centre and fit it using an elliptical Gaussian-like model. The details of the model are then appended to an output text file as a new row.

The resulting output catalogue has a row for every input NDF, and contains the following columns:

- `TAI`: The MJD (in the TAI timescale) associated with the NDF
- `UTC`: The UTC date and time string associated with the NDF
- `SEQSTART` - RTS index number of first frame
- `SEQEND` - RTS index number of last frame
- `DLON`: The longitude offset at the centre of the source, in arc-seconds
- `DLAT`: The latitude offset at the centre of the source, in arc-seconds
- `AMP`: The peak value in the source
- `SUM`: The total data sum in the source. This is the integral of the Gaussian-like model calculated by KAPPA:PSF
- `FWHM`: The seeing-disc size: the full width at half maximum across the minor axis of the source (arc-seconds)
- `AXISR`: The axis ratio of the source: the ratio of the major axis length to that of the minor axis.
- `GAMMA`: The radial fall-off parameter of the source. A gamma of two would be a Gaussian.
- `ORIENT`: The position angle (measured from north through east) of the major axis of the source, in degrees.
- `NDF`: The path to the NDF
- `ROW`: The row number for the bolometer (if known)
- `COL`: The column number for the bolometer (if known)
- `ARRAY`: The array name for the bolometer (if known)

For comparison, a set of extra columns is appended to these that give the source properties as calculated by KAPPA:BEAMFIT rather than KAPPA:PSF. These columns have the same names, but are prefixed by the letter B. Note, BEAMFIT assumes a gamma value of 2.0 (i.e. a pure Gaussian) and so there is no `BGAMMA` column in the output catalogue.

The `SEQSTART` and `SEQEND` columns will be set to 'null' if the supplied map does not contain these FITS headers.

If the supplied data is for the Moon, then all the above is ignored and the output contains the following columns:

- SUM: The integrated value within a 35 arc-minute aperture centred on (0,0)
- ROW: The row number for the bolometer (if known)
- COL: The column number for the bolometer (if known)
- ARRAY: The array name for the bolometer (if known)

Usage:

```
smas [-a diam] [-b] [-e error] [-r perc] [-p] <in-list> <out-table> [<out-image>]
```

Parameters:

−a

Specifies that the catalogue SUM column values (the total data sum) should be the integrated data value within a circular aperture centred on the source peak position, with radius specified by "diam" (a numerical value in arc-seconds). This is calculated using KAPPA:APERADD. If this option is not supplied, the SUM value is the integrated value under the model source calculated by KAPPA:PSF. If "diam" is not supplied, defaults of 30 and 15 are used for 850 and 450 μm data respectively.

−b

Specifies that the catalogue should contain only ROW, COL and AMP only. Using this option also implies the "−p" option. This reduces the run-time for script significantly.

−e

Specifies the maximum allowed error (in arc-seconds) between the source position as determined by KAPPA:BEAMFIT and KAPPA:PSF. If the discrepancy in X or Y is greater than this value, then the shortmap or bolomap is not included in the returned catalogue or image. If the −e option is unspecified, then no sources are rejected. If no "error" value is supplied, a default of 1 arc-second is used.

−p

Specifies that the catalogue AMP column values (the peak value in the source) should be the maximum data value within a box of size 20 pixels centred on the expected source position. If this option is not supplied, the AMP value is the peak value of the model source calculated by KAPPA:PSF.

−r

The following number ("perc") specifies the minimum percentage of good pixel values in a small box around the expected source position. Sources that have insufficient good pixels in the box are reported and then ignored. If the "-r" option is not supplied (or is supplied without a "perc" value), a default of 95 is used.

<**in-list**>

The path to an NDF holding a SCUBA-2 map of a point source (or Moon). Alternativelty, a text file containing the paths to one or more such NDFs can be supplied. If an NDF contains a MORE.SMURF.SHORTMAPS or MORE.SMURF.BOLOMAPS extension item, then the NDFs in the SHORTMAPS or BOLOMAPS array are used in place of the supplied NDF (SHORTMAPS is used in preference to BOLOMAPS if both are present). *Note*, all the data must be for the same wavelength.

<**out-table**>
The name of a text file in which to store a catalogue containing details of the source in each supplied map, as described below. This catalogue can be displayed and analysed using `topcat -f ascii <out-table>`.

<**out-image**>
If supplied, and if the input data contains bolomaps, a 2D NDF containing the `SUM` (or `AMP` if the "−b" option is used) values for each bolometer is created with the name given by <out-image>. The FITS headers from the first input NDF are copied to the output NDF. If the supplied data contains values from more than one sub-array, the name of the sub-array is appended the end of each output NDF name.

F Configuration Parameters

These pages describe the values that can be included in the configuration supplied via the CONFIG parameter when running a SMURFcommand.

- 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.

ANG0
Specifies the start of each fitting box

Description:

The half-wave plate position at which each fitting box begins, in degrees. [90]

SMURF Usage :

CALCQU

ANGROT
Specifies the orientation of the fixed analyser

Description:

The angle from the focal plane X axis to the fixed analyser, in degrees. Measured positive in the same sense as rotation from focal plane X to focal plane Y. [90]

SMURF Usage :

CALCQU

APOD
Control the removal of low frequencies from the time-stream data

Description:

The number of samples to apodise at each end of the data stream prior to smoothing as part of the initial cleaning before the iterative algorithm begins. If <undef>, a default value is used equal to 1/(steptime∗freq) where freq is the lowest edge or notch frequency. Note, apodisation is only performed if parameter "zeropad" is set non-zero. [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

AST.FILT_DIFF
Reduces spurious large scale structure in the final map - an experimental alternative or addition to AST masking

Description:

If ast.filt_diff is non-zero, the map created at the end of each iteration is modified by removing any large scale structures that have been introduced by the iteration. Such structures are a natural outcome of the iterative process and often take the form of ripples on the scale of the FLT filter size that get stronger with each iteration.

If " ast.filt_diff" is non-zero it gives the size of the largest features to be retained, in arcsconds (a value equal to half of the FLT filter size seems to be a good starting point). At the end of each iteration the difference is taken between the new map and the previous map. This difference is filtered using a sharp-edged high-pass filter, of size specified by " ast.filt_diff" . The remaining high frequencies (i.e. small structures that have been introduced by the new iteration) are added back onto the map created by the previous iteration. This modified map is then used, instead of the original map, to define the AST model. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

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.MAPSPIKE_FREEZE
Indicates when to stop flagging extra spikes in the map

Description:

If ast.mapspike_freeze is non-zero, then the map-based de-spiking process (see parameter "ast.mapspike") is only applied for a limited number of iterations. No further spikes are then flagged, but spikes flagged on previous iterations remain flagged. The limit may be specified either as an integer number of iterations (in which case the supplied value is incremented by the value of parameter "ast.skip"), or as a normalised map-change value in the range 0.0 to 1.0 (not including the limits). In this case, the check on the normalised map-change will not be performed until the iterations specified by parameter "ast.skip" have been performed. If zero is supplied, spikes are flagged on all iterations. [0.0]

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_ACCUM
Prevents unstable SNR masks halting convergence

Description:

Masks defined by SNR limits can show instabilities in which pixels can repeatedly enter the mask on one iteration and leave it on the next. Such oscillating pixels tend to occur around the edges of source areas in the mask, and can prevent convergence of the iterative map-making process. Setting AST.ZERO_ACCUM to a non-zero value can prevent this by forcing source pixels to be accumulated rather than replaced on successive iterations. Thus, if AST.ZERO_ACCUM is non-zero, any map pixel which is flagged as a source pixel on some iteration will never be unflagged - once a source pixel, always a souce pixel. [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_MASK0
Reduces spurious large scale structure in the final map within fixed regions specified by an external mask

Description:

This parameter allows an alternative mask to be used on the first iteration. If a value is supplied for " ast.zero_mask0" , it will be used in place of parameter "ast.zero_mask" on the first iteration only. If no value is supplied for " ast.zero_mask0" , the value supplied for " ast.zero_mask" will be used on all iterations including the first. Setting " ast.zero_mask0" to zero will cause no masking to be used on the first iteration. [<undef>]

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_HIPASS
Flatten map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "ast.zero_snr" to be based on a copy of the SNR map that has been high-pass filtering to remove structures larger than the number of arc-seconds given by ast.zero_snr_hipass. This will in general reduce the number of source pixels in the mask. See also parameter "ast.zero_snr_ffclean" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

AST.ZERO_SNR_LOPASS
Smooths map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "ast.zero_snr" to be based on a copy of the SNR map that has been low-pass filtering (i.e smoothed) to remove noise and other features smaller than the number of map pixels given by ast.zero_snr_lopass. This may help to prevent pixels oscillating in and out of the mask on successive iterations, and thus aid smoother convergence. [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 un-masked " 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

CHITOL
Specifies when to stop iterating

Description:

If the difference in reduced chi^2 between subsequent iterations falls below the value of chitol, then the map-maker performs one more iteration and then terminates. Only used if parameter "numiter" is negative. Parameter "maptol" provides an alternative (and usually better) termination criterion. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

CHUNKFACTOR
Specifies the relative calibration of each continuous chunk

Description:

This parameter is mainly for use when combining data from multiple observations within a single invocation of makemap (such as happens when makemap is invoked from within the skyloop script). It specifies a calibration correction factor to use for each chunk of continuous time series data that goes into the final map. The chunk's time-stream data is multiplied by this factor whenever it is binned into a map, and the inverse correction is applied when a map is sampled to form simulated time-stream data for the chunk. This allows multiple observations, which may have different FCFs, to be normalised so that they all use the same calibration. The factors can be specified in two ways: 1) The " chunkfactor" 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 factor for that chunk. 2) A comma-separated list of explicit numerical values can be supplied. The list must be enclosed in parentheses. 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 factor of 1.0 to be used for every chunk. [1.0]

SMURF Usage :

MAKEMAP, CALCQU

CHUNKWEIGHT
Specifies a relative weight for each continuous chunk

Description:

This parameter specifies the weight to use for each chunk of continuous time series data that goes into the final map. These weights are used when the maps made from the individual chunks are combined together to form the total map. They can be specified in two ways: 1) The " chunkweight" 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 weight for that chunk. 2) A comma-separated list of explicit numerical values can be supplied. The list must be enclosed in parentheses. 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 weight of 1.0 to be used for every chunk. [1.0]

SMURF Usage :

MAKEMAP, CALCQU

CLEANDK.<XXX>
Controls the cleaning of the dark squid signals

Description:

The cleaning of dark squid signals is controlled by a set of parameters that are directly analagous to those that control the cleaning of the main bolometer time series. They use the same names, except that they are prefixed by " cleandk." . So for instance, " apod" becomes " cleandk.apod" , " fillgaps" becomes " cleandk.fillgaps" < etc. [?]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.BOXCAR
Controls the estimate of the common-mode signal

Description:

Specifies a low-pass boxcar filter on COM to assist with convergence. Positive values for com.boxcar are interpreted as a number of samples, and negative values as a number of seconds (converted to samples using the downsampled sample rate). Note, this parameter can only be used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.BOXCARD
Controls the estimate of the common-mode signal

Description:

This records the size of the COM boxcar filter on the previous iteration and should not be set by the user (see parameter "com.boxcar" and parameter "com.boxfact"). Note, this parameter is only used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.BOXFACT
Controls the estimate of the common-mode signal

Description:

If com.boxfact is non-zero, reduce width of com.boxcar by this factor each iteration. Note, this parameter is only used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.BOXMIN
Controls the estimate of the common-mode signal

Description:

Specifies a minimum width below which the com boxcar filter can't be reduced. See parameter "com.boxfact" and parameter "com.boxcar" . Note, this parameter is only used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, 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.FILL
Controls the estimate of the common-mode signal

Description:

If com.fill is non-zero, then any holes in each time slice left by the previous flagging of bolometer values will be filled with values representative of the neighbouring bolometers before finding the mean value. This avoids bias in the common-mode caused by areas of systematically high or low bolometer values being flagged. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.FIT_BOX
Controls the estimate of the common-mode signal

Description:

The number of adjacent time slices (or seconds if negative) that are to be used when fitting the bolometer data to the common-mode template. The value of com.fit_box should be no smaller than the value of parameter "com.gain_box" . If it is not supplied, it defaults to the value of com.gain_box. [<undef>]

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_IS_ONE
Controls the estimate of the common-mode signal

Description:

Setting com.gain_is_one non-zero causes all bolometer gains to be forced to 1.0. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.GAIN_POSITIVE
Controls the estimate of the common-mode signal

Description:

By default negative gains are used to flag bad bolometer data. However, if this parameter is set to 0 negative values will be allowed [1]

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.NITER
Controls the estimate of the common-mode signal

Description:

The number of n-sigma clipping iterations (a value of 1 implies no clipping). Only used by the new COM algorithm (see parameter "com.oldalg"). [1]

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.NOREMOVE
Controls the estimate of the common-mode signal

Description:

