Reduction of pointing measurements

Location of the Raw Data

The raw data of the 100m Effelsberg telescope is stored in MBFITS-Format. In Effelsberg the files are located in the directory /daten/Raw which should be available on every **Observer-PC**. Older data can be found in /daten/Raw/Raw-YYYY-MM. Most programs listed here support the flag "fdir" to point the program to the directory where the MBFITS data can be found. Default is always /daten/Raw.

Every 30 minutes the raw data is synced to Bonn. It is accessible in /effbg/effdata/. More details on the location of data can be found here under Data storage and archive

Inspecting scans by hand using the Toolbox

The MBFITS data can be inspected with any pogram that understand FITS Format e.g. "fv" fits-viewer. You can look at the headers and tables and plot different data columns...

However, most users might prefer a kind of pre-reduced view where you see the amplitude of the scan calibrated in units of the calibration temperature and with real arcseconds for the scanning axis. This is provided by the "toolbox" program. It is currently in a development phase, but the latest stable version is running on all of the **observer-PCs** using user: **obs2**. Your normal MPI account should also work, the observer-PCs are connected to the /homes server, but you might need to adjust your PATH environment to get access to all programs.

The toolbox can be used interactively by calling

```
toolbox
```

from the command line. A file browser opens from which the scan can be selected (see Fig. 1). You can also select whether you want to have a ps-plot of the scan written to your current directory. By default you will find a .fit-file for every scan you looked at with some scan specific data and the fitted parameters.

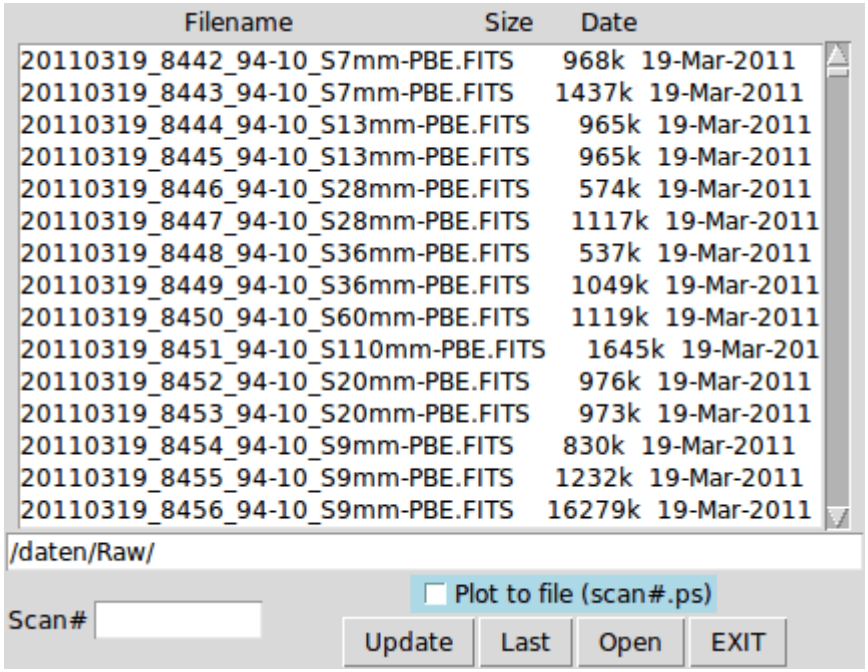

Figure 1 Toolbox file browser.

Content of fit-file:

```
#
# Source   Scan   direct  Subsc   MJD           LST             Azi
Elv       Amp         err        Offset      err     FWHM       err
ColS       NulE     Parangle    Tsys         Cal
3C286      4047    ALON    1  55497.2278085  30085.1592055     78.008
30.621  7.61715e-01  6.32183e-03     9.59      0.42     69.78      0.84
16.51      7.39    -46.17   8.673e+00  1.8370e+03
3C286      4047    ALON    2  55497.2281492  30114.6799258     78.097
30.697  7.66583e-01  5.80441e-03    -0.38      0.38     68.84      0.75
16.51      7.39    -46.19   8.668e+00  1.8373e+03
3C286      4047    ALAT    3  55497.2285233  30147.0884189     78.188
30.780  7.62980e-01  8.15973e-03     0.82      0.54     69.42      1.07
16.51      7.39    -46.21   8.659e+00  1.8389e+03
3C286      4047    ALAT    4  55497.2288663  30176.8003259     78.274
30.859  7.61237e-01  7.00963e-03     1.35      0.46     68.80      0.91
16.51      7.39    -46.23   8.652e+00  1.8386e+03
```

