Introduction to Continuum Self-Calibration

This guide is written for CASA 6.2.1.7 and uses Python 3.

This script steps you through continuum imaging and self-calibration of the science data for the science target TW Hydra. You should have downloaded the data package as part of the previous imaging tutorial. If you haven't done that yet, check the First Look At Imaging Guide for instructions.

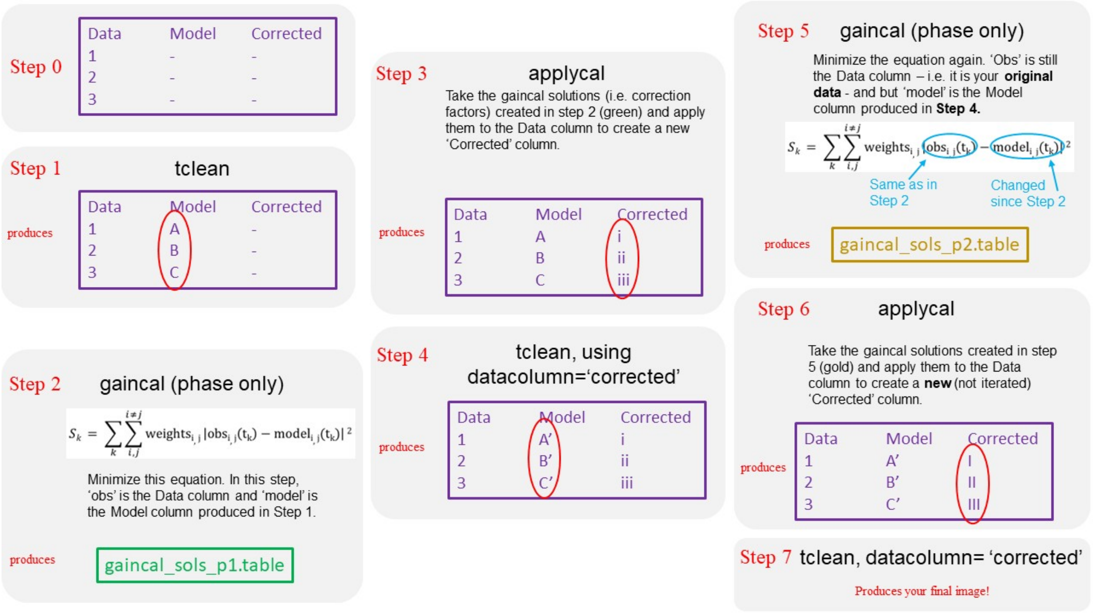
In that first tutorial you made a first continuum image in the previous imaging lesson. We start here by repeating that step and then we iteratively self-calibrate the data. The purpose of self-calibration is to improve the phase (or phase and amplitude) corrections for your data beyond what has already been achieved by the calibration pipeline using the phase calibrator. The phase calibrator used for pipeline calibration is in a different part of the sky and observed at different times than your science target(s). These differences should be slight, and pipeline calibration should already be quite good, but we can do even better by finding phase solutions for the exact periods of time, and exact sky position, over which our science data were taken. In other words, with a sufficiently good model of your source, you can calibrate on your source. You can get a model of your source through an initial image of your source. You solve for the calibrations that best match your data to your model. Self-calibration will improve the signal-to-noise ratio of your final science image, thus enabling higher-sensitivity science than you might previously have been able to achieve.

You must carefully consider whether self-cal is appropriate in the first place, and you should choose your self-cal source carefully. In particular, you need to have a high-enough S/N in your data *before* self-calibration to get a sufficiently good model of your source in the first place. For an array with ∼25 antennas, a S/N ≥ may be worth trying to self-cal. A source with a lower S/N than this may not be sufficient − you will never get good self-cal results because you will not be able to create a good enough model of your source to self-cal against.

Because you are using your science target to derive the phase (or phase+amplitude) corrections for itself, you cannot apply self-calibration solutions for one set of observations to another set. Self-cal solutions are time-, antenna-, and position-based solutions. If you observe your source on two different days, and derive self-cal solutions for the first day, you cannot apply these to the second day - you must separately derive self-cal solutions for the second day. If you observe two sources back-to-back in very different parts of the sky, you should likewise self-cal those sources separately, as there is no guarantee the solutions for source 1 will be applicable to those for source 2.

General Self-Calibration Procedure & Getting Started

Broadly speaking, the self-calibration procedure is this: 1) create a model of your source, 2) compare your data to that model, for each antenna and at a user-defined solution time interval, 3) use the difference between your data and your model to derive and apply a correction to the data to bring it in line with the model, 4) repeat as necessary. (Usually you will do 1-3 rounds of phase-only self-calibration and 0-1 rounds of amplitude+phase self-calibration.) The iterative and self-referencing nature of self-calibration is the reason self-cal sources should be chosen carefully, and solutions inspected regularly. In this procedure, `tclean` is the step that creates the model of our source, `gaincal` is the task that determines the correction factors to be applied, `plotms` is the primary way we inspect the quality of the solutions determined by gaincal, and `applycal` applies the gaincal solutions to our data. The flowchart below describes the general procedure. Don't worry too much about it at the moment - come back to it after you've finished the tutorial. You should be able to follow it clearly then.