If non-zero, the common-mode will be estimated while iteratively flagging and removing outlier detectors as usual. However, once the flagging is completed, the common-mode will not actually be removed from the time-series (and if the model is exported, it will be set to 0). Note, this parameter is only used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.NOTFIRST
Controls the estimate of the common-mode signal

Description:

If non-zero, delay calculation of COM until after the first iteration (good if the astronomical signal is expected to dominate the sky signal). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.NSIGMA
Controls the estimate of the common-mode signal

Description:

The number of standard deviations at which the n-sigma clipping algorithm clips. Only used by the new COM algorithm (see parameter "com.oldalg"). [3]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.OFFSET_IS_ZERO
Controls the estimate of the common-mode signal

Description:

Setting non-zero causes all bolometer offsets to be forced to 0.0. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.OLDALG
Controls the estimate of the common-mode signal

Description:

If non-zero, us the pre 15-MAY-2012 algorithm for estimating COM (the " old" algorithm). The new algorithm in general provides faster and smoother convergence. It uses the sigma-clipped weighted mean of all bolometer values at each time slice as the common-mode signal. See parameter "com.niter" and parameter "com.nsigma" . [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.PERARRAY_LAST
Controls the estimate of the common-mode signal on the last iteration

Description:

This is the value to be used for parameter "com.perarray" on the last iteration. If it is "<undef>", then the same value will be used as for earlier iterations. [<undef>]

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.SIG_WING
Controls the rejection of time slices with inconsistent common-modes

Description:

This value is only used if parameter "com.sig_limit" is set to a non-zero value. Each contiguous block of time slices flagged by the consistency filter is extended at its start and end by an extra block of samples equal in length to " com.sig_wing" times the FLT filter size. [1.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.WEIGHT
Controls the estimate of the common-mode signal

Description:

If non-zero, the bolometer data is weighted when it is combined to form the common mode signal. Each weight is the reciprocal of the assocoiated bolometer noise estimate. If the data has no noise estimates, the common-mode is formed from the unweighted mean of the bolometer data. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

COM.ZERO_ACCUM
Prevents unstable SNR masks halting convergence

Description:

Masks defined by SNR limits can show instabilities in which pixels can repeatedly enter the mask on one iteration and leave it on the next. Such oscillating pixels tend to occur around the edges of source areas in the mask, and can prevent convergence of the iterative map-making process. Setting COM.ZERO_ACCUM to a non-zero value can prevent this by forcing source pixels to be accumulated rather than replaced on successive iterations. Thus, if COM.ZERO_ACCUM is non-zero, any map pixel which is flagged as a source pixel on some iteration will never be unflagged - once a source pixel, always a souce pixel. [0]

SMURF Usage :

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_MASK0
Reduces spurious large scale structure in the final map within fixed regions specified by an external mask

Description:

This parameter allows an alternative mask to be used on the first iteration. If a value is supplied for " com.zero_mask0" , it will be used in place of parameter "com.zero_mask" on the first iteration only. If no value is supplied for " com.zero_mask0" , the value supplied for " com.zero_mask" will be used on all iterations including the first. Setting " com.zero_mask0" to zero will cause no masking to be used on the first iteration. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_NITER
Allows COM masking to be switched off after a given number of iterations

Description:

If com.zero_niter is non-zero, it gives the number of iterations for which the COM model should be masked. Subsequent iterations are not masked. A value of zero means " mask on all iterations" . However, if parameter "com.zero_notlast" is set, the mask will will not be applied on the last iteration, even if com.zero_niter is zero. Note, 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 COM model. [0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_NOTLAST
Prevent COM masking being performed on the last iteration

Description:

If com.zero_notlast is 1, then the COM model is not masked on the final iteration. This means that data samples that fall inside the masked areas are include in the estimate of the common-mode signal. This should make little difference since the astronomical signal should have been removed by then. [1]

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_SNR_HIPASS
FLatten map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "com.zero_snr" to be based on a copy of the SNR map that has been high-pass filtering to remove structures larger than the number of arc-seconds given by com.zero_snr_hipass. This will in general reduce the number of source pixels in the mask. See also parameter "com.zero_snr_ffclean" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

COM.ZERO_SNR_LOPASS
Smooths map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "com.zero_snr" to be based on a copy of the SNR map that has been low-pass filtering (i.e smoothed) to remove noise and other features smaller than the number of map pixels given by com.zero_snr_lopass. This may help to prevent pixels oscillating in and out of the mask on successive iterations, and thus aid smoother convergence. [0.0]

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

CYCLEMAP
Produce cycle-folded maps to visualize periodic signals

Description:

If defined, an extension called .MORE.SMURF.CYCLEMAPS is added to the output map NDF. This parameter should consist of two or three comma-separated values in parentheses. These are the number of maps desired, the cycle period (in seconds) and (optionally) the cycle start time (as TAI MJD, otherwise the start of this TAI MJD). [<undef>]

SMURF Usage :

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

DECONVMCE
Diagnostic tool for checking effects of MCE filter

Description:

If non-zero, each bolometer time stream is deconvolved to remove the effects of the MCE anti-aliasing filter. See also parameter "fakemce" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DELAY
A diagnostic tool to check for time delays in bolometer data streams

Description:

A delay, in seconds, to apply to each time stream. This results in each bolometer sample being associated with a different position on the sky. [0.0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DIAG.APPEND
Qualifies the required dignostic information

Description:

If non-zero, it indicates that diagnostic info should be appended to the container file specified by parameter "diag.out" , which should already exist. If zero, then any existing container file is first deleted before storing new diagnostics information in it. Only used if a value is supplied for diag.out. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.ARRAY
Qualifies the required dignostic information

Description:

The name of the array (S8A, S8B, etc) containing the data to be written out. If <undef>, the first available array is used. Only used if a value is supplied for parameter "diag.out" . [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.ASTHITS
Qualifies the required dignostic information

Description:

If non-zero, the AST model time-stream values dumped to the file specified by parameter "diag.out" will be the corresponding hits value rather than the actual AST data values. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.BOLO
Qualifies the required dignostic information

Description:

Indicates the bolometer for which diagnostic information is required (the sub-array is indicated by diag.array). Only used if a value is supplied for parameter "diag.out" . It can be:

- A pair of integers, separated by a comma, contained in parentheses, giving the column and row of the bolometer. The first integer should be in the range 0 to 31, and the second should be in the range 0 to 39.
- A single integer in the range 0 to 1279.
- The string " MEAN" (case insensitive), in which case all unflagged data from all good bolometers is averaged to form the time-stream to dump.
- The string " WMEAN" (case insensitive), in which case all unflagged data from all good bolometers is averaged using weights derived from the bolometer noise estimates to form the time-stream to dump.
- The string " TYPICAL" (case insensitive), in which case a bolometer with typical noise charactersitics is chosen and used. The index of the chosen bolometer is reported, and stored in the dumped NDFs.
- The string " NONE" , in which case the output NDFs describing a single bolometer are not produced (maps and cubes may still be produced though - see parameter "diag.map" and parameter "diag.cube"). [WMEAN]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.BTABLE
Qualifies the required dignostic information

Description:

If supplied, a set of text files is created, each containing a table in TOPCAT " ascii" format with columns that hold the data values in a single bolometer (see parameter "diag.bolo"). Each row corresponds to a single time slice. The columns are:

- ITIME: The time slice index
- RTS_NUM: The RTS_NUM value of the time slice
- IX: The X pixel index of the map pixel into which the bolometer sample falls.
- IY: The Y pixel index of the map pixel into which the bolometer sample falls.
- DATA: The Data value of the bolometer sample.
- VAR: The Variance of the bolometer sample (if available).
- QUAL: The Quality of the bolometer sample (if available).

Each file has a name of the format " <name>_<chunk>_<iter>_<mod>_<when>.asc" where:

- <name> is the value of the parameter diag.btable.
- <chunk> is the chunk index
- <iter> is the iteration number
- <mod> is the model name (COM, FLT, PCA, etc)
- <when> is " BEF" , " AFT" or " MOD" , indicating when the bolometer values were dumped. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.CLEANED
Qualifies the required dignostic information

Description:

If non-zero, the cleaned data at the start of the first iteration is dumped. The data is stored in the same way as the models specified by parameter "diag.models" using the psuedo-model name " CLN" . Note, if diag.cleaned is non-zero, the initial cleaned data is dumped even if parameter "diag.lastonly" is non-zero. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.CUBE
Qualifies the required dignostic information

Description:

If non-zero, create 3D cubes containing the required residuals or models for all bolometers at each iteration. Warning - this can be a huge amount of data so only use on very short time series (e.g. a single subscan) and for very few iterations. The cubes have names of the form " <where>_<chunk>_cube_<iter>" , (see parameter "diag.out"). Only used if a value is supplied for diag.out. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.LASTONLY
Qualifies the required dignostic information

Description:

If non-zero, diagnotics information is dumped only for the last iteration. Otherwise, information is dumped on all iterations. Note, if parameter "diag.cleaned" is non-zero, the initial cleaned data is still dumped even if diag.lastonly is non-zero. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.MAP
Qualifies the required dignostic information

Description:

If non-zero, a 2D map containing the binned time-stream data for all bolometers is created at each iteration, for each required model and set of residuals. These are placed in 2D NDFs with names in the following format: " <where>_<chunk>_map_<it>" , where <chunk> and <where> are described above (under " OUT") and <it> is the iteration number. If the " diag.map" value is positive, the map will contain data for just the subarray specified by " diag.array" . If " diag.map" is negative, the map will contain data for all available subarrays. Only used if a value is supplied for " diag.out" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.MASK
Qualifies the required dignostic information

Description:

If non-zero, then the AST model will be masked using the current AST mask before being dumped. Otherwise, the AST model will not be masked before being dumped. Only used if a value is supplied for parameter "diag.out" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.MINGOOD
Qualifies the required dignostic information

Description:

The minimum fraction of good values in a time stream for which data should be dumped. An error is reported if the required minimum value is not met. Only used if a value is supplied for parameter "diag.out" . [0.2]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.MODELS
Qualifies the required dignostic information

Description:

Indicates the models that are to be written out. It should be a comma separated list of model names (e.g. COM, FLT, AST, RES, etc) contained within parentheses, or a single model name. A model name of RES here refers to the residuals after subtraction of all models in use (typically COM, FLT and AST). The residuals can also be written out at other times - see diag.res_before and diag.res_after. Only used if a value is supplied for parameter "diag.out" . [(com,ext,flt,ast,res)]

SMURF Usage :

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

DIAG.POWER
Qualifies the required dignostic information

Description:

If non-zero, write out the power spectrum for each selected model. Only used if a value is supplied for parameter "diag.out" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.QUAL
Qualifies the required dignostic information

Description:

If zero, no Quality arrays will be stored in the diagnostic NDFs. Instead, flagged Data values will be set to the starlink bad value. If non-zero, each NDF holding diagnostic information will include a Quality array, and flagged Data values will retain their internal values (i.e. they will not be set to the starlink bad value). This makes the NDF seriously bigger. [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.RES_AFTER
Qualifies the required dignostic information

Description:

If non-zero, then in addition to writing out the requested models, the residuals are also written out immediately after each requested model is subtracted. Only used if a value is supplied for parameter "diag.out" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.RES_BEFORE
Qualifies the required dignostic information

Description:

If non-zero, then in addition to writing out the requested models, the residuals are also written out immediately before each requested model is subtracted. Only used if a value is supplied for parameter "diag.out" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.TABLE
Qualifies the required dignostic information

Description:

If supplied, a text file is created with the given name containing a table with columns that hold the bolometer and time slice index (zero-based) of each sample that falls within the map pixel given by parameter "diag.xpix" and parameter "diag.ypix" . The table also holds columns for the other quantities requested by the other " diag..." parameters. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.TIME
Qualifies the required dignostic information

Description:

If non-zero, write out the time-series for each selected model. Only used if a value is supplied for parameter "diag.out" . [1]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.XPIX
Qualifies the required dignostic information

Description:

The X pixel index of the map pixel that is to be recorded in the table specified by parameter "diag.table" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DIAG.YPIX
Qualifies the required dignostic information

Description:

The Y pixel index of the map pixel that is to be recorded in the table specified by parameter "diag.table" . [0]

SMURF Usage :

MAKEMAP, CALCQU

DKCLEAN
Control dark squid cleaning

Description:

If non-zero, clean the dark squid signals. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

DKS.REPLACEBAD
Controls the DKS model

Description:

If non-zero, replace dead dark squids with average of working dark squids. [0]

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

DOWNSAMPFREQ
Controls data down-sampling

Description:

Allows the down-sampled frequency to be specified directly in Hz. See parameter "downsampscale" . [0]

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

DUMPDIR
Directory for NDFs containing exported models etc

Description:

This parameter specifies an existing directory in which to place NDFs created by configuration parameters such as " exportndf" , " noi.export" , " exportclean" etc. It is also the directory from which NDFs are read by parameters such as " noi.import" and " ext.import" . The current directory is used if " dumpdir" is undefined. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

ENSUREFLAT
Controls flat-fielding of supplied time-series data

Description:

Apply the flatfield if loading raw data? Setting this to zero will produce a map in raw DAC units (and you must also supply raw - i.e. unflat-fielded - data). [1]

