Introduction to (Maser) Line Self-Calibration

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

This guide pulls heavily from the continuum self-calibration tutorial from CASAGuides and the PDF I uploaded for the UF Data Workshop. Areas where there is additional text related to line self-cal specifically are highlighted **in bold.**

This script steps you through line imaging and self-calibration of the science data for the science target G14.33−0.64, a massive star-forming region in the Milky Way. You should have downloaded the data package (2 .zip files) from the UF Data Workshop website. If you haven't done that yet, please do so now.

In the self-cal continuum tutorial you iteratively self-calibrated by running tclean to get a model of your data, running gaincal to determine the correction factors needed to make the phases from each antenna agree with each other, and applying those corrections with applycal. This process will be very similar, but instead of using a continuum image we will use a line cube.

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. **For line data in particular, it may be more difficult to achieve the S/N you need for good self-calibration. (Your noise level is inversely proportional to both your bandwidth and your integration time − for a given integration time, line observations will always have worse sensitivity in a given channel than continuum emission, because the bandwidth of continuum emission will always be much larger than any single channel. For this tutorial, you are using observations of a** *maser line* **- masers are extremely bright, unresolved sources undergoing a non-LTE process. (They are basically naturally-occurring lasers, at radio wavelengths.) Because they are often very bright, they can be ideal for line self-cal[1].**

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. **However, if you have taken line and continuum observations simultaneously for a given source, and either your line or continuum emission are suitable for self-cal, then you can apply your self-cal solutions derived from your continuum (or line) data to your line (or continuum) data for that source.**

[1] Not all masers are bright enough for self-calibration - it's not a magic bullet! But if you know you can observe a maser line for free, and you know that that maser line is usually very bright (e.g. 22 GHz water line, 6.7 GHz methanol line), you might have a good, easy opportunity for self-calibrating your data.

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 in the continuum self-cal PDF uploaded to the Workshop website and in some of the lecture notes describes this graphically. **For line self-calibration, you will select a single channel to use to derive your phase solutions, and then apply the solutions you derive from that one channel to all the channels in your data cube.**

First, unzip the G14.33_water_Jun2018.bright.ms.contsub.cvel file if you haven't already. This file name is slightly different because I have done some additional processing to it (continuum subtraction, regridding in velocity space), but it is still an MS file. Then, run a quick listobs to get oriented:

```
# In CASA
listobs('G14.33_water_Jun2018.bright.ms.contsub.cvel')
```

Make a note of the integration time for your science target, and of the typical length of a scan. 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.

Unlike for continuum self-calibration, for line self-cal, we need to spend a bit more time getting to know our data first. Specifically, we need to decide which specific channel to use for self-cal. Plot your data in plotms() with amplitude on the y-axis, channel number on the x-axis, and time averaging set to 1E08 and scan averaging on. (Setting time averaging to an unrealistically large number just tells plotms to average the data over the largest time period it can. For instance, if scan averaging is off, "the largest time period it can" will be one scan.) Figure 1 shows what you should see in the plotms window. Good candidates for which channel to choose - and some bad ones - are indicated with arrows. Remember, for an antenna with $\sim$25 antennas, an S/N $\gtrsim$20 is worth trying to self-cal on. Useful buttons in plotms are circled in green at the bottom of the figure - hover over the buttons in your own plotms window to see what they do.

We want good signal-to-noise in our gaincal solutions, which means we want the most signal possible. So, start by finding which channel has the brightest emission using plotms's 'locate' tool. Once you have found what you think is the brightest channel, write that number down in your selfcal.py file (or whatever you've chosen to call it). Go ahead and also find the *second*-brightest channel and write that down too. This is in case we can't use the brightest channel for some reason.