Step 0

Data	Model	Corrected
1	-	-
2	-	-
3	-	-

Step 1 tclean

produces

Data	Model	Corrected
1	A	-
2	B	-
3	C	-

Step 2 gaincal (phase only)

$$S_k = \sum_k \sum_{i,j}^{i \neq j} \text{weights}_{i,j} \, |\text{obs}_{i,j}(t_k) - \text{model}_{i,j}(t_k)|^2$$

Minimize this equation. In this step, 'obs' is the Data column and 'model' is the Model column produced in Step 1.

produces gaincal_sols_p1.table

Step 3 applycal

Take the gaincal solutions (i.e. correction factors) created in step 2 (green) and apply them to the Data column to create a new 'Corrected' column.

produces

Data	Model	Corrected
1	A	i
2	B	ii
3	C	iii

Step 4 tclean, using datacolumn='corrected'

produces

Data	Model	Corrected
1	A'	i
2	B'	ii
3	C'	iii

Step 5 gaincal (phase only)

Minimize the equation again. 'Obs' is still the Data column – i.e. it is your **original data** - and but 'model' is the Model column produced in **Step 4.**

$$S_k = \sum_k \sum_{i,j}^{i \neq j} \text{weights}_{i,j} \, |\text{obs}_{i,j}(t_k) - \text{model}_{i,j}(t_k)|^2$$

Same as in Step 2

Changed since Step 2

produces gaincal_sols_p2.table

Step 6 applycal

Take the gaincal solutions created in step 5 (gold) and apply them to the Data column to create a **new** (not iterated) 'Corrected' column.

produces

Data	Model	Corrected
1	A'	I
2	B'	II
3	C'	III

Step 7 tclean, datacolumn= 'corrected'

Produces your final image!

First, copy the calibrated and flagged data from the working directory. Remember that this is currently our best version of the data.

```
# In CASA
os.system("rm -rf sis14_twhya_calibrated_flagged.ms")
os.system("tar -xvf ../working/sis14_twhya_calibrated_flagged.ms.tar")

Run a quick listobs to get oriented:
# In CASA
listobs("sis14_twhya_calibrated_flagged.ms")
```

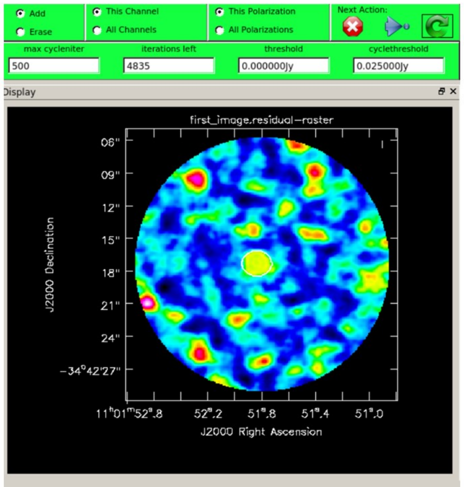
Make a note of the integration time for your science target, and of the typical length of a scan. (Make sure you are looking at science-target scans - check the field number!) Scan start and end times are in decimal days, so to get scan length in seconds, multiply by the number of seconds in a day. Then, split out the data containing only TW Hydra (field 5) - we do not need any of the other fields for self-cal, by definition.

```
  split(vis="sis14_twhya_calibrated_flagged.ms", outputvis="sis14_twhya_selfcal.ms",field=5,
datacolumn="data")
```

Phase-Only Self-Calibration Procedure

Use tclean to make a continuum image of TW Hydra. Clean until the residuals near TW Hydra are comparable to those in the rest of the image. For example, you might place your clean mask over the central feature only, and clean with for two main cycles. That is, use the green arrow twice, and then click the red X to finish tclean. Please note that *in general, it is good procedure to clean conservatively prior to the initial iterations of self-cal, which is to say: shallowly, erring on the side of missing flux instead of including flux that might be spurious.* Tw Hydra, being dominated by bright, compact emission, is relatively forgiving in that respect. For detailed discussion of these points see the TW Hydra Casa Guide and the self-calibration section of the guide to the NA Imaging Template Script. Before you close tclean, make a note of the stopping threshold for the last round of cleaning.

```
# In CASA
os.system('rm -rf first_image.*')
tclean(vis='sis14_twhya_selfcal.ms',
imagename='first_image',
spw='',
specmode='mfs',
deconvolver='hogbom',
nterms=1,
gridder='standard',
imsize=[250,250],
cell=['0.1arcsec'],
weighting='natural',
threshold='0mJy',
niter=5000,
interactive=True,
savemodel='modelcolumn')
```



● Add ○ Erase
● This Channel ○ All Channels
● This Polarization ○ All Polarizations

max cycleniter: 500
iterations left: 4835
threshold: 0.000000Jy
cyclethreshold: 0.025000Jy

Figure 1. Final round of tclean on non-self-cal'd data.
Image Beam: 0.652", 0.504", -65.889°, RMS = 12.01 mJy

What is the signal-to-noise ratio of the science target?