SMURF Usage :

MAKEMAP, CALCQU

EPSIN
Aid convergence and image flatness

Description:

In some cases, each iteration can add a fixed constant " error map" into the astronomical map, which accumulates with each subsequent iteration, preventing convergence and causing strong large-scale structures to appear in the final map (typically ripples on the scale of the FLT filter). This parameter attempts to counteract this by allowing a map to be specified which will be subtracted from the astronomical map at the end of each iteration. At present, the dimensions of this map must be identical to those of the real map. See also parameter "epsout" . [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

EPSOUT
Aid convergence and image flatness

Description:

If a value is supplied for this parameter, an output NDF will be created holding the median of the difference maps created on the three final iterations (a difference map is the difference between the astronoimical maps created on two succesive iterations). This may be suitable for feeding back in using parameter "epsin" on a subsequent run of makemap. Note, currently this parameter may only be used if the data is processed in a single continuous chunk (an error is reported if this is not the case). [<undef>]

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

EXPORTLONLAT
Export sample sky positions for use with other software

Description:

If non-zero, the logitude/latitude of every sample will be exported into a pair of NDFs. The file names will be the same as model components, except with sufficies of " lon" and " lat" . [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

EXPORTQBITS
Specifies the quality bits to be written out

Description:

If the value of parameter "exportndf" includes" RES" or " QUA" , then the final quality array is exported to an NDF. An NDF can only contain 8 quality flags, but makemap can sometimes use more than 8 flags internally. The default action in such cases is to combine flags with similar purposes into more general flags in order to reduce the number of used flags to 8 or fewer. This is a lossy process - it is impossible to undo the combination to recover the original flags. However, this can be avoided by using this parameter to specify a list of explicit quality flags to be exported, ensuring that the list contains no more than 8 entries. Any information for flags not in the list will simply be discarded, so choose the flags that you are pf particularly interest to you. If the parameter is left at " <undef>" , all quality flags will be exported, with consequent possible lossy compression. Otherwise the parameter should be a comma-separated list of quality names, enclosed in parentheses. Valid quality names are: BADDA, BADBOL, SPIKE, DCJUMP, PAD, APOD, STAT, COM, FILT, NOISE, EXT, LOWAP, BADEF, RING, SSN, PCA, IP. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

EXT.CSOFIT
Path to file containing the CSO fit parameters

Description:

Full path to file containing the CSO fit parameters. [${STARLINK_DIR}/share/smurf/csofit2.dat]

SMURF Usage :

EXTINCTION, 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.DESPIKEWVM
Controls the extinction values used in the EXT model and the EXTINCTION task

Description:

Indicates if we want WVM despiking. The value is the size in seconds for the despiking window. A value of 0 disables despiking. [30.0]

SMURF Usage :

EXTINCTION, MAKEMAP, CALCQU

EXT.DESPIKEWVMTOL
Controls the extinction values used in the EXT model and the EXTINCTION task

Description:

Indicates tolerance level for WVM despiking. The value is the fractional spike threshold. A value of 0 disables despiking. [0.10]

SMURF Usage :

EXTINCTION, 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.IMPORT
Controls the extinction values used in the EXT model

Description:

If non-zero, import the the extinction values to use in the EXT model from an NDF created by a previous run of makemap. The NDF is expected to have the same name as would be created by setting parameter " exportNDF" to " (ext)" in the makemap configuration. [0]

SMURF Usage :

MAKEMAP, CALCQU

EXT.SMOOTHWVM
Controls the extinction values used in the EXT model and the EXTINCTION task

Description:

Indicates if we want WVM smoothing. The value is the size in seconds for the tophat function. A value of 0 disables smoothing. [10.0]

SMURF Usage :

EXTINCTION, 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

EXT.WVMFIT
Path to file containing the WVM fit parameters

Description:

Full path to file containing the WVM fit parameters. [${STARLINK_DIR}/share/smurf/wvmfit2.dat]

SMURF Usage :

EXTINCTION, MAKEMAP, CALCQU

FAKEDELAY
Diagnostic tool for checking delays in bolometer time streams

Description:

A delay, in seconds, to apply to each fake time stream sampled from the supplied fake map, prior to adding it to the real data. See parameter "delay" . The fakedelay value should usually be equal and opposite to the delay value, so that they cancel out, leaving the fake signal unaffected. See parameter "fakemap" . [0]

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

FAKEMCE
Diagnostic tool for checking effects of MCE filter

Description:

If non-zero, the time-series data generated from the fakemap is smoothed using the MCE response. See parameter "fakemap" . This should normally be set non-zero if parameter "deconvmce" is set non-zero, so that the fakemap signal is left unaffected by the MCE filter. [0]

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

FILLGAPS
Controls handling of missing data within FFT code

Description:

If non-zero, fill vicinity of spikes / DC steps with artificial data. See also parameter "fillgaps_noise" . [1]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FILLGAPS_NOISE
Controls handling of missing data within FFT code

Description:

If non-zero, and if parameter "fillgaps" is non-zero, gaps are filled with a constrained realization of noise. Otherwise they are filled with a noise-free linear function that connects the values before and after the gap. Note, the specific noise values used to fill a specific gap will be different each time the gap is filled. So for instance, this could slow down convergence of the iterative algorithm used by makemap. [1]

SMURF Usage :

SC2CLEAN, 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_EDGEHIGH
Specifies the lowest frequency to be retained by the initial data cleaning

Description:

If non-zero, this is the cut-off frequency of a high 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_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_NOTCHHIGH
Specifies additional frequencies to be removed by the initial data cleaning

Description:

Together with parameter "filt_notchlow" , this parameter specifies a set of 1 or more hard-edge band-cut frequency-domain notch filters, specifying additional frequencies to be removed by the initial data cleaning. The filt_notchhigh value should be a comma-separated list of frequencies in Hz, enclosed in parentheses. These are the frequencies of the upper edges of the notch filters. The filt_notchlow and filt_notchhigh parameters should contain the same number of frequencies. [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

FILT_NOTCHLOW
Specifies additional frequencies to be removed by the initial data cleaning

Description:

Together with parameter "filt_notchhigh" , this parameter specifies a set of 1 or more hard-edge band-cut frequency-domain notch filters, specifying additional frequencies to be removed by the initial data cleaning. The filt_notchlow value should be a comma-separated list of frequencies in Hz, enclosed in parentheses. These are the frequencies of the lower edges of the notch filters. The filt_notchlow and filt_notchhigh parameters should contain the same number of frequencies. [<undef>]

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

FILT_WLIM
Enables an experimental filtering method

Description:

If supplied, this will switch on an experimental method for handling missing or flagged data when filtering the time streams, based on replacing missing data with zero, and then normalising the smoothed data using a mask of good samples smoothed in the same way. The filt_wlim value specifies the minimum fraction of good values that must contribute to a filtered value. For instance, if wlim is 0.9 then a filtered data value is flagged as bad unless at least 0.9 of the input values that contribute to it are good (i.e. have not been flagged as unusable for any reason). Thus a high filt_wlim value (i.e. close to 1.0) will cause more data to be rejected, and a low value (i.e. close to 0.0) will cause less data to be rejected. A value of <undef> causes the old filtering algorithm to be used that is based on filling gaps with artificial data. If the experimental algorithm is used the following additional settings can be made: " apod=0, fillgaps=0, noi.fillgaps=0" . [<undef>]

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

FLAGMAP
Dump diagnostic info about the time series qualities

Description:

If set to a comma-separated list of quality names, enclosed in parentheses, then an extension called .MORE.SMURF.FLAGMAPS will be created in the output map NDF. This will contain a set of " flagmaps" , one for each continuous chunk of time-series data, each containing a count of the number of samples with a quality bit matching at least one of the specified flags. Any of the standard quality names can be specified. These are: BADDA, BADBOL, SPIKE, DCJUMP, PAD, APOD, STAT, COM, NOISE, EXT, LOWAP, RING, SSN, PCA, IP.

Note: If BADBOL is set the behaviour is slightly different than expected; the bolometer will be completely ignored when creating the flag map.

Alternatively, if the single value " CUBE" is supplied for flagmap, then each created NDF will be 3-dimensional, containing a plane for each individual quality flag. Each pixel in the lowest plane (pixel index -1) will count the number of samples that had no flags set. Each of the remaining planes will count the number of samples that had a specific quality flag set. The planes are stored in the same order in which the quality flags are listed above. Thus, the plane with pixel index 0 will count the number of samples that have the BADDA flag set, and the plane with pixel index 16 will count the number of samples that have the IP flag set

The flagmap is mostly useful to verify whether the spike, jump, and common-mode rejection is correlated with features in the map (e.g. bright sources), for all of the detectors that were used to produce the map, e.g. flagmap = (BADBOL,SPIKE,DCJUMP,COM). [<undef>]

SMURF Usage :

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.APOD
Control the removal of low frequencies from the time-stream data

Description:

The number of samples to apodise at each end of the data stream prior to taking the FFT. If <undef>, a default value is used equal to 1/(steptime∗freq) where freq is the lowest edge or notch frequency. Note, apodisation is only performed if parameter "flt.zeropad" is set non-zero. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.CLIP
Speeds up convergences and reduces ringing by excluding sharp features in the bolometer time streams from the filtering performed by the FLT model

Description:

If this parameter is non-zero, each bolometer time stream is examined for sharp features prior to applying the hi-pass filter specified by parameter "flt.filt_edge_largescale" etc. Any such sharp features are replaced by a linear function joining the data values before and after the sharp feature. Sharp features are found by first smoothing the time stream using a simple box kernel with a width corresponding to the largest features passed by the FLT filter. The RMS of the residuals between the smoothed and original time stream is then found. Any samples that have residual greater than " flt.clip" times the RMS are set bad. Additional samples equivalent to 0.25 of the filter size are also set bad on either side of a clipped sample. If any such samples are found, the process is repeated (i.e. the remaining data is smoothed, the RMS residual recalculated and the check for outliers repeated). Once no further outliers are found, the bad data values are replaced by values that interpolate the adjoining good values using a linear function.

The value supplied for this parameter can optionally be a comma-separated list of two clipping levels enclosed in parentheses. In this case the first clipping level is used as described above. The second clipping level should be smaller than the first. It is used in much the same way as the first clipping level, except that points that would be clipped using the second clipping level are only clipped if at least 3 of the 5 nearest neighbours (including the point itself) are above the second clipping level.

This parameter is used to prevent the ringing that would otherwise be caused by the action of the filter on the sharp features. It should usually be seen as an alternative to the masking controlled via the " flt.zero_" set of parameters. Note, in some cases it may slow down convergence due to the fact that each bolometer is masked independently of the sky, thus reducing the consistency between each bolometer and the sky map.

Note, the masking described above can be restricted to selected iterations by setting a value for the parameter "flt.zero_niter" . [0.0]

SMURF Usage :

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_EDGE_SMALLSCALE
Specifies the smallest scale size to be retained by the FLT model

Description:

If non-zero, features with spatial sizes less than this value (in arcsec) will be removed by the FLT model. See parameter "flt.filt_edge_largescale" . [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_EDGE_SMALLSCALE_LAST
Specifies the smallest 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_smallscale" 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_EDGEHIGH
Specifies the lowest frequency to be retained by the FLT model

Description:

If non-zero, this is the cut-off frequency of a hard-edged high pass filter that is applied to the data stream as part of the FLT model. See also parameter "flt.filt_edge_largescale" . [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_EDGEHIGH_LAST
Specifies the lowest frequency to be retained by the FLT model on the last iteration

Description:

This is the value to be used for parameter "flt.filt_edgehigh" 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_EDGELOW
Specifies the highest frequency to be retained by the FLT model

Description:

If non-zero, this is the cut-off frequency of a hard-edged low pass filter that is applied to the data stream as part of the FLT model. See also parameter "flt.filt_edge_largescale" . [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_EDGELOW_LAST
Specifies the highest frequency to be retained by the FLT model on the last iteration

Description:

This is the value to be used for parameter "flt.filt_edgelow" 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_NOTCHHIGH
Specifies additional frequencies to be removed by the FLT model

Description:

Together with parameter "flt.filt_notchlow" , this parameter specifies a set of 1 or more hard-edge band-cut frequency-domain notch filters, specifying additional frequencies to be removed by the FLT model. The flt.filt_notchhigh value should be a comma-separated list of frequencies in Hz, enclosed in parentheses. These are the frequencies of the upper edges of the notch filters. The flt.filt_notchlow and flt.filt_notchhigh parameters should contain the same number of frequencies. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_NOTCHHIGH_LAST
Specifies additional frequencies to be removed by the FLT model on the last iteration

Description:

This is the value to be used for parameter "flt.filt_notchhigh" 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_NOTCHLOW
Specifies additional frequencies to be removed by the FLT model

Description:

