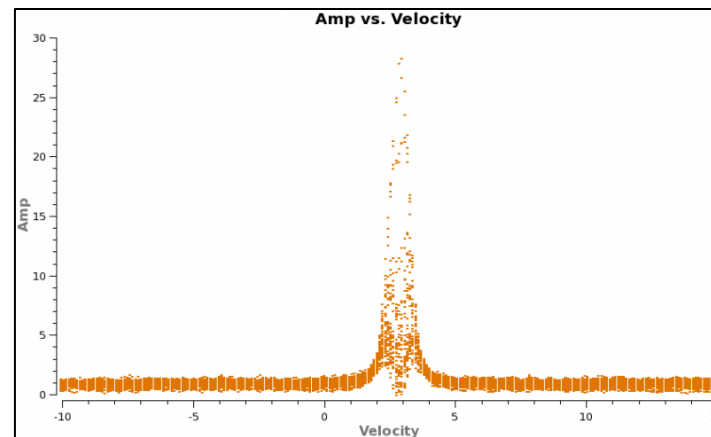


TW Hya CASA

Spectral Line Reduction Tutorial

Day 2, Monday September 7th 16:15



CO (3-2) spectrum of TW Hya

Tutors:

Katharine Johnston

Andy Biggs, Sandra Etoke, Liz Humphreys,
John McKean, Rosita Paladino, Anita Richards and
Lorant Sjouwerman

Data and Script

Data that we will start from is:
TWHYA_averaged_wvrtsys.tgz (1.4GB)
(this is different from full dataset!)

Will follow script given here:

www.mpia.de/~johnston/ERIS/TWHya_reduction_script.txt

Extended version of reduction can be found
on the CASA guides website:

<https://casaguides.nrao.edu>

**You can copy-paste commands from script
into CASA as we go along**

Making a script

Don't do now (we will copy and paste this time),
but in future:

The script (preferably minus the plotting commands) can
also be run using the **execfile** command to produce the
calibrated dataset, e.g.:

```
CASA <2>: execfile('myscript.py')
```

Copy-paste script: www.mpia.de/~johnston/ERIS/TWHya_reduction_script.txt

First off, **create a reduction directory**
and un-tar this file in it:

TWHYA_averaged_wvrtsys.tgz

To untar, you can use:

```
tar -zxvf file_to_untar
```

Copy-paste script: www.mpia.de/~johnston/ERIS/TWHya_reduction_script.txt

Listobs and for loops (CASA=python)

Run listobs for all three datasets using a python list and for loop:

```
splitdata=[ 'X3c1_wvrtsys.ms' , 'X5d8_wvrtsys.ms' ,  
'X7ef_wvrtsys.ms' ]  
for vis in splitdata:  
    os.system('rm '+vis+'.listobs')  
    listobs(vis, verbose=T,  
            listfile=vis+'.listobs')
```

If you are lacking disk space or speed: from now on only use one dataset (there are 3), e.g.

```
splitdata=[ 'X3c1_wvrtsys.ms' ]
```

Fields...

From listobs you can see there are five fields:

Fields: 5

ID	Code	Name	RA	Decl	Epoch	SrcId	nRows
0	none	3c279	12:56:11.166571	-05.47.21.52471	J2000	0	360
1	none	Titan	12:49:26.512450	-02.22.27.28171	J2000	1	360
2	none	TW Hya	11:01:51.844982	-34.42.17.16089	J2000	2	14400
3	none	J1147-382=QSO	11:47:01.381512	-38.12.11.11788	J2000	3	1440
4	none	J1037-295=QSO	10:37:16.089888	-29.34.02.98884	J2000	4	5040

But which fields are calibrators and which is the target? (look at scan intents!)

Fields...

From listobs you can see there are five fields:

Fields: 5

ID	Code	Name	RA	Decl	Epoch	SrcId	nRows
0	none	3c279	12:56:11.166571	-05.47.21.52471	J2000	0	360
1	none	Titan	12:49:26.512450	-02.22.27.28171	J2000	1	360
2	none	TW Hya	11:01:51.844982	-34.42.17.16089	J2000	2	14400
3	none	J1147-382=QSO	11:47:01.381512	-38.12.11.11788	J2000	3	1440
4	none	J1037-295=QSO	10:37:16.089888	-29.34.02.98884	J2000	4	5040

0 = 3c279 = Bandpass calibrator

1 = Titan = Flux calibrator

2 = TW Hya = Target

3 = J1147-382 = secondary phase calibrator (we won't use)

4 = J1037-295 = primary phase calibrator

Data flagging using flagdata task

```
splitdata=['X3c1_wvrtsys.ms', 'X5d8_wvrtsys.ms',  
          'X7ef_wvrtsys.ms']  
  
for vis in splitdata:  
    flagdata(vis=vis, mode='manual', autocorr=True,  
            action = 'apply', flagbackup=T)  
    flagdata(vis=vis, mode='shadow', tolerance=12.0,  
            action='apply', flagbackup=F)  
    flagdata(vis=vis, mode='manual', antenna='DV04',  
            action='apply', flagbackup=F)
```

Question: What are these three flagdata commands doing?