In addition to creating an image, tclean saves the cleaned "model" of the science target with the measurement set if the parameter savemodel="modelcolumn" is set. This model is required for later self-calibration steps. Note, in the previous lessons we only had models for the calibrators, not the science target itself. Of course this model for our science target is not perfect, only as good as the first clean, but it's a good starting point.

You may see a warning requesting you check in the CASA log that the model was created. Look for the line in the log that says 'Saving model column':

```
WARN tclean Please check the casa log file for a message confirming that the model was saved
after the last major cycle.  If it doesn't exist, please re-run tclean with niter=0, calcres=False,
calcpsf=False in order to trigger a 'predict model' step that obeys the savemodel parameter.
```

With a model in place, we are in a position to calibrate the science target directly. We use `gaincal`, which is the task used both for general gain calibration using an external calibrator, and for self-calibration. We will focus here on phase corrections - generally good practice for self-calibration - because amplitude self-calibration has a larger potential to change the source characteristics (i.e. introduce artifacts).

Gaincal does the following: it looks at your 'data' column, looks at your 'model' column, and figures out what correction needs to be applied to 'data' in order to make it look like 'model.' It produces a calibration table ('*.cal' file) with these corrections for each antenna and time, but it does not actually *apply* these corrections. Applying the corrections is done with the applycal task (see below).

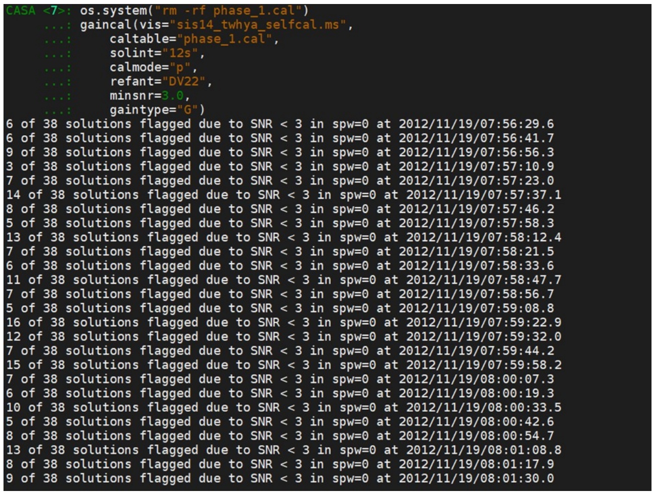
Figuring out the best averaging parameters is often the key to good self-calibration. You would like the solution interval to be short enough so that it tracks changes in the atmospheric phase with high accuracy, but long enough so that you measure phases with good signal-to-noise. Also, ideally you'd like to keep solutions separate for different spws and polarizations. For faint sources when you need to boost SNR, however, it may be necessary to average over these parameters to achieve good solutions.

Solint Values: In gaincal, the parameter 'solint' determines the time interval over which gaincal will derive phase solutions. Ideally this should be an integer multiple of your integration time (i.e. if your integration time is 3s, your solint values might be solint=3s, solint=6s, solint=9s, etc.). Solint can range from 'int' at minimum (i.e., 'integration,' your integration time for a single integration) to 'inf' at maximum (i.e. 'infinity,' which is not actually infinity but rather the maximum time interval allowed.). If you have combine='' set, setting 'inf' will be equivalent to combining all integrations within a single scan, but will not combine across scans (i.e. if your integration time is 3s, and each scan was 2 minutes long, setting solint='inf' is equivalent to setting solint='120s'). If you set combine='scan,' the effective value of solint='inf' will likewise increase.

Gaintype: 'gaintype'='G' will cause gaincal to derive solutions for each polarization independently. This becomes more important if you have a polarized source or a source whose polarization state you do not know. If your source is unpolarized, it is still a good idea to use 'gaintype=G' if you can, as this will yield the most accurate solutions. However, if you need to increase the S/N of your solutions, 'calmode'='T' will combine the data across polarizations and derive a single solution for both.

In the gaincal example below, the first choice of solint is entirely arbitrary. Try it and see what you get. Our goal is to have no failed solutions at all, and to have good S/N on the solutions we do get. (Always S/N > 3, but ideally S/N>5 or better.)

```
# In CASA
os.system("rm -rf phase_1.cal")
gaincal(vis="sis14_twhya_selfcal.ms",
caltable="phase_1.cal",
solint="12s",
calmode="p",
refant="DV22",
minsnr=3.0,
gaintype="G")
```



```
CASA <7>: os.system("rm -rf phase_1.cal")
     ...: gaincal(vis="sis14_twhya_selfcal.ms",
     ...:     caltable="phase_1.cal",
     ...:     solint="12s",
     ...:     calmode="p",
     ...:     refant="DV22",
     ...:     minsnr=3.0,
     ...:     gaintype="G")
6 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:56:29.6
6 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:56:41.7
9 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:56:56.3
3 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:57:10.9
7 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:57:23.0
14 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:57:37.1
8 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:57:46.2
5 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:57:58.3
13 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:58:12.4
7 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:58:21.5
6 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:58:33.6
11 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:58:47.7
7 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:58:56.7
5 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:59:08.8
16 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:59:22.9
12 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:59:32.0
7 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:59:44.2
15 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/07:59:58.2
7 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:00:07.3
6 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:00:19.3
10 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:00:33.5
5 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:00:42.6
8 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:00:54.7
13 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:01:08.8
8 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:01:17.9
9 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:01:30.0
```

Figure 2. An example of what failed solutions will look like in your CASA terminal.

