EUROPEAN ARC
ALMA Regional Centre || Allegro

ALMA CASA Calibration

Allegro - CASA Tutorial Day

Luke T. Maud

3 March 2017

Calibration - the basics

- Remove effects of the instrument itself
- Remove effects of the atmosphere
- Scaling to the correct flux

Please ask questions throughout if anything is unclear

Calibration - the basics

- Remove effects of the instrument itself
  - variations in frequency as a function of amplitude and phase
  - variation in receiver noise (Tsys)
  - effects of shadowing
  - antenna positions
- Remove effects of the atmosphere
  - atmospheric variability (phase)
  - atmospheric attenuation as function of time (Tsys)
- Scaling to the correct flux

-using an astronomical source

Calibration - the basics

- **Remove effects of the instrument itself**
  - variations in frequency as a function of amplitude and phase — **- Bandpass**
  - variation in receiver noise (Tsys) — **- Tsys load**
  - effects of shadowing — **- Flagging**
  - antenna positions — **- All sky runs**
- **Remove effects of the atmosphere**
  - atmospheric variability (phase) — **- WVRs + Gain Calibrator**
  - atmospheric attenuation as function of time (Tsys) — **- Tsys load**
- **Scaling to the correct flux**
  - using an astronomical source — **- Flux Calibrator**

Calibration - the basics

Bandpass

QSO -freq. response

Flux

planet/QSO

Temporal Gains

WVR system

QSO - phase/amp

CHECK THE CALIBRATIONS

ON THE CALIBRATORS !!

Apply calibrations

to the science target

Calibration - with CASA

All run from the scriptForPI.py

- manual calibration
  - uid__XXXXX_scriptForCalibration.py

- pipeline calibration
- casa_piperestorescript.py
- casa_pipescript.py

Calibration - with CASA

All run from the scriptForPI.py

- manual calibration
  - uid__XXXXX_scriptForCalibration.py
    - + scriptForImagingPrep.py
    - + scriptForImaging.py

- pipeline calibration
- casa_piperestorescript.py
- casa_pipescript.py
  - + scriptForImagingPrep.py
  - + scriptForImaging.py

Calibration - with CASA

All run from the scriptForPI.py

- manual calibration
  - uid__XXXXX_scriptForCalibration.py
    - + scriptForImagingPrep.py
    - + scriptForImaging.py

- pipeline calibration
- casa_piperestorescript.py
- casa_pipescript.py
  - + scriptForImagingPrep.py
  - + scriptForImaging.py

Calibration - with CASA

MANUAL

PIPELINE

Please navigate to your:

‘uid_XXXXX_scriptForCalibration.py’