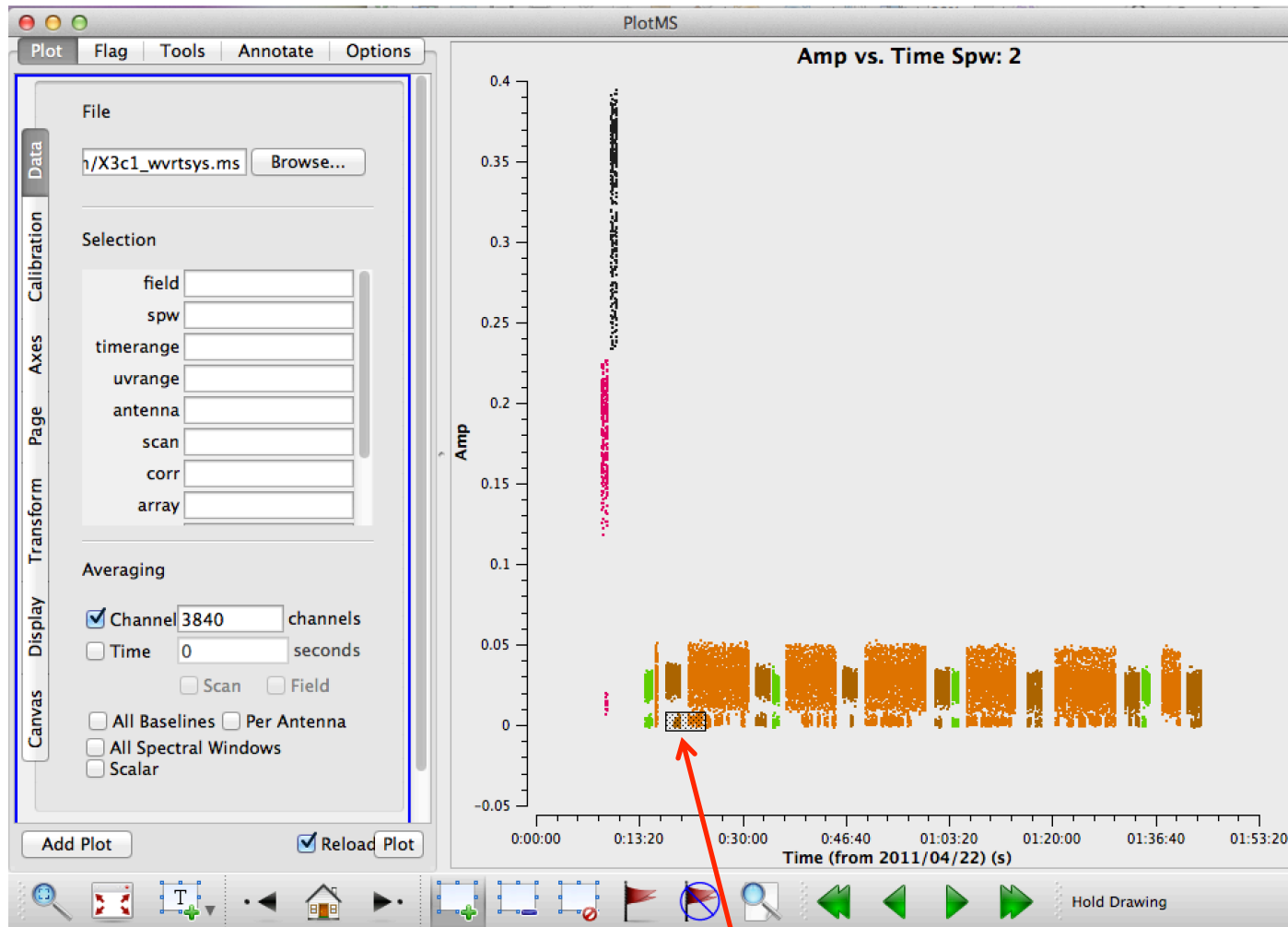
Data inspection in time using plotms

```
plotms(vis='X3c1_wvrtsys.ms', spw='',  
       xaxis='time', yaxis='amp', field='',  
       avgchannel='1280', coloraxis='field',  
       iteraxis='spw')
```

To do:

- Use the mark region and locate buttons to determine where the bad data in spw=2 comes from
- Run the plotms command, removing the bad antenna (use antenna='!*ant_name*')
- Plot amplitude vs. time for the two other datasets

Data inspection using plotms



Bad data in spw=2!

Flagging the bad data we found...

```
splitdata=[ 'X3c1_wvrtsys.ms' , 'X5d8_wvrtsys.ms' ,  
            'X7ef_wvrtsys.ms' ]
```

```
for vis in splitdata:
```

```
    flagdata(vis=vis, mode='manual' ,  
            action= 'apply' , flagbackup=T,  
            spw='2' , antenna='PM03' ,  
            correlation='YY' )
```

Data inspection with frequency using plotms

Plot the phase vs. frequency for the bandpass calibrator (3c279, field=0) and one antenna (here DV06):

```
plotms(vis='X3c1_wvrtsys.ms', spw='',  
       axis='frequency', yaxis='phase',  
       field='0', antenna='DV06',  
       avgtime='1e8', avgscan=T,  
       coloraxis='baseline', iteraxis='spw',  
       xselfscale=T)
```

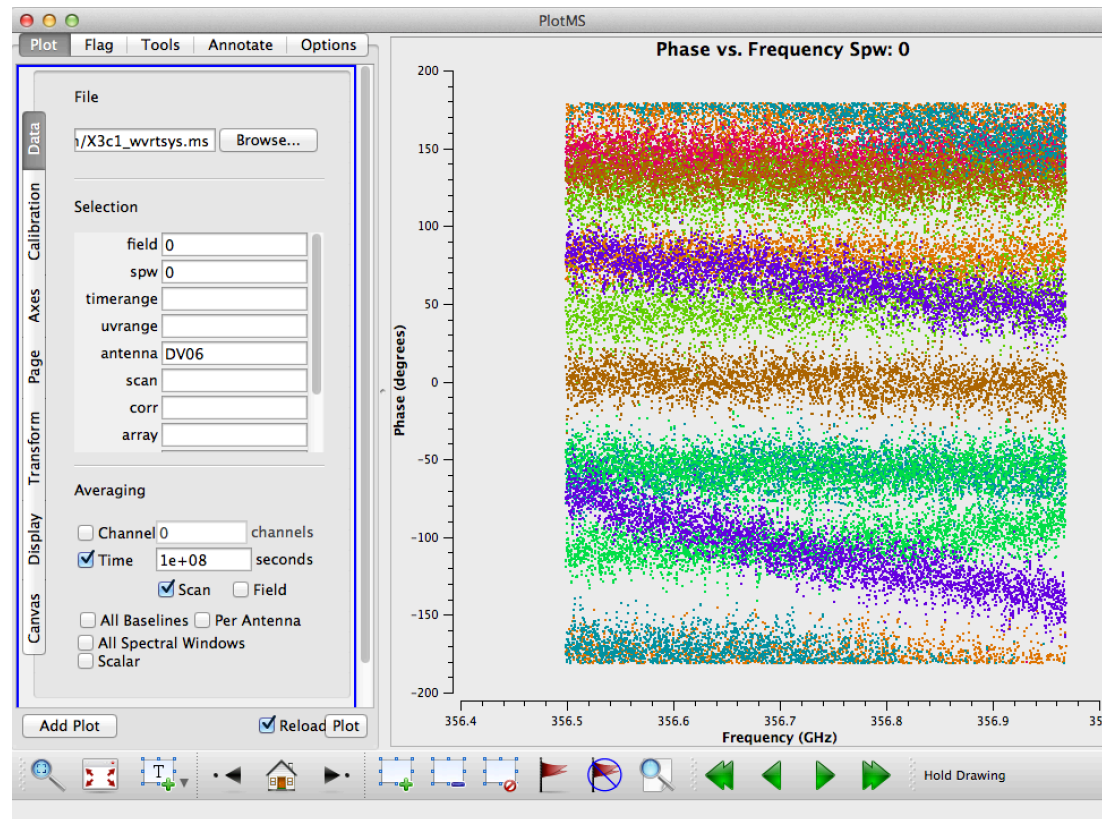
Question:

Are there large delay errors (slopes with frequency) in the data?

