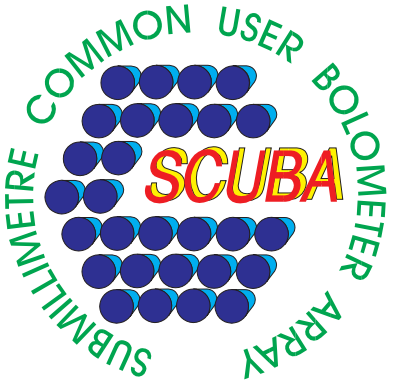
SC/11.2

Starlink Project
Starlink Cookbook 11.2

G. Sandell, N. Jessop, T. Jenness
Joint Astronomy Centre, Hilo, Hawaii

29th October 2001

The SCUBA map reduction cookbook



Contents

List of Figures

1 Introduction

This is the revised guide on how to reduce, and analyze maps obtained with SCUBA using the off line SCUBA reduction package SURF [2] and the Starlink versions of KAPPA[4], FIGARO[5], GAIA[6], and CONVERT[10].

The easiest way of using these packages is to run-up ORAC-DR, a general purpose pipeline for reducing data from any telescope. A set of data reduction recipes are available to ORAC-DR for use when working with scuba maps, these recipes utilize the SURFand KAPPA packages.

This cookbook makes no attempts to explain why and how, for that there is a comprehensive Starlink User Note (**SUN/216**) written by T. Jenness and J. Lightfoot, which properly documents all the software tasks in SURF, which should be consulted for those who need to know details of a task, or how the task really works.

2 The really easy way

Given that ORAC-DR is what you will use at the telescope, and that it provides the easiest introduction to the various steps necessary to reduce your data, we will deal with it first. If you are at the telescope log into `mamo` (if you are working on another machine just `ssh mamo`), and then type:

```
% oracdr_scuba
% oracdr
```

If you are at your home institution or at the JAC, you should use the new x-windows gui. Simply type:

```
%xoracdr
```

and the interface shown in Figure 1 should pop-up. Fill in the appropriate date, the location of the raw data files (see below if you are at the JAC and unsure) as well as where you want the reduced data to go, and press Start!

Having done one or other of the previous set-ups you should then find a dialogue window pops up, with explanatory messages about what the pipeline is doing, followed shortly by a xwindow which will display all the key stages of the reduction. You will find that ORAC-DR starts working through the observations one by one ignoring some (like the focus and align measurements) but reducing most. For each distinct type of observation (pointing, skydip, jigglemap) ORAC-DR has a recipe it uses to reduce the data. If you miss a stage (maybe you went out to make a coffee) or you want to examine something in detail, don't despair look at the output directory for the reduction, you will find many of the final reduced files and intermediate stages eating into your disk space. From the name of the file you should be able to work out at what stage of the reduction it was produced – the final 're-binned images' should finish in "`_reb`".

Given this is a cookbook for reducing map data, let's look at the two recipes for reducing jiggle-maps, and emerson2 maps. You can find these by typing "`ls $ORAC_DIR/recipes/SCUBA`" (if the environment variable is not set type "`oracdr_scuba`"). The jiggle map recipe reads,

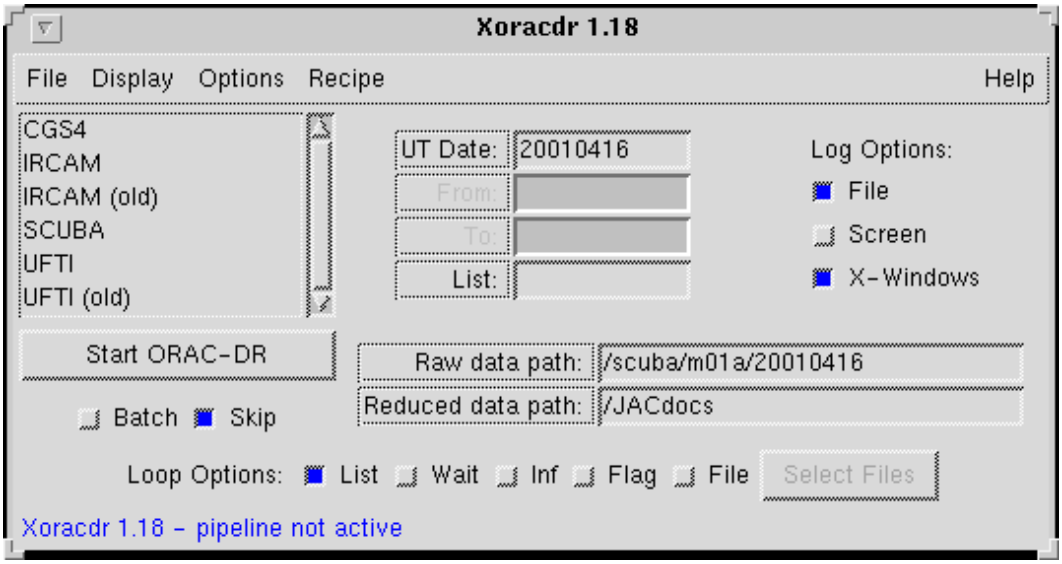


Figure 1: The `Xoracdr` gui for ORAC-DR. All the information needed to start the data reduction (ie. where the data is, what instrument is being used, what ut date) is prompted for.

```
=head1 NAME

SCUBA_JIGMAP - Standard reduction for jiggle map data

=head1 SYNOPSIS


=head1 DESCRIPTION

This is the standard recipe to use for reduction of SCUBA
jiggle map data.


=cut

_PRE_PROCESS_

_FLAT_FIELD_

_SET_BAD_PIXELS_

_EXTINCTION_CORRECT_

_CLIP_BOLOMETERS_ NSIGMA=5.0

_REMOVE_SKY_NOISE_JIGGLE_  BOLOMETERS=r3 MODE=median

_REBIN_FRAME_ PIXEL_SIZE=3.0 REBIN_METHOD=GAUSSIAN

_FIND_CALIBRATION_MAP_
```

```
_CALIBRATE_DATA_

_REBIN_GROUP_ PIXEL_SIZE=1.0 REBIN_METHOD=LINEAR

_DELETE_TEMP_FILES_ KEEP=_reb,_ext,_sky,_cal
```

while the scan map recipe reads (omitting the verbose header information),

```
_PRE_PROCESS_

_FLAT_FIELD_

_SET_BAD_PIXELS_

_DESPIKE_SCAN_

_EXTINCTION_CORRECT_

_REMOVE_SCAN_BASELINE_

_REMOVE_SKY_NOISE_SCAN_

# Comment this if the processing of the individual frame is
# not required.
_REBIN_FRAME_ PIXEL_SIZE=3.0 REBIN_METHOD=LINEAR

_REBIN_EM2_GROUP_ PIXEL_SIZE=3.0 REBIN_METHOD=GAUSSIAN

# Tidy up
# Need to make sure that the _rlb file is kept for the
# sky removal and that the _sky file is kept for the group processing.
_DELETE_TEMP_FILES_ KEEP=_rlb,_sky,_reb
```

Both read surprisingly like English, each line in the recipes is a step to be done in the data processing (or in the ORAC-DR parlance a call to a 'primitive'). Also it is worth noting that the first few steps are nearly identical. The pre-processing, flat-fielding, despiking and extinction correction is the same for both jiggle, and scan map data, they only differ in the despiking, removal of baselines and skynoise, and in the rebinning. Note many of the steps have variables which can be set to customize the recipe, i.e. N_SIGMA on the clipping, and PIXEL_SIZE in the rebinning. It is possible to accurately reduce your data using ORAC-DR alone, using the supplied variables in the recipe, customizing the recipes to use the wide range of set primitives issued with ORAC-DR or even by altering primitives or writing new ones (though this requires you to become acquainted with object-orientated Perl). At the very least ORAC-DR should be run twice, once at the summit when the data is being taken, and once at home to give you something to compare to. However the next sections describe how to reduce the data at the 'bare-bones' level using SURF and the standard Starlink data-reduction packages.

3 The difficult way

Given the lack of detailed documentation on ORAC-DR it is difficult to not see it as a black-box. Most people who use it is as their preferred method of data reduction are likely to have spent some time with somebody who knows a lot about it, or to have dug down into the primitive code to see what is actually done on a step by step basis. The new `xoracdr` interface will to some extent make the reduction less opaque, it can step you through the recipes. However most astronomers, being by nature suspicious and distrustful, are likely to want to reduce their data at least once in a step by step manner. The rest of the cookbook outlines how to do this. It is by no means a waste of space for the ORAC-DR user, ORAC-DR calls on the SURF routines to reduce the data.

First log in on a unix workstation and create a directory where you want to store your reduced data. Next type

```
% surf
% kappa
% figaro
% convert
```

This starts up the SCUBA software and the main Starlink packages needed for the data reduction.

The next thing you will need to do is to find the data and create logs of your observing run, so that you know what scan numbers to reduce.

3.1 How to find and access SCUBA data?

All SCUBA data are stored on disk at JAC and also for a shorter period at JCMT. The data are initially stored on the summit VAX, and copied to a unix disk (mounted as `/jcmtarchive`) after each observation is completed. Each night a new subdirectory is created using the convention of year, month and UT date, i.e. 19980215, would contain data for the night between the 14th and 15th of February 1998. As of now data obtained on for example the evening shift of February 15 are stored on the UNIX disk in

```
/jcmtarchive/19980215
```

and copied to Hilo the following day, where they reside in a protected UNIX directory. If you are a visiting astronomer, it may be wise to copy all your files into your local directory, so that you later can transfer them to your home institution. Another way to obtain SCUBA data is to search the JCMT archive. If you find what you are looking for, please mark and retrieve the files and within an hour you can start reducing data. Since the source files are not tagged to calibration information, you will have to do an additional search for the date or dates that the files originate from and retrieve skydips, pointing and calibration data, so that you can calibrate your data sets.

In the following we assume that you have copied the files into your local directory called `/home/user/scubadata`. To start the data reduction you will therefore need to set up some environmental variables and get a listing of your data.

```
% setenv DATADIR /home/user/scubadata
% setenv SCUBA_PREFIX yyyymmdd
% setenv HDS_SCRATCH /tmp
```

Here yyyy stands for the year, mm is the month (as a numeral) and dd is the UT date for the night you want to reduce, i.e. 19980401 would be the night between March 31st and April 1st in 1998. For the best performance you should try to ensure that your working directory is on a local disk. At the very least, you should make sure that HDS_SCRATCH is local since this is where temporary files will be written by Starlink applications.

You are now ready for creating a log of all maps obtained during the night by typing:

```
% mapsum -all -demod > Dec8.maps
```

Here the summary information about all the maps in our directory is piped into a text file, which we called Dec8.maps. You can choose any name you want. This is a simple ASCII-file, which you can edit and print. If you want to be systematic in your data reduction, you can edit the file to include your reduction notes. The same way one can make summaries of photometry (photsum) and pointsum, to get a log of the pointing, which you will need if you plan to correct your maps for pointing drifts. You may find it useful to include a complete log of the night with sculog (or obssum), if you want to make sure that you know when the telescope was focused and when noise measurements were done, although the latter you can easily find with scunoise.

4 Jiggle maps

The basic SCUBA jiggle map reduction consists of 4 steps, each of which is done with a separate task: reduce_switch, flatfield, extinction and rebin. For a complete reduction, you will also need some or all of the following tasks: change_flat, change_pointing, change_quality, despike, remsky, scuclip, scuover, and scunoise. For interactive despiking you will also need sclean or dspbol. For scan map data one additionally needs the tasks despike2, scan_rlb, and restore or remdbm.

4.1 The bare minimum

We now try our first map (scan 86), which is a single observation (3 integrations) of IRC+10216, one of our secondary calibrators, obtained on December 8 1997. Normally we would do the first three steps with the script scuquick, i.e. scuquick -tau=0.185 -sub=lon IN 86 out=i86, but here we do it step by step, so that you can see what is involved. First we execute reduce_switch, and flatfield:

```
% reduce_switch
IN - Name of input file containing demodulated data > 86
SURF: Opening 19971208_dem_0086
SURF: run 86 was a MAP observation of object IRC+10216
SURF: file contains data for 2 switch(es) in 4 exposure(s) in 3
integration(s)
in 1 measurement(s)
OUT - Name of output file to contain reduced switch data /'o86'/ > i86
```

```
% flatfield
IN - Name of input file /@o86/ > i86
SURF: run 86 was a MAP observation of IRC+10216
OUT - Name of output file /'i86_flat'/ >
SURF: applying flatfield from lwswphot.dat
```

Next we want to correct for the atmospheric attenuation of the signal. At present there are two measures of the atmosphere's opacity - $\tau_{\rm CSO}$, and skydips. A large amount of effort has gone into understanding the relationship between these two over the past few semesters and the results are presented in Archibald et al. [11]. The relationships found for the wideband filters now used are:

$$\tau_{850} = 4.02 \times (\tau_{CSO} - 0.001) \tag{1}$$

$$\tau_{450} = 26.2 \times (\tau_{CSO} - 0.014) \tag{2}$$

A careful observer will examine the $\tau_{\rm CSO}$ data for the night, note whether it appears stable or not (it commonly 'spikes' on nights with poor atmospheric conditions) and compare the results with the skydips obtained (one can use the SURF routine skydip to reduce the data if you want or use ORAC-DR). The fits made to 450-dips are not always particularly good, there seems to be a problem with the minimization routine, and so the recommended method of calculating opacity at 450 is always to convert from the 850 or $\tau_{\rm CSO}$. Which you use is certainly open to debate; the skydips may well have the advantage of being made at the same azimuth as your source, but the $\tau_{\rm CSO}$ is measured more regularly (every ten minutes or so). The preferred method used at the JAC is to fit a polynomial to the $\tau_{\rm CSO}$ data, and convert the value from *the fit* to opacity at 450 and 850, these fits are regularly made and available at the JCMT's calibration web page ORAC-DR will make use of these fits if one configures it to do so.