OR

pipeline-weblog -> index.html

> cd /lustre/allegro/home/guestX/

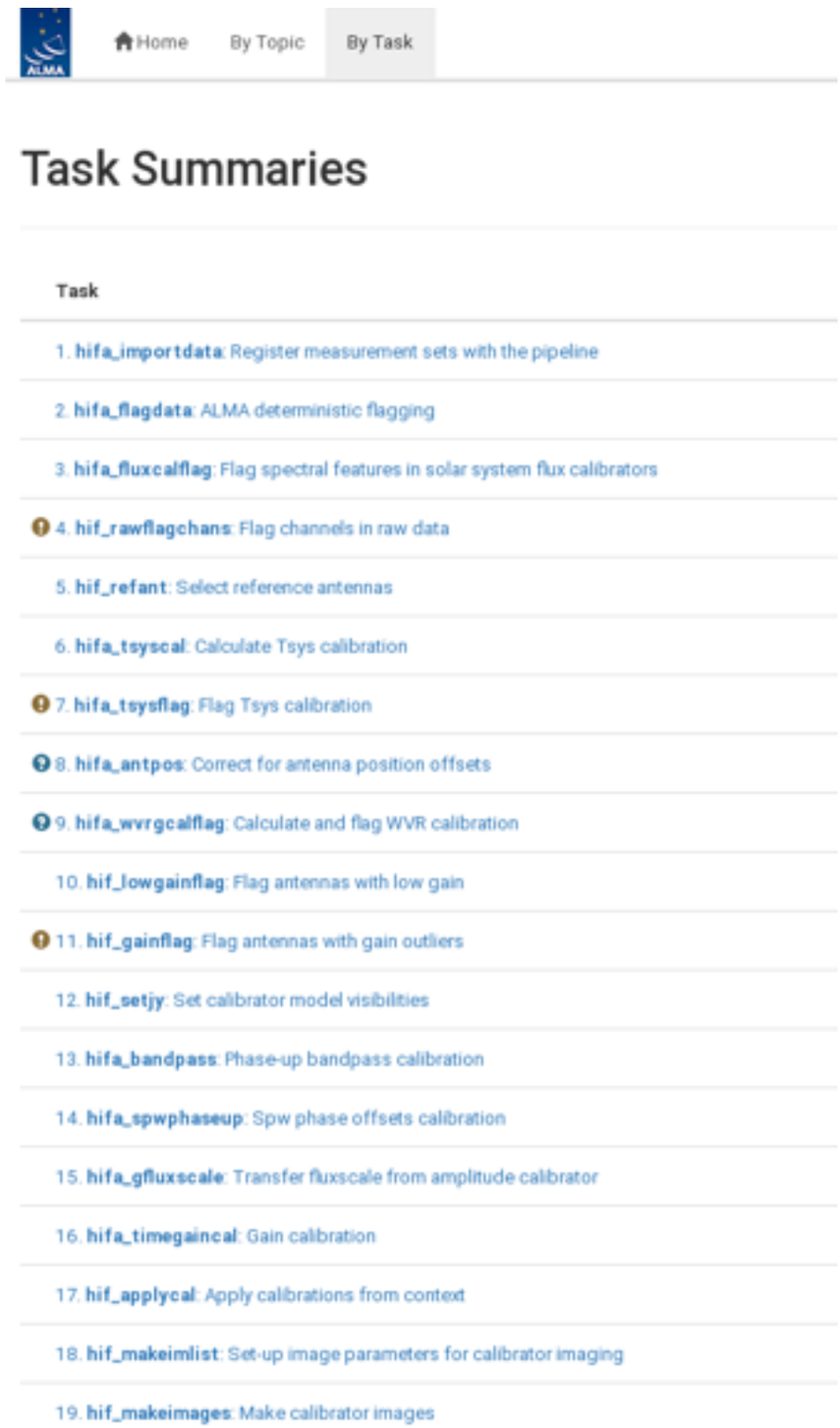
open_CASA_training_day/analysis/guestX

Calibration - with CASA

MANUAL

```
# ALMA Data Reduction Script

# Calibration

thesteps = []
step_title = {0: 'Import of the ASDM',
              1: 'Fix of SYSCAL table times',
              2: 'listobs',
              3: 'A priori flagging',
              4: 'Generation and time averaging of the WVR cal table',
              5: 'Generation of the Tsys cal table',
              6: 'Generation of the antenna position cal table',
              7: 'Application of the WVR, Tsys and antpos cal tables',
              8: 'Split out science SPWs and time average',
              9: 'Listobs, and save original flags',
              10: 'Initial flagging',
              11: 'Putting a model for the flux calibrator(s)',
              12: 'Save flags before bandpass cal',
              13: 'Bandpass calibration',
              14: 'Save flags before gain cal',
              15: 'Gain calibration',
              16: 'Save flags before applycal',
              17: 'Application of the bandpass and gain cal tables',
              18: 'Split out corrected column',
              19: 'Save flags after applycal'}

if 'applyonly' not in globals(): applyonly = False
try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
if (thesteps==[]):
  thesteps = range(0,len(step_title))
  print 'Executing all steps: ', thesteps
```

PIPELINE



Calibration - with CASA

MANUAL

```
# ALMA Data Reduction Script

# Calibration

thesteps = []
step_title = {0: 'Import of the ASDM',
              1: 'Fix of SYSCAL table times',
              2: 'listobs',
              3: 'A priori flagging',
              4: 'Generation and time averaging of the WVR cal table',
              5: 'Generation of the Tsys cal table',
              6: 'Generation of the antenna position cal table',
              7: 'Application of the WVR, Tsys and antpos cal tables',
              8: 'Split out science SPWs and time average',
              9: 'Listobs, and save original flags',
              10: 'Initial flagging',
              11: 'Putting a model for the flux calibrator(s)',
              12: 'Save flags before bandpass cal',
              13: 'Bandpass calibration',
              14: 'Save flags before gain cal',
              15: 'Gain calibration',
              16: 'Save flags before applycal',
              17: 'Application of the bandpass and gain cal tables',
              18: 'Split out corrected column',
              19: 'Save flags after applycal'}

if 'applyonly' not in globals(): applyonly = False
try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
if (thesteps==[]):
  thesteps = range(0,len(step_title))
  print 'Executing all steps: ', thesteps
```

