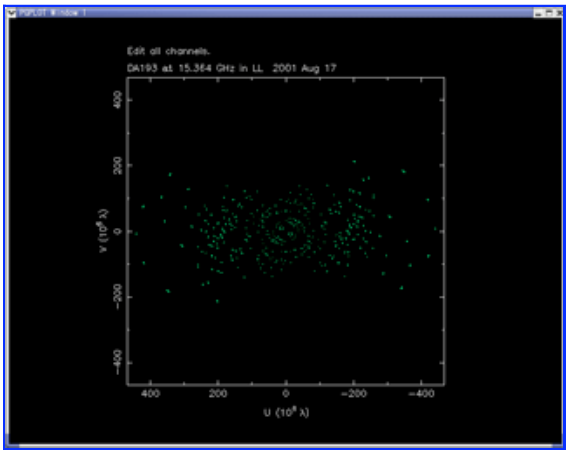
Asian Radio Astronomy Winter School 2007
VLBI Data Reduction Course

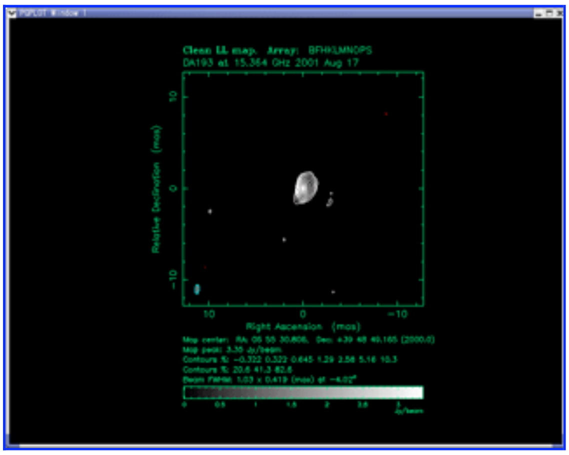
Written by Seiji Kameno (Kagoshima University, Japan)
Translated and edited by Tomoharu Kurayama
(VERA, National Astronomical Observatory of Japan, Japan)

Japanese version

We are going to analyze the data observing a simple-structure radio source DA193. The observation frequency is 15 GHz. The VLBA stations are used, and the observation time is only about 20 minutes. Although this is very short observation, the map dynamic range exceeds 1000. At first, let' s see the results.



(u, v) plot



Final image

Observation parameters

Observation date	2001 August 17
VLBI array	VLBA (10 stations)
Observed sources	DA193 (Note: Originally the target source of this observation is NGC1052. DA193 is observed for calibration.)
Observation frequency	15.4 GHz
Bandwidth	16 MHz × 2 IF, 64 frequency channels/IF
Data size	97.3 MB

Procedure of analysis

We are going to analyze the data with AIPS and difmap. We carry out the data calibration mainly with AIPS, and imaging and self-calibration with difmap.

Next

Lesson 1 : Loading data into AIPS

Previous Top Next

1. Preparition before starting AIPS

1.1. From login to starting AIPS

After you login the PC, start xterm. AIPS runs only on xterm. (Problems remain the other terminals.) Move the directory to $AIPS_ROOT on your xterm. Here, the hostname of PC is “kaimon”, the user account of PC is “kameno”, and the AIPS user ID number is 3018.

```
kaimon[kameno]5: cd /usr/local/aips
/usr/local/aips
```

Let’s define the environment variable in order to start AIPS. Your login shell is tcsh, so

```
kaimon[kameno]6: source LOGIN.CSH
kaimon[kameno]7: $CDTST
AIPS_VERSION=/usr/local/aips/31DEC06
```

These commands are same if your login shell is csh. If your login shell is bash,

```
kaimon[kameno]6: . LOGIN.SH
kaimon[kameno]7: $CDTST
AIPS_VERSION=/usr/local/aips/31DEC06
```

Start AIPS. You must input your AIPS user ID when you are asked.

```
kaimon[kameno]8: aips tv=local
START_AIPS: Will use or start first available Unix Socket based TV

< many massages... >

AIPS 2: Enter user ID number? 3018

< some messages... >
```

When you succeed the starting, the command prompt ">" is shown in your xterm window (the window you type the command aips). From here, we call this window as the "main window". You can also find the window of "X-AIPS TV Screen Server" (TV window), "Message Server" (message window) and "TK Server". These windows may be minimized in the bottom of your display.

2. Basic operation of AIPS

When you can see the command prompt ">" in the main window, you can type the commands of AIPS. We have two kinds of commands in AIPS, task and verb.

2.1. Task

Tasks are the commands which have many parameters. The procedure to run a task is:

1. Declaring a task by typing task '(task name)' or tget (task name).
2. Setting parameters.
3. Running the task by typing go.

Examples of tasks are the following:

FITLD	Loading FITS files
FRING	Carrying out the fringe fitting
POSSM	Showing spectra
TACOP	Copying extension tables

2.2. Verb

Verbs are the simpler commands which have not so many parameters. When you run a verb, please type the name of the verb. Examples of verbs are the following:

PCAT	Listing AIPS files
IMHEADER	Showing the headers of AIPS files
INP	Listing the parameters of the current task
TGET	Loading a task and setting parameters in the previous one

In this course, we use CAPITAL LETTERS for the name of tasks, verbs, and parameters. Commands you type on your keyboard are displayed with green, underlined characters
.

3. Loading FITS files into AIPS

Let's load the FITS file into AIPS file system with the task FITLD. Type the underlined parts in the following box.

If you do not have the data file in $AIPS_ROOT/FITS, please download it from http://astro.sci.kagoshima-u.ac.jp/omodaka-nishio/member/kameno/AIPS/BK084.DA193.FITS (97.3 MB FITS file).

```
> task 'fitld'              Declaring that we use the task FITLD.
> inf 'fits:BK084.DA193.FITS'
                           Specifying the input file name.  The part “fits:” represents $AIPS_ROOT/
> outn 'bk084'             Specifying the file name in AIPS file system.
> outcl 'uvdata'
> wtthresh 0.1             Abandoning the visibilities whose weights are less than 0.1.
> inp                      Checking the list of parameters.
 (Here is the parameter list of task FITLD.)
```

Type "> go." AIPS runs FITLD. Messages of FITLD are written in your MSGSRV window. FITLD is successfully finished if you can see the following messages in your MSGSRV window.

```
KAIMON> FITLD1: Appears to have ended successfully
KAIMON> FITLD1: kaimon       31DEC06 TST: Cpu=       4.1  Real=       5
```

4. AIPS file system

Let’s check the loaded file in AIPS file system. You can use the verb PCAT.

```
> pcat
AIPS 1: Catalog on disk  1
AIPS 1:  Cat Usid Mapname      Class    Seq  Pt     Last access      Stat
AIPS 1:    1 3018 BK084       .UVDATA.     1 UV 11-SEP-2006 17:30:52
```

There is an AIPS file “BK084.UVDATA.1” shown above. AIPS file system has three parts in the filename: name, class and sequence (seq). Names and classes are arbitral name, and sequences are numbers. Each name must be eight or less ASCII characters. Each class must be six or less ASCII characters. In the above example, the name is “BK084”, the class is “UVDATA”, and the sequence is 1. These coincide the parameters OUTNAME, OUTCLASS and OUTSEQ in the task FITLD.

When you specify the file in AIPS, you need to tell name, class and sequence correctly. This is confusing, so you can specify AIPS files with catalog number. The first column, below the “Cat”, in the list shown by PCAT is catalog number. The catalog number of “BK084.UVDATA.1” is 1. You can specify the file with catalog numbers with the verb GETNAME.

```
> getn 1
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.UVDATA.1
```

5. Sorting data

In the file loaded FITLD, the order of visibility data is sometimes not appropriate. However, some tasks require that the order of visibility data must be the two-dimensional array. This visibility order is called “TB order.” You must sort visibility data with the task MSORT.

```
> task 'msort'           Declaring the we use the task MSORT.
> getn 1                 The input file is catalog number 1 (BK084.UVDATA.1).
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.UVDATA.1
> outn=inn               The name of the output file is same as that of the input file (BK084).
> outcl='msort'          The class of the output file is MSORT.
> inp                    Checking the list of parameters.
 (Here is the parameter list of task MSORT.)
```

Type "> go." AIPS runs MSORT. Messages of MSORT are written in your MSGSRV window. After AIPS finishes MSORT, check the files with PCAT. You can see a new AIPS file.

```
> pcat
AIPS 1: Catalog on disk  1
AIPS 1:  Cat Usid Mapname      Class   Seq  Pt     Last access      Stat
AIPS 1:    1 3018 BK084       .UVDATA.    1 UV 11-SEP-2006 17:34:40
AIPS 1:    2 3018 BK084       .MSORT .    1 UV 11-SEP-2006 17:34:40
```

Previous Top Next

Lesson 2 : Confirming the data

Previous Top Next

Before we start the analysis, let's see the outline of data. We are going to see the observed date, observed frequency, observed stations, (u, v) coverage, observed sources, observed time, etc.

1. Showing the header of data (IMHEADER)

AIPS files consist of three parts: header part, data part and extension table part. Basic parameters are written in the header part. The values of visibilities (in the case of image file, the values of brightness) are written in the data part. Information of antennas and calibrations is written in the extension table part.
You can list the header with the verb IMHEADER. Let's list the header of the file whose catalog number is 2 with IMHEADER.

```
> getn 2
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> imh
AIPS 1: Image=MULTI     (UV)         Filename=BK084       .MSORT .   1
AIPS 1: Telescope=VLBA               Receiver=VLBA
AIPS 1: Observer=BK084               User #= 3018
AIPS 1: Observ. date=17-AUG-2001     Map date=11-SEP-2006
AIPS 1: # visibilities     31439     Sort order  TB
AIPS 1: Rand axes: UU-L-SIN  VV-L-SIN  WW-L-SIN  TIME1  BASELINE
AIPS 1:            SOURCE  FREQSEL  INTTIM  GATEID  CORR-ID  WEIGHT
AIPS 1:            SCALE
AIPS 1: ----------------------------------------------------------------
AIPS 1: Type    Pixels   Coord value     at Pixel     Coord incr   Rotat
AIPS 1: COMPLEX      1   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: STOKES       1  -2.0000000E+00       1.00 -1.0000000E+00    0.00
AIPS 1: FREQ        64   1.5348000E+10       0.62  1.2500000E+05    0.00
AIPS 1: IF           4   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: RA           1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: DEC          1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: ----------------------------------------------------------------
AIPS 1: Coordinate equinox 2000.00
AIPS 1: Maximum version number of extension files of type HI is   1
AIPS 1: Maximum version number of extension files of type FQ is   1
AIPS 1: Maximum version number of extension files of type AT is   1
AIPS 1: Maximum version number of extension files of type CT is   1
AIPS 1: Maximum version number of extension files of type OB is   1
AIPS 1: Maximum version number of extension files of type WX is   1
AIPS 1: Maximum version number of extension files of type AN is   1
AIPS 1: Maximum version number of extension files of type CL is   1
AIPS 1: Maximum version number of extension files of type CQ is   1
AIPS 1: Maximum version number of extension files of type FG is   1
AIPS 1: Maximum version number of extension files of type GC is   1
AIPS 1: Maximum version number of extension files of type IM is   1
AIPS 1: Maximum version number of extension files of type MC is   1
AIPS 1: Maximum version number of extension files of type PC is   1
AIPS 1: Maximum version number of extension files of type SU is   1
AIPS 1: Maximum version number of extension files of type TY is   1
```

In the above header, we can find the following information:

- This data is observed with the VLBA on 2001 August 17. (Telescope=VLBA, Observ.

date=17-AUG-2001)
- Total number of visibilities is 31439. (# visibilities 31439)
- Observed frequency is 15.348 GHz. The total number of IFs is four. Each IF has sixty-four frequency points. (FREQ 64 1.5348000E+10, IF 4)
- This file has 16 extension tables as follows:
  - HI : History of analysis
  - FQ : Frequency information
  - CT : CALC table (input parameters for the program CALC which carry out the phase/delay tracking)
  - OB : Orbit (This table is appended to treat the data of space VLBI.)
  - WX : Weather information
  - AN : Antenna information
  - CL : Calibration table
  - FG : Flagging table (list of bad data)
  - GC : Gain curve (antenna gain information)
  - IM : Interferometry model (result of phase/delay tracking)
  - MC : Model components (calculation model for IM table)
  - PC : Phase calibration table
  - SU : Source information
  - TY : Tsys (system temperature)

See the page of NRAO AIPS Cookbookfor the details of extension tables.

2. Creating NX (index) table

In order to see further information, we need to create the NX (index) table. This table is created by the task INDXR.

```
> task 'indxr'              Using the task INDXR.
> getn 2                    Choosing the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> cparm 0, 0, 0.1, 1        First argument is the allowance of scan gap (0 means 10 minutes).
                            Second argument is the maximum duration of scan (0 means
                            60 minutes).
                            Third argument is the time interval of CL (calibration)
                            table [min].
                            Fourth argument means whether AIPS carry out the re-calculation
                            of delay.
> inp               Checking the list of parameters.
 (Here is the parameter list of task INDXR.)
```

Carry out INDXR by typing "> go". INDXR finishes immediately with writing messages like these. Check whether NX table is created with imheader.

3. Listing scan information (LISTR)

“Scan information” means which source is observed in each observing time. Scan information is listed by the task LISTR.

```
> task 'listr'              Using the task LISTR.
> getn 2                    Selecting the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> optyp 'scan'              Specifying the listing of scan information.
> inp                       Checking the list of parameters.
 (Here is the parameter list of task LISTR.)
```

Type "> go". AIPS runs LISTR and lists the scan information on the main window.

```
 kaimon    LISTR(31DEC06)   3018     11-SEP-2006  17:42:56    Page    1
File = BK084       .MSORT .   1 Vol = 1  Userid = 3018
Freq = 15.348000000 GHz   Ncor =  1   No. vis =     31439
Scan summary listing

Scan       Source      Qual  Calcode Sub         Timerange          FrqID   START
   1 DA193           : 0000  B        1  0/12:58:14 -   0/13:03:35     1        1
   2 DA193           : 0000  B        1  0/14:17:30 -   0/14:22:51     1     8078
   3 DA193           : 0000  B        1  0/15:30:52 -   0/15:36:09     1    15161
   4 DA193           : 0000  B        1  0/16:38:09 -   0/16:43:28     1    23207

Source summary
Velocity type = 'GEOCENTR'    Definition = 'OPTICAL '

  ID Source           Qual  Calcode RA(2000.0)     Dec(2000.0)  No. vis
   6 DA193           : 0000   B     05:55:30.8056  39:48:49.165   31439

  ID Source            Freq(GHz) Velocity(Km/s) Rest freq (GHz)
   6 All Sources         15.3480         0.0000          2.2660
     IF(  2)             15.3560         0.0000          2.2660
     IF(  3)             15.3640         0.0000          2.2660
     IF(  4)             15.3720         0.0000          2.2660

Frequency Table summary
FQID IF#      Freq(GHz)      BW(kHz)   Ch.Sep(kHz)  Sideband
   1   1      15.34800000    8000.0005    125.0000      1
       2      15.35600000    8000.0005    125.0000      1
       3      15.36400000    8000.0005    125.0000      1
       4      15.37200000    8000.0005    125.0000      1
AIPS 1: Resumes
```

From the list above, you can find the total number of scans is four and observed source is all DA193. The time of scan is written in the column of "Timerange." "0" before slash ("/") means 0 days (relative days from the start of the observation). Times after the slashes are displayed in UT (Universal Time). We specify the time with this format. You also find other information, such as coordinates of sources, radial velocities of sources, setups of observing frequency.

4. Listing antenna information (PRTAN)

We can see the list of used antennas with the task PRTAN. We specify antennas by antenna numbers in AIPS, so remain this list in your memorandum.

```
> task 'prtan'  Using the task PRTAN.
> getn 2        Selecting the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
```

Type "> go". AIPS run PRTAN, and write information on the main window.

```
 kaimon    PRTAN(31DEC06)   3018     11-SEP-2006  17:44:22    Page    1
File=BK084       .MSORT .   1     An.ver=   1     Vol= 1     User= 3018
Array= VLBA          Freq= 15348.000000 MHz     Ref.date= 17-AUG-2001

Array reference position in meters (Earth centered)
Array BX=         0.00000     BY=          0.00000     BZ=          0.00000
Polar X =   0.19586 Polar Y =   0.17670 arcsec
```

```
Earth rotation rate =  360.9856449733 degrees / IAT day
GST at UT=0 =  325.4423303684 degrees
UT1-UTC=     -0.0225580   Data time(UTC     )-UTC=      0.0000000 seconds
Solutions not yet determined for a particular FREQID

Ant   1 = BR       BX= -2112065.0245 BY=  3705356.5077 BZ=  4726813.7400
Mount=ALAZ  Axis offset=  2.1319 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   2 = FD       BX= -1324009.1731 BY=  5332181.9811 BZ=  3231962.4404
Mount=ALAZ  Axis offset=  2.1353 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   3 = HN       BX=  1446375.0598 BY=  4447939.6583 BZ=  4322306.1211
Mount=ALAZ  Axis offset=  2.1313 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   4 = KP       BX= -1995678.6758 BY=  5037317.7123 BZ=  3357328.0835
Mount=ALAZ  Axis offset=  2.1368 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   5 = LA       BX= -1449752.4107 BY=  4975298.5896 BZ=  3709123.8879
Mount=ALAZ  Axis offset=  2.1367 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   6 = MK       BX= -5464075.0208 BY=  2495248.8389 BZ=  2148296.9785
Mount=ALAZ  Axis offset=  2.1370 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   7 = NL       BX=  -130872.3102 BY=  4762317.1195 BZ=  4226851.0199
Mount=ALAZ  Axis offset=  2.1353 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   8 = OV       BX= -2409150.1782 BY=  4478573.1991 BZ=  3838617.3558
Mount=ALAZ  Axis offset=  2.1336 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant   9 = PT       BX= -1640953.7650 BY=  5014816.0373 BZ=  3575411.8409
Mount=ALAZ  Axis offset=  2.1395 meters    IFA               IFB
Feed polarization type =                    R                 L

Ant  10 = SC       BX=  2607848.5696 BY=  5488069.6559 BZ=  1932739.5746
Mount=ALAZ  Axis offset=  2.1266 meters    IFA               IFB
Feed polarization type =                    R                 L
AIPS 1: Resumes
```

Numbers (1, 2, 3,…) and names (BR, FD, HN,…) of ten antennas are listed like above. BX, BY and BZ are the position coordinates of antennas.