Together with parameter "flt.filt_notchhigh" , this parameter specifies a set of 1 or more hard-edge band-cut frequency-domain notch filters, specifying additional frequencies to be removed by the FLT model. The flt.filt_notchlow value should be a comma-separated list of frequencies in Hz, enclosed in parentheses. These are the frequencies of the lower edges of the notch filters. The flt.filt_notchlow and flt.filt_notchhigh parameters should contain the same number of frequencies. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_NOTCHLOW_LAST
Specifies additional frequencies to be removed by the FLT model on the last iteration

Description:

This is the value to be used for parameter "flt.filt_notchlow" 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.FILT_ORDER_LAST
Indicates the shape of the filter on the last iteration

Description:

This is the value to be used for parameter "flt.filt_order" 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_WLIM
Enables an experimental filtering method

Description:

If supplied, this will switch on an experimental method for handling missing or flagged data when filtering the time streams, based on replacing missing data with zero, and then normalising the smoothed data using a mask of good samples smoothed in the same way. The flt.filt_wlim value specifies the minimum fraction of good values that must contribute to a filtered value. For instance, if wlim is 0.9 then a filtered data value is flagged as bad unless at least 0.9 of the input values that contribute to it are good (i.e. have not been flagged as unusable for any reason). Thus a high filt_wlim value (i.e. close to 1.0) will cause more data to be rejected, and a low value (i.e. close to 0.0) will cause less data to be rejected. A value of <undef> causes the old filtering algorithm to be used that is based on filling gaps with artificial data. If the experimental algorithm is used the following additional settings can be made: " apod=0, fillgaps=0, noi.fillgaps=0" . [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

FLT.FILT_WLIM_LAST
Enables an experimental filtering method on the last iteration

Description:

This is the value to be used for parameter "flt.filt_wlim" 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.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.RING_BOX2
Controls the flagging of samples that suffer from ringing

Description:

Specifies the size of the box used to determine the mean squared residual after FLT removal. It is specified as a multiple of the filter size, and should be greater than parameter "flt.ring_box1" . [1.5]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_FREEZE
Controls the flagging of samples that suffer from ringing

Description:

The flags will be frozen after the number of iterations specified by this parameter. This helps convergence. Set it to zero to allow the flags to change on every iteration. Note, any initial iterations specified by parameter "ast.skip" are not included in the count of iterations. [10]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_MASK
Controls the flagging of samples that suffer from ringing

Description:

If this parameter is set to a non-zero value, then samples that fall within a source region (as defined by the AST and/or FLT mask) are never flagged by the ringing filter. If this parameter is set to zero (the default) then no distinction is made between background and source samples within the ringing filter.

If an FLT mask is available, it is used to define source regions. Otherwise, the AST mask is used. If no mask is available, then behaviour is as if zero was supplied for this parameter.

Bright compact sources can cause ringing, which the ringing filter would normally identify, resulting in the source samples being flagged and excluded from the map. This results in fewer samples in the source pixels and so higher noise. In bad cases, it can result in all source samples being rejected, thus causing a hole of bad pixels in the map at the position of the source. Setting this parameter to a non-zero value can help in such cases. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_MINSIZE
Controls the flagging of samples that suffer from ringing

Description:

A contiguous section of residuals that are considered to suffer from ringing will only be flagged if it is longer than the size specified by this parameter. It is given as a multiple of the filter size. See parameter "flt.ring_box1" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_NSIGMA
Controls the flagging of samples that suffer from ringing

Description:

Specifies the number of standard deviations at which to flag samples that suffer from ringing. See parameter "flt.ring_box1" . [2.0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.RING_WING
Controls the flagging of samples that suffer from ringing

Description:

For each contiguous section of residuals that are considered to suffer from ringing, this parameter gives the number of extra samples to flag at the beginning and end of the section. It is given as a multiple of the filter size. See parameter "flt.ring_box1" . [0.3]

SMURF Usage :

MAKEMAP, CALCQU

FLT.UNDOFIRST
Improves convergence

Description:

If non-zero, add the old FLT model back into the residuals at the start of the iteration (and so before COM) rather than just before estimating a new FLT Model. [1]

SMURF Usage :

MAKEMAP, CALCQU

FLT.WHITEN
Experimental

Description:

If non-zero, then a whitening filter will be applied to each time stream prior to apply the other FLT filters. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.WHITEN_LAST
Experimental

Description:

This is the value to be used for parameter "flt.whiten" 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.ZERO_ACCUM
Prevents unstable SNR masks halting convergence

Description:

Masks defined by SNR limits can show instabilities in which pixels can repeatedly enter the mask on one iteration and leave it on the next. Such oscillating pixels tend to occur around the edges of source areas in the mask, and can prevent convergence of the iterative map-making process. Setting FLT.ZERO_ACCUM to a non-zero value can prevent this by forcing source pixels to be accumulated rather than replaced on successive iterations. Thus, if FLT.ZERO_ACCUM is non-zero, any map pixel which is flagged as a source pixel on some iteration will never be unflagged - once a source pixel, always a souce pixel. [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_MASK0
Reduces spurious large scale structure in the final map within fixed regions specified by an external mask

Description:

This parameter allows an alternative mask to be used on the first iteration. If a value is supplied for " flt.zero_mask0" , it will be used in place of parameter "flt.zero_mask" on the first iteration only. If no value is supplied for " flt.zero_mask0" , the value supplied for " flt.zero_mask" will be used on all iterations including the first. Setting " flt.zero_mask0" to zero will cause no masking to be used on the first iteration. [<undef>]

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_NOTLAST
Prevent FLT masking being performed on the last iteration

Description:

If flt.zero_notlast is 1, then the FLT model is not masked on the final iteration. [1]

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_SNR_HIPASS
Flatten map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "flt.zero_snr" to be based on a copy of the SNR map that has been high-pass filtering to remove structures larger than the number of arc-seconds given by flt.zero_snr_hipass. This will in general reduce the number of source pixels in the mask. See also parameter "flt.zero_snr_ffclean" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZERO_SNR_LOPASS
Smooths map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "flt.zero_snr" to be based on a copy of the SNR map that has been low-pass filtering (i.e smoothed) to remove noise and other features smaller than the number of map pixels given by flt.zero_snr_lopass. This may help to prevent pixels oscillating in and out of the mask on successive iterations, and thus aid smoother convergence. [0.0]

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

FLT.ZEROPAD
Avoid unnecessary loss of data due to apodisation

Description:

If flt.zeropad is non-zero, the extra samples used to pad each end of the data streams are set to 0 prior to filtering within the FLT model, and apodization is used to slowly roll-off the ends of the time series to 0 (see parameter "flt.apod"). If flt.zeropad is zero, a cubic polynomial is used to interpolate smoothly between the end of the time series and the beginning of the time series, ensuring continuity in both the value and first derivative. In both cases, the purpose is to remove sharp edges that may cause ringing in the FFT filtering steps. [0]

SMURF Usage :

MAKEMAP, CALCQU

FLT.ZEROPAD_LAST
Avoid unnecessary loss of data due to apodisation on the last iteration

Description:

This is the value to be used for parameter "flt.zeropad" on the last iteration. If it is "<undef>", then the same value will be used as for earlier iterations. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

GAI.FLATFIELD
Provides an alternative flat-fielding mechanism

Description:

If non-zero, use the GAIn/COMmon models to re-calculate the flatfield. May be useful in some cases, but dangerous for short scans of very bright sources because the astronomical signal may completely dominate sky signal. Note, this parameter is only used with the " old" COM algorithm (see parameter "com.oldalg"). [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

GROUPSUBARRAY
Produce separate maps for each sub-array

Description:

If non-zero, handle each subarray separately (as if each subarray was in a different continuous chunk). Normally all data that were taken simultaneously at a given wavelength are combined into a single map at each iteration. [0]

SMURF Usage :

SC2CLEAN, 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

IMPORTLUT
Controls the creation of the LUT model

Description:

If non-zero, import the the LUT model (the map pixel index associated with each data sample) from an NDF created by a previous run of makemap. The NDF is expected to have the same name as would be created by setting parameter " exportNDF" to " (lut)" in the makemap configuration. [0]

SMURF Usage :

MAKEMAP, CALCQU

IMPORTSKY
Allow a map created by a previous run of makemap to be used as the inital estimate of the sky

Description:

If importsky 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 the initial estimate of the sky. The image value at the position of each bolometer sample will be found and subtracted from the bolometer value immediately before the start of the first iteration. In addition, the supplied image will be added back onto the map esimated at the end of the first iteration, before using the map to determine the AST model. Setting importsky 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 inital sky being removed. 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 maps are aligned with the output map in pixel coordinates). [0]

SMURF Usage :

MAKEMAP, CALCQU

IPANGOFF
Modify the angle of Instrumental Polarisation correction

Description:

This parameter is only used when processing POL2 Q or U time-streams as created by SMURF:CALCQU, and if a value is supplied for ADAM parameter IPREF. It specifies an offset to be added onto the " D" constant defined by the selected IP model, and should be supplied in units of degrees. The " D" value is the elevation at which the IP is parallel to the focal plane Y axis (i.e. all Q and no U, if using the cocal plane Y axis as the polarimetric reference direction). Creating maps with different values of ipangoff provides a handle on how sensitive the map is to the details of the IP correction. See also parameter "ipoffset" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

IPMODEL
Specify the Instrumental Polarisation model to use

Description:

This parameter is only used when processing POL2 Q or U time-streams as created by SMURF:CALCQU, and if a value is supplied for ADAM parameter IPREF. It specifies the IP model to be used. Currently the following models are available:

- " JAN2018" : See report " IP model without the wind blind" (written January 2018) in section " Data Reduction and Analysis" of the POL2 commissioning wiki.
- " APR2019" : See email from Pierre, sent to DSB and PF on 15th April 2019 - subject " 850 IP model" . Note, this model only covers 850 data at the moment.
- " AUG2019" : See post " New IP models for POL2 data" posted 9th August 2019 on the JCMT software blog. [AUG2019]

SMURF Usage :

MAKEMAP, CALCQU

IPOFFSET
Modify the level of Instrumental Polarisation correction

Description:

This parameter is only used when processing POL2 Q or U time-streams as created by SMURF:CALCQU, and if a value is supplied for ADAM parameter IPREF. It specifies an offset to be added onto the expected level of IP predicted by the selected IP model. The value should be supplied as a percentage. Thus, setting ipoffset to (say) 0.3 will cause the expected level of IP to be increased by 0.3% (typical IP levels are in the range 1-2%, and vary with elevation). Creating maps with different values of ipoffset provides a handle on how sensitive the map is to the details of the IP correction. The current IP model is roughly accurate to about +/- 0.3%. See also parameter "ipangoff" . [0.0]

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

MAPLAG
Specifies the degree of lagging between iterations

Description:

If maplag is non-zero, then the map created at the end of each iteration is replaced by a weighted mean of the new map and the map from the previous iteration. The " maplag" parameter gives the weight for the previous map, and should be in the range 0.0 to 1.0. The weight for the new map is (1.0-maplag). Thus if maplag=0.0 (the default), no change is made to the new map. Using a non-zero value for maplag may help to prevent oscillations in the SNR mask from iteration to iteration. [0.0]

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

MAPTOL_BOX
Specifies when to stop iterating

Description:

If greater than zero, the array holding the normalised change at each map pixel is smoothed using a box filter before estimating the mean or max value (as determined by parameter "maptol_mean"). The box size is specified in arc-seconds. This will have little effect if " maptol_mean" is non-zero (i.e. if the mean map change is being used as the stopping criterion), but will tend to produce smoother convergence if " maptol_mean" is zero (i.e. if the maximum map change is being used as the stopping criterion). [0.0]

SMURF Usage :

MAKEMAP, CALCQU

MAPTOL_HITS
Specifies when to stop iterating

Description:

If non-zero, pixels that receive very few bolometer samples are not included in the estimate of the mean or max map change between iterations. The limiting number of bolometer samples is equal to " maptol_hits" times the mean number of hits per pixel, averaged over the map pixels that recieve at least one bolometer sample. The default value for " maptol_hits" is the value of parameter "hitslimit" . [[hitslimit]]

SMURF Usage :

MAKEMAP, CALCQU

MAPTOL_MASK
Specifies when to stop iterating

Description:

This parameter specifies the mask that defines the area over which the normalised map change should be calculated when determining the mean or maximum map change at the end of each iteration. See parameter "maptol" . The supplied value can be " AST" , " FLT" , " COM" , " PCA" or " <undef>" . If the specifies mask is not defined, or if " <undef>" is supplied, then the map change is determined over the entire map. [AST]

SMURF Usage :

MAKEMAP, CALCQU

MAPTOL_MEAN
Specifies when to stop iterating

Description:

If this value is non-zero, then the value specified by parameter "maptol" defines a target value for the mean change in normalised pixel value between iterations. Otherwise, "maptol" defines a target value for the maximum change in normalised pixel value between iterations. [1]

SMURF Usage :

MAKEMAP, CALCQU

MAPTOL_RATE
Specifies when to stop iterating

Description:

The normalised mean change in pixel value (see parameter "maptol") con sometimes settle down to a constant value, and thus never drop to the target " maptol" value. This may be the case for instance, if every iteration adds a fixed error map onto the current astronomical signal (see parameter "epsin"). Specifying a value for " maptol_rate" causes the iterative loop to be left if the normalised mean change seems to settle down to a fix value. The supplied value for " maptol_rate" should be a fraction in the range 0.0 to 1.0, and gives the minimum required fractional change in " normalised mean change" between iteration. If the actual value falls below this minimum value, the iterative loop is left. [<undef>]

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

MEMCHECK
Indicates what should happen if chunking is detected

Description:

This parameter indicates what should happen if the available memory is too low to process the supplied time-series data without splitting it into chunks. It can take any of the following values:

0 - A warning is issued but makemap then continues to produce a map from the chunked data.