PIPELINE

Home | By Topic | By Task

Task Summaries

Task
1. **hifa_importdata**: Register measurement sets with the pipeline
2. **hifa_flagdata**: ALMA deterministic flagging
3. **hifa_fluxcalflag**: Flag spectral features in solar system flux calibrators
4. **hif_rawflagchans**: Flag channels in raw data
5. **hif_refant**: Select reference antennas
6. **hifa_tsyscal**: Calculate Tsys calibration
7. **hifa_tsysflag**: Flag Tsys calibration
8. **hifa_antpos**: Correct for antenna position offsets
9. **hifa_wvrgcalflag**: Calculate and flag WVR calibration
10. **hif_lowgainflag**: Flag antennas with low gain
11. **hif_gainflag**: Flag antennas with gain outliers
12. **hif_setjy**: Set calibrator model visibilities
13. **hifa_bandpass**: Phase-up bandpass calibration
14. **hifa_spwphaseup**: Spw phase offsets calibration
15. **hifa_gfluxscale**: Transfer fluxscale from amplitude calibrator
16. **hifa_timegaincal**: Gain calibration
17. **hif_applycal**: Apply calibrations from context
18. **hif_makeimlist**: Set-up image parameters for calibrator imaging
19. **hif_makeimages**: Make calibrator images

Calibration - Tsys, AntPos, WVR

MANUAL

```
os.system('rm -rf uid___A002_Xa44acb_Xadb.ms.tsys')
gencal(vis = 'uid___A002_Xa44acb_Xadb.ms',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.tsys',
  caltype = 'tsys')

# Flagging edge channels

flagdata(vis = 'uid___A002_Xa44acb_Xadb.ms.tsys',
  mode = 'manual',
spw = '9:0~3;124~127,11:0~3;124~127,13:0~3;124~127,15:0~3;124~127',
flagbackup = F)
```

- **gencal - this generates the Tsys correction table**
- **flagdata - used to flag the bad response TDM edge channels**

PIPELINE

6. hifa_tsyscal
7. hifa_tsysflag
8. hifa_antpos
9. hifa_wvrgcalflag

Weblog

WHY? : Correct for sky and receiver noise/variation
Plots : in 'QA' (or 'calibration') directory / weblog
'*tsys*plots'

