SMA Data Reduction with CASA

Masaaki Hiramatsu
ALMA Regional Center Taiwan node/ASIAA
hiramatsu@asiaa.sinica.edu.tw

November 10, 2010

Abstract

Here I introduce the SMA data reduction procedure with CASA.

1 Introduction

1.1 Minimum Introduction of CASA

CASA, the Common Astronomy Software Applications package, is a software for the data reduction in radio astronomy. After installing CASA into your computer properly, you can launch CASA by typing

```
> casapy
```

A CASA logger window appears and now you can execute various CASA commands. The command list is shown by typing

```
CASA> tasklist
```

The parameters for the tasks can be checked with `inp TASKNAME` and specified at once by

```
CASA> plotxy(vis="hh212_all.ms",xaxis="channel",yaxis="amp",....)
```

or one by one

```
CASA> vis="hh212_all.ms"
CASA> xaxis="channel"
....
CASA> go plotxy
```

If the parameters are specified incorrectly, that part turns red in the `inp` output. Blue means good and black indicates the default values. You can read the help file by

```
CASA> help TASKNAME
```

I suggest you to read the help file and the cookbook (see §1.4) whenever you have questions.

1.2 Outline of the Procedure

In this material I show the data reduction procedure as follows.

1. Data Preparation and Import
2. Data Inspection and Flagging
3. Bandpass Calibration
4. Flux Scaling
5. Gain Calibration
6. Imaging

The detailed explanation of each step is out of the scope of this article. Useful lecture materials are available in the web site for the Second Asian Radio Astronomy School in 2008[1].

1.3 Sample Data and Script

The sample data are the two-field mosaic observation of a protostar HH212 with SiO ($J = 8 - 7$, 347 GHz) and CO ($J = 3 - 2$, 345 GHz) in USB. The calibrators are Uranus (Flux calibrator), 3c454.3 (Bandpass calibrator), and 0530+135 (Gain calibrator). The data were also used in the Second Asian Radio Astronomy School and the data reduction procedure with MIR/Miriad is explained in the web site of the School, so you can compare the process with CASA and MIR/Miriad. For the detailed description of the observations, please read the paper Lee et al. (2007) *ApJ*, 659, 499.

The CASA (Python) script data used in this material as well as the sample data (a tar.gz file) are available in **http://www.asiaa.sinica.edu.tw/%7Ehiramatsu/CASA/**. This script requires CASA ver 3.0.2. Some tasks do not work properly on the older versions of CASA because of the modification of the parameter keywords. Please make sure the version of CASA. You can simply run this script and get the results, but I strongly recommend to copy and execute each command to check what's happening.

There is another sample CASA script for SMA data reduction written by Dr. Crystal Brogan at NRAO. The script and the sample data are available at "Tutorials and Training" on NRAO CASA website[2]. The script has more careful calibration process than the one used in this tutorial. It would be good to start with the script in this tutorial to figure out the basic calibration/imaging process with CASA. Once you get familiar with the CASA's methodology and if you would like to do more careful and complicated calibration, you might try Crystal's script.

1.4 SMA-specific issues

From the next section I introduce how to reduce the SMA data with CASA. There are several SMA-specific issues, especially in importing the data to CASA. CASA can handle the ASDM (ALMA Science Data Model) dataset which is the default data format of ALMA, so you do not need to consider the following issues.

First, the current version of CASA cannot apply the $T_{\rm sys}$ correction. Therefore you need to do it in MIR in advance. Second, CASA cannot load raw SMA data. So the data should be converted to UVFITS after the $T_{\rm sys}$ correction in MIR. You can use CASA's task `importuvfits` to convert the UVFITS to CASA format (Measurement Set, MS). Note that you need to convert each SMA chunk to separate UVFITS, otherwise all the chunks are merged into one spectral window and you cannot calibrate the bandpass properly.

[1] `https://www.asiaa.sinica.edu.tw/act/radio_school/2008/index.php`
[2] `http://casa.nrao.edu/tutorial.shtml`

1.5 References

You can obtain the various useful information of CASA in the NRAO CASA website[3].

- Obtaining CASA: `http://casa.nrao.edu/casa_obtaining.shtml`
- CASA Wiki: `http://casaguides.nrao.edu/index.php`
- Cookbook: `http://casa.nrao.edu/ref_cookbook.shtml`
- Tutorials and Training: `http://casa.nrao.edu/tutorial.shtml`

2 Data Preparation and Import

The sample tar.gz file contains the UVFITS files for all the sources. Put this file and the sample script in your working directory and launch CASA by typing `> casapy`. Hereafter the CASA prompt is shown simply by `>`. All the commands in the following should be run on CASA.

At the beginning of the sample script we prepare the environment. After deleting unnecessary files and directories,

```
> os.system('mkdir spw_ms_files')
> os.system('tar zxf hh212uvfits.tar.gz')
```

creates a data directory for MS files and extract the tar.gz file to a directory `uvfits`. This directory contains a number of UVFITS files. The files are named as `SOURCENAME.spw(1-24).fits`.

The commands from Line 43 in the script are the Python loops to convert the UVFITS to MS format (`importuvfits`) and concatenate 24 MSs for 24 spectral windows into one MS for each source (`concat`). Note that if you copy and paste the commands, please copy from Lines 43 to 57 (from the beginning to the end of the python for loop) at once and paste them. Concatenate the MSs for each source into one MS with

```
> concat(vis=visfiles,concatvis='hh212_all.ms',timesort=True)
```

The option `timesort=True` sorts the data along the time so that we can calibrate the time-dependent gain. The Measurement Set `hh212_all.ms` contains all the data.

3 Data Inspection and Flagging

3.1 Data Inspection

A task `listobs` displays the detailed information of the data in the CASA logger. You can check the names and Field IDs of the objects, the channel numbers and frequencies of the spectral windows. Note that the spectral window (spw) ID starts from 0, not 1. We use these source (field) and spw IDs later.

```
 ID   Code Name         RA              Decl           Epoch   SrcId
0         URANUS       22:35:52.4345 -09.40.22.2038 J2000   0
1         HH212_S      05:43:51.1604 -01.03.01.2800 J2000   1
2         HH212_N      05:43:51.6393 -01.02.44.7198 J2000   2
3         3C454.3      22:53:57.7464 +16.08.53.5648 J2000   3
4         0530+135     05:30:56.4165 +13.31.55.1505 J2000   4
```

[3] `http://casa.nrao.edu/`

```
SpwID  #Chans Frame Ch1(MHz)    ChanWid(kHz)TotBW(kHz)  Ref(MHz)    Corrs
0          64 LSRK  347471.17   -1625       104000      347471.17   XX
1         256 LSRK  347389.776  -406.25     104000      347389.776  XX
2          64 LSRK  347313.164  -1625       104000      347313.164  XX
3          64 LSRK  347231.161  -1625       104000      347231.161  XX
...
```

You can check the data with `plotms` command. `plotms` is a powerful data plotter with various options. You can set the options with the command line as well as the GUI interface. Figure 1 is the Time v.s. Phase plot. If you find some bad data points in this plot, you can select these with "Mark Regions" button and flag them with "Flag" button.