Before getting too involved in correcting for sky-opacity it is worth keeping in mind that at 850, and particularly for faint sources, the exact value is not necessary – other uncertainties are likely to dominate. We therefore adopt a value $\tau_{850} = 0.185$ for this observation. The extinction correction is applied by running extinction on the flatfielded data.

```
%extinction
IN - Name of NDF containing demodulated data /@i86_flat/ > i86_flat
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
SURF: file contains data for 4 exposure(s) in 3 integration(s) in 1
measurement(s)
SURF: observation started at sidereal time 9 41 05 and ended at 9 50
09
SURF: file contains data for the following sub-instrument(s)
 - SHORT with filter 450
 - LONG with filter 850
SUB_INSTRUMENT - Name of sub-instrument to be extinction corrected
/'SHORT'/ >
long
FIRST_TAU - First zenith sky opacity measured /0/ > 0.185
FIRST_LST - Sidereal time of first opacity measurement; hh mm ss.ss
/'0.0'/ >
SECOND_TAU - Second zenith sky opacity measured /0.185/ >
```

```
SECOND_LST - Sidereal time of second opacity measurement; hh mm ss.ss
/'0.0'/ >
OUT - Name of output NDF /'i86_lon_ext'/ >
```

Note that extinction allows you to supply two values of opacity measured at different times if you want – in this case we have bypassed this option. This is also where the long wavelength array gets separated from the short wavelength one. When we want the short wavelength data we have to run extinction again and choose *short* as the SUB_INSTRUMENT. Our data are now extinction corrected, but still in instrumental units (Volts). In order to have a feeling for the true signal and noise level in our data, we therefore need to apply a scaling factor, FCF (Flux Calibration Factor), that converts the instrumental units to *Jy/beam* or *Jy/arcsec*^{2}. Calibration is discussed in detail in Section 7. For just a quick look we ignore the intricacies of calibration and use nominal FCSs, which for the current filters (850 W & 450 W) is 220 and 310 Jy/beam/V for 850 and 450 μm, respectively. Since this map was taken with the old 850 μm filter, 850 N, we use a different calibration factor, FCF = 280 Jy/beam/V, which is more appropriate. To scale our extinction corrected data we use the KAPPA command cmult.

```
%cmult
IN - Input NDF data structure /@i86_lon_ext/ >
SCALAR - Multiplication constant > 280
OUT - Output NDF > i86_lon_cal
```

Here we gave the calibrated data set the extension `_cal`. Now we are ready to convert our extinction corrected and calibrated data onto a spatial grid using the SURF task rebin.

```
%rebin
REBIN_METHOD - Rebinning method to be used /'LINEAR'/ >
OUT_COORDS - Coordinate sys of output map; PL,AZ,NA,RB,RJ,RD or GA
/'RJ'/ >
SURF: output coordinates are FK5 J2000.0
REF - Name of first data file to be rebinned /'i86_lon_cal'/ >
SURF: run 86 was a MAP observation of IRC+10216 with JIGGLE sampling
SURF: file contains data for 4 exposure(s) in 3 integrations(s) in 1
measurement(s)

WEIGHT - Weight to be assigned to input dataset /1/ >
SHIFT_DX - X shift to be applied to input dataset on output map
(arcsec) /0/ >
SHIFT_DY - Y shift to be applied to input dataset on output map
(arcsec) /0/ >
IN - Name of next input file to be rebinned /!/ >
SURF Input data: (name, weight, dx, dy)
   -- 1: i86_lon_cal (1, 0, 0)

LONG_OUT - Longitude of output map centre in hh (or dd) mm ss.ss
format
/'+09 47 57.38'/ >
LAT_OUT - Latitude of output map centre in dd mm ss.ss format /'+ 13
16 43.7'/ >
OUT_OBJECT - Object name for output map /'IRC+10216'/ >
PIXSIZE_OUT - Size of pixels in output map (arcsec) /3/ >
SURF: Initializing LINEAR weighting functions
```

```
SIZE - Number of pixels in output map (NX,NY) /[70,65]/ >
OUT - Name of file to contain rebinned map /'i86_lon_reb'/ >
WTFN_REGRID: Beginning regrid process
WTFN_REGRID: Entering second rebin phase (T = 0.03516951 seconds)
WTFN_REGRID: Entering third rebin phase (T = 0.1326885 seconds)
WTFN_REGRID: Regrid complete. Elapsed time = 0.1400405 seconds
```

The resulting map can be viewed with KAPPA's display

```
% display axes clear i86_lon_reb
```

or by using `Gaia`. The resulting map does not look particularly nice, because we have not yet blanked out any noisy bolometers, done sky noise reduction or despiking.

5 A Fuller Reduction: Removing bad bolometers, sky-noise and spikes

5.1 Identifying and Blanking Noisy Bolometers

From the plot shown in Fig. 2 we can see that the map suffers from some sky noise, e.g. visible as the dark striping at the beginning of integration 2. The central bolometer, 19 (h7) shows a clear signal, and we do not see any really bad (noisy) bolometers, except perhaps bolometer 23. Even so, we first need to blank out bad bolometers. There are several ways to identify bad bolometers. The easiest way, although it may not pick up all noisy bolometers for your particular map, is to run scunoise, which is a GUI that allows you to plot and identify all noisy bolometers using noise measurements done during the run. If we do this for the night of 19971208 we find a total of 9 noise measurements and we can see that some bolometers come and go, but 23 is always noisy. From the most nearby noise measurement, #88, we find that 23, 32 and 37 have noise levels above 100 nV and 12 is also about twice as noisy as the majority of the array, which should have a noise level around 40 nV. Before we set these bolometers to bad we plot the bolometers with mlinplot to check that these bolometers are indeed noisy in our map.

```
% mlinplot i86_lon_ext absaxs=2 lnindx='21:32'
YLIMIT - Vertical display limits /[-0.002588027,0.09385466]/ >
DEVICE - Name of display device /@xwindows/ >
```

In the above example the bolometers 21 – 32 are plotted as a function of integration time (Fig. 3). Bolometer 23 is indeed the noisiest pixel, but 22, and 24 are noisy as well and actually worse than 32, which was picked up by scunoise. Fig. 3 shows strong sky noise, which makes it difficult to see the true noise level of the bolometers, and it is often wise to do a sky noise reduction (see below), so that one can more easily identify noisy bolometers. Go through the whole array by choosing suitable bolometer ranges with the parameter `lnindx`. In this example we choose to only blank out bolometer 23, which we set to bad using change_quality. Bolometers 8 and 14 also appear noisy, more so than 12, which we picked up in the noise measurement, but for the time being we let them stay. When we go through the bolometers with mlinplot we can also see a few spikes, which we will deal with shortly. Let us now set bolometer 23 with change_quality. Note that we need to surround the file name with both single and double quotes if we list more than one bolometer.

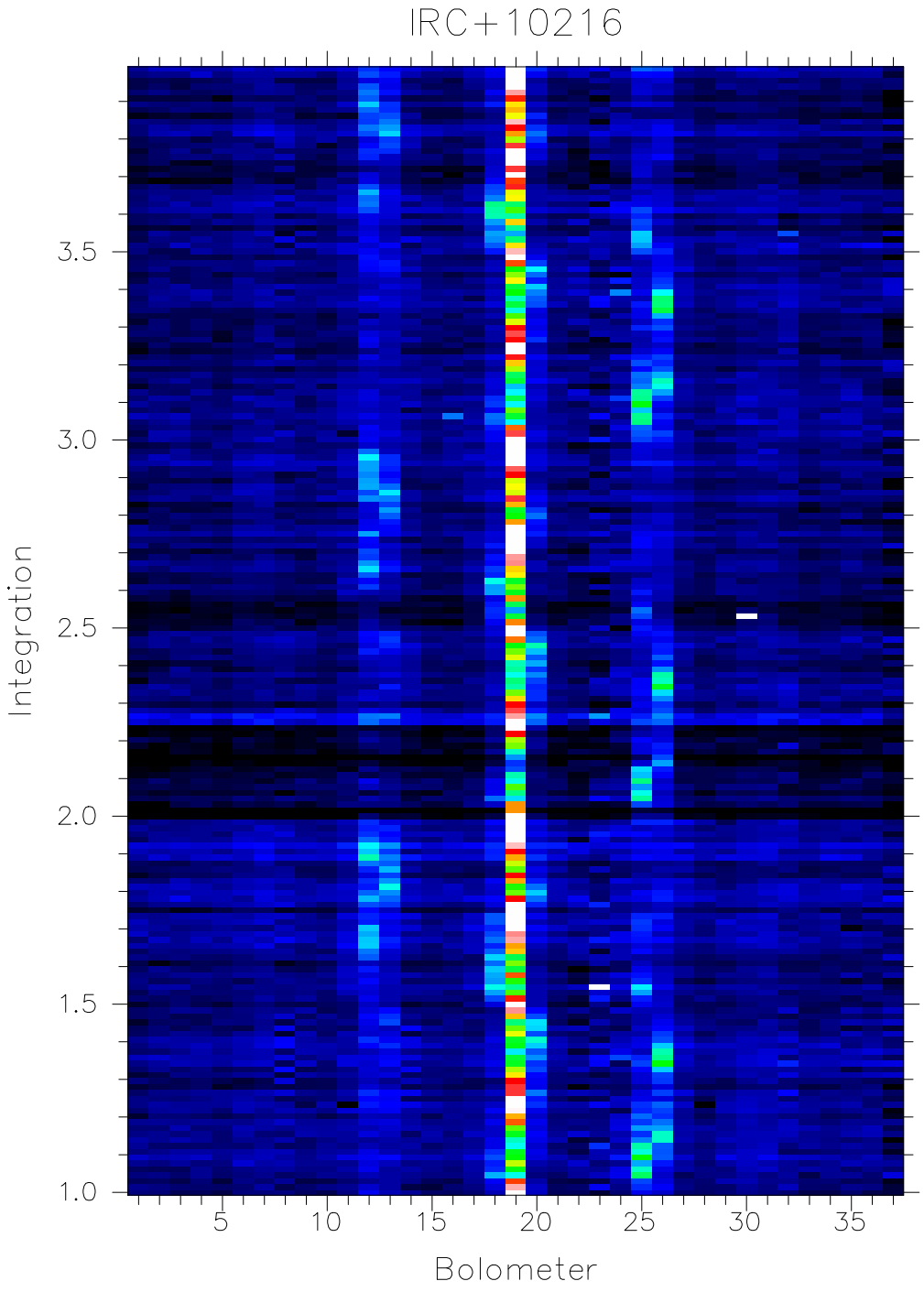


Figure 2: The extinction corrected data of IRC+10216 without despiking and sky removal. In this image the bolometers are on the x-axis and the y-axis is time. A strong signal is seen in the central bolometer (19) throughout the jiggle cycle, while neighboring bolometers (12, 13, 18, 20, 25, 26) pick up the signal only during part of the jiggle cycle. Sky noise variations, which occur at the same time for all pixels are easily seen, e.g. at the beginning of integration 2. This image is produced using the KAPPA display command with the command line option *fill=true.*

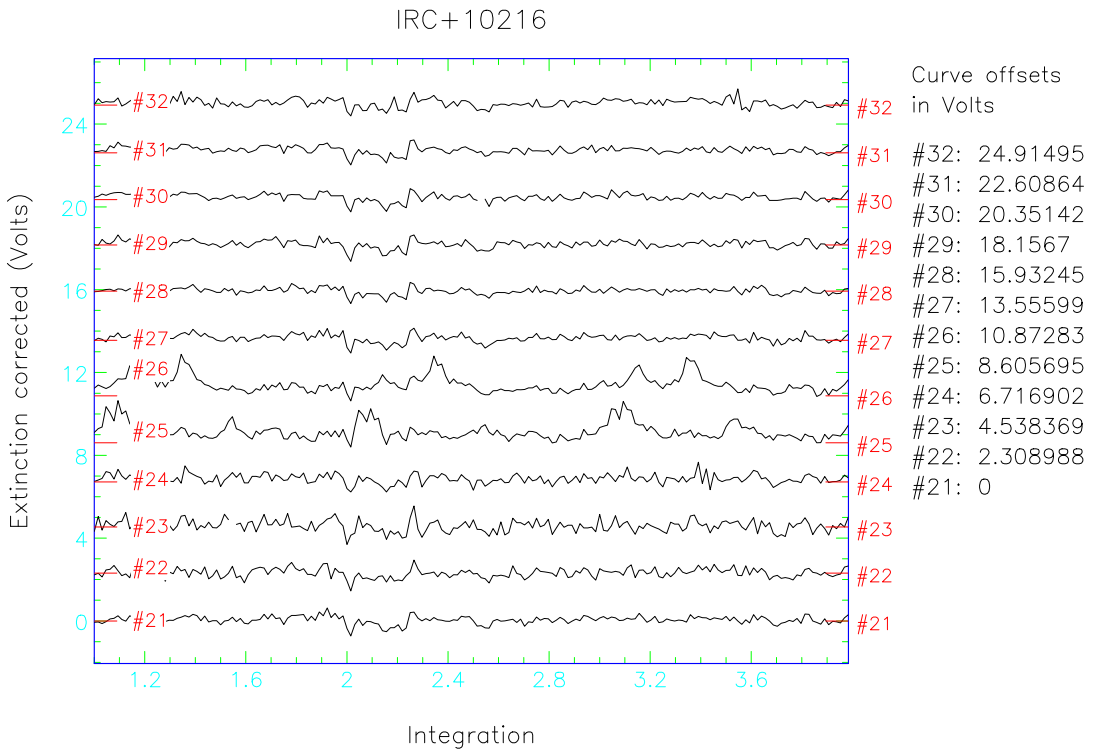


Figure 3: Using mlinplot to display a set of bolometers as a function of time is a good way to identify noisy bolometers. From this plot we can see that bolometers 22 – 24 are noisier than the rest of the sample. Note that bolometers 25 and 26, which are in the first ring surrounding the central bolometer, pick up the source at certain jiggle positions and this signal should not be confused with noise, which is random and erratic. One can also see the strong sky noise variations, which repeat the same signature for each bolometer, e.g. the dip in signal at the beginning of integration 2.

```
% change_quality "'i86_lon_ext{b23}'"
SURF: run 86 was a MAP observation of IRC+10216
SURF: file has data for 37 bolometers, measured at 192 positions.
 - there are data for 4 exposure(s) in 3 integration(s) in 1
measurements.
BAD_QUALITY - Set quality to bad? (No will set quality to good) /YES/
>
```