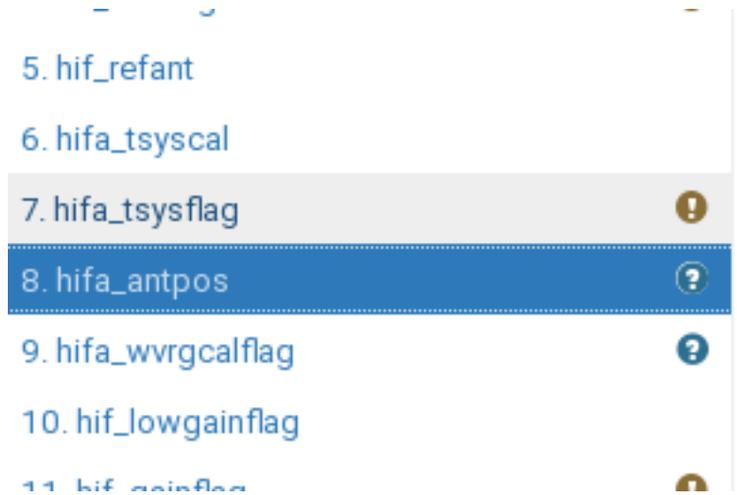
Calibration - Tsys, AntPos, WVR

MANUAL

```
  os.system('rm -rf uid___A002_Xa44acb_Xadb.ms.antpos')
  gencal(vis = 'uid___A002_Xa44acb_Xadb.ms',
    caltable = 'uid___A002_Xa44acb_Xadb.ms.antpos',
    caltype = 'antpos',
    antenna = 'DA45,DA51,DA53,DA55,DA58,DA60,DA63,DV01,DV03,DV04,DV08,DV
i18,DV19,DV24,PM01,PM02,PM03,PM04',
    #parameter = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
i0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    parameter = [-2.38629e-04,6.51773e-04,2.24732e-04,2.86116e-05,-5.424
i-2.70048e-05,-1.74361e-04,-2.25566e-04,-7.66657e-05,-7.26511e-04,-2.7060
i.25562e-05,-4.89141e-04,-1.33922e-04,-3.32959e-04,-2.69390e-04,7.35842e-
i8316e-04,-1.34949e-05,-5.56033e-04,-1.81481e-04,4.73415e-05,-4.94519e-04
i29e-05,-1.77334e-04,-2.49834e-04,5.08055e-05,2.10684e-05,-2.19155e-04,-9
i04,-2.88958e-04,4.48046e-04,-1.07709e-03,-6.35845e-04,-4.46767e-04,-1.07
i,1.40105e-04,6.45677e-04,-4.75390e-04,-1.65702e-04,4.36814e-04,-1.03487e
i762e-04,-2.25950e-04,4.71491e-05,-4.81501e-04,-1.01380e-04,5.41472e-04,8
i04,4.63888e-04,-1.72769e-03,-1.29760e-03,5.52132e-04,-1.73621e-03,-1.023
i1.65628e-03,-1.30656e-03])
```

- **gencal - this generates the antpos correction table - comment out/in the parameters**

PIPELINE



Weblog

WHY? : Correct for antenna pad location

Plots : none - tabular list of offsets

Calibration - Tsys, AntPos, WVR

MANUAL

```
wvrgcal(vis = 'uid___A002_Xa44acb_Xadb.ms',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.wvr',
  spw = [17, 19, 21, 23],
  smooth = '6.048s',
  toffset = 0,
  tie = ['W33A,J1733-1304'],
  statsource = 'W33A')
```

- **wvrgcal - reads the water vapour radiometer signals per antenna and creates the antenna based phase solutions**

PIPELINE

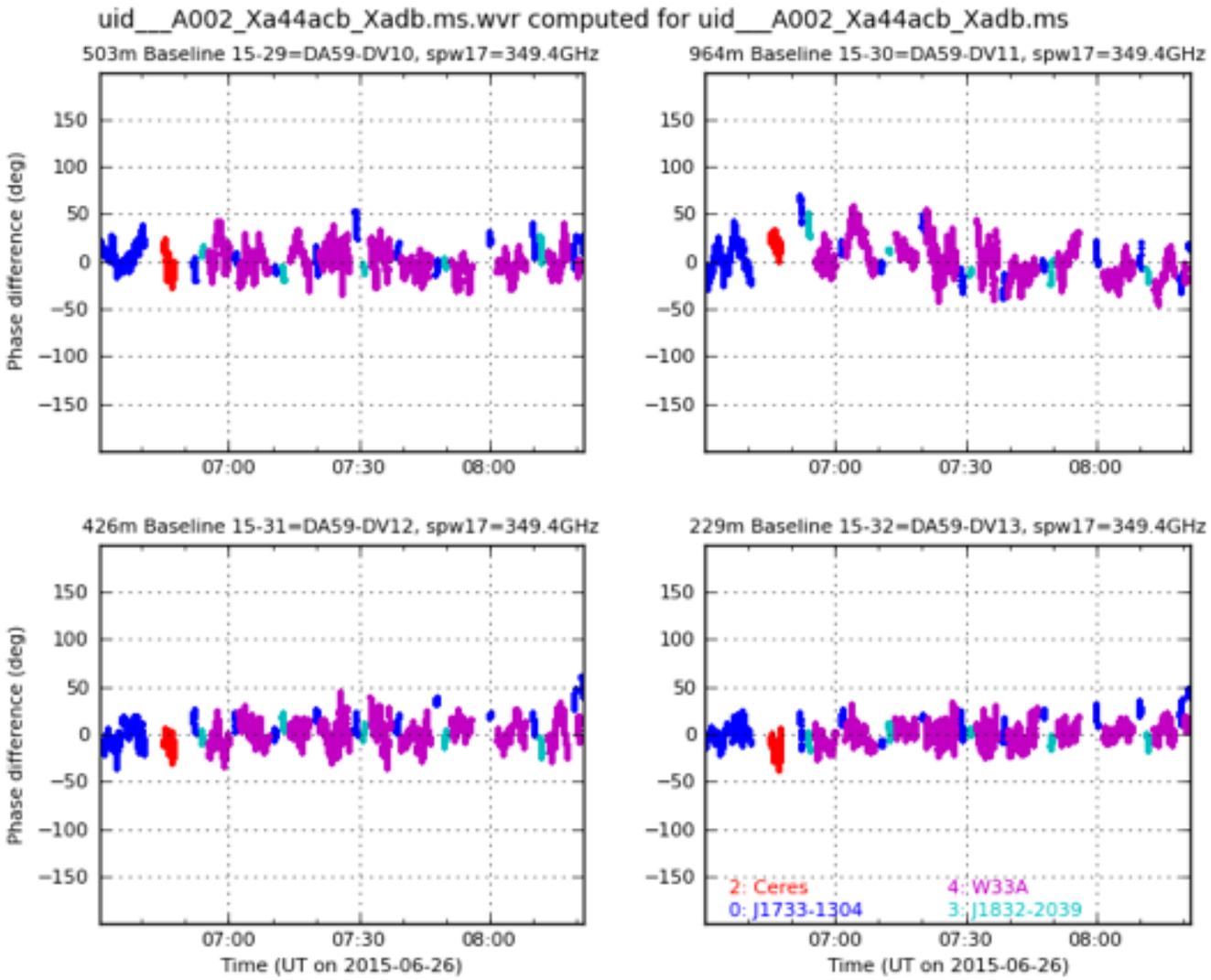
9. hifa_wvrgcalflag
10. hif_lowgainflag
11. hif_gainflag
12. hif_setjy
13. hifa_bandpass
14. hifa_spwphaseup