Phase-Only Self-Calibration Procedure

Use tclean to make a continuum image of your brightest channel *only*. Clean until the residuals 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.* Maser emission, being very bright and almost by definition a point source, 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
imsize=[400,400]
```

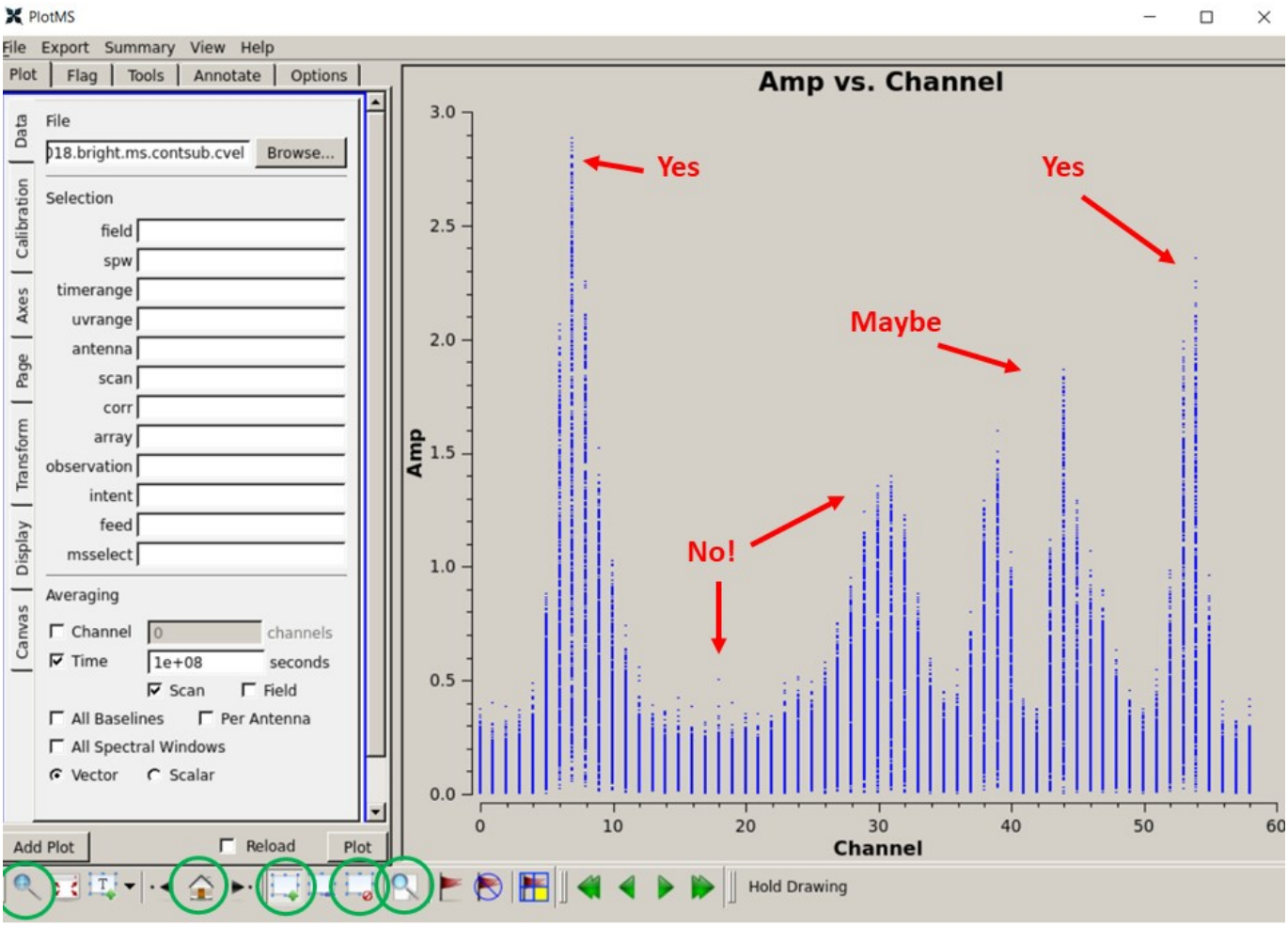

Figure 1. Examining line cube MS file to select which channel to use for self-calibration.