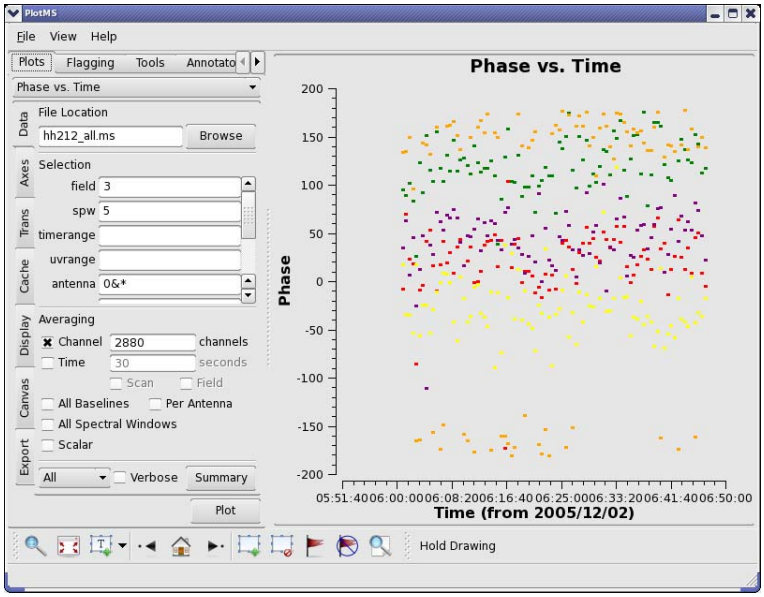


Figure 1: Sample of the plotms window. Time-Phase plot of the field "3", spw "5" with the baseline which contains antenna "0". The color shows different baselines, which can be specified in the Display tab.

`plotms` is still under development, and for example, currently `plotms` cannot plot the data points in multiple panels like MIR and always overlays them. You can also use `plotxy` for the data inspection as follows.

```
> plotxy(vis="hh212_all.ms",xaxis="channel",yaxis="amp",datacolumn="data",
       iteration="baseline", antenna="",spw="1,5,11,14,15,19,20",field="3",
       averagemode="vector",width="1",timebin="all", crossscans=T,
       crossbls=False,stackspw=F, subplot=441,plotcolor="darkcyan",
       overplot=False,showflags=False,interactive=True,figfile="")
```

The above commands produce the Channel-Amplitude plot for each baseline in the specified spectral windows (spw) and the output is shown in Figure 2. The data are integrated in the time domain (`timebin="all"`). Data flagging can be done in plotxy GUI interface with the "Mark Regions"and "Flag" button. If you want to check the data for each baseline more precisely, please change to `subplot=111`. Because we set `iteration="baseline"`, the plot for the next baseline appears by pushing "Next" button, not the arrow button.

`plotxy` cannot plot the spectral windows with different channel numbers at once, therefore you need to plot the low- and high-resolution spw separately. `plotms` can handle the multi-resolution data and plot them with the x-axis of Frequency. If you set the x-axis Channel, all the spectral windows

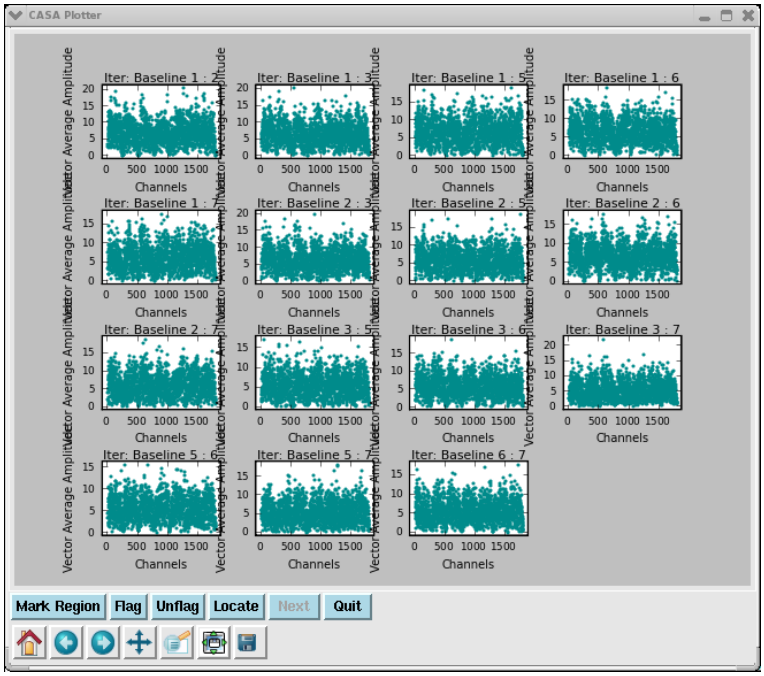


Figure 2: Channel-Amplitude plot of the field "3" (Bandpass calibrator) in high-resolution chunks for each baseline made with plotxy.

are overlaid, which would not be the one we want to see. Please note that execute `> clearstat()` to clear the status after you display the data with `plotxy`, otherwise the file is locked and `plotms` fails to display the data.

The sample script contains several `plotxy` commands to check the data for the field "3" (bandpass calibrator), but those do not cover the other field (source). You should take a look at the data for the other sources with either `plotxy` or `plotms`.

3.2 Flagging

In addition to the flagging with GUI, `flagdata` command provide the data flagging function with direct specifications for the bad data points. From the Line 184 in the sample script, we flag the edge channels of each spectral windows.

```
> flagdata(vis='hh212_all.ms', mode='manualflag',spw='1:0~9;246~255')
> flagdata(vis='hh212_all.ms', mode='manualflag',spw='5:0~9;246~255')
...
> flagdata(vis='hh212_all.ms', mode='manualflag',spw='0:0~3;60~63')
> flagdata(vis='hh212_all.ms', mode='manualflag',spw='2~4:0~3;60~63')
...
```

Here we flag 10 channels in each edge of the high-resolution spectral windows and 4 channels in the low-resolution windows. Maybe you want specify these with something like `spw='1,5:0~10;245~255'` for short, but this does not work as you want. CASA interprets this as "flag ALL the channels in spw 1 AND channel 0-10/245-255 in spw 5". Therefore here the flagging is executed for each spectral window separately. Of course you can make use of a Python for-loop here.