Data inspection with frequency using plotms

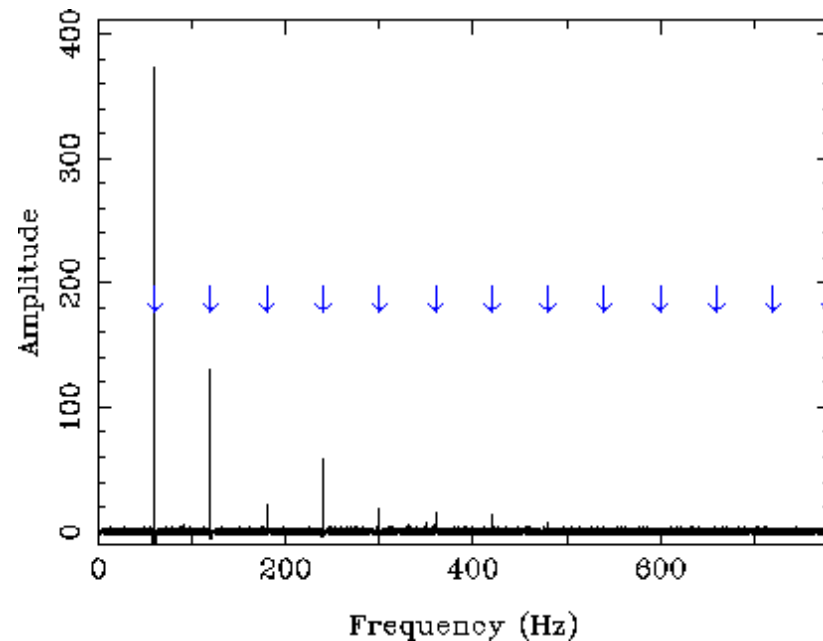
Question: Are there large delay or baseline errors (large variations with frequency) in the data?

Answer: nope, looks ok. Slopes are manageable.



“Birdies”

- Very narrow weak spectral features that are internally generated in the system
- Check by looking at amplitude vs. frequency for high signal-to-noise sources like the bandpass calibrator
- Birdies should be **present in all sources** (otherwise probably is real line emission!)



“Birdies”

Here plot for TW Hya and the phase calibrators, (field= 2,3,4) for one of the datasets:

```
plotms(vis='X3c1_wvrtsys.ms', spw='',  
       xaxis='channel', yaxis='amp',  
       field='2,3,4', avgtime='1e8',  
       avgscan=T, coloraxis='field',  
       iteraxis='spw', xselfscale=T,  
       yselfscale=T)
```

To determine the channels for flagging, zoom in and use the mark region and locate buttons.

To do: plot the other two datasets to check whether the birdies are in the same place.

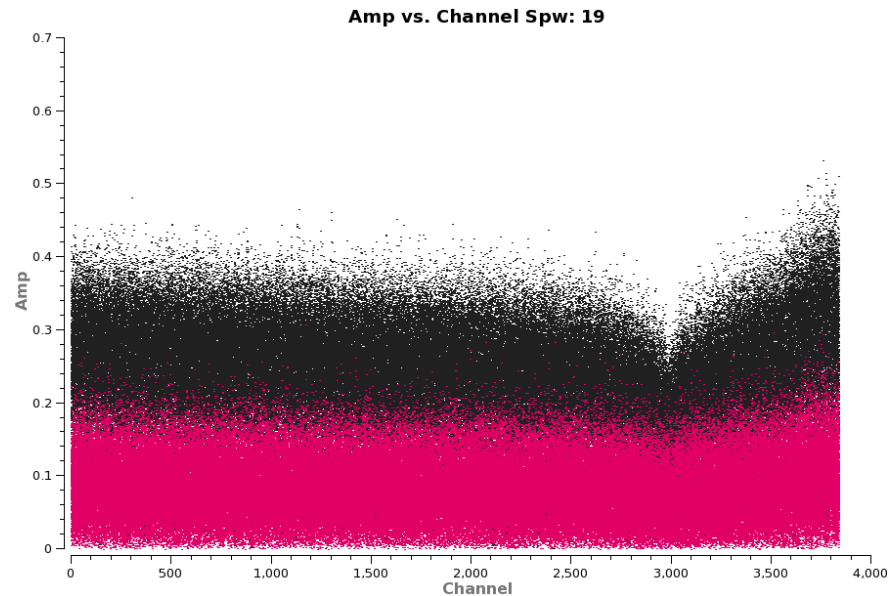
Flag the Birdies

```
splitdata=[ 'X3c1_wvrtsys.ms' , 'X5d8_wvrtsys.ms' ,  
            'X7ef_wvrtsys.ms' ]
```

```
for vis in splitdata:  
    flagdata(vis=vis, mode='manual' ,  
            action='apply' , flagbackup=T,  
            spw='0:355~356;426~427;789~790;1258~1259 ,  
                1:426~427;789~790;1258~1259 ,  
                2:426~427;1258~1259 ,  
                3:277~278;511~512;789~790 ;  
                1258~1259;1279~1279' )
```

Flag spectral features in calibrators using flagdata

There are spectral features in the calibrators, e.g. for Titan



We need to flag these using task flagdata.

Flag spectral features in calibrators using flagdata

Back up flags before flagging absorption features using **flagmanager**:

```
basename=[ "X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys" ]  
for asdm in basename:  
    flagmanager(vis=asdm+'.ms', mode='save',  
                versionname=asdm+'.before_calspectralflags')
```

Flag absorption lines in all sources:

```
for asdm in basename:  
    print "Flagging absorption features for "+asdm  
    flagdata(vis=asdm+'.ms', mode='manual',  
            spw='1:666~1279, 2:648~653')
```

Flag spectral features in calibrators using flagdata

Flag line in spw=3 for Titan:

```
for asdm in basename:  
    print "Flagging emission for "+asdm  
    flagdata(vis=asdm+'.ms', mode='manual',  
            field='1', spw='3:333~1000')
```

Check flagging using plotms

```
for asdm in basename:  
    plotms(vis=asdm+'.ms', spw='', xaxis='channel',  
          yaxis='amp', field='0,1', avgtime='1e8',  
          coloraxis='field', iteraxis='spw')  
    print('When you are done with the graphics window,')  
    print('quit that window, and')  
    user_check=raw_input('press enter to  
                          continue script \n')
```

Notice spw=2 of Titan's spectrum (pink) **has not been flagged.**

As there are no line-free channels, this spectral window cannot be used for flux calibration. **We will use another spw to calibrate spw=2 instead.**

Finished flagging!