With solint='12s', we see a large number of failed solutions that gaincal has flagged due to insufficient signal to noise (compared to the threshold set by minSNR). A small number of failed solutions may be expected depending on the solution interval, especially for more extended antennas in the configuration, although ideally you will find a solint that gives you no failed solutions. Failed solutions mean that those data will get flagged (i.e. no longer used) automatically. It is important to take into account how the resultant flagged baselines affect the beam size and image fidelity after each gaincal iteration.

Try playing around with different solution intervals or averaging options. Bear in mind that you want the shortest possible interval while also retaining separate SPW and polarizations − after all, we are solving for atmospheric distortion over time, so the smaller the solint we can get, the more precisely we can correct our data. However, none of this helps you if you don't get good solutions. If you can't get good solutions by increasing the solint value, even with solint='inf', you generally will experiment with the following options: (1) combine='scan' or 'spw' to allow solutions to cross SPW/scan boundaries, or you can do both using combine='scan,spw'; (2) toggling gaintype between 'G' and 'T'.

Plot and inspect the resulting solutions.

```
# In CASA
plotms(vis='phase_1.cal', xaxis='time',
  yaxis='SNR',gridrows=3,gridcols=3,iteraxis='antenna',
  coloraxis='corr')
```

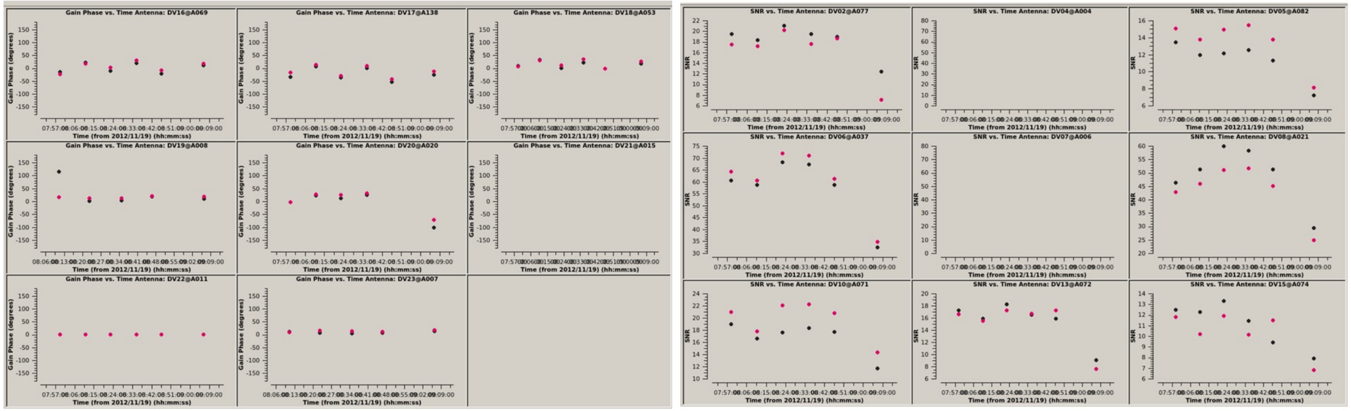


Figure 3. An example of reasonably good phase-only self-cal solutions. Left: phase vs time, Right: SNR vs time. Derived for TW Hya with solint='inf'.

```
# In CASA
plotms(vis='phase_1.cal', xaxis='time',
  yaxis='phase',gridrows=3,gridcols=3, iteraxis='antenna',
  plotrange=[0,0,-180,180],coloraxis='corr')
```

You should always examine the signal-to-noise ratio (SNR) vs. time and the phase vs. time for different settings using plotms. Note the differences in the SNR for different solint values. Ideally we want to see that the phase solutions are smoothly-varying for each antenna, that there are no jumps in the reference antenna, and that not all of our SNR values are hovering right above our cutoff value. See Figure 6 for an example of reasonably good phase and SNR values.