Another option, that is more appropriate for reducing a number of scans, is to use the toolbox from the command line:

```
toolbox scan=XXXX plot='/xs'
```

This will reduce scan XXXX and shows the plot in a PGPlot window (see Fig. 2).

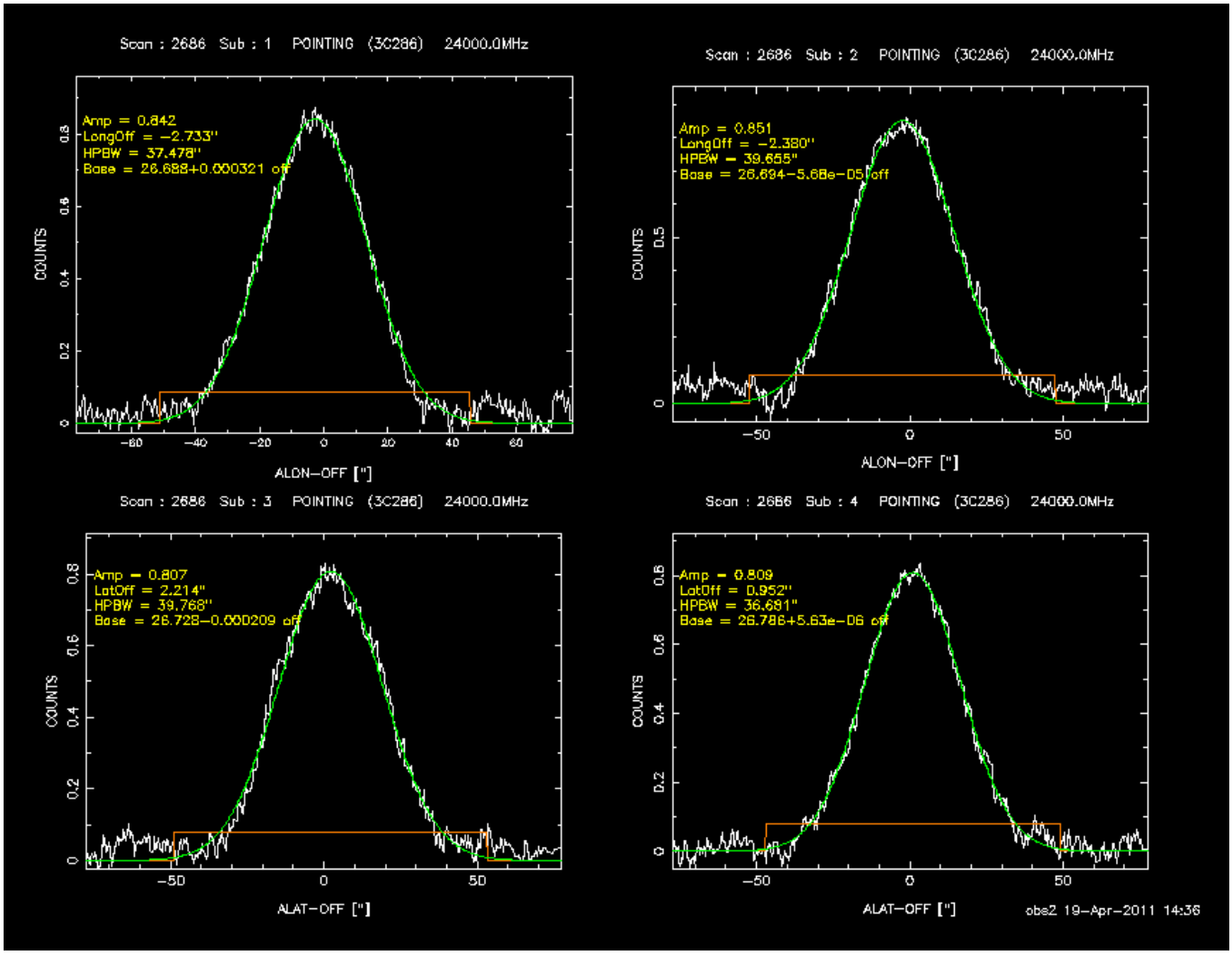

Figure 2 PGPlot window of a cross scan. The data is baseline subtracted and a Gaussian is fitted.

Toolbox Options

There are a number of options to control the toolbox. The order of their appearance on the command line is not important. Every keyword will be recognized.

Option	Comment
fdir='PATH'	Path to the MBFITS data
scan=xxxx	Direct selection of scan numbers. No File-Browser will be opened.
scan=last	Will be open the most recent scan.
sub=X	Displays sub scans X only., e.g. sub=2 shows sub-scan 2 only.
take=XXXX:x1,x2,x3,YYYY:y1,y2,y3,...	Reduces the given sub scan numbers of scan XXXX and YYYY together. All scans have to be specified with scan=XXXX,YYYY. Works also together with "aver" (see below)
del=XXXX:x1,x2,x3,YYYY:y1,y2,y3,...	Excludes the given sub scan numbers from the reduction of scan XXXX and YYYY. All scans have to be specified with scan=XXXX,YYYY. Works also together with "aver" (see below)

Option	**Comment**
plot	Enables postscript plots. Name of the plot: xxxx.ps, xxxx="scan number"
plot='/xs' or '/gif'	Opens plots in a PGPlot-XServer or writes gif-images: xxxx.gif
pol=u or q	in combination with plot the graphs for the Q and U stokes parameters will be shown
nos5	to reduce polarimetry data from receivers without the phase-switch (S5).
nofit	Displays the data as taken. No baseline subtraction and fitting is done.