Figure 3 is the Time-Amplitude plot of the field 1,2 (target fields), and 4 (gain calibrator) in the baseline ANT2-ANT5. The scatter in the amplitude of the data taken in the beginning (06:47:0 - 07:32:0) and the end (14:57:0 - 15:20:0) of the observation is larger than the rest, therefore it is good

to flag them. This scatter is seen in the other baselines, so flag them by specifying the time range as follows.

```
> flagdata(vis='hh212_all.ms',mode='manualflag', timerange='06:47:0~07:32:0')
> flagdata(vis='hh212_all.ms',mode='manualflag',timerange='14:57:0~15:20:0')
```

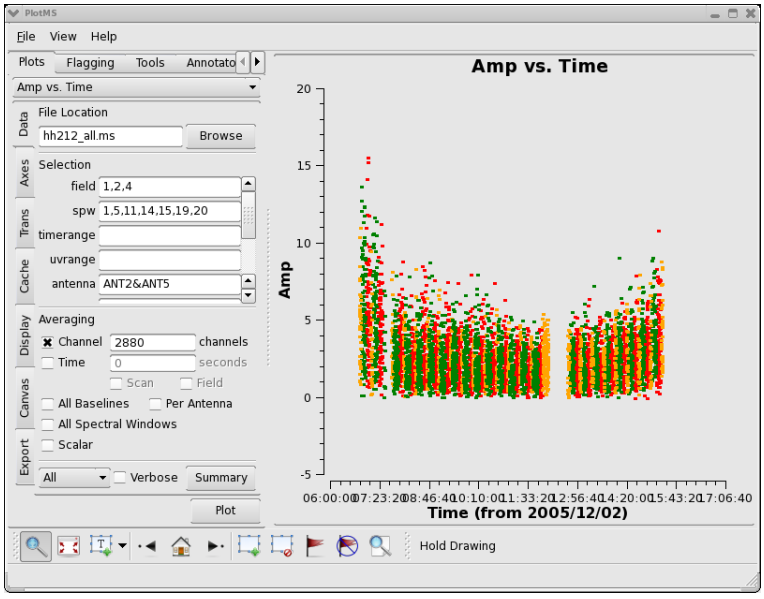


Figure 3: Time-Amplitude plot for the baseline ANT2-ANT5. The large scattering in 06:47:0-07:32:0 is seen in the other baselines. Color shows the difference of the fields

Also the field 4 (gain calibrator) was observed for the radio pointing during 11:56:40-12:06:40, we need to flag them.

```
> flagdata(vis='hh212_all.ms',mode='manualflag', timerange='11:56:40~12:06:40')
```

4 Bandpass Calibration

After flagging the bad data points, we move on to the Bandpass calibration. It would be good to check the bandpass with `plotxy` before the calibration. The following command displays the Channel-Phase plot of the low-resolution spectral windows of the field 3, the bandpass calibrator. You would need to check the bandpass of the high-resolution spectral windows, and the Channel-Amplitude bandpass.

```
> plotxy(vis="hh212_all.ms",xaxis="channel",yaxis="phase",datacolumn="data",
       iteration="baseline",
       antenna="",spw="0,2~4,6~10,12~13,16~18,21~23",field="3",
       averagemode="vector",width="1",timebin="all",
       crossscans=T,crossbls=False,stackspw=F,
       subplot=441,plotcolor="darkcyan",
       overplot=False,showflags=False,interactive=True,figfile="")
```

`bandpass` is the task for the antenna-based bandpass calibration. The output is a calibration table, so the original data has not been modified here.

```
> bandpass(vis='hh212_all.ms', caltable='hh212_all.ms.bpoly4', field='3',
          spw='0~23', bandtype='BPOLY', maskedge=5, solint='inf',refant='2',
          degamp=4, degphase=4)
```

Here we fit the data with 4th order polynominals in amplitude and phase by setting `bandtype='BPOLY'` and `degamp=4, degphase=4`. The 5% of the channels in each band edge are not used for the fitting by setting `maskedge=5`, which is default. If you want to derive the bandpass solution in each channel (without fitting), try `bandtype='B'`. `solint='inf'` ("solution interval is infinite") means that the calibration table is generated by the data integrated in time. The output is a calibration table, `hh212_all.ms.bpoly4`.

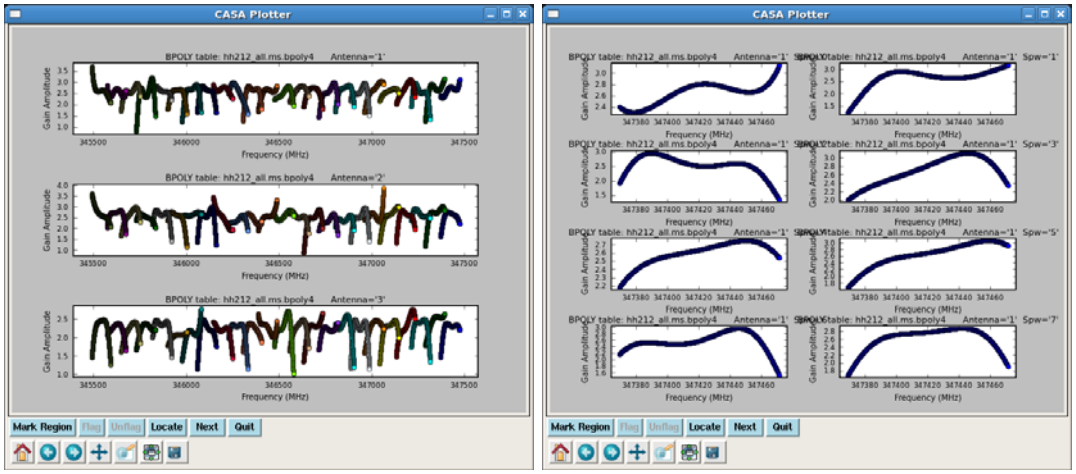
The calibration table can be checked with `plotcal`. The output is shown in Figure 4a.

```
> plotcal(caltable="hh212_all.ms.bpoly4",xaxis="freq",yaxis="amp",
        field="3",antenna="",spw="",timerange="",subplot=311,
        overplot=False,clearpanel="Auto",iteration="antenna",
        plotsymbol="o",plotcolor="blue",showgui=True,figfile="")
```

The above command makes a Frequency-Amplitude plot of the calibration table. Plotting all the data points takes time. If you want to check the results for each antenna and each spectral window, you can set `iteration="antenna,spw"` (Figure.4b).

```
> plotcal(caltable="hh212_all.ms.bpoly4",xaxis="freq",yaxis="amp",
         field="3",antenna="",spw="",timerange="",subplot=421,
         overplot=False,clearpanel="Auto",iteration="antenna,spw",
         plotsymbol="o",plotcolor="blue",showgui=True,figfile="")
```

There seems several bugs in `plotcal`. When setting the iteration on "spw", note that the frequency labeling is incorrect on the plotcal. The `xaxis="chan"` is ignored and plotted as the xaxis of frequency. In addition, if you specify the spectral window with `spw=""`, plotcal only plots part of the specified spectral window.