Now we can start with the most important parts of the reduction:

- Flux calibration
- Bandpass calibration
- Gain calibration

Set up the Flux Calibrator Model using task setjy

To flux calibrate the data, we need to set the flux of Titan using a model (i.e. a uniformly illuminated disk):

```
basename=["X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys"]
for asdm in basename:
    setjy(vis=asdm+'.ms', field='1', usescratch=F,
          standard='Butler-JPL-Horizons 2012',
          scalebychan=F)
```

Check fluxes are set correctly

Question: what are the fluxes of the Titan model in each spectral window?

Can also plot model of Titan using plotms:

```
for asdm in basename:
    plotms(vis=asdm+'.ms', field='1',
           xaxis='uvdist', yaxis='amp',
           coloraxis='spw', ydatacolumn='model')
    print('When you are done with the
           graphics window,')
    print('quit that window, and')
    user_check=raw_input('press enter to
                           continue script\n')
```


Check fluxes are set correctly

Question:

what are the fluxes of the Titan model in each spectral window?

Answer: setjy gives these values in its output e.g.,

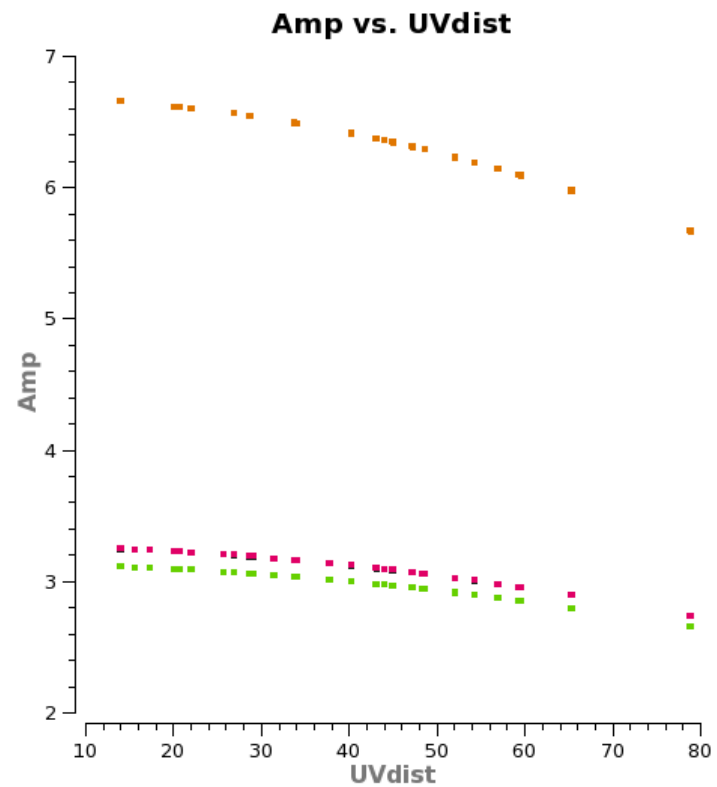
```
{'1': {'0': {'fluxd': array([ 3.27123547, 0. , 0., 0.])},
      '1': {'fluxd': array([ 3.27488661, 0. , 0., 0.])},
      '2': {'fluxd': array([ 6.69543791, 0. , 0., 0.])},
      '3': {'fluxd': array([ 3.13487768, 0. , 0., 0.])},
      'fieldName': 'Titan'},
 'format': "{field Id: {spw Id: {fluxd:[I,Q,U,V] in Jy},
 'fieldName':field name}}"}}
```

Check fluxes are set correctly

Question:

what are the fluxes of the Titan model in each spectral window?

Answer: output from plotms:



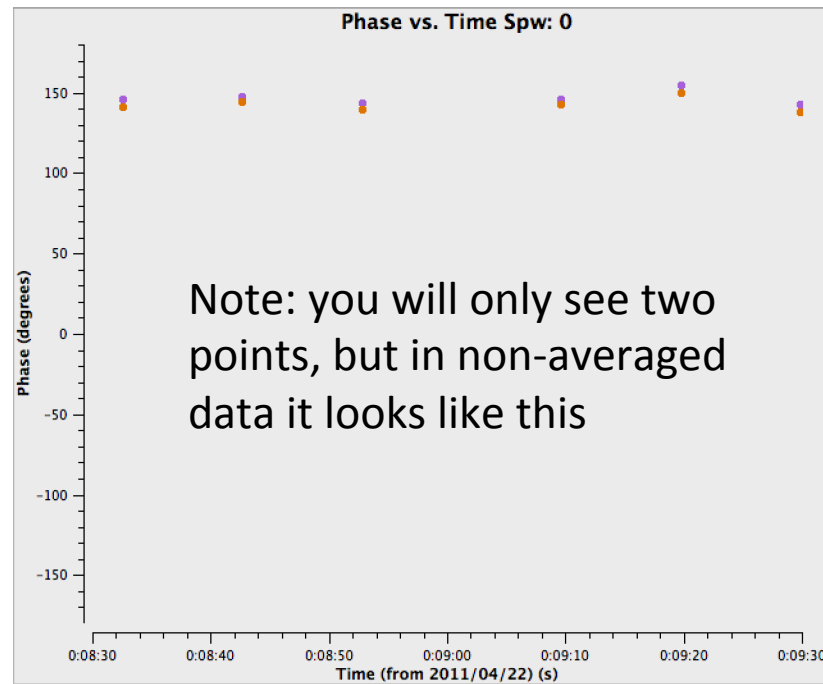
Bandpass calibration

Check if phase varies with time in bandpass calibrator scans using task plotms:

```
basename=["X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys"]
for spw in ['0', '1', '2', '3']:
    for asdm in basename:
        print "Now showing spw %s from %s" %
              (spw, asdm+'.ms')
        plotms(vis=asdm+'.ms', spw=spw, xaxis='time',
              yaxis='phase', coloraxis='corr',
              iteraxis='spw', field='0',
              avgchannel='1280', antenna='DV06&DV07',
              plorange=[0,0,-180,180])
        user_check=raw_input('press enter to go to the
                              next plot\n')
```