baseline=[no,0,1, or 2]	Set the degree of the base line that is subtracted. no=no baseline, 0=constant offset, 1=liner fit, and 2=2nd degree polynomial.
cut=-x,x	Restricts the range of the Gaussian fit in scanning direction on the left (-x) and right (x). E.g. cut=0.1,0.1 will reduce the range by 10%.
aver	Causes an average over all longitude and latitude subscans
aver **and** scan=XXXX,YYYY,ZZZZ	Causes an average over all longitude and latitude subscans and scans (XXXX, YYYY, and ZZZZ). The resulting parameters will be written in XXXX+YYYY+ZZZZ.fit
use='(c[1]+c[2])/2' or '(fac1*c[1])+(fac2*c[2])/2'	Enable to add or subtract (e.g. for beam switch) any combination of available channels and to multiply them by factors. Useful to apply T_{cal} [K] to the measured counts to get T_A [K]. Which signals are exactly available is listed in the receiver folders in the control room and can be seen in the FEBEPAR header of the MBFITS file. As obs2 on Observer3 also FEBEinfo.py can be used to obtain information about the written channels.
spikes=X	Option to despike data with interference. X=hight of spike in multiples of the RMS (default 3). Despike is turned off if X=-1.
print	Writes out the data in ascii format. One file for every subscan with scan position, counts, and Gauss-fit is written.
fcent=x	Defines the starting point (centre) of the Gaussian fitting, 'x' in arcsec.
febe=XXX	Sets the frontend/backend combination, XXX, for scans that are observed with more than one.
fit=n	Number of iterations for the Gaussian fitting. Default is 2.
header	Prints the MBFIT header.