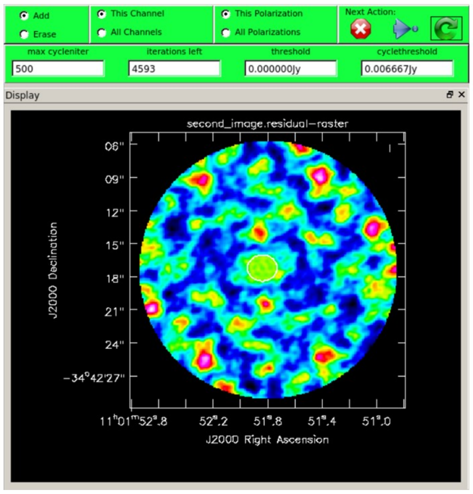
Once you are happy with your gaincal solutions, apply them to the data using applycal.

```
# In CASA
applycal(vis="sis14_twhya_calibrated_flagged.ms",   gaintable=["phase_1.cal"],   interp="linear")
```

Applycal takes your 'data' column, applies the corrections in the 'phase_1.cal' table to those values, and writes out the result to the 'corrected' column. So, at this point the self-calibrated data are stored in the "corrected data" column in our MS file. We want to use the corrected data to make our new image (and thus our new model). We therefore add the parameter `datacolumn='corrected'` to our tclean command.

Again, clean until the residuals on TW Hydra resemble those in the surrounding image.

```
# In CASA
os.system('rm -rf second_image.*')
tclean(vis='sis14_twhya_selfcal.ms',
imagename='second_image',
datacolumn='corrected',
spw='',
specmode='mfs',
deconvolver='hogbom',
nterms=1,
gridder='standard',
imsize=[250,250],
cell=['0.1arcsec'],
weighting='natural',
threshold='0mJy',
interactive=True,
niter=5000,
savemodel='modelcolumn')
```



● Add ○ Erase
● This Channel ○ All Channels
● This Polarization ○ All Polarizations
Next Action:

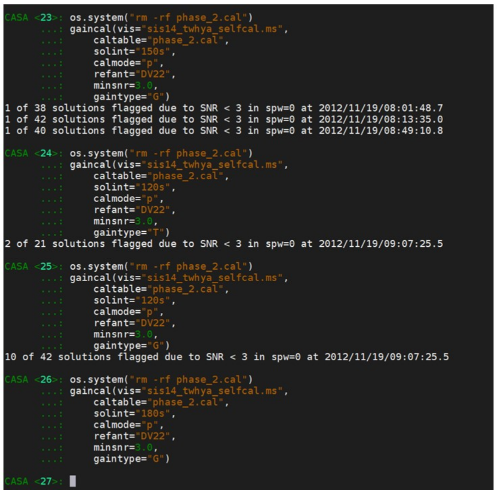
max cycleniter	iterations left	threshold	cyclethreshold
500	4593	0.000000Jy	0.006667Jy

Figure 4. Final round of tclean on data after one round of phase-only self-calibration. Image Beam: 0.639", 0.494", -65.483°, RMS = 5.1 mJy

The residuals do look better this time around. Run the viewer and compare the first (dirty) and second (self-cal'd) images. What is the signal-to-noise ratio of the science target? You should see a noticeable improvement in the noise and some improvement in the signal, so that the overall signal-to-noise (dynamic range) is much improved. You may also notice (should notice!) that you can confidently clean more deeply with progressive rounds of self-cal. If none of these things are happening, STOP! Either something has gone wrong or your data are not suitable for self-cal.

This second clean also produces a model (if the savemodel parameter is set!), hopefully a mildly better one this time. Now we will run a second round of phase-only self calibration using the improved model.

```
# In CASA
os.system("rm -rf phase_2.cal")
gaincal(vis="sis14_twhya_selfcal.ms",
caltable="phase_2.cal",
solint="180s",
calmode="p",
refant="DV22",
minsnr=3.0
gaintype="G")
```



```
CASA <23>: os.system("rm -rf phase_2.cal")
      ...: gaincal(vis="sis14_twhya_selfcal.ms",
      ...:     caltable="phase_2.cal",
      ...:     solint="150s",
      ...:     calmode="p",
      ...:     refant="DV22",
      ...:     minsnr=3.0,
      ...:     gaintype="G")
1 of 38 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:01:48.7
1 of 42 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:13:35.0
1 of 40 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/08:49:10.8

CASA <24>: os.system("rm -rf phase_2.cal")
      ...: gaincal(vis="sis14_twhya_selfcal.ms",
      ...:     caltable="phase_2.cal",
      ...:     solint="120s",
      ...:     calmode="p",
      ...:     refant="DV22",
      ...:     minsnr=3.0,
      ...:     gaintype="T")
2 of 21 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/09:07:25.5

CASA <25>: os.system("rm -rf phase_2.cal")
      ...: gaincal(vis="sis14_twhya_selfcal.ms",
      ...:     caltable="phase_2.cal",
      ...:     solint="120s",
      ...:     calmode="p",
      ...:     refant="DV22",
      ...:     minsnr=3.0,
      ...:     gaintype="G")
10 of 42 solutions flagged due to SNR < 3 in spw=0 at 2012/11/19/09:07:25.5

CASA <26>: os.system("rm -rf phase_2.cal")
      ...: gaincal(vis="sis14_twhya_selfcal.ms",
      ...:     caltable="phase_2.cal",
      ...:     solint="180s",
      ...:     calmode="p",
      ...:     refant="DV22",
      ...:     minsnr=3.0,
      ...:     gaintype="G")

CASA <27>: 
```

Figure 5. Results of different solints for the second round of phase self-cal.

