Memo: Phasing in pyuvdata

Bryna Hazelton, Miguel Morales and the pyuvdata team

June 28, 2018

1 Introduction

This memo discusses the phase types that are supported in pyuvdata, the implementation of phasing in pyuvdata (based on the authors' conceptual understanding of phasing in radio interferometry) and the testing that has been done on the pyuvdata phasing code.

2 Types of phasing supported in pyuvdata

pyuvdata supports two primary kinds of phasing, zenith drift phasing (drift) and data that are phased to one particular RA and Dec defined at a particular epoch (phased) with very limited support for labelling and handling other or unknown phasing situations (unknown). Which kind of phasing any given UVData object has is stored in the phase_type attribute (allowed values: drift, phased, unknown).

Zenith drift phasing is the phasing type that natively comes off of many low-frequency radio telescope correlators (e.g. MWA, PAPER), meaning that the correlations for each time are calculated phased to zenith (i.e. with no baseline-dependent phase applied). The baseline location vectors, the uvw_array attribute, are then just given by the antenna separations in a local East-North-Up (ENU, also called topocentric) coordinate system for all times.

Phased data sets have had rotations applied to the u, v, w coordinates (the ENU baseline location vectors) and baseline-dependent phasing applied to the visibilities to make the visibilities add coherently toward a particular RA and Dec defined at a particular epoch (typically J2000). Phased data sets have thus been corrected to account for the motion of the earth (rotation, precession, nutation and aberration due to the earth's motion around the sun). This is generally required for imaging and to save the data in some file formats (e.g. uvfits).

Moving between these phase types requires functionality to phase zenith drift data to any given RA and Dec (defined at an epoch) and the ability to undo the phasing to get back to zenith drift. Changing between different phase centers can be achieved by first unphasing to drift and then phasing to the new phase center.

3 Conceptual description of phasing

Phasing seems like it should be straightforward, but is actually quite subtle. The phasing implemented in `pyuvdata` should handle most imaging and widefield radio telescopes to good accuracy, but is not accurate enough for VLBI. For a full description of astrometry and phasing the reader is encouraged to read the USNO Circular 179 [1], the `astropy` documentation, and the paper by Kaplan [2] relating optical and radio astrometric traditions and nomenclature. Here we give a very abbreviated conceptual description of phasing as implemented in `pyuvdata`.

We find it useful to think about a pair of antennas on the ground with a fixed physical separation. The baseline between the antennas with respect to the fixed stars (well, really AGN) of course depends on LST as the Earth spins, but over longer periods of time the orientation also changes with the precession and nutation of the Earth's spin axis. So at the same LST separated by a sidereal year the orientation of the antennas will change due to the motion of the Earth's pole. A widefield camera will see both the field center (phase center) and field orientation (rotation) change due to this effect.

However, there is a additional relativistic effect that depends on the velocity of the antennas at the time of observation. An observer at the barycenter of the solar system will see the distance between the antennas relativistically shorten and rotate (Terrell rotation) due to the velocity of the Earth around the solar system (and to a lesser degree the movement of the antennas due to Earth rotation). This effect is known as 'aberration' and is a 20" scale effect over the course of a year. From the radio perspective, if we define the location of distant AGN relative to the solar system barycenter (so they don't move with time during a year), then the baseline is shortened and rotated in this barycentric frame relative to the locations on the ground due to the special relativistic transform between the frames.[1]

So "J2000 coordinates" is more properly known as the ICRS frame: a set of coordinates defined by the locations of distant quasars in the barycenter inertial frame. ICRS coordinates do not change with time. To phase properly we need to translate the baselines into the ICRS frame. We start with the uvw or antenna positions on the ground relative to the array center (East-North-Up). This is then translated into coordinates fixed to the earth and centered at the earth's center (ITRS, also referred to as ITRF or Earth Centered Earth Fixed ECEF). Conceptually we can then unwind the Earth's rotation (approximately LST) and the motion of the pole to put this into a coordinate system that is centered on the Earth but does not rotate and is aligned with J2000—the GCRS frame. But the GCRS frame is still orbiting with the earth, so has a large velocity. To correct for relativistic aberration we must apply the relativistic length contractions to transform to the barycentric ICRS frame. In practice `astropy` goes directly from ITRS coordinates locked to the

[1] The barycenter is actually not an inertial frame due to motion of the solar system around the galaxy, and this is not a small effect (e.g. 200"). However, this is hard to measure and changes slowly, so the barycenter is assumed to be inertial. But travelers over StarTrek distances should beware.

Earth to ICRS barycentric J2000 coordinates, without going through GCRS.

At this point we finally have what the baseline distances and orientations appear to be for light from distant astronomical sources. We can then pick a phase center (direction of zero delay), and phase the data correctly in that J2000 direction.