Weblog

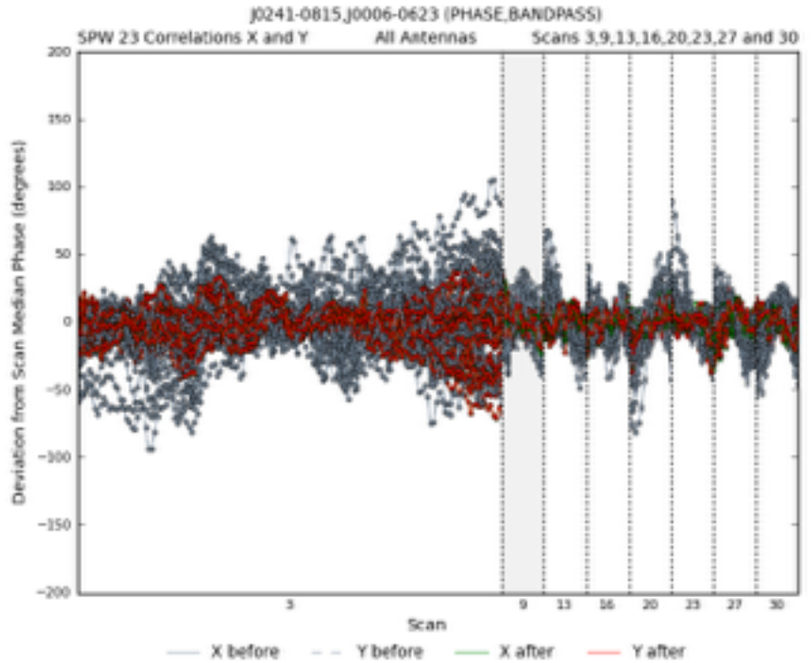
WHY? : Correct for atmospheric phase variations caused by the water vapour - short term
Plots : in 'QA' (or 'calibration') directory / weblog
'*wvr.smooth.plots'

