The CASA calibration model

(and how it differs from AIPS)

Mark Kettenis, JIVE

JUMPING JIVE
Joint Institute for VLBI ERIC

RadioNet

JIVE

Measurement Equation
(RIME)

- Formulated by: Hamaker, Bregman & Sault, 1996, A&AS, **117**, 137
- Reformulated in: Smirnov, 2011, A&AS, **527**, A106
- Mathematical basis for calibration of a radio interferometer
- Fully incorporates polarization

Electric field at the source: $\mathbf{e} = \begin{pmatrix} e_r \\ e_l \end{pmatrix}$

Recorded voltages of feeds at telescope: $\mathbf{v} = \mathbf{J}\mathbf{e}$ with (2x2) Jones matrix $\mathbf{J}$

Visibility matrix produced by the correlator: $\mathrm{V}_{pq} = 2\langle \mathbf{v}_p \mathbf{v}_q^H \rangle$

Measurement equation: $\mathrm{V}_{pq} = 2\langle \mathbf{J}_p (\mathbf{e}_p \mathbf{e}_q^H) \mathbf{J}_q^H \rangle = \mathbf{J}_p \mathbf{B} \mathrm{J}_q^H$ with brightness matrix B $= \begin{pmatrix} I+Q & U+iV \\ U-iV & I-Q \end{pmatrix}$

Goal is to determine $\mathbf{J}_p$ for all antennas p.

Measurement Equation
continued

$\mathbf{J}_p = \mathbf{B}_p \mathbf{G}_p \mathbf{D}_p \mathbf{E}_p \mathbf{P}_p \mathbf{K}_p \mathbf{T}_p$

- $\mathbf{T}_p$ Polarization-independent multiplicative effects introduced by the troposphere, such as opacity and path-length variation.
- $\mathbf{K}_p$ Delay (this is VLBI!)
- $\mathbf{P}_p$ Parallactic angle, which describes the orientation of the polarization coordinates on the plane of the sky. This term varies according to the type of the antenna mount.
- $\mathbf{E}_p$ Effects introduced by properties of the optical components of the telescopes, such as the collecting area's dependence on elevation.
- $\mathbf{D}_p$ Instrumental polarization response. "D-terms" describe the polarization leakage between feeds.
- $\mathbf{G}_p$ Electronic gain response due to components in the signal path between the feed and the correlator.
- $\mathbf{B}_p$ Bandpass (frequency-dependent) response, such as that introduced by spectral filters in the electronic transmission system.

CASA always applies these in the same (physically correct) order!

CASA calibration

SN table

- CASA calibration tables represent Jones matrices
  - Have an identity
  - Contain real or complex parameters that are used to calculate elements

    Complex gain: $\mathbf{G} = \begin{pmatrix} g_r & 0 \\ 0 & g_l \end{pmatrix}$ is described by tow complex paramaters.
  - Can be given arbitrary (meaningful) names
- **Always explicitly specify calibration tables to be applied!**
  - There is no equivalent of an AIPS CL table

CASA calibration
continued

- Calibration tables are specified with task parameters:
    - `gaintable = [caltable1, caltable2]`
    - `gainfield = [field1, field2]` e.g. `'3C84', 'J1023+43'`
      (*field1* applies to *caltable1*, *field2* to *caltable2*)
    - `interp = [interp1, interp2]` e.g. `'linear', 'nearest'`
      (*interp1* applies to *caltable1*, *interp2* to *caltable2*)
    - `parangle = True` or `False` (default)
- Data without calibration solutions is automatically flagged!
    - Can be bypassed when applying the final calibration
- Data is aggressively flagged if it is partly flagged:
    - `corrdepflags = True` or `False` (default); `True` prevents flagging both pols if one is flagged

New in 5.7/6.1

Data Formats

- MeasurementSet (v2)
  Native data format of CASA
- UV-FITS
  What AIPS writes
- FITS-IDI
  Produced by the SFXC (EVN) and DiFX (VLBA, LBA, …) correlators