It should be noted that this calculation ignores a couple of important effects. First, we assumed the same observation time for both antennas. While the antennas don't move a large distance between when the wavefront hits one antenna and when it hits the other, this is an important effect for VLBI. This leads to a small baseline correction that depends on the direction of the source, and as such does not fit will into the mental model of `pyuvdata`. Or said another way, the `pyuvdata` corrections should correctly apply relativistic aberration at the field center and the first order-stretch across the field, but higher order distortions are not included. Similarly doppler shifts and relative doppler shifts are not accounted for in `pyuvdata` phasing.

4 How phasing is done in `pyuvdata`

`pyuvdata`'s phasing uses `astropy` to do all the coordinate conversions between earth referenced and solar-system barycenter referenced coordinate systems (called 'frames' in `astropy`). Of particular interest are the ITRS frame, which is fixed to the earth and rotates with the earth, useful for describing antenna and telescope locations, and the ICRS frame which is a non-rotating frame referenced to the solar system barycenter with axes aligned with J2000 axes, useful for defining fixed celestial sources like radio galaxies. One other frame that can be useful is the GCRS frame which is non-rotating with axes aligned with ICRS but geo-center referenced so that it moves with the earth around the solar system barycenter.

Phasing and unphasing in `pyuvdata` works from the existing `uvw_array` by default, but can optionally work from the antenna positions (by setting `use_ant_pos=True`). By default, phasing and unphasing use the ICRS frame for the phased uvws, but the code can optionally phase to and from GCRS (by setting `phase_frame=GCRS`). This is not as accurate, but is included to support testing against other codes (e.g. `MWA_tools`) and to support unphasing data sets that were originally phased to GCRS (e.g. MWA uvfits files).

To phase zenith drift data, the code performs the following steps:

- Define the phase center in an `astropy` ICRS frame. If the epoch is 'J2000' or '2000' use the RA and Dec to define a SkyCoord in the ICRS frame. Otherwise define a SkyCoord in the 'FK5' frame at the given RA, Dec and epoch and convert it to an ICRS frame for phasing. This means that all phasing is done in the ICRS (J2000) frame, but if the RA and Dec are specified at another epoch, the RA and Dec are converted to ICRS before phasing. Optionally transform the phase center to the GCRS frame if phasing to GCRS.

- For each time in the `time_array` attribute: Define the telescope location as a SkyCoord object in the ITRS frame with `obstime` equal to the time.
- For each time in the `time_array` attribute: Define the antenna positions or uvws as a SkyCoord object in the ITRS frame with `obstime` equal to the time. For antenna positions, this requires combining the `telescope_location` attribute (which holds the telescope location in the ITRS frame) and `antenna_positions` attribute (which holds the antenna positions relative to the telescope location in the ITRS frame). For uvws (which are intrinsically in an ENU frame) this requires converting from ENU to ITRS using `pyuvdata.utils.ECEF_from_ENU` using the `telescope_location` attribute as well as the `uvw_array` attribute.
- For each time in the `time_array` attribute: Transform the telescope location and the antenna positions or uvws to the ICRS frame (or GCRS if phasing to GCRS).
- Calculate the relative antenna positions or uvws in the ICRS (or GCRS) frame by subtracting the telescope location in the ICRS (or GCRS) frame.
- Calculate the phased antenna positions or uvws using the `pyuvdata.utils.phase_uvw` function[2]. If using antenna positions, difference these phased positions to calculate the uvws for each baseline.

To unphase to drift, the steps are essentially followed in reverse order and unit testing guarantees that phasing followed by unphasing gets back to nearly the same answer. A cycle of phasing and unphasing does incur some error, differences between the u, v, w coordinates before and after such a cycle are about $3 - 7 \times 10^{-5}$ and these errors scale linearly with the number of cycles. This error comes from `astropy`, which can be seen by a cycle of converting antenna positions from ITRS to ICRS and back to ITRS, which introduces errors on the same scale. The investigations of these errors are documented in two Jupyter notebooks[3].

5 Testing `pyuvdata`'s phasing

The primary external testing reference for `pyuvdata`'s phasing code is the MWA phasing code found in the `MWA_tools` repository under `CONV2UVFITS` (also in the MWA Cotter

[2] One way to think about what this function does is that is mathematically identical to calculating ENU values, except that it is in the ICRS (or GCRS) frame and the center of the ENU coordinate system is given by the phase center. This makes sense in that if you choose a phase center which is aligned with zenith at the observation time, the changes to the uvws are small and are just due to the changes in the earth spin axis (due to precession and nutation) and the length contraction due to earth's motion around the sun (aberration, excluded if phasing to GCRS).