5.2 Initial sky noise removal

As we have already seen, sky noise can be a dominant noise source in a map, but as long as the sky noise is correlated over the array, it can be removed. One can do this in several different ways, but for a single short integration map one will always use remsky. In the automated SCUBA reduction Jenness et al. [14] use the median option in remsky and include all bolometers. Since we have a centrally symmetric source, we take the second ring, r2, of bolometers using the median option to avoid spikes that may be present in the data.

```
 % remsky
IN - Name of input file containing demodulated map data
/@o86_lon_ext/ >
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
OUT - Name of output file /'o86_lon_sky'/ >
BOLOMETERS - The Sky bolometers, [a1,a2] for an array
/['4','9','29']/ > r2
SURF: Using 12 sky bolometers
MODE - Sky removal mode /'median'/ >
Sky noise: 0.000602097 (192 points)
Adding mean background level back onto data (value=0.0003676229)
```

Note that the default behavior of this routine is to add the mean bolometer signal back in to data (this may not be what you wish to do) if it's not then type:

```
% remsky add=false
```

5.3 Initial Despiking

Let us now remove the worst spikes by using the SURF task scuclip to take out any spikes exceeding 5 sigma.

```
% scuclip
IN - Name of input file containing demodulated map data
/@i86_lon_sky/ >
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
OUT - Name of output file /'i86_lon_clip'/ >
NSIGMA - How many sigma to despike bolometers /5/ >
SURF: Removed 3 spikes
```

scuclip removed 3 spikes in the data, which will do for the moment.

Following despiking and sky removal one can then run rebin. The process of sky removal will significantly improve any maps which have a 'tiled' pattern due to varying sky emission between the two switch positions. Below we discuss in more detail how to most efficiently remove sky noise, which depends on what data sets we have.

5.4 Sky noise removal (a)- remsky.

In Section 5.2 we already used remsky to remove sky noise by choosing a ring of bolometers around the center and we could even have used all bolometers in the array. However, for short chop throws, or where we have an extended source or where the source is not centered in the array one should be more careful. A good way to identify suitable sky bolometers (i.e. bolometers without any source emission) is to convert the map into Az/El (assuming we chopped in Az) so that we can identify bolometers at the edge of the array, which are not affected by the chop nor by any source emission. We regrid the extinction corrected data with the SURF task rebin, setting the parameter OUT_COORDS to *az*. In the example below we use the same data set on IRC+10216 as we used before. We plot the map immediately with display using the option *faint* and overlay the bolometer array using scuover (use *noname* if you want the bolometers as numbers).

```
% display axes clear
IN - NDF to be displayed /@i86_lon_reb/ >
DEVICE - Name of display device /@epsfcol_p/ > xwindows
MODE - Method to define the scaling limits /'fa'/ >
Data will be scaled from -0.0012709096772596 to 0.013754481449723.
% scuover
DEVICE - Name of graphics device /@xwindows/ >
Current picture has name: DATA, comment: KAPPA_DISPLAY.
Using /home/sandell/dec8/i86_lon_reb as the input NDF.
SURF: file contains data for 4 exposure(s) in 3 integration(s) in 1
measurement(s)
```

IRC+10216 is extended at 850μm, because of its large CO envelope, and we therefore want to use edge pixels away from the chop direction (usually az) to avoid removing any real signal from the data. In our map of IRC+10216 (Fig. 4) one would therefore take bolometers 4,9,15 and 22 in the south and 16, 29, and 34 in the north. If we now run remsky on the extinction corrected data, we notice that the background level added back to the map was lower than when we used ring 2, which therefore picks up some of the faint extended emission surrounding IRC+10216.

```
 % remsky
IN - Name of input file containing demodulated map data
/@o86_lon_sky/ > o86_lon
_ext
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
OUT - Name of output file /'i86_lon_sky'/ >
BOLOMETERS - The Sky bolometers, [a1,a2] for an array /'r2'/ >
[4,9,15,16,29,34]

SURF: Using 6 sky bolometers
```

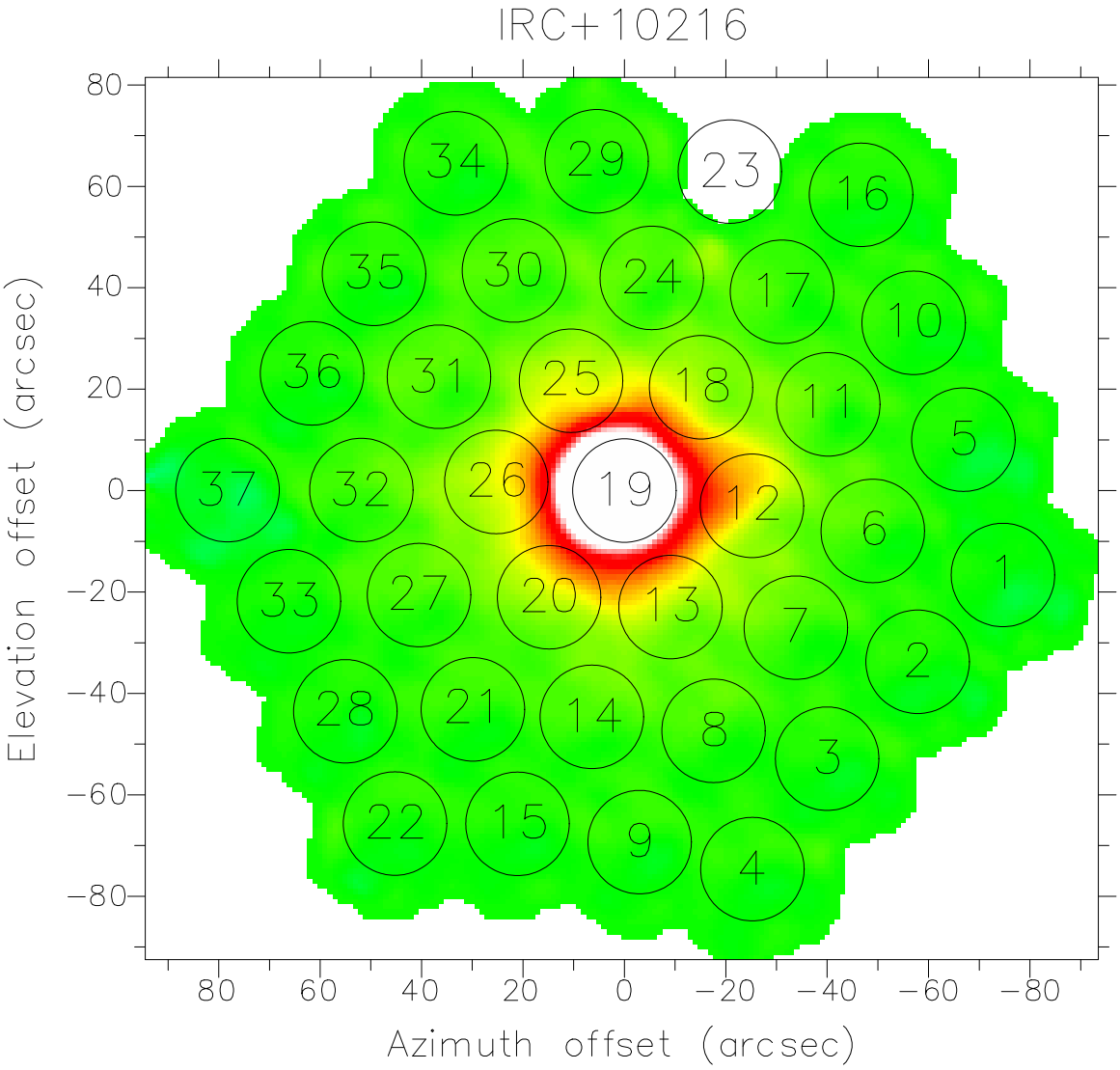


Figure 4: This image of IRC+10216 has been regridded in Az/el and overlaid with the bolometer array in order to allow us to identify bolometers for sky noise removal. For any extended source we want to avoid bolometers in the chop direction, and we therefore choose bolometers at the bottom and top of the map, i.e. for example 4,9 and 22 as well as 16, 29, and 34.

```
MODE - Sky removal mode /'median'/ >
Sky noise: 0.0005996103 (192 points)
Adding mean background level back onto data (value=6.3598651E-5)
```

5.5 Sky noise removal (b)- calcsky

Quite often, especially when we observe galactic sources embedded in dark or molecular cloud, it is impossible to find any bolometers that do not pick up source emission. As long as the

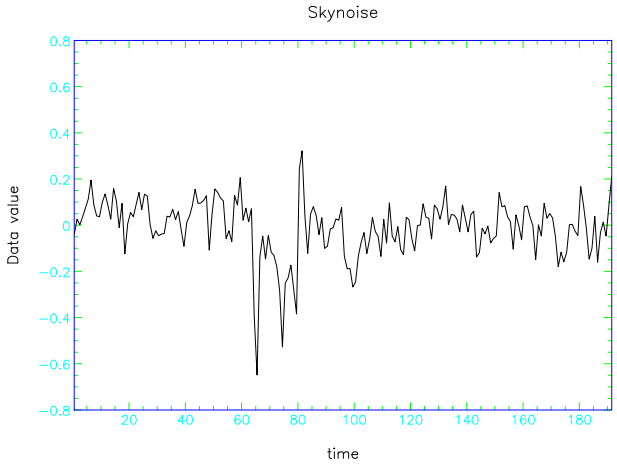


Figure 5: This plot shows the mean sky variations in scan 86 as computed by calcsky as a function of time. Therefore x = 65 corresponds to start of integration 2 in Fig 3. We can see that it has clearly picked up the strong sky variations that occur in the beginning of integration 2 and continued during part of integration 2. Since this is a calibrated image, we can see that the sky variations are extremely strong, and would easily mask a faint source.

source emission is relatively smooth, this will only affect the mean level in the map, and any base level that gets removed with remsky can be added back into the image using ADD = TRUE. But, if there is structure in the source emission over our sky bolometers, these variations will be interpreted as sky noise and therefore affect the morphology of our map. For extended sources we should therefore use calcsky. The task calcsky computes a model of the source, which it then subtracts from each bolometer to give an estimate of the sky variation, which is put into the file extension `.more.reds.sky`, which can be examined with e.g. linplot, e.g.,

```
% linplot i86_lon_cal.more.reds.sky device=xwindows
```

shows the computed sky noise variations for the scan 86, that we will re-analyze below (Fig. 5).

We can improve the model by adding data sets to calcsky the same way as we use rebin for coadding data. If one has already produced a final map of the source, one can use this map as the input model for calcsky. For multiple data sets we should make an input file with weights and offsets as we do for rebin, see Section 5.8. We now choose the default model, which will be the median of all the observed maps. Once calcsky is done, we go back and run remsky on each data file which was included in the model computed by calcsky.

calcsky works extremely well even for extended sources, if one has kept the chop position fixed. For extended sources it is therefore an advantage to chop in a fixed ra/dec frame. One can also use calcsky for a spherically symmetric source like IRC+10216 even for a chop throw of 60", but then one should use *az* option for the OUT_COORDS. calcsky also includes the option to account for the chop throw and direction, and it should therefore work even when we chop differently on extended emission from one map to the next.

For scan maps calcsky is our only option for sky noise removal, because in scan maps each bolometer will normally include both source and sky (see later).

