VLBA Scientific Memo #38

VLBA Data Calibration with CASA

Phase-Referenced Observations

For CASA 6.3 and Later

Justin Linford

30 June 2022

Version 1.1

Abstract

This document is intended to guide users through the process of calibrating phase-referenced (continuum or spectral line) VLBA data with CASA. It was written for CASA 6.3, but will also work for CASA 6.4 or higher. Most of the functionalities discussed are also included in CASA 5.8 and 6.2, however there are some benefits to using 6.3 and higher. Projects that require fringe-rate mapping, polarimetric calibration, Earth Orientation Parameter (EOP) corrections, or pulse-cal tone corrections should use AIPS for calibration because these functionalities are not yet available in CASA. Projects that require Total Electron Content (TEC) corrections should also use AIPS because the CASA TEC correction procedure for very long baselines has not been verified.

1 Introduction

The first official NRAO VLBA CASA Guide tutorial is available from the VLBI CASA Guide website: https://casaguides.nrao.edu/index.php?title=VLBI_Tutorials.
The tutorial covers the calibration and imaging of a basic VLBA phase-referenced continuum observation.

In addition to the VLBA tutorial, our colleagues at the Joint Institute for VLBI ERIC (JIVE) and the European VLBI Network (EVN) have created some excellent tutorials for calibrating EVN continuum observations with CASA:
Des Small's EVN tutorial: http://www.jive.eu/~small/FringeFitting/n14c3_tutorial.html
DARA EVN tutorial: http://www.jb.man.ac.uk/DARA/unit4/Workshops/EVN_continuum.html
EVN Data Reduction Guide: https://www.evlbi.org/evn-data-reduction-guide

This document uses the VLBA CASA Guide tutorial and the DARA EVN tutorial as the basis for a "recipe" for calibrating VLBA phase-referenced observations with CASA. Readers are encouraged to consult the tutorials for additional details of VLBI calibration.

Major differences between calibrating VLBA and EVN data in CASA:

A. For VLBA data, there is no need to attach the system temperature table to the FITS-IDI file with an external script. VLBA system temperatures are included in the FITS-IDI file and CASA can generate the necessary calibration table natively.

B. As long as you are using CASA 5.8/6.2 or later, there is no need to extract the VLBA gain curve with an external script. CASA 5.8/6.2 and later can natively import and extract the VLBA gain curve table from the FITS-IDI file.

C. DO NOT flag the autocorrelations for VLBA data! VLBA data calibration requires using the autocorrelations for correcting amplitude mis-normalizations due to sampling errors.

NOTE: At the present time, NRAO continues to recommend that users rely on AIPS for all VLBA calibration. CASA should be used with caution, and only for relatively simple observations. In particular, AIPS should be used to calibrate the following types of VLBA observations:

- Polarimetric – Calibration of resolved polarized sources is not yet available in CASA.
- Astrometric – EOP and pulse-cal tone corrections are not yet available in CASA.
- Spectral line projects requiring fringe-rate mapping – Fringe-rate mapping is not yet available in CASA.
- Low Frequency (< 4 GHz) – TEC corrections have not yet been fully tested for VLBI observations in CASA.
- VLBA+Y1 – Adding the gain curve values for the Y1 antenna is currently difficult (although not impossible) in CASA.

1.1 First Things First

Before getting into the actual calibration, you may need to install CASA and get your VLBA data.

1.1.1 Obtaining CASA

CASA is available to download for free from the NRAO CASA homepage (casa.nrao.edu). NRAO currently recommends using CASA 6.3 or higher for VLBA calibration, but 5.8 and 6.2 will also work (although they are missing some recent improvements). In general, it is best to use the latest version of CASA.

1.1.2 Obtaining Your Data

Download your VLBA data from the NRAO archive (data.nrao.edu)[1]. The easiest way to find your data is to provide the project code for the observation(s). When using the NRAO archive, you will need to log in (upper right-hand corner of page) and be listed as a co-investigator on the project to have access to any proprietary data. If you are not a co-investigator on the project but you have a legitimate need to access propreitary data (e.g., a student who joined the project after the proposal was submitted), please contact NRAO staff via the helpdesk (help.nrao.edu).

1.2 Document Format

Throughout the remainder of this document, CASA tasks and their associated parameter values are shown in `Courier Font`. Any input parameters that depend on a user's choice are given in <>. For example, `vis='<your MS filename>'` indicates that the name of the visibility file is dependent on a user's choice, but the single tics (`'`) are necessary to define the parameter properly. If a user named their visibility file "MyBigScaryVLBAProject.ms", they would enter `vis='MyBigScaryVLBAProject.ms'`. (PRO TIP: Keep your filenames concise, but meaningful!)

[1]The legacy archive, archive.nrao.edu, was retired on 2022 May 02.

1.3 The Calibration Steps

In the calibration procedure outlined below, the associated CASA tasks are in [ ] and the relevant sections of this document are in { }. This procedure is for a phase-referenced observation (continuum or spectral line). Remember that it is good practice to inspect the calibration solutions after they are created using `plotms` (CASA 5.8, and 6.2 or higher) or `plotcal` (CASA 5.8 only).

1. Start CASA
2. Convert the FITS-IDI file into a measurement set (MS) that CASA can use and inspect the antenna parameters at this point to make sure they do not need to be corrected [`importfitsidi`, `listobs`]{§2.1.1}
3. Inspect the data, select a reference antenna, and identify a good scan/timerange for the instremental delay and the bandpass calibration [`plotms`, `viewer`]{§2.1.2, §2.1.3, §2.1.4}
4. Flag the data [`flagdata`]{§2.1.5}
5. Determine the amplitude corrections from autocorrelations [`accor`]{§2.2}
6. Prepare the *a priori* calibration tables (system temperature and gain curve) [`gencal`]{§2.3}
7. Correct instrumental delays, also called "fringe finding" or "single band delay" (fringe fitting a single bright source – the "fringe finder") [`fringefit`, `applycal`]{§2.4.1}
8. Correct the phase, delay, and rate for the entire observation, also called "global fringe fitting" or "multi-band delay" (fringe fitting on all sources bright enough to have sufficient SNR, usually the fringe finder and phase reference calibrators) [`fringefit`, `applycal`]{§2.4.2}
9. Correct for the shape of the bandpass, also called "bandpass calibration" (requires a bright source, often the fringe finder) [`bandpass`, `applycal`]{§2.5}
10. For wide bandwidth observations ($\geq$ 256 MHz of bandwidth per polarization), observers should perform a second amplitude correction at this point (in AIPS, this is the ACSCL step) [`accor`, `applycal`]{§2.6}

At this point, the data should be mostly calibrated. However, there could be lingering problems that can be corrected using iterative calibration techniques which derive additional corrections from models of a source. This is referred to as "self-calibration". For the first round of self-calibration, the phase reference calibrator is used.

11. Make an image of the phase reference calibrator, making sure to save the model [`tclean` with `savemodel='modelcolumn'`]{§3.4}
12. Refine the delays and phases using the image of the calibrator as an input model. This is usually called "phase self-calibration" [`gaincal`, `applycal`]{§3.5}
13. Make a new image using the refined delays and phases and check for improvement [`tclean`, `imstat`]{§3.5.1}
14. Repeat the phase self-calibration (steps 14 and 15) as many times as necessary, usually until no major improvement is seen in the resulting image. Make a new image after each `applycal`, always setting `savemodel='modelcolumn'`. Always check for improvement in each new image. [`gaincal`, `applycal`, `tclean`, `imstat`]
15. Refine the amplitude calibration, also known as "amplitude self-calibration" [`gaincal`, `applycal`]{§3.6}

16. Make a new image with amplitude self-calibration solutions applied and check for improvement [`tclean`, `imstat`]{§3.6.1}
17. Apply the self-calibration solutions to the science target [`applycal`]{§3.7}

With improved calibration, it is now time to make an image of the science target(s)!

18. Image the science target(s) [`tclean`]

If the science target is bright enough ($\sim$ 0.1 Jy or brighter and SNR of at least 10 in the image), you may attempt to do self-calibration to improve the calibration further. However, take great care when doing this! Improper self-calibration can lead to blatantly incorrect solutions that create false features in images.

For an excellent discussion of the steps in VLBI calibration, refer to Janssen et al. 2019 ("rPICARD: A CASA-based Calibration Pipeline for VLBI Data", Astronomy & Astrophysics, volume 626, p. A75).

1.4 Taking Notes

It is a *very* good idea to take notes as you work through the calibration process. The specific method is up to you (some prefer to keep a text file open and type things as they go, but others prefer pen & paper). For each step, record the inputs you used and the results. If anything looks odd, if you are unsure about something, or if you realize you made a mistake, write it down in your notes. These notes will be invaluable if you need to go back and start over again, or if someone (e.g., an advisor or collaborator) has a question about your results. The CASA log will record everything that is done (all inputs that were set, all messages that were displayed, etc.), however it is not as easy to read as your own notes. Also, finding exactly where any plots were saved from `plotms` inside the logger is difficult, so remember to name your plots in a consistent way and record the parameter settings that were used in your notes.