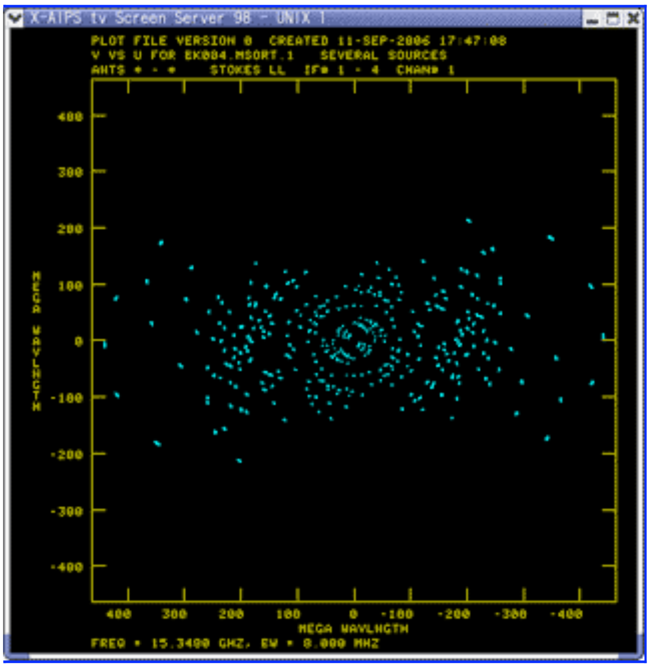
5. Seeing (u, v) coverage (UVPLT)

(u, v) is a spatial frequency of visibility. East-west component is "u". North-south component is "v". These are the lengths of baselines divided with the observed wavelength, so they are non-dimensional values. If you would like to know why these are called "spatial frequencies", please see here.

In interferometry observations, we measure visibilities of various spatial frequencies and make an image from them. As the range of the spatial frequencies is wider, spatial resolution becomes smaller. As the lack of spatial frequencies is smaller, the quality of images becomes better. The range of measured spatial frequencies is called "(u, v) coverage". (u, v) coverage shows the resolution and quality of the image.

```
> task 'uvplt'            Using the task UVPLT.
> getn 2                  Selecting the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> bparm=6,7,2,0           Horizontal axis is u.  Vertical axis is v.  Selecting auto scaling.
> echan=1                 Plotting frequency channel No. 1.  It takes much time
                          when we plot all frequency channels.
> dotv=1                  Plotting results on the TV window.
> inp                     Checking the list of parameters.
 (Here is the parameter list of UVPLT.)
```

Type "> go". AIPS shows (u,v) coverage on the TV window.



(u, v) plot : click to enlarge

6. Seeing spectrum (POSSM)

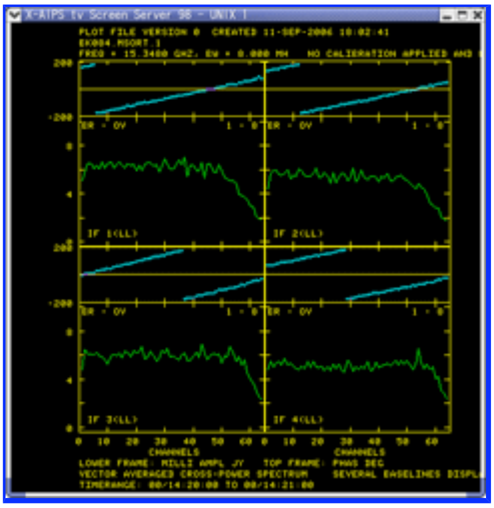
Visibilities are the functions of frequencies. Plots of amplitudes and phases of visibilities against frequencies are called cross power spectra. Cross power spectrum is displayed by the task POSSM.

In the case of the continuum source observation, cross power spectra should be flat against frequencies. However, the frequency characteristic is not flat, so the cross power spectra before calibrations are not flat. From the cross power spectra of continuum sources, we can roughly estimate the frequency characteristics of instruments.

```
> task 'possm'            Using the task POSSM.
> getn 2                  Selecting the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> timer 0 14 20 0 0 14 21 0
                          Time range is 1 minute from 0day 14h20m00s to 0day 14h21m00s.
                          When you specify TIMERANG, use the format above.
> echan=0
```

```
                          Setting ECHAN to zero in order to display all frequency channels.
> aparm 1, 1, 0.0, 0.01, -180, 180, 0
                          First parameter means displaying vector averaged spectra.
                          Second parameter means using fixed scale.
                          Third and Fourth parameters mean the range of amplitude is
                          from 0 to 0.01.
                          Fifth and Sixth parameters mean the range of phase is from
                          -180 deg to 180 deg.
> CODETYPE 'a&p'          Plotting both amplitudes and phases.
> nplot 4                 Plotting four graph panels a page.
> bparm 0
> inp                     Checking the list of parameters.
 (Here is the parameter list of POSSM.)
```

Type "> go". AIPS runs POSSM and shows cross power spectra on your TV window.



Cross power spectra with POSSM : click to enlarge

In order to move to the next page, please type b or c after clicking on your TV window. In order to finish POSSM, type d on your TV window.

Please see the spectra. Amplitudes are almost flat, but they decrease at the edges of the passband. They must be the characteristic of frequency filters. We can calibrate them at the "bandpass calibration". Phases have constants slope against frequencies. These are caused by the residuals of delays. After you calibrate these delay residuals by "fringe fitting", phases become almost flat. If you integrate visibilities for the frequency direction before this calibration, visibility amplitudes decrease systematically by the coherence loss. This will result in the wrong images. Fringe fitting is necessary to get the correct visibilities.

Previous Top Next

Lesson 3 : Calibrating amplitudes of visibilities

Previous Top Next

From here, we are going to start the calibration of data. The definition of calibration is to calculate the parameters of transfer functions in order to estimate the true input values from observed raw data (simply speaking, to calculate the difference between true input values and observed data). In general, observed values we can get are affected with the transfer functions. Transfer function is the relationship between inputs and outputs. Details are seen in this page (sorry, in Japanese!).

For example, visibilities measured with the interferometers are products of true visibilities and antenna complex gains. Formulation is as follows:

$$\hat{V}_{i,j}(\nu,t) = g_i(\nu,t)g_j^*(\nu,t)V_{i,j}(\nu,t)$$

where t is time, ν is frequency, i, j are antenna numbers, g is an antenna complex gain, V is a true visibility, $\hat{V}$ is an observed visibility.

In order to estimate correct visibilities, we need the calibration with antenna complex gains. Antenna complex gains are complex numbers as their name shows, and have amplitude terms and phase terms. AIPS carries out the amplitude calibration and the phase calibration separately. You can run whichever first. Here, we will run the amplitude calibration first. That is, we will estimate the amplitude terms of the antenna complex gains.

1. Amplitude term of antenna complex gain

Visibilities outputted from correlators are the Fourier transform of normalized correlation function. You can get the flux density (unit : Jy) by multiplying SEFD (System Equivalent Flux Density, unit : Jy). That is,

$$|g| = \sqrt{\mathrm{SEFD}}.$$

Here, we take square roots because the geometrical mean of two antenna gain is the synthesized SEFD of the baseline.

SEFD is shown with system noise temperature (T_{sys}) and antenna effective aperture area (A_e), as

$$\mathrm{SEFD} = \frac{2k_\mathrm{B}T_\mathrm{sys}}{A_\mathrm{e}},$$

where $k_B = 1.38\times10^3$ is Boltzmann constant (This value is multiplied by 10^{26} because 1 Jy = 10^{-26} W Hz^{-1} m^{-2}). The constant 2 is multiplied because we treat the situation receiving one component of polarimetry. After all, we need the effective aperture area A_e and the system noise temperature T_{sys} of each antenna in order to get the amplitude term of gain.

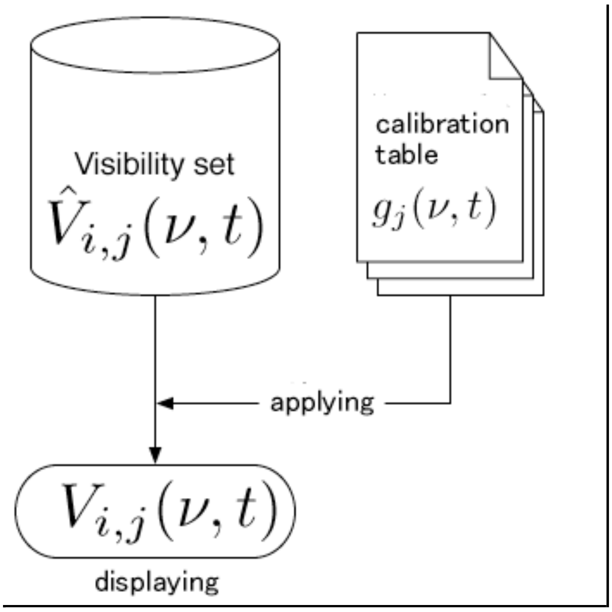
T_{sys} is the noise temperature including the Earth’s atmosphere, so it is a function of time. It varies with frequencies, but we can assume that it is almost constant in the narrow bandwidth of an IF, so we treat T_{sys} as a function of time only. T_{sys} data measured in the observation are recorded in the TY extension table. Effective aperture areas depend on elevations, rather than time. This is because the distortion by gravity force results in the decrease of the effective aperture area. In AIPS, we treat the effective aperture area as a polynomial function of elevations (EL). Factors of this polynomial function are recorded in the GC extension table. This table treats $A_e/2k_B$, not A_e.

2. Policy of calibration in AIPS

2.1. Data is holy

The process of AIPS calibration has a policy, "Data is holy". This means that the basic concept that of SIPA calibration is to save the original visibilities without modifying them. How do we make calibrated visibilities? The calibration in AIPS is to create calibration tables.
In the calibration tables, antenna complex gain $g^{-1/2}$ is recorded. By calculating the products of $g_i^{-1/2}$, $g_j^{-1/2}$ and $\hat{V}_{i,j}$, the calibrated visibilities $V_{i,j}$ are displayed (or outputted). Observed visibilities are left without any modifications.
The merits of this procedure are:

- Visibility data, which usually have large size of files, are not modified or copied. We can save calculations and data disks.
- We can make many calibration tables easily. We often failed to make calibration tables. When you think you failed by checking calibration tables, you can make a new calibration table. It is also possible to choose the versions of calibration tables by saving some calibration tables.
- Each calibration table have each step of calibration. For example, 1st version is normalization, 2nd version is amplitude calibration, 3rd version is calibration of the time variation of phases, 4th version is bandpass calibration, etc. We can easily upgrade calibration tables step-by-step.



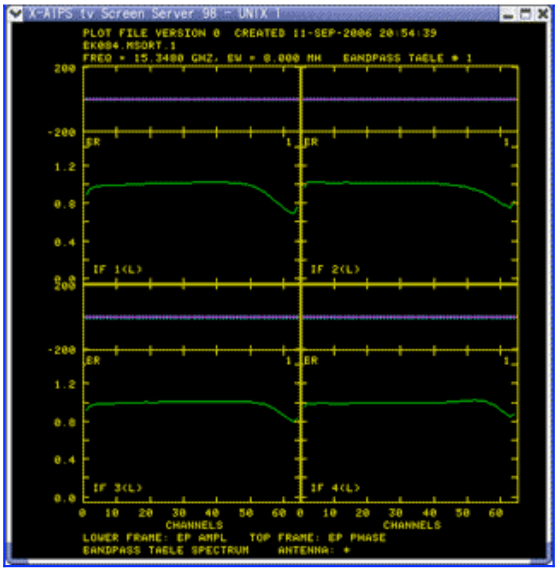
Treatment of visibility data and CL (calibration) table in AIPS

2.2. CL table and BP table

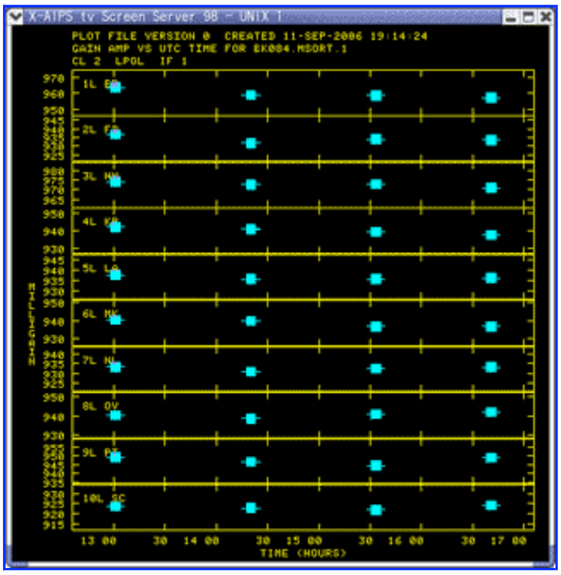
In AIPS, complex gains g(ν, t) called "calibration table" are divided into two parts: frequency-dependent term and time-dependent term.

$$g(\nu,t) = B(\nu)\cdot G(t)$$

B(ν) is the bandpass characteristic (frequency-dependent term) and is recorded in the BP extension table. G(t) is the time-dependent term and is recorded in the CL extension table. In AIPS, we can treat a time-variable BP table, but in most cases, the bandpass characteristics are almost constant. We treat them as time-independent values. The examples of BP tables and CL tables are shown below.



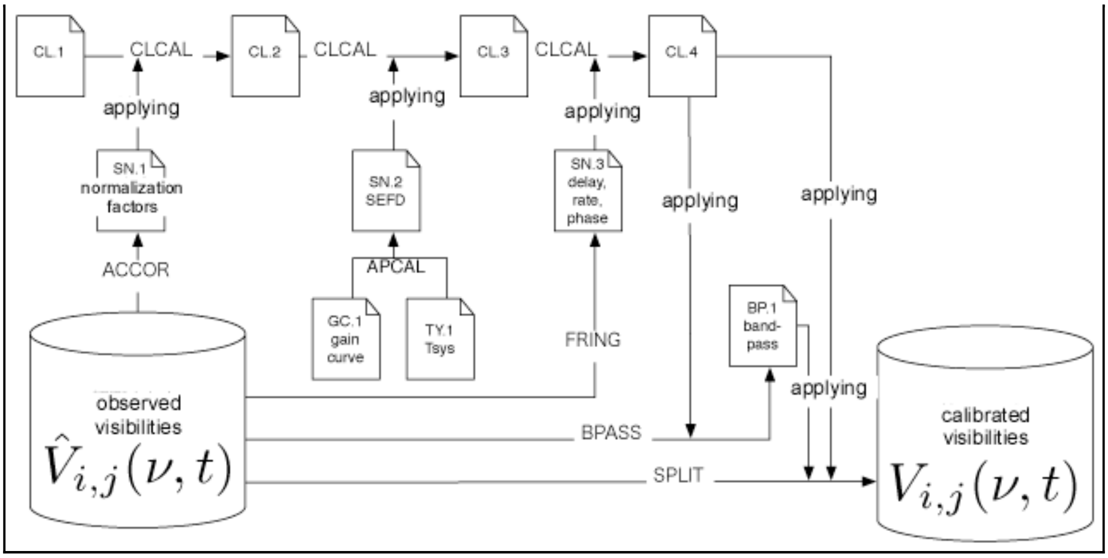
Example of BP tables : The horizontal axis is frequency channel number. The purple lines show the phases. The green lines show the amplitude. In this case each antenna has four IFs. Data of four IFs of one antenna are displayed. Amplitudes are decreased at the higher frequencies. This is caused by the characteristics of filters.



Example of CL tables : The horizontal axis is time, and the vertical axis is amplitude. Gains of IF=1 for ten antennas are plotted. The vertical value, about 940 milligain, means the value of amplitudes are about 0.94. In this stage, we have not applied the calibration of SEFD yet, and have applied the calibration of normalization with auto-correlation function only. This is why the vertical values are around 1.

3. Process of calibration

Let's see the outline of calibration process in this time. Look at the following figure.