Option	Comment
horn=n	Defines the horn number 'n' to reduce. See also the "use" option, that is more flexible.
rfi=n	Will filter RFI peaks looking for peak increasing n times the rms.
window=n	Defines the window size in multiples of the beam width for the second iteration of the Gaussian fit (orange box in the plot window). Default is 1.22
mapping options:	For viewing maps or converting them to FITS or nod2 standards.
color	Shows color plots instead of black and white.
resize	Allows to plot the map on a new grid.
rfi=n	Will filter RFI peaks looking for peak increasing n times the rms.
tab=n	Allows to fill every n'th line with dummies for under sampled maps.
taper	Gaussian taper for the sinc-function.
xstart=n	Defines the start coordinate for the x-axis in degree.
more dangerous option:	"dangerous" means you have to know what you do
sig="(1,1)+(1,2)+(1,3)+(1,4)"	Definition of an individual formula to add channels and phases. Usually the phases 3 and 4 contain the calibration signal.
cal="-(1,1)-(1,2)+(1,3)+(1,4)"	Corresponding calibration formula. Phases 1 and 2 are subtracted to only get the calibration signal.
sig="(1,1)+(1,2)+(1,3)+(1,4)+(2,1)+(2,2)+(2,3)+(2,4)"	Another example: formula to add channel 1 and 2, e.g. LCP + RCP. But it is recommended to use the "use" option add or subtract channels.

Observing Logs

A program logging a number of useful information is running on the observer PC. If you are observing by your self you can just send the log to your email address or you can ask the operator to do so.

If you don't have that log there are a number of ways to obtain an observing log from previous dates.

ObsLog.py

In **/daten/ObsLogs** there are log-files for every observing day starting from 12 UT noon on the previous day. The files are named for each date YYMMDD.prot. A small script that can be used to search through these files is called **ObsLog.py**

```
 Usage: ObsLog.py [date=YYMMDD] or [fromto=YYMMDD,YYMMDD] and/or
[scan=from#,to#]
```

```
ObsLog.py date= and fromto= work without scan=, but scan= not without one of
the previous
```

It can be used to print out a specific day, many days, and restrict the scan number range. Using only scan numbers might be misleading because scan number repeat after 9999. ObsLog.py is the fastest way to produce a log, since it only searches through some ascii files.

For example:

```
obs2@observer3:~$ ObsLog.py date=110418
# 1    2         3      4            5    6      7      8     9    10    11
12     13   14     15    16    17  18    19   20    21  22  23 24
#Scan Date       UTC  Source        Azm   Elv   Freq   col*  Nule Nula  X-
lin Y-lin Z-lin X-rot Y-rot Opos  Temp Hum  Press Ref   wv  wd  ts es
....
2684 2011-04-18 01:13 3C286         236.9 61.0  24.00  4.3   2.1   0.0   0.0
27.0  -7.8  28.0   0.0   0.0  3.7 70.0 984.6 62.0  0.9  98  4  4
2685 2011-04-18 01:15 3C286         237.8 60.7  24.00  2.6  -1.9   0.0   0.0
27.0  -7.8  28.0   0.0   0.0  3.7 70.0 984.6 62.0  0.3 157  2  2
2686 2011-04-18 01:17 3C286         238.2 60.5  24.00  2.6  -1.9   0.0   0.0
27.0  -7.3  28.0   0.0   0.0  3.7 70.0 984.6 62.0 -0.2 208  4  4
2687 2011-04-18 01:25 W3MAIN        369.2 23.5  23.41 -0.0  -0.5   0.0   0.0
27.0  -7.3  28.0   0.0   0.0  3.7 70.0 984.1 62.0  0.0 158  2  2
2688 2011-04-18 01:35 3C345         131.9 75.4  23.41 -0.0  -0.5   0.0   0.0
27.0  -7.3  28.0   0.0   0.0  3.5 72.0 984.1 63.0 -0.0 164  4  4
2689 2011-04-18 01:38 3C345         133.5 75.7  23.41  1.2   7.7   0.0   0.0
27.0  -7.3  28.0   0.0   0.0  3.5 71.0 984.1 62.0 -0.0 191  2  2
2690 2011-04-18 01:39 3C345         134.2 75.8  23.41  1.2   7.7   0.0   0.0
27.0  -7.6  28.0   0.0   0.0  3.5 71.0 984.1 63.0 -0.0 238  4  4
2691 2011-04-18 01:42 3C345         136.1 76.2  25.06  0.4   6.1   0.0   0.0
27.0  -7.5  28.0   0.0   0.0  3.6 71.0 984.1 62.0  0.4 230  4  4
2692 2011-04-18 01:45 3C345         137.8 76.4  25.06 -0.1   7.3   0.0   0.0
27.0  -7.5  28.0   0.0   0.0  3.6 71.0 984.1 62.0  0.5 232  2  2
....
```