1.5 A Brief Discussion of CASA Syntax

For users who are familiar (and comfortable) with AIPS, CASA can be run in a very similar manner. A task can be initialized with `default <taskname>` or `inp <taskname>`. If `default` is used, all of the task's associated parameters will be set to the default values. If `inp` is used, any parameters that have been set previously will retain their current values. Once all of the parameters are set to your liking, you can type `go` or the taskname to run the task. The `tget` function works as in AIPS and will retrieve all parameter values from the last time a task was run. Typing `help <taskname>` will display the help file for that task (to exit the help file, hit "q").

Example 1a: Creating the gain curve calibration table ("AIPS style")

```
default gencal
vis='<your MS filename>'
caltable='<your output gain curve filename>'
caltype='gc'
gencal
```

One of the major advantages of using the `inp` command is that it will display the inputs with a color code. Any inputs that are not allowed will be displayed in red. Allowed inputs will be displayed in blue. However, this will not alert you to any typos or other mistakes in the inputs. For example, if your MS is called “MyMS_1.ms” and you enter `vis='myMS-10.ms'`, CASA will not check that the specified MS actually exists and the `vis` input will display in blue.

Because CASA is python-based, you may also use parentheses when initializing or calling tasks. In other words, typing `default(gencal)` is functionally identical to typing `default gencal`, and typing `gencal()` is functionally identical to typing `gencal`.

One of the major benefits of CASA being python-based is that you can also call and run tasks in one line by entering all the parameters in parentheses. This functionality is particularly useful for writing scripts. The following example will run `gencal` in the exact same way as in Example 1a, but using only one line.

Example 1b: Creating the gain curve calibration table (“scripting style”)

```
gencal(vis='<your MS filename>', caltable='<your output gain curve filename>',
caltype='gc')
```

Throughout this document, the tasks and their associated parameters will be presented in the “scripting style”. This is the preferred way of doing “hands-on” calibration because it reduces the risk of accidentally leaving a variable set from a previous task. Users who opt to use the “AIPS style” (also called “inp/go” or “inp-&-go”) are *strongly* encouraged to make heavy use of the `default` command to reset all variables to their default values before running a task.

Some users like to use a combination of the “AIPS style” and “scripting style”. They first use the `inp` functionality to make sure all of their parameters are set correctly, and then copy their inputs to a script file using the “scripting style” format. It is also possible to keep notes in the script file as comments! This is an excellent way to have a repeatable calibration process, with notes on how things work, all in one place.

Another benefit of CASA being python-based is that you can use the list .append() functionality to add elements to any of the parameters that are lists. One example where this might be useful is the task `applycal`, which users will often run multiple times adding new calibration tables as they are created. If you are using CASA in the “AIPS style” and want to simply add your latest calibration table to the `gaintable` list, you would type:

```
tget applycal
gaintable.append('<new table name>')
```

One final important thing to note is that CASA is case-sensitive, so `'MyBigScaryVLBAProject.ms'` is *not* the same as `'mybigscaryvlbaproject.ms'`.

2 VLBA CASA Calibration Recipe

This recipe assumes that you have downloaded and installed the appropriate version of CASA (preferably 6.3 or higher, but it will also mostly work for 5.8 or 6.2) and retrieved your data from the NRAO archive. The procedure described is for a phase-referenced observation (which is the most common type of observation done on the VLBA).

In general, it is good practice to check the solutions derived by each task before applying them to the data. Most users prefer to inspect the solutions graphically. This can be done using `plotms` (in CASA 5.8, and 6.2 or higher) or `plotcal` (in CASA 5.8). More advanced users may also wish to try the JIVE plotting tool jiveplot (also sometimes called ‘jplotter’); however, users should be aware that there may be compatibility issues depending on the version of casacore. Note that both `plotcal` and jiveplot are non-interactive plotting tools (you cannot change the data being displayed inside the tool), but they will display the data faster than `plotms`.

2.1 Loading Data and Initial Flagging

2.1.1 Import FITS-IDI File

Convert the FITS-IDI file to a CASA Measurement Set (MS) with `importfitsidi`.

```
importfitsidi(fitsidifile='<your filename>.idifits',
  vis='<output filename>.ms',
  constobsid=True,
  scanreindexgap_s=15)
```

The `scanreindexgap_s` parameter is used to reconstruct scan boundaries in those cases where sources do not change between scans. In general, it is good to set `scanreindexgap_s` to some non-zero number to help CASA properly organize the scan list. The recommended value is 15. Shorter times may work as well, but you probably do not want to go much shorter than about 5 seconds. If you find that the resulting MS contains too few scans, run `importfitsidi` again with `scanreindexgap_s` set to a smaller number. If your MS has too many scans, especially multiple scans on the same source when you think it should just be one scan, run `importfitsidi` again with `scanreindexgap_s` set to a larger number.

2.1.2 Inspect the Data

Take a look at the scans, sources, and antennas with `listobs`. It may be useful to write the `listobs` output to a file on disk so you can refer to it later (e.g., when you need to find a single scan on a fringe finder).

```
listobs(vis='<your filename>.ms', listfile='<output filename>.txt')
```

Defining a `listfile` filename will mean that the output is not shown in the logger, only written to the file. If you want to have the `listobs` output in the logger and in a file, you will need to run `listobs` twice; once with `listfile=''`, and once with `listfile` set to your desired filename.

Important things to note in the listobs output are:

- Source names and the associated field numbers
- The number of spectral windows (spw) and their associated channels, frequencies, channel bandwidths, total bandwidths, and polarizations ("Corrs")
- The station names and antenna diameters (all VLBA antennas should have diameters of 25.0 m)

If necessary, correct the antenna tables.

```
ants= ['BR','FD','HN','KP','LA','MK','NL','OV','PT','SC']
diams= [25.0,25.0,25.0,25.0,25.0,25.0,25.0,25.0,25.0]
tb.open('<your filename>.ms/ANTENNA', nomodify=False)
tb.putcol('DISH_DIAMETER', diams)
tb.close()
```

NOTE: All ten VLBA antennas are listed in alphabetical order here, because that is the standard for most VLBA observations. You should check the order and antennas present for your specific observation.

Next, inspect the data with `plotms`.

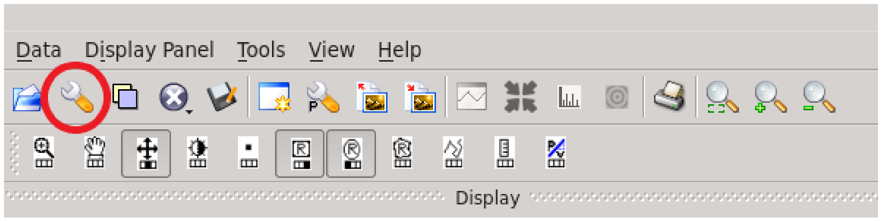


Figure 1: Viewer tool bar; the red circle indicates the "Data Display Options" tool.

```
plotms(vis='<your filename>.ms',
  xaxis='frequency',
  yaxis='amp',
  field='<bright source>',
  antenna='*&*',
  correlation='RR,LL',
  iteraxis='antenna',
  coloraxis='spw')
```

NOTE: Setting `antenna='*&*'` will avoid plotting the autocorrelations. If you want to look at just the autocorrelations (to look for RFI, for example), you would set `antenna='*&&&'`.

You should play around with plotms a bit. Try using the GUI to make changes. Change the axes that are plotted (e.g., try amplitude vs time, phase vs. frequency, amplitude vs phase, etc.). Be warned that `plotms` may crash on you, and you may need to quit and restart CASA to get it to work again.

You may also want to inspect the visibility data using `viewer`. This can reveal issues that are harder to see in plotms. The viewer window can be configured to look very similar to TVFLG in AIPS.

```
viewer(vis='<your filename>.ms', displaytype='raster')
```

NOTE: In the near future, `viewer` will be replaced by `CARTA`.

It may take `viewer` a long time to load, and it will require a significant amount of RAM if you have a large dataset. Once `viewer` has loaded the data, click on Data Display Options (the wrench icon in the upper-left, see Figure 1) to change the view. Change the display axes such that the x-axis is Baseline and y-axis is Time (see Figure 2, left-hand side). Animation axis can be either Channel or Spectral Window, depending on what you want to investigate. Under "flagging options", set the "show flagged regions..." to "Masked to Background" (see Figure 2). As long as the autocorrelations are included, you will also need to adjust the contrast under "basic settings", use the "data maximum" slider to find a good setting for your data. Under "ms and visibility selection", you can change which fields are displayed.

You can also use `viewer` to inspect the data in a manner similar to SPFLG in AIPS by changing the x-axis to "Channel" and the animation axis to "Baseline" (See Figure 2). Unfortunately, unlike SPFLG, you can only inspect a single baseline + spectral window at a time in `viewer`.

NOTE: Occasionally, `viewer` will crash and refuse to start again. The most reliable way to recover from this is to quit CASA and start it again. No data or tables will be lost due this issue. Alternatively, you can run `viewer` outside of the current CASA instance by typing "`casaviewer`" in another terminal window.