Procedure of calibration. "CL.(version)" means the CL extension table. These versions are incremented from 1 to 4 with the steps of calibrations. "GC.(version)" is the GC (gain curve) extension table. It has values of $A_e/2k_B$ as functions of EL. "TY.(version)" means the TY (T_{sys}) extension table. It has the values of system noise temperatures as functions of time. "SN.(version)" is the SN (solution) table. It is made from observed visibilities or data in GC tables and TY tables. CL table is upgraded by applying an SN table to a CL table. “BP.(version)” means BP (bandpass) table. It has bandpass characteristics. ACCOR, CLCAL, APCAL, FRING, BPASS, SPLIT are the names of tasks used in this calibration procedure.

As is shown above, the procedure of calibrations is to upgrade CL tables. In the CL version 1, all values are 1, "doing nothing." Multiplying 1 makes no change. We start from the CL version 1, apply normalization factors, apply SEFDs, and so on. After we finish the calibration, we can get the calibrated visibilities by applying the final version of CL table.

SN tables are used to upgrade CL tables. They are created by solving some equations from visibilities, from GC tables and/or from TY tables.

In the CL tables, the calibration data is recorded in a constant interval, whereas not in the SN tables. In the task CLCAL, with which an SN table is applied to CL table, we need interpolations or smoothings.

4. Normalizing with auto-correlation data (ACCOR)

Observed visibilities are normalized in the correlation process, but we need auto-correlation data for the strict normalization. This process is done by the task ACCOR.

An observed visibilitiy $\hat{V}_{i,j}$ is obtained from the Fourier transform of cross-correlation function $C_{i,j}(\tau)$

$$C_{i,j}(\tau) \leftarrow \mathrm{FT} \rightarrow \hat{V}_{i,j}(\nu).$$

The normalization of cross-correlation function is to divide with the geometrical mean of the value of correlation function at $\tau = 0$:

$$\rho_{i,j}(\tau) = \frac{C_{i,j}(\tau)}{\sqrt{C_{i,i}(\tau=0) \cdot C_{j,j}(\tau=0)}}.$$

Therefore, we can normalize visibilities by recording the values of $C_{i,i}^{-1/2}(0)$ in calibration tables. ACCOR calculates the mean value of $C_{i,i}^{-1/2}(0)$ with the given time inteveral and outputs to an SN extension table. Let's run the task ACCOR.

```
> task 'accor'      Using the task ACCOR.
> getn 2             Selecting the input file whose catalog number is 2.
AIPS 1: Got(1)     disk= 1  user=3018   type=UV   BK084.MSORT.1
> timer 0            Normalizing for all time ranges.
> solin 1            Time interval is 1 min.
> inp                Checking the list of parameters.
 (Here is the parameter list of task ACCOR.)
```

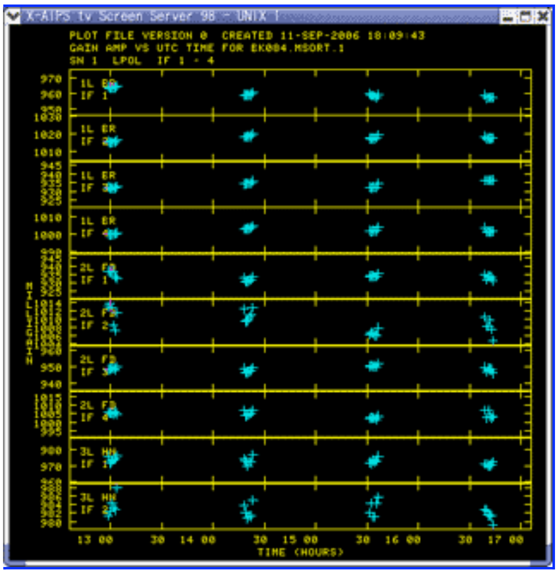
Type "> go". After AIPS finishes ACCOR, please check the version 1 of SN extension table with imheader.

4.1. Plotting and checking the new SN table (SNPLT)

We use the task SNPLT to check and display an SN extension table on your TV window.

```
> task 'snplt'      Using the task SNPLT.
> getn 2             Selecting the input file whose catalog number is 2.
AIPS 1: Got(1)     disk= 1  user=3018   type=UV   BK084.MSORT.1
> inext 'sn'         Displaying SN extension table (SNPLT can display CL tables or TY tables).
> inv 1              Displaying the version 1 of the SN extension tables.
> optyp 'amp'        Displaying amplitudes.
> nplot 10           Displaying ten panels in a page.
> inp                Checking the list of parameters.
 (Here is the parameter list of SNPLT.)
> tvinit             Initializing your TV window.
```

Type "> go. The following plot is displayed on your TV window. The horizontal axis is time. The vertical axis is amplitude. Note that the vertical axis is displayed in "milligain" unit. That is, 960 milligain = 0.96 and it is around 1.



4.2. Applying the SN table to the CL table (CLCAL)

We are going to apply the gains in the SN extension table to the CL extension table. We use the task CLCAL for this purpose.

In the CL table, gains are recorded in the interval of 0.1 minutes (This time interval is specified by the previous step, INDXR). On the other hand, the time interval between the solutions in the SN table is 1 minute. Thus we need to interpolate gains with CLCAL. We need to specify how to interpolate with the input parameters of CLCAL.

```
> task 'clcal'            Using the task CLCAL.
> getn 2                  Selecting the file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018    type=UV    BK084.MSORT.1
> source 'da193' ''       Applying to CL table data in the time duration observing DA 193.
> calsou 'da193' ''       Using SN table data in the time duration observing DA 193.
> freqid 1                The frequency ID is No. 1 (15.4 GHz).
> INTERPOL 'self'         Interpolating in each source.
> SAMPTYPE 'mwf'          Using median window filter for the interpolation method.
> BPARM 1                 The time window of the median window filter is 1 hour.
> SMOTYPE 'ampl'          Smoothing amplitudes.
> SNVER 1                 Using the version 1 of SN extension table.
> GAINVER 1               Applying to the version 1 of CL extension table.
> gainu 2                 Storing the results to the version 2 of the CL extension table.
> inp                     Checking the list of parameters.
 (Here is the parameter list of task CLCAL.)
```

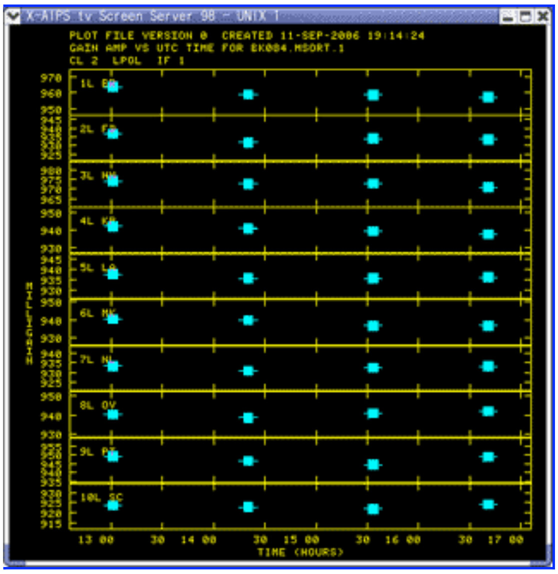
Type " > go". After CLCAL is finished, confirm the version 2 of the CL extension table with imheader.

4.3. Plotting and checking the new CL table (SNPLT)

We can plot and check CL tables with the task SNPLT, like SN tables.

```
> tget snpl        Using the task SNPLT.  Using the verb TGET, the parameters are set to that of p
> inext 'cl'       Plotting CL extension table.
> inv 2            Plotting version 2 of CL extension table.
> optyp 'amp'      Plotting amplitudes.
> inp              Checking the list of parameters.
 (Here is the parameter list of task SNPLT in this step.)
> tvinit           Initializing your TV window.
```

Type "> go". The following figure is displayed.



5. Calibrating with SEFDs (VLBAMCAL, SETJY, APCAL)

Normalized visibilities are calibrated to Jy-unit visibilities by multiplying the geometrical mean of SEFDs. SEFDs are estimated from antenna gains (effective aperture areas) and system noise temperatures (T_{sys}). Antenna gains are recorded in the GC extension table. T_{sys} is recorded in the TY extension table. We must take three steps to create an SN table calculating SEFD from these information: a procedure, VLBAMCAL, and two tasks, SETJY and APCAL.

5.1. Preparing TY and GC with VLBAMCAL

VLBAMCAL is a procedure. A procedure is a kind of script and executes tasks and verbs in the order of the script. VLBAMCAL is a procedure to prepare the GC extension table and TY extension table.

```
> run vlbautil     Preparing to run the procedure VLBAMCAL.
> getn 2            Selecting the file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018   type=UV    BK084.MSORT.1
> inp vlbamcal     Checking the list of parameters of the procedure VLBAMCAL.
AIPS 1: VLBAMCAL: Procedure to merge redundant calibration data
AIPS 1: Adverbs       Values                  Comments
AIPS 1: ----------------------------------------------------------------
AIPS 1: INNAME       'BK084'                  Input file name
AIPS 1: INCLASS      'MSORT'                  Input file class
AIPS 1: INSEQ           1                     Input file sequence number
AIPS 1: INDISK          1                     Disk number for input file
AIPS 1: BADDISK      *all 0                   List of disks not to be used
AIPS 1:                                       for scratch files.
AIPS 1:
AIPS 1:              VLBAMCAL is defined in the VLBAUTIL run file.
> vlbamcal         Running VLBAMCAL.
```

You can see a lot of messages. They represent what is carried out, but you do not need to care them. After AIPS finishes VLBAMCAL, please confirm the version 1 of GC table and TY table with imheader.

5.2. Setting the flux density of sources when we measure the system noise temperatures

System noise temperatures (T_{sys}) are recorded in the TY extension table. In the case of VLBA, they measure T_{sys} when they observing sources, so the measured temperature is $T_{sys} + T_a$, not T_{sys}. Here, T_a is the antenna temperature of the source. When we observe a source whose flux density is S, the antenna temperature is $T_a = A_e S / 2k_B$. When we do not observe very strong sources, we can ignore T_a because $T_{sys} \gg T_a$, but for strict calibrations, we need to calibrate T_a by sending AIPS the flux densities of sources. The flux densities of sources are recorded in the SU extension table in AIPS. We use the task SETJY to set the flux densities.

In this observation, we observe a source whose name is DA193, so we set the flux density of DA193 with SETJY. What is the flux density of DA193? The monitoring observations by NRAO (National Radio Astronomical Observatory) and those by UMRAO (University of Michigun Radio Astronomy Observatory) are useful. Here, we use the data by UMRAO. The flux density of DA193 at 15 GHz is 4.8 Jy.

```
> task 'setjy'           Using the task SETJY.
> getn 2                  Selecting a file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018   type=UV   BK084.MSORT.1
> source 'da193' ''      Source name is DA193.
> zerosp 4.8, 0          Flux density is 4.8 Jy.
> freqid 1               Frequency ID is No. 1 (15.4 GHz).
> optyp ''               Clearing the parameter of OPTYPE.
> inp                    Checking the list of parameters.
 (Here is the parameter list of task SETJY.)
```

Type "> go". AIPS displays these messages and finishes SETJY.

5.3. Writing SEFDs to the SN extension table with APCAL

TY, GC and SU tables are prepared. We can calculate SEFDs. We calculate SEFDs with the task APCAL and write them to the SN extension table.

```
> task 'apcal'           Using the task APCAL.
> getn 2                  Selecting a file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018   type=UV   BK084.MSORT.1
> source 'da193'''       Source name is DA193.
> freqid 1               Frequency ID is No. 1 (15.4 GHz).
> opcode 'LESQ'          Fitting with the least square method.
> dotv 1                 Showing the results on your TV window.
> inv 1                  Using the version 1 of WX extension table (weather data).
> tyv 1                  Using the version 1 of TY extension table (Tsys data).
> gcv 1                  Using the version 1 of GC extension table (antenna gain data).
> snv 2                  Writing the results to the version 2 of SN extension table.
> dofit 1, 0             Calibrating the optical depths of the Earth's atmosphere.
> inp                    Checking the list of parameters.
 (Here is the parameter list of APCAL.)
```

Type "> go". AIPS runs APCAL. After AIPS finishes APCAL, please confirm the version 2 of SN extension table with imheader.

If APCAL does not run correctly : Use ANTAB

When you are using AIPS version older than 31DEC06 or the date of correlation is long before, we can not use GC table created by VLBAMCAL in APCAL. In such cases, we make a GC table from text file with the task ANTAB before we run APCAL.

First, download the text file of antenna gain information (BK084.GAIN). Save it in $AIPS_ROOT/FITS. If your $AIPS_ROOT is /usr/local/aips, you save it as /usr/local/aips/FITS/BK084.GAIN.

http://astro.sci.kagoshima-u.ac.jp/omodaka-nishio/member/kameno/AIPS/BK084.GAIN : 739-byte ASCII file

Second, we create the GC table from the text file above with the task ANTAB.

```
> task 'antab'                    Using the task ANTAB.
> getn 2                           Selecting a file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018   type=UV   BK084.MSORT.1
> infile 'FITS:BK084.GAIN'        Specifying loading text file.
> gcv 2                            Writing to the version 2 of the GC extension table.
> tyv 2                            Writing to the version 2 of the TY extension table.
                                    The text file we load does not contain the Tsys information,
                                   so this is a dummy parameter.
> inp                              Checking the list of parameters.
```

Type "> go". AIPS runs ANTAB. After AIPS finishes ANTAB, confirm the version 2 of the GC extension table with imheader.

We finished the preparation for APCAL. Let's set APCAL.

```
> tget apcal    Using tget to use the parameters you run APCAL last time.
> gcv 2         Using the version 2 of GC extension table you created with ANTAB.
```

Type "> go". AIPS runs APCAL. After AIPS finishes APCAL, confirm the version 2 of SN extension table with imheader.

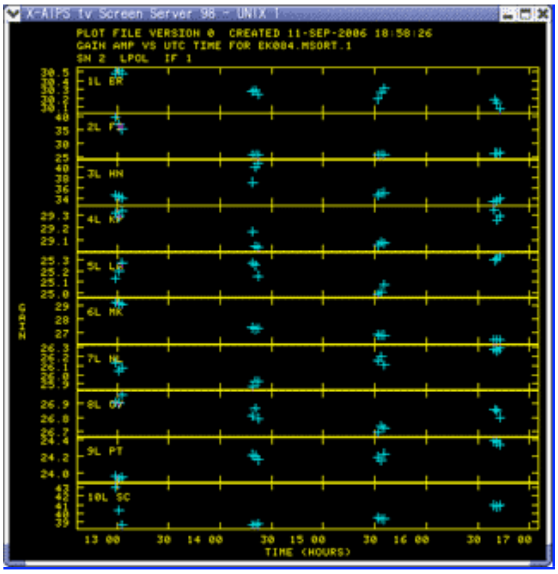
Recovery with ANTAB is finished. Let's proceed to the next step.

5.4. Plotting the SN table created by APCAL (SNPLT)

Let's plot and confirm the SN version 2 with SNPLT.

```
> task 'snplt'  Using the task SNPLT.
> getn 2        Selecting the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018    type=UV    BK084.MSORT.1
> inext 'sn'    Plotting SN extension tables (SNPLT can plot SN, CL or TY).
> inv 2         Plotting the version 2 of the SN extension table.
> optyp 'amp'   Plotting amplitudes.
> bif 1;eif 1   Plotting the data of IF 1.
> inp           Checking the list of parameters.
 (Here is the list of parameters of the task SNPLT in this step.)
> tvinit        Initializing your TV window.
```

Type "> go". AIPS displays the following plot on your TV window. The values of the vertical axis are the square roots of SEFDs.



5.5. Applying SN table created by APCAL to CL table (CLCAL)

We apply (the square root of) the SEFDs recorded in the SN extension table to the CL extension table with CLCAL.

```
> task 'clcal'  Using the task CLCAL.
> snver 2       Using the version 2 of SN extension table.
> gainver 2     The version of the modifying CL table is 2.
> gainu 3       The version of the CL table recording results is 3.
> inp           Checking the list of parameters.
 (Here is the list of parameters of the task CLCAL.)
```

Type "> go". AIPS runs CLCAL. After AIPS finishes CLCAL, confirm the version 3 of the CL extension table with imheader. Plot this CL table 3 with SNPLT, and confirm it.

Amplitude calibration is finished. The next step is phase calibration.

Previous Top Next

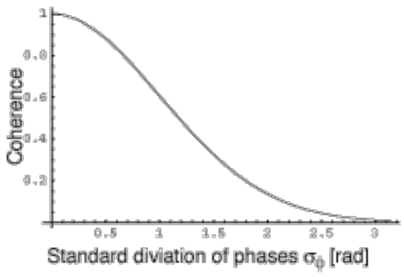
Lesson 4 : Fringe fitting (calibration of delay)

Previous Top Next

There are two purposes for the calibration of phases: (1) The visibility phases reflect the positions of radio sources, so we need calibrated visibilities to estimate the correct positions and structures of them. (2) The visibility phases must be almost constant to integrate visibilities coherently. Let's think about (2). We need the integration of visibilities to improve the signal-to-noise (S/N) ratio. The operation to make an image with Fourier transform is also integration (Fourier transform is also called Fourier integration. It is an integration you can see in the equations.)
If we integrate time-variation visibilities, visibility amplitudes decrease because of the coherence loss.
When the visibility phases are shifted from the true value by (φ), effective visibility value is the projection of the true visibility, that is, V cos φ. When the phases φ have a distribution with the standard deviation σ, the visibility amplitude is given by

$$\langle V\rangle = \int_{\phi=-\infty}^{\infty} V\cos\phi\, p(\phi)d\phi = \int_{\phi=-\infty}^{\infty} V\cos\phi\, \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\phi^2}{2\sigma^2}}d\phi = Ve^{-\frac{\sigma^2}{2}}$$

Where the probability density distribution of phases p(φ) is Gaussian distribution. For example, when phases distributes with the standard deviation σ = 1 rad, amplitude decrease about 40%. We need to calibrate phases correctly and to integrate visibilities in the constant-phase state.