Logbook.py

Logbook.py lists by default the last 1000 scans. This might take some time, because Logbook.py is reading the original MBFITS data files to get the information. Using

```
Logbook.py 100
```

it lists only the last 100 scans. The time range can be restricted by the option

```
Logbook.py 10000 tstart=YYYY-MM-DD tstop=YYYY-MM-DD
```

where "tstart" marks the more recent time (if not specified the print out starts today) and "tstop" specifies the stop time in the past. The 10000 is just chosen to be big enough to cover most time

ranges. Logbook.py can be used only to produce logs from more recent data because it searches only in /daten/Raw.

For example:

```
obs2@observer3:~$ Logbook.py tstart=2011-04-18 tstop=2011-04-18
# SCAN   SUB   OBJECT         SCANTYPE    FEBE            PROJECT
DATE_OBS              MJD
....
  2692    2    3C345           FOCUS      P13mm-PBE       46-09
2011-04-18T01:45:17   55669.0731
  2691    4    3C345           POINT      P13mm-PBE       46-09
2011-04-18T01:42:56   55669.0715
  2690    4    3C345           POINT      P13mm-PBE       46-09
2011-04-18T01:40:05   55669.0695
  2689    2    3C345           FOCUS      P13mm-PBE       46-09
2011-04-18T01:38:44   55669.0686
  2688    4    3C345           POINT      P13mm-PBE       46-09
2011-04-18T01:36:22   55669.0669
  2687    2    W3MAIN          ONOFF      P13mm-FFTS      46-09
2011-04-18T01:26:08   55669.0598
  2686    4    3C286           POINT      P13mm-PBE       46-09
2011-04-18T01:17:28   55669.0538
  2685    2    3C286           FOCUS      P13mm-PBE       46-09
2011-04-18T01:16:08   55669.0529
  2684    4    3C286           POINT      P13mm-PBE       46-09
2011-04-18T01:13:45   55669.0512
....
```

log.py

log.py is a small script that prints some more detailed information than Logbook.py.

```
 Usage: log.py scan1 scan2 [fdir=<PATH>]

 Options:
 fdir=<PATH>: if the data are not in /daten/Raw any more
```

It can be also used to produce a log from older data in the archival directories, /datan/Raw/Raw-YYYY-MM specifying "fdir".

For example:

```
obs2@observer3:~$ log.py 2684 2692
# ***  Date: 2011-04-18  Project-ID: 46-09     Observer-Operator: ALEX
***
# **  Temp =   3.7 deg C     Humid = 70.0 %     Press =  984.6 hPa
#Scan Sub Source         UT       LST    FE-BE        Mode       Freq
Type  Sw-mode AZI  ELV   Col*  NULE  z-Lin
 2684  4  3C286        01:13:45 15:23:59 P13mm-PBE    CONTINUUM  24000.00
```

```
POINT   TOTP 236.9 61.0   4.3   2.1  -7.8
 2685  2  3C286        01:16:08 15:26:36 P13mm-PBE    CONTINUUM  24000.00
FOCUS   TOTP 237.8 60.7   2.6  -1.9  -7.8
 2686  4  3C286        01:17:28 15:27:44 P13mm-PBE    CONTINUUM  24000.00
POINT   TOTP 238.2 60.5   2.6  -1.9  -7.3
 2687  2  W3MAIN       01:26:08 15:36:25 P13mm-FFTS   SPECTROSCO 23407.18
ONOFF   TOTP 369.2 23.5  -0.0  -0.5  -7.3
 2688  4  3C345        01:36:22 15:46:40 P13mm-PBE    CONTINUUM  23407.18
POINT   TOTP 131.9 75.4  -0.0  -0.5  -7.3
 2689  2  3C345        01:38:44 15:49:18 P13mm-PBE    CONTINUUM  23407.18
FOCUS   TOTP 133.4 75.7   1.2   7.7  -7.3
 2690  4  3C345        01:40:05 15:50:24 P13mm-PBE    CONTINUUM  23407.18
POINT   TOTP 134.2 75.8   1.2   7.7  -7.6
 2691  4  3C345        01:42:56 15:53:15 P13mm-PBE    CONTINUUM  25056.25
POINT   TOTP 136.0 76.1   0.4   6.1  -7.5
 2692  2  3C345        01:45:17 15:55:56 P13mm-PBE    CONTINUUM  25056.25
FOCUS   TOTP 137.7 76.4  -0.1   7.3  -7.5
```