2.1.3 Identify a Reference Antenna

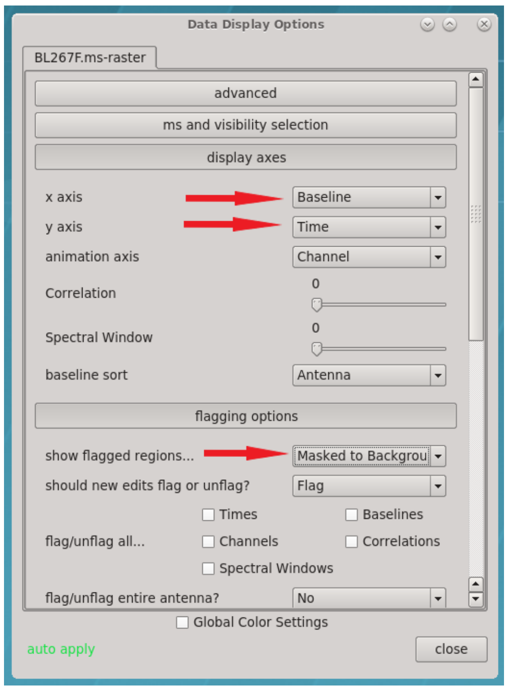
While you are inspecting the data, it is a good time to choose a reference antenna for various tasks during the calibration. Usually, you will want to use one of the more central stations: FD, KP, LA, or PT. To determine which is the best to use as the reference antenna, plot phase vs frequency for a single scan on a bright source (usually the fringe finder or bandpass calibrator). Inspect each of the 4 central antennas and use the one which has the most well-behaved phases. Look at each antenna in both right- and left-hand polarization (don't worry about the cross-hands since they will usually be too low to see anything useful).

```
plotms(vis='<your filename>.ms',
  xaxis='frequency',
  yaxis='phase',
  field='<bright source>',
  scan='<one scan on the bright source with all antennas on source>',
  correlation='LL',
  iteraxis='baseline',
  coloraxis='spw')
```

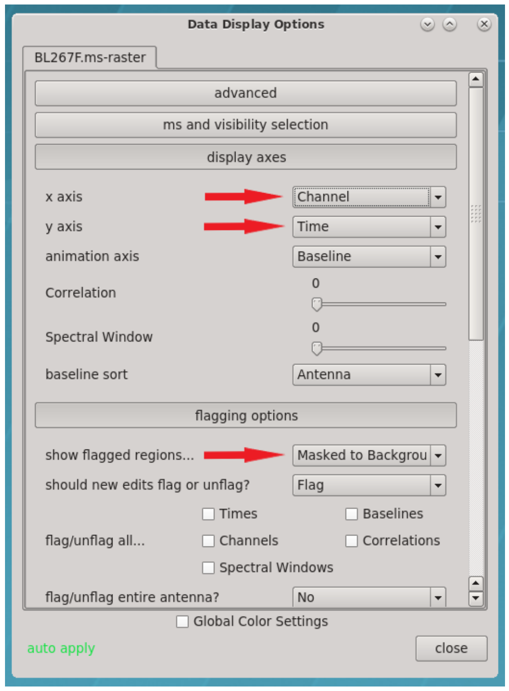
Use the iteration control (green arrows at the bottom of the plot) to look at all the baselines to the selected antenna. Use the GUI controls to change the antenna and correlation to inspect each of the candidate antennas. You should also change the y-axis to amplitude to inspect the bandpass shape on each baseline and look for RFI. The ideal reference antenna will have smoothly varying phases (preferably with few wraps) and no RFI spikes in amplitude. Also, do not to use an antenna that drops out during the course of the observation as the reference antenna. Be sure to make a note about which antenna you decide to use for the reference antenna.

2.1.4 Identify a Good Scan/Timerange for the Instrumental Delay

During the calibration, you will need to provide a scan or timerange for the instrumental delay (a.k.a., "single band delay" or "fringe finding"). While you are inspecting the data, take a moment to identify a good candidate scan. Ideally, you want a time range on the fringe finder where all antennas are on source.



Data Display Options
BL267F.ms-raster
advanced
ms and visibility selection
display axes
x axis: Baseline
y axis: Time
animation axis: Channel
Correlation: 0
Spectral Window: 0
baseline sort: Antenna
flagging options
show flagged regions...: Masked to Backgrou
should new edits flag or unflag?: Flag
flag/unflag all...: ☐ Times ☐ Baselines ☐ Channels ☐ Correlations ☐ Spectral Windows
flag/unflag entire antenna?: No
☐ Global Color Settings
auto apply
close



Data Display Options
BL267F.ms-raster
advanced
ms and visibility selection
display axes
x axis: Channel
y axis: Time
animation axis: Baseline
Correlation: 0
Spectral Window: 0
baseline sort: Antenna
flagging options
show flagged regions...: Masked to Backgrou
should new edits flag or unflag?: Flag
flag/unflag all...: ☐ Times ☐ Baselines ☐ Channels ☐ Correlations ☐ Spectral Windows
flag/unflag entire antenna?: No
☐ Global Color Settings
auto apply
close

Figure 2: Viewer Display Options menu. Left: options selected to mimic TVFLG. Right: options selected to mimic SPFLG

You can find the scan numbers for each scan on the fringe finder by looking at the output of `listobs`. Inspect the separate scans on the fringe finder using `plotms`.

```
plotms(vis='<your filename>.ms',
  xaxis='frequency',
  yaxis='amp',
  field='<fringe finder>',
  antenna='<refant>',
  scan='<one scan on the fringe finder>',
  correlation='RR,LL',
  iteraxis='baseline',
  coloraxis='corr')
```

Step through all the baselines and check that each other antenna has data to the reference antenna for that scan. If there is more than one scan on the fringe finder, inspect each scan one at a time using the GUI to change the scan number. In most cases, you will want to specify a one-minute timerange within the scan. Once you find a scan that you think will work, try limiting the timerange in the GUI to find a one-minute interval with good data. Make a note of this timerange so you can use it later in the calibration (§2.4.1).

2.1.5 Flagging

"Flagging" is the term radio astronomers use to describe the process of getting rid of bad or questionable data. The term "flag" is used rather than "erase" because the data are not being deleted, just marked so the program does not to use it (i.e., the flagged data are still there, but the program ignores them).

The most reliable and least (potentially) destructive flagging tool in CASA is `flagdata`.

Quacking (optional)

NOTE: Quacking is an optional step, and you can skip it if your data do not appear to need it.

It is often a good idea to get rid of the first and last bits of each scan of a VLBA observation. This is called "quacking" in radio astronomer jargon[2].

Flag the first 4 seconds of each scan using `flagdata`.

```
flagdata(vis='<your filename>.ms',
  mode='quack',
  quackinterval=4.0,
  quackmode='beg',
  quackincrement=True)
```

Flag the last 4 seconds of each scan using `flagdata`.

```
flagdata(vis='<your filename>.ms',
  mode='quack',
  quackinterval=4.0,
  quackmode='endb',
  quackincrement=True)
```

[2]There is some uncertainty about the origins of the term "quack". However, some historical research (i.e., talking with people who have been doing radio astronomy for a lot longer than I have) indicates it has nothing to do with waterfowl. Instead, "quack" refers to an unscrupulous/incompetent physician who treats the symptoms of a disease without treating the disease itself. The original QUACK routine was written for the VLA DEC-10 computers and flagged the beginnings of scans because they often contained bad data, but nobody could figure out what was causing the bad data.

Automated Flagging

The `flagdata` task includes a useful automated flagging option known as "TFCrop".

To begin, you just want to get a feel for what TFCrop will do to the data, which means you should set `action='calculate'`.

```
flagdata(vis='<your filename>.ms',
  mode='tfcrop',
  datacolumn='data',
  timecutoff=4.0,
  freqcutoff=3.0,
  action='calculate',
  display='both')
```

NOTE: If you are working on a spectral line project, you may want to set `flagdimension='time'` to avoid accidentally flagging any spectral features of interest.

The top row of the GUI will display the data as it currently is. The bottom row shows what the data will look like after the flags are applied. Step through the antennas by clicking "Next Baseline". Step through the spectral windows by clicking "Next SPW'. Step through at least a few scans by clicking "Next Scan". If you think the automatic flagging is being too aggressive, click "Quit" and adjust the threshold parameters `timecutoff` and `freqcutoff`. Setting the threshold parameters to larger values should result in fewer flags. Note that TFCrop will often aggressively flag the spectral channels at the edges of each spectral window. If you would like to prevent this, you can set `flagdimension='time'` to avoid any flagging on the frequency axis. Re-run `flagdata` until you are satisfied that the threshold values are set properly for your observation.

Once you are satisfied with the threshold settings, you will want to apply the flags.

```
flagdata(vis='<your filename>.ms',
  mode='tfcrop',
  datacolumn='data',
  flagdimension=<'time' if you want to avoid flagging edge channels>
  timecutoff=<the value you determined>,
  freqcutoff=<the value you determined>,
  action='apply',
  display='both')
```

As before, step through a few spectral windows and baselines to make sure that the flags are appropriate. If you need to adjust the threshold values before applying the flags, click "Quit". Once you are satisfied that the flagging will do what you want, click "Stop Display". The program will then apply the flags.

It is a good idea to take another look at the data with `plotms` at this point to see how TFCrop did.