1 - An error is reported with status SMF__NOMEM and no map is created. The output map is created as normal if the available memory is sufficient to avoid chunking.

2 - An error is reported with status SMF__NOMEM and no map is created. In addition, an error is also reported with status SMF__MEMCHK and no output map is created if the available memory is sufficient to avoid chunking. Thus if memcheck is 2, no map is ever created. This is intended as a means to check whether chunking would or would not be used if makemap was to be used subsequently to create a map.

Note, in either case the SMF__NOMEM error is reported only if chunking is caused by lack of memory, not if it is caused by the supplied data being discontiguous. [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

NOEXPORTSETBAD
Controls contents of diagnostic maps

Description:

Normally all data points in exported model components (see parameter "exportndf") are set to the Starlink " bad" value wherever there is a bad bolometer established during map-making. Set this flag to a non-zero value to prevent this behaviour. [0]

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

NOI.DCFITBOX
Controls the detection and correction of steps within the NOI model

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 in the NOI model. If positive, in units of samples. If negative, in units of seconds. If zero, do not perform step correction during estimation of the NOI model. [0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.DCLIMCORR
Controls the detection and correction of steps within the NOI model

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 NOI model. Only used if parameter "noi.dcfitbox" is non-zero. [10]

SMURF Usage :

MAKEMAP, CALCQU

NOI.DCMAXSTEPS
Controls the detection and correction of steps within the NOI model

Description:

The maximum number of steps that can be corrected in each minute of good data (i.e. per 12000 samples) 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 "noi.dcfitbox" is non-zero. [10]

SMURF Usage :

MAKEMAP, CALCQU

NOI.DCSMOOTH
Controls the detection and correction of steps within the NOI model

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 "noi.dcfitbox" is non-zero. [50]

SMURF Usage :

MAKEMAP, CALCQU

NOI.DCTHRESH
Controls the detection and correction of steps within the NOI model

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 "noi.dcsmooth" . In order to find the equivalent threshold in the unsmoothed data, multiply the noi.dcthresh value by 1.25/sqrt(noi.dcsmooth). For instance, the default values for noi.dcsmooth (50) and noi.dcthresh (25) correspond to a threshold of 25∗1.25/sqrt(50) = 4.4 sigma in the unsmoothed data. [25.0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.EXPORT
Controls the exporting of the NOI model

Description:

If set to a non-zero value, export the values in the NOI model to an NDF. The name of the NDF is similar to that used by parameter "exportndf" , but with a trailing suffix of " _noi" instead of " _res" . Unlike " exportndf" , the NOI values are stored in the " Data" array of the NDF, and only one value is stored for each box in the NOI model (see parameter "noi.box_size"). This makes the NDF much smaller than that produced by " exportndf" . The box size (in samples) is stored in the NOI_BOXSIZE item in the SMURF extension of the NDF. [0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.FILLGAPS
Controls handling of missing data within NOI FFT code

Description:

If non-zero, fill vicinity of spikes / DC steps with constrained realization of noise. [1]

SMURF Usage :

MAKEMAP, CALCQU

NOI.IMPORT
Controls the noise values used in the NOI model

Description:

If set to a non-zero value, import the noise values to use in the NOI model from an NDF created by a previous run of makemap, rather than calculating them from scratch. The NDF should have been created by setting parameter "noi.export" to 1 in the configuration for a previous run of makemap. (the file should end in " _noi.sdf"). Note, the parameter "noi.box_size" must be set to the same value it had in the previous run of makemap. [0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.SPIKEBOX
Controls time-based spike detection within NOI model

Description:

Size of filter box for the sigma-clipper within the NOI model, in units of samples if positive and seconds if negative. For instance, setting noi.spikebox to 50 will check for excursions from a rolling median filter in a box of length 50 samples. Also see parameter "noi.spikethresh" . [50]

SMURF Usage :

MAKEMAP, CALCQU

NOI.SPIKETHRESH
Controls time-based spike detection within NOI model

Description:

The SNR value at which to flag spikes within the sigma-clipper used by the NOI model. Also see parameter "noi.spikebox" . No de-spiking is performed by the NOI model if a value of zero is supplied. [0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.USEVAR
Controls how the NOI model is created

Description:

If non-zero, then the NOI model is created prior to the first iteration from the variance values in the supplied data files. Otherwise, it is found by analysing the residual bolometer values in the manner specified by parameter "noi.box_type" . [0]

SMURF Usage :

MAKEMAP, CALCQU

NOI.ZEROPAD
Controls how the NOI model handles missing data

Description:

See parameter "flt.zeropad" . [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

OPTEFFDIV
Controls optical efficiency correction

Description:

If non-zero, the bolometer data is divided by the values read from the optical efficiency files (see parameter "opteffs4a" , etc). Otherwise, the bolometer data is multiplied by the values read from the optical efficiency files. [1]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS4A
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S4A bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS4B
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S4B bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS4C
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S4C bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS4D
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S4D bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS8A
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S8A bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS8B
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S8B bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS8C
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S8C bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

OPTEFFS8D
Controls optical efficiency correction

Description:

The path to an NDF defining the optical efficiency correction for each S8D bolometer. This should be a 2D NDF with pixel bounds (0:31,0:39). [<undef>]

SMURF Usage :

SC2CLEAN, 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

PAD
Specifies the length of padding to add to the start and end of each time series

Description:

Specifies the length, in down-sampled time slices, of the padding to add the the start and end of each time series to prevent wrap-around effects between the start and end of the data streams when taking the FFT. If <undef>, a default value is used which is determined from the filtering being performed by the FFT. See also parameter "zeropad" . [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

PAOFF
Specifies the orientation of the have-wave plate origin

Description:

This is the angle from the fixed analyser to the have-wave plate for a POL_ANG value of zero, in degrees. Measured positive in the same sense as rotation from focal plane X to focal plane Y. [0.0]

SMURF Usage :

CALCQU

PASIGN
Indicates the sense of rotation of the spinning half-wave plate

Description:

If non-zero, it is assumed that a positive POL_ANG value corresponds to rotation from focal plane X to focal plane Y axis. If zero, it is assumed that a positive POL_ANG value corresponds to rotation from focal plane Y to focal plane X axis. [1]

SMURF Usage :

CALCQU

PCA.BACKOFF
Controls the determination of multiple background signals

Description:

This parameter controls whether the residuals after subtraction of the PCA model should be flattened by the removal of a quadratic background from each bolometer. Disabling this option speeds up the PCA model considerably, and seems to produce little effect if the PCA model occurs after the COM model within the model order. [0]

SMURF Usage :

MAKEMAP, CALCQU

PCA.COMFILL
Controls the determination of multiple background signals

Description:

This parameter controls how missing bolometer values are handled by the PCA model (see parameter "pca.pcathresh"). Such missing values, whether caused by rejection of bad data or masking of sources, are replaced by artificial data values representative of the surrounding data prior to the calculation of the PCA model. If the pca.comfill parameter is false (i.e. zero) then such gaps in the data are filled using linear interpolation between the neighbouring good values in the time-stream. if pca.comfill is true (i.e. non-zero) then gaps are filled using the common-mode signal (i.e. the mean of the remaining good values at each time-slice), together with corresponding bolometer gains. This can in some case produce a more reliable PCA model. However, using this option is only a good idea if there is a strong common mode present in the bolometer data. This will usually only be the case if PCA is the first model specified in parameter "modelorder" .

If pca.comfill is non-zero, gaps are filled using a COM and GAI model determined using a set of parameters equivalent to those by the " normal" COM and GAI models, except that the name of each parameter is changed from " com.xxx" to " pca.xxx" . Thus " pca.perarray" is used in place of the normal parameter "com.perarray" . [0]

SMURF Usage :

MAKEMAP, CALCQU

PCA.NOISELIM
Controls the determination of multiple background signals

Description:

This parameter gives the smallest allowed noise level after removal of the PCA model. If the removal of the PCA model causes the noise in a bolometer to fall to lower than " pca.noiselim" times the noise level it had before removal of the PCA model, then the bolometer is flagged as bad until the next iteration. This usually happens for bolometers that appear to be " unique" for some reason (i.e. look completely unlike other bolometers). In such cases one of the PCA components will be nearly identical to the bolometer, and so leave no significant residuals after subtraction of hte PCA model. Note, this parameter should normally be set to zero if the PCA model occurs before the COM model within pareameter " modelorder" . This is because the PCA model will then remove the typically huge common mode, resulting in an apparent huge fall in noise in all bolometers. This would cause all bolometers to be rejected if the default value is used for pca.noiselim. [0.1]

SMURF Usage :

MAKEMAP, CALCQU

PCA.PCATHRESH
Controls the determination of multiple background signals

Description:

An alternative to modelling the background in each bolometer using a simple common-mode (the COM model) is to use principal component analysis (the PCA model). The PCA model calculates a new set of basis vectors (components) from the bolometer time-series such that the components have zero covariances. The amplitudes of these new correlated signals are then calculated, and the largest amplitude components can be removed. Whereas the COM model assumes a single background signal that is the same for all bolometers, the PCA model is capable of detecting multiple background signals with different correlation patterns across the array.

As with the common-mode, PCA can be used either as a distinct model within the iterative map-making process (in which case " PCA" should be included in the string supplied for parameter "modelorder"), or as a step in the cleaning performed prior to the start of the iterative map-making loop (see parameter "pcathresh").

The main parameter for the PCA model is the number of components to remove, specified by " pca.pcathresh" . If a positive value is supplied, a sigma-clipping algorithm is used to determine the number of components to remove (see below). If a negative value is supplied, the nearest integer (after removal of the minus sign) is found and used as the absolute number of components to remove. Thus if pca.pcathresh is set to -10, the 10 strongest components are removed.

If a positive value is supplied for pca.pcathresh, the RMS amplitude across all bolometers is calculated for each component – a single positive number related to the average strength of the component. An iterative sigma clipper is then used to flag components that are more than pcathresh$*$RMS above the mean value. This approach is quite arbitrary, but a value of about 4 seems reasonable for some test data.

Be warned that this statistical black box will remove real correlated astronomical signals as well, thus slowing down the rate of convergence of iterative map-making algorithm. As with common-mode removal, the actual impact on science will need to be calibrated with simulations.

The PCA model can be masked in the same way as other models to reduce the influence of bright sources on the model. A set of parameters " pca.xxx" can be used for this purpose, where " xxx" is any of " zero_accum" , " zero_circle" , " zero_freeze" , " zero_lowhits" , " zero_mask" , " zero_niter" , " zero_notlast" , " zero_snr" , " zero_snr_ffclean" , " zero_snr_hipass" , " zero_snr_lopass" , " zero_snr_fwhm" , " zero_snr_low" , " zero_snrlo" or " zero_union" . See the documentation for the equivalent " com.xxx" parameters for details. [4.0]

SMURF Usage :

MAKEMAP, CALCQU

PCA.PCATHRESH_FREEZE
Controls the determination of multiple background signals

Description:

If a positive value is supplied for parameter "pca.pcathresh" (and the PCA model is included in parameter "modelorder"), then the number of PCA components to remove on each iteration is determined dynamically using a sigma-clipping algorithm, and the number of components removed can change between iterations. A potential problem with this is that the number of components removed may not settle down to a fixed value but may continue to change, thus slowing convergence. To prevent this, a non-zero value may be specified for " pca.pcathresh_freeze" , which prevents any further change to the number of components removed. If the value of pca.pcathresh_freeze is between zero and 1.0, it specifies the normalised mapchange value at which to freeze the number of removed components. If the pca.pcathresh_freeze value is greater than 1 it specifies an integer number of iterations, in addition to the initial AST-skip iterations, after which the number of removed components should be frozen. If pca.pcathresh_freeze is negative, the number of components to remove is frozen as soon as any initial AST-skip iterations have been performed. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

PCA.SAMPLIM
Controls the determination of multiple background signals

Description:

This parameter gives the minimum fraction of good samples that a bolometer must have for it to be used in the determination of the principal components. [0.8]

SMURF Usage :

MAKEMAP, CALCQU

PCA.SKIP
Controls the determination of multiple background signals

Description:

This parameter allows the PCA model (see parameter "pca.pcathresh") to be skipped on a specified number of initial iterations A fractional value between 0.0 and 1.0 specifies the normalised map change value at which to start using the PCA model. A positive integer specifies the number of initial iterations for which no PCA model should be used. A negative value causes the PCA model to be used only on the very last iteration. Zero causes the PCA model to be used on all iterations. If it is left undefined, the PCA model is used on all iterations other than any initial iterations for which the AST model is skipped (See parameter "ast.skip"). [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

PCALEN
Controls PCA cleaning

Description:

This specifies the chunk length that will be cleaned as a number of time slices (be careful if you have down-sampling turned on!). If set to 0 it will default to the length of the full time-series. See parameter "pcathresh" . [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

PCATHRESH
Controls PCA cleaning

Description:

An alternative to removing the simple common-mode is cleaning by way of principal component analysis (PCA). In this case, a new set of basis vectors (components) are calculated from the bolometer time-series such that they have 0 covariances. The amplitudes of these new correlated signals are then calculated, and the largest amplitude components can be removed. In this simplest case of white detectors noise + a common atmospheric signal, this operation would be similar to using parameter "compreprocess" . However, PCA is capable of detecting multiple signals with different correlation patterns across the array.