If the first round of phase-only self-cal was successful, we should be able to achieve a shorter (sometimes significantly shorter!) solint than during the first round, without increasing our number of failed solutions at all or decreasing our S/N too much.

```
# In CASA
plotms(vis='phase_2.cal', xaxis='time', yaxis='SNR',gridrows=3,gridcols=3,iteraxis='antenna',
  coloraxis='corr')
```

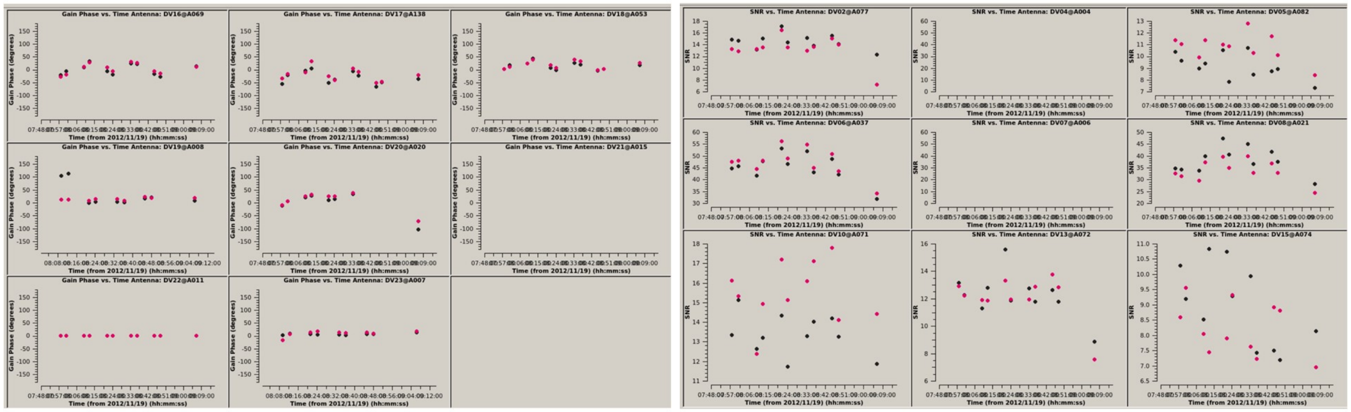


Figure 6. Phase-only self-cal solutions for the second round of phase-only self-cal. Left: phase vs time, Right: SNR vs time. Derived for TW Hya with solint='180s'. Note that we now have more solutions because we were able to decrease our time interval (solint).

```
# In CASA
plotms(vis='phase_2.cal', xaxis='time', yaxis='phase',gridrows=3,gridcols=3,iteraxis='antenna',
  plotrange=[0,0,-180,180],coloraxis='corr')
```

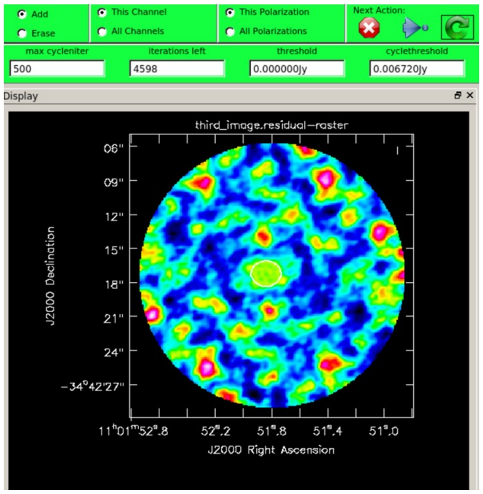
"But wait!" (You say.) "We already applied the first round of phase solutions to these data with applycal. Why don't the phase vs time plots look better?" Recall that gaincal only ever compares your 'data' column to your 'model' column. When you use applycal, you are taking the 'data' column, doing something to it, and then writing out that result to the 'corrected' column. The 'data' column never, ever gets overwritten. So, what is happening in this round of gaincal is that we are using the same uncorrected data as in the last round, but we have improved our *model*, because we used our corrected data to create a better model with tclean. In other words, we are using tclean and gaincal together to iteratively improve our model, which we will keep comparing to our uncorrected data until we are satisfied that we've made the results as good as they can be.

Apply the solutions again:

```
# In CASA
applycal(vis="sis14_twhya_selfcal.ms",gaintable=["phase_2.cal"],interp="linear")
```

Clean a third time.

```
# In CASA
os.system('rm -rf third_image.*')
tclean(vis='sis14_twhya_selfcal.ms',
imagename='third_image',
datacolumn='corrected',
spw='',
specmode='mfs',
deconvolver='hogbom',
nterms=1,
gridder='standard',
imsize=[250,250],
cell=['0.1arcsec'],
weighting='natural',
threshold='0mJy',
interactive=True,
niter=5000,
savemodel='modelcolumn')
```



● Add ○ Erase

● This Channel ○ All Channels

● This Polarization ○ All Polarizations

Next Action:

max cycleniter	iterations left	threshold	cyclethreshold
500	4598	0.000000Jy	0.006720Jy

Figure 7. Final round of tclean on data after one round of phase-only self-calibration. Image Beam: 0.639", 0.439", -65.540°, RMS = 5.08 mJy

What is the signal-to-noise ratio of the science target after the second round of phase self-cal?