For more details on using TFCrop, including a tutorial, see the Automatic RFI Excision section of the VLA Flagging CASA Guide. Note that the VLA tutorial recommends setting `flagbackup=False` when running TFCrop. For VLBI projects, it is recommended to back up the flags whenever possible, so setting `flagbackup=True` would be more appropriate.

Flagging "By Hand"

As telescope arrays become larger and more complex, automated flagging will become the primary means of excluding bad data (just look at what ALMA does). However, the current VLBA is still small enough that inspecting and flagging data oneself is not too painful.

You can look for obvious bad data using `plotms`. Be aware that while you can flag data in `plotms`, a major drawback to using `plotms` to flag data is that it makes undoing flags very difficult. It is recommended to be somewhat careful with flagging early on in the calibration process and only flag those data points that are obviously bad.

To start, take a look at the autocorrelations on the fringe finder to see if TFCrop missed any obvious RFI.

```
plotms(vis='<your filename>.ms',
  xaxis='frequency',
  yaxis='amp',
  field='<fringe finder>',
  antenna='*&&&',
  correlation='RR,LL',
  iteraxis='antenna',
  coloraxis='spw')
```

Next, take a look at the cross-correlations.

```
plotms(vis='<your filename>.ms',
  xaxis='uvwave',
  yaxis='amp',
  field='<one source>',
  antenna='*&*',
  correlation='RR,LL',
  iteraxis='antenna',
  coloraxis='spw')
```

You should inspect one source at a time, focusing on the brighter calibrators. Dimmer science targets may just look like noisy junk at this point, so it is recommended to wait until some calibration has been applied before flagging them (although some spurious high amplitude bad data may be obvious from the start). Use the GUI to select different sources. It is often good to start by looking at amplitude vs uv distance in units of wavelength (often called a "uv plot" or a "radplot"), but it may also be useful to look at amplitude vs time, amplitude vs frequency, phase vs uv distance, phase vs time, and amplitude vs phase.

Changing the `avgtime` can help identify outliers and decrease the time it takes to plot the data.

!! WARNING !!

DO NOT USE `plotms` TO FLAG AVERAGED DATA!

There is currently a problem with the way `plotms` handles flagging averaged data that may lead to previously flagged data becoming unflagged. It may also cause improper values ('nan') to be written to the MS, especially if `plotms` crashes during flagging. If bad data is only obvious in `plotms` when averaged, use `plotms` to identify the problematic data and then create the flags using `flagdata`. This is a longer, more cumbersome approach to flagging, but it will ensure that the MS is not corrupted.

According to the CASA documentation, flagging can also be done in `viewer`. However, `viewer` is a bit unstable and the flagging does not always function properly. As with `plotms`, it is recommended to use `viewer` to identify the bad data, and then use `flagdata` to create the flags.

Keep in mind that if you end up flagging a large amount of data on the station you chose for the reference antenna (§2.1.3), you will probably need to pick a different reference antenna. It may also be necessary to pick a different timerange for the instrumental delay calibration if you ended up flagging too much data on the one you picked previously (§2.1.4).

2.2 Amplitude Corrections from Autocorrelations

Determine the amplitude corrections from auto-correlations with `accor`.

```
accor(vis='<your filename>.ms',
  caltable='<your filename>.accor',
  solint='30s')
```

NOTE: The solution interval (`solint`) may need to be longer for some frequencies.

Remember to inspect the solutions as they are created. In CASA 6.2 and higher, you will need to use `plotms`.

Inspecting calibration solutions using `plotms`:

```
plotms(vis='<your filename>.accor',
  xaxis='time',
  yaxis='amp',
  iteraxis='antenna')
```

You will need to use the green arrows at the bottom of the `plotms` GUI to see the solutions for the next (or previous) antenna.

In CASA 5.8, it is faster to use `plotcal` to plot solutions.

Inspecting calibration solutions using `plotcal` (for CASA 5.8 only):

```
plotcal(caltable='<your filename>.accor',
  xaxis='time',
  yaxis='amp',
  field='',
  subplot=521,
  iteration='antenna')
```

Setting the `subplot` variable will tell `plotcal` to display several plots on a single page. The first number in `subplot` is the number of rows you want to plot. The second number is the number of columns. So, setting `subplot=521` will display 10 plots: 5 rows with 2 plots in each row.

When inspecting the `accor` solutions, you would like to see all of the solutions within a few percent of 1.0. Any egregious outliers should be flagged. Flagging solutions can be done inside `plotcal` or `plotms`, or can be done using `flagdata`.

It should be noted that the AIPS VLBA utility script VLBACCOR smooths the autocorrelation corrections by default (with a smoothing time of 30 minutes). It is possible to do this smoothing in CASA 6.3 and later using the `smoothcal` task.

```
smoothcal(vis='your filename>.ms,
  tablein='<your filename>.accor',
  caltable='<your filename>_smooth.accor',
  smoothtype='median',
  smoothtime=1800.0)
```

NOTE: It is *not* possible to smooth the `accor` solutions in CASA 5.8 or 6.2.

2.3 *A Priori* Calibration

Generate the system temperature table with gencal.

```
gencal(vis='<your filename>.ms',
  caltable='<your filename>.tsys',
  caltype='tsys',
  uniform=False)
```

Remember to inspect the calibration tables with plotms or plotcal. For the system temperature table, plot the solutions with yaxis='amp' and look at both xaxis='freq' and xaxis='time'. Look for any egregious outliers and flag them.

Generate the gain curve table with gencal.

```
gencal(vis='<your filename>.ms',
  caltable='<your filename>.gcal',
  caltype='gc')
```

You usually will not need to inspect the gain curve corrections.

2.4 Delay, Rate, and Phase Calibration

The next two steps in the calibration process both involve the fringefit task.

2.4.1 Instrumental Delay

Solve for the instrumental delays by using fringefit on a bright source (the "fringe finder"). Set the timerange to the time span you identified while inspecting the data earlier (§2.1.4).

```
fringefit(vis='<your filename>.ms',
  caltable='<your filename>.sbd',
  field='<fringe finder>',
  timerange='<one minute on fringe finder with data on all baselines>',
  solint='inf',
  zerorates=True,
  refant='<the reference antenna; usually FD, LA, PT, or KP>',
  minsnr=10,
  gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys'],
  interp=['nearest', 'nearest', 'nearest,nearest'],
  parang=True)
```

Ideally, the SNR for each station should be very high ($\gg 10$). Watch the logger for any reports of low SNR and failures to converge on a solution.

Apply the instrumental delay corrections using applycal.

```
applycal(vis='<filename>.ms',
  gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
    '<filename.sbd>'],
  interp=['nearest', 'nearest', 'nearest,nearest', 'nearest'],
  parang=True)
```

NOTE: In `applycal`, the order in which you specify the list for `gaintable` sets the order for both `interp` and `spwmap`. If you change the order of `gaintable`, be sure to also change the order of `interp` and `spwmap`! Also, if you smoothed any of the solutions, be sure to use the appropriate filename for the smoothed table.

WARNING: If any station failed to converge on a solution in `fringefit`, it will be flagged with `applycal` unless you specify `applymode='calonly'`. So, if one of your antennas suddenly "disappears" from the data, check that it had good solutions in `fringefit`.

After the solutions have been applied, take a look at the data with `plotms`.

```
plotms(vis='<your filename>.ms',
  xaxis='frequency',
  yaxis='phase',
  ydatacolumn='corrected',
  field='<fringe finder>',
  timerange='<same timerange as used in fringefit above>',
  correlation='rr,ll',
  antenna='*&*',
  iteraxis='antenna',
  coloraxis='antenna2')
```

When looking at the phases in the `data` column of the MS, you may see that the phases have steep slopes and may have jumps between spectral windows. Using the `plotms` GUI, change the `ydatacolumn` to `corrected`. This will display the phases with the instrumental delay corrections applied. The phases should be centered on 0 degrees and they should be mostly flat with no jumps between spectral windows. If the corrected phases still look gross, something has gone wrong! For example plots of phase vs frequency before and after the instrumental delay, see the DARA EVN tutorial Part 1, Section 5A.

2.4.2 Global Fringe Fitting

Solve for the time and frequency-dependent effects in phase using `fringefit`, also known as "global fringe fitting" (or sometimes "multi-band delay"). Pick a solution interval that is appropriate for your data. It should be at least 10 seconds, and no longer than the scan length on the phase reference calibrator(s). Try starting with either 30s or 60s. For `refant`, enter a list of antennas to try as the reference antenna. The preferred `refant` should be listed first, followed by the second choice, then third choice, and so on. It is not recommended to include Mauna Kea (MK) or Saint Croix (SC) in the list, unless the phase reference calibrator is very bright on the longest baselines. Set `field` to a list of bright sources, usually the fringe finder and phase reference calibrator.
WARNING: This step may take a *very* long time to run ($\sim$ 10 minutes to several hours, depending on the size of the dataset and your computer's hardware). It may also fail a few times until you find a good solution interval (CASA is currently extremely picky about the `solint`, but the developers are working on that). Try not to throw your computer through the wall – it is CASA's fault, not your computer's.

```
fringefit(vis='<your filename>.ms',
  caltable='<your filename>.mbd',
  field='<list of bright sources>',
  solint='<your solution interval, e.g. 60s>',
  minsnr=5,
  zerorates=False,
  refant='<recommendation: FD,PT,LA,KP,OV,NL,BR,HN>',
  combine='spw',
  gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
    '<filename>.sbd'],
  interp=['nearest', 'nearest', 'nearest,nearest', 'nearest'],
  parang=True)
```

NOTE: If this step fails and you see error messages like "Can't do 2-dimensional FFT on a single timestamp! Please consider changing solint to avoid orphan timesteps", try using a `solint` that is exactly the scan time divided by an integer (2, 3, or 4 are recommended). If that does not fix the problem, you may have to set the `solint` to `'inf'`.