The relationship between the standard deviation of phases and coherences. At σ = 1 rad, the coherence is about 0.6, so the loss is about 40%.

1. Fringe fitting (FRING)

1.1. Why we need fringe fitting?

Fringe fitting is an operation to flatten visibility phases by calibrating delay residuals and time derivatives of delay residuals. After this, we can integrate visibilities coherently. AIPS records the values of delay residuals and time derivatives of delay residuals to the SN extension table with the task FRING, and applies them to CL extension table with CLCAL.
Delay residuals are the delays remaining after the correlation process. The delay tracking at the correlator is not perfect because of the clock offset at antennas, error of station position and error of the Earth's atmosphere. When Δτ is the delay residual, the phase shift is a function of frequency ν,

$$\phi = 2\pi\nu\Delta\tau$$

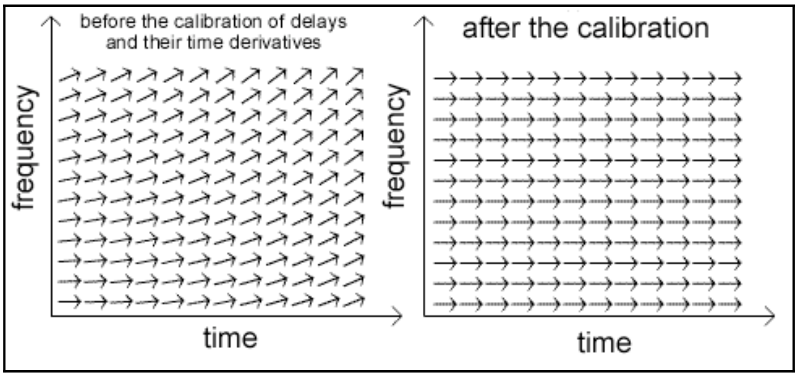
When the delay residuals varies with time, we can approximate them with

$$\Delta\tau = \Delta\tau_0 + \Delta\dot{\tau}(t-t_0)$$

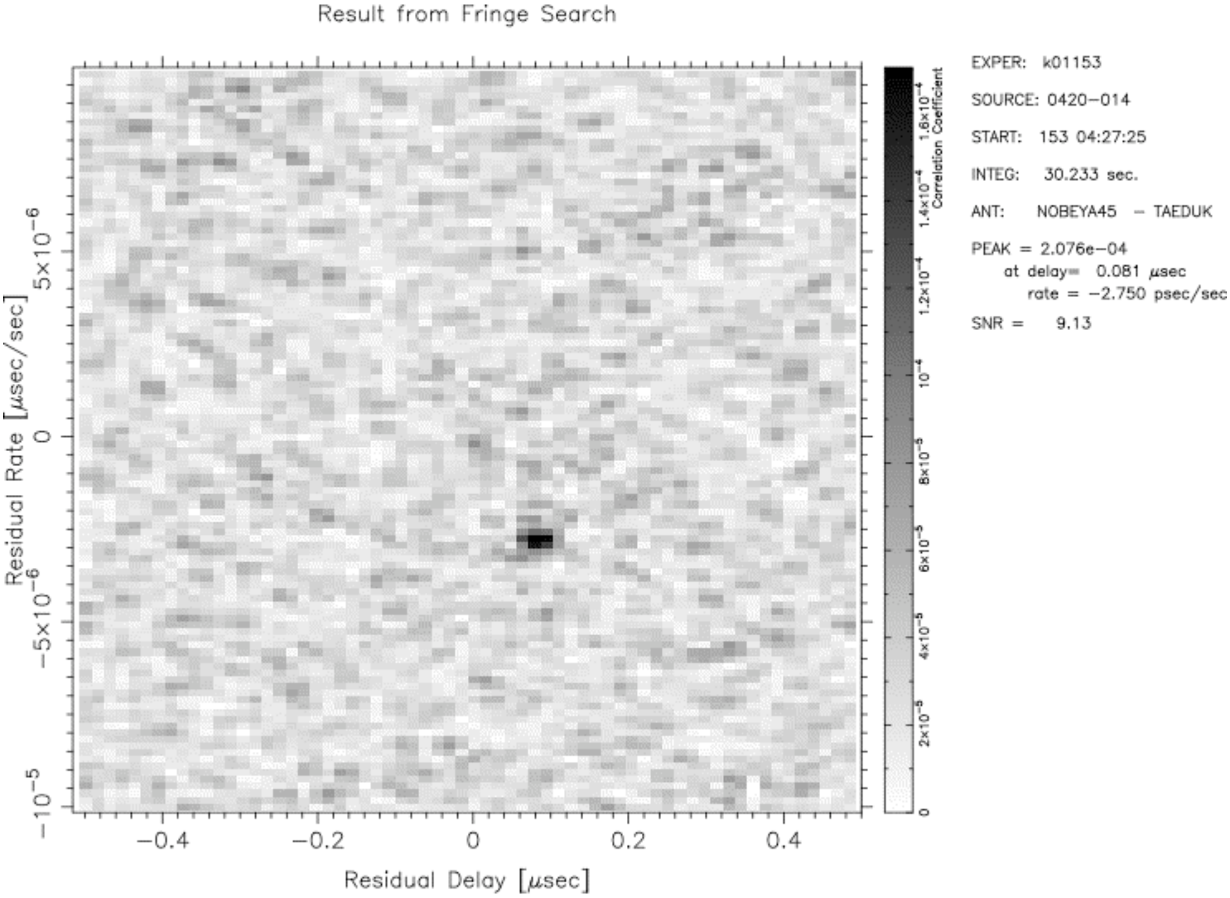
where $\Delta\dot{\tau}$ is the time derivative of delay residuals. The suffix 0 means the value at the time t_0. Thus, the phase becomes

$$\phi = 2\pi\nu\left[\Delta\tau_0 + \Delta\dot{\tau}(t-t_0)\right]$$

When we represent visibilities by vectors, phases by the directions of arrows, this situation is the below-left panel.



As the above-left panel, if visibility phases are not constant in the (time-frequency) domain by delay residuals and time derivatives of delay residuals, coherence loss occurs in the integration. Visibility amplitudes decrease. How calibrate delay residuals or time derivatives of delay residuals? For this calibration, we integrate visibilities in $(\Delta\tau_0, \Delta\dot{\tau})$ domain and search $(\Delta\tau_0, \Delta\dot{\tau})$ where maximize amplitudes. This operation is called fringe search.



The above figure is an example of fringe search. The horizontal axis is the delay residual (Residual Delay). The vertical axis is the time derivative of delay residual (Residual Rate). Gray scale shows visibility amplitudes. Visibility amplitude is maximized at the delay residual of 0.081 μsec and time derivative of delay residual of -2.75×10^{-12}. If this maximum amplitude is larger than the noise level significantly (If the signal-to-noise ratio is large), fringe is detected.

The searching range of $(\Delta\tau_0, \Delta\dot{\tau})$ is called "fringe search window" or simply "search window". The range of above figure is the "window".

1.2. Running FRING

Let's run the task FRING and carry out the fringe fitting.

```
> task 'fring'              Using the task FRING.
> getn 2                    Choosing the file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1   user=3018    type=UV    BK084.MSORT.1
> calsou 'da193' ''         Searching scans observing DA193.
```

```
> freqid 1                 Setting frequency ID to No. 1 (15,4 GHz).
> timer 0                  Searching all time range.
> docal -1                 Not applying the calibration of CL table.
> doban -1                 Not applying the calibration of BP table.
> outn ''                  Not creating a new file for outputs (writing to the extension table of t
                           current file).
> outcl ''                 Not creating a new file for outputs.
> outs 0                   Not creating a new file for outputs.
> outd 0                   Not creating a new file for outputs.
> refan 9                  Choosing antenna No. 9 (PT) for the reference antenna.
> solin 2                  Calculating delay residuals and their time derivatives with the two minu
                           integral.
> inv -1                   Not using the structure model (CC extension table) of the source (assumi
                           point source).
> flagv 1                  Using the version 1 of FG table for the flagging information.
> search 0                 Not specifying the order of baseline in the search.
> dofit 0                  Estimating relay residuals and time derivatives of delay residuals for a
                           antennas.
> aparm 2, 0, 0, 0, 0, 2, 5, 0, 1, 0
                           Fifth parameter means searching for each IF.
                           Sixth parameter means displaying the process.
                           Seventh parameter means the SNR threshold of the detection limit.
> dparm 0, 400, 100, 0
                           Second parameter means the range of the delay
                           residual of search window.
                           Third parameter means the range of the time derivative of the delay
                           residual.
> snv 3
                           Version of the SN table writing relay residuals and their time derivativ
                           of the search results.
> bchan 2                  Range of integrating frequency channels.
> echan 63                 Range of integrating frequency channels (abandoning both sides).
> inp                      Checking the list of parameters.
 (Here is the parameter list of FRING.)
```

Type "> go". This task requires a little long time. AIPS displays running messages on your message window, so check the process. The values after "R=" are time derivatives of delay residuals (displaying the products of $\Delta\dot{\tau}$ and observation frequency ν, milihertz unit). The values after "D=" are delay residuals (nano-second unit). The values after "SNR=" are signal-to-noise ratios. When this value exceeds the value you specified in APARM(7) (seventh parameter of APARM), a fringe is detected and AIPS writes the values of $(\Delta\tau_0, \Delta\dot{\tau})$ to the SN extension table. When the value of SNR does not exceed APARM(7), AIPS considers that a fringe is not detected significantly and abandons the values of $(\Delta\tau_0, \Delta\dot{\tau})$.

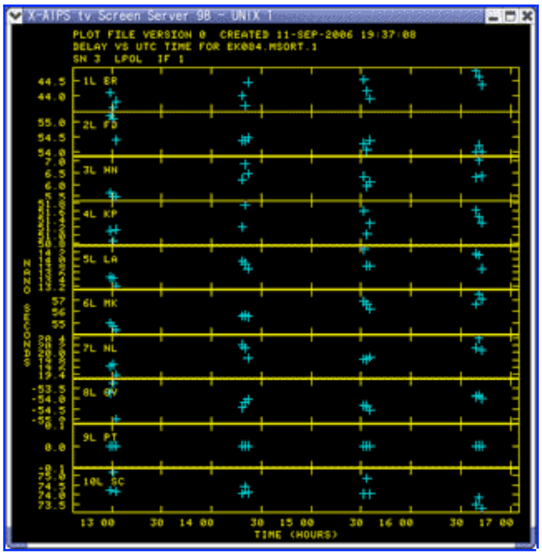
After AIPS finishes FRING, confirm the version 3 of SN extension table with imheader.

1.3. Plotting and checking the new SN table (SNPLT)

Let's plot and check the new SN extension table with SNPLT.

```
> task 'snplt'             Using the task SNPLT.
> getn 2                   Choosing the file whose catalog number is 2.
AIPS 1: Got(1)    disk= 1  user=3018    type=UV     BK084.MSORT.1
> inext 'sn'               Plotting the SN extension table (SNPLT can plot CL or TY).
> inv 3                    Plotting the version 3 of the SN extension table.
> optyp 'dela'             Plotting delay residuals.
> nplot 10                 Plotting 10 (the number of antennas) data a page.
> inp             Checking the list of parameters.
 (Here is the parameter list of task SNPLT in this step.)
> tvinit                   Initializing your TV window.
```

Type "> go". AIPS displays a plot like the following figure on your TV window. The horizontal axis is time. The vertical axes are the delay residuals. Setting optyp 'rate', and type "go". The vertical axes become the time derivative of delay residuals. In the case of VLBA, delay residuals are normally ±100 nsec, time derivatives of delay residuals are normally ±50 mHz.



1.4. Applying SN table to CL table (CLCAL)

We apply $(\Delta\tau_0, \Delta\dot{\tau})$ in the SN extension table to the CL extension table with CLCAL.

```
> task 'clcal'                 Using the task CLCAL.
> getn 2                       Choosing the file whose catalog number is 2.
AIPS 1: Got(1)     disk= 1   user=3018    type=UV    BK084.MSORT.1
> BPARM 0.1                    Setting the time window of median window filter to 1 hour.
> SMOTYPE 'full'               Smoothing phases, delays and time derivative of delays.
> SNVER 3                      Using the version 3 of the SN extension table.
> GAINVER 3                    Setting the version of modifying CL extension table to 3.
> gainu 4                      Setting the version of the CL table recording results to 4.
> refan 9                      Selecting antenna No. 9 (PT) as the reference antenna.
> inp                          Checking the list of parameters.
 (Here is the list of parameters of the task CLCAL at this stage.)
```

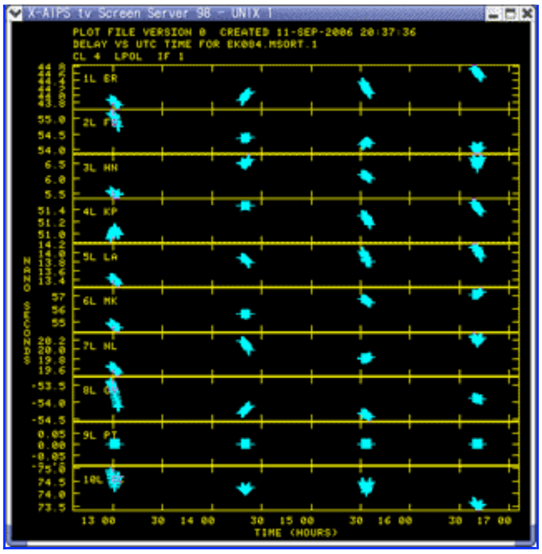
Type "> go". CLCAL runs. After AIPS finishes CLCAL, confirm the version 4 of the CL extension table with imheader.

1.5. Confirming the new CL table (SNPLT)

We can plot and check CL tables with the task SNPLT, like SN tables.

```
> tget snpl                    Using the task SNPLT.  Using the verb TGET, the parameters are set to tl
> inext 'cl'                   Plotting CL extension table.
> inv 4                Plotting the version 4 of the CL extension table.
> optyp 'dela'                 Plotting the calibrating value of delay residuals.
> inp                  Checking the list of parameters.
 (Here is the list of parameters of the task SNPLT at this stage.)
> tvinit                       Initializing your TV window.
```

Type "> go". The following plot appears on your TV window.



Fringe fitting has finished! Let's go to the bandpass calibration.

Previous Top Next

Lesson 5 : Bandpass calibration (calibration of frequency characteristics)

Previouw Top Next

We have finished the calibration of the time variability with FRING. Next, we carry out the bandpass calibration. We use the task BPASS for this calibration.
Bandpass characteristics B(ν) is complex values which have both amplitudes and phases. When we normalize the visibilities of observations of the continuum sources whose spectrum is flat, their visibilities must be the bandpass characteristics B(ν). It is possible and better to use the auto-correlations for the amplitude terms. This is because SNR of the auto-correlation is higher than that of the cross correlation, and because there is no effect of coherence loss in the auto-correlations. The phases of auto-correlations are always zero, so we can not use auto-correlations for phases. We must take two steps of BPASS for amplitudes with auto-correlations, for phases with cross correlations. We can assume that B(ν) is constant in the observing time if we do not have a terrible trouble.

1. Estimating amplitude terms of bandpass characteristics with auto-correlations

First, we carry out BPASS with auto-correlations. AIPS record the amplitude term of bandpass calibration into the BP extension table.

```
> task 'bpass'           Using the task BPASS
> getn 2                 Choosing the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV    BK084.MSORT.1
> docal 1                Calibrating with a CL table.
> gainu 4                Using the version 4 of the CL table.
> timer 0 15 34 0 0 15 36 0
                         Specifying the time range used for calibration (from 15h34m UT to
                         15h36m UT.  2 minutes).
> solin -1               Valid time range for the estimated passband characteristics. (-1 means all
                         time)
> outv 1                 Writing the output version of the BP extension table.
> bif 1; eif 0           Calibrating for all IFs.
> refan 9                Selecting antenna No. 9 (PT) for the reference antenna.
> soltyp 'L1R'           How to calculate norms.
> smooth 1, 5, 7         Using the sinc function whose range is seven frequency channels and
                         whose width of first nulls is five channels as the smoothing function.
> bpassprm 1, 0          Using auto-correlation.
> inp                    Checking the list of parameters.
 (Here is the parameter list of BPASS for amplitudes.)
```

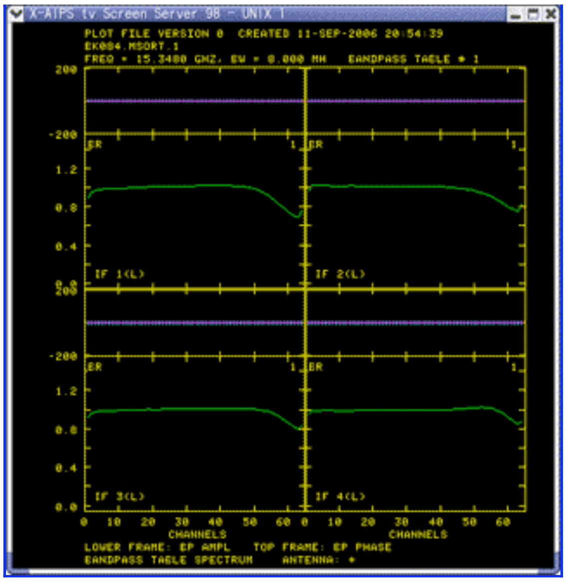
Type "> go". AIPS runs BPASS. After AIPS finishes BPASS, confirm the version 1 of the BP extension table with imheader.