```
phasecenter=''
cell=['0.05arcsec']
robust=0.0
velstart= <yourpeakchannelhere>
chans=1
width=1
outframe='LSRK'
threshold='1.0mJy'
data='G14.33_water_Jun2018.bright.ms.contsub.cvel'
restfreq='22.23508GHz'
stokes='RRLL'
refant='ea02,ea05,ea09'

imagename=data.replace('cvel','cvel.1ch')
os.system('rm -rf '+imagename+'.*')
tclean(vis=data,
  imagename=imagename,observation='',stokes=stokes,
  imsize=imsize,cell=cell,phasecenter=phasecenter,
  gridder='standard',restoringbeam='common',
```

```
weighting='briggs',robust=robust,datacolumn='data',
specmode='cube',start=velstart,nchan=chans,width=width,
restfreq=restfreq,outframe=outframe,
pblimit=0.1,savemodel='modelcolumn',
interactive=True,niter=10000000,threshold=threshold)
```

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 in the measurement set if the parameter savemodel="modelcolumn" is set. This model is required for later self-calibration steps. 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 this case, because we are using line data and not continuum data, we need to tell gaincal which specific spectral window and channel to use. We do this with the 'spw' parameter inside the gaincal command.

```
# In CASA
os.system('rm -rf phasecal1')
gaincal(vis=data,caltable='phasecal1',gaintype='G',
  refant=refant,calmode='p',combine='',spw='0:<yourchan>',
  solint='<yoursolint>',minsnr=3.0)
```

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). 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='phasecal1',xaxis='time',yaxis='phase',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,plotrange=[0,0,-180,180],coloraxis='corr')

  plotms(vis='phasecal1',xaxis='time',yaxis='snr',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,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.

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

```
# In CASA
applycal(vis=data,gaintable=['phasecal1'],calwt=False,flagbackup=False,interp='linearPD')
```

Applycal takes your 'data' column, applies the corrections in the 'phasecal1' 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 resemble those in the surrounding image. **Because these are maser data and were taken with a different telescope (the VLA), you may start to get point-like image artifacts as you clean deeper. If you are concerned about these, either stop cleaning or come get me or both.**

```
# In CASA
imagename=data.replace('cvel','cvel.1ch_p1')
os.system('rm -rf '+imagename+'.*')
tclean(vis=data,
  imagename=imagename,observation='',stokes=stokes,
  imsize=imsize,cell=cell,phasecenter=phasecenter,
  gridder='standard',restoringbeam='common',
  weighting='briggs',robust=robust,datacolumn='corrected',
  specmode='cube',start=velstart,nchan=chans,width=width,
  restfreq=restfreq,outframe=outframe,
  pblimit=0.1,savemodel='modelcolumn',
  interactive=True,niter=10000000,threshold=threshold)
```

Once you've finished cleaning, run the viewer and compare the first (non-self-cal'd) 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 phasecal2')
gaincal(vis=data,caltable='phasecal2',gaintype='G',
  refant=refant,calmode='p',combine='',spw='0:<yourchan>',
  solint='<yoursolint>',minsnr=3.0)
```

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='phasecal2',xaxis='time',yaxis='phase',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,plotrange=[0,0,-180,180],coloraxis='corr')

plotms(vis='phasecal2',xaxis='time',yaxis='snr',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,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=data,gaintable=['phasecal2'],calwt=False,flagbackup=False,interp='linearPD')
```

Clean a third time.

```
  # In CASA
imagename=data.replace('cvel','cvel.1ch_p2')
os.system('rm -rf '+imagename+'.*')
tclean(vis=data,
imagename=imagename,observation='',stokes=stokes,
imsize=imsize,cell=cell,phasecenter=phasecenter,
gridder='standard',restoringbeam='common',
weighting='briggs',robust=robust,datacolumn='corrected',
specmode='cube',start=velstart,nchan=chans,width=width,
restfreq=restfreq,outframe=outframe,
pblimit=0.1,savemodel='modelcolumn',
interactive=True,niter=10000000,threshold=threshold)
```

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, with maser data, it's probably not necessary. 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 apcal')
gaincal(vis=data,caltable='apcal',gaintype='G',
  gaintable=['phasecal2'],interp='linearPD',
  refant=refant,calmode='ap',combine='',spw='0:<yourchan>',
  solint='<yoursolint>',minsnr=3.0)
```

You will probably have to increase your solint back to something greater than what you used for your second round of phase self-cal. 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='apcal',xaxis='time',yaxis='phase',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,plotrange=[0,0,-180,180],coloraxis='corr')

plotms(vis='phasecal2',xaxis='time',yaxis='amp',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,plotrange=[0,0,0,0],coloraxis='corr')
```

```
plotms(vis='phasecal2',xaxis='time',yaxis='snr',spw='',iteraxis='antenna',
  gridcols=3,gridrows=2,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).