Bandpass calibration

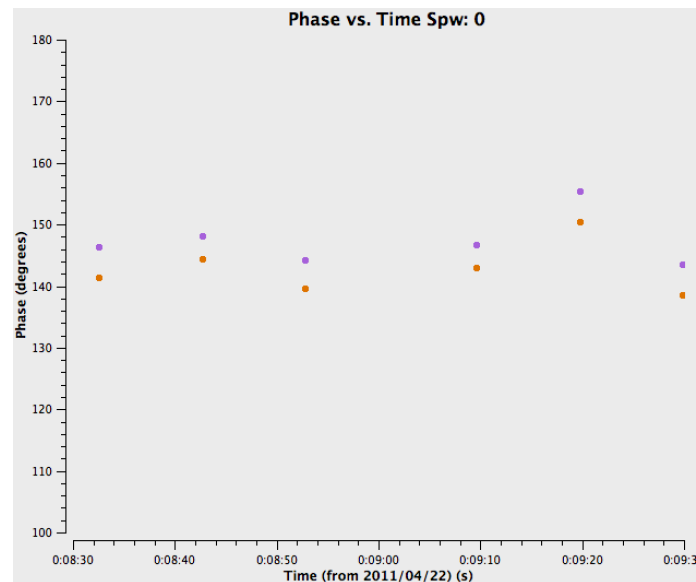
Question: Do we need to remove the phase variations across the bandpass scans before finding the bandpass solutions?



Bandpass calibration

Question: Do we need to remove the phase variations across the bandpass scans before finding the bandpass solutions?

Answer: Yes, or the bandpass phase solutions will be decorrelated.



Bandpass calibration: first remove the phase variations across the bandpass scans

Find gain solutions for the bandpass calibrator to remove the phase variations over time using task **gaincal**:

```
basename=["X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys"]
```

```
for asdm in basename:
```

```
    os.system('rm -rf ' + asdm + '.bpphase.gcal')
```

```
    gaincal(vis=asdm+'.ms',  
           caltable=asdm+'.bpphase.gcal', field='0',  
           spw='0~3:300~360', refant='DV10',  
           calmode='p', solint='int', minsnr=2.0,  
           minblperant=4)
```

Only chose the central channels
because you have not calibrated
the bandpass shape yet!

Check phase solutions for bandpass calibrator using task plotcal

```
basename=["X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys"]
for asdm in basename:
    plotcal(caltable=asdm+'.bpphase.gcal', xaxis='time',
           yaxis='phase', spw='', antenna='1~8',
           iteration='antenna', subplot=421,
           plorange=[0,0,-180,180],
           figfile=asdm+'.bpphase.X.png', poln='X')
    print('When you are done with the graphics window,')
    print('quit that window, and')
    user_check=raw_input('press enter to continue script\n')
```

To do: plot for Y polarisation as well

Bandpass calibration using task bandpass

```
basename=[ "X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys" ]
```

```
for asdm in basename:
```

```
    os.system('rm -rf ' + asdm + '.bandpass.bcal')
```

```
    bandpass(vis=asdm+'.ms',
```

```
             caltable=asdm+'.bandpass.bcal',
```

```
             field='0', spw='', combine='', refant='DV10',
```

```
             solint='inf', solnorm=T, minblperant=4,
```

```
             bandtype='B', fillgaps=10,
```

```
             gaintable=asdm+'.bpphase.gcal')
```

Question:

What do the parameters combine, soint, solnorm, minblperant, gaintable and fillgaps do?

Bandpass calibration using task bandpass

Question:

What do the parameters `combine`, `soint`, `solnorm`, `minblperant`, `gaintable` and `fillgaps` do?

Answer:

combine – combine by scan (default), obs, spw, and/or field to get solutions

solint – the time interval for one solution (for bandpass want = 'inf')

solnorm – whether the bandpass solutions are normalised to 1

(useful if you want to apply the gain solutions of one spw to another)

minblperant – required min. number of baselines per antenna

gaintable – the phase solutions to apply on-the-fly

fillgaps – number of channels to fill by interpolation

(for atmospheric lines and birdies flagged out before)

Plot the bandpass solutions using task plotbandpass

```
basename=[ "X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys" ]
```

```
for asdm in basename:
```

```
    for spw in [0,1,2,3]:
```

```
        plotbandpass(asdm+'.bandpass.bcal',  
                    xaxis='freq', yaxis='amp', spw=spw,  
                    antenna='1~8', subplot=42,  
                    figfile=asdm+'.bandpass.amp',  
                    showatm=T, interactive=True)
```

To do:

Also plot phase as well

Gain calibration using task gaincal

Determine antenna-based phase calibration solutions **per integration** (to use to derive amp. solutions shortly):

```
basename=[ "X3c1_wvrtsys", "X5d8_wvrtsys", "X7ef_wvrtsys" ]
```

```
for asdm in basename:
```

```
    os.system('rm -rf ' + asdm + '.intphase.gcal')
```

```
    gaincal(vis=asdm+'.ms',
```

```
            caltable=asdm+'.intphase.gcal',
```

```
            field='0,1,4', spw='0~3:13~1266',
```

```
            refant='DV10', calmode='p', solint='int',
```

```
            minsnr=2.0, minblperant=4,
```

```
            gaintable=[asdm+'.bandpass.bcal']])
```

min. SNR for
each solution

Apply bandpass table on the fly

Include all channels
except edge channels

one solution
per integration

Gain calibration using task gaincal


Determine antenna-based phase calibration solutions **per scan** (for applying to data later):

```
for asdm in basename:
    os.system('rm -rf ' + asdm + '.scanphase.gcal')
    gaincal(vis=asdm+'.ms',
            caltable=asdm+'.scanphase.gcal',
            field='0,1,4', spw='0~3:13~1266',
            refant='DV10', calmode='p', solint='inf',
            minsnr=2.0, minblperant=4,
            gaintable=[asdm+'.bandpass.bcal'])
```