2. Plotting and checking the amplitude terms of BP

The BP extension table is plotted with the task POSSM.

```
> tget possm             Using the task POSSM.  Using the verb TGET, the parameters are set to the
                         previous running.
> getn 2                 Choosing the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV    BK084.MSORT.1
> doban 1                Plotting a BP table.
> bpv 1                  Plotting the version 1 of the BP table.
> aparm 1, 1, 0, 1.5, -180, 180, 0, 2, 0
                         Eighth parameter means plotting the BP table itself.
> timer 0                Plotting all time range.
> inp                    Checking the list of parameters.
 (Here is the parameter list of POSSM for amplitudes.)
```

Type "> go". AIPS runs POSSM. The following figure is shown in your TV window. Amplitude terms of the bandpass characteristics are reduced at the both ends of IFs. This is mainly affected by the filters. Phase terms are zero because we are using auto-correlation functions. Phase terms are estimated in the next step.



3. Estimating phase terms of bandpass characteristics

We use the visibility phases of a sufficiently strong continuum source after integrating them within the coherence time. Here, we estimate the phase terms of BP from the two-minute integration (15h34m-15h36m) of visibilities of observing source, DA193 itself.

```
> tget bpass           Using the task BPASS.  Using the verb TGET, the parameters are set to the
                       previous running.
> bpassprm 0           Using cross correlations.
> bpassprm(4) 1        Writing phase terms only without changing amplitude terms.
> inp                  Checking the list of parameters.
 (Here is the parameter list of BPASS for phases.)
```

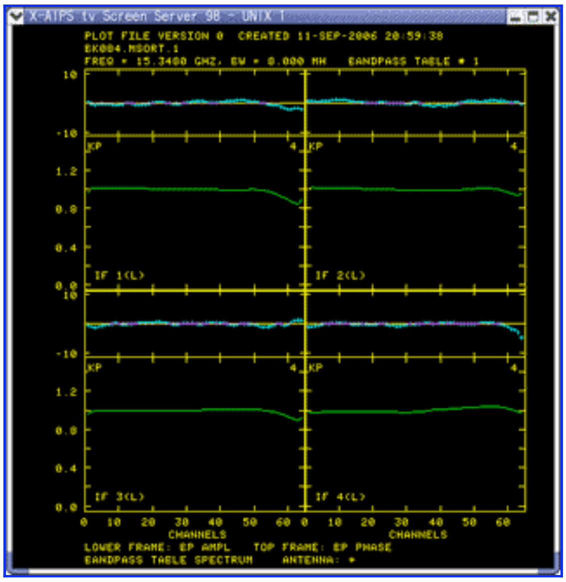
Type "> go". AIPS runs BPASS. This task overwrites the version 1 of the BP extension table, so the result of IMHEADER does not change.

4. Plotting and checking the phase terms of BP

Let's plot and check the BP extension table to which phase terms are appended.

```
> tget possm           Using the task POSSM.  Using the verb TGET, the parameters are set to the
                       previous running.
> inp                  Checking the list of parameters.
 (Here is the parameter list of POSSM for phases.)
```

Type "> go". AIPS runs POSSM. The following figure is plotted on your TV window. Amplitude terms are not changed. Phase terms are appended. Phases are not zero although they are very small.



5. Checking the calibrated visibilities

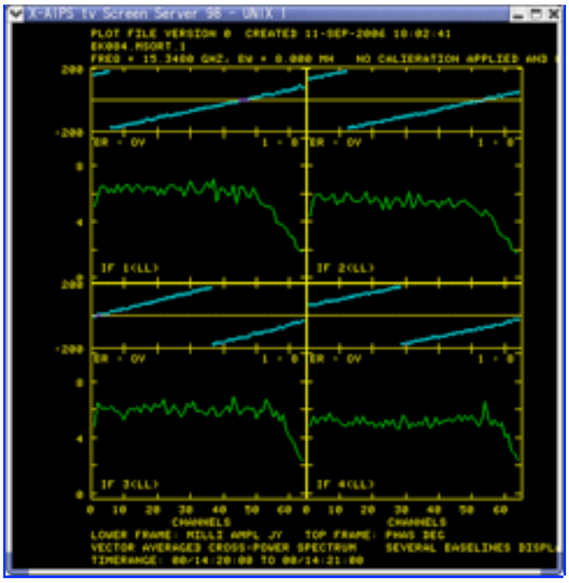
Calibrations of visibilities are finished. Let's check whether they are correct.

5.1. Plotting and checking the spectrum (POSSM)

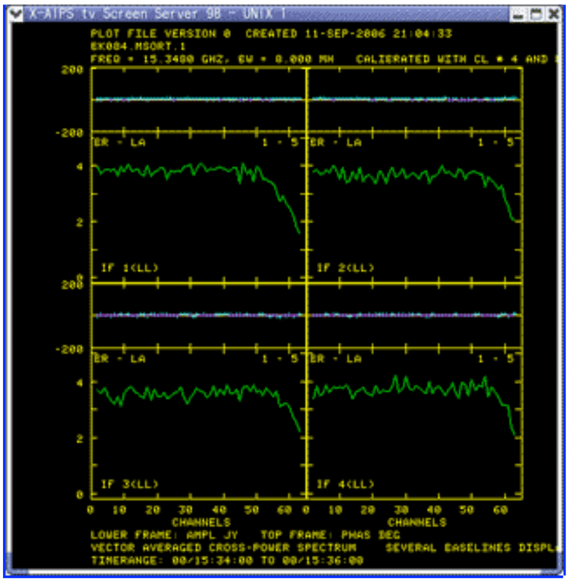
We check the calibrated spectrum applying CL and BP. We use POSSM.

> tget possm	Using the task POSSM. Using the verb TGET, the parameters are set to the previous running.
> aparm (4) 5.0	Maximum amplitude is 5 Jy.
> aparm(8) 0	Plotting visibilities.
> timer 0 15 34 0 0 15 36 0	Time range of plotting visibilities.
> docal 1	Applying the CL table.
> gainu 4	Using version 4 of the CL table.
> inp	Checking the list of parameters.
(Here is the parameter list of POSSM at this stage.)	

Type "> go". AIPS runs POSSM. The below right figure, the plot of calibrated visibilities is plotted. The below left figure is the visibilities before the calibration to compare.



Spectra before the calibrations



Spectra after the calibrations

Although phases have a slope before the calibrations, they become flat after the calibrations.

This is because we carry out the calibration of delay residuals and bandpass characteristics. The values of amplitudes are also changes. The phase fluctuation in the passband is almost flat. Amplitudes are decreased at the higher edge of IFs. This is because the BP table from the auto-correlation contains the folding noise. This is a limit using auto-correlation for the amplitude term of BP. We will set not to use this are at the stage of frequency integration afterward. Calibration is finished here. Next we reduce the data size by integrating visibilities for the frequency direction.

Previous Top Next

Lesson 6 : Reducing data size by frequency integration

Previous Top Next

Finally, we create the final calibration table. Let's output the calibrated visibilities by applying them. In this stage, we integrate for the frequency direction and reduce the frequency channel from 64 an IF to 1 an IF. This operation reduces data size significantly.

1. Applying the calibration table and integrating for the frequency direction (SPLIT)

The task name SPLIT is named after creating single-source file by splitting multi-source format file to each source. In this process, the calibration table is applied. The visibilities are integrated for the frequency direction. Calibrated visibilities are output.

```
> task 'split'          Using the task SPLIT.
> getn 2                Choosing the file whose catalog number is 2.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   BK084.MSORT.1
> timer 0               Using all time range.
> source ''             Using all sources.
> freqid 1              Using frequency No.1 (15.4 GHz).
> docal 1               Applying CL tables.
> gainu 4               Applying the version 4 of the CL table.
> doban 1               Calibrating frequency characteristics with the BP table.
> bpv 1                 Using the version 1 of the BP table.
> flagv 1               Using the version 1 of the FG table.
> outd 1                The output file is written to Disk No. 1.
> outcl 'sp2cm'         Class of the output file name is SP2CM.
> aparm 2, 0            First parameter means to integrate all frequency points in one IF to 1 p
> inp                   Checking the list of parameters.
 (Here is the parameter list of SPLIT.)
```

Type "> go". AIPS runs SPLIT. Messages are plotted like this. If the number of visibilities in "Fully flagged by gain" is very large and that in "Fully kept", check them. Something is wrong. After AIPS finishes SPLIT, confirm the new file DA193.SP2CM.1 is created in the catalog No. 3. We do not specify OUTNAME, but it is changed to the source name with SPLIT.

```
>  pcat
AIPS 1: Catalog on disk  1
AIPS 1:  Cat Usid Mapname      Class   Seq  Pt     Last access      Stat
AIPS 1:    1 3018 BK084       .UVDATA.    1 UV 11-SEP-2006 17:34:40
AIPS 1:    2 3018 BK084       .MSORT .    1 UV 11-SEP-2006 21:08:11
AIPS 1:    3 3018 DA193       .SP2CM .    1 UV 11-SEP-2006 21:08:11
```

The procedure in AIPS will finish soon. The final step is to write to FITS file.

Previouw Top Next

Lesson 7 : Creating FITS file in order to image with difmap

Previous Top Next

We output the calibrated file as a FITS file in order to make an image with difmap.

1. Outputting to a FITS file (FITTP)

FITTP is named after writing FITS files to magnetic tapes. But we usually output them to hard disks these days.

```
> task 'fittp'          Using the task FITTP.
> getn 3                Choosing the file whose catalog number is 3.
AIPS 1: Got(1)   disk= 1  user=3018   type=UV   DA193.SP2CM.1
> outf 'fits:BK084.DA193.SP2CM.fits'
                        Outputting file name ($FITS/BK084.DA193.SP2CM.FITS).
> outt 0                Writing to hard disks, not to magnetic tapes.
> inp           Checking the list of parameters.
 (Here is the parameter list of FITTP.)
```

Type "> go". AIPS runs FITTP. After AIPS finishes FITTP, a new file whose name is $AIPS_ROOT/FITS/BK084.DA193.SP2CM.FITS on the UNIX file system. We will use this file in difmap.
Calibrations with AIPS are finished. From the next step, we will make an image with difmap.

Previous Top Next

Lesson 8 : Loading data into difmap

Previous Top Next

Those who skip the procedure with AIPS

Calibrated data is stored in your computer. If you have some trouble, please download it from the following:

http://astro.sci.kagoshima-u.ac.jp/omodaka-nishio/member/kameno/AIPS/BK084.DA193.SP2CM.FITS
: 1.99 MB FITS file

0. Before you start difmap

Please make a setting for the text editor.

If you use vi, please do nothing.

If you use emacs, please type EDITOR=emacs; export EDITOR on your terminal.

If you use gedit, please type EDITOR=gedit; export EDITOR on your terminal.

If you use the other text editors, please set the environment variable $EDITOR as above examples.

1. Starting difmap

If difmap is installed in your computer, its program is usually stored in /usr/local/uvf_difmap/difmap. So if you make a path to /usr/local/uvf_difmap/, you can start difmap after you find an appropriate working area. We do not need any special file systems or directory structures.

Here, we make a working directory BK084 under your home directory and move the FITS file you wrote with AIPS in the previous lesson to BK084 to analyze with difmap.

```
$ cd ~            Tilde (~) means your home directory.
$ mkdir BK084
$ cd BK084
$ cp /usr/local/aips/FITS/BK084.DA193.SP2CM.FITS .
                   (Do not forget the last period!!!)
$ difmap          Starting difmap.
```

2. Loading FITS file

After you start difmap, difmap waits commands. First, we load the FITS file with observe command.

```
observe BK084.DA193.SP2CM.FITS
! Reading UV FITS file: BK084.DA193.SP2CM.FITS
! AN table 1: 596 integrations on 45 of 45 possible baselines.
! Apparent sampling: 0.866443 visibilities/baseline/integration-bin.
! Found source: DA193
!
! There are 4 IFs, and a total of 4 channels:
!
!  IF  Channel    Frequency  Freq offset  Number of   Overall IF
!       origin    at origin  per channel   channels    bandwidth
!  ------------------------------------------------------------- (Hz)
!  01        1   1.5352e+10       8e+06          1        8e+06
!  02        2    1.536e+10       8e+06          1        8e+06
!  03        3   1.5368e+10       8e+06          1        8e+06
!  04        4   1.5376e+10       8e+06          1        8e+06
!
! Polarization(s): LL
!
! Read 933 lines of history.
!
! Reading 92952 visibilities.
```

Information of the loaded file is shown.

3. Selecting IF numbers and polarizations

IF numbers and polarizations are selected with the select command. We use all IF numbers (1 - 4) and LL polarization (correlation between left-circular polarization and left-circular polarization).

```
select LL, 1, 4
! Selecting polarization: LL,  channels: 1..4
! Reading IF 1 channels: 1..1
! Reading IF 2 channels: 2..2
! Reading IF 3 channels: 3..3
! Reading IF 4 channels: 4..4
```

The preparation of data is finished. In the next lesson, we treat visibility data.

Previous Top Next

Lesson 9 : Displaying data and time integration

Previous Top Next

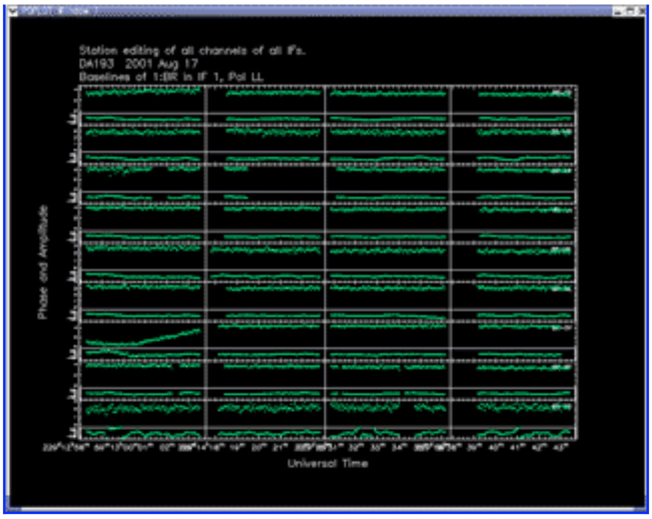
---

1. Displaying visibilities with vplot

First, let’s display visibilities as a function of time. We use vplot command.

```
vplot 9          9 means plotting nine panel a page.
! Using default options string "efbm3"
! For help move the cursor into the plot window and press 'H'.
! Applying 62 buffered edits.
```

After difmap runs vplot, the following window is popped up.



In the first page, visibilities including antenna No. 1 (BR) is displayed. The horizontal axis is time. The vertical axis is amplitude or phase.

We can move to the next page, that is, next antenna (antenna No. 2: HN) by typing [n] after activating the window with the mouse. Type [n] to move the next page... and so on. Type [p] to move the previous page. All visibilities for each baseline are displayed in difmap like this, so if you find a curious thing in some time (for example, amplitudes are very small), you can assume something is wrong with that antenna.

Type [1] to display amplitudes only (to remove phases). Type [2] to display phases only. Type [3] to display both amplitudes and phases. If the plot does not change when you type, plot is refreshed type [L] before type [(space key)].

Type [h] to display all operations with keys.

Type [x] to finish vplot.

2. Time integration with uvaver

We carry out the time integration for each twenty seconds from the four-second-step data (That is, average for each five points). We use uvaver command.

```
uvaver 20, tru          tru is an option to append the error bars to visibilities
```

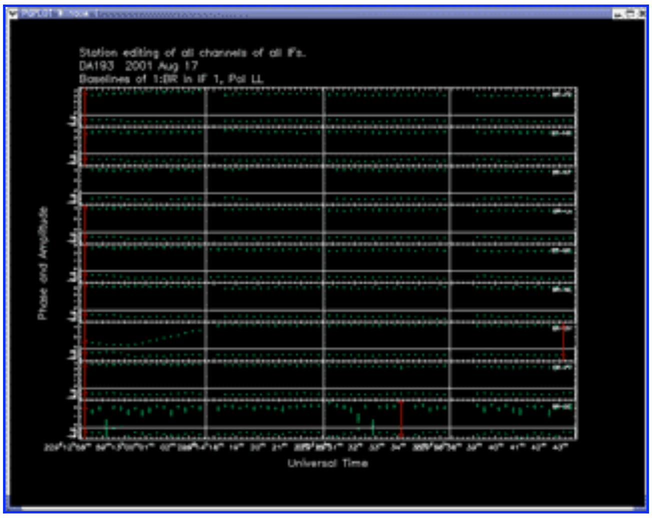
```
                        from statistics.
! Averaging into 20 second bins.
! Selecting polarization: LL,  channels: 1..4
! Reading IF 1 channels: 1..1
! Reading IF 2 channels: 2..2
! Reading IF 3 channels: 3..3
! Reading IF 4 channels: 4..4
```

3. Plotting the time-integrated visibilities with vplot

Check the time-integrated visibilities with vplot again.

```
vplot 9
! Using default options string "efbm3"
! For help move the cursor into the plot window and press 'H'.
! Applying 62 buffered edits.
```

After difmap runs vplot, the following PGPLOT window appears. If you asked to set your device type, set it to /xwin.