Below we show how to use calcsky on the same file we already reduced with remsky. When we now run remsky it will not ask for sky bolometers, but will use the sky extension stored in the header to remove sky noise variations.

```
% calcsky
OUT_COORDS - Coordinate sys for sky determination /'RJ'/ >
SURF: output coordinates are FK5 J2000.0
REF - Name of first data file to be processed /'i86_lon_sky'/ >
i86_lon_cal
SURF: run 86 was a MAP observation of IRC+10216 with JIGGLE sampling
SURF: file contains data for 4 exposure(s) in 3 integrations(s) in 1
measurement(s)

WEIGHT - Weight to be assigned to input dataset /1/ >
SHIFT_DX - X shift to be applied to input dataset on output map
(arcsec) /0/ >
SHIFT_DY - Y shift to be applied to input dataset on output map
(arcsec) /0/ >
IN - Name of next input file to be processed /!/ >
SURF Input data: (name, weight, dx, dy)
   -- 1: i86_lon_cal (1, 0, 0)

BOXSZ - Size of smoothing box (seconds) /2/ >
MODEL - File containing source model /!/ >
% remsky
IN - Name of input file containing demodulated map data
/@i86_lon_ext/ >
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
OUT - Name of output file /'i86_lon_sky'/ > i86_lon_sky
REMSKY: Using SKY extension to determine sky contribution
```

The sky noise we see in our jiggle maps is due to changes in the sky emission between nods of the telescope. For unstable sky conditions this results in a tile-pattern in your map. This type of structure will not be removed by calcsky if we apply it to a single data set. Indeed in this particular example, remsky with selected sky bolometers gave a much better result than using calcsky. However, for really extended sources and scan maps we may not have a choice. We will have to use calcsky followed by remsky.

Another advantage of calcsky which is worth mentioning is that we can run calcsky on the 450μm array and copy the calculated sky noise variations to the 850μm array after appropriate scaling. This enables us to remove sky noise variations on a very faint extended 850μm–source, without subtracting out source emission.

5.6 Manual Despiking

The way we despike jiggle maps depends on whether we have done short or long integrations and on whether our source is compact or extended. In the above example we only have 3 integrations and despike will miss most spikes. We can still use scuclip but if we want to go

deep, we have to make sure that we don't clip the source as well. We know that IRC+10216 is relatively compact with a faint 'halo' type emission surrounding the core. It may therefore enough to use scuclip, but go deeper than in our initial despike effort. To be on the safe side, i.e. to make sure that we do not clip the source, we set the central bolometer to bad before we start and reset it back to good afterwards using change_quality.

```
% change_quality 'i86_lon_sky{b19}'
SURF: run 86 was a MAP observation of IRC+10216
SURF: file has data for 37 bolometers, measured at 192 positions.
 - there are data for 4 exposure(s) in 3 integration(s) in 1
measurements.
BAD_QUALITY - Set quality to bad? (No will set quality to good) /YES/
>
% scuclip
IN - Name of input file containing demodulated map data
/@i86_sho_sky/ >
SURF: run 86 was a MAP observation with JIGGLE sampling of object
IRC+10216
OUT - Name of output file /'i86_lon_clip'/ >
NSIGMA - How many sigma to despike bolometers /5/ > 4
SURF: Removed 10 spikes
% change_quality 'i86_lon_clip{b19}'
SURF: run 86 was a MAP observation of IRC+10216
SURF: file has data for 37 bolometers, measured at 192 positions.
 - there are data for 4 exposure(s) in 3 integration(s) in 1
measurements.
BAD_QUALITY - Set quality to bad? (No will set quality to good) /YES/
> no
```

Compared to our first 5 sigma we now found 10 additional spikes. We could probably have used 3 sigma, but with a small data set it is better to be conservative, For the short array, we have to be really careful when using scuclip, if we work with 64–point jiggle maps. If we are looking at a point source centered in the array, the jiggle steps are so large, that the first ring of bolometers will pick up the source in part of the jiggle pattern. scuclip can in this case interpret the source as being noise, since it only shows up in a just a couple of points, and flag them. The end result can easily be that when we regrid the map, we get an artificially narrow image, because we despiked the "flanks" of the source. If we want to run scuclip on the short array, we should not only blank the central bolometer, but also the first bolometer ring (seven bolometers), and then reset the flagged bolometers afterwards and despike them manually, which becomes somewhat tedious.

For short maps on strong or extended objects we therefore mostly end up doing manual despiking. We can either use the interactive shell script dspbol or the FIGARO routine sclean.

```
% sclean
IMAGE - (IMage) Name of image to be cleaned /o86_lon_sky/ >
o86_lon_sky
OUTPUT - (OUTput) Name of resulting image /o86_lon_clip/ >
o86_lon_clip
IDEV - Device for image display /'xwindows'/ >
```

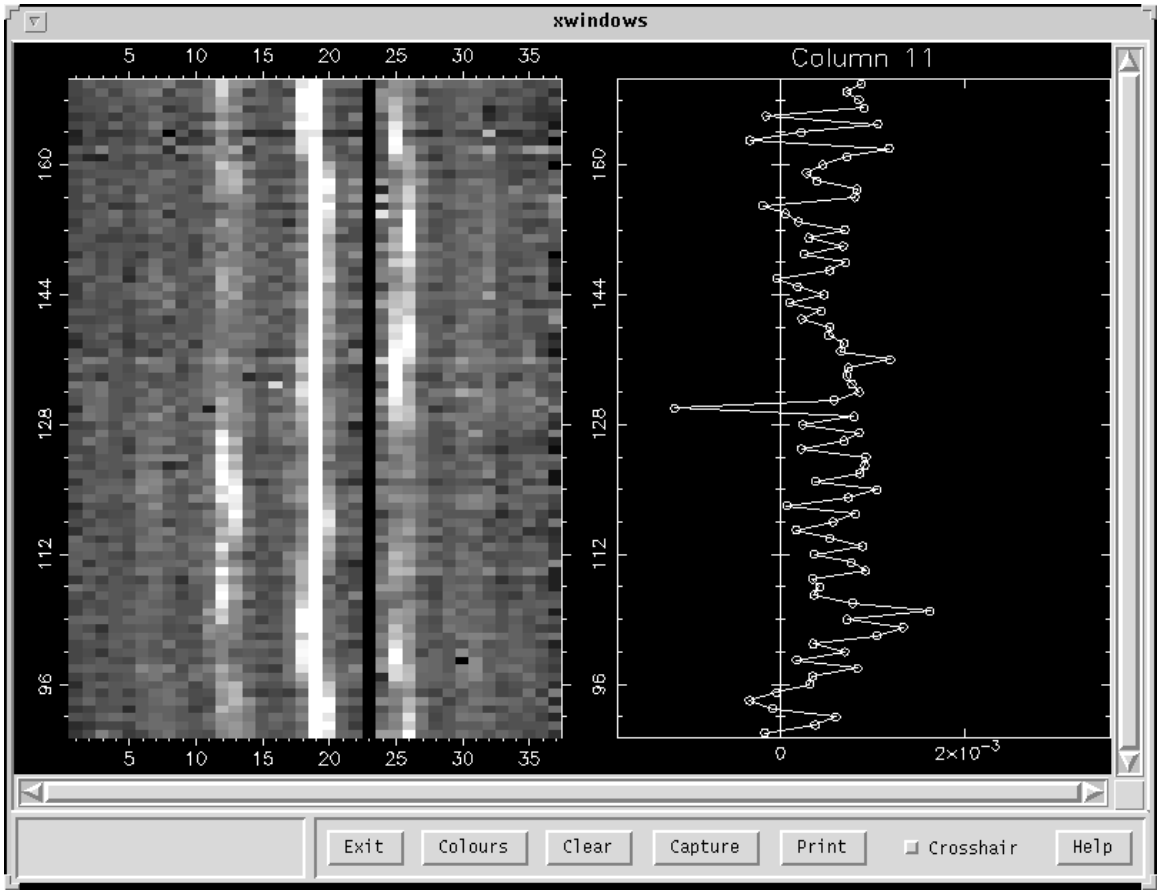


Figure 6: The display from the FIGARO package sclean. sclean allows you to easily inspect the time series data and despike data.

which will give you an image similar to Fig. 6. sclean is an interactive application and if you select the xwindow, and type 'b' while the cursor is over a selected bolometer you will get the display shown in Fig. 6. Here the left hand side shows all the bolometers, and the right hand side shows just the bolometer you selected. The two most useful commands are probably 'a' - delete the pixel selected, and 'y' - delete the indicated area and fix by interpolation along the y-axis. Manual despiking like this is usually only needed for short integration maps of strong sources or when taking out more extended glitches in scan maps, which are not picked up by automatic despiking.

5.7 Pointing corrections and final map creation

Before we create our final rebin, we will still need to calibrate the map (Section 7) and correct for pointing drifts that occurred over the map. Since we have only dealt with a single map so far, we omit the calibration stage and proceed to correct for pointing drifts. To do this we reduce the closest pointing observations before and after the map in the same way as we reduced our map, but with minimal despiking and regridding the pointing map in *az*. These pointing errors we find have to be negated in the task change_pointing. For scan 86 I find one pointing done on the star itself before the map (#81), and none afterwards. The final pointing errors of # 81 are

DAZ/DEL = 0.38″/+0.33″ at an LST of 9:26 as deduced from the FITS header of that scan. These errors are the residual from on-line pointing corrections and off-line data reduction of scan 81. If we treat our map as if it was a pointing observation we find pointing errors of +1.52″/+0.12″, which we apply directly after negating the sign as shown below. Instead of choosing a time halfway through the map, we have to take the LST time at the end of the map. This is kind of cheating, because we derive pointing from the same objects that we are supposed to correct the pointing for, but here we do it only to illustrate how pointing corrections are done.

```
% change_pointing
IN - Name of input file containing demodulated map data
/@i86_lon_sky/ > i86_lon_clip
SURF: run 86 was a MAP observation of IRC+10216
SURF: observation started at LST 9 41 05 and ended at 9 50 09
SURF: no pointing corrections found

CHANGE_POINT - Do you want to change the pointing correction data > y
POINT_LST - The sidereal time of the pointing offset (hh mm ss.ss)
/!/ > 9 26
POINT_DAZ - The azimuth pointing correction to be added (arcsec) >
-0.38
POINT_DEL - The elevation pointing correction to be added (arcsec) >
-0.33
POINT_LST - The sidereal time of the pointing offset (hh mm ss.ss)
/!/ > 9 50 10
POINT_DAZ - The azimuth pointing correction to be added (arcsec) >
-1.52
POINT_DEL - The elevation pointing correction to be added (arcsec) >
-0.12
POINT_LST - The sidereal time of the pointing offset (hh mm ss.ss)
/!/ >
```

We can now run the sky noise corrected, despiked, and pointing corrected data through rebin to obtain our final map *i86_lon_reb*. Looking at the map, we can see that bolometers, which we knew were noisy still can be seen, but since we do not have any additional data sets to add to the map, we will leave the map as such. The pointing offsets, as determined with centroid give pointing offsets of -0.18″ in RA and 0.0″ in Dec, suggesting that the pointing corrections worked quite well.

Now we would do the short array in the same way either by re–running scuquick or by starting with the SURF task extinction. We now run extinction on `i86_flat`, which was created when we reduced the long array, and which also contains the flat fielded data for the short array.

5.8 Coadding data, or how to deal with long integrations

Reducing multiple observations of the same source is about as easy as reducing a single map, it is just a bit more time consuming. Some aspects of the reduction process, like despiking, now becomes simpler and more reliable.

All the basic steps up to and including sky removal are done the same way as in the example above, but now we can despike differently, because we have much more redundant data. After we have extinction corrected, calibrated, done sky removal and pointing corrections we can now try despike on all the data sets.

To demonstrate despike we have collected 10 jiggle maps with two or three integrations of HL Tau, one of our secondary calibrators. All the maps are taken in good weather conditions, but some suffer from sharp variations in sky noise. We have set the really noisy bolometers to bad for all maps, but in some cases we have still left bolometers with 1.5 - 2 times the average noise level. For all maps the chop throw was 120" in Azimuth. Note that if you chop on the array this despiking technique will not work very well, unless you use a fixed chop angle, or regrid in *az* which works very well for a point source or a centrally condensed spherically symmetric source. To make the reduction easier, and to enable us to go back and redo the despiking, we make a small ASCII file for the maps, which includes the name, weight and pointing shift – one line per map. Here we start without applying pointing corrections because the pointing errors are negligible. Neither do we apply weighting, because we will need despiked maps in order to compute the weights. In this example we call the file `hl.inp`. The first line in this file is `hl39_lon_sky 1 0 0`, where *hl39_lon_sky* is the file name and 1 is the weight. The two zeros that follow the weight are the pointing shifts in the coordinate frame that the data are rebinned to, which in this case *RJ*. Since HL Tau is a point source with negligible extended emission one could equally well have rebinned the whole data set in *az*. We are now ready to try despike.

```
% despike noloop
OUT_COORDS - Coordinate sys of output map; PL,AZ,NA,RB,RJ,RD or GA
/'RJ'/ >
SURF: output coordinates are FK5 J2000.0
REF - Name of first data file to be rebinned /'hltau_lon'/ > hl.inp
SURF: Reading file hl39_lon_sky
SURF: run 39 was a MAP observation of HLTau with JIGGLE sampling
SURF: file contains data for 4 exposure(s) in 3 integrations(s) in 1
measurement(s)

   ..............   list continues until it has read in all maps,
then

SURF Input data: (name, weight, dx, dy)
   -- 1: hl39_lon_sky (1, 0, 0)

     ..............
   -- 10: hl42_lon_sky (1, 0, 0)

NSIGMA - Sigma levels at which to despike /3/ >
SMODE - Smoothing mode for clipping envelope (none,hann) /'hann'/ >
DEVICE - Name of graphics device /@xwindows/ >
DMODE - Display mode (Spiral, Xlinear, Ylinear, Diag1, Diag2)
/'spiral'/ >
XRANGE - X range of plot (! to continue) /[1,5112]/ > 1,2000
XRANGE - X range of plot (! to continue) /[1,2000]/ > !
SURF: 8 spikes detected in file hl39_lon_sky
.........................
SURF: 196 spikes detected in file hl42_lon_sky
OUT - Name of despiked data file /'hl39_lon_dsp'/ >
.........................
OUT - Name of despiked data file /'hl42_lon_dsp'/ >
```

The display for the first 2000 data points are shown in Fig. 7. Note that after you run despike the display may get messed up. It is normally sufficient to clear the display with gdclear and

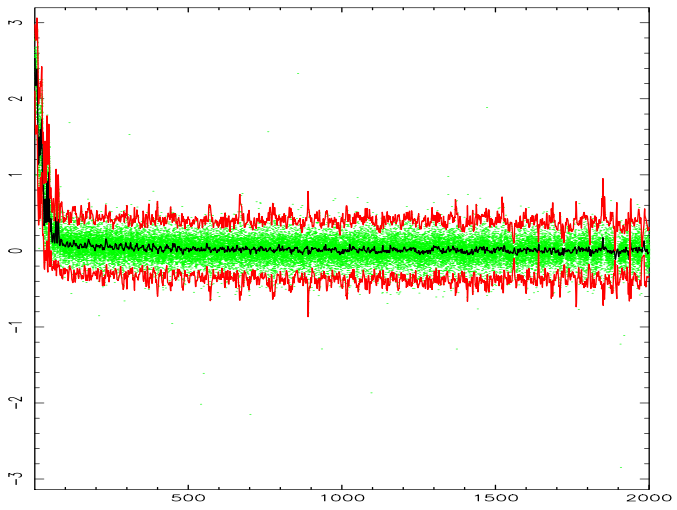


Figure 7: These are the first 2000 data points analyzed by despike. The observed data are plotted in green, median in black and the Hanning smoothed 3–σ envelope is plotted in red. Any data points outside this limit will be flagged as bad.

re-issue the the color table, e.g. lutbgyrw. If there are still problems, delete the AGI-file from your home directory (`agi_xxx.sdf`, where `xxx` is the name of your work station).