In principle, we could keep going on phase-only self-cal for several more rounds. Sometimes this is desirable. In this case, our S/N improvement for the last round was really quite marginal, so we stop the phase-only self-cal here. If you wish, you can go through another several rounds of phase-only self-cal to see what the effects will be on these data.

Amplitude+Phase Self-Calibration

In many cases, it would be perfectly fine to stop self-calibrating once we are satisfied with our phase-only self-calibration solutions. In this case, we will move on to demonstrating amplitude+phase self-cal. This type of self-cal is potentially dangerous as it has much more potential to change the characteristics of the source than phase self-calibration. We mitigate this somewhat by setting solnorm=True, so that the solutions are normalized.

```
  # In CASA:
os.system("rm -rf ap.cal")
gaincal(vis='sis14_twhya_selfcal.ms',
  caltable="ap.cal",
  gaintable=["phase_2.cal"],interp=["linear"],
  solint="inf",
  calmode="ap",
  refant="DV22",
  gaintype="G",
  solnorm=True)
```

Note that we have increased our solint back to 'inf.' This is common for amplitude self-cal: you will want a solint for the amplitude self-cal that is larger than the solint for your best round of phase-only self-cal. We have also added the line 'gaintable=...' This is because we don't want to let ap self-cal have too many degrees of freedom. Instead, we force it to use our phase-only solutions first, before it is allowed to calculate amplitude+phase solutions.

Plot the calibration tables again, including the amplitude solutions:

```
# In CASA
plotms(vis='ap.cal', xaxis='time', yaxis='SNR',gridrows=3,gridcols=3,iteraxis='antenna',
  coloraxis='corr')
```

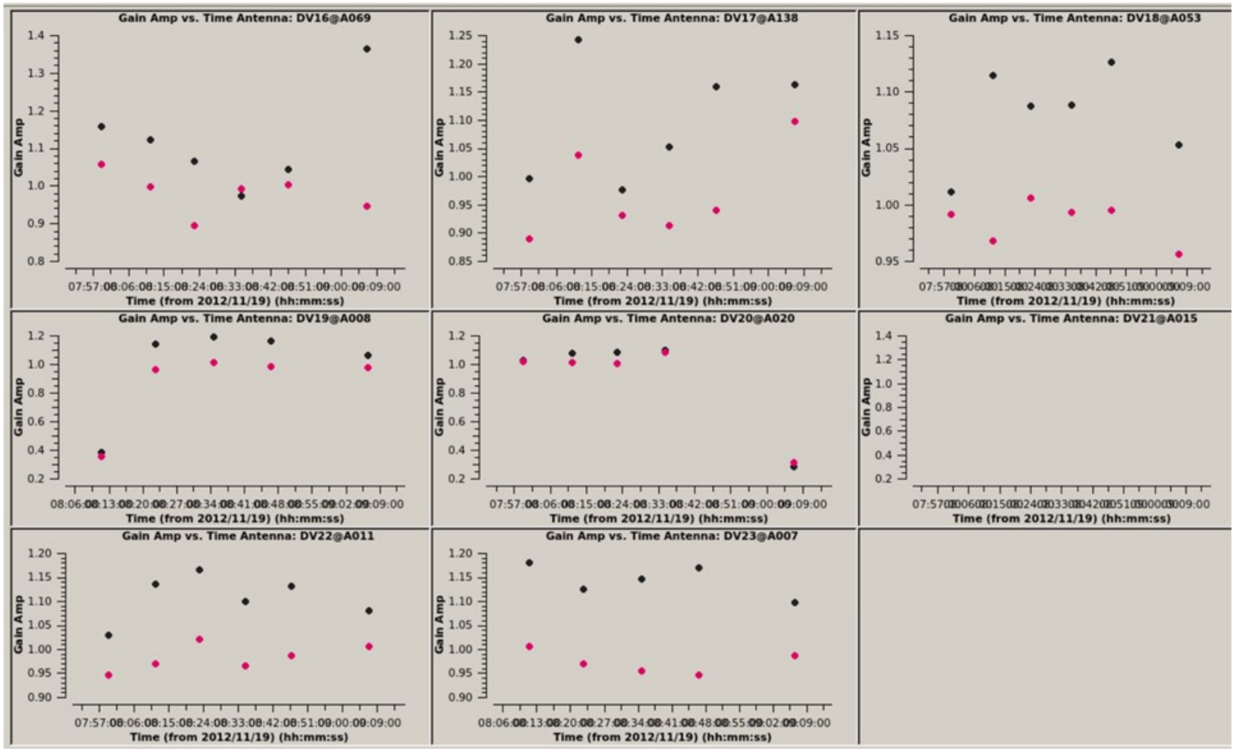


Figure 8. Amplitude vs time for ap-cal with solint='inf'.

```
# In CASA
plotms(vis='ap.cal', xaxis='time', yaxis='phase',gridrows=3,gridcols=3,iteraxis='antenna',
  plotrange=[0,0,-180,180],coloraxis='corr')
```

```
# In CASA
plotms(vis='ap.cal', xaxis='time', yaxis='amp',gridrows=3,gridcols=3,iteraxis='antenna',
  plotrange=[0,0,0,0],coloraxis='corr')
```

For phase and SNR, we are looking for the same things here as we did during phase-only self-cal. For the amplitude plots, we ideally want the amplitude corrections to be <10%, but absolutely no more than 20% (i.e. between 0.9 and 1.1 on the y-axis, or at most between 0.8 and 1.2).