When the `fringefit` task is done, check the logger for the solution statistics ("expected/attempted/succeeded"). You want all three numbers to be the same, or at least very similar. You may need to try longer solution intervals to get it to work optimally.

NOTE: If your phase reference calibrator is bright enough (about 0.1 Jy or brighter on all baselines to the reference antenna) or if you are only observing very bright sources, you can do the global fringe fit on each spectral window separately. In that case, set `combine=''`. Be prepared for `fringefit` to take even longer to run in this manner.

After `fringefit` has successfully completed and you are satisfied that the number of solutions is appropriate, take a look at the solutions with `plotms` (or `plotcal`, if you are using CASA 5.8) Use `xaxis='time'`, and switch between `yaxis='delay'`, `yaxis='phase'`, and `yaxis='delayrate'`. The delay and phase solutions should both vary smoothly with time. If they do not, you may need to smooth the table before applying it. The rates should be centered on zero with some scatter.

When you are confident that the global/multi-band solutions are good, apply them with `applycal`.

```
  applycal(vis='<your filename>.ms',
    gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
      '<filename>.sbd', '<filename>.mbd'],
    interp=['nearest', 'nearest', 'nearest,nearest', 'nearest', 'linear'],
    spwmap=[[], [], [], [], <number of spw per polarization>*[0]],
    parang=True)
```

It will probably take a while (∼ 5 minutes or longer) for `applycal` to run this time. It is necessary to set the `spwmap` because you combined all of the spectral windows to get the `fringefit` solutions (hence "multi-band" delay). To properly assign the `fringefit` solutions to each spectral window, you need to tell CASA to use the same solution for all spectral windows.

NOTE: If you determined fringe fit solutions for the spectral windows separately, just use `spwmap=[]`.

Take a look at the data with `plotms` to make sure the corrections are improving the phases. For example plots of the global fringe fit solutions and the phase vs frequency before and after the global corrections, see the DARA EVN tutorial Part 1, Section 5B.

NOTE: It is possible to run `fringefit` multiple times, selecting a subset of the spectral windows each time. For more details on doing the global fringe fit in this way, see §4.6.

2.5 Bandpass Calibration

Correct for the shape of the bandpass with `bandpass`. You will need a bright source for this step, which is referred to as the "bandpass calibrator". Many observations will use the same source for bandpass calibration and fringe finding.

```
  bandpass(vis='<your filename>.ms',
    gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
    '<filename>.sbd', '<filename>.mbd'],
    interp=['nearest', 'nearest', 'nearest,nearest', 'nearest', 'linear'],
    spwmap=[[], [], [], [], <number of spw per polarization>*[0]],
    parang=True,
    field='<bandpass calibrator>',
    solnorm=True,
    solint='inf',
    refant='<reference antenna>',
    bandtype='B',
    caltable='<your filename>.bpass')
```

NOTE: Use all of the channels in each spectral window to determine the bandpass solutions. DO NOT limit the bandpass solutions to the center 75% of the spectral windows.

Use `plotms` or `plotcal` to check that the bandpass solutions (both amplitude and phase) look similar to the bands themselves (i.e., similar to what you see when you plot amplitude or phase vs frequency). Because CASA applies solutions by dividing visibilities by solutions (rather than multiplying the two, as AIPS does), the solutions should look similar to the data itself.

Apply bandpass solutions with `applycal`.

```
  applycal(vis='<your filename>.ms',
    field='',
    gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
      '<filename>.sbd', '<filename>.mbd', '<filename>.bpass'],
    interp=['nearest', 'nearest', 'nearest,nearest', 'nearest', 'linear',
      'linear,linear'],
    spwmap=[[], [], [], [], <number of spw per polarization>*[0], []],
    parang=True)
```

Because you are applying the global fringe fit calibration again here, this step will probably take a while ($\sim$ 5 minutes or longer) again.

2.5.1 Check the Bandpass Correction and Do More Flagging

After applying the `bandpass` solutions, look over the data with `plotms`.

```
  plotms(vis='<your filename>.ms',
    xaxis='frequency',
    yaxis='amp',
    field='<the source used in bandpass>',
    antenna='*&*',
    correlation='RR,LL',
    iteraxis='antenna',
    coloraxis='spw')
```

For example plots of the bandpass solutions (for both amplitude and phase) and the data before and after the bandpass corrections are applied, see the DARA EVN tutorial Part 1, Section 6.

The bandpass calibration often does not perfectly calibrate the channels at the beginning and end of each spectral window. It is often a good idea to get rid of the edge channels that are not well-calibrated. If you notice that the edges of the spectral windows look at bit nasty (much higher than the rest of the band), feel free to flag those channels. This flagging can be done in `plotms`, but it is often easier (and more reliable) to do it in with `flagdata`.

The following example illustrates how to flag the edge channels with `flagdata`. It assumes you have 128 channels per spw and you want to get rid of the first and last 3 spectral channels of each spw.

```
flagdata(vis='<your filename>.ms', spw='*:0~2;125~127')
```

Check the results with `plotms`. Many experienced VLBA observers will not have any second thoughts about cutting out 8 or 12 channels on each side of the spw for continuum observations. If you start with flagging 3 channels on each side and think the amplitude vs frequency plots still look pretty gross, feel free to flag some more.

It is a good idea to also look over the bandpasses for your phase reference calibrator and any check sources. You will probably want to select just a few scans to inspect for each source, but try to look over things at various stages of the observation. If you notice that RFI is still present in the data, it may be useful to run the second automatic flagging routine, "RFlag". Running RFlag is very similar to running TFCrop (see § 2.1.5 above). First, run `flagdata` in RFlag mode with `action='calculate'` to check that the program will flag everything you want and leave the good data alone.

```
flagdata(vis='<your filename>.ms',
  mode='rflag',
  datacolumn='corrected',
  action='calculate',
  display='both')
```

If your project involves spectral lines, you may want to set the `spw` parameter so that the RFlag routine will ignore the channels containing lines. Step through the baselines and spw to see what flags are being calculated. Adjust the parameters `freqdevscale` and `timedevscale` to change how aggressively the program will flag the data; both default to a value of 5.0, and lowering the value will result in more flags. If the RFlag routine is not performing as expected, try setting the `uvrange` to only include the longer baselines (high flux on short baselines can bias the statistics RFlag uses to determine what needs to be flagged). Once you are satisfied RFlag will properly flag the data, run `flagdata` with `action='apply'` to actually create the flags and remove the bad data.

NOTE: The RFlag automatic flagging routine should only be run on data for which the bandpass has been properly calibrated!

For more information on RFlag, see the Automatic RFI Excision section of the VLA Flagging CASA Guide.

2.6 Final Amplitude Scaling and Flux Calibration

If your observation was done using wide bandwidths ($\Delta\nu \geq 256$ MHz) on the VLBA, which is any observation done a bit rate of 2 Gbps or more, you will need to do one extra step at this point. The flux density scale for wideband VLBA observations can be off by up to 30% (although it usually only off by a few percent) if the calibration does not correctly account for the wide bandpasses. To do this, you need to run `accor` again after the bandpass calibration has been applied. The AIPS task that was developed to address this issue is called ACSCL. For more details on this topic, and how it was handled in AIPS, see Walker 2014 (VLBA Scientific Memo #37).

Prior to running `accor` this time, it is strongly recommended that users inspect the calibrated data and determine whether any channels will need to be excluded from imaging or other post-processing. Edge channels may need to be excluded if the bandpass calibration did not properly correct the band at the edges. From Walker 2014, the standard recommendation is to use the inner $\sim 75\%$ of channels for PFB observations and the inner $\sim 89\%$ of channels for DDC observations. Any channels that are suspected to contain RFI should also be excluded. The actual channels used will depend on the individual observation and science goals.

Final re-scaling of the auto-correlation amplitudes with `accor`.