Calibration - Tsys, AntPos, WVR

MANUAL



PIPELINE

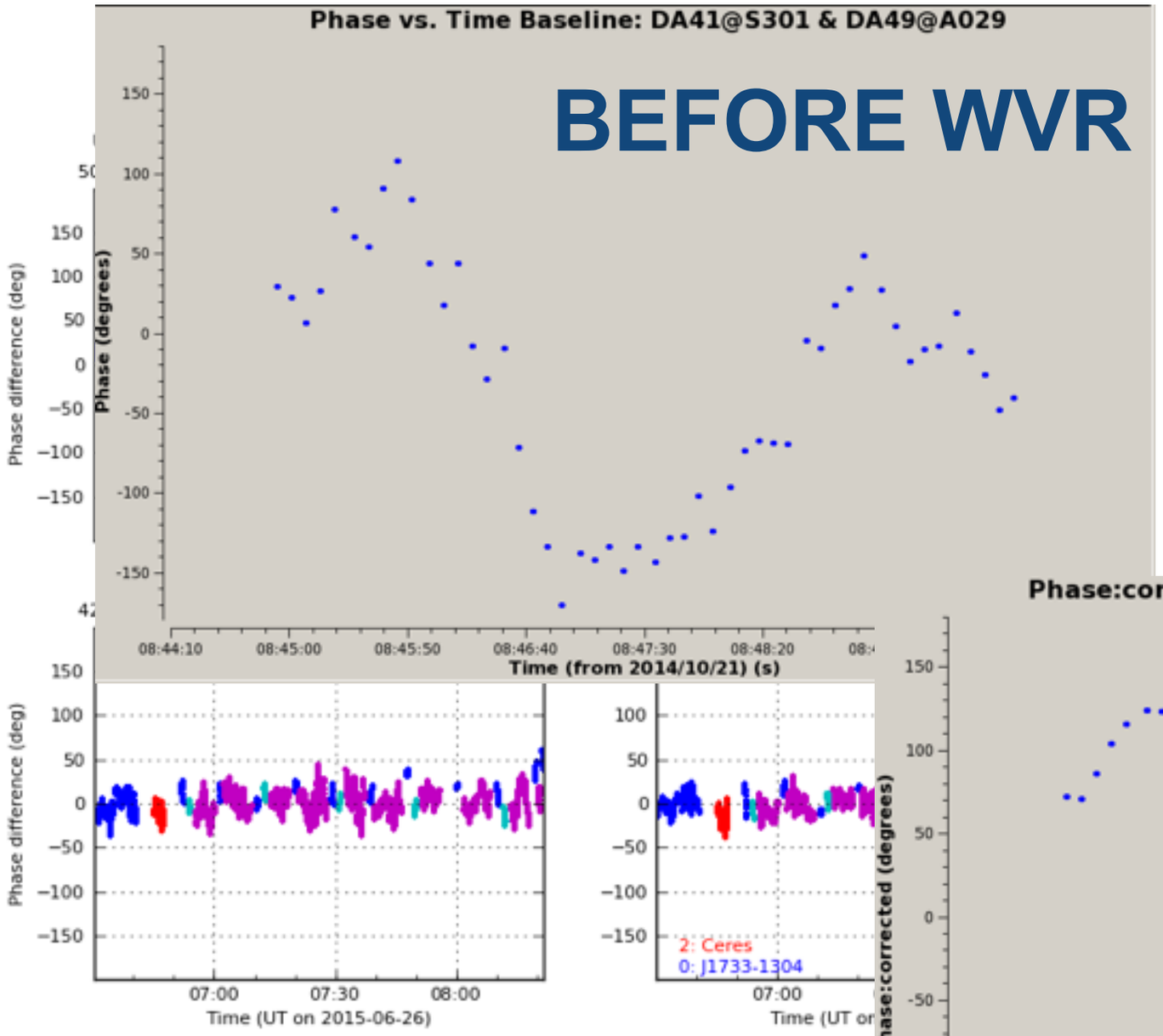


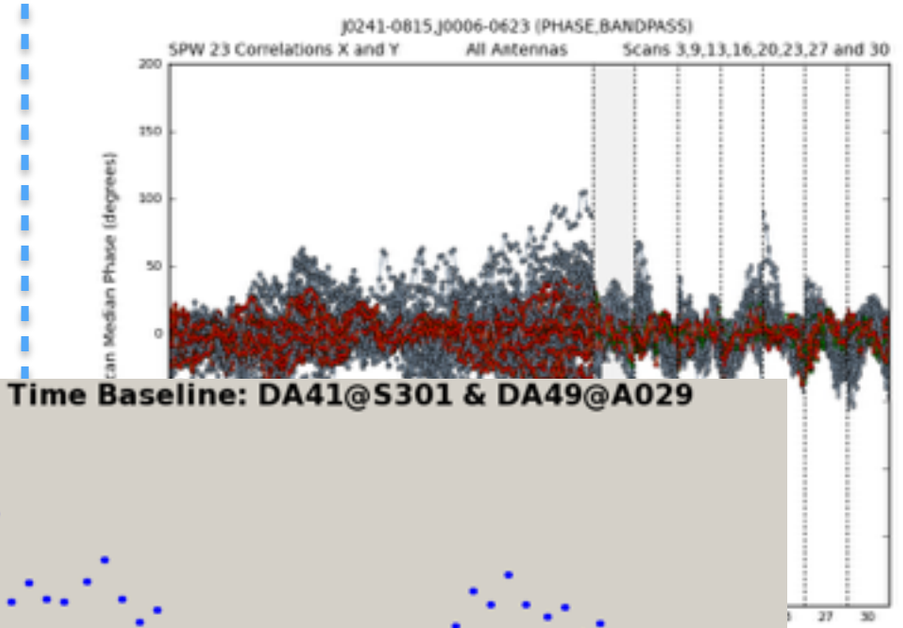
The atmosphere is continually varying - when on source and when on the calibrators - WVR acts to reduce the phase variations on ALL sources!