[3] https://github.com/bhazelton/random_stuff/blob/master/jupyter_notebooks/phase_unphase_match_mwa.ipynb and https://github.com/bhazelton/random_stuff/blob/master/jupyter_notebooks/phase_unphase_match_5km_sim.ipynb

code[4]). The MWA code uses the SLA C Library which is proprietary. The authors of this memo and of the `pyuvdata` code did not attempt to copy or replicate parts of the SLA library, they simply compared calculations done using `astropy` to calculations performed in the MWA code (which calls the SLA library routines in various ways).

The calculation comparisons made in this process are documented in a Jupyter notebook[5] and the ones most relevant to the phasing code have been extracted into unit tests (in `test_utils.py`).

5.1 Limitations of the MWA reference code

By comparing the values calculated by the MWA code at various stages to values calculated using various `astropy` conversions, the authors of this memo determined that the MWA code is only correcting the u, v, w coordinates for precession and nutation and not for aberration. This is supported by a few lines of evidence, one is that the uvw correction is done using a rotation matrix, which cannot account for length contraction and the other is that the corrected values closely match the values generated using `astropy` conversions to the GCRS frame, but not to conversions to the ICRS frame (which differ by the inclusion of the effects of earth's motion around the sun, also called aberration).

In addition, there are some discrepancies in the LST calculations done in the MWA code and in `astropy`. The MWA code uses the mean LST to calculate the RA of zenith in the True Equator and Equinox frame (TEE, also referred to as the current epoch). The mean LST calculated by `astropy` differs from the mean LST calculated in the MWA code by about 4 arcseconds at the particular time used for testing (mjd = 55780.1). Furthermore, the authors believe that the LST used in this calculation should actually be the apparent LST (which corrects for nutation in addition to precession). The apparent LST calculated by `astropy` differs from the mean LST calculated in the MWA code by about 12 arceseconds at the same testing time. Note that the LST is not used in `pyuvdata`'s phasing code, instead all the precession and nutation corrections are applied in the `astropy` frame conversions.

5.2 External phasing test

We received a couple uvfits files from David Kaplan containing identical MWA data phased to two different phase centers. The authors presume that the phasing was done with the MWA phasing code (likely as implemented in Cotter). To test our phasing code, we compared the u, v, w coordinates from each file after unphasing both files and after unphasing and rephasing one of the files to the phase center of the other file (all using `phase_frame=GCRS`). The results show that the uvws in each comparison (drift and rephased) agree to better than 2 cm. We also did the unphase and rephase test with cal-

[4] https://github.com/MWATelescope/cotter

[5] https://github.com/bhazelton/random_stuff/blob/master/jupyter_notebooks/phasing_compare_mwatools.ipynb

culating the uvws based on the antenna positions (rather than working with the uvws in the file) and in that test the uvws agree to better than 5 mm. These tests are documented in a Jupyter notebook[6].

We extracted a single time and 10 frequencies from these files and used the cut-down files in unit tests (in `test_uvdata.py`).

6 Future directions

The VLBI `calc` program is probably the most accurate code available for comparing phasing against. It is possible that it includes some effects that we don't wish to include, however. The information we have about `calc` comes from the following email from Randall Wayth on other ways to test phasing calculations:

> Way back when we were trying to sort out the precession corrections for MWA, I was comparing the output from Al's new code to that of the VLBI program called 'calc'. 'calc' is the tool that most (all?) VLBI observatories use to calculate delays and u, v, w coords for data, and as far as I know is a fully complete system with all known effects. For low frequencies it is overkill of course, but as a reference system it is probably ideal.
>
> The way calc works is slightly clunky, but it is fairly easy to use. You set it up as a server, then programs make requests to the server by passing required info and calc returns all the info and derivatives so that you can interpolate between two time intervals. I probably have some notes and code somewhere for how to set it up and use it if you are interested.
>
> On a related note, when MWA and ASKAP were being set up and commissioned around 2012/2013 I asked CSIRO if they were planning to set up a calc server for ASKAP and they responded that they have (somehow) pulled the core functionality out of calc and set it up as a C library. I remember they sent me some code, but I never got it to work and they didn't respond when I asked for help. But there are probably other examples of good libraries floating around if you're willing to ask. Presumably the VLA has a core library that does all this stuff also.

The test code that Randall developed with `calc` is available in `MWA_tools` under `CONV2UVFITS/mwacalc.c`

[6] https://github.com/bhazelton/random_stuff/blob/master/jupyter_notebooks/external_phasing_test.ipynb

References

[1] George Kaplan, *USNO Circular 179*, http://aa.usno.navy.mil/publications/docs/Circular_179.php, 2005

[2] George H. Kaplan, High-Precision Algorithms for Astrometry: A Comparison of Two Approaches, *The Astronomical Journal*, 115, 1, 1998