Apply bandpass table on the fly



one solution
per scan



Gain calibration using task gaincal

Determine antenna-based amplitude calibration solutions **per scan** (for applying to data later):

```
for asdm in basename:
    os.system('rm -rf ' + asdm + '.amp.gcal')
    gaincal(vis=asdm+'.ms',
            caltable=asdm+'.amp.gcal',
            field='0,1,4', spw='0~3:40~3800',
            refant='DV10', calmode='ap', solint='inf',
            minsnr=2.0, minblperant=4,
            gaintable=[asdm+'.bandpass.bcal',
            asdm+'.intphase.gcal'])
```

↑
one solution
per scan

↑
Apply phase solution per
integration on the fly

Plotting gain solutions using task plotcal

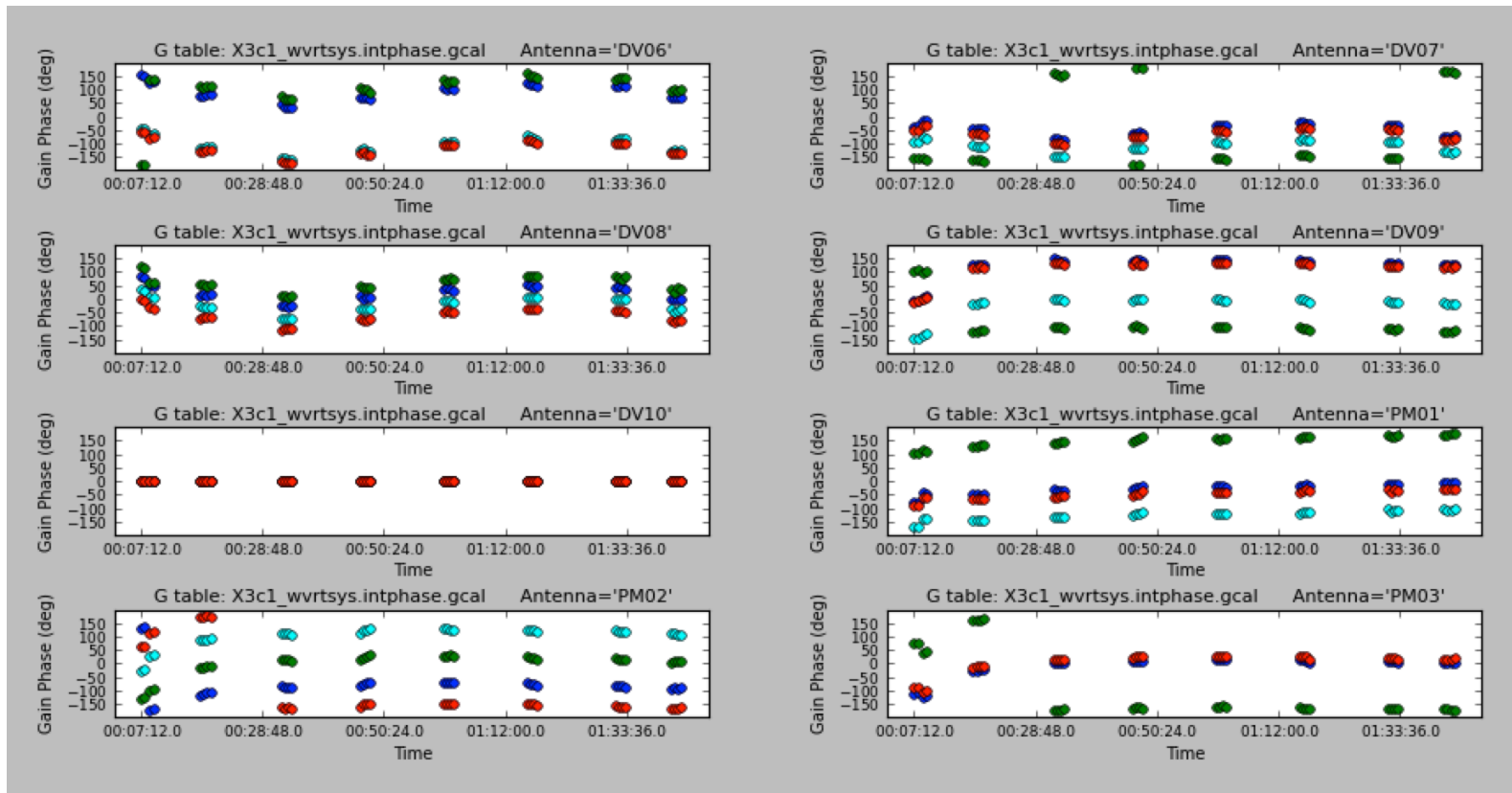
Example command:

```
for asdm in basename:  
    plotcal(caltable=asdm+'.intphase.gcal',  
           xaxis='time', yaxis='phase',  
           antenna='1~8', spw='', field='0,1,4',  
           iteration='antenna', subplot=421,  
           plorange=[0,0,-180,180], poln='X',  
           figfile=asdm+'.intphase_X.png')  
    print('When you are done with the graphics  
          window,')  
    print('quit that window, and')  
    user_check=raw_input('press enter to continue  
                          script\n')
```

To do:

plot **other phase solutions** in **X and Y** polarisation, plot **amplitude** solutions

Plotting gain solutions using task plotcal



Plot of phase vs. time for the solution per integration

Flux calibration using task fluxscale

```
for asdm in basename:
    os.system('rm -rf ' + asdm + '.flux.cal')
    fluxscale(vis=asdm+'.ms',
              caltable=asdm+'.amp.gcal',
              fluxtable=asdm+'.flux.cal',
              reference='1', refspwmap=[0,1,3,3],
              listfile=asdm+'.fluxscale.txt')
```

amplitude solutions to scale

new scaled solutions

reference is flux calibrator Titan, field=1

spw 2 will be flux calibrated by spw 3 as mentioned above (remember the emission line in Titan's spectrum?)