All thee formats can contain metadata such as gain curves and T_sys

Preparing your data

ANTAB

scripts at https://github.com/jive-vlbi/casa-vlbi

- Extract gain curves
  - Use gc.py script to import gaincurves from ANTAB files (EVN & Co)

    `casa -c gc.py` *`antabfile gcfile`*

  - Use gc2.py script to import gain curves from FITS-IDI files (VLBA)

    `casa -c gc2.py` *`fitsfile gcfile`*

- Attach T_sys measurements (EVN & Co)

  `casa -c append_tsys.py` *`antabfile fitsfiles…`*

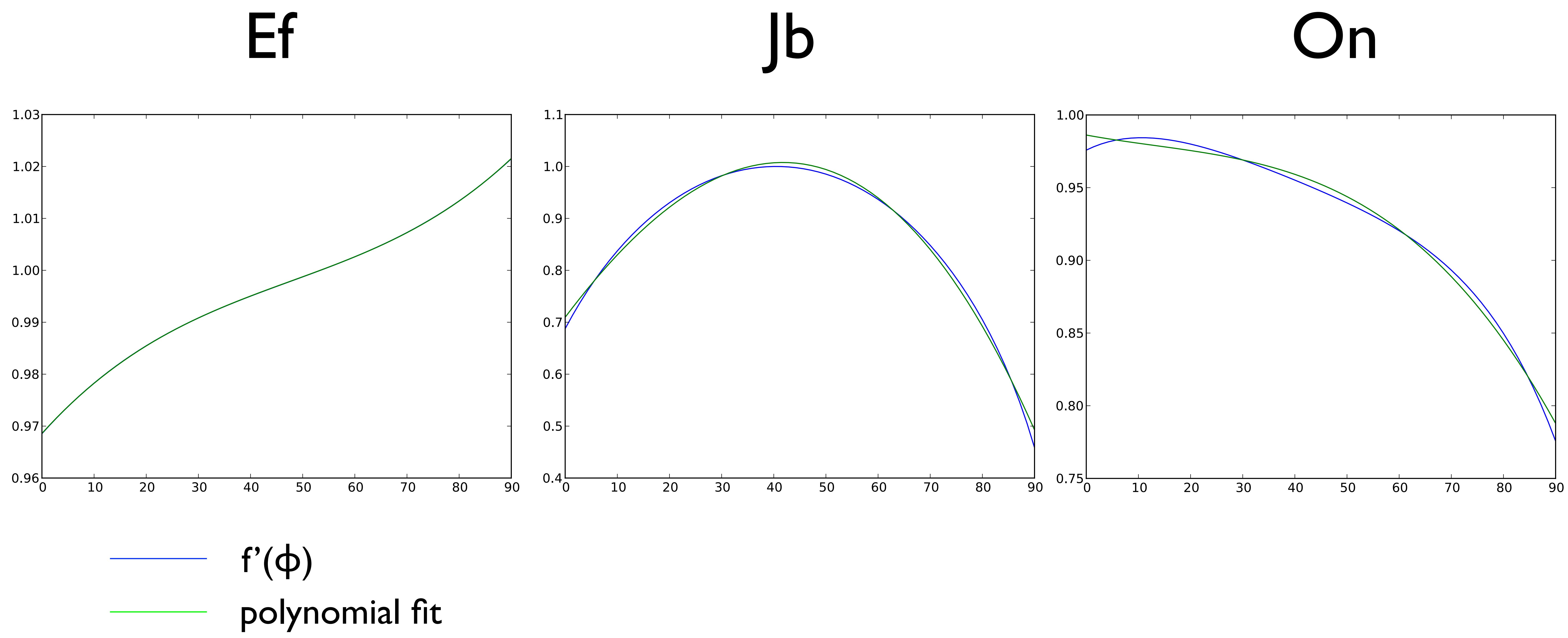
Gain curves

- Different ways to express gain curves
  - voltage vs. power
  - parametrization (function of zenith angle vs. elevation)
- CASA 5.7/6.1 only supports voltage as function of zenith angle
  - `gc.py` and `gc2.py` scripts convert by sample and refit
  - gain curves are not always well-behaved
    use `—min-elevation` and `—max-elevation options`