Reducing a number of scans at once

Looking at single scan might be appropriate for checking the data during an observation, but for calibration or flux density monitoring a more automatic way is preferred. There is a collection of scripts and programs that can be used to perform all the tasks to obtain flux density calibrated data. The scripts are located in /home/obs2/bin on the observer3-PC.

Raw Data Processing

The scripts and programs mostly use a file that contains all the scan numbers to be reduced. The scripts **log2scan.py** or **logbook2scan.py** produce such lists from observing logs written by log.py or Logbook.py, respectively.

In log2scan.py one can optionally restrict the frequency if the log contains entries at different frequencies.

```
log2scan.py
 Usage: log2scan.py log-file [freq GHz]
```

The script **reduce.py** can be used to reduce a number of given scans using the toolbox with a list of options. Some example parameter files are stored in /home/obs2/flux_monit/reduce-par. E.g., reduc28.par for the 2.8cm SFK receiver. Calling just reduce.py prints out some help as well.

```
obs2@observer3:~$ reduce.py

 Usage: reduce.py par-file
```

```
 The par-file consists of two parts

 1. The Toolbox options: e.g.
  # Toolbox options start here
  start_options
  #
  # Beam switch
  use='(c[1]+c[2]-c[5]-c[6])/2'
  #
  # Use XServer from PGPlot
  plot='/xs'
  #
  # Average scans in ALON and ALAT
  aver
  end_options

 2. A scan list:
  #
  # sub-scans can be deleted by the del= option
  # in the second line: e.g.
  # scan=0001
  # del='1,4,5'
  # to delete sub-scans 1, 4, and 5
  #
  # New root dir for data can be specified as e.g.
  # fdir=/daten/Raw/Raw-2011-01
  # default is /daten/Raw
  scan=0001
  scan=0002
  scan=0003
```

For example:

```
> reduce.py reduc28.par

Remove previous fit-files

Start reducing data:
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8464
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8465
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8479
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8480
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8494
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8495
toolbox  use='(c[1]+c[2])/2' plot='/xs' scan=8511
...
```

will average over channel 1 and 2 and plot the scans in a PGPlot window. If you prefer to save the figures instead of looking at them as they are being processed try just "plot" or "plot='/gif'". All options from the table can be specified in options section of the parameter file. In the scan section each scan entry can be followed by a delete line to exclude some subscans from the data reduction.

That might help in case of RFI or short term weather effects. The keyword "fdir" can be used to change the PATH to the data. It has to be given only once and the following scan numbers will be all processed from this directory.

The resulting fit-files have an entry for the system temperature. This can be used to correct the data for opacity effects (see next section). Since Tsys is calculated from the baseline parameters it only works when a baseline was fitted and it was done in total power. If one uses a software beam switch the baseline will just be the residuals between the two horns and does not correspond to Tsys any more. In the case you want to use the software beam-switch to improve the fitting of the Gaussian it might be wise to first do one run with total power to produce a Tsys-file and then run the procedure again using software beam-switch to obtain a better fit.