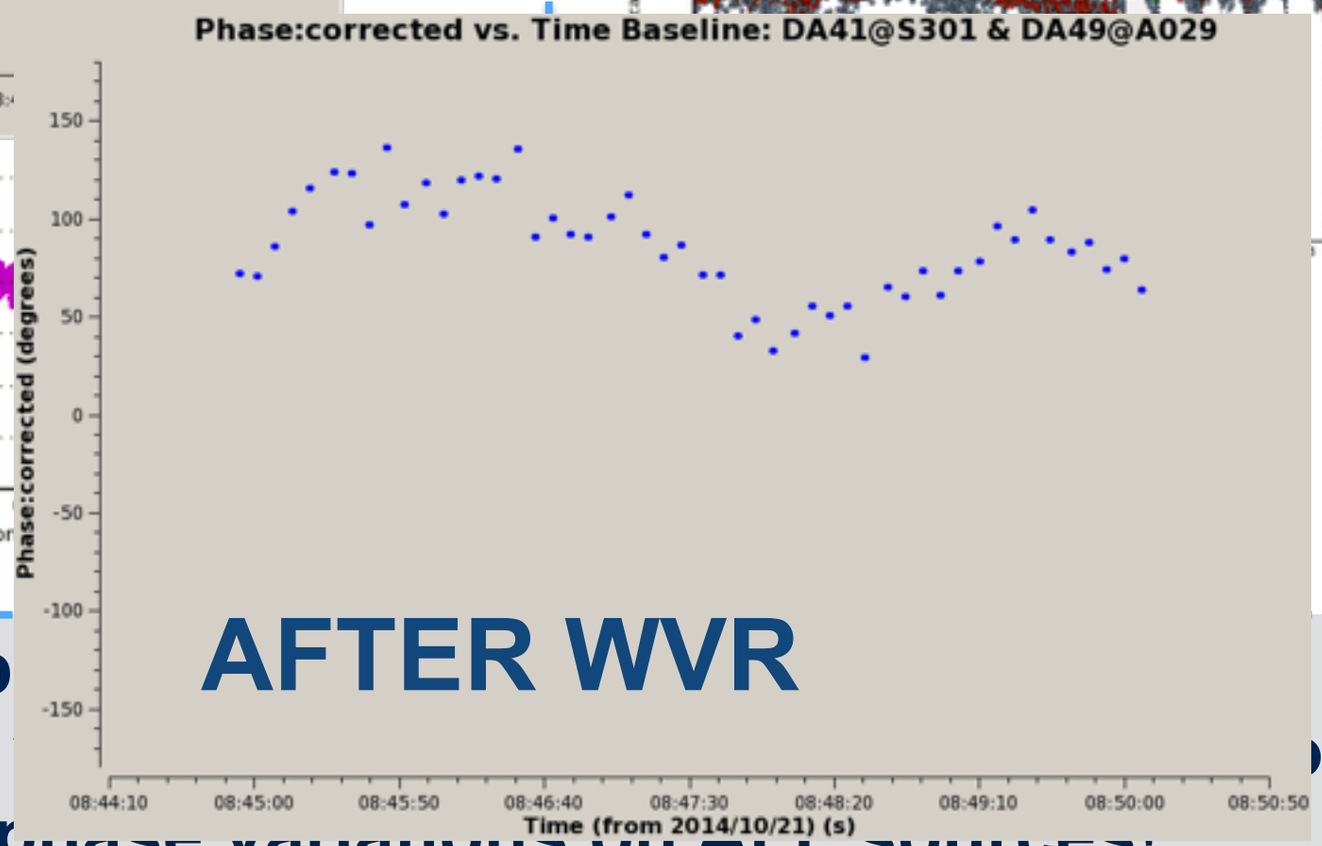
Calibration - Tsys, AntPos, WVR

PIPELINE

BEFORE WVR







AFTER WVR