Gain curves

Fitted gain curves

Third order polynomial fit of $f'(\phi) = \sqrt{f(90-\phi)}$



Preparing your data

The future

Expected in CASA 5.8/6.2

- Attach gain curves (EVN & Co)

  `append_gc.py` *`antabfile fitsfiles..`*

- Attach T_sys measurements (EVN & Co)

  `append_tsys.py` *`antabfile fitsfiles…`*

No preparation needed for VLBA data!

Importing your data

FITLD

- FITS-IDI data can be imported using the importfitsidi
  - A single FITS-IDI file:

```
importfitsidi(vis=ms, fitsidifiles=[fitsfile],
              scanreindexgap_s=seconds)
```

15 seconds is good (matches FITLD)

  - Multiple FITS-IDI files for a single observation:

```
importfitsidi(vis=ms, fitsidifiles=[fitsfile1, fitsfile2],
              constobsid=True, scanreindexgap_s=seconds)
```

Use Python glob module for EVN data

  - Applies digital corrections for DiFX correlator (VLBA & Co)

```
import glob
fitsfiles =  sorted(glob.glob(”N20C2_1_1.IDI*”)
```

- UVFITS data can be imported using importuvfits

```
importuvfits(vis=ms, fitsfile=[fitsfile])
```

This does not import most of the VLBI metadata correctly!

Normalizing your data

ACCOR

- Fix correlation amplitudes based on autocorrelations (VLBA & Co)

```
accor(vis=ms, caltable=caltable)
```

- Generates G-type calibration table

- CASA data selection provides AIPS `ACSCL` functionality

Flagging your data

UVFLG

- Apply a-priori flagging (EVN & Co)

```
$ flag.py uvflgfile fitsfile > flagfile
```

```
flagcmd(vis=ms, inpmode='list',inpfile=flagfile)
```

- Additional flagging is don using `plotms`

Amplitude calibration

ANTAB

- Generate caltables for gain curves:

  ```
  gencal(vis=ms, type=’gc’, infile=gcfile, caltable=gctable)
  ```

- Generate caltables for T_sys:

  ```
  gencal(vis=ms, type=’tsys’, caltable=tsystable)
  ```

- Generates G-type calibration tables
- To apply use:

  ```
  gaintable=[gctable, tsystable]
  ```

  In subsequent calibration tables.

Bandpass calibration

BPASS

- Generate caltables for gain curves:

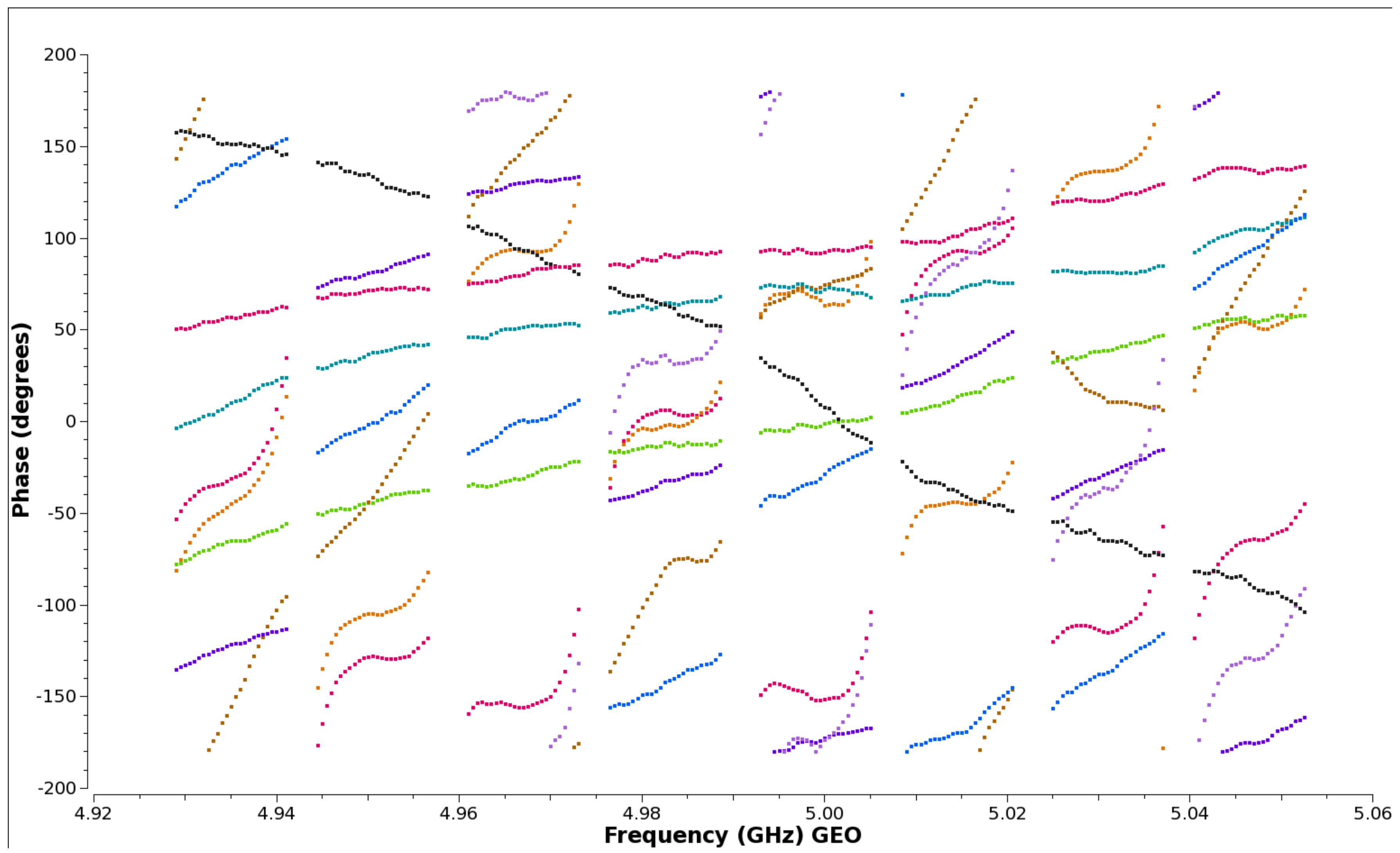
```
bandpass(vis=ms, field=field, refatnd=refant,
    gaintable=[…], solnorm=True, caltable=bptable)
```