Flux calibration using task fluxscale

Output in the logger, e.g.:

```
Assuming all non-reference fields are transfer fields.  
Found reference field(s): Titan  
Found transfer field(s): 3c279 J1147-382=QSO J1037-295=QSO  
Spw=2 will be referenced to spw=3  
Flux density for 3c279 in SpW=0 (freq=3.56732e+11 Hz) is: 10.4538 +/- 0.0189042 (SNR = 552.988, N = 16)  
Flux density for 3c279 in SpW=1 (freq=3.57969e+11 Hz) is: 10.5748 +/- 0.0256807 (SNR = 411.782, N = 16)  
Flux density for 3c279 in SpW=2 (freq=3.458e+11 Hz) (ref SpW=3) is: 10.4554 +/- 0.0528874 (SNR = 197.692, N = 14)  
Flux density for 3c279 in SpW=3 (freq=3.43722e+11 Hz) is: 10.1136 +/- 0.0217821 (SNR = 464.309, N = 16)  
.  
.  
.
```

To do: plot flux-scaled solutions using plotcal

Apply calibration using task applycal

Most importantly, calibrate the target source TW Hya:

```
for asdm in basename:
```

```
    applycal(vis=asdm+'.ms', field='2',  
            gaintable=[asdm+'.bandpass.bcal',  
                      asdm+'.scanphase.gcal',  
                      asdm+'.flux.cal'],  
            interp=['nearest', 'linear', 'linear'],  
            gainfield=['0', '4', '4'],  
            flagbackup=T, calwt=F)
```

TW Hya

Q: why are these
set this way?

Don't calibrate the
weights, only the data

Apply calibration using task applycal

Most importantly, calibrate the target source TW Hya:

```
for asdm in basename:
```

```
    applycal(vis=asdm+'.ms', field='2',
```

TW Hya

```
        gaintable=[asdm+'.bandpass.bcal',
```

```
                  asdm+'.scanphase.gcal',
```

```
                  asdm+'.flux.cal'],
```

```
        interp=['nearest', 'linear', 'linear'],
```

```
        gainfield=['0', '4', '4'],
```

```
        flagbackup=T, calwt=F)
```

Q: why are these
set this way?

Don't calibrate the
weights, only the data

(A: 0=bandpass cal. 4=phase cal. , set for each calibration table)

Apply calibration using task applycal

To do:

if have disk space, check the calibration by applying the calibration to the other sources (make sure field and gainfield are set appropriately)

If disk space low, just apply to target and phase calibrator (fields=2 and 4).

Next we will check what the calibrated data look like...

Plotting the calibrated data using plotms

Plot the calibrated **amplitudes** of the data with **time**:

```
for asdm in basename:  
    plotms(vis=asdm+'.ms', spw='', xaxis='time',  
          yaxis='amp', field='2,4', avgchannel='1280',  
          coloraxis='field', iteraxis='spw',  
          ydatacolumn='corrected')  
    print('When you are done with the graphics window,')  
    print('quit that window, and')  
    user_check=raw_input('press enter to continue script\n')
```

Plotting the calibrated data using plotms

Plot the calibrated **phases** of the calibrators with **time**:

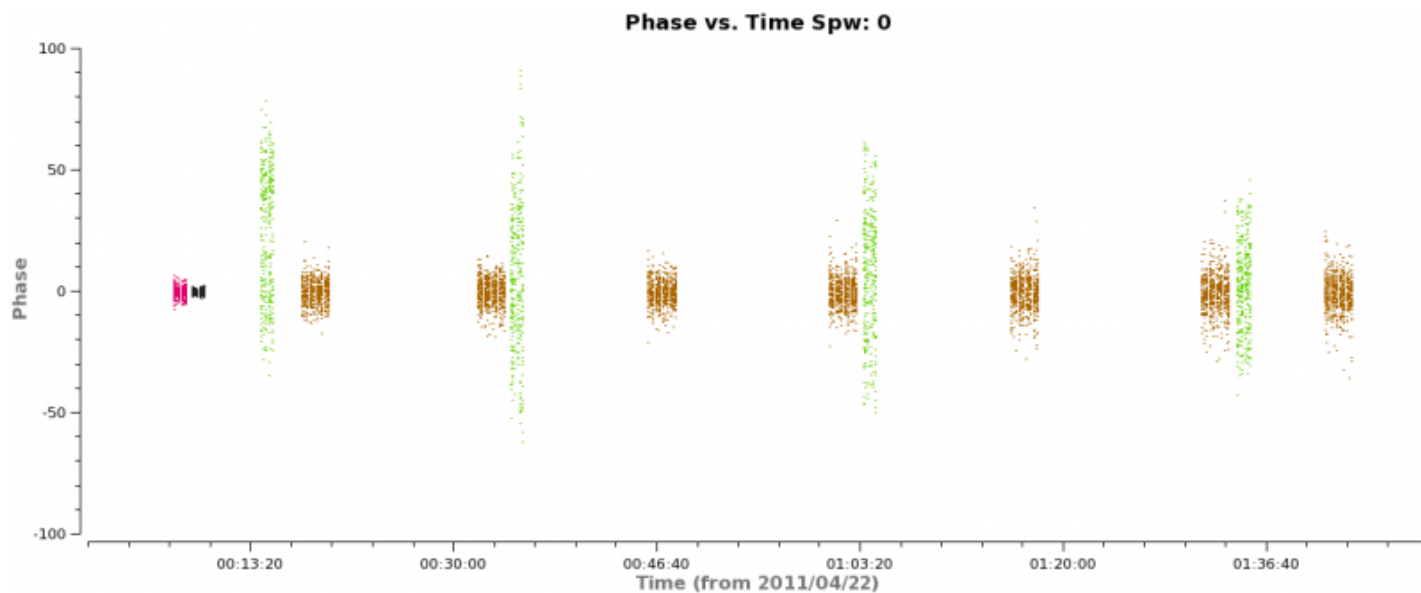
```
for asdm in basename:
    plotms(vis=asdm+'.ms', spw='', xaxis='time',
           yaxis='phase', field='4',
           avgchannel='1280', coloraxis='field',
           iteraxis='spw', ydatacolumn='corrected')
    print('When you are done with the graphics window,')
    print('quit that window, and')
    user_check=raw_input('press enter to continue script\n')
```

Question: what do you expect the plots to look like?

Plotting the calibrated data using plotms

Question: what do you expect the plots to look like?