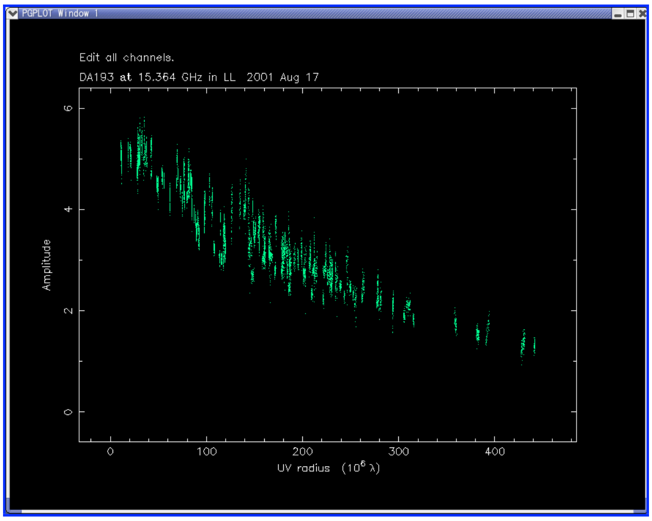


The total number of data points is decreased by the integrations. Each visibility has an error bar. This is plotted because we append the tru option in uvaver.
Do you find some visibility points displayed with red color? These indicate that the data is abandoned (flagged). When we carry out uvaver, the integrated points in the period in which the scatter of data is very large are flagged automatically. Flagging is also carried out by human eyes (this procedure is done in the next lesson). We can recover the flagged data. This is because we do not delete the data, but save some sign indicating "abandoning".
After you check, type x to finish vplot.

4. Displaying visibilities with radplot

radplot is also a command to display visibilities. The horizontal axis is a spatial frequency (that is, baseline length / wavelength).

```
radpl
! Using default options string "m1"
! Move the cursor into the plot window and press 'H' for help
```

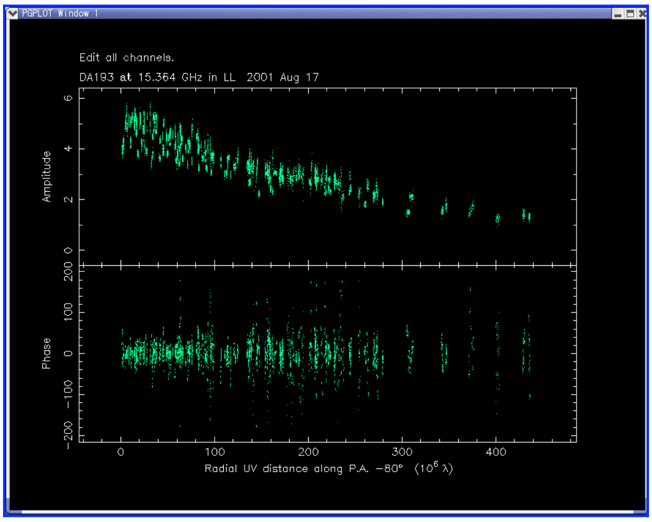


Difmap displays the amplitude components of all visibility data of all baselines. Type [n] or [p] to display the data including a special antenna. The visibilities including that special antenna are contrasted with white. Type [x] to finish.

5. Displaying visibilities with projplot

projplot is also a command to display visibilities. This is quite similar to radplot. The difference is the horizontal axis. It is “a spatial frequency along a direction”. For example projplot plot P.A. (position angle) = 0 deg in the initial state. This is the u component of (u, v). If P.A. = 90 deg, it is the v component, projplot can set the spatial frequency along an arbitral position angle as the horizontal axis.

```
projpl
! Using default options string "m3"
! Move the cursor into the plot window and press 'H' for help
```



Usage is almost similar to radplot, too ([n], [p] and [x]). We can increase or decrease by typing [<] key or [>] key, respectively (That is, we can turn the direction of spatial frequencies).

We can also set the position angle when we start radplot. For example,

```
projpl -80
```

```
! Using default options string "m3"
! Move the cursor into the plot window and press 'H' for help
```

displays the plot whose position angle is the spatial frequencies along P.A.= - 80 deg.
Next, we abandon bad data from visibilities.

Previous Top Next

Lesson 10 : Flagging bad data (avoiding bad data from analysis)

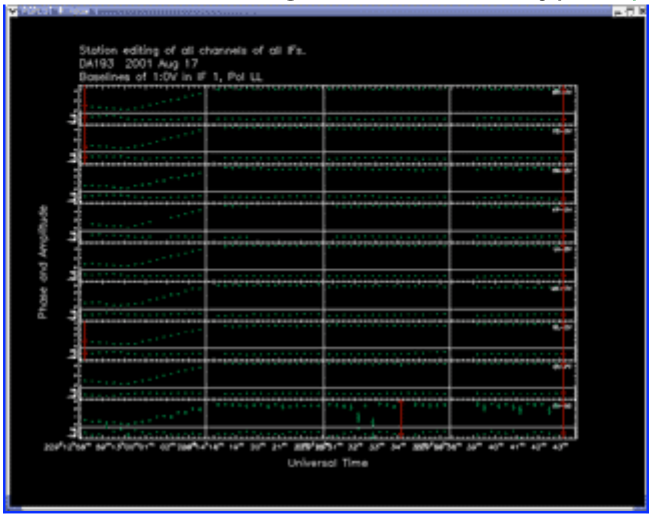
Previous Top Next

1. Flagging data with vplot

Flagging means to record which data are avoided from the analysis. Here, we flag data by human eyes. We use vplot for this purpose.

```
vplot 9          Plotting nine panels a page.
! Using default options string "efbm3"
! For help move the cursor into the plot window and press 'H'.
! Applying 62 buffered edits.
```

We display visibilities with vplot. Type n key some time and display baselines including antenna OV. Type p to back the page.



1.1. Why we need flagging?

We can find the amplitude of visibilities for baselines including OV are very small during 229d 12h58m - 13h03m. The linear increase after 13h00m is also strange.
It is possible to change the visibility amplitude with the value of (u, v) because of the source structure. But this does not explain the common variation of amplitudes for all baselines including OV. It is natural to think that the antenna OV causes this amplitude variation common for the baselines including OV. Probably, the calibration table of OV may be strange or some trouble may be occurred for OV antenna during the observation.
Anyway, we can not use these data. We need flagging.

1.2. Flagging by specifying each point

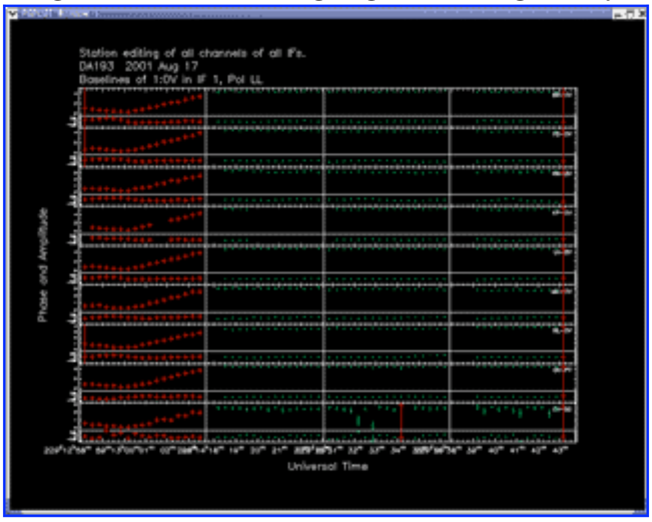
In order to flag the data, move the mouse pointer near the data and click . The nearest data from the place you clicked becomes red. This data is flagged. If you click again, the color returns from red to green. This data is not flagged.

1.3. Station editing と baseline editing

In the above procedure, nine visibilities of the same time are flagged while you click only one point. When "Station editing" is displayed at the top of the window of vplot, all visibilities of the same time in the displayed visibilities (that is, visibilities of the same time for baselines including OV) are flagged. This is useful.
However, how do we operate difmap to flag the visibilities for ONE baseline? For this purpose, type (space key). After checking "Station editing" changes to "Baseline editing", click the bad data. To change from "Baseline editing" to "Station editing", type (space key) again.

1.4. Flagging visibilities in a range

We can flag each visibility by clicking each point. But it is terrible to flag a lot of visibilities. When you flag visibilities in a range, type c and specify the range with the mouse cursor. The range is a rectangular on the vplot window and specified by clicking the opposite points. Note that this is not the drag. The range can not exceed the boundary of baselines.
When you release the flagging, type r and specify the range with mouse cursor.
Flag as the following figure using the procedure above.



Flag the other data by the similar operation.
From the next lesson, we start imaging.

Previous Top Next

Lesson 11 : Creating a dirty map

Previous Top Next

1. Creating a dirty map

We create a map at last. But we first make a "dirty" map. What does the name "dirty" means? If we can sample the spatial frequency (u, v) without any lack, we can create a complete map (brightness distribution I(l, m)). But in the real observation, the (u, v) coverage have many holes and are far from the "complete sample". When the spatial frequencies have holes, the synthesized beam B(l, m) has many sidelobes. In this case, the map we can obtain is the convolution of the true brightness distribution I(l, m) and the synthesized beam B(l, m). Therefore, the map we can obtain is also a dirty one by the effect of sidelobes. This is why the map we first create called the "dirty map". Please see this page if you would like to the relationship between the true brightness distribution and the dirty map (Sorry, in Japanese!).

However, we can estimate the true brightness distribution from the dirty map with the algorithm such as CLEAN. This is called deconvolution. This is named after deconvolution solves the convolution. If you would like to know the deconvolution algorithm with CLEAN, see this page (Sorry, mainly in Japanese!)

Anyway, Let's make a dirty map. We need the following preparation.

1. Setting the weighting of visibilities (uvweight)
2. Setting the total number and size of pixels (mapsize)

1.1. Setting the weighting of visibilities with uvweight

The dirty map is the inverse Fourier transform of visibilities. The visibility data is sampled discretely, so we carry out the discrete inverse Fourier transform. In this transform, it is not always the best way to set the same weight for all visibility data points. Visibilities are observed values, so each data point has an error. It is appropriate to set a larger weight for a small-error data, a smaller weight for a larger-error data. From the statistics, we can minimize the standard deviation by setting the weight $w_k = 1/\sigma_k^2$ for the data point whose standard deviation is σ_k. But when we set the weights proportional to the -2 powers of the standard deviations and when we use the array contains both large antennas and small antennas, the data of baselines contains small antennas are scarcely used. Here, we use the weights which are proportional to the -1 powers of the standard deviations ($w_k \propto \sigma_k^{-1}$).

When we set the weights in difmap, we use uvweight command.

```
uvweigh 2, -1   First argument means if it is a plus value, using uniform wright.
                Second argument means we use the weights which are proportional to the
                -1 powers of the standard devations.
! Uniform weighting binwidth: 2 (pixels).
! Gridding weights will be scaled by errors raised to the power -1.
```

```
! Radial weighting is not currently selected.
```

The first argument in uvweight ("2") means to make a statistics in the 2 pixel region on the grid on the (u, v) plane. This parameter is for the field of view, but we do not mention the details here.

1.2. Setting the range of map and the pixel size with mapsize

Next, we wet the range and the pixel size of the map. We should set the pixel size sufficiently smaller than the spatial resolution (size of synthesized beam). It should not be larger than the half of FWHM of the synthesized beam (Nyquist sampling). It is appropriate to be set to 1/10 - 1/8 of FWHM. The pixel size of the horizontal (east-west) direction can be different from that of the vertical (north-south) direction.
The range of the map is specified with the total number of pixels for horizontal (east-west) and vertical (north-south) direction. The total number of pixels must be the integer powers of 2 (2^n). This is because we calculate the Fourier transform with FFT (Fast Fourier Transform). The total number of pixels for horizontal (east-west) direction can be different from that for vertial (north-south) direction.
In this observation, we observe at 2 cm wavelength with VLBA (9000 km baseline). The size of the synthesized beam is (wavelength) / (baseline length) (unit : rad), so $0.02 \div 9{,}000{,}000 = 2.2 \times 10^{-9}$ rad = 0.46 mas. We set the pixel size to 0.1 mas.
We set the imaging area as 256 × 256 pixels, that is 25.6 × 25.6 mas. The observed source DA193 is a compact radio source, so the whole image is in this area. Note that we set the twice of mapping pixel in the mapsize command. Because of some affairs about FFT, the pixel size is the half of pixels in the mapsize command. It is confusing, but anyway, we need to set 512 in the mapsize command to get the imaging area of 256 pixels.

```
mapsize 512, 0.1         First argument is twice of the total number of pixels.
                         Second argument is the pixel size (mas).
! Map grid = 512x512 pixels with 0.100x0.100 milli-arcsec cellsize.
```

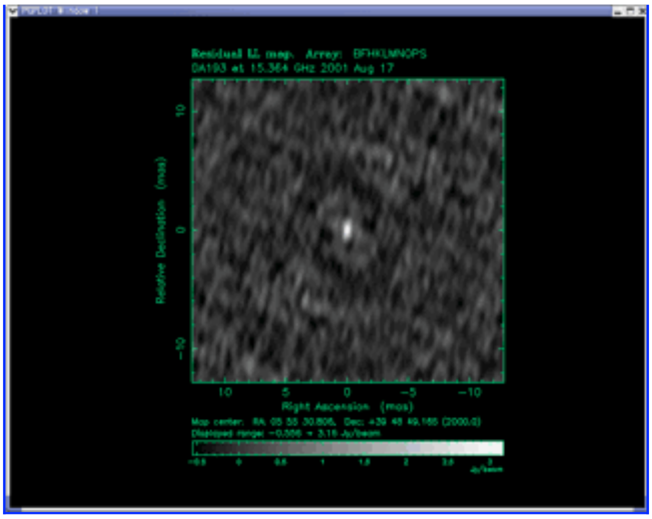
If you would like to set the different pixel size between the vertical direction and the horizontal direction, supply the arguments first for the horizontal direction, second for the vertical direction. For example, when the total number of pixels for the horizontal direction and vertical direction is 256 (= 512 / 2) and 512 (= 1024 / 2) respectively, and the pixel size of the horizontal direction and vertical direction is 0.1 mas and 0.2 mas respectively, type mapsize 512, 0.1, 1024, 0.2.

1.3. Creating a dirty map with mapplot

When you run mapplot, the dirty map is displayed.

```
mappl
! Inverting map and beam
! Estimated beam: bmin=0.4196 mas, bmaj=1.028 mas, bpa=-3.999 degrees
```

```
! Estimated noise=1.21083 mJy/beam.
!
! Move the cursor into the plot window and press 'H' for help
```

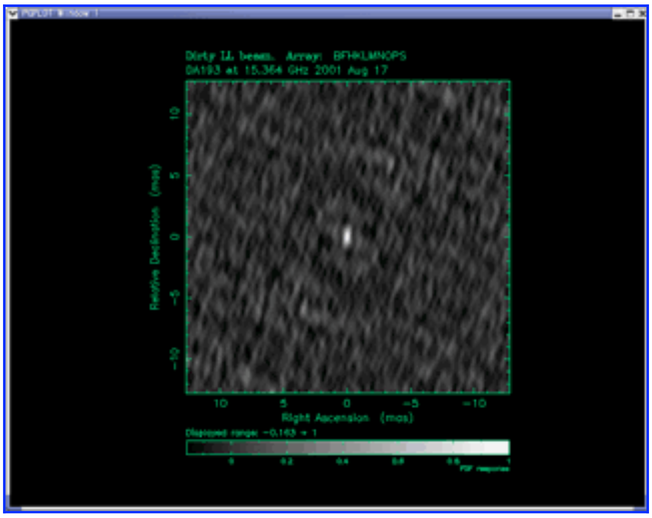


Messages tell us that the minor and major axis of synthesized beam is 0.4196 mas and 1.028 mas respectively, and that the position angle of major axis is -3.999°. We can also find that the noise level is 1.21083 mJy/beam estimated from the statistics of visibilities. These values are important, so please write them to your memorandum although they are recorded to the log file automatically. Here, we just look at the dirty map. Let's back to the command prompt without CLEAN. Type x on the PGPLOT window. You can return to the command prompt.

1.4. Displaying the synthesized beam with mapplot

When you append the beam option to the mapplot command, the synthesized beam is displayed instead of the dirty map. Let's make an image where the effect of sidelobe exists by looking at the pattern of the synthesized beam.

```
mappl beam
!
! Move the cursor into the plot window and press 'H' for help
```



Return to the command prompt by typing x key on the PGPLOT window as the previous section.

The next step is the deconvolution with CLEAN.

Previous Top Next

Lesson 12 : CLEAN

Previous Top Next

1. Running CLEAN

If you would like to know the details of the deconvolution algorithm with CLEAN, please see here. Consequently, we put a CLEAN component at the maximum brightness on the dirty map, subtract model visibilities calculated from the CLEAN component from observed visibilities, make residual visibilities and make a new dirty map with the residual visibility. CLEAN is an iteration method repeating these procedures.