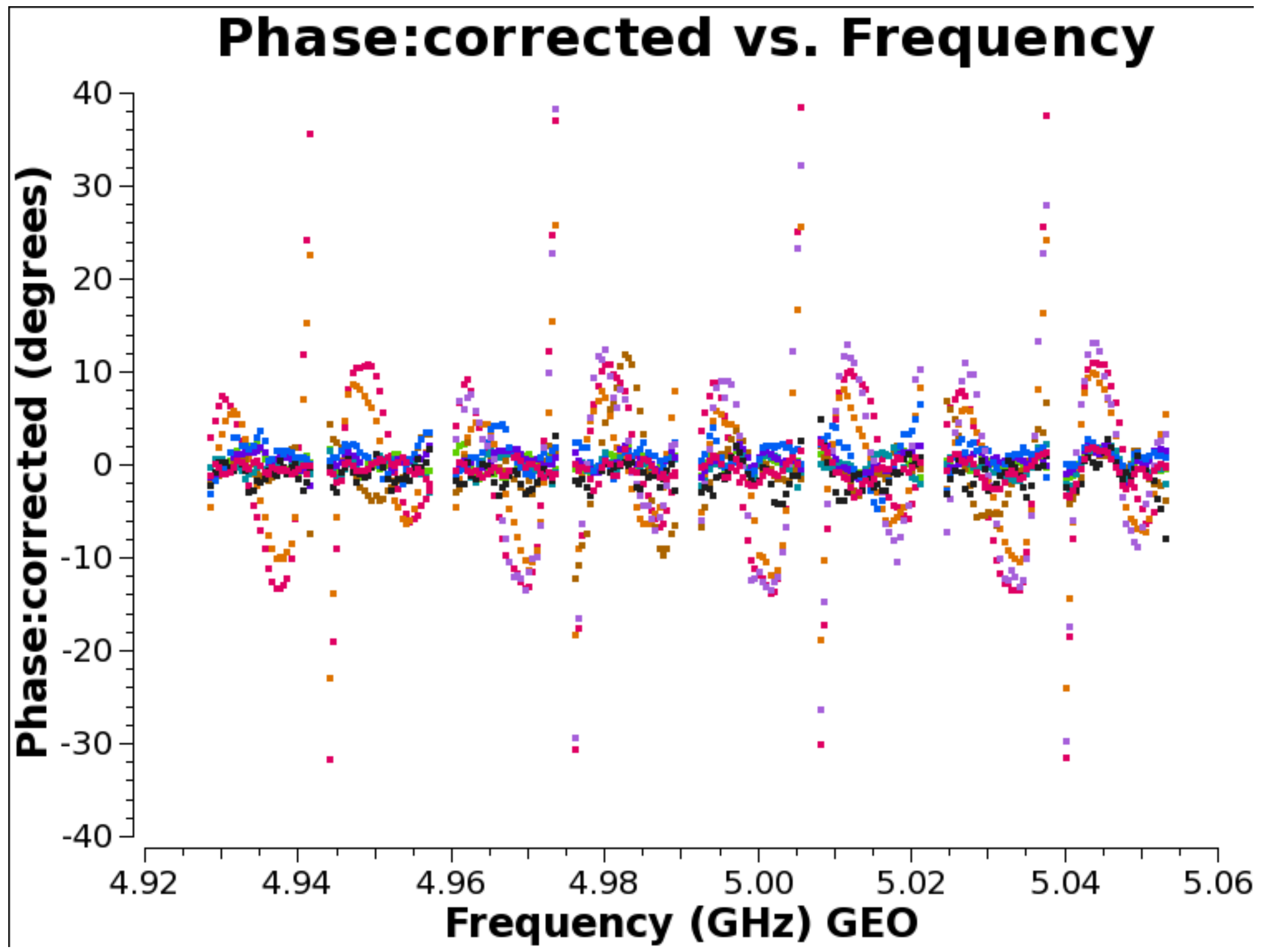
- Generates B-type calibration tables

Fringe Fitting

FRING

- See lecture by Des Small on tuesday





Apply calibration

SPLIT

- Applying calibration to the whole MeasurementSet:

  ```
  applycal(vis=ms, gaintable=[…], interp=[…], …)
  ```

  - Adds a CORRECTED_DATA colum; full copy of the data
- Split the MeasurementSet:

  ```
  split(vis=ms, outputvis=splitms, field=field, …)
  ```

  - Supports averaging (time & frequency)
  - Needs to be run for each field you want to image
  - The `mstransform` task can also be used.
    - Ends up running the same code.

CASA5 vs CASA6

Python 2 or Python 3

- CASA 5.x uses Python 2.7
  - Python 2 is no longer supported
  - Python scripts need to be invoked using `casa -c`
  - Still includes `plotcal`
  - Will go away in the future
- CASA 6.x uses Python 3.6
  - The world is moving to Python 3
  - No longer includes `plotcal`
  - Proper Python modules, can be easily included in Python

Thanks to our sponsors

JUMPING JIVE
Joint Institute for VLBI
ERIC

RadioNet

This event has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements 730562 (RadioNet) and 7308844 (JUMPING JIVE)

JIVE