NOTE: DO NOT APPLY THE SYSTEM TEMPERATURE OR GAIN CURVE TABLES WHEN DERIVING THESE SOLUTIONS.

```
accor(vis='<your filename>.ms',
  spw='<all channels that will be used for imaging>',
  caltable='<your filename>.acscl',
  solint='2min',
  gaintable=['<filename>.accor', '<filename>.sbd', '<filename>.mbd',
    '<filename>.bpass'],
  interp=['nearest', 'nearest', 'linear', 'linear,linear'],
  spwmap=[[], [], <number of spw per polarization>*[0], []])
```

The AIPS VLBA utility script VLBAAMP smooths the autcorrelation corrections by default in exactly the same way as VLBACCOR (see §2.2). If you would like to replicate the AIPS method, you can use the `smoothcal` task as long as you are using CASA 6.3 or higher.

```
smoothcal(vis='your filename>.ms,
  tablein='<your filename>.acscl',
  caltable='<your filename>_smooth.acscl',
  smoothtype='median',
  smoothtime=1800.0)
```

Apply the final amplitude solutions with `applycal`. You should apply the system temperature and gain curve tables in this step.

```
applycal(vis='<your filename>.ms',
  field='',
  gaintable=['<filename>.accor', '<filename>.gcal', '<filename>.tsys',
    '<filename>.sbd', '<filename>.mbd', '<filename>.bpass', '<filename>.acscl'],
  interp=['nearest', 'nearest', 'nearest,nearest', 'nearest', 'linear',
    'linear,linear', 'nearest'],
  spwmap=[[], [], [], [], <number of spw per polarization>*[0], [], []],
  parang=True)
```

NOTE: If you smoothed any of the solutions, be sure to enter the appropriate filenames for the smoothed tables in `applycal`.

The data should be mostly calibrated at this point. At the very least, you should be able to make images of the calibrators.

2.7 Split Out Calibrated Data

It is generally recommended to split the measurement set after the initial calibration is complete. If you have multiple science targets, you should create a new MS for each science target + phase reference calibrator pair by setting the `field` paremeter to the appropriate values. Even if your observation only involved a single target, it is a good idea to split the MS once the initial calibration is complete. This will preserve the initially-calibrated data in case you make a mistake in any of the next steps and need to start over. Think of it as a “save point” in a video game (do you really want to have to go back to the beginning of the game when you could just start from the beginning of the current level?).

Split the calibrated MS using `split`.

```
split(vis='<your filename>.ms',
  outputvis='<your split filename>.ms',
  field='<all fields you want in the new MS>',
  spw='<all specrtal windows and channels that you want for the next steps>',
  antenna='*&*',
  datacolumn='corrected')
```

You can name the new MS whatever you want, but using a naming scheme that keeps track of where it was in the calibration process will make your life easier if you make mistakes down the line. For example, you could use filenames such as "`<project code>_cal1.ms`", or "`<science target>_cal1.ms`" if you have more than one science target. Setting `antenna='*&*'` will leave the autocorrelations out of the new MS. Also, note that you can select only the spectral windows and channels you want to keep using the `spw` parameter. If the edge channels are not that great (and especially if you did not use them in the preceding `accor` step), you can exclude them from the split file and save a little hard drive space.

3 Self-Calibration of VLBA Data In CASA

The initial calibration should do a good job of getting the data to the point where you can make images of the fringe finder and phase reference calibrator(s). However, VLBA observations almost always require self-calibration to get the data to the point where one can actually do scientific analysis. At the very least, observers should expect to do self-calibration on the phase reference calibrator(s) and apply those solutions to the science target as well. If the science target is bright enough for self-calibration, it will greatly help reduce noise in the image and may reveal dim structures that were not noticeable before. However, observers should be *extremely* cautious when performing self-calibration. It is possible to introduce fake structures into an image, especially with amplitude self-calibration.

Observers should always start with phase self-calibration before moving on to amplitude self-calibration. A dataset may require multiple iterations of phase self-calibration before it is ready for amplitude self-calibration.

3.1 Tracking Improvement

To ensure that self-calibration is actually helping to improve the quality of the images, you should track some image statistics. The most reliable way to do this is to determine 2 boxes in the image; one box that contains the source, and one box that does not contain any part of the source ("off-source"). You can use `viewer` to make some boxes, and then record the pixel locations of the lower left and upper right corners.

Once you have your on- and off-source boxes defined, use the task `imstat` to gather some statistics about those regions.

```
      imstat(imagename='<image of interest>',
        box='<lower left corner x pixel>,<lower left corner y pixel>,
             <upper right corner x pixel>,<upper right corner y pixel>')
```

Running `imstat` will return several values in the logger. The values of interest for the self-calibration process are the on-source peak value, the on-source total flux density, and the off-source RMS value. It is a *very* good idea to record each of these values for each image (including the first image without any self-calibration applied) as you proceed with self-calibration. Ideally, the on-source peak and total flux values will increase slightly and the off-source RMS will decrease significantly as you improve the calibration. However, be very suspicious of large increases ($> 10\%$) in flux density values, especially when the off-source RMS does not improve dramatically.

3.2 Self-Calibration Concept

Because the VLBA antennas are separated by such large distances, the phases may not be perfectly calibrated by fringe fitting alone. This is partly because each antenna may be looking through vastly different troposphere/ionosphere, and partly because the phase reference calibrators are rarely point sources on long baselines (and fringe-fitting usually assumes the calibrators are point sources on all baselines).

The process of self-calibration involves building models of the sources, refining the calibration based on those models, and then repeating until you converge to a good solution. The models are created during the imaging process. Each iteration self-calibration should produce images with lower noise.

Each time the noise is reduced, you may notice dimmer structures appear. This is where you need to be careful. Including an apparent structure in a model for self-calibration can "build in" features that are not real. A good rule of thumb is "if you can't self-calibrate a structure away, it is probably real". So, if you are uncertain whether a newly apparent component is real or not, first try to self-calibrate without including that component in the model. If it sticks around, it is probably a real structure and including it in the next model should further improve the calibration and lower the noise in the next image.

The first step in phase self-calibration is to create an image/model of a source, usually the phase reference calibrator. Fringe finders and bandpass calibrators are easy to self-calibrate (because they are so bright), but they are usually only observed a few times throughout the observation so any solutions derived from self-calibrating on them will not have a large impact on the overall calibration of your science target.

3.3 Some More Flagging

Before imaging, it is usually a good idea to flag the calibrated data a little more aggressively than when you first started the calibration process. Remember the radio astronomers' motto: "Bad data is worse than no data!" (If you really mess up and flag something you shouldn't, you can always go back to the initially-calibrated MS and split out a fresh MS for self-calibration.)

Inspect and flag the phase reference calibrator data using `plotms`.

```
plotms(vis='<your split filename>.ms',
  xaxis='uvwave',
  yaxis='amp',
  field='<phase reference calibrator>',
  correlation='rr,ll')
```

You will probably need to look at the data in several ways. Plotting amplitude vs uv distance should get you off to a good start. Flag anything that is obviously too high (way above anything else). Also, try plotting different axes: amplitude vs time, amplitude vs frequency, phase vs uv distance, phase vs time, and amplitude vs phase. Once you are pretty sure that you have gotten rid of most of the bad data, exit `plotms`. Keep in mind that it will be difficult to undo any of the flags you create using `plotms`, so take care not to flag something that might be real. You can always do more flagging between iterations of self-calibration. It may help to average the spectral channels together (`avgchannel`) to show where the bad junk really is. However, keep in mind that *you should not flag any averaged data* in `plotms`. Instead, use `plotms` to identify the problematic averaged data and then create the flags in `flagdata`.

3.4 Make An Image of the Phase Reference Calibrator

Now that the phase reference calibrator is looking more respectable, it is time to make an image. You need to determine a few imaging parameters before starting the interactive imaging process.

To determine the `cell` size for the image, you need to get the maximum uv distance in our data. You can do this by plotting the amplitude vs uv distance (in units of wavelength) in `plotms`, as you did for the flagging.

The formula to estimate the appropriate `cell` size in arcseconds is:

$$\text{cell size} \approx \frac{206265}{N_s D_{max}[\lambda]} \; \textit{arcseconds} \tag{1}$$

where $D_{max}[\lambda]$ is the longest baseline in units of observed wavelength and N_s is the Nyquist sampling factor.

Typically, you want the N_s to be about 5 or 6. As an example, if our maximum baseline is about 150 mega-wavelengths (1.5×10^8 wavelengths), you should get the `cell` size to be between about 2.29×10^{-4} and 2.75×10^{-4} arcseconds, or 0.229 and 0.275 milliarcseconds (usually abbreviated "mas"). However, this assumes a Gaussian beam shape, which is often not true for VLBI observations. It is usually a good idea to use $N_s = 6$ in the calculation and then round down a bit. For example, if you calculated a `cell` size of 0.229 mas, you should round it down to 0.2 mas.

Many programs (like difmap and AIPS) require that the image size be a power of 2 (256, 512, 1024, etc.). CASA does not have this requirement, but it is recommended that the image size be even and divisible by 2,3,5, and 7 only. A good rule of thumb is to start with a power of 2 and then multiply by 10. So, sizes like 320, 640, 1280, etc. would all work well. For most phase reference calibrators, an image size of 640 should be fine. (You can always change it, if you need a bigger or smaller image.)

To choose which deconvolver to use, you will need to calculate the fractional bandwidth of the observation. The fractional bandwidth is the total per-polarization bandwidth divided by the center frequency. If your fractional bandwidth is less than about 0.1, you can use the `clark` or `hogbom` deconvolver. If the fractional bandwidth is about 0.1 or larger, you should probably use the multi-term, multi-frequency synthesis deconvolver (`mtmfs`). As an example, assume you were observing in 4 Gbps mode, dual polarization, at a center frequency of 5 GHz. Your total per-polarization bandwidth would be 512 MHz (or 0.512 GHz). Therefore, your fractional bandwidth would be $0.512/5.0 = 0.1024$, which is just large enough to worry about deconvolution errors creeping into the image. So, you should probably use the `mtmfs` deconvolver. A guide to using `tclean` with the `mtmfs` deconvolver can be found in the VLA Imaging CASA Guide.