We now run rebin on each data set to determine the offsets we need to center the map, which will ensure that we get the sharpest possible image. We also determine the rms noise level for each map. Here it is a clear advantage to use GAIA. After displaying the map with your preferred color scheme and contrast level using the `View` menu, go to the `Image-Analysis` menu and click on *image regions*. Choose the polygon option of *ARD regions* and outline the region you want to use with the cursor. After it is done, click on *Stats selected* and you are done. For determining pointing offsets it is easier to use the KAPPA command centroid i.e. `centroid search=19 cerror`, where we adjust the search parameter to minimize the fit errors. For 850 μm and a 1 arcsec cell setting *search* to 17 or 19 generally gives a good result. After we have determined the noise level and position offsets for each map we compute the weights for each map. An easy way to do this is to set the weight for the first map equal to one. All other maps are then weighted relative to this map. If we call the integration times t_1 and t_2, and the corresponding noise values n_1 and n_2 for map 1 and map 2 respectively, then we can compute the weighting factor from the following equation:

$$weight = (n_1/n_2)^2 \times (t_1/t_2) \tag{3}$$

We now edit in the weights and position offset in `hl.inp` and run despike again on the original data set. The end result appears good enough.

Now we can form the final co-add of the data very easily, since we have already computed the weights and position shifts that we need to get the best S/N and sharpest image from our data. The only thing we need to do is to change the file extensions from *sky* to *dsp*, the name we used for despiked data files. We rename the input file to `hl.dsp` in case we would like to redo the despiking. We now run this file through rebin, i.e.,

```
% rebin noloop
REBIN_METHOD - Rebinning method to be used /'LINEAR'/ >
SURF: Initializing LINEAR weighting functions
OUT_COORDS - Coordinate sys of output map; PL,AZ,NA,RB,RJ,RD or GA
/'RJ'/ >
SURF: output coordinates are FK5 J2000.0
REF - Name of first data file to be rebinned /'hl42_lon_dsp'/ >
hl.inp
SURF: Reading file hl39_lon_dsp
SURF: run 39 was a MAP observation of HLTau with JIGGLE sampling
SURF: file contains data for 4 exposure(s) in 3 integrations(s) in 1
measurement(s)
..................
SURF: Reading file hl42_lon_dsp
SURF: run 42 was a MAP observation of HLTau with JIGGLE sampling
SURF: file contains data for 4 exposure(s) in 3 integrations(s) in 1
measurement(s)

SURF Input data: (name, weight, dx, dy)
   -- 1: hl39_lon_dsp (1, 0.08, 1.19)
   -- 2: hl64_lon_dsp (0.4, 0.64, 0.75)
   -- 3: hl41_lon_dsp (0.9, -0.23, 2.7)
   -- 4: hl55_lon_dsp (1, -0.86, 0.38)
   -- 5: hl61_lon_dsp (0.8, 0.14, 1.04)
   -- 6: hl75_lon_dsp (0.75, 0.4, -0.13)
   -- 7: hl84_lon_dsp (0.5, 0.42, 0.83)
   -- 8: hl60_lon_dsp (0.9, 0.11, -1.41)
   -- 9: hl23_lon_dsp (0.75, 0.1, 0.46)
   -- 10: hl42_lon_dsp (0.3, 0.75, 1.06)

LONG_OUT - Longitude of of output map centre in hh (or dd) mm ss.ss
format
/'+04 31 38.46'/ >
LAT_OUT - Latitude of output map centre in dd mm ss.ss format
/'+ 18 13 57.8'/ >
OUT_OBJECT - Object name for output map /'HLTau'/ >
PIXSIZE_OUT - Size of pixels in output map (arcsec) /3/ > 1
SIZE - Number of pixels in output map (NX,NY) /[194,192]/ >
OUT - Name of file to contain rebinned map /'HLTau'/ > hltau_lon
WTFN_REGRID: Beginning regrid process
WTFN_REGRID: Entering second rebin phase (T = 3.113513 seconds)
WTFN_REGRID: Entering third rebin phase (T = 14.55025 seconds)
WTFN_REGRID: Regrid complete. Elapsed time = 15.23921 seconds
```

We now have our final co-added map, which is almost as good as it can get from the data we had available.

If you have extremely large data sets, then you may find that your computer runs out of memory in rebin. The recommended solution, if you cannot find a more powerful workstation, is to split the data set in two parts, run each separately through rebin, and co-add the two final maps using the KAPPA command add or the task makemos in CCDPACK.

6 Scan maps

Scan maps, i.e. mapping while continuously scanning the array over the source, require a few additional steps in the reduction procedure. The reduction process is also different depending on whether we made conventional scan maps or used the "Emerson2" technique. Conventional scan maps refer to maps done while chopping in the scan direction and restoring the resulting dual beam map with the EKH algorithm (the Emerson-Klein-Haslam algorithm – known to all who have ever used NOD2 or JCMTDR [7]) before transforming the map into equatorial coordinates. The "Emerson2" technique is essentially a basket weaving technique, where one can scan in an arbitrary angle but chop in two orthogonal directions and restore the dual beam map in the Fourier plane after converting the dual beam maps to equatorial coordinates. This method therefore requires a minimum of two maps, one where we chop in RA and one where we chop in Dec. The standard setup for SCUBA is to use six maps, three of which are done while chopping in RA with chop throws of 30, 44 and 68", and three while chopping in Dec with the same three chop throws. The chop throws are chosen so that we should be sensitive to most spatial frequencies in the map. If possible one should try to choose the map size so that it covers the whole source and provides an additional baseline region off source, but as we all know this is not always possible.

For all scan maps we can do the first three reduction steps: reduce_switch, flatfield, and extinction, the same way as we would do for any jiggle map. We also blank out noisy bolometers, but from here onwards we need to apply slightly different methods. Scan maps are also affected by sky noise, especially when we use the "Emerson2" technique. This is because the time difference between when the positive and the negative beam passes the same position on the sky can be significant, and sometimes even longer than in jiggle-maps. This is especially noticeable for large maps and large chop throws. We can crudely remove sky noise in scan maps, but not as well as in jiggle maps, calcsky is our main tool and needs several repeats of the same maps to work efficiently.

6.1 Despiking

After we have extinction corrected the map and taken out noisy bolometers, we need to despike the data. This is done with the SURF task despike2, which takes a small portion from each end of a scan and computes the rms–variations for each bolometer and then does a standard sigma clip. If you want to be conservative, use 4 sigma. This does a reasonable job, but large spikes (extending over several seconds in time) are not detected and these will have to be removed manually using sclean or dspbol. It is often necessary to run despike2 a second time after scan_rlb, because even with the same sigma threshold, the rms is now smaller and one finds a fair amount of residual spikes missed in the first round.

6.2 Base line removal - scan_rlb

The SCUBA on-line software normalizes each scan in conventional (EKH) scan maps, which leads to baseline offsets, but even the "Emerson 2" maps have baseline uncertainties. Spillover, large spikes and sky noise add to these baseline uncertainties. It is therefore absolutely necessary to remove the baseline offset for each bolometer. If it is omitted one may end up with severe striping in the map. If your map is large enough, i.e. you have no source emission at the end of

your scan, you can run scan_rlb, and fit a linear baseline for each scan (exposure and bolometer) by taking the end portions of the scan as a measure of the signal level. The default size of the region used for the baseline subtraction is the number of data points in one chop throw. This can be inadequate for the small chops, especially if the data are spiky or suffer from sky noise variations, and you may consider increasing the default to perhaps 100". The output from scan_rlb is the basic map name, now appended by `_rlb`, which is the file you will use as an input for the next stage in the data reduction.

However, if your map is not large enough to start and end on a region free of emission scan_rlb will result in a gradient over the scan. Taking the default behavior of scan_rlb in this situation is probably the leading cause of poor results obtained from scan mapping. When you map galactic regions it is usually better to use the median rather than the default, which is *linear*. Another, sometimes more successful approach is to use SCUBA sections.

In scan map mode each 'sweep' or 'raster' is a section (exposure), and the complete map is an integration. If you believe that you have sections of the map that are free of emission, you may be in luck, and you can use these emission free sections to provide the baseline level for the rest of your map. To find out which section is which, display your rebinned map and then use scuover. To produce the image in Fig. 8 we typed:

```
%display 20000721_0023_lon_reb
DEVICE - Name of display device /@xwindow/ >
MODE - Method to define the scaling limits /'SCALE'/ >
LOW - Low value for display /-0.86096328496933/ >
HIGH - High value for display /12.873136520386/ >
% scuover exposure=1
DEVICE - Name of graphics device /@xwindow/ >
Alignment has occurred within the AXIS Domain.

NDF - Image to display bolometers over /@20000721_0023_lon_reb/ >
SURF: file contains data for 21 exposure(s) in 1 integration(s) in 1
measurement(s)
```

One can see that the scan map started at the top right of the map (a scan map of the moon's limb) and took 11 'sweeps' or exposures to complete the map. One can see that only exposures 1, 9, 10 and 11 started and ended off source. Therefore when one wishes to remove the baselines (remembering that one now has to go back 2 or 3 steps to do this) from this image the best method would be:

```
% scan_rlb
IN - Name of input file containing demodulated data
/@20000721_0023_lon_ext/ >
SURF: run 23 was a MAP observation of object MOON
SURF: file contains data for 21 exposure(s) in 1 integration(s) in 1
measurement(s)
OUT - Name of file to contain restored data /'20000721_0023_lon_rlb'/
>
METHOD - Method to use for baseline removal /'LINEAR'/ > section
RLB - Remove fitted baseline from output data /YES/ >
SECTION - Please enter a section (including {}) > [{e1}{e9}{e10}{e11}]
REMOVE_DC_OFFSET_SECTION: Processing integration 1
```

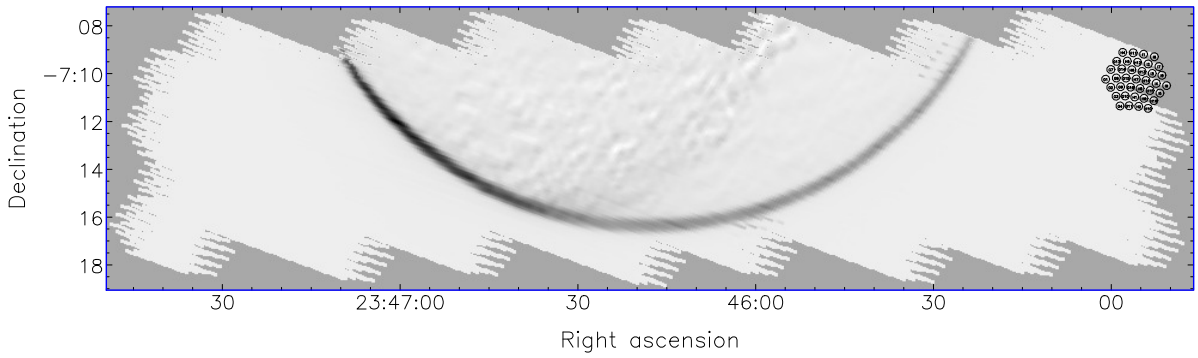


Figure 8: An example of a scanmap of the moon's limb. The SCUBA array's position at the beginning of the observation is shown. In total it took 11 'sweeps' or exposures to map the area requested.