We make a special setting to see the effect of CLEAN easily. When we display the residual map with mapplot, in the default setting, the correspondence between the values of map brightness and brightness on the display is that the minimum of map brightness is minimum brightness on the display and the maximum of map brightness is maximum brightness on the display. This correspondence is called transfer function. This auto scaling is user-friendly, but it is difficult to feel decreasing the residuals by CLEAN. Therefore, we fix the transfer function and make it easy to see decreasing residuals by CLEAN. Type

```
 mapfunc linear, -0.5, 3.0
                First argument specifies the linear transfer function.
                Second argument specifies the minimum of brightness.
                Third argument specifies the maximum of brightness.
! Mapplot transfer-function = linear, Data range = -0.6 -> 3 Jy.
```

The transfer function of brightness is a linear function and the range of brightness is -0.5 – 3.0 Jy/beam.

In this setting, let's plot the dirty map again.

```
mappl
!
! Move the cursor into the plot window and press 'H' for help
```

Type x key on the PGPLOT window. We return to the command prompt. Next, we carry out CLEAN with the clean command.

```
clean -1000, 0.001      CLEAN gain is 0.001.  Repeating 1000 times.
! clean: niter=-1000  gain=0.001  cutoff=0
! Component: 050  -  total flux cleaned = 0.166858 Jy
! Component: 100  -  total flux cleaned = 0.325574 Jy
! Component: 150  -  total flux cleaned = 0.476546 Jy
! Component: 200  -  total flux cleaned = 0.620151 Jy
! Component: 250  -  total flux cleaned = 0.756749 Jy
! Component: 300  -  total flux cleaned = 0.886682 Jy
! Component: 350  -  total flux cleaned = 1.01027 Jy
! Component: 400  -  total flux cleaned = 1.12784 Jy
! Component: 450  -  total flux cleaned = 1.23966 Jy
! Component: 500  -  total flux cleaned = 1.34603 Jy
! Component: 550  -  total flux cleaned = 1.44721 Jy
```

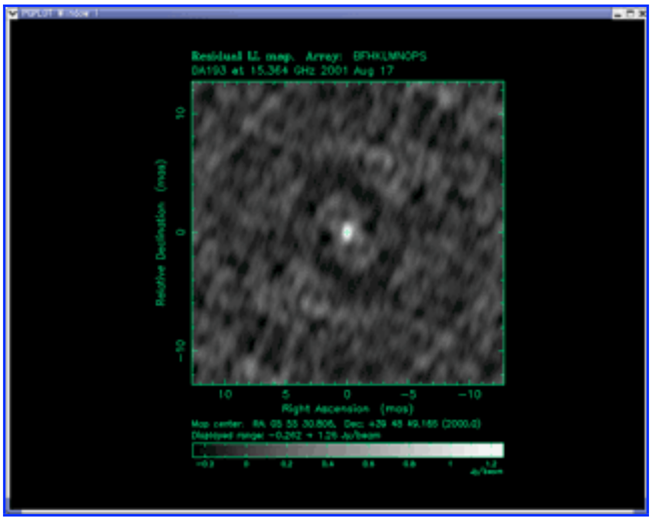
```
! Component: 600  -  total flux cleaned = 1.54345 Jy
! Component: 650  -  total flux cleaned = 1.635 Jy
! Component: 700  -  total flux cleaned = 1.72208 Jy
! Component: 750  -  total flux cleaned = 1.80491 Jy
! Component: 800  -  total flux cleaned = 1.8837 Jy
! Component: 850  -  total flux cleaned = 1.95864 Jy
! Component: 900  -  total flux cleaned = 2.02993 Jy
! Component: 950  -  total flux cleaned = 2.09774 Jy
! Component: 1000  -  total flux cleaned = 2.16224 Jy
! Total flux subtracted in 1000 components = 2.16224 Jy
! Clean residual min=-0.261898 max=1.257721 Jy/beam
! Clean residual mean=0.000468 rms=0.100505 Jy/beam
! Combined flux in latest and established models = 2.16224 Jy
```

The first argument of the clean command is repeating time. The minus value means the option that we stop the repeat if the minimum (minus) absolute value exceeds the maximum brightness. That is, we can avoid minus CLEAN components. The second argument is the CLEAN gain. The flux density of putting CLEAN component is the products of the maximum of residual map and the CLEAN gain. CLEAN gain is usually less than unity. Here we set it to 0.001 to carry out CLEAN very carefully. If the CLEAN gain is too large, it causes over CLEANing (that is, reducing too much CLEAN components). While the residual is large (CLEAN is not processed so much), you should set the CLEAN gain to a smaller value, say, at least less than 0.1. If you are careful, start with about 0.001.
Look at the message during running CLEAN. "total flux cleaned" means the sum of the flux densities of CLEANed components until the step. Ideally, you should proceed CLEAN until "total flux cleaned" becomes the total flux density (= visibility amplitude at zero spatial frequency : check with the radpl) of the observed sources. After the 1000 times repeats, the minimum value of residual map is -0.26 Jy\beam and its maximum is 1.26 Jy/beam. The positive value is still larger than the negative value, so the repeat time of CLEAN is not sufficient.

2. Checking after CLEAN

You can continue CLEAN. But display the residual with the mappl command to see the effect of CLEAN.

```
mappl
!
! Move the cursor into the plot window and press 'H' for help
```



You can find that the brightness at the center becomes fairly low and the residual is reduced. Type m. A Green cross appears at the center. This is CLEAN components. You wonder why there is only on CLEAN component though we repeat 1000 times. Difmap merge the CLEAN components at the same position. Return to the command prompt with x key.

2.1. Displaying CLEAN components with edmod

You can treat CLEAN components with text editors with the edmod command. Text editor is vi in the default. If you set the environment variable EDITOR before running difmap, you can use any kinds of editors as you like.
Anyway, edmod command is a useful command to view or edit CLEAN components.

```
! Flux (Jy) Radius (mas)  Theta (deg)  Major (mas)  Axial ratio   Phi (deg) T
! Freq (Hz)      SpecIndex
  2.0033      0.00000      0.00000
```

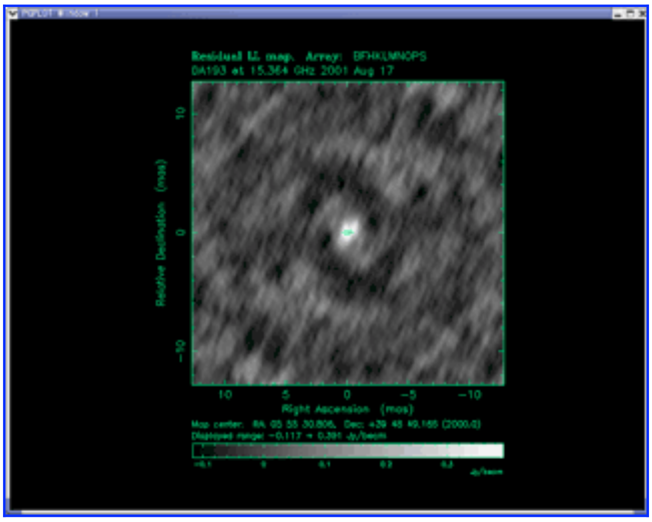
In the list displayed with edmod, the lines whose top character is "!" are comments. Therefore, only the last line has some meanings. The first column 2.16224 is the flux density of the clean component. The second and third columns show the center position of clean component. The second parameter is the distance from the map center (mas). The third parameter is the position angle (north-south direction is 0°, and measure to the counter clockwise direction). The fourth, fifth and sixth parameters have meanings when the components have a width such as in the case of elliptical Gaussian model, but we do not use because we treat CLEAN components as delta functions here.
After you check the CLEAN components, exit from edmod with the exiting command (in the case of vi, type :q).

3. CLEANing further

Let's carry out CLEAN further. If you do not write any arguments, that is, type clean, the parameters used in the previous clean are used.

```
clean
! clean: niter=-1000  gain=0.001  cutoff=0
! Component: 050  -  total flux cleaned = 0.0613528 Jy
! Component: 100  -  total flux cleaned = 0.119712 Jy
! Component: 150  -  total flux cleaned = 0.175224 Jy
! Component: 200  -  total flux cleaned = 0.228089 Jy
! Component: 250  -  total flux cleaned = 0.278769 Jy
! Component: 300  -  total flux cleaned = 0.327504 Jy
! Component: 350  -  total flux cleaned = 0.374422 Jy
! Component: 400  -  total flux cleaned = 0.419855 Jy
! Component: 450  -  total flux cleaned = 0.463995 Jy
! Component: 500  -  total flux cleaned = 0.506972 Jy
! Component: 550  -  total flux cleaned = 0.548815 Jy
! Component: 600  -  total flux cleaned = 0.589556 Jy
! Component: 650  -  total flux cleaned = 0.629223 Jy
! Component: 700  -  total flux cleaned = 0.667844 Jy
! Component: 750  -  total flux cleaned = 0.705447 Jy
! Component: 800  -  total flux cleaned = 0.742059 Jy
! Component: 850  -  total flux cleaned = 0.777706 Jy
! Component: 900  -  total flux cleaned = 0.812413 Jy
! Component: 950  -  total flux cleaned = 0.846206 Jy
! Component: 1000  -  total flux cleaned = 0.879108 Jy
! Total flux subtracted in 1000 components = 0.879108 Jy
! Clean residual min=-0.175274 max=0.649203 Jy/beam
! Clean residual mean=0.000386 rms=0.063478 Jy/beam
! Combined flux in latest and established models = 3.04134 Jy
```

After difmap finishes clean, check with mapplot. The residuals are reduced further. Type m key on the PGPLOT window. CLEAN components are displayed with green crosses. CLEAN components are placed the other places in addition to the map center. Type x key and return to the command prompt.



The residuals become small and it is not easy to see. From here, we use the auto scaling of the brightness transfer function.

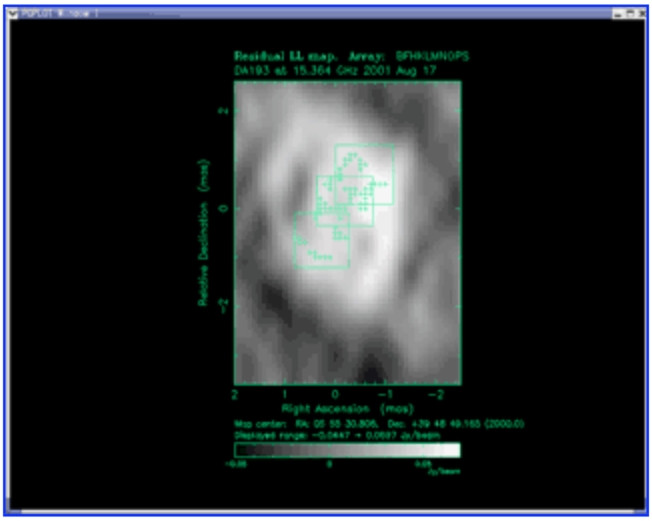
```
mapfunc linear, 0, 0     Auto scaling is used if the second and third arguments are zero.
! Mapplot transfer-function = linear, Data range = data mim -> data max.
```

4. Setting a CLEAN box

After the residual level becomes lower as CLEAN is preceded, there is a possibility to place CLEAN components at the place there are normally no brightness such as noises or sidelobes. In order to avoid it, we use CLEAN box. CLEAN boxes limit the place of CLEAN components only in the box. CLEAN boxes are set on the window of mapplot.

```
mappl
!
! Move the cursor into the plot window and press 'H' for help
```

On the window of mapplot, click at the corner of the clean box. The green lines move with the mouse cursor. Click at another corner again. A CLEAN box is set.



You can set more than one CLEAN boxes. You can place a clean box over another box. To delete a CLAN box, type d after you move the mouth pointer near the clean box.

The important things to set CLEAN boxes are the following:

1. Do not set a CLEAN box where brightness should be zero.
2. Do not set a CLEAN box where brightness is negative not to place negative-flux CLEAN components.

Set CLEAN boxes and proceed CLEAN.

After CLEAN is not effective, we aim the improvement of the map dynamic range with self-calibration.

Previous Top Next

Lesson 13 : Self-calibration of phase

Previous Top Next

---

1. What is self-calibration?

Self-calibration is a calibration method to calibrate the remaining errors of phases and amplitudes for each antenna (this is called gain errors) after a prior calibration from the observed visibilities. However, the amplitudes and phases of visibilities depend on the source structure as well as gain errors caused in each antenna. This is why we can make a map from visibilities. How can we distinguish components from the source structure with gain errors? We use the following procedure:

1. Assuming a model for the source visibility and calculate theoretical visibilities without gain errors. This is called "model visibilities".
2. The residuals are the differences between observed visibilities and model visibilities. When we change the values of antenna gains, residuals are changed because "observed visibilities" are changed.
3. Estimating the values of antenna gains which minimize the residuals. This is a solution of self-calibration.
4. Imaging again using the new solution of self-calibration.

When we use the solution of self-calibration, we must obtain a better source structure. Here, "better" means that the structure is more similar to the true source structure. Therefore, we expect that the obtained structure converges to the true brightness distribution by repeating CLEAN, self-calibration, CLEAN, self-calibration, and so on. If you would like to the algorithm of self-calibration in detail, see this page (sorry, in Japanese).

2. Self-calibration for phases only

Self-calibration is the method to solve antenna gains. An antenna gain is a complex value and can be separated to the amplitude component and the phase component. In self-calibration scheme, we can solve amplitude components and phase components simultaneously or separately. For example, solving for phase components only means to use the least-square method without changing the amplitude components.
In the normal imaging of interferometry, first we carry out the self-calibration for phase components only, and then we carry out the self-calibration for

amplitude components. This is because

1. In the least-square method, we can get the more stable solutions as the total number of estimating parameters is smaller. By fixing the amplitude components, the total number of estimating parameters becomes about half. If the total number of antennas is N_{ant}, the total number of estimating parameters for phase components is N_{ant} - 1 and that for amplitude components is N_{ant}.
2. Normally, the amplitude components of antenna gains have around 10% accuracy after the a prior calibration (effective aperture area and system noise temperature).
3. The time variation of amplitude components of antenna gains is relatively slow (stable during more than 10 minutes). But the time variation of phase components is fast (several seconds). This is because the reason of phase fast variation is the time variation of propagating delays by the change of refractive index. The only method to calibrate this fast phase variation is self-calibration if the observation is not the phase-referencing observation. Therefore, we must hurry up the self-calibration for phase components.
4. Self-calibration for phase components only is relatively safe. If you apply a wrong solution based on the wrong model, closure phases are conserved, so we can avoid converging to a strange map. On the other hand, the error of amplitude components has no limit. For example, if you give the model whose amplitude components of antenna gains are ten times and those of brightness are a hundred times, there is not confliction for the solution. Therefore, it is safe to apply the self-calibration finally after we obtain confident enough.
5. In fact, fringe fitting is a part of self-calibration for phase components modeling the point source. When you apply FRING, we already carried out the self-calibration for phase components.

2.1. Carrying out self-calibration anyway

Anyway, carry out self-calibration with the selfcal command. When you failed, you can reset by abandoning solutions.

```
 selfcal          When you carry out without arguments, difmap runs self-calibration for
                  phase components.
! Performing phase self-cal
! Adding 12 model components to the UV plane model.
! The established model now contains 12 components and 3.33074 Jy
!
! Correcting IF 1.
!
```

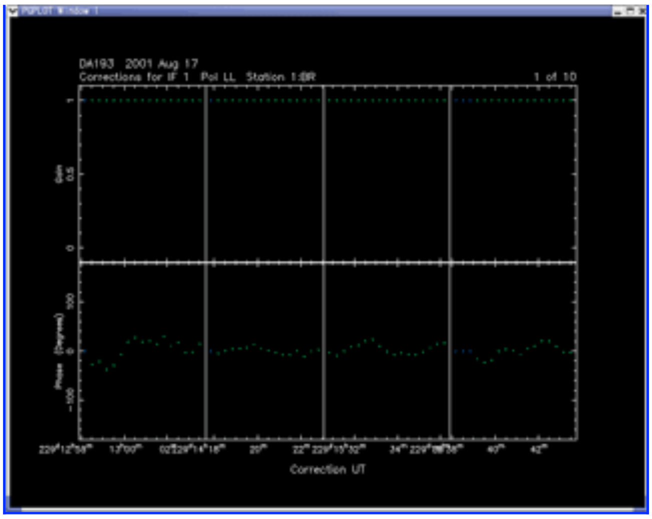
```
! Correcting IF 2.
!
! Correcting IF 3.
!
! Correcting IF 4.
!
! Fit before self-cal, rms=1.670329Jy   sigma=14.540719
! Fit after  self-cal, rms=0.985157Jy   sigma=10.196081
```

Please note the difference between "before" and "after". "rms" is the standard deviation of the residual map. The aiming value of "rms" is the estimated noise displayed you run mapplot first, 1.21083 mJy. "rms" is decreased from 1670 mJy to 980 mJy, but this is still far from the aiming value. "sigma" means how many times of visibility errors the residuals is. This is a square root of the reduced χ^2. When this value becomes 1, there is no systematic error (that is, gain error). We reached the goal.