NOTE: If a source is compact (point-like) and the fractional bandwidth is ~ 0.1, you probably will not see any major differences between the `hogbom`, `clark`, and `mtmfs` deconvolvers. However, if the source is extended or complex (e.g., has a long jet) and/or the fractional bandwidth is $\gtrsim 0.2$ (e.g., two widely-spaced sidebands that you are imaging together), the `mtmfs` deconvolver will produce vastly superior images.

For the weighting, VLBA continuum observers will often use pure natural weighting (increasing the weight on shorter baselines making the image more sensitive to diffuse material). However, you can try using some of the `briggs` robust weighting schemes, but you will probably want to set `robust` somewhere between 0 and +2 (i.e., more on the natural weighting side).

For continuum imaging, you will most likely want to keep the default `stokes='I'` (total intensity image). Set `niter`, the number of iterations, to be some arbitrarily large number; at least a few hundred, but 1000 is usually a good choice. For VLBI observations (and any other interferometric observation where there is a reasonable expectation that the target will not be a point source), set `interactive=True` to allow you to define areas where there is real flux from the source.

Make an image of the phase calibrator with `tclean`.
(This example assumes that using the `clark` deconvolver is justified.)

```
tclean(vis='<your split filename>.ms',
  field='<phase reference calibrator>',
  imagename='<base name for image files>',
  imsize=[640],
  cell=['<cell size>'],
  stokes='I',
  deconvolver='clark',
  weighting='natural',
  niter=1000,
  interactive=True,
  savemodel='modelcolumn')
```

NOTE: For an excellent illustration of what is happening during the imaging process, see the DARA EVN tutorial Part 2, Section 3A.

Once you are done cleaning, `tclean` will produce several files: .image, .mask, .model, .pb, .psf, .residual, and .sumwt. If you used the `mtmfs` deconvolver, you will also see .alpha and .alpha.error files relating to the spectral index, and various .tt# files relating to the different scales.

Remember to get the image statistics with `imstat` and record them somewhere so you can track how things change with self-calibration.

3.5 Phase Self-Calibration

Once you have created an initial model of the source (which the above example saves to the MS model column), you can begin to self-calibrate.

NOTE: If you forgot to set `savemodel='modelcolumn'`, you can create a model from the image using the `ft` task. See §4.7 for an example of running `ft`.

Refine the delays on the phase reference calibrator using `gaincal`.

```
gaincal(vis='<your split filename>.ms',
  field='<phase reference calibrator>',
  caltable='<calibration table name>.dcal',
  solint='inf',
  refant='<reference antenna>',
  minblperant=3,
  gaintype='K',
  calmode='p',
  parang=False)
```

It is usually a good idea to name the calibration tables with the source name that is being calibrated. For example, "DA193.dcal", where "dcal" stands for "delay self-calibration". Note that `parang` is set to `False` because we already accounted for the parallactic angle when we split the data from the initially calibrated MS.

Refine the phases on the phase calibrator using `gaincal`.

```
gaincal(vis='<your split filename>.ms',
    field='<phase reference calibrator>',
    caltable='<calibration table name>.pcal',
    solint='60s',
    refant='<reference antenna>',
    minblperant=3,
    gaintype='G',
    calmode='p',
    solmode='',
    gaintable=['<the calibraton table from delay self-cal>'],
    interp=['linear'],
    parang=False)
```

The solution interval ("solint") should be set as short as possible. For most phase reference calibrators, this will be somewhere between 20 and 60 seconds. Dimmer source may require longer solution intervals. Watch the CASA logger for the SNR on each baseline and the number of good solutions found. If the SNR is below the minimum (the default minimum SNR is 3.0) or the number of good solutions is significantly smaller than the expected number of solutions, try using a longer solution interval.

Apply the delay and phase solutions to the phase reference calibrator with applycal.

```
  applycal(vis='<your split filename>.ms',
    field='<phase reference calibrator>',
    gaintable=['<delay table>','<phase table>'],
    gainfield=['<phase ref>','<phase ref>'],
    interp=['linear','linear'],
    parang=False)
```

3.5.1 Make A New Image

Now that the phase self-calibration solutions have been applied to the phase reference calibrator, make a new image using the corrected data with tclean.

```
  tclean(vis='<your split filename>.ms',
    field='<phase reference calibrator>',
    imagename='<new base name for image files>',
    imsize=[640],
    cell=['<cell size>'],
    stokes='I',
    deconvolver='clark',
    weighting='natural',
    niter=1000,
    interactive=True,
    savemodel='modelcolumn')
```

The new image should have a different name from the image made before self-calibration. Note that we are still saving the model to the MS model column. Remember to get the image statistics with imstat and compare with the results from the first (no self-cal) image. Most of the time, you will only see minor or marginal improvement from the first image. Be suspicious if you see dramatic changes in the source structure and/or image statistics at this point.

3.6 Amplitude Self-Calibration

Once the (hopefully) improved phase self-calibrated image has been made, it is time to move on to amplitude self-calibration.

WARNING: This is the most dangerous step in self-calibration! Any structures that are in the model will now be "built in" to the image. Make sure that you fully trust your model before moving forward with amplitude self-calibration.

Refine the amplitudes with gaincal.

```
gaincal(vis='<your split filename>.ms',
  field='<phase reference calibrator>',
  caltable='<calibration table name>.apcal',
  solint='inf',
  refant='<reference antenna>',
  minblperant=4,
  gaintype='G',
  calmode='ap',
  solnorm=True,
  solmode='',
  gaintable=['<delay table>','<phase table>'],
  gainfield=['<phase ref>','<phase ref>'],
  interp=['linear','linear'],
  parang=False)
```

Amplitude self-calibration generally requires longer solution intervals than phase self-calibration. Most phase reference calibrators will need solution intervals between 120 seconds and 180 seconds, although very bright calibrators may be able to use 60 to 90 seconds. It is usually a good idea to start with using the entire scan (`solint='inf'`) before trying shorter solution intervals.

Apply the amplitude self-calibration solutions to the phase reference calibrator with applycal.

```
applycal(vis='<your split filename>.ms',
  field='<phase reference calibrator>',
  gaintable=['<delay table>','<phase table>','<amplitude table>'],
  gainfield=['<phase ref>','<phase ref>','<phase ref>'],
  interp=['linear','linear','linear'],
  parang=False)
```

3.6.1 Make Another New Image

Make sure that the self-calibration has led to an improved image by making one more image of the phase reference calibrator with tclean.

```
tclean(vis='<your split filename>.ms',
  field='<phase reference calibrator>',
  imagename='<another new base name for image files>',
  imsize=[640],
  cell=['<cell size>'],
  stokes='I',
  deconvolver='clark',
  weighting='natural',
  niter=1000,
  interactive=True,
  savemodel='modelcolumn')
```

NOTE: If you are confident that no more self-calibration will be necessary, you can avoid saving the model to the MS model column (`savemodel='none'`). This will save a little space on your hard drive and help `tclean` run a little faster.

Remember to check the image statistics with `imstat` again to make sure self-calibration is leading to real improvement.

3.7 Apply Self-Calibration Solutions to Science Target(s)

Once you are happy with the self-calibration of the phase reference calibrator, apply all of the self-calibration solutions to the science target(s) with `applycal`.

```
applycal(vis='<your split filename>.ms',
  field='<science target>',
  gaintable=['<delay table>','<phase table>','<amplitude table>'],
  gainfield=['<phase ref>','<phase ref>','<phase ref>'],
  interp=['linear','linear','linear'],
  applymode='calonly',
  parang=False)
```

When applying the solutions from the calibrator to the science target, it is best to use `applymode='calonly'`, which will avoid flagging data on the science target.

NOTE: If your observation did not require phase referencing, you can self-calibrate each science target individually.

3.8 Split Out the Science Target

Now that the science target is better calibrated, you should split it out into a MS of its own with `split`.

```
split(vis='<your split filename>.ms',
  outputvis='<your final split filename>.ms',
  field='<science target>',
  datacolumn='corrected')
```

The science target can now be imaged on its own using `tclean` (refer back to §3.4 for details on imaging). Additional flagging may be necessary on the science target before running `tclean`, so inspect it with `plotms` first. If the science target is bright enough, you can attempt to self-calibrate it using the same steps as above (except that you can set all of the interpolations to `'nearest'`, since there is only one source in the MS). If you do attempt to self-calibrate the science target, it would be a good idea to split out a new MS after each step (i.e., split after phase self-calibration, then split again after amplitude self-calibration), just in case the self-calibration steps cause problems.