Even with solint='inf,' we have some antennae with amplitude corrections > ±20% for these data. What can you do to try to improve these solutions further?

Once you have a set of solutions you like (if any), apply them and your best round of phase-only self-cal to the data:

```
# In CASA
applycal(vis='sis14_twhya_selfcal.ms', gaintable=["phase_2.cal","amp.cal"],interp=["linear","linear"])
```

Let's take as an example a set of 'ap' gaincal solutions with solint='inf' and combine='scan.' In this case we have good SNR and good phase behavior, but at least a couple of antennas with amplitude corrections of −20% to −30%.

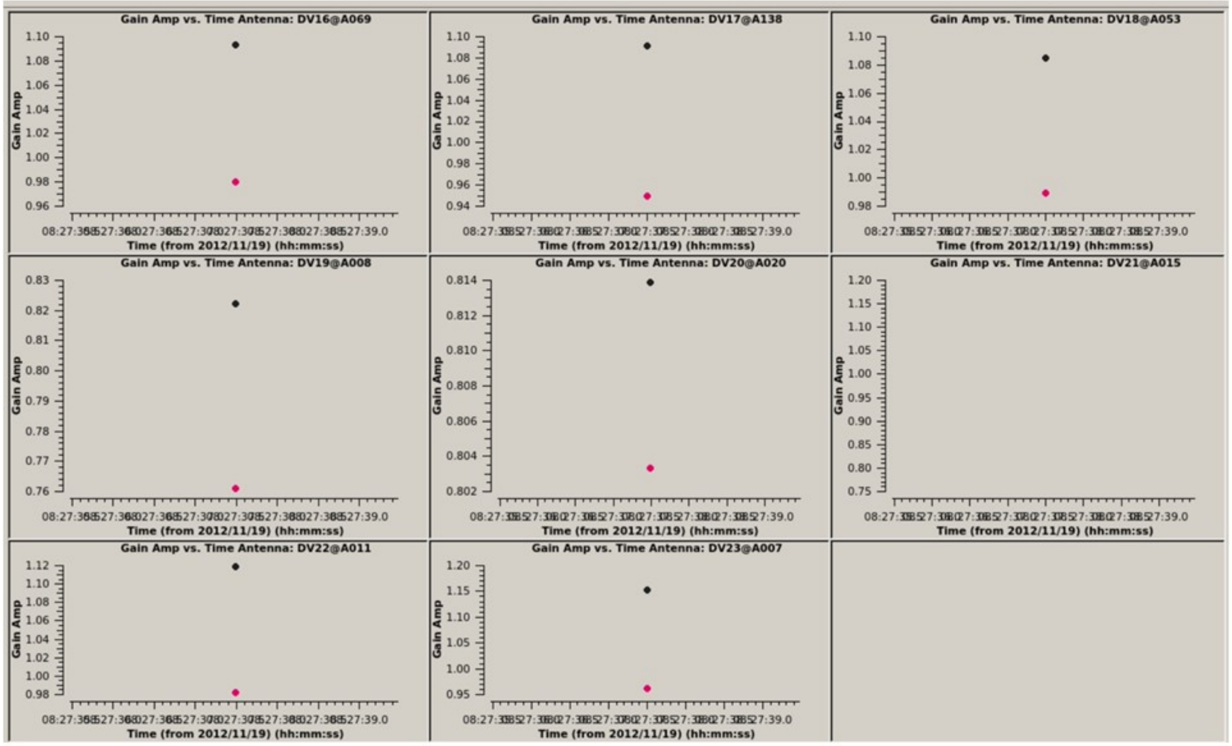


Figure 9. Amplitude vs. time for ap selfcal with solint='inf' and combine='scan'.

If we apply these solutions to our data, they may lower our measured flux values in an unrealistic way. Go ahead and apply these solutions to your data, and then run tclean again, this time with the name stem 'apcal_image'. Measure the RMS noise, and the peak of the science target, and determine the SNR. Compare these to previous values from this process. How have the peak, the RMS, and the SNR all changed? Have they all changed in the same way? Are these numbers all telling you the same thing?

Whether you are satisfied with the apcal self-calibration in this case or not, it is very important to understand that amplitude+phase self-cal is not always successful, nor is it always necessary. If you can get good solutions with apcal self-calibration, you will likely have improved your S/N ratio by an order of magnitude or more over the non-self-calibrated data. Likewise, if you have had a significant improvement with phase-only self-cal, and you can't get good solutions for amplitude+phase, that's okay - you have still significantly improved your data!