Correct for Opacity and Pointing Offset

A short introduction to the background of the further steps of the data calibration can be found here. The procedure described here that fits the lower envelope of the Tsys vs. airmass distribution only works if you have several scans covering a larger range of elevations. An alternative would be to perform skydips during the observations to measure the opacity and and lower frequencies one can also use the common values from our receiver page.

Since the amplitudes from the toolbox are in units of the Tcal one just have to multiply the given numbers by the Tcal to get the correct system temperature. The reduce.py procedure provides an all.fit file that contains all the single fit-files. The further programs work with this all.fit.

The script **weather.py** reads the weather information for the MBFITS files, multiplies the Tsys found in the all.fit by a given Tcal and prints out a weather.dat and a LIST.tsys, with the opacity information for each scan that is listed in the scanlist-file. The minimum zenith opacity is fitted from the Tsys vs. airmass distribution and is presented in the plot Opacities.eps. Weather.eps contains the weather info.

```
obs2@observer3:~$ weather.py
 Task to read weather data from MBFITS files,
 save them in ASCII format, and compute LIST.tsys.

 Usage: weather.py scan-file [Tcal] [fdir=<PATH>]


 The scan-file should contain the scan numbers
 to be reduced.
 Comments can be inserted with a leading '#', e.g.:

 # Good pointing scans
 2567
 2568
 #2569 removed because of RFI!
 2570
 ...

 Options:
```

```
 Tcal: temperature of the noise diode to get correct Tsys (default=1)
 fdir=<PATH>: if the data are not in /daten/Raw any more
```

The script **corr_point.py** allows to apply the opacity corrections from LIST.tsys or to apply a single value for all data. The question how to proceed will be ask when the the script is started. The amplitudes will be corrected for pointing offsets. Offsets in longitude will be applied to the latitude data and vice versa. To calculate the correction the actual FWHM of the Gaussian fit will be used. If you don't trust that fit the value after the Tcal is interpreted as the FWHM to use.

```
obs2@observer3:~$ corr_point.py

 Task to reduce fit-files from Peters Toolbox

 Usage: corr_point.py scan-file [Tcal (K)] [FWHM (asec)]

 The scan-file should contain the scan numbers
 to be reduced.
 Comments can be inserted with a leading '#', e.g.:

 # Good pointing scans
 2567
 2568
 #2569 removed because of RFI!
 2570
 ...
```

For example the Tcal at 2.8cm is 7.5 K and scan numbers are stored in scanlist

```
obs2@observer3:~/ubach/measurements/2011_03_16_poi/2.8cm/0.Raw$
corr_point.py scanlist 7.5

 ****************************************************
 *          Procedure to analyse Cross-scans        *
 *          from the 100m Effelsberg telescope      *
 *                                                  *
 *        Version 3.4  (I. Marti-Vidal)             *
 *                     (T. Krichbaum)               *
 *                     (U. Bach)                    *
 *                                                  *
 *   adapted from A. Kraus corr_point.f Ver. 2.7    *
 *                                                  *
 ****************************************************


 Raw data must be in file: all.fit

 Apply opacity correction?

 No                                -> [n]
 Apply a single value              -> [s]
```

```
 Apply computed values (LIST.tsys) -> [f]
n

 No opacity correction!

Tcal=7.50 [K]

 Write LIST.raw

 Write Pointing.dat

In total there have been 312 data sets in 156 Scans.

   <FWHM>    = 69.09 +- 1.42"
   <Off_LON> = -1.13 +- 5.36"
   <Off_LAT> =  0.43 +- 5.56"

 Write SCAN-AVG.dat
```

The applied corrections are stored in Pointing.dat the data for further processing in LIST.raw and the averages over LON and LAT of the raw scans are written in SCAN-AVG.dat. The LIST.raw file now contains one entry for each scan with the following information.

```
#  JD      Scan Source         Flux     Err     Azi.  Elv.   UT     LST
par.Ang
#-------------------------------------------------------------------------
----
#
15519.4576 6894 3C454.3       27.5893  0.1693  261.8  27.7  22.98   3.30
41.0
15622.2729 6896 3C48           2.6540  0.0159  271.8  43.7  18.55   5.62
49.4
15622.3106 6905 3C295          2.6072  0.0189   38.4  26.4  19.45   6.53
-40.0
15622.3167 6908 3C295          2.5999  0.0148   39.5  27.3  19.60   6.68
-41.2
15622.3370 6914 3C295          2.6271  0.0156   43.3  30.4  20.09   7.16
-45.2
....
```

The next step will be to correct this data for the gain elevation dependence and to convert it from Kelvin to Jansky.