(a) for all spectral windows

(b) for each spectral window

Figure 4: Output of plotcal.

A task `applycal` applies the calibration table to the data.

```
> applycal(vis='hh212_all.ms',spw='0~23', field='3',
        gaintable='hh212_all.ms.bpoly4',
        spwmap=[],gainfield='3')
```

Here you need to give the name of the calibration table and the way how the table should be applied. The parameter `spwmap` controls this. Setting `spwmap=[]` tells that the calibration table is spw-dependent, which is appropriate in the bandpass calibration, while `spwmap=[0]` means that the solution is independent of the spectral windows, such as in the time-dependent gain calibration. You also need to give the field ID (source name) of the calibrator with the parameter `gainfield`.

The result of the application of the bandpass calibration can be checked with `plotxy` and `plotms`. Here is the example of the plotxy command. Note that `datacolumn="corrected"`, instead of "data" for the calibrated data points. The raw data always stay in the "data" column and `applycal` only add or modify the "corrected" column.

```
> plotxy(vis="hh212_all.ms",xaxis="channel",yaxis="amp", datacolumn="corrected",
        iteration="baseline", antenna='',spw="1,5,11,14,15,19,20",field="3",
        averagemode="vector",width="1",timebin="all",crossscans=T,
        crossbls=False,stackspw=False,extendflag=F, extendchan="",extendspw="",
        extendant="",extendtime="",plotcolor='darkcyan',subplot=411,
        multicolor="none",figfile="")
```

Figure.5 is the comparison between the Channel-Amplitude plot for field '3' before and after the application of the bandpass calibration. The figure clearly shows that the bandpass becomes flat after the calibration. Note that the amplitude is scaled to 1 Jy after applying the calibration. This scaling is corrected later. The above command is for the Channel-Amplitude plot for the high-resolution spectral windows, so you should check the Channel-Phase plot and both for the low-resolution spectral windows, as described in the sample script.

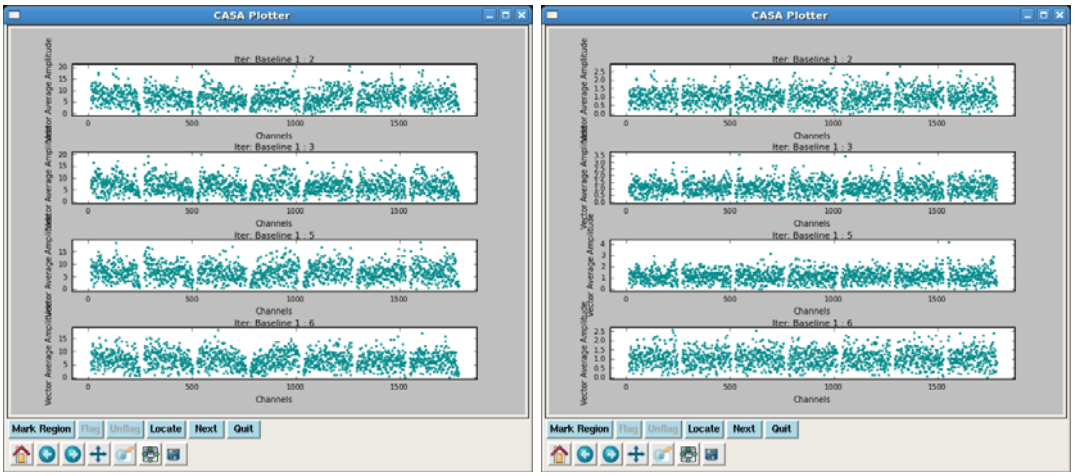


Figure 5: Before (*left*) and after (*right*) the amplitude bandpass calibration for field '3' in the high-resolution spectral windows.

If you would like to try the baseline-based calibration, you can use `blcal` task as follows.

```
> blcal(vis='hh212_all.ms', caltable='hh212_all.ms.blcal', field='3', spw='0~23',
        solint='inf', combine='scan', calmode='ap',freqdep=True)
```

The parameter `freqdep` controls whether the calibration in frequency-dependent (`True` for bandpass) or independent (`False` for time-dependent gain calibration). In the sample script, we don't do the baseline-based calibration.

5 Flux and Gain Calibration

5.1 Flux calibration

Originally Uranus was observed as an absolute flux calibrator, however, CASA 3.0.2 can not handle the resolved planets as a flux calibrator properly. Therefore, we use the bandpass calibrator (a quasar 3c454.3, field '3' in the dataset) as a flux calibrator. The flux of this source is 9.0 Jy (9.02 ± 0.47 Jy on 22 Nov 2005 and 8.37 ± 0.91 Jy on 10 Jan 2006), based on the SMA Calibrator List[4]. Although this list is not intended for this kind of use, it is OK for us to use the flux in the list in the data reduction tutorial (i.e. not for "real" science). Let's wait for the update of CASA.

The flux calibration can be done with `setjy`, in which you just tell the flux of the calibrator.

```
> setjy(vis='hh212_all.ms', field='3', spw='0~23', fluxdensity=[9.0,0.,0.,0.])
```

You can specify the flux density for four stokes parameters $[I, Q, U, V]$, if you have. If not, just set the I value. This task only put the flux of the calibrator and that flux is not applied to the actual data value. We use a task `fluxscale` to scale the flux of the other sources after the gain calibration.

5.2 Gain calibration

In order to check the data of the gain calibrator, first you need to apply the bandpass calibration to the calibrator (field '4').

```
> applycal(vis='hh212_all.ms',spw='0~23', field='4',
      gaintable='hh212_all.ms.bpoly4', spwmap=[], gainfield='3')
```

Note the field selection `field='4'`. After `applycal` you should check the result with `plotxy` or `plotms`. The comparison between before and after gain calibration for the field 4 is shown in Figure.6. You can see the bandpass is well calibrated.