It is ∗ESSENTIAL∗ that the bolometer data first have their mean values removed (i.e., parameter "order" should be 0), as this assumption is used to speed up the calculation.

The main parameter for PCA cleaning is the threshold above which components will be removed from the bolometer time-series. For each component, the RMS amplitude across all bolometers is calculated – a single positive number related to the average strength of the component. An iterative sigma clipper is then used to flag components that are more than thresh∗RMS away from the mean value.

This approach is quite arbitrary, but a value of about 4 seems reasonable for some test data. Be warned that this statistical black box will remove real correlated astronomical signals as well! As with common-mode removal, the actual impact on science will need to be calibrated with simulations.

If this parameter is set to 0 no pca cleaning will occur. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

PLN.NOTFIRST
Controls the PLN model

Description:

If non-zero, the PLN model is skipped on the first iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

POL2FP
Controls creation of maps holding POL2 Q/U values

Description:

If zero, an error is reported if an attempt is made to create a map from Q or U time series that uses the focal plane Y axis as the reference direction. Normally, maps should be made from Q/U values that use celestial north as the reference direction in order to avoid problems caused by sky rotation. However, if " pol2fp" is set non-zero, then no error will be reported if the Q/U time series uses focal plane Y as the reference direction. This may be necessary, for instance, when processing data to determine the parameters of the IP (Instrumental Polarisation) model. Note, in such cases the output map will not contain a POLANAL Frame in the WCS FrameSet, and so will be unusable with any POLPACK applications. [0]

SMURF Usage :

MAKEMAP, CALCQU

POLBOX
Controls the least squares fit used to determine Q and U

Description:

The number of half-waveplate rotations in a fitting box. Only used if ADAM parameter LSQFIT is set to TRUE. [1]

SMURF Usage :

CALCQU

SAMPCUBE
Provides extra diagnostic information

Description:

If non-zero, then an extension called .MORE.SMURF.SAMPCUBES is added to the output map NDF, holding data cubes of data samples that go into each map pixel. [0]

SMURF Usage :

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

SMO.BOXCAR
Controls the SMO model

Description:

The width of the smoothing box used by the SMO model. This is roughly equivalent to FLT with a high-pass filter at cutoff frequencies 0.1 and 0.3 Hz for 450 and 850 respectively). If positive, value is in samples. If negative, value is in seconds. [-10 (for 450 um), -3 (for 850 um)]

SMURF Usage :

MAKEMAP, CALCQU

SMO.NOTFIRST
Controls the SMO model

Description:

If non-zero, skip the SMO model on the first iteration. [1]

SMURF Usage :

MAKEMAP, CALCQU

SMO.TYPE
Controls the SMO model

Description:

The type of filter used by the SMO model: median or mean. [median]

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

SSN.BINSIZE
Controls the Scan Synchronous Noise model

Description:

ssn.binsize gives the size of the azimuth bins used to form the profile of bolometer value against azimuth offset. If zero is zupplied , the map pixel size is used. If a negative value is supplied, the bin size is specified as a multiple of the map pixel size. If a positive value is suppled, it is given directly in arc-seconds. [-4]

SMURF Usage :

MAKEMAP, CALCQU

SSN.KSIGMA
Controls the Scan Synchronous Noise model

Description:

If ssn.ksigma is greater than zero, then the profile that gives the scan-synchronous noise for each bolometer is smoothed with a Gaussian kernel before being used. The value supplied for ssn.ksigma is the standard deviation of the kernel in units of azimuth bins (see parameter "ssn.binsize"). [1.0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.NOTFIRST
May improve convergence by avoiding the use of a scan sychronous noise model on the first iteration

Description:

If this is non-zero, then scan synchronous noise will not be removed from the time streams on the first iteration. [0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.SECLEN
Controls the Scan Synchronous Noise model

Description:

If this is non-zero, then each bolometer time stream is split into sections and a separate SSN model is calculated for each section. The value of ssn.seclen determines the length of each section, and should be supplied as a multiple of the basic azimuth period of the scan pattern. For instance, a value of " 5" will result in each section containing five azimuth periods (a period is twice the time taken for the boresight to pass through zero azimuth offset). [10]

SMURF Usage :

MAKEMAP, CALCQU

SSN.THRESH
Controls the Scan Synchronous Noise model

Description:

If this is non-zero, then bolometers for which the magnitude of the scan synchronous noise is less that the specified multiple of the noise level are assumed to have zero scan synchronous noise. Raising the value will result in only the bolometers with the strongest SSN being corrected. Bolometers with weaker SSN are left uncorrected. [0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_ACCUM
Prevents unstable SNR masks halting convergence

Description:

Masks defined by SNR limits can show instabilities in which pixels can repeatedly enter the mask on one iteration and leave it on the next. Such oscillating pixels tend to occur around the edges of source areas in the mask, and can prevent convergence of the iterative map-making process. Setting SSN.ZERO_ACCUM to a non-zero value can prevent this by forcing source pixels to be accumulated rather than replaced on successive iterations. Thus, if SSN.ZERO_ACCUM is non-zero, any map pixel which is flagged as a source pixel on some iteration will never be unflagged - once a source pixel, always a souce pixel. [0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_CIRCLE
May speed up convergences by excluding sources within a circle of given radius from the SSN estimate

Description:

Using ssn.zero_circle causes any samples falling within a specified circle on the map to be excluded from the estimation of the SSN model. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_FREEZE
Prevent the SSN mask from changing after a given number of iterations. This can help convergence

Description:

If ssn.zero_freeze is 1.0 or more, the SSN 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 ssn.zero_freeze is greater than zero but less than 1.0, the SSN mask will be frozen when the normalized change in the map between iterations drops below the ssn.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

SSN.ZERO_LOWHITS
Experimental

Description:

Using ssn.zero_lowhits causes samples to be excluded from the estimation of the SSN model if they fall in regions of the map where the number of samples falling in each pixel is higher than ssn.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

SSN.ZERO_MASK
May speed up convergences by excluding sources within a region specified by an external mask file from the estimation of the SSN model

Description:

If ssn.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 ssn.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 estimation of the SSN model should be set to the bad value in the mask. All other pixels will be excluded from the model. [0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_NITER
Allows SSN masking to be switched off after a given number of iterations

Description:

If ssn.zero_niter is non-zero, it gives the number of iterations for which the SSN model should be masked. Subsequent iterations are not masked. A value of zero means " mask on all iterations" . However, if parameter "ssn.zero_notlast" is set, the mask will will not be applied on the last iteration, even if ssn.zero_niter is zero. Note, using SSN 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 SSN model. [2]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_NOTLAST
Prevent SSN masking being performed on the last iteration

Description:

If ssn.zero_notlast is 1, then the SSN model is not masked on the final iteration. [1]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_SNR
May speed up convergences by excluding samples that correspond to high SNR pixels in the map

Description:

Setting the ssn.zero_snr parameter will prevent samples contributing to the SSN model if they fall within map pixels that have SNR values greater than ssn.zero_snr. A ssn.zero_snr value of zero means no SNR mask is used. See also parameter "ssn.zero_snr_ffclean" .

Note, the SNR values are only available once a map has been created, and so using this parameter results in no SSN masking on the first iteration. Consequently the map at the end of the first iteration may have noticable straight lines aligned with azimuth that pass through bright sources, since no SSN masking was done. Normally, these lines would polute the AST model derived from the map, and thus polute the residuals on the next iteration, resulting in the lines 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 SSN mask can be formed from these initial iterations before any AST model is calculated and used. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.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 "ssn.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 "ssn.zero_snr" gives the clipping level of the ffclean algorithm, and the parameter "ssn.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

SSN.ZERO_SNR_HIPASS
Flatten map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "ssn.zero_snr" to be based on a copy of the SNR map that has been high-pass filtering to remove structures larger than the number of arc-seconds given by ssn.zero_snr_hipass. This will in general reduce the number of source pixels in the mask. See also parameter "ssn.zero_snr_ffclean" . [0.0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_SNR_LOPASS
Smooths map before making SNR mask

Description:

Setting this parameter to a positive value causes the SNR mask requested by parameter "ssn.zero_snr" to be based on a copy of the SNR map that has been low-pass filtering (i.e smoothed) to remove noise and other features smaller than the number of map pixels given by ssn.zero_snr_lopass. This may help to prevent pixels oscillating in and out of the mask on successive iterations, and thus aid smoother convergence. [0.0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_SNRLO
May speed up convergences by increasing the size of the SNR mask without introducing noise

Description:

If values are supplied for ssn.zero_snrlo and parameter "ssn.zero_snr" , then the basic mask created by thresholding at the SNR value specified by ssn.zero_snr is modified by expanding each un-masked " source" area down to an SNR equal to ssn.zero_snrlo, without introducing any new isolated source areas. The ssn.zero_snrlo should be lower than the ssn.zero_snr value. [0]

SMURF Usage :

MAKEMAP, CALCQU

SSN.ZERO_UNION
Controls how multiple SSN masks are combined

Description:

If more than one SSN mask is specified (for instance, if values are supplied for both parameter "ssn.zero_lowhits" and parameter "ssn.zero_snr"), then they are combined into a single mask. If ssn.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 ssn.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

STARTUP
Blanks initial section of time-stream

Description:

This specifies the number of initial time slices that are to be excluded from the map. This can be used to exclude the data at the start of each observation during which the telescope is getting up to speed and establishing the required scan pattern. If positive, the value is the number of (down-sampled) time slices. If negative, the absolute value is a number of seconds. [0]

SMURF Usage :

MAKEMAP, CALCQU

SUBMEAN
Experimental

Description:

If non-zero, the mean value is subtracted from each time-slice before calculating the Q and U values. This typically has very little effect. [0]

SMURF Usage :

CALCQU

TMP.DOCOS
Controls the TMP model

Description:

Indicates that the cosine of the template should be taken before fitting the TMP model. It is assumed that the template quantities are in radians. [0]

SMURF Usage :

MAKEMAP, CALCQU

TMP.DOSIN
Controls the TMP model

Description:

Indicates that the sine of the template should be taken before fitting the TMP model. It is assumed that the template quantities are in radians. [0]

SMURF Usage :

MAKEMAP, CALCQU

TMP.SOURCE
Controls the TMP model

Description:

The TMP model fits an externally defined template to each bolometer time-series. The tmp.source parameter defines what that template is. Presently the only two valid options are " state_az" and " state_el" to use the telescope azimuth and elevation respectively. The former seems to be a good way for removing magnetic field pickup. [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

TMP.TRIGOFFSET
Controls the TMP model

Description:

Indicates an offset that is to be added to the TMP model template. [0]

SMURF Usage :

MAKEMAP, CALCQU

TSTEP
Determines the accuracy of each sample location within the map

Description:

The gap (in time slices if positive and seconds if negative) between full calculations of the output map bolometer positions. Setting a larger value for this will speed up the map maker but will introduce larger spatial errors. The default value of -0.5 (i.e. 100 samples at a sample rate of 200 Hz) seems to produce spatial errors of under 0.1 arc-sec. This level of errors seems to cause about 1% of bolometers samples to be pushed into a neighouring map pixel. For tstep=100, the calculation of bolometer positions speeds up by about a factor of 60. Setting tstep to 1 (or zero) causes all bolometer positions to be calculated in full, without any approximation. [-0.5]

SMURF Usage :

MAKEMAP, CALCQU

TWO.AMAP
Controls the TWO model

Description:

The " B" map to be used in the TWO model [<undef>]

SMURF Usage :

MAKEMAP, CALCQU

VALIDATE_SCANS
Controls the rejection of bad scan patterns

Description:

If non-zero, each raw data file is checked to see if the telescope goes crazy during the subscan (i.e. spends a significant amount of time outside the expected map area). If it does, a warning message is issued and the pointing information associated with the subscan is set blank so that the subscan is excluded from the map. It may be necessary to set this value to zero (i.e. inhibit the check) if any subscans intentionally spend a significant amount of time outside the expected map area.