Its obvious with a source like the moon when you are on and off source and because the moon fills most of the image, the data were additionally masked. This is a rather special case, where it would have been very difficult to do the baseline subtraction any other way without ending up writing special purpose software. For molecular clouds and star forming regions it is often very difficult to find emission free regions. For really extended emission, like the Galactic center, it was therefore found that the best way to do baseline removal is to use the median level of all the scans for each bolometer (Pierce-Price et al. [15]), which corresponds to specifying SECTION as {}. Quite often you will find that you have to do an initial map reduction to see how extended the region is and where you can find emission free areas. Once you know this, it is much easier to go back to noisy sub-maps and redo the baselines.

6.3 Sky noise removal in scan maps

There are no sky bolometers in a scan map, i.e. each bolometer can be on source, and furthermore a bolometer will cover a different region of the sky for each map exposure. It is therefore not possible to remove sky noise the way we usually do for jiggle maps. Sky noise can be extremely severe and rapid on Mauna Kea. Under such circumstances sky noise variations can completely distort a scan map, especially for large chop throws. calcsky was originally developed to give us a technique for reducing sky noise in scan maps, but as we have seen it works also quite well for jiggle maps, as discussed in Section 5.5. calcsky computes a model of the source emission, and subtracts it from the data for all bolometers as a function of time. Several maps can be co-added.

Below we test calcsky on a small scan map, taken with a 20" chop in RA. The map file, *rn14_lon_dsp* has already been baseline subtracted, pointing corrected, calibrated and despiked.

```
% calcsky
OUT_COORDS - Coordinate sys for sky determination /'RJ'/ >
SURF: output coordinates are FK5 J2000.0
REF - Name of first data file to be processed
/'rn14_lon_reb'/ > rn14_lon_rlb
SURF: run 14 was a MAP observation of RNO1b with RASTER sampling
SURF: file contains
SCULIB_PROCESS_BOLS: no data for exp 7 in int 1, meas 1
```

```
SCULIB_PROCESS_BOLS: no data for exp 7 in int 2, meas 1
SCULIB_PROCESS_BOLS: no data for exp 7 in int 3, meas 1
WEIGHT - Weight to be assigned to input dataset /1/ >
SHIFT_DX - X shift to be applied to input dataset on output map
(arcsec) /0/ >
SHIFT_DY - Y shift to be applied to input dataset on output map
(arcsec) /0/ >
IN - Name of next input file to be processed /!/ >
SURF Input data: (name, weight, dx, dy)
   -- 1: rn14_lon_rlb (1, 0, 0)

BOXSZ - Size of smoothing box (seconds) /2/ >
MODEL - File containing source model /!/ >
```

We can now examine the sky variations with linplot. There seems to be clear systematic variations as a function of time, but the maximum deviation is only $\sim$ 150 mJy/beam, c.f. the jiggle map we did earlier (Fig. 5), which showed sky noise variations of about 600 mJy/beam.

However, we can easily check how much we gain in noise performance if we remove the sky noise from the data. We therefore run remsky on the same data file that we processed with calcsky.

```
% remsky
IN - Name of input file containing demodulated map data
/@rn14_lon_reb(280:340,90:300)/ > rn14_lon_rlb
SURF: run 14 was a MAP observation with RASTER sampling of object
RNO1b
OUT - Name of output file /'rn14_lon_sky'/ >
REMSKY: Using SKY extension to determine sky contribution
```

In this case the gain was rather marginal. The despiked data file gave an rms noise of 70 mJy/beam after running it through rebin while the sky corrected one improved by $\sim$ 0.5 mJy/beam (i.e. an improvement of less than one percent), when examined over the same area of the map, which means that it was not really worth doing. Nevertheless, I go through all six maps in the set, and find as I had expected the largest sky fluctuations for maps taken with a 65″ chop throw. In the last map of the set (65″ chop in Dec), the maximum sky fluctuations were $\sim$ 250 mJy/beam, or peak–to–peak sky noise variations of $\sim$ 500 mJy/beam, resulting in a 7% improvement in noise after subtracting the calculated sky noise variations.

calcsky does not work very well on a single map, but since calcsky can account for the chop throw, one can use a first version of the final map as a model for the individual sub–maps. If necessary, one can do a second iteration by using the sky corrected sub–maps to create a new improved map, which can be used as an even better model for calcsky.

From here onwards the rest of the reduction differs depending on the scan map mode.

6.4 Conventional scan maps

After we calibrated the maps and done the baseline removal, we need to restore the map from a dual to a single beam map. This is done by using restore. This task does a standard EKH restoration. Below we show an example of restoring a scan map of NGC 7538, a high mass star forming region. In this case we accept the default for the chop throw, but if the restored map looks poor, it is most likely because the chop throw deviates from the nominal value.

```
% restore
IN - Name of input file containing demodulated data /@n39_lon_rlb/ >
SURF: run 39 was a MAP observation of object NGC7538
CHOP - Size of chop /60/ >
SURF: file contains data for 8 exposure(s) in 2 integration(s) in 1
measurement(s)
OUT - Name of file to contain restored data /'n39_lon_res'/ >
2POS_DECONV: Processing exposure 1 of integration 1
...........
SCULIB_2POS_DECONV : no data for exp 8 in int 1, meas 1
2POS_DECONV: Processing exposure 1 of integration 2
 ..........
2POS_DECONV: Processing exposure 7 of integration 2
SCULIB_2POS_DECONV : no data for exp 8 in int 2, meas 1
```

Once this is done, we can apply pointing corrections to remove any pointing drifts we had during the duration of the map. If the map is still uncalibrated (we strongly recommend to apply calibration immediately after the extinction correction), we should do it now. Once we have all the maps calibrated we can proceed and co-add any additional maps that we may have using rebin, exactly like we do for jiggle maps. Note that if you co-add data sets taken in different weather conditions or during different nights, you will have to weight the individual data sets in order to minimize the noise in your final map.

6.5 Scan maps taken with the "Emerson2" technique

Scan maps taken with the "Emerson2" technique, i.e. basket weaving in Fourier space, have to be run through rebin without restoring them from dual to single beam maps and in the same coordinate system that was used for the chop throw (e.g. RJ or PL). You have to make one map for each chop throw and each map has to have identical dimensions and pixel size. We recommend that you make the maps larger than the area mapped and then cut them in size after running them through remdbm, the final reduction stage for "Emerson2" scan maps. Since remdbm makes use of fast Fourier transforms, it is advisable to choose map sizes, which are a power of two. Make sure that you do not choose a size equal to the default size for any of the sub–maps, because in that case rebin will choose its own map center, which is not equal to the center pixel of the map. The end result will be a garbled map. To make it easy to identify the map sets we need for remdbm one can give the calibrated, noise weighted and co-added maps names like : *m30ra.sdf, m44ra.sdf, m30dec.sdf, m44dec.sdf etc.*, not because the tasks need it, but it easier for book keeping purposes. These maps also have to be corrected for pointing drifts but they should not be restored. The maps are still dual beam maps and each source in the map should show up as a positive and a negative feature in the image. Fig. 9 shows the first sub–map of RNO1b, i.e. scan *rn14_lon_sky*, which we used to test calcsky in Section 6.3. It has now run through rebin and given a size 384 $\times$ 384 with a pixel size of 1″. Since this map was taken with a 20″ chop in RA, we called it *ra20.sdf*. We should have made the map slightly bigger, because the maps with 30″ and 65″ chops will cover a larger area. Note that this map was taken at a time when the recommended chop throws were 20″, 30″, and 65″, now the recommended minimum set is 30″, 44″ and 68″, which gives a somewhat better recovery of spatial frequencies.

Once we have all maps pointing corrected and co-added to the same pixel size and dimension, we can run remdbm which converts the maps into our final image. In the example below I take the six sub–maps of RNO1b and convert them into a map called *rno1b_lon_reb* using remdbm. We

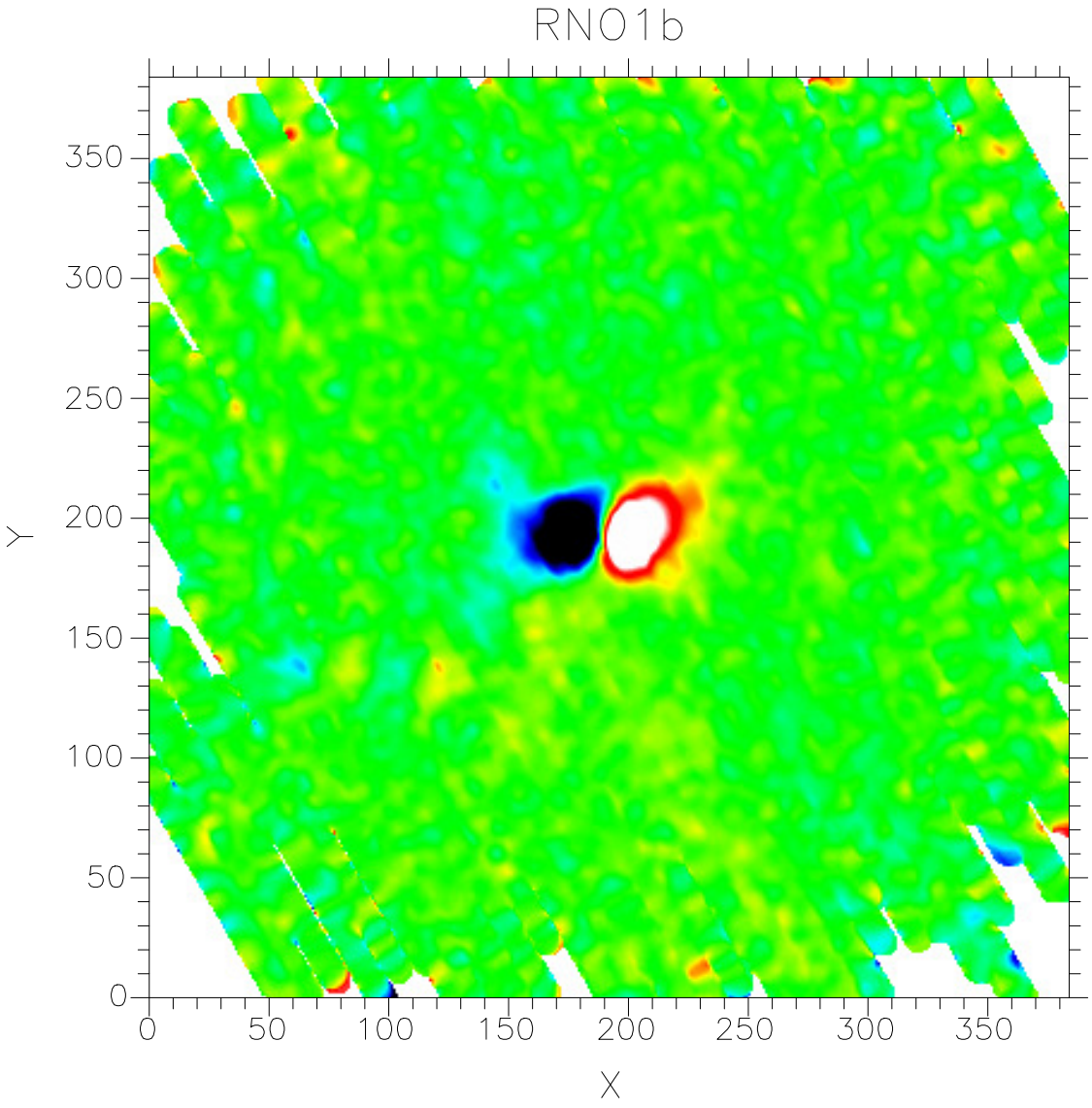


Figure 9: An example of the first map in a series of 6 maps taken with the "Emerson2" mapping technique. This is a calibrated map with a 20" chop in RA, which is converted to RJ with rebin but still unrestored. We can see the positive and the negative features in the map, and a few noise spikes at the edges of the map.

specify the name of the final map with the parameter `out`, and a provide the task with a listing of the files, see below:

```
% remdbm ra20 ra30 ra65 dec20 dec30 dec65 -out=rno1b_lon_reb
Perl/ADAM messaging is present. Good
Starting monoliths...Done
Loop number 1
Chop: PA=90 THROW=20

Doing forward transformation
..................................................

Loop number 6
Chop: PA=0 THROW=65

Doing forward transformation

There was 1 element changed in the Data array.
Running inverse FFT...
Maximum difference between estimates of the same Fourier component is
0.02118739.

Doing inverse transformation

Result stored in rno1b_lon_reb
ADAM exited
```

remdbm also accepts wild–cards and we could therefore have listed the files as *ra* dec** and it would have picked up the whole set of six maps. Neither do we have to give it chop throw or chop direction. It will extract that information from the FITS–header. The task can restore a single map file, but even two maps in orthogonal directions looks rather ugly, and it is strongly recommended to use a set of 4 or 6 maps.

Due to the way the Fourier transformations are done, remdbm forces the sum of all pixels to be zero. This will introduce a small negative background level in the map. We can remove this baseline by analyzing the map with KAPPA's stats or with GAIA and add back the level we deduce with cadd. For our map we find the negative baseline to be ~ 0.3 Jy/beam, which we add to the map. The final, baseline subtracted map is shown in Fig. 10.

7 Map calibration

Until now we have deliberately avoided the issue of calibrating your data. This means that your reduced data, up until this stage, are in units of volts. Since the calibration varies from night to night and even within a single night, one should generally calibrate individual maps before coadding to achieve the best result. So how does one convert instrumental units into a physical measure of luminosity or surface brightness? The solution as in most astronomy is to look at a source of known brightness in exactly the same way, i.e. using the same mode of observing for your target as well as for your calibrator(s).