2.2. Checking the result of self-calibration

Let's check the result after you carry out self-calibration. We will check two items: the solutions of antenna gain and the map.
The solutions of antenna gain can be checked with the corpl command.

```
corpl
! Move the cursor into the plot window and press 'H' for help
```



The value corrected the antenna gains is displayed on the PGPLOT window. The upper panel is amplitude components. This time we fixed them, so they are 1. The lower panel is phase components. You can see they vary with time. The points displayed in blue are the time failed to estimate solutions. They are flagged. To display the solution of the next antenna, type n key. Type p key to display the solution of the previous antenna. After you finish the check, type x key and return to the command prompt.
If the result is wrong, you can reset the result of self-calibration with the uncal

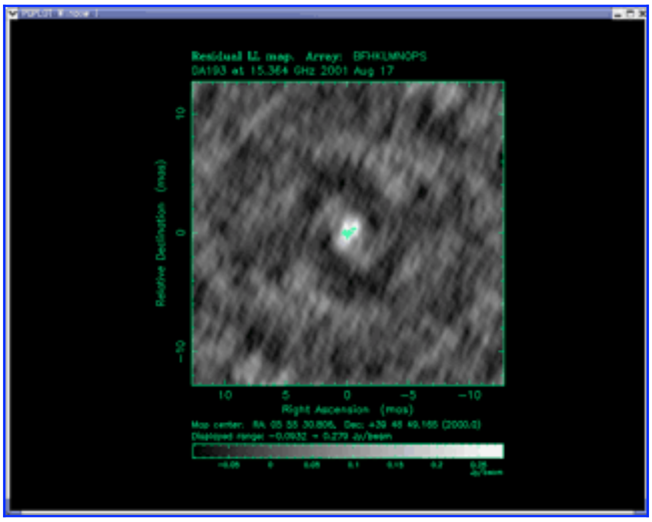
command, but here we do not carry out uncal.

```
uncal  true,  true,  true
                        Three  arguments  are  phase  components,  amplitude  components,  flagging.
                       When  they  are  "true",  difmap  resets  the  solutions.
                       When  they  are  "false",  difmap  retain  the  solutions.
```

2.3. Map after self-calibration

What is the residual map after self-calibration? Let's check it with the corpl command.

```
mappl
!  Inverting  map
```



Comparing the map before self-calibration, the brightness where CLEAN components are placed is increased. Self-calibration for phase components have an effect to correct brightness scattered by the phase errors of antenna gains.

3. CLEAN after self-calibration

We continue CLEAN using visibilities improved by self-calibration. If the residuals become small, you had better carry out self-calibration for phase components again.

3.1. How deep you should CLEAN?

How many times we should keep the repeating process of self-calibration for phase components and CLEAN? The next step is self-calibration for both phase and amplitude components. The amplitude terms of antenna gains are operated to fit visibilities to all flux density of models. Therefore, we need to continue CLEAN until "total flux cleaned" reaches almost the total flux density. You can see the total flux density by displaying visibility amplitude at zero spatial frequencies with the radpl command or the projpl command. In this

observation, it is about 5.15 Jy.
The next step is self-calibration for amplitude term.

Previous Top Next

Lesson 14 : Self-calibration of amplitude

1. Calibrating amplitude components of antenna gains with gscale

After CLEAN is enough, let's calibrate the amplitude components with self-calibration.
Two commands, gscale and selfcal are supplied. First we use gscale.
Self-calibration with gscale has the following restrictions:

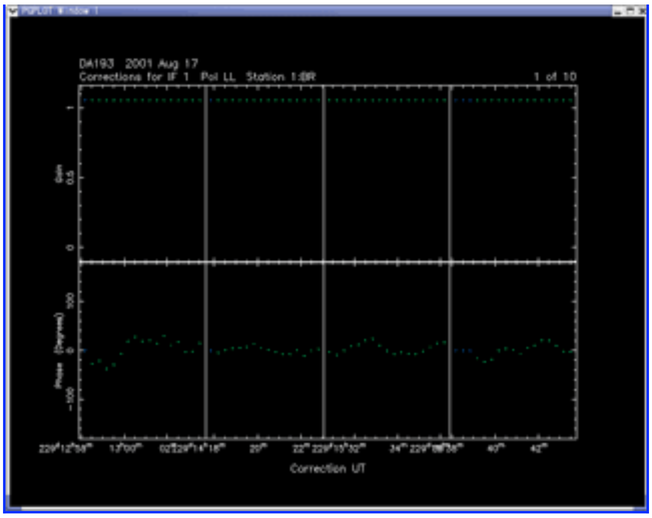
1. Do noting about phase.
2. Estimating common solutions for all time range
3. Correcting relative amplitudes between antennas

The second restriction means not to consider the time variation of amplitudes. This command is used to correct the characteristics of antennas, for example, the difference between the catalog value and true value of effective aperture area, systematic high (or low) system noise temperatures. The third restriction means to normalize the amplitude components. Total flux density is conserved. The gscale command is relatively safe self-calibration of amplitude because the total number of estimating parameters is small.
Then, let's run gscale.

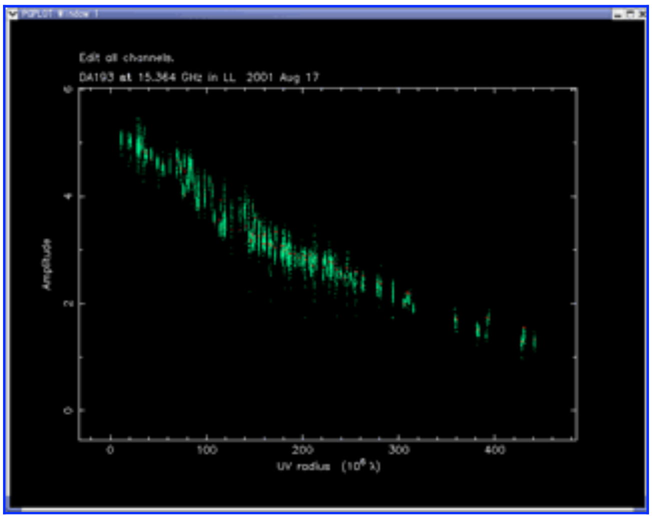
```
gscale          When we run gscale without arguments, amplitude components are normalized.
! Performing overall amplitude self-cal
!
! Correcting IF 1.
!  Amplitude normalization factor in sub-array 1: 1.00565
!  Telescope amplitude corrections in sub-array 1:
!   BR         1.05      FD         0.97      HN         0.96      KP         0.93
!   LA         0.99      MK         1.03      NL         1.01      OV         0.97
!   PT         1.05      SC         0.99
!
!
! Correcting IF 2.
!  Amplitude normalization factor in sub-array 1: 1.00874
!  Telescope amplitude corrections in sub-array 1:
!   BR         1.06      FD         0.97      HN         0.95      KP         0.94
!   LA         0.99      MK         1.03      NL         1.00      OV         0.97
!   PT         1.03      SC         0.97
!
!
! Correcting IF 3.
!  Amplitude normalization factor in sub-array 1: 1.01128
!  Telescope amplitude corrections in sub-array 1:
!   BR         1.07      FD         0.98      HN         0.94      KP         0.95
!   LA         0.99      MK         1.01      NL         1.00      OV         0.98
!   PT         1.04      SC         0.93
!
!
! Correcting IF 4.
!  Amplitude normalization factor in sub-array 1: 1.00202
!  Telescope amplitude corrections in sub-array 1:
!   BR         1.06      FD         0.97      HN         0.94      KP         0.99
!   LA         0.97      MK         1.04      NL         1.03      OV         1.03
```

```
!   PT        1.01     SC        0.93
!
!
! Fit before self-cal, rms=0.285136Jy  sigma=2.327047
! Fit after  self-cal, rms=0.209040Jy  sigma=1.621514
```

Amplitude correction factors for each antenna are output in the messages. If they are 1.0, no correction is required. In this data, maximum is 1.07 and minimum is 0.93, so they are in ±7%. The a prior calibration of amplitudes was relatively accurate. You can check antenna gains with the corpl command.



Check the visibility amplitudes corrected with gscale with radpl. You can find that the amplitude components of gains are not 1.



The above figure is the result of radpl after gscale. Here is the result of radpl before gscale. The scatter of amplitudes becomes smaller after gscale.

2. CLEAN further

The residual rms of brightness becomes fairly small by gscale. The square root of reduced χ^2 also becomes small, 1.621514. Let's proceed CLEAN further with these visibilities. In some cases, you should CLEAN after you abandon all CLEAN components.

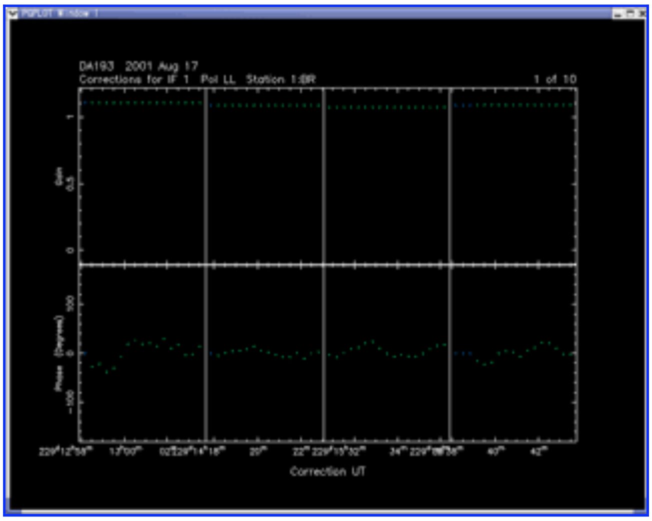
3. Self-calibration including amplitude components

After you think CLEAN is sufficient, you should carry out self-calibration of amplitude components. Note that the "total flux cleaned" has to be around the total flux density at this point. You can check total flux density with the radpl command or the projpl command.

First we set the variation time scale of amplitudes to a long duration, 60 minutes. This is because we would like to obtain a stable solution.

```
selfcal true, true, 60
                First argument : true means to correct amplitude components.
                Second argument : true means to allow the time variation of
                amplitude components.
                Third argument means the time scale (unit : minute)
! Performing amp+phase self-cal over 60 minute time intervals
!
! Correcting IF 1.
!
! Correcting IF 2.
!
! Correcting IF 3.
!
! Correcting IF 4.
!
! Fit before self-cal, rms=0.187854Jy  sigma=1.442203
! Fit after  self-cal, rms=0.161802Jy  sigma=1.144859
```

The square root of the reduced χ^2 approaches to the target value, 1. We can check the solution with the corpl command similar to the previous lessons.



The amplitudes are constant in the five minute duration because the time scale of the solution is sixty minute. But they change between the five minute durations because they are separated by more than one hour.

Then, proceed CLEAN further and repeat self-calibration including amplitude components with the smaller time scale. When reduced χ^2 becomes almost 1, CLEAN is finished.

Finally, make a final map and save it.

Previous Top Next

Lesson 15 : Creating a final image

Previous Top

---

1. Displaying the CLEAN Map

If you finish CLEAN and self-calibration, let's display a final CLEAN map. When you append the cln option to the mapplot command, difmap displays a CLEAN map which is the brightness by CLEAN components and the residual map.

The brightness by CLEAN components is the convolution of CLEAN components with restoring beam. Normally, CLEAN components are delta functions, so if we put them simply, the brightness distribution has a lot of spikes. We carry out a moderate smoothing. The function convolved for this smoothing is called restoring beam. In defaults, an elliptical Gaussian whose major and minor axes are same as those of the synthesized beam is used for the restoring beam. When we carried out the mapplot command for the first time, the major and minor axes of the synthesized beam is 1.0 mas and 0.4 mas respectively. When you specify nothing, these values are used. When you would like to use other restoring beams than defaults, you can use restore command. However, we do not mention here.

```
 mappl cln               Specifying to make a CLEAN map with the argument cln.
! Inverting map
! restore: Substituting estimate of restoring beam from last 'invert'.
! Restoring with beam: 0.4186 x 1.03 at -4.021 degrees (North through East)
! Clean map  min=-0.011036  max=3.349 Jy/beam
!
```

If you satisfied this map, the remaining thing is to improve the display and save the results.

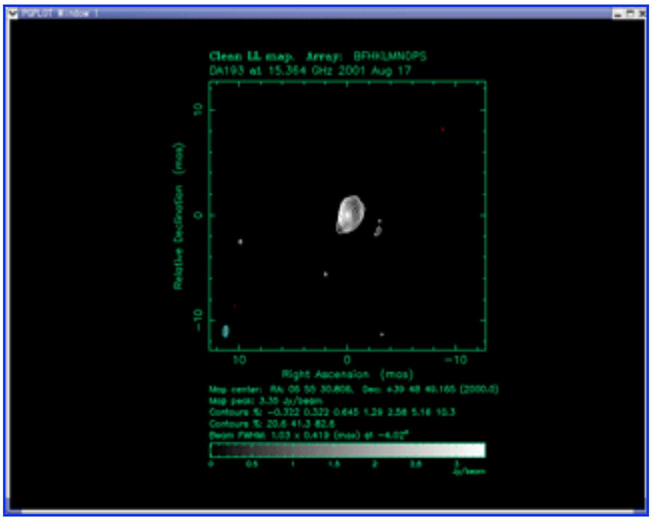
2. Setting the contour levels

In defaults, the contour levels of a CLEAN map are 1%, 2%, 4%... that is, 1% $\times 2^n$ of the brightness peak. We can set these contour levels with the loglevs command.

Here, we set the minimum contour levels to $\pm 3\sigma$, and $3\sigma \times 2^n$, that is, 6σ, 12σ, etc. The messages we CLEAN finally say that the image r.m.s. is σ = 0.0036 Jy/beam. On the other hand, the messages we carry out mappl cln, the peak brightness is 3.349 Jy/beam. Therefore 3σ is 0.3224843236787100627% of the peak brightness （0.003600 $\times$ 3 / 3.349 $\times$ 100 = 0.3224843236787100627). We type loglevs command as follows.

```
loglevs 0.32248432367871006270, 100, 2
                 Specifying the contour levels from 0.322...% to 100% of the peak brightness
                 with the 2^n steps.
! The new contour levels are:
!  -0.322484 0.322484 0.644969 1.28994 2.57987 5.15975 10.3195 20.639 41.278 82.556
```

When you display the CLEAN map with mappl cln, the contour level becomes what we specified.



3. Saving the results

We save the results of imaging. Here, we explain three saving methods.

3.1. Saving for difmap

The save command saves all, that is, visibilities, CLEAN components, maps, parameters and BOXes.

```
save BK084.DA193.FINAL.IMAGE     Specifying the output file name with the argument.
! Writing UV FITS file: BK084.DA193.FINAL.IMAGE.uvf
! Writing 112 model components to file: BK084.DA193.FINAL.IMAGE.mod
! Writing clean map to FITS file: BK084.DA193.FINAL.IMAGE.fits
! Writing difmap environment to: BK084.DA193.FINAL.IMAGE.par
```

Do not forget the save command. It is possible to save on the halfway of analysis. When we start difmap next time, we can return the situation with the get command. We can read the settings of mapsize, uvweight etc if you specify the parameter file with the @ command.

```
get BK084.DA193.FINAL.IMAGE
@ BK084.DA193.FINAL.IMAGE.par
```

3.2. Saving a map as an image file

The mappl command display a map on the PGPLOT window in defaults. But, when you specify the displaying device beforehand, it can write to a PostScript file or GIF file. We use the device command to specify the device.

```
device /vps              Setting the output to a portrait PostScript file.
! Attempting to open device: '/vps'
mappl cln
```

A file "pgplot.ps" is created in the directory you start difmap. The relationship between the argument of device command and the output is as follows:

Argument	Output	Notes

/xw	X-Window	Displaying a PGPLOT window on the screen, not a file
/xserv	X-Window	Remaining as a PGPLOT window
/ps	pgplot.ps	A landscape PostScript file
/vps	pgplot.ps	A portrait PostScript file
/gif	pgplot.gif	A landscape GIF file
/vgif	pgplot.gif	A portrait GIF file
/null	nothing	Using to close a file correctly.

3.3. Saving a map as an image FITS file

By writing a map to an image FITS file, we can read the map data with other imaging software such as AIPS. We use the wmap command. This is a shortened style of Write MAP. Probably it has no relationship with Wilkinson Microwave Anisotropy Probe…

```
wmap BK084.DA193.FINAL.MAP.FITS          Argument is the name of the output file.
! Writing clean map to FITS file: BK084.DA193.FINAL.MAP.FITS
```

When you read the output file with AIPS, all of the file name should be capital characters.

The course is finished here. Thank you. Use the quit command to finish difmap.

```
quit
! Quitting program
! Log file difmap.log closed on Mon Sep 11 22:35:00 2006
```

The log for all operation is saved as a file, "difmap.log". Let's change the log-file name to a name which is easy to understand the contents. We also change the filename of pgplot.ps.

```
mv difmap.log BK084.DA193.FINAL.IMAGE.log
mv pgplot.ps BK084.DA193.FINAL.IMAGE.ps
```

Finally, we display an example of final files.

File name	Notes
BK084.DA193.FINAL.IMAGE.fits	The image file saved with the save command (Binary)
BK084.DA193.FINAL.IMAGE.uvf	The visibility file saved with the save command (Binary)
BK084.DA193.FINAL.IMAGE.mod	The CLEAN components file saved with the save command (ASCII)
BK084.DA193.FINAL.IMAGE.par	The parameter file saved with the save command (ASCII)

BK084.DA193.FINAL.MAP.FITS	The image FITS file saved with the wmap command (Binary)
BK084.DA193.FINAL.IMAGE.ps	The image PostScript file saved with the mapplot command (ASCII)
BK084.DA193.FINAL.IMAGE.log	The log file (ASCII)

Previous Top