The expected map area is defined by the MAP_HGHT and MAP_WDTH FITS headers, and is assumed to be centred on the tracking centre. If more than 10% of the frames are outside this area, the whole subscan is rejected. [1]

SMURF Usage :

MAKEMAP, CALCQU

VARMAPMETHOD
Controls how the map variance values are found

Description:

If zero, propagate variances from the time-domain noise esimates (requires the NOI model component to be present in parameter "modelorder"). Otherwise, use the sample variance of the data values that land in each pixel. [1]

SMURF Usage :

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

ZEROPAD
Avoid unnecessary loss of data due to apodisation

Description:

If zeropad is non-zero, the extra samples used to pad each end of the data streams are set to 0 prior to filtering within the initial cleaning, and apodization is used to slowly roll-off the ends of the time series to 0 (see parameter "apod"). If zeropad is zero, a cubic polynomial is used to interpolate smoothly between the end of the time series and the beginning of the time series, ensuring continuity in both the value and first derivative. In both cases, the purpose is to remove sharp edges that may cause ringing in the FFT filtering steps. [0]

SMURF Usage :

SC2CLEAN, MAKEMAP, CALCQU

G DREAM/STARE data reduction workflow

The DREAM/STARE observing modes ([4]) were originally designed for mapping compact sources, i.e. those which are smaller than the field of view of SCUBA-2 (approximately 8 arcmin). Note that these modes have not been commissioned at the time of writing and it is thought that STARE mode will not work.

The workflow for processing DREAM and STARE data may proceed in one of two ways: a simplified workflow using images calculated by the data acquisition (DA) at the time of the observation, or the full workflow starting from the raw data. The full procedure is described below, and steps which are unnecessary in the simplified workflow are noted.

The workflow proceeds as follows:

(1) Apply flatfield solution (full only);

(2) Remove atmospheric emission;

(3) Correct for atmospheric extinction;

(4) Calculate images from time-series (full only);

(5) Combine images to create output mosaic.

G.0.1 DA images

The images calculated by the DA are stored as NDF extensions in the raw data files under `.MORE.SCU2RED.In` where `n` is an integer (non-zero-padded). Each image is an average of the data in typically 1 second. Note the final image is created from whatever samples remain if the duration of the file is not an integer number of seconds. Each image has its own FITS header associated with it which contains values which vary between images.

G.0.2 Applying the flatfield correction

This section describes how to apply the flatfield solution to raw data. This step is not necessary for the simplified workflow.

A FLATFIELD observation records variation in the sensitivity of each bolometer. Ideally, every bolometer gives the same reading when exposed to a calibrated radiation source. In practice, each one responds in a slightly different way.

The flatfield response for each sub-array is derived at the telescope from a dedicated FLATFIELD observation. This solution is written to all subsequent data files until a new solution is derived (see Section 3.2.1 above).

Bolometer output is modified by the gain of the individual bolometer and its offset from a one-to-one relationship with the input radiation. Thus the output power from bolometer i is related to the input (optical) power by:

$$P_{\mathrm{out},i} = G_i P_{\mathrm{in},i} + O_i. \quad (1)$$

where G is the gain and O is the offset.

The SMURF routine that applies the flatfield correction is called flatfield. Since the flatfield information is already included in each subscan, multiple files can be flatfielded with a single call to the flatfield routine; there is no need to process files from different sub-arrays separately. Dark frames should be included in the list of input files in order to correct for bolometer zero-point drifts. Note that only flatfielded data (non-dark) files are written out, and that the naming scheme will change. The list of output files given is truncated at the appropriate number. The list of files actually written to disk is obtained via the OUTFILES parameter if specified.

G.0.3 Removing the atmosphere

This section describes how to subtract the signal from the atmosphere (or sky) from your observations. This step is required for both the full and simplified workflows.

The sky is highly emissive (bright) at submillimetre wavelengths[1], and the radiation received from an astronomical source is only a very small fraction of the radiation received from the atmosphere. Typically the atmospheric signal is a factor of 10^5 times greater than that from the source (see Figure 5 for a pictorial representation). In addition, the atmospheric emission varies significantly over the timescales of a typical observation, further swamping any contribution from the source. Fortunately, the emission from the sky is correlated over the SCUBA-2 field of view, which enables simple sky removal techniques to be used effectively[1, 11].

In a single sample the signal collected from an astronomical source is negligible in comparison with the contribution from the atmosphere. Many samples are required to collect a statistically significant signal from the source.

The SMURF routine that removes the sky is called remsky and has several options for fitting and removing the atmospheric contribution to the signal. All take advantage of the fact that the sky signal is correlated between adjacent bolometers.

The atmospheric signal can be fit in the following ways:

- Calculate the mean value of the signal;
- Fit a gradient in elevation only;
- Fit a plane of arbitrary orientation.

Furthermore, remsky can make use of the data from all available sub-arrays simultaneously to make a better estimate of the atmospheric emission (see the GROUP parameter).

For simple cases, subtracting a mean value may be sufficient, though it is not recommended for bright sources, as the mean will be biassed too high by the presence of the source. A more sophisticated solution may be obtained by fitting a plane to the data. The plane may have arbitrary orientation (in azimuth and elevation) or may be fixed so that the slope is in elevation only.

G.0.4 Correcting for atmospheric extinction

This section describes how to correct your data for atmospheric extinction, the loss of radiation from the source as it travels through the atmosphere. This step is required for both the full and simplified workflows.

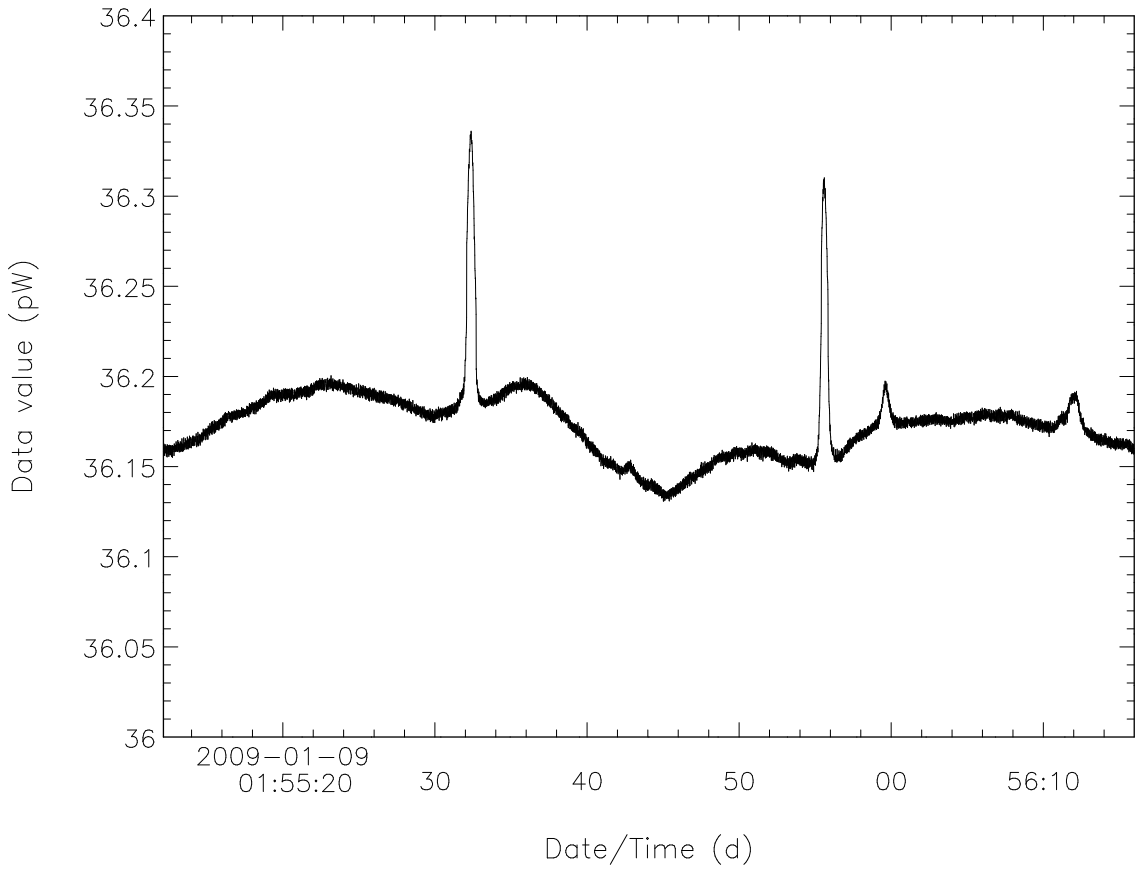


Figure 5: Illustration of the relative magnitudes of atmospheric and source signals at submillimetre wavelengths. Data are from 850 μm observations of Jupiter with SCUBA-2. The sharp features occur when Jupiter is seen by the bolometer. The increase in optical power from Jupiter is $\sim$0.5%. Most sources will be many thousands of times fainter.

The observed signal from an astronomical source decreases as the atmosphere along the line of sight increases. At lower elevations, there is more atmosphere along the line of sight between the source and the telescope.

The radiation received by SCUBA-2 is the sum of the radiation from the astronomical source (represented by $I_{\rm src}$ in Equation 2 below) and the radiation from the atmosphere ($I_{\rm atm}$). However, $I_{\rm src}$ is diminished by the optical depth of the atmosphere (τ) as follows:

$$I_{\rm obs} = I_{\rm atm} + I_{\rm src} \exp(-\tau). \quad (2)$$

The optical depth is usually quoted as the value at the zenith ($\tau_{\rm zenith}$), but you need the value of tau at the same elevation as your observations ($\tau_{\rm obs}$). For a plane-parallel atmosphere, $\tau_{\rm obs} = A \times \tau_{\rm zenith}$, where airmass ($A$) is a measure of the total atmosphere along the line of sight.

Airmass is defined as $1/\cos\theta$, where θ is the zenith angle. The zenith angle is the vertical angle measured from the zenith ($\theta = 0^\circ$) towards the horizon ($\theta = 90^\circ$) and is thus equal to $90 - \phi$, where ϕ is the elevation. You can now calculate $\tau_{\rm obs}$ from $\tau_{\rm zenith}$ and ϕ as follows:

$$\tau_{\rm obs} = \tau_{\rm zenith} / \cos(90^\circ - \phi). \quad (3)$$

Using Equation 3, rewrite Equation 2 as:

$$I_{\rm obs} = I_{\rm atm} + I_{\rm src} \exp\left(\frac{-\tau_{\rm zenith}}{\cos(90^\circ - \phi)}\right). \quad (4)$$

The elevation angle of each bolometer can be calculated, and given $\tau_{\rm zenith}$ remsky calculates the $\tau_{\rm obs}$ at the elevation angle of each bolometer.

The zenith optical depth can be obtained via three methods:

(1) Skydips: The zenith optical depth is derived from a SKYDIP observation, in which the signal at each wavelength is recorded over a range of airmasses. SMURF does not yet have the capability to process SCUBA-2 skydips.

(2) CSO tipping radiometer: The Caltech Submillimeter Observatory (CSO) radiometer measures $\tau_{\rm zenith}$ from skydip observations at 225 GHz. The CSO radiometer records a new $\tau_{\rm zenith}$ every 10 minutes and values are stored in the FITS headers corresponding to the start and end timestamps of the file (`TAU225ST` and `TAU225EN`). SMURF converts CSO $\tau_{\rm zenith}$ to $\tau_{\rm zenith}$ at 450 or 850 μm, relying on an empirical relationship between $\tau_{\rm zenith}$ at 225 GHz and $\tau_{\rm zenith}$ at 450 or 850 μm.

(3) JCMT Water Vapour Monitor: The JCMT Water Vapour Monitor (WVM) provides opacity estimates every 1.2 seconds by measuring the spectral shape of the 183 GHz water line along the telescope line-of-sight, and then converting this to an effective $\tau_{\rm zenith}$ at 225 GHz. Since the measurement is aligned with the submillimetre beam this method circumvents the need to apply the plane parallel approximation (which breaks down at the horizon). The raw WVM data are stored in the JCMT state structure, though the values at times corresponding to the start and end of the file are stored in the FITS headers (`WVMTAUST` and `WVMTAUEN`).

Optical depth measurements derived from the WVM are most useful in variable atmospheric conditions where the CSO radiometer may not be giving sufficiently-frequent readings.

The SMURF routine that corrects for atmospheric extinction is called extinction. The extinction routine can use $\tau_{\rm zenith}$ determined by all of the above methods, and obtains values automatically from the FITS headers for the WVM and CSO cases or uses the raw WVM data if desired.

The user can choose how accurate a correction they wish to apply using the `METHOD` parameter. By default, extinction will assess the magnitude of the correction and if it does not change significantly across the field of view, then a single value is used (corresponding to the elevation of the tracking position). If there is a significant variation, then extinction will apply a correction to each bolometer using the elevation of that bolometer. It is possible to enforce either of these options by setting `METHOD` to `QUICK` or `FULL` respectively.

Note that sky subtraction must take place before the extinction correction is applied. extinction checks the history to see if remsky has been run and exits with an error if not. However, if an alternative method of sky subtraction has been used (such as using KAPPA csub to subtract a mean or median value) then the `HASSKYREM` may be set to `TRUE` to override the history check by extinction.

