AIPS: An Introduction

Jayanti Prasad
Inter-University Centre for Astronomy & Astrophyics (IUCAA) Pune (India)

July 20, 2011

This short tutorial is meant to help beginners (including me !) to get started with the Astronomical Image Processing System $\mathcal{AIPS}$without knowing much of the physics or mathematics behind (for that read my other notes). The goal of this tutorial is to introduce various $\mathcal{AIPS}$tasks which are used for flagging (editing the bad data), calibration and imaging. In order to use $\mathcal{AIPS}$we use the following sequence of tasks.

1. Set the environment:-

   ```
   $export H=/data/aips/data/FITS/
   $source ../aips/LOGIN.SH
   ```

   This is done to specify the location from where the input files will be read and/or the output files will be written. Note that the above is true only form the *bash* shell.

2. Starting and ending $\mathcal{AIPS}$:-

   ```
   $aips tv=local
   ```

   or

   ```
   $aips tv
   ```

   This will start $\mathcal{AIPS}$and which we can stop (once we are done) using

   ```
   >kleenx
   ```

   The next task is to load a FITS file which is done using the next task

3. Load the input FITS file (**FITLD**):-

   ```
   >task 'fitld'
   >infile 'H:FIELD2.FITS'
   >go
   ```

   Note that "infile" is now know as "datain" in the newer versions.

4. Index the data for quick access (**INDXR**):-

   ```
   >tget indxr
   >getname  1
   >go
   ```

This task "index" the UV data. Note that if you are facing any difficulty, run the command "default" which will reset all the parameters for the task.

5. Viewing the data in a tabular form (**LISTR**):-

```
> tget  'listr'
> getn 1
> optype 'SCAN'
> inp
> go
```

if we set optype 'LIST', then the data will be presented in a matrix form.

6. Plotting visibility for various baselines (**VPLOT**):-

```
> tget 'vplot'
> getn 1
> ant 0
> baselines 0
> timerange 0
> stokes 'rr'
> bchan 35
> echan 35
> nplots 5
> dotv 1
> inp
> go
```

7. Plotting visibilities as a function of baselines (**UVPLT**):-

```
> tget 'uvplt'
> getn 1
> source 'FIELD2';
> inp
> go
```

8. To find very low and high values of visibilities(**UVFND**):-

```
> tget 'uvfnd'
> getn 1
> aparam 50 0 10 0
> inp
> go
```

This task will show all the data points which are above 50 and below 10.

9. Flagging the data using external FLAG file (**UVFLG**):-

```
> tget 'uvflg'
> getn 1
> infile 'H:TEMP.FLG'
> inp
> go
```

Here 'TEMP.FLG' is the external flag file which which follows some syntax.

10. Setting the flux of the primary calibrator i.e., fluxcal (**SETJY**):-

```
>tget  'setjy'
>sources '3C147'
>optype 'CALC'
> go
```

This task will write the flux of the flux calibrator in the SU table which can be checked by the following command:

```
>inext 'SU'
>invers 1
>go prtab
```

11. Computing antenna based complex gains (**CALIB**):-

```
>tget 'calib'
>getn 1
>calsour ''
>SOLMODE    'A&P'
>solint 16/60
>refant 1
>bchan  35
>echan  35
>docalib 1
>inp
>go ;
```

This will create an 'SN' table.

12. Managing table (**INEXT**):- Note that at any point of time the list of tables can be seen using the follwing command

```
>imheader
```

Any table can be printed on the terminal using **PRTAB**. For example if we want to print 'SN' table, we will use

```
> inext 'SN'
> inver 1
> go prtab
```

Any table be removed using

```
> inext 'SN'
> inver 1
> go extdest
```

If the status of the table is busy (read/write) use the following:

```
> inext 'SN'
> inver 1
> clrst
> go extdest
```

13. Finding the absolute values of fluxes for secondary calibrators (**GETJY**):-

```
>tget  'getjy'
>calsour '3C147''
>sources '0632+103' '3C172'
>inp
>go
```

Now check that all the calibrators got the flux values. Note that sources in the 'SETJY' should be from the subset of calsources in 'CALIB'.

```
>inext 'SU'
>invers 1
> go prtab
```

Note that the sources given in GETJY should be from the subset of calsources given in CALIB.

14. Apply the calibration (**CLCAL**):-

```
> tget 'clacl'
> getn 1
> sources ''
> inp
> go
```

Note that here it is useful to specify all the sources because it will update the 'CL' tables for all. The calibration for any task like **uvplt**, **imagr** etc., can be invoked by giving DOCALIB 1 and gainuse 0 (if it is not that). This will apply the highest version of the CL tables.

15. Making the image (**IMAGR**):-

```
> tget imagr
> getn 1
> source '3c147' ''
> bchan 35
> echan 35
> stokes 'rr'
> imsize 512 512
> cellsize 8 8
> niter 1
> inp
> go
```

16. Gray scale plots (**GREYS**)- We can label the image with coordinates and flux using the following:

```
> tget greys
> getn 20
> go
```

This assumes that the catalogue of the image is 20

17. Writing the image in a file (**TVCPS**)- We can capture the TV screen shot and write the image using the following:

```
> tget tvcps
> getn 1
> infile 'H:IMAGE.PIX'
> inp
> go
```

18. Loading an image in TV (**TVLOD**):-

```
> getn 4
> tvlod
```

Will load an image corresponding to cat 4 in the TV.

try this

```
>tvwindow; tvall
```

19. Subtracting an image from UV data (**UVSUB**):-

```
> tget uvsub
> getn 1
> getn2n 4
> go
```

will subtract an image with cat 4 from a uv data set with cat 1 will write the resultant image in the next cat.