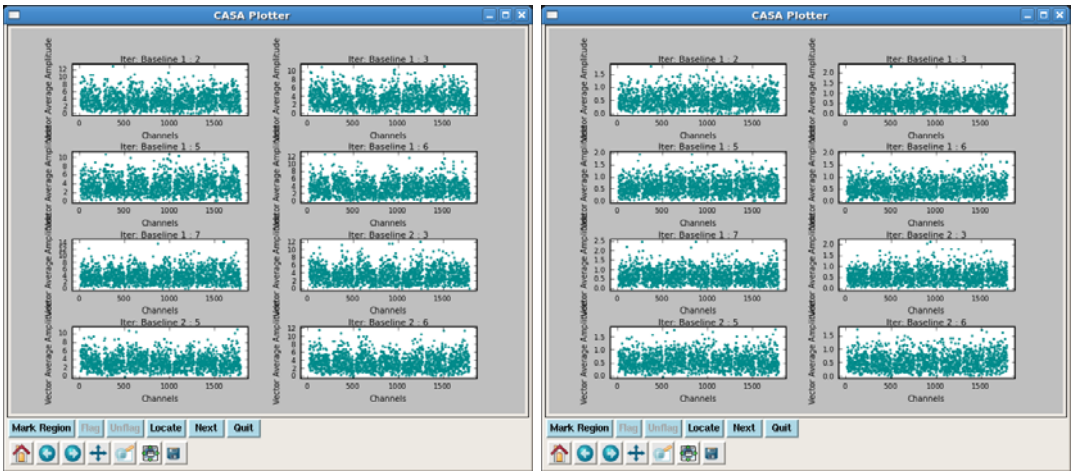


Figure 6: Before (*left*) and after (*right*) the amplitude bandpass calibration for field '4' in the high-resolution spectral windows.

Then, check the time dependence of the data as follows.

[4] http://sma1.sma.hawaii.edu/callist/callist.html

```
> plotxy(vis="hh212_all.ms",xaxis="time",yaxis="phase",datacolumn="corrected",
       iteration="baseline", antenna="",spw="1,5,11,14,15,19,20",field="4",
       averagemode="vector",width="allspw",timebin="0", subplot=421,
       plotsymbol=".",multicolor="none",plotcolor="darkcyan",
       overplot=False,showflags=False,interactive=True,figfile="")
```

This is a command for making the Time-Phase plots for field '4' in the high-resolution spectral windows. Here all the channels are summed up by `width="allspw"`.

The gain solution can be derived with a task `gaincal`. This is antenna-based, and if you want baseline-based calibration, use `blcal` with the parameter `freqdep=True`.

```
> gaincal(vis='hh212_all.ms', caltable='hh212_all.ms.apcal',field='3,4',
       spw='0~23', gaintype='G', minsnr=2.0,refant='2', calmode='ap',
       solint='180s', combine='spw',gaintable='hh212_all.ms.bpoly4',spwmap=[])
```

Here we include field '3' as well as field '4' in order to keep the relative flux for further absolute flux scaling. The parameter `gaintype='G'` indicates polarization-dependent gain. If you want polarization-independent gain, use `gaintype='T'`. (Anyway here we don't care the polarization...) If the signal-to-noise ratio is too low even when you set the infinite solution interval, you can try `gaintype='GSPLINE'`. With this option the data are fitted by a spline function, however, it cannot be used in the `fluxscale` task. See the cookbook. Here we set the solution interval of 180 seconds with the parameter `solint='180s'`, which means that we integrate the data points in 180 seconds to get one gain solution.

The results can be checked with `plotcal` (Figure. 7)

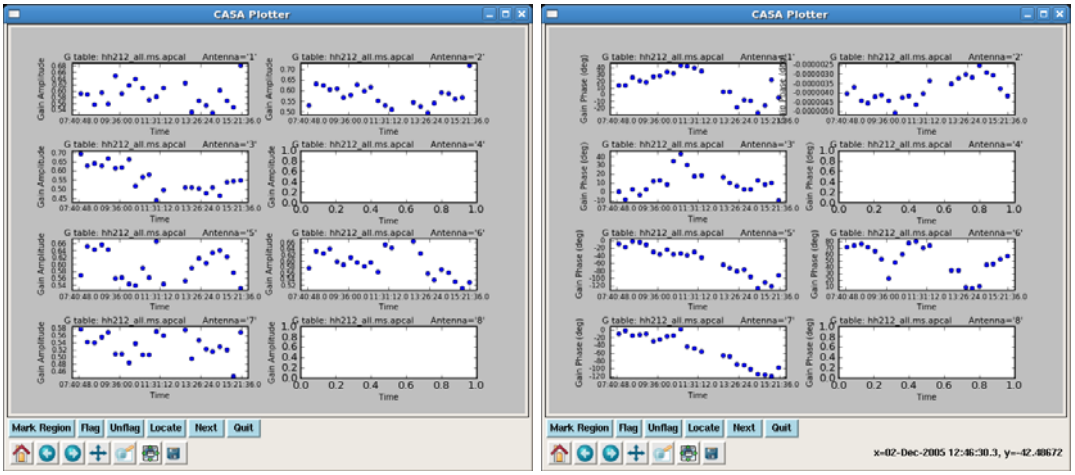


Figure 7: Plots of the gain calibration table. Time-Amplitude (*left*) and Time-Phase (*right*)

After deriving the gain solution, we scale the flux of the sources based on the flux calibrator specified with `setjy`. The output (`hh212_all.ms.fluxcal`) is a flux-scaled gain table.

```
> fluxscale(vis='hh212_all.ms',caltable='hh212_all.ms.apcal',
       fluxtable='hh212_all.ms.fluxcal',reference='3')
```

5.3 Applying all the calibrations

Now we can apply all the calibrations to the fields with `applycal`. Be careful on the parameter `spwmap` and `gainfield` to apply the solution in appropriate way. `spwmap=[]` means that the solution depends

on the spectral window (e.g. bandpass), while `spwmap=[0]` indicates the solution is independent of the spectral window (e.g. time-dependent gain calibration).

At first, we apply the calibration to the bandpass calibrator and the gain calibrator to check whether the calibration is good or not.

```
> applycal(vis='hh212_all.ms',spw='0~23', field='3',
        gaintable=['hh212_all.ms.fluxcal','hh212_all.ms.bpoly4'],
        spwmap=[[0],[]],gainfield=['3','3'])

> applycal(vis='hh212_all.ms',spw='0~23', field='4',
        gaintable=['hh212_all.ms.fluxcal','hh212_all.ms.bpoly4'],
        spwmap=[[0],[]],gainfield=['4','3'])
```

To check the result, we can use `plotxy` and `plotms`.