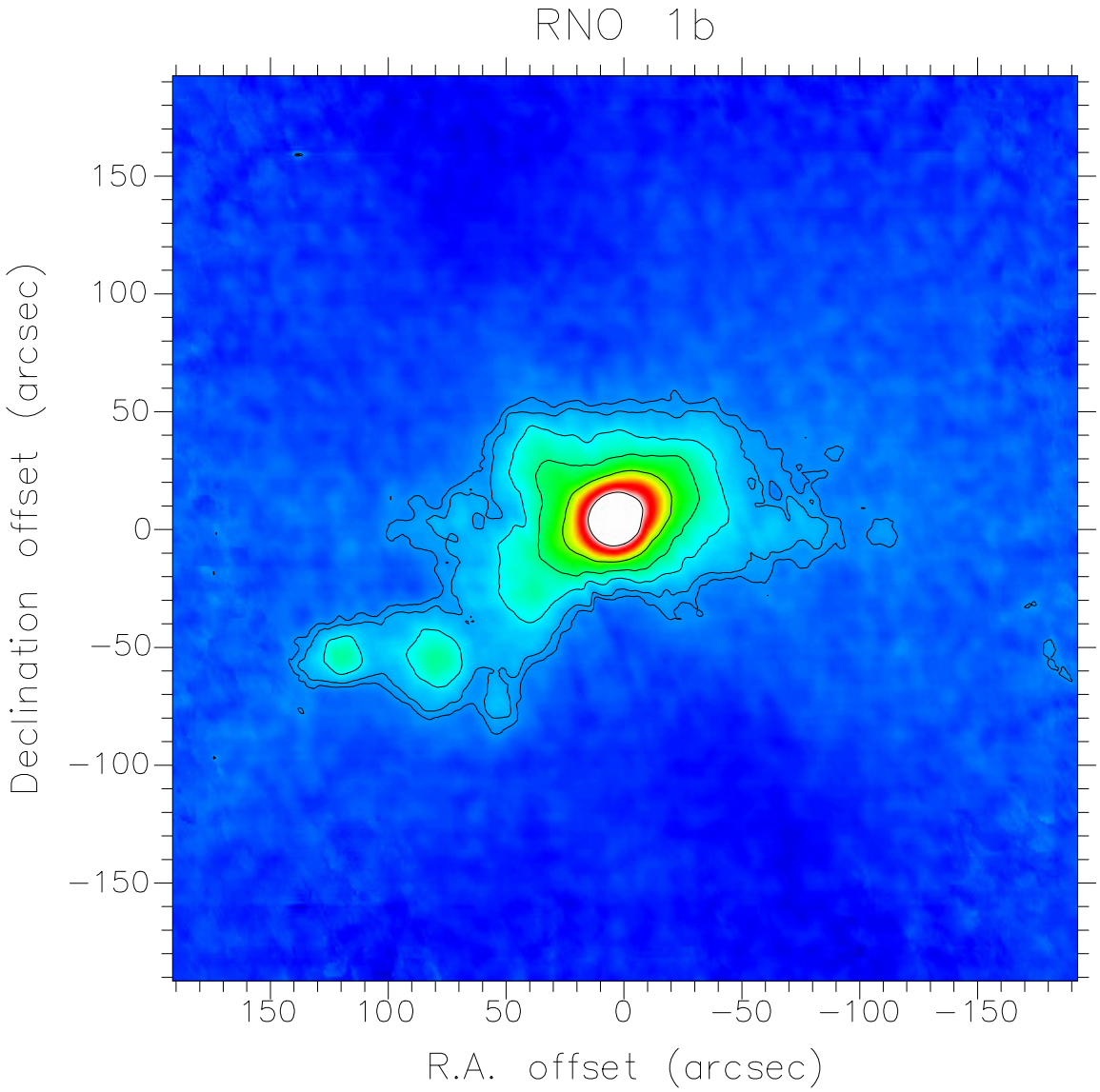


Figure 10: Our final "Emerson2" scan maps of RNO 1b. The map has a little bit of negative residuals in the scans that went over the strongest part of the map (edge of the map at p.a. $\sim$ 30 and 210 degrees), indicating that the baseline removal was not perfect, but it does not show up with the contrast used for this figure. For a published version of the map, see Sandell and Weintraub [16].

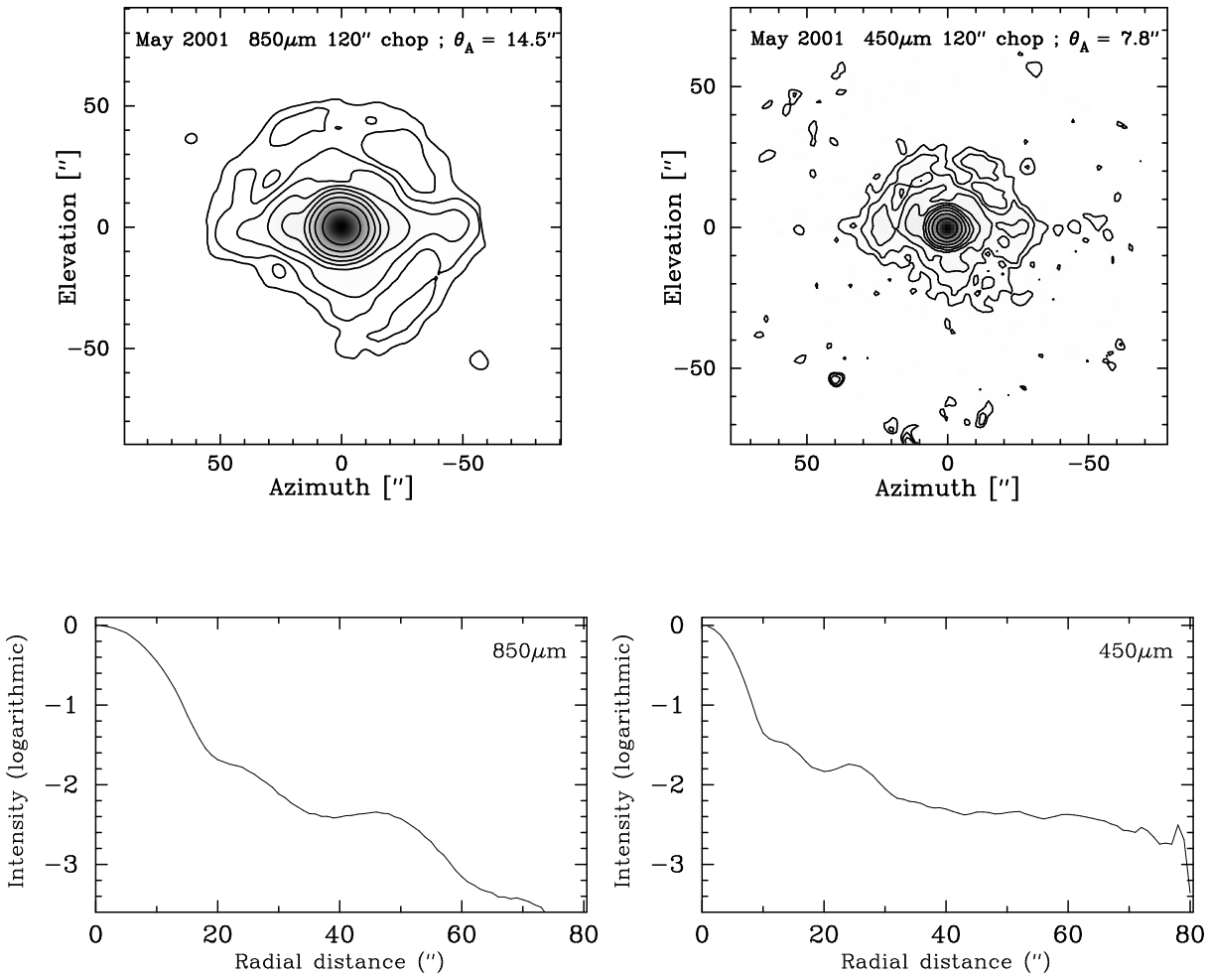


Figure 11: Upper panel: Logarithmic contour plots of Uranus beam maps overlaid on grayscale. At 850 μm we plot ten contours starting from 0.25% of the peak intensity, while the 450 μm beam map goes from 1% of the peak intensity. One can see a diffraction like ring in the 850 μm map at a radius of $\sim 47''$, but the amplitude is $< 1\%$ of the peak. At 450 μm the ring is at $\sim 25''$ and has an average amplitude of $\sim 2\%$ of the peak intensity. This is seen more clearly in the lower panels, which show radially averaged beam profiles plotted in a logarithmic scale

In the optical and infra-red the standard sources are almost always point sources, standard stars, and the point spread function is well defined. In the submillimetre things are more complicated. Our primary calibrators, Mars and Uranus, are not point sources, and the point spread function is very extended and strongly wavelength dependent. The JCMT beam at 450 μm is actually much worse than the ill-fabled Hubble before the mirror was corrected.

The way we calibrate may differ depending on whether we observe point sources or extended sources. For point sources we can ignore the error beam and do simple aperture photometry, for extended sources we normally have to calibrate in Jy/beam and characterize our beam profile. In the following we first go through how to characterize the beam profile, then the case of calibrating in Jy/aperture and finally we proceed to the more general case of calibrating in Jy/beam, which is valid for all cases.

7.1 Analyzing beam maps

The calibration differs for jiggle maps and scan maps and it is also, although more weakly, dependent on chop throw. The relatively large difference in calibration for scan maps is due to the different chop wave form used for scan maps. The difference between a jiggle map with a 120″ chop throw compared to one with a 60″ chop throw is mostly dictated by duty cycle and to a lesser extent by changes in the beam. The beam is slightly broader with a 120″ chop throw, but the duty cycle (time spent on source) is also slightly lower, both of these factors decrease the efficiency for large chops.

In the following example we are going to look at beam maps of Uranus taken in stable night time conditions during three nights in late May, 2001. These maps have been extinction corrected, we have blanked out bad bolometers and corrected each map for pointing drifts. There are slight calibration differences from night to night, but for this purpose the difference is negligible. The final coadded beam maps were rebinned in *az* and are shown in Fig. 11.

A quick way to diagnose that the beam profile looks reasonable is to use KAPPA's psf. The task psf fits a radial profile, $A \times exp(-0.5 \times (r/\sigma)^{\gamma})$, where r is calculated from the true radial distance of the source allowing for ellipticity, σ is the profile width, and γ is the radial fall-off parameter. psf can also fit a standard Gussian profile. However, the JCMT beam is better described by a two or three component Gaussian (main lobe plus inner and outer error lobes) and psf therefore overestimates the Half Power Beam Width (HPBW). If we specify `norm=no` psf will also return the fitted peak value of the source.

```
 % psf norm=no
IN - NDF containing star images /@u120_lon_reb/ >
INCAT - Positions list containing star positions /@coords/ > !
COFILE - File of x-y positions /@coords/ > u120l.psf
     Mean axis ratio =   1.093
   Mean orientation of major axis =   52.96 degrees
     (measured from X through Y)
DEVICE - Name of graphics device /@xwindow/ >
   FWHM seeing = 14.72    arcsec
   Gamma =  2.153
   Peak value = 0.2477
```

This produces the plot shown in Fig. 12. The value of FWHM is 14.72″ across the minor axis. The geometrical mean is simply $\sqrt{\text{mean axis ratio}}$ times 14.72, i.e. the measured FWHM (including

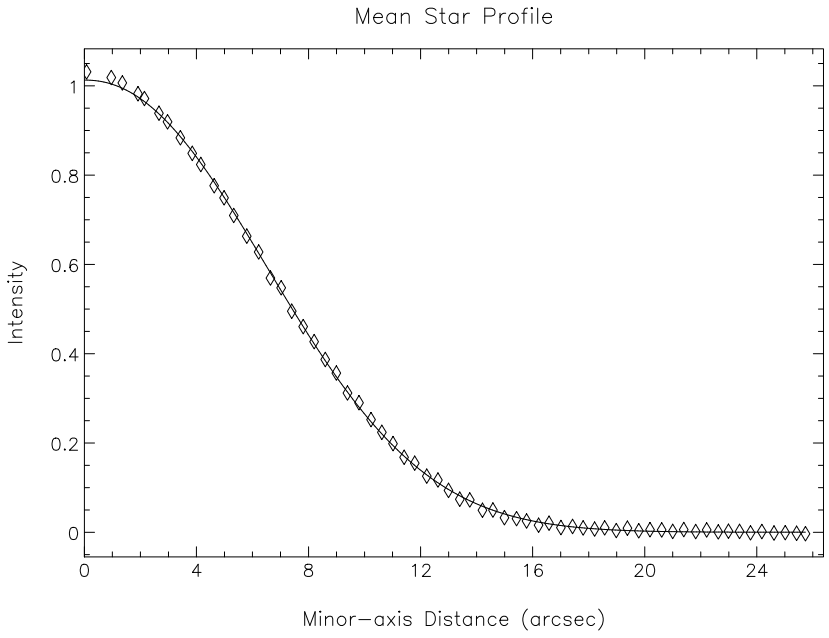


Figure 12: The radially averaged gaussian fit produced by psf

the broadening from Uranus) is therefore predicted to be 15.4″ if we use psf. However, if we fit a double Gaussian to the same data set we obtain 15.56″ × 14.30″ with a position angle of 85 for the main beam, and 55.8″ × 49.6″ for the inner error lobe. To find the true (HPBW) we need to remove the broadening caused by Uranus being an extended source. Using the program FLUXES (just type `fluxes` at the command line and answer the prompts) we find out that Uranus had a diameter (W) of 3.54″ that day. We convert the FWHM measured, θ_m, to the true HPBW of the telescope, θ_A, using the equation

$$\theta_A = \sqrt{{\theta_m}^2 - \frac{ln2}{2} \times W^2} \tag{4}$$

where W the diameter of the planet. In this case we get 14.5″ for the HPBW and ∼ 53″ for the near (inner) error beam. If we do the same for 450 μm we obtain 7.8″ for the HPBW and 34″ for the near error beam. These agree with nominal values for the telescope.