20. Subtracting an image from UV data (**UVAVG**):-

```
>tget uvavg
>getname 1
>yinc 16
>go
```

This will average the uv data over 16 seconds and will make another uv file.

21. Copying a file on disk (**FITTP**) :-

```
>task 'fittp'
>getname 5
>outname 'H:TEST.FITS'
>go
```

Note that now ”outname” is replaced by ”dataout” in newer versions.

22. Joining two UV data FITS files(**DBCON**) :-

```
>task 'dbcon'
>getname 1
>getn2n  2
>outname 'TEST.FITS'
>go
```

Note that now "outname" is replaced by "dataout".

Self calibration

After flagging and calibrating the data, it is time to do self calibration. The first step of self calibration is to make a single source file from the mutisource file using the task 'SPLIT'.

1. Make a single source file:-

```
>tget 'split'
>getn 1
>source 'FIELD2'
>inp
>go
```

This will create a UV data file on which we can apply imagr.

2. Create Clean Components (CC):-

```
> tget imagr
> getn 2
> bchan 65
> echan 65
> stokes 0
> cellsize 8  8
> imsize 512 512
> niter 10000
> inp
> go
```

Note that **niter 0**, will provide an interactive window for carrying out various tasks. There are at least three tasks which have to done. In order to chose an option, click on the button and double hit A, B or C. The first thing which is generally done is to select 'TVFIDLE' option and double hit A and move mouse around till there are just a few bright spots. Make boxes around these bright spots selecting the option 'MAKEBOX'. Once you are done with that, double hit 'D' which will bring you back the main menu. Now select 'CLEAN' option from the main menu. Follow the suggestions pooping up on messenger window. Once you are done with cleaning, you will have two new files, one a Clean component file (CC) and another a dirty beam file. It is suggested to see the CC image file using the task 'TVLOAD' and check its rms using tvstat which is using in the following way:

```
tvstat go
```

make a close contour on the screen using mouse and then hit 'A' and 'C'. It will print the image statistics on the terminal including rms. This is important to check at every cycle of cleaning. Now you have a clean component which you can use to do self calibration.

3. Self-Calibrate using a Clean Component (CC):-

   Self calibrating takes two input, one UV data file and another CC.

```
>task 'calib'
>getn 1
>get2n 4
>ncomp 81 0
>docalib -1
>doband -1
>refant 6
>solint 5
>aparm 3 0
>doflag -1
>soltype 'l1r'
>solmode 'p'
>inp
>go
```

   Note that by setting "docalib=-1" we are informing the CALIB that this is a self calibration. This task will take some time and if successful then will create a new UV file. Now we can create CC from the new UV file using imagr and run the loop: imagr-calib-imagr-calib, and keep checking the rms of image after every cycle. If we find that rms is not changing (falling) from one iteration to another we can stop self calibration.

Comments:-

- One of the common mistakes beginners do is that they do not clear the parameters before running a task. In general by using 'default' one can delete all the parameters of the last run. To make sure that all the parameters for a task are correct use 'inp' before running (typing 'go') a task.
- The command **ucat** will print the catalogue of all the UV file in the aips.
- The command **pcat** will print the list of images with their catalogue numbers.
- Any data file or image file can be deleted using the following task;
- If you are running AIPS first type replace 'tget' by 'task' everywhere.

```
> getn 4
> zap
```

Note that if the status of the file is busy, we can use the following:

```
> getn 4
> clrst
> zap
```

If we want to delete all the file from catalogue number 1 to 9 we use the following:

```
> for i = 1 to 9; getn i; zap ; end
```

- $\mathcal{AIPS}$ is case insensitive. You can use upper or lower case letters.
- You do not need to type full command. If there is no ambiguity $\mathcal{AIPS}$ will do the job. For example in place of 'imheader' only 'im' will also work.
- To know all the commands starting from a letter enter 'TAB' after that letter.
- If you are not sure about any task or parameter use 'help' and 'explain' options. For example if you want to know more about the task 'calib' use

  ```
  > explain calib
  ```

  or

  ```
  > help   calib
  ```

- If $\mathcal{AIPS}$complains that the log file has become large clear that using

  ```
  > clrmsg
  ```

- Any running task can be aborted using ABORT

  ```
  >abort uvplot
  ```

  will abort UVPLT task.

1 Some important points

Cleaning :-

Due to finite number of different baselines, the UV plane is not properly sampled, or amplitude and phase information for certain "frequencies" is missing. Whatever, amplitude or phase these missing frequencies have, does not have any effect on the image. The image which we get (dirty image) is the convolution of the real image and the response (dirty beam) of the UV covarge.

The real image of the source is recovered by deconvolving the dirty beam from the observed image using an iterative method called cleaning.

If we assume that the amplitude of the frequencies which are not measured is zero, then the corresponding image is called the *principle solution*