Then, apply the calibrations to the target fields.

```
> applycal(vis='hh212_all.ms',spw='0~23', field='1,2',
          gaintable=['hh212_all.ms.fluxcal',
          'hh212_all.ms.bpoly4'],spwmap=[[0],[]],
          gainfield=['3,4','3'])
```

6 Split the data

Now we split off the science target data from the dataset.

```
> split(vis="hh212_all.ms", outputvis="hh212_targets.ms", field="1,2")
```

With `split`, the "corrected" data column in the input MS file is copied to the "data" column. In addition, the field ID numbers are reset and now the field 0 and 1 are the IDs of the science targets. You can check the result of `split` with `listobs` and `plotms`.

```
> listobs(vis="hh212_targets.ms")

> plotms(vis='hh212_targets.ms',xaxis="frequency",yaxis="amp",field='0',
       avgtime='1e7',avgscan=True,avgbaseline=True)
```

The left panel of Figure 8 is the Frequency-Amplitude plot (or, spectrum) of the field 0 averaged over all the baselines. The parameter `avgtime='1e7'` means the integration over the whole observation (here just set a huge number to do this). You can see the two molecular lines (CO $J = 3 - 2$ at $\sim$ 345.8 GHz and SiO $J = 8 - 7$ at $\sim$ 347.3 GHz) in this plot.

We can plot with the x-axis of velocity. However, `plotms` cannot handle the manual setting of the reference frequency. We can do this with `plotxy`, and hopefully `plotms` will support these functions in the future release. The spectrum drawn with the following command, plotting the spectral windows 19 and 20, is shown in the right panel of Figure 8.

```
> plotxy(vis="hh212_targets.ms",xaxis="velocity",yaxis="amp",
         datacolumn="data",iteration="",selectdata=True,
         antenna='',spw="19,20",field="0", timerange="",
         averagemode="vector", restfreq='345.79599GHz',
         width="1",timebin="all",crossscans=T,crossbls=T,
         stackspw=False,subplot=111,interactive=T,figfile="")
```

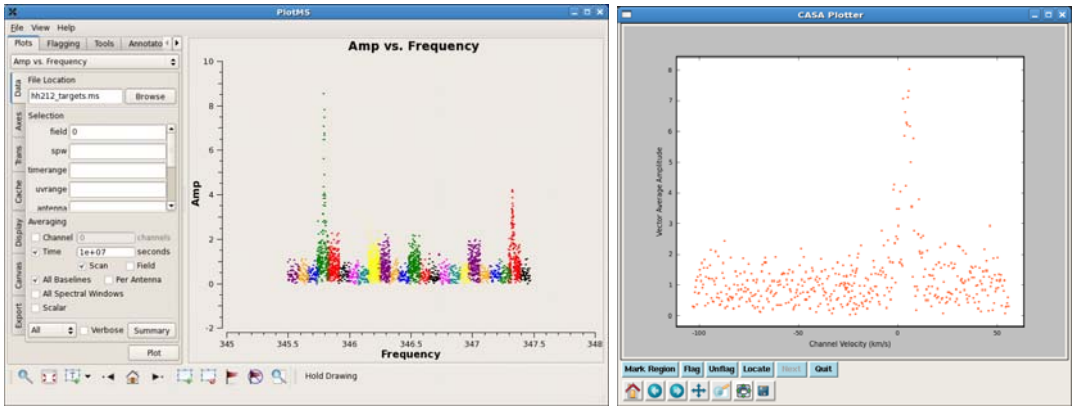


Figure 8: Spectra plot with plotms for whole bandwidth (*left*) and the spectrum of CO $J = 3 - 2$ plotted with plotxy (*right*).

7 Continuum Subtraction

The continuum emission can be estimated from the line-free channels. In order to identify the line-free spectral window, use `plotxy`.

```
> plotxy(vis="hh212_targets.ms",xaxis="channel",yaxis="amp",
         datacolumn="data",iteration="",selectdata=True,
         antenna='',spw="0,2~4,6~10,12~13,16~18,21~23",
         field="0",timerange="",averagemode="vector",
         width="1",timebin="all",crossscans=T,crossbls=T,
         stackspw=False,subplot=111,interactive=T,figfile="")
```

Here we plot the low-resolution spectral windows and there are no lines. You can do the same thing for the high-resolution spectral windows and will find the lines in spw 1 and spw 20 as shown in Figure 8.

After identifying the line-free spectral windows, we can subtract the continuum. However, the current version of `uvcontsub2` changes the "model" and "corrected" column in the MS. So, before running `uvcontsub2`, please duplicate the data for the back up. The Linux commands can be run with `os.system( )` on CASA.

```
> os.system('cp -r hh212_targets.ms hh212_targets_line.ms')
```

Now we run the task `uvcontsub2` for subtracting continuum emission estimated from the line-free spectral windows.

```
> uvcontsub2(vis='hh212_targets_line.ms',fitspw='0,2~19,21~23',
         solint='int',combine='spw',want_cont=True)
```

For the moment `uvcontsub2` supports only the first-order fit and subtraction. Two MS files are created by `uvcontsub`, `hh212_targets_line.ms.cont` contains the estimated continuum and `hh212_targets_line.ms.contsub` contains the continuum-subtracted line data.

8 Imaging

8.1 Continuum Imaging of the Calibrator

Here we make a image of the point-like quasar (field 4, the gain calibrator) to check the calibration. All the imaging procedures (inverse-FT, CLEANing, and deconvolution) can be done in a task `clean`.

```
> clean(vis='hh212_all.ms',imagename='hh212_f4_cont_dirty',
      field='4',spw='',mode='mfs',niter=0,gain=0.1,threshold='40mJy',
      psfmode='clark',imagermode='csclean',
      interactive=F,imsize=256,cell="0.35arcsec",
      phasecenter='',pbcor=F,minpb=0.2)
```

At first, we make a "dirty" image with setting the parameter `niter=0`. `mode='mfs'` indicates the multi-frequency synthesis to make a single-channel image (because here we are making a continuum image). `psfmode='clark'` sets the algorithm to make the synthesized beam as "clark", which is the default choice. The other choice is "hogbom" and this is the classical image-domain CLEAN. The parameter `imagermode='csclean'` sets the mode of the CLEAN on the visibility plane. We can set the image size and pixel size with `imsize` and `cell`. Here we switch off the parameters `interactive` and `pbcor` for the interactive clean and primary beam correction, respectively.

The task `clean` creates five datasets with the names with `.flux`, `.image`, `.model`, `psf`, and `residual` for the image with primary beam correction, the resulting image, the CLEAN model, the PSF, and the residual of the CLEAN process, respectively. We can browse these data with the task `viewer`.

```
> viewer(infile='hh212_f4_cont_dirty.image')
```

You can measure the noise level on the viewer. Select a region with rectangular/polygon box selection button and double-click it, then the statistics are shown in the terminal and a new window. Measuring at the emission-free region, the rms is $\sim$ 20 mJy/beam. We use 2σ value (40mJy/beam) as the threshold in CLEAN in the following.