4 Additional Information

4.1 Acronyms and Abbreviations

AIPS = Astronomical Image Processing Software
CASA = Common Astronomy Software Applications
DARA = Development in Africa with Radio Astronomy
EOP = Earth Orientation Parameters
ERIC = European Research Infrastructure Consortium
EVN = European VLBI Network
GUI = Graphical User Interface
JIVE = Joint Institute for VLBI ERIC (formerly, Joint Institute for VLBI in Europe)
Jy = Jansky (1 Jy = 10^{-26} W m^{-2} Hz^{-1} = 10^{-23} erg s^{-1} cm^{-2} Hz^{-1})
mas = milliarcseconds (1 mas = 1/3600000 degrees = 4.8481368×10^{-9} radians)
MS = Measurement Set
NRAO = National Radio Astronomy Observatory
RFI = Radio Frequency Interference
RMS = Root Mean Square
SNR = Signal-to-Noise Ratio
spw = spectral window(s)
TEC = Total Electron Content
VLBA = Very Long Baseline Array
VLBI = Very Long Baseline Interferometry

4.2 VLBA Station Abbreviations

BR = Brewster (Washington)
FD = Fort Davis (Texas)
HN = Hancock (New Hampshire)
KP = Kitt Peak (Arizona)
LA = Los Alamos (New Mexico)
MK = Mauna Kea (Hawaii)
NL = North Liberty (Iowa)
OV = Owens Valley (California)
PT = Pie Town (New Mexico)
SC = Saint Croix (US Virgin Islands)

4.3 Some Useful Websites

AIPS Homepage: `http://www.aips.nrao.edu/index.shtml`
CASA Download page: `https://casa.nrao.edu/casa_obtaining.shtml`
CASA Homepage: `https://casa.nrao.edu`
DARA EVN CASA tutorial: `http://www.jb.man.ac.uk/DARA/unit4/Workshops/EVN_continuum.html`
Des Small's EVN CASA tutorial: `http://www.jive.eu/~small/FringeFitting/n14c3_tutorial.html`
difmap: `https://sites.astro.caltech.edu/~tjp/citvlb/`
EVN Data Reduction Guide: `https://www.evlbi.org/evn-data-reduction-guide`
EVN Homepage: `https://www.evlbi.org/`
JIVE CASA-VLBI Workshop 2020: `https://www.jive.eu/casa-vlbi2020/programme.php`
JIVE Homepage: `http://www.jive.eu/`
jiveplot: `https://github.com/haavee/jiveplot`

NRAO Legacy Archive: https://archive.nrao.edu
NRAO New Archive: https://data.nrao.edu
NRAO Science Homepage: https://science.nrao.edu
NRAO 17th Synthesis Imaging Workshop: http://www.cvent.com/events/virtual-17th-synthesis-imaging-workshop/agenda-0d59eb6cd1474978bce811194b2ff961.aspx
VLA CASA Guides: https://casaguides.nrao.edu/index.php?title=Karl_G._Jansky_VLA_Tutorials
VLBA Homepage: https://science.nrao.edu/facilities/vlba
VLBI CASA Guides: https://casaguides.nrao.edu/index.php/VLBI_Tutorials

4.4 Some Useful Publications

Janssen et al. 2019, rPICARD: A CASA-based Calibration Pipeline for VLBI Data
Walker 2014, VLBA Scientific Memo 37: Flux Density Calibration on the VLBA
The AIPS Cookbook: html version, linked pdf version

4.5 Antenna Wildcards in `plotms`

Using the `antenna` parameter in `plotms` can help display what you want to see and reduce the time it takes to generate the plots.

- To display all baselines to all antennas, including autocorrelations, set `antenna=''`.
- To display all baselines to a specific antenna (not including autocorrelations), set `antenna='<antenna number or name>'`.
- To display all baselines to all antennas *without autocorrelations*, set `antenna='*&*'`.
- To display *only the autocorrelations* for all antennas, set `antenna='*&&&'`.

4.6 Combining Select Spectral Windows in `fringefit`

In some cases, you may wish to combine a subset of the spectral windows when doing the global fringe fitting. For example, if your observation used the dual S/X capability of the VLBA, you would not want to combine all spectral windows when doing the global fringe fit, but your phase reference calibrator may not be bright enough to fit each specrtral window individually. AIPS users may recognize this as using the aparm(5) adverb in FRING. In CASA, you will need to run `fringefit` multiple times, once per spectral window combination, but assigning the same output calibration table each time.

Presented here is an example of doing the global fringe fit on a project where the observational setup had two widely-spaced subbands using the 6cm receiver. Assume we named the measurement set "csplit.ms" and that we only have a single phase reference calibrator called "PhaseRef1". The `listobs` output for the spectral windows was:

```
Spectral Windows:  (4 unique spectral windows and 1 unique polarization setups)
  SpwID  Name   #Chans   Frame   Ch0(MHz)  ChanWid(kHz)  TotBW(kHz) CtrFreq(MHz)  Corrs
  0      none     128    GEO    4112.000       500.000     64000.0    4143.7500   RR  LL
  1      none     128    GEO    4176.000       500.000     64000.0    4207.7500   RR  LL
  2      none     128    GEO    7696.000       500.000     64000.0    7727.7500   RR  LL
  3      none     128    GEO    7760.000       500.000     64000.0    7791.7500   RR  LL
```

When running `fringefit` on these data, it might be advantageous to combine spw 0 and 1 for one set of solutions, and combine spw 2 and 3 for another set of solutions. To do this, you would use:

```
    fringefit(vis='csplit.ms',
      caltable='csplit.mbd',
      field='PhaseRef1',
      spw='0,1',
      solint='60s',
      minsnr=5,
      zerorates=False,
      refant='FD,PT,LA,KP,OV,NL,BR,HN',
      combine='spw',
      gaintable=['csplit.accor', 'csplit.gcal', 'csplit.tsys', 'csplit.sbd>'],
      interp=['nearest', 'nearest', 'nearest,nearest', 'nearest'],
      parang=True)
```

When this completes, you would see something like this in the CASA logger:
Spw 0: 16/16/16
Spw 1: 0/0/0
Spw 2: 0/0/0
Spw 3: 0/0/0

Next, you would run `fringefit` a second time using:

```
  fringefit(vis='csplit.ms',
    caltable='csplit.mbd',
    field='PhaseRef1',
    spw='2,3',
    solint='60s',
    minsnr=5,
    zerorates=False,
    refant='FD,PT,LA,KP,OV,NL,BR,HN',
    combine='spw',
    gaintable=['csplit.accor', 'csplit.gcal', 'csplit.tsys', 'csplit.sbd>'],
    interp=['nearest', 'nearest', 'nearest,nearest', 'nearest'],
    parang=True)
```

In the CASA logger, you would see something like:
Spw 0: 0/0/0
Spw 1: 0/0/0
Spw 2: 16/16/16
Spw 3: 0/0/0

To apply the solutions, we would need to be careful with the `spwmap` parameter in `applycal`.

```
  applycal(vis='csplit.ms',
    gaintable=['csplit.accor', 'csplit.gcal', 'csplit.tsys', 'csplit.sbd', 'csplit.mbd'],
    interp=['nearest', 'nearest', 'nearest,nearest', 'nearest', 'linear'],
    spwmap=[[], [], [], [], [0,0,2,2]],
    parang=True)
```

NOTE: You will need to use the `spwmap` parameter correctly for any following steps (e.g., `bandpass` and applying the `bandpass` solutions).

4.7 Creating a Model Column from an Image Using `ft`

In the event that you forget to set `savemodel='modelcoumn'` in `tclean` when doing self-calibration, you can use `ft` to create the model.

```
  ft(vis='<your split filename>.ms',
    field='<phase reference calibrator>',
    imagename='<same as the imagename from tclean>',
    usescratch=True)
```

5 Version Tracking

Here are summaries of the changes made for each version after 1.0.

Version 1.1

- Updated §1 to include the recently published VLBA CASA Guide tutorial.
- Updated §1 to mention difficulties with calibrating VLBA+Y1 observations in CASA.
- Updated §1.1.2 to reflect the retirement of the NRAO legacy archive (archive.nrao.edu) on 2022 May 02.
- In §2.1.5, added information about setting `flagdimension='time'` to avoid flagging the channels at the edges of the spectral windows when doing the automated flagging with TFCrop.
- Corrected §2.4.1 instructions for using `plotms` to inspect the data after applying the instrumental delay solutions.
- Added §4.6 giving an example of combining select spectral windows in `fringefit`.
- Made other minor changes and corrected various typos.

Acknowledgments

Thanks to Andrew Sargent and Alexander van der Horst at GWU for helping to test early versions of the CASA VLBA calibration procedure and providing useful feedback on the calibration instructions. Special thanks to Ilse van Bemmel at JIVE for providing helpful input on the drafts, which lead to a greatly improved document. Huge thanks to Mark Kettenis, Des Small, Ilse van Bemmel, and Harro Verkouter at JIVE, and George Moellenbrock and Neal Schweighart at NRAO for all of their work developing VLBI tools in CASA. The first EVN CASA tutorial was created by Des Small. The EVN CASA tutorials for the DARA Workshop were based on Des Small's work, with updates from Anita Richards (Univ. of Manchester) and Jack Radcliffe (SARAO/Univ. of Pretoria). The first VLBA CASA Guide tutorial was created by Justin Linford, and was largely based on the EVN tutorials. The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.