G.0.5 Calculating images from time-series data

This step is not required for the simplified workflow.

SMURF provides separate routines for calculating images from STARE and DREAM data. STARE images are calculated with starecalc; DREAM images are calculated with dreamsolve.

In starecalc the user may specify how many samples to average to create images (the default is to calculate averages once a second).

dreamsolve requires the DREAM weights file specified in the FITS header keyword `DRMWGHTS` to reside in the current directory. Data from each sub-array require a corresponding weights file. See section G.0.7 below for details on calculating alternative weights files.

G.0.6 Combining the images

SMURF does not have any image mosaicking tasks itself. This step is handled by routines in KAPPA and CCDPACK. Suitable tasks are wcsmosaic in KAPPA and a combination of KAPPA wcsalign (to align all the images to a common coordinate frame) followed by CCDPACK makemos.

See the SCUBA-2 cookbook (SC/21) for more detailed information about mosaicking SCUBA-2 images.

G.0.7 Using alternative DREAM solutions

This step is not usually necessary for either workflow but is included for completeness.

The DREAM observing mode moves the secondary mirror in a pattern which causes adjacent bolometers to observe the same region of the sky multiple times [4, 7]. The data are then described by a series of simultaneous equations which are solved via a matrix inversion (using singular value decomposition). Since the pattern traced out by the secondary mirror is known, the inverse of the matrix can be pre-calculated (a time-consuming task) for a given output grid and applied to data at the time of observation (quick).

It is *strongly* recommended that users become familiar with the details of DREAM [7] before using weights other than the default for science data processing.

The user may wish to experiment with different regridding schemes or there may be other reasons, not known at the time of observation, which require the inverse matrix to be re-calculated. The task for doing this is called dreamweights, and this allows the user to specify different output grid parameters. The calculation of the inverse is moderately time-consuming (compared with calculating the images) and takes of order 5–10 minutes on current hardware. The output file from dreamweights is the weights file required by dreamsolve. KAPPA fitsedit may be used to specify weights files which differ in name from the default. Note that this process must be repeated for each sub-array for which data exist and that data which are to be combined should all use the same grid.

G.1 Moving sources

Mosaicking of DREAM and STARE images of moving sources will require the WCS attributes `AlignOffset` and `SkyRefIs` to be changed to 1 and 'origin' respectively. Use the KAPPA task wcsattrib to modify these attributes.

H Rebinning map-maker

The processing of SCAN data with the rebin method proceeds in a multi-step approach that follows the method listed above for DREAM and STARE data using makemap. A one-pass rebinning algorithm is applied to the data, which regrids the signal from each bolometer on to a map in the desired co-ordinate system.

As outlined in G, the procedure for processing SCUBA-2 data follows these steps:

(1) Apply the flatfield correction (see Section G.0.2);

(2) Clean the data set using sc2clean. This can remove spikes, DC steps and low- and high-frequency components in the data.

(3) Remove the contribution of atmosphere to the signal (see Section G.0.3);

(4) Correct for atmospheric extinction (see Section G.0.4);

(5) Make a map using the rebin method.

The rebinning algorithm shares the observed signal from a single bolometer between neighbouring output map pixels, as determined by the pixel-spreading function. The value of each pixel in the map is the sum of the signal from every bolometer that observed at that position.

There are many options for the pixel spreading scheme (see the wcsmosaic documentation for a summary). The fastest option is to use the nearest-neighbour pixel spreading scheme ('Nearest'), which assigns the whole signal from one bolometer to the nearest map pixel. This option does not always produce smooth maps (if the map is poorly sampled, for example, but it produces a robust estimate of the noise. In the limit of a large number of 'hits' per pixel, and with a pixel size sufficiently small compared to the beamsize, this method is optimal.

For smoother maps (with poor sampling, for example), the 'Linear' scheme can be used to share the input signal equally among the four nearest map pixels. There are more complicated options available, e.g., Sinc-type functions, or Gaussian distributions. These methods produce aesthetically pleasing maps, at the risk of introducing correlations into the noise and additional smoothing to the map.

Figure 6 shows an example map created from simulated observations of an extended source with bright features. It was created using the rebin method and the nearest-neighbour pixel spreading scheme.

Note the dark regions around the bright sources. They are negative bowls introduced by the sky subtraction method (Section G.0.3). The random dots are cosmic ray spikes which are not removed by this particular workflow.

I SCUBA-2 Simulator

SMURF has the capability to produce simulated data from SCUBA-2 with the sc2sim task. The simulator samples an input astronomical image in the presence of a model atmosphere and

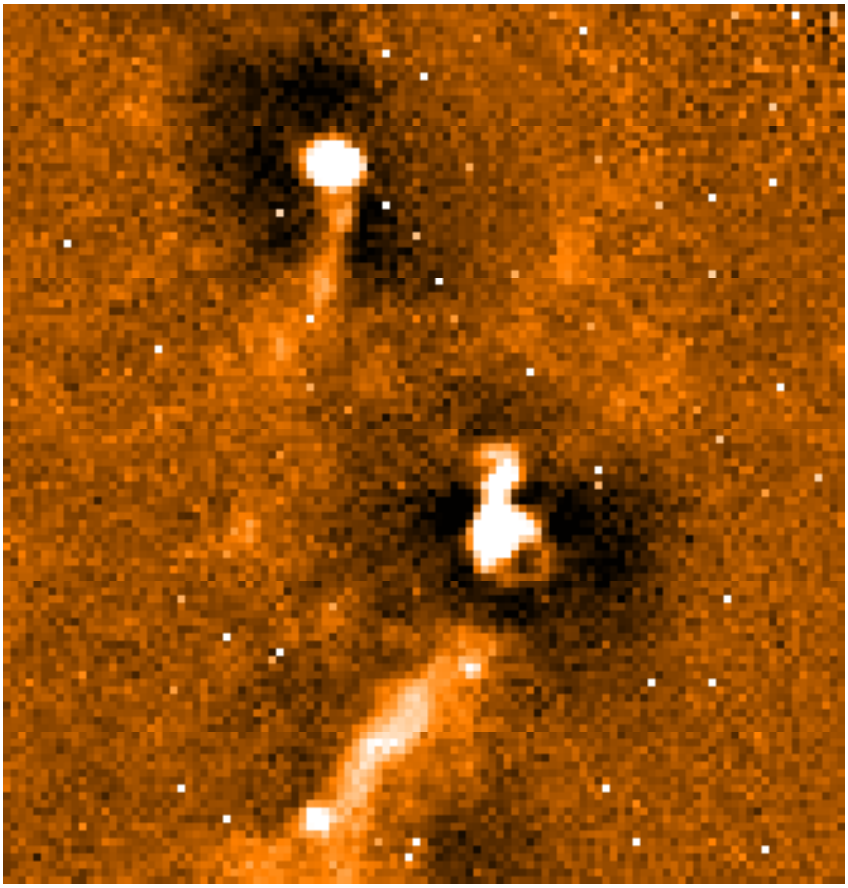


Figure 6: Image reconstructed from simulated data using the REBIN method. Note the deep wells around bright sources and the numerous spikes caused by (simulated) cosmic rays.

other sources of noise. Output files are written in the same format as real data from SCUBA-2 with complete flatfield, WVM, WCS and JCMT state structure information. These files can be processed with other SMURF tasks. Model atmospheres may be generated with the skynoise task.

The simulator supports the primary observing modes: DREAM, STARE and SCAN along with FLATFIELD, POINTING and FOCUS observations. Rudimentary support exists for the SCUBA-2 polarimeter though this is untested. Instrument apertures may be set (offsets which adjust the tracking position to a specific sub-array), and microsteps (small offsets from the tracking position) are fully supported. Simulating the motion of the major Solar System bodies (Venus, the Moon, Mars, Jupiter, Saturn, Uranus and Neptune) is supported.

For SCAN observations, the simulator has two modes controlled by the SIMTYPE parameter. A FULL simulation generates simulated complete astronomical data including the effect of the atmosphere. A WEIGHTS simulation generates data which may be used to assess areal coverage and 'hit' density (i.e. the number of times a point on the sky is observed) for different mapping strategies or the impact of non-working bolometers. Several different scanning models are supported: boustrophedon (conventional back-and-forth raster scan), two variations of 'pong' (or box-scan) with either sharp or gentle turnarounds and a lissajous pattern. [REF TO ANA SCAN DOC]

I.1 Simulator workflow

The simulator requires some preparatory work before data can be simulated. The basic procedure is outlined below:

- Obtain an astronomical sky image;
- Calculate a model atmosphere;
- Calculate a flatfield solution;
- Decide on simulation parameters;
- Run simulation.

It is not necessary to recalculate a model atmosphere or flatfield solution for every simulation, even for different input sky images.

Example files are installed as part of SMURF and may be found in the `$STARLINK_DIR/share/smurf` directory. An explanation may be found in the `README` file in that directory.

I.1.1 Astronomical image

The astronomical image may be any suitable image, provided it has WCS information. It should be as large as the size of the map to be made. The WCS is used unless the source is a moving object (listed above). The image must be an NDF. It is recommended that the image pixel scale be set to be much less than the likely output map pixel spacing (typically 3 arcsec at 850 μm, 1 arcsec at 450 μm) to avoid imaging and sampling artefacts.

I.1.2 Model atmosphere

The model atmosphere is calculated with the skynoise task. The atmosphere is modelled as a Kolmogorov turbulent thin-screen [11] with a characteristic turnover frequency and scaling law. Different models may be generated for the same parameters using a random number seed (specified or calculated internally using the system clock). The parameters may be modified though the user should be familiar with the details of this model of the atmosphere before exploring the parameter space too widely.

The model atmosphere calculated is generic and may be used for simulations at both 850 μm and 450 μm. The model is scaled at the time of simulation according to the particular wavelength.

I.1.3 Flatfield simulation

Before a data simulation can take place, the flatfield solution for each sub-array must be calculated. This is achieved by specifying a special observing mode called `heatrun` and running a simulation.

The flatfield solution is written to a file using a naming scheme which follows that for raw data. The convention is `s[4|8][a-d]heatYYYYMMDD_NNNNN.sdf` where `s[4|8][a-d]` is the sub-array name (e.g. `s8a`), `heat` indicates a flatfield observation, `YYYYMMDD` is the date of observation and `NNNNN` is a zero-padded, 5-digit observation number.

The simulator relies on this naming scheme for reading the flatfield information, and the files must be present in the current directory.

I.1.4 Running the simulator

The simulator has a large number of parameters which may be freely adjusted. The list of parameters is divided between two groups: simulation parameters (usually telescope- and instrument-specific); and observation parameters (properties of the observation in question). These are specified using the SIMPAR and OBSPAR ADAM parameters to sc2sim. For convenience, the parameters are stored in plain text files and are passed in using the ^ scheme mentioned above in Section 1.2.5. Example input files for all major observing modes may be found in the $STARLINK_DIR/share/smurf directory.

The main observation parameters (OBSPAR) of interest are the RA/Dec of the source, wavelength, observing mode (obsmode), observation date and duration of observation (which in the case of SCAN observations, is determined by the size of the area to map). Planets may be specified by name. The RA/Dec must match that in the astronomical image. If the source is not above the horizon, the simulator will exit with an error message. The date must be specified as a UT modified Julian day number. As a point of reference, 20090701T00:00:00 corresponds to MJD 55013.

The main simulation parameters (SIMPAR) are the sub-arrays to simulate data for, the names of the atmosphere model and astronomical source files and the zenith opacity (at 225 GHz). The zenith opacty is converted to a value at the appropriate wavelength using the empirical relations derived for SCUBA by Archibald et al. (2002). Similar relations for SCUBA-2 will be derived.

Data for all sub-arrays (at the specified wavelength) may be generated in a single run. The size of the output files is governed by the MAXWRITE parameter for sc2sim which specifies the number of samples to write per file. Multiple observations may be simulated by setting the OVERWRITE parameter to FALSE (the default behaviour is to overwrite existing files).

A word of caution. Since the data output from the simulator mimics the real instrument, it is possible to generate many GB of data which may take many minutes to hours of CPU time. The sc2sim parameter SIMSTATS tells the simulator to estimate the properties of the simulation including the duration of the simulation, quantity of data, memory requirements and CPU time. The results are printed to the screen, allowing the user to modify the simulation parameters if necessary.

I.1.5 A note on DREAM simulations

For STARE simulations, the images are calculated at 1-second intervals at the time the simulator is run and written to the output files. For DREAM the images must be reconstructed after the simulation has run. This means that the DREAM weights file must be calculated using dreamweights as desribed in Section G.0.7. Likewise, the images are calculated using dreamsolve.

I.2 Processing simulated data

The processing of simulated data proceeds exactly as for real SCUBA-2 data and depends on the observing mode. See the respective workflows in Sections G and 3.3 above. Note that dark frames are not produced by the simulator and do not need to be included in any processing (there is no drift in the model bolometer response).

FOCUS observations are designed to be processed by the ORAC-DR pipeline rather than SMURF, as their analysis involves multiple steps. POINTING observations are equivalent to short science observations and should be processed in the same way.