Answer: the calibrators should have zero phase and amplitudes approx. the values given by fluxscale



Plotting the calibrated data using plotms

Plot the calibrated amplitude of the phase calibrator with frequency:

```
for asdm in basename:
    plotms(vis=asdm+'.ms', spw='', xaxis='frequency',
           yaxis='amp', field='4', avgtime='1e8',
           avgscale=T, coloraxis='field',
           ydatacolumn='corrected', iteraxis='spw',
           xselfscale=T)
    print('When you are done with the graphics window,')
    print('quit that window, and')
    user_check=raw_input('press enter to continue script\n')
```

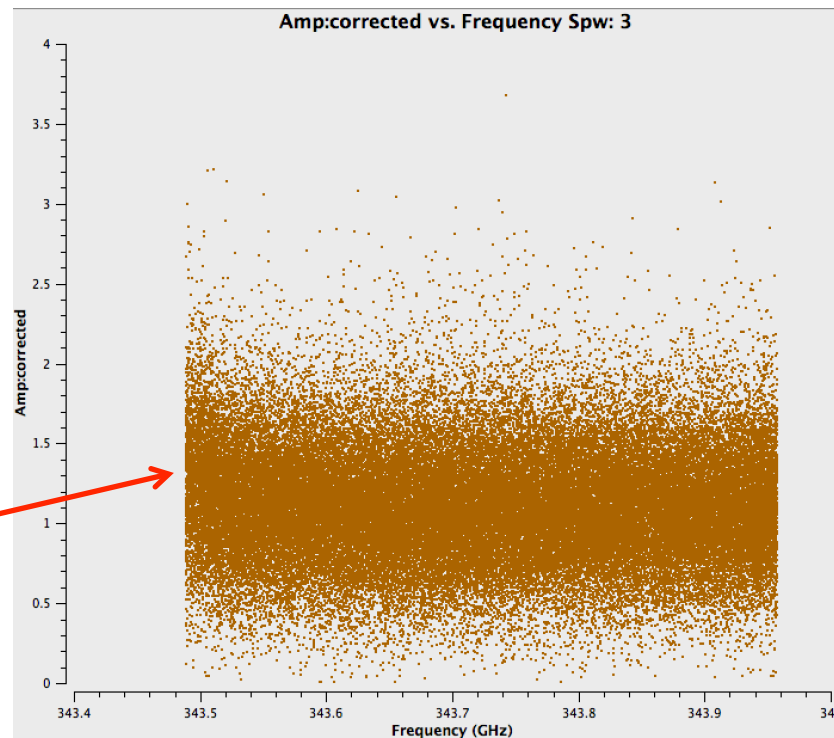
Question: is there any part of the spectrum which may need flagging?

Plotting the calibrated data using plotms

Plot the calibrated amplitude of the phase calibrator (field=4) with frequency:

Question: is there any part of the spectrum which may need flagging?

Answer:
Upturn in spw=3 seen in
phase calibrator



Finally! Plot calibrated spectra of the target (TW Hya)

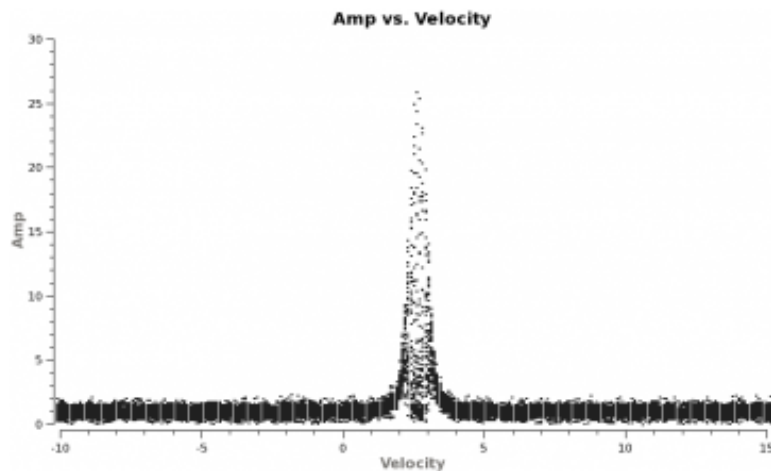
```
for asdm in basename:
    plotms(vis=asdm+'.ms', spw='', xaxis='frequency',
           yaxis='amp', field='2', avgtime='1e8',
           avgscan=T, coloraxis='field',
           ydatacolumn='corrected', iteraxis='spw',
           xselfscale=T)
    print('When you are done with the graphics window,')
    print('quit that window, and')
    user_check=raw_input('press enter to continue script\n')
```

Question: which spectral windows have lines in them?

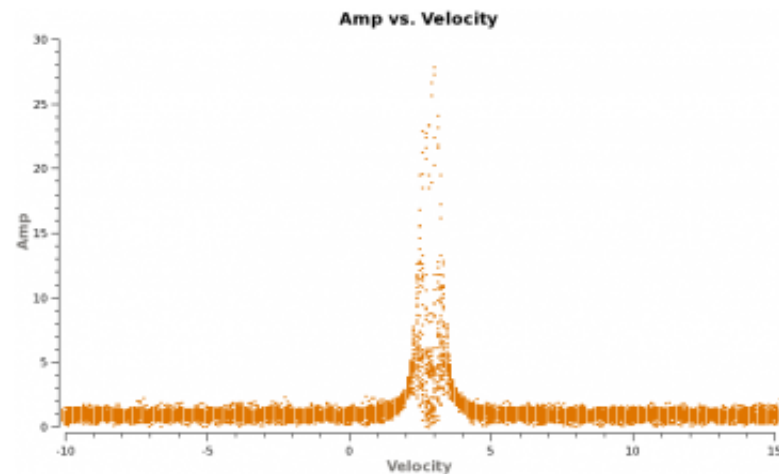
Finally! Plot calibrated spectra of the target (TW Hya)

Question: which spectral windows have lines in them?

Answer: spectral windows 0 and 2



HCO+ 4-3 rest freq. = **356.7342 GHz**



CO 3-2 rest freq at **345.79599 GHz**

Question:

Can you make these plots of flux vs. velocity? What is the systemic velocity?

You'll need the parameters: **transform**, **freqframe** and **restfreq**

(for help type "help plotms")

Concatenate the data before imaging using task concat

```
splitdata=['X3c1_wvrtsys.ms', 'X5d8_wvrtsys.ms',  
          'X7ef_wvrtsys.ms']  
os.system('rm -rf Band7multi_april22.ms')  
concat(vis=splitdata, concatvis='Band7multi_april22.ms')
```

To do:

Plot the CO spectrum (spw=2) again for the concatenated data

Split the data into separate source datasets using task split

For example, for TW Hya:

```
os.system('rm -rf TWHydra_corrected.ms')
split(vis='Band7multi_april22.ms',
      outputvis='TWHydra_corrected.ms',
      datacolumn='corrected', spw='', field='2')
```

To do: split out the phase calibrator as well using the spit command

If you're running short on space, afterwards you can delete `Band7multi_april22.ms` as it is easily regenerated.

FIN!

In the next tutorial, we image and
analyse the data!