7.2 Calibrating in Jy/(solid angle)

If your maps show simple source morphology and you are only interested in integrated flux densities, the simplest approach is to calibrate your map Jy/aperture for the aperture size you want to use. The listing of FLUXES also gives us the total flux, S_{tot} for Uranus at 850μm is 67.9 Jy. Let us first see how we can use this value to calibrate our image in terms of Jy/arcsecond^2. In order to do this we need to derive a value for the Flux Conversion Factor (FCF) which is in units of Jy/arcsecond^2/V. To do this we first need to work out the sum of the pixel values (V_{sum}) in

an aperture of radius r. We then find the FCF is given by

$$FCF(Jy/\text{arcsecond}^2/V) = 67.9/V_{\text{sum}}a \quad (5)$$

were a is the pixel area in square arcseconds. The easiest way to get the integrated signal in an aperture is to use KAPPA's aperadd. For our 850 μm map of Uranus we derive V_{sum} for a set of different circular apertures and compute the FCFs.

Radius (arcseconds)	20	30	40	60	120
V_{sum}	45.75	60.76	64.89	70.07	77.08
FCF (Jy/arcsecond^2/V)	1.48	1.12	1.05	0.97	0.88

We can see from this table that the FCF is dependent on the aperture size that is used [1] This is because there is significant signal in the sidelobes and extended error beam of the telescope. Clearly then the value of FCF can be somewhat ambiguous. What you have to remember is that if you are doing photometry of an extended object, you should use a value for the FCF derived for the same aperture.

If you need to use small apertures, i.e. the size of your HPBW, you will need to use a point source or point like source as a calibrator. Flux densities for our secondary calibrators for a 40″ aperture are given by Jenness et al. [14]. However, several of our secondary calibrators are not point sources. If you end up with, for example, IRC+10216 and IRAS 16293−2422 or Mars near perhelion as your only calibrators during your run, you are in trouble. You may be able to use a large aperture to recover all the flux and use the ratios between different apertures derived for a point source. But, you may as well bite the bullet and calibrate in *Jy/beam*.

7.3 Calibrating maps in *Jy/beam*

If your images show a lot of structure, you will need to calibrate your maps in *Jy/beam*. This is true for most observations of dark and molecular clouds, young supernovae, protostars or young stars and even for nearby galaxies. However, if you are only dealing with faint point sources and low S/N maps, you probably need to integrate over the map. If this is the case, it does not matter whether you calibrate in *Jy/beam* or *Jy/aperture*, both methods will give the same result. Since Starlink packages do not deal with Jy/beam, it may appear more complicated to integrate over an image calibrated in Jy/beam, but the only difference is that one needs to normalize the integral over the source with the beam integral, $\int F(\Omega)_\nu d\Omega$, where F(Ω) is the normalized power pattern of the telescope. For a Gaussian beam the beam integral is simply $1.134 \times \theta_A^2$. Radio astronomical reduction packages of course do this normalization automatically. Since the JCMT beam is not a simple Gaussian beam, we need to account for the error beam, which is equivalent to having an FCF which varies with aperture, when we calibrate in *Jy/aperture*. We discuss how this is done towards the end of this section.

To calibrate in *Jy/beam* we have to know the beam size. Ideally we would derive both the flux density conversion factor and the beam size, θ_A, from planet observations. If there are no planets available, we can use one of the secondary calibrators. To determine the beam size at 850 μm it is usually sufficient to make a weighted average from our pointing observations during the run, if we don't have a planet observation or a point like secondary calibrator, but for 450 μm we

[1] During this time period SCUBA had reduced sensitivity due to a problem in the optics, affecting primarily the 850 μm array. Normally you would expect to find a FCF for a 40″ aperture of 0.84, see Jenness et al. [14]

need a planet or a secondary calibrator. All JCMT secondary calibrators are directly calibrated in *Jy/beam*. In this case the FCF is simply the quoted flux divided by the peak signal of the source.

7.3.1 Calibrating on Planets

For a planet we have to account for the loss of signal due to the coupling to the beam, because all planets used for calibration are extended relative to the JCMT beam. For our Uranus data the flux density S_{beam} is therefore the total flux density, S_{tot} divided by the coupling of the planet to the beam, given by:

$$K = \frac{x^2}{1 - e^{-x^2}} \tag{6}$$

where x is

$$x = \frac{W}{1.2 \times \theta_A}. \tag{7}$$

The FCF, in *Jy/beam/V* is therefore

$$FCF(Jy/\mathrm{beam}/V) = S_{\mathrm{beam}}/V_{\mathrm{peak}} \tag{8}$$

For 850 μm we find K = 1.021 for θ_A = 14.5″, which gives S_{beam} = 66.5 Jy/beam. The peak signal that we found for our high S/N Uranus map, V_{peak} = 0.2477 V, or an FCF = 268.5 Jy/beam/V. This FCF applies to a jiggle maps with a 120″ chop throw. If we do the same for our jiggle maps of Uranus with a 60″ chop throw, we derive FCF = 245.2 Jy/beam/V, i.e. a map with a 60″ chop throw is $\sim$ 10% more efficient than one with a 120″ chop throw. Even though Jenness et al. ([14]) found no difference in FCF as a function of chop throw when calibrating in *Jy/aperture* we find that the difference is now smaller than compared to when calibrating in *Jy/beam* but still noticeable. For a 40″ aperture the difference is 6%.

7.3.2 Using secondary calibrators

If we use a secondary calibrator to calibrate our maps, it is even simpler. We just take the quoted flux value, S_{beam}, from the secondary calibrator page and divide it with the peak flux in our map of the same calibrator. If the map of the calibrator has poor S/N, we may want to fit a gaussian to the source to get a more accurate measure of the peak signal.

7.3.3 How do we extract information from a map calibrated in *Jy/beam*

Analyzing maps calibrated in *Jy/beam* is easy; especially if we want to deduce flux densities for point sources or compact sources even when the source is embedded in a cloud with strong extended emission. For a point source the peak flux of the source is the same as the total flux corrected for any background emission. For an extended source we need to measure the FWHM and correct it for the measured HPBW of the telescope. We normally do this by fitting a double Gaussian, one for the source and one for the background. At 850 μm the fitted peak signal minus background, S_{peak}, is now the peak flux density measured in Jy/beam. From the fitted Full Width at Half Maximum (FWHM) we can derive the true FWHM, θ_s, by deconvolving with the measured HPBW, θ_A. This is trivial, because now we can assume a Gaussian source and a Gaussian beam. After we know the source size, θ_s we multiply the peak flux with the correction

factor we derive from the size, i.e. for a spherically symmetric source with the source size, θ_s, the total flux, S_{tot} is simply

$$S_{\rm tot} = S_{\rm peak} \times (1 + (\theta_s/\theta_A)^2) \tag{9}$$

.

For 450 μm the error beam amplitude is no longer negligible, but when we fit a double Gaussian, the error beam will blend in with the extended cloud emission, i.e. it adds into the background level, or we may fit the source with a single Gaussian plus a second order surface, or whatever best approximates the background in a limited area around the source. From our analysis of the 450 μm beam maps of Uranus, we find that the combined error beam amplitude is of the order of 5% of the peak amplitude, and we should therefore multiply the peak signal by 1.05 before applying a source size correction (see e.g. Weintraub et al. [17]).

To find integrated intensities over large areas is more complicated, because now we need to correct for the error beam pickup, which now depends on the area we integrate over. This is equivalent to the varying the FCF as a function of aperture that one has to account for if the map is calibrated in *Jy/pixel*, but with the map calibrated in *Jy/beam* it is much easier to separate compact sources and extended emission. To determine the excess emission from the error beam, we again have to go back to our beam map. If we calibrate our 850 μm map in Jy/beam and then integrate over 120" circular aperture, we find that the flux we derive is 86.8 Jy, while we know that the total flux of Uranus is only 67.9 Jy. We therefore have to scale our derived total flux density by the ratio of true flux density over measured flux density (for our calibrated planet map), which in this case is 0.78. At 450 μm the situation is much worse. Even though the amplitude of the error lobe is still low, the area is large, and if we integrate over our calibrated 450 μm beam map we now derive 415.6 Jy, if we integrate over the same 120" circular aperture, while the total flux density from FLUXES is only 179.3 Jy. In this case our scaling factor is 0.43, i.e. we pick up more emission in the extended error beam than we do in the main beam.

For careful work, you may therefore want to deconvolve your SCUBA maps. This becomes especially important if you want to ratio the 450 and the 850 μm maps, because if you want to smooth the 450 μm map to the same resolution as the 850 μm map, you first have to remove the error beam. For example of how this can be done, see e.g. Hogherheijde and Sandell [13] or Sandell and Weintraub [16].

8 Common error-messages – what have I done wrong?

The computer does not find the command task

You probably have forgotten to initialize SURF or KAPPA.

The computer does not recognize the command SURF

Let's assume that you have followed the manual, and you type `surf` to initialize the SURF software package, but nothing happens. You get a reply saying command not found, instead of the listing you saw in the manual. In this case you probably lack the Starlink initialization in your `.cshrc` and `.login` files. Type

```
% source /star/etc/login
% source /star/etc/cshrc
```

and you should be able to initialize SURF and KAPPA. The Starlink initialization scripts should be placed in your `.cshrc` and `.login` files if you intend to use SURF and KAPPA regularly:

i.e. in your `.login` file put the line:

```
if (-e /star/etc/login) source /star/etc/login
```

and in your `.cshrc` file put the line:

```
if (-e /star/etc/cshrc) source /star/etc/cshrc
```

See, for example, SUN/212 [3] for more information on using and installing Starlink software.

KAPPA tasks like centroid pick up the wrong image

This seems to occasionally happen when we switch between program packages. The cure is to delete the AGI [1] resource file, called `agi_computer.sdf`, where `computer` is the name of your computer, and the file resides in your home directory. In my case the file is called `/home/sandell/kala_agi.sdf`.

KAPPA display plots a new image inside an old image

This means that your AGI database has become corrupt (e.g. by using control-C to exit early from a task that displays graphics (scuover for example) – this prevents the task from performing normal cleanup duties). This problem can be fixed by issuing the KAPPA command gdclear or by removing the AGI database file from your home directory (see earlier fault).

I get a grayscale hardcopy, even though I specified a color device

Any color printer will need the color table supplied with the plot. In addition to setting the device to a color printer (e.g. epsfcol_p), you will also have to specify the color table with the parameter `lut`. In the example below we plot the final NGC 7129 image using the bgyrw color table, which resides in $KAPPA_DIR as bgyrw_lut

```
% display axes clear n7129_lon lut=$KAPPA_DIR/bgyrw device=epsfcol_p \
    mode=scale
```

This produces a color postscript file gks74.ps.n. A list of valid devices can be obtained with the KAPPA command gdnames.

Cannot create data array ...

Error messages of this type typically indicate that you are out of disk space – a quite common occurrence for anybody reducing SCUBA maps – or that you do not have write permission in your current directory.

References

[1] Eaton N., McIlwrath, B., 2000, *AGI–Applications Graphics Interface Library* AGI 8

[2] Jenness T., Lightfoot J. F., 1997, *SURF – SCUBA User Reduction Facility*, Starlink User Note 216 1

[3] Bly M. J., 2000, *Starlink Software CD-ROMs Spring 2000 User's Guide* Starlink Software CD-ROMs Spring 2000 User's Guide 8

[4] Currie M. J., 1997, *KAPPA — Kernel Application Package*, Starlink User Note 95 1

[5] Shortridge K., Meyerdierks H., Currie M. J., Clayton M., *FIGARO – A general data reduction system*, Starlink User Note 86 1

[6] Draper P. W., 1997, *GAIA – Graphical Astronomy and Image Analysis Tool*, Starlink User Note 214 1

[7] Lightfoot J. F., Harrison P. A., Meyerdierks H., 1995, *JCMTDR – Applications for reducing JCMT data*, Starlink User Note 132 6

[8] Duncan, W. D., SCUBA project documentation, SCU/WDD/31.1/1093

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

[10] Currie M. J., Privett G. J., Chipperfield A. J., 1995 *CONVERT – A format-conversion package*, Starlink User Note 55 1

[11] Archibald, E. et al. 2002, MNRAS, 336, 1 4.1

[12] Coulson, I., 2001 Scuba Jiggle Maps, `http://docs.eao.hawaii.edu/JCMT/SCD/SN/003`

[13] Hogherheijde, M.R., Sandell, G., 2000, ApJ, 534, 880 7.3.3

[14] Jenness, T., Stevens, J.A., Archibald, E.N., Economou, F., Jessop, N.E., Robson, E.I., 2002, MNRAS, 336, 14 5.2, 7.2, 1, 7.3.1

[15] Pierce-Price, D., et al., 2000, ApJ, 545, L121 6.2

[16] Sandell, G., Weintraub, D.A., 2001, ApJS, 134, 115 10, 7.3.3

[17] Weintraub, D.A., et al., 1999, ApJ, 517, 819 7.3.3