Final Calibration

For frequencies below 2.5 GHz the gain of the Effelsberg antenna is more or less constant over the whole elevation range, but at high frequencies a gain curve should be applied to the data to correct for the losses. The gain curve for each receiver is given on the receiver page. The Fortran-program

eff_flux can be used to correct the data for various effects including the gain curve and to calculate and apply the conversion factor from Kelvin to Jansky.

The program requests an input file called "eff_flux.par". It is structured as follows, sorry most of the variable are called in German, but some comments are introduced on the right side here to explain the meaning.

```
##Control file for eff_flux
ANZAHL QUELLEN: [  5]                          # number of sources
##Source name: 8 characters
3C273
3C279
3C286
3C295
3C309.1
3C345
4C39.25
NRAO150
OJ287
##---------------------------------------------
ANZAHL ELEVATIONSKORREKTUREN: [ 1]                # number of elevation
corrections
##source name ('0836+71') or 'allsour', JD time interval, poly. coeff. A0-A5
'allsour' 0.0 99999.9 0.88196 6.6278E-3  -9.2334E-5 0.0 0.0 0.0
##---------------------------------------------
KORREKTURKURVE (y/n): [n]                         # is there a free
correction curve*
##---------------------------------------------
ANZAHL ZEITKORREKTUREN: [ 2]                      # number of time
correction
##source (s.o.), JD time interval, coeff. A0-A3, and p,B1,B2 with
## B1*sin(p*t) und B2*cos(p*t)
'allsour' 0.0 99999.9 1.000 0.0 0.0 0.0 0.0 0.0 0.0
'NGC7027' 0.0 99999.9 0.981 0.0 0.0 0.0 0.0 0.0 0.0 # constant factor to
correct fora partly
                                                   # resolve calibrator
##---------------------------------------------
ANZAHL KALIBRATIONEN: [ 1]                         # number of calibrations
##AJD time interval, cal.-factor,-error
0.0 99999.9 1.0 0.0
##---------------------------------------------
ANZAHL KALIBRATORQUELLEN: [ 8]                    # number of prime
calibrators to calculate K/Jy
## name and flux density in Jy ('3C286'  7.58)
'3C286'     2.82
'3C295'     1.18
'3C196'     0.91
'3C138'     1.57
'3C48'     1.40
'3C161'     1.57
'3C147'     2.07
```

```
'3C123'     3.79
```

* the correction curve should be named "Corr_curve" and has the following form: three columns with JD corr-fact. err

Some example eff_flux.par files can be found in /home/obs2/flux_monit/eff_flux-par/. The eff_flux program creates several files.

- LIST.corr contains the corrected entries from LIST.raw
- FLUX.<source name> contain the individual calibrated scans of each source
- Averages gives a summary for all soruces
- Calibrators reports the data of prime calibrators and states the conversion factor from Kelvin to Jansky. The factor can then be written into the eff_flux.par. After the value was entered in eff_flux.par the Calibrators file should give 1.0 as the calibrations factor.

The output of the program looks as follows

```
obs2@observer3:~$ eff_flux

 ****************************************************
 *        Analysis of flux density                  *
 *      measurements made at the 100m               *
 *      radio-telescope in Effelsberg               *
 *                                                  *
 *                Version 2.4                       *
 *            Alexander Kraus, MPIfR                *
 *                                                  *
 ****************************************************
 Files needed:
 LIST.raw (total-power-Data) and
 the commandfile Command.tot.
 Maybe a time-dependent correction (File Corr_curve).
 Commandfile read!
 Starts to read data from LIST.raw
 100 data sets read!

 Derives averages!!

 Starts writing results!!
```

Done!