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=data,gaintable=['phasecal2','apacal'],calwt=False,flagbackup=False,
  interp=['linearPD','linearPD'])
```

Now image your final, amplitude+phase self-calibrated data:

```
# In CASA
imagename=data.replace('cvel','cvel.1ch_ap')
os.system('rm -rf '+imagename+'.*')
tclean(vis=data,
imagename=imagename,observation='',stokes=stokes,
imsize=imsize,cell=cell,phasecenter=phasecenter,
gridder='standard',restoringbeam='common',
weighting='briggs',robust=robust,datacolumn='corrected',
specmode='cube',start=velstart,nchan=chans,width=width,
restfreq=restfreq,outframe=outframe,
pblimit=0.1,savemodel='modelcolumn',
interactive=True,niter=10000000,threshold=threshold)
```

Compare the no-selfcal, phase1, phase2, and amp+phase self-calibrated images to see how the S/N, noise, and overall image quality change between each.

Applying Line Self-Calibration to the Full Cube, and to Other Data

So now that you have your lovely self-cal solutions from your single channel, how do you apply them to the rest of your data? Well, for the 'G14.33_water_Jun2018.bright.ms.contsub.cvel' file, you already have! We didn't tell applycal to operate on only one channel. Every time you used gaincal, you were using a single channel (spw='0:<yourchan>'), but every time you ran applycal, you were applying the calibration tables you derived to the entire MS file. It is possible to run applycal with the same spw='0:<yourchan>' parameter as gaincal, if you really want, but once you're satisfied with your gaincal solutions there's no reason not to apply them to the whole dataset.

In order to see your nice, final, self-calibrated data *cube*, instead of just one channel, run the following command:

```
# In CASA
imsize=[200,200]
phasecenter=''
cell=['0.05arcsec']
robust=0.0
velstart='30.0km/s'
chans=8
width='0.25km/s'
outframe='LSRK'
data='G14.33_water_Jun2018.bright.ms.contsub.cvel'
```

```
restfreq='22.23508GHz'
threshold='5.0mJy'

imagename=data.replace('cvel','cvel.cube')
os.system('rm -rf '+imagename+'.*')
tclean(vis=data,
  imagename=imagename,observation='',
  imsize=imsize,cell=cell,phasecenter=phasecenter,
  gridder='standard',restoringbeam='common',
  weighting='briggs',robust=robust,datacolumn='corrected',
  specmode='cube',start=velstart,nchan=chans,width=width,
  restfreq=restfreq,outframe=outframe,
  uvtaper=uvtaper,pblimit=0.1,
  mask='',
  interactive=True,niter=10000000,threshold=threshold)
```

So what about applying your self-cal solutions to other data sets? As stated above, this must be done with caution. You can only apply your gaincal-derived calibration tables ('phasecal2','apcal') to data that were taken for the same target, at the exact same times, with the same antennas as the data that you used for self-cal. But, this is not as restrictive a caveat as it may seem. If you took simultaneous continuum and line observations of the same source, and did self-cal with the line data, you can apply those solutions to your continuum data as well.

The file 'G14.33_water_Jun2018.dim.ms.contsub.cvel' is a very small cube (cut from a much larger one) that was taken simultaneously with the 'bright.ms' data. Open the 'dim.ms' dataset in plotms and find a velocity range over which you might want to image (maybe 5 channels or so). Go ahead and make a quick cube with the 'dim' dataset, using a similar tclean command as the first tclean command in this document but with an appropriate difference in filenames and velocity ranges. Make a note of your stopping threshold. (If you just want to do this exercise without having to image the whole cube, use velstart='14.5km/s', nchan=1, and width='0.25km/s'.) Then, instead of going through the whole self-cal process again, run the command:

```
# In CASA
applycal(vis='G14.33_water_Jun2018.dim.ms.contsub.cvel',gaintable=['phasecal2','apacal'],
  calwt=False,flagbackup=False,interp=['linearPD','linearPD'])
```

Now re-run the tclean command above, but with a new imagename (or else you'll delete the cube you just made!). Compare the first cube with the second cube (S/N, noise, etc). Can you see any improvement with self-cal for these data as well?