After checking the dirty image, let's move on to CLEAN with the same task `clean` and modifying several parameters.

```
> clean(vis='hh212_all.ms',imagename='hh212_f4_cont_clean',
      field='4',spw='',mode='mfs',niter=50000,gain=0.1,threshold='40mJy',
      psfmode='clark',imagermode='csclean',
      interactive=True,imsize=256,cell="0.35arcsec",
      phasecenter="",pbcor=F,minpb=0.2)
```

Here we set the number of iteration of 50000 and turn on the iterative cleaning. In running this task, after the first iteration (making dirty image), a GUI window appears as shown in Figure 9. With this window you can make a clean box with a rectangular selection or a polygon selection (see the caption of Figure 9). After editing the clean box, you can push the blue arrow or green circular arrow in the "Next Action" section on the GUI window. Pushing the former arrow make CLEAN continue till the end when the residual falls below the threshold you set. On the other hand, pushing the latter arrow make CLEAN proceed more cycles as you specified, and stops. The result at that point is shown again if the residual is still above the threshold, and you can change the clean box, continue the clean, or finish it with the red stop button. The resulting image and flux can be checked with the task `viewer`.

```
> viewer(infile='hh212_f4_cont_clean.image')
```

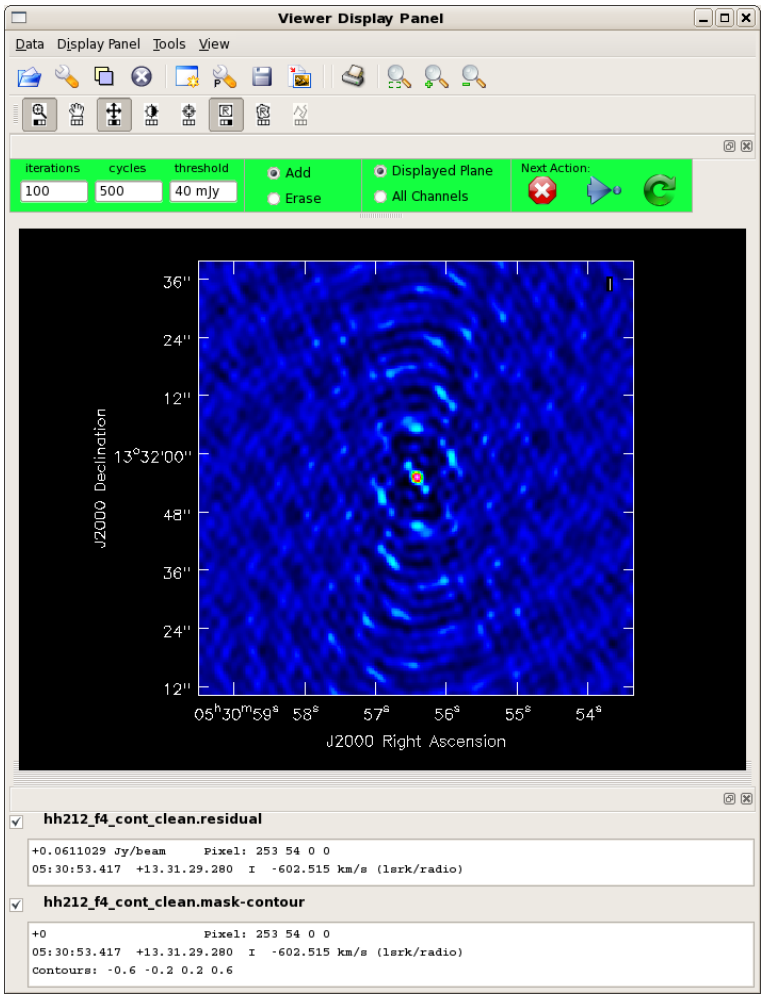


Figure 9: GUI window for the interactive clean. We can make a clean box with this window. Click the rectangular selection button ("R" in rectangular box) or polygon selection button ("R" in irregular box) to select the shape of your clean box. Then draw the clean box on the image. Double-click the drawn box to finish editing the clean box and the box outline turns white.

8.2 Continuum Imaging of the Targets

Here we make a continuum image of the targets with 2-field mosaic.

```
> clean(vis='hh212_targets_line.ms.cont',imagename='hh212_cont_dirty',
      field='',spw='',
      mode='mfs',niter=0,gain=0.1, threshold='40mJy',
      psfmode='clark',imagermode='mosaic', scaletype='SAULT',
      ftmachine='mosaic',
      interactive=F, imsize=256,cell="0.35arcsec",
      phasecenter="J2000 5h43m51.404 -1d02m53.10",
      pbcor=F,minpb=0.2)

> viewer('hh212_cont_dirty.image')
```

The differences from the calibrator imaging are related to the mosaicing (`imagermode='mosaic'`). See the CASA cookbook for details. The noise level is ~ 20 mJy/beam measured at the emission-free region of the dirty map. We use 40 mJy/beam (2σ) as the threshold of the CLEAN process.

For cleaning, you can manually set the pixel values of the bottom-left/top-right corner of the clean box with the parameter `mask` as follows.

```
> clean(vis='hh212_targets_line.ms.cont',imagename='hh212_cont',
      field='',spw='',
      mode='mfs',niter=20000,gain=0.05,threshold='40mJy',
      psfmode='clark',imagermode='mosaic',scaletype='SAULT',
      ftmachine='mosaic',
      interactive=T,imsize=256,cell="0.35arcsec",
      phasecenter="J2000 5h43m51.404 -1d02m53.10",
      mask=[122,122,133,133], pbcor=F,minpb=0.2)

> viewer('hh212_cont.image')
```

You can see a compact peak at the center of the map.

You may make a "continuum" image from the continuum-subtracted data for examining whether the continuum subtraction worked well or not.

```
> clean(vis='hh212_targets_line.ms.contsub',imagename='hh212_contsub',
      field='',spw='0,2~19,21~23', mode='mfs',niter=0,gain=0.1,threshold='40mJy',
      psfmode='clark',imagermode='mosaic',scaletype='SAULT', ftmachine='mosaic',
      interactive=F,imsize=256,cell="0.35arcsec",
      phasecenter="J2000 5h43m51.404 -1d02m53.10", pbcor=F,minpb=0.2)

> viewer('hh212_contsub.image')
```

Browsing the dirty image with `viewer`, the peak flux is about 18 mJy which corresponds to $\sim 1\sigma$ noise level. The continuum emission was successfully subtracted.

8.3 Spectral Imaging

Next, we make images of the SiO $J = 8 - 7$ line. Before moving on to the imaging process, it is good to inspect the spectrum to have the velocity range to be imaged. The `plotxy` output is shown in Figure 10

```
> plotxy(vis="hh212_targets_line.ms.contsub",xaxis="velocity",yaxis="amp",
          datacolumn="data",iteration="",selectdata=True, antenna='',spw="1",
          field="0",timerange="",averagemode="vector",
          restfreq='347.33082GHz',
          width="1",timebin="all",crossscans=T,crossbls=T,
          stackspw=False,subplot=111,interactive=T,figfile="")
```

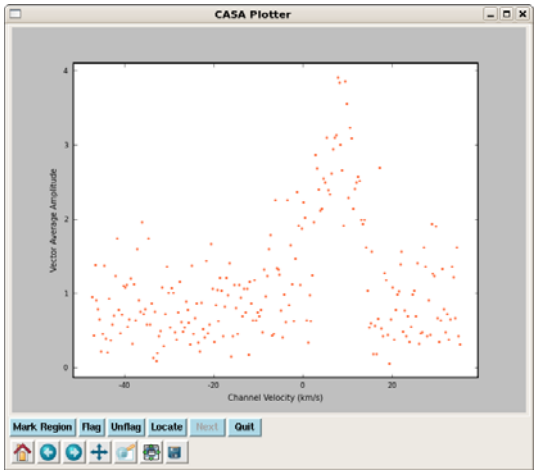


Figure 10: The spectrum of the SiO $J = 8 - 7$ line for the field 0.

Judging from the spectrum, it is good to make images in the velocity range from -20 km/s to 30 km/s. The command for creating a dirty image is as follows.

```
> clean(vis='hh212_targets_line.ms.contsub',imagename='hh212_SiOline_dirty',
      field='',spw='',
      mode='velocity',start='-20km/s',nchan=50,width='1.0km/s',
      interpolation='linear', outframe='LSRK',
      niter=0,gain=0.1,threshold='0.4Jy',
      psfmode='clark',imagermode='mosaic',scaletype='SAULT',
      ftmachine='mosaic',restfreq='347.33082GHz',
      interactive=F,
      imsize=256,cell="0.35arcsec",
      phasecenter="J2000 5h43m51.404 -1d02m53.10",
      pbcor=F,minpb=0.15)
```

We set the parameter `mode='velocity'` for making a 3D cube data and set the starting velocity, number of channels, and channel width with the parameter `start`, `nchan`, and `width`. We need to set the rest frequency of the molecular line with `restfreq`. A dirty map is generated by this command and you can check it with `viewer` (Figure 11). `viewer` provides the animation function useful for inspecting the 3D cube data.

After checking the dirty image and having an idea of the clean box, we move on to CLEAN.

```
> clean(vis='hh212_targets_line.ms.contsub',imagename='hh212_SiOline',
      field='',spw='',
      mode='velocity',start='-20km/s',nchan=50,width='1.0km/s',
      interpolation='linear', outframe='LSRK',
      niter=10000,gain=0.05,threshold='1.0Jy',
      psfmode='clark',imagermode='mosaic',scaletype='SAULT',
      ftmachine='mosaic',restfreq='347.33082GHz',
      interactive=T,
```

```
imsize=256,cell="0.35arcsec",mask=[100,70,155,185],
phasecenter="J2000 5h43m51.404 -1d02m53.10",
pbcor=F,minpb=0.15)
```

Here we set a rectangular clean box by inputting the blc/trc pixel, or you can set the clean region with an external mask file. To make a mask file, first select the region you like with the rectangular or polygon selection tool on `viewer`, then double-click the region. A window for the statistics appears but ignore it. Click the "Tools"-"Region in File"menu of the viewer window. You need to specify the channels in which the mask is effective, then save it. The resulting mask file ("XXX.rgn") can be used in the CLEANing procedure by setting `mask='XXX.rgn'`.

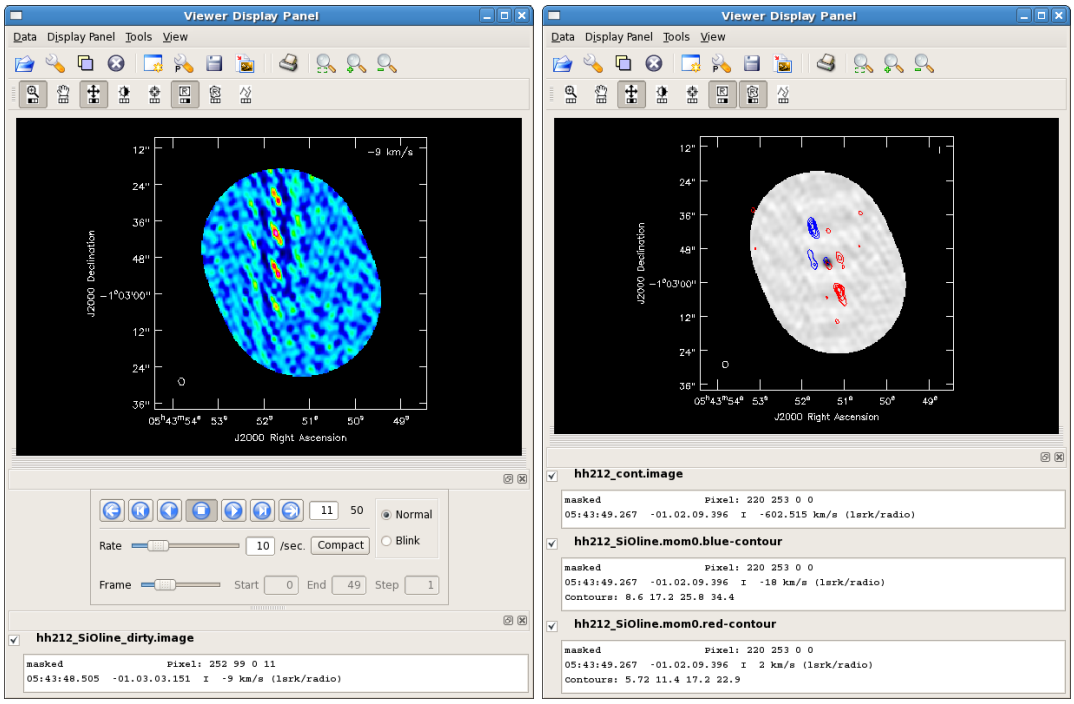


Figure 11: Dirty map of SiO $J = 8-7$ (*left*) and CLEANed SiO (contour) overlaid on the continuum map (gray scale) plotted with the viewer (*right*).

Next, we make moment 0 maps of SiO. Since the systemic velocity is 1.7 km/s, simply divide the data with that velocity.

```
> immoments(imagename='hh212_SiOline.image',moments=[0],axis='spectral',
          chans='2~21',outfile='hh212_SiOline.mom0.blue')
> immoments(imagename='hh212_SiOline.image',moments=[0],axis='spectral',
          chans='22~38',outfile='hh212_SiOline.mom0.red')
```

The 0th moment maps can be overlaid on the continuum image with `viewer` (Figure 11). There are two red/blueshifted blobs just next (east-west) to the continuum peak. These blobs are probably made by the side lobes. Please try to set the clean box with polygon to remove them.