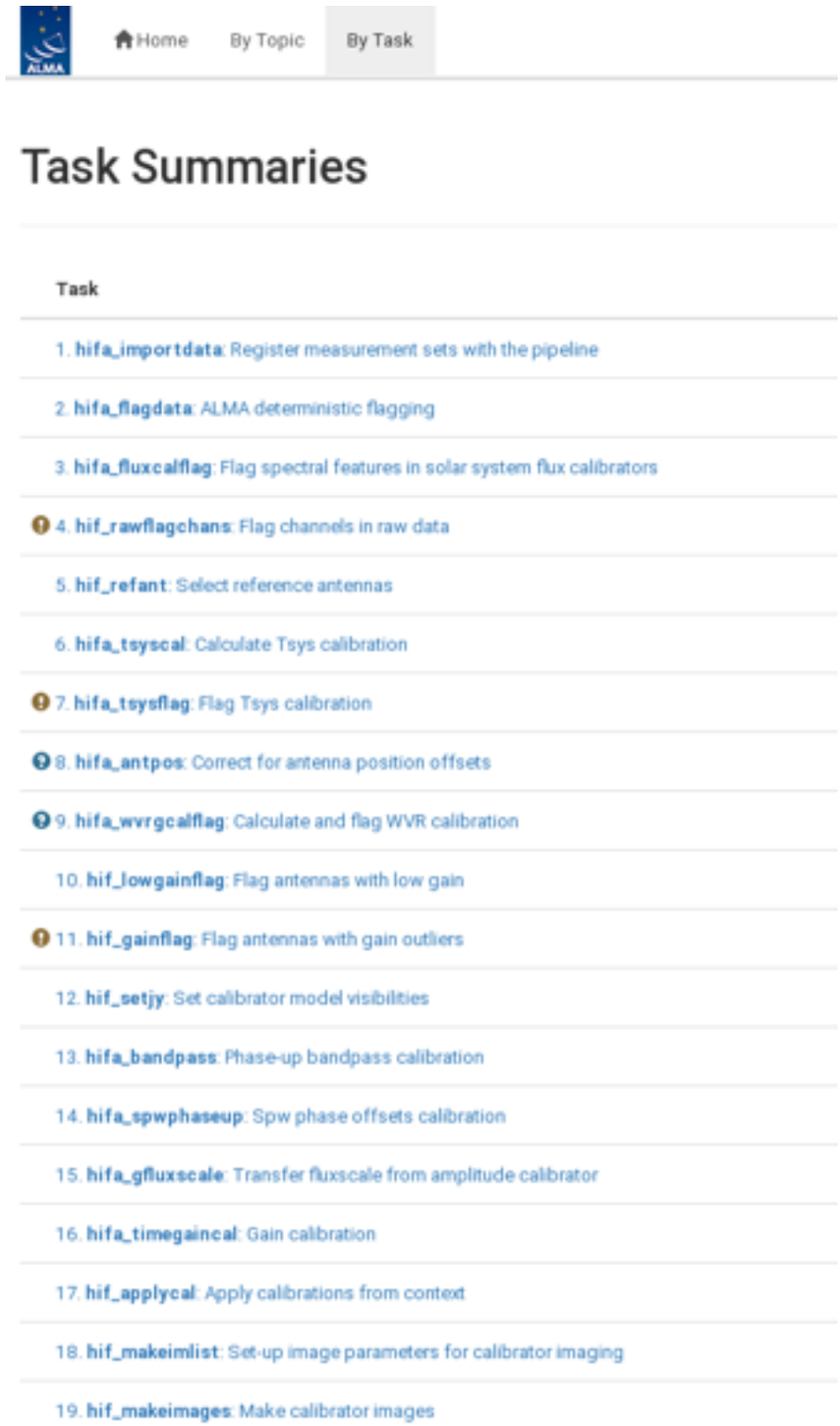
The atmosp
source and
reduce the phase variations on ALL sources!

Calibration - intermediate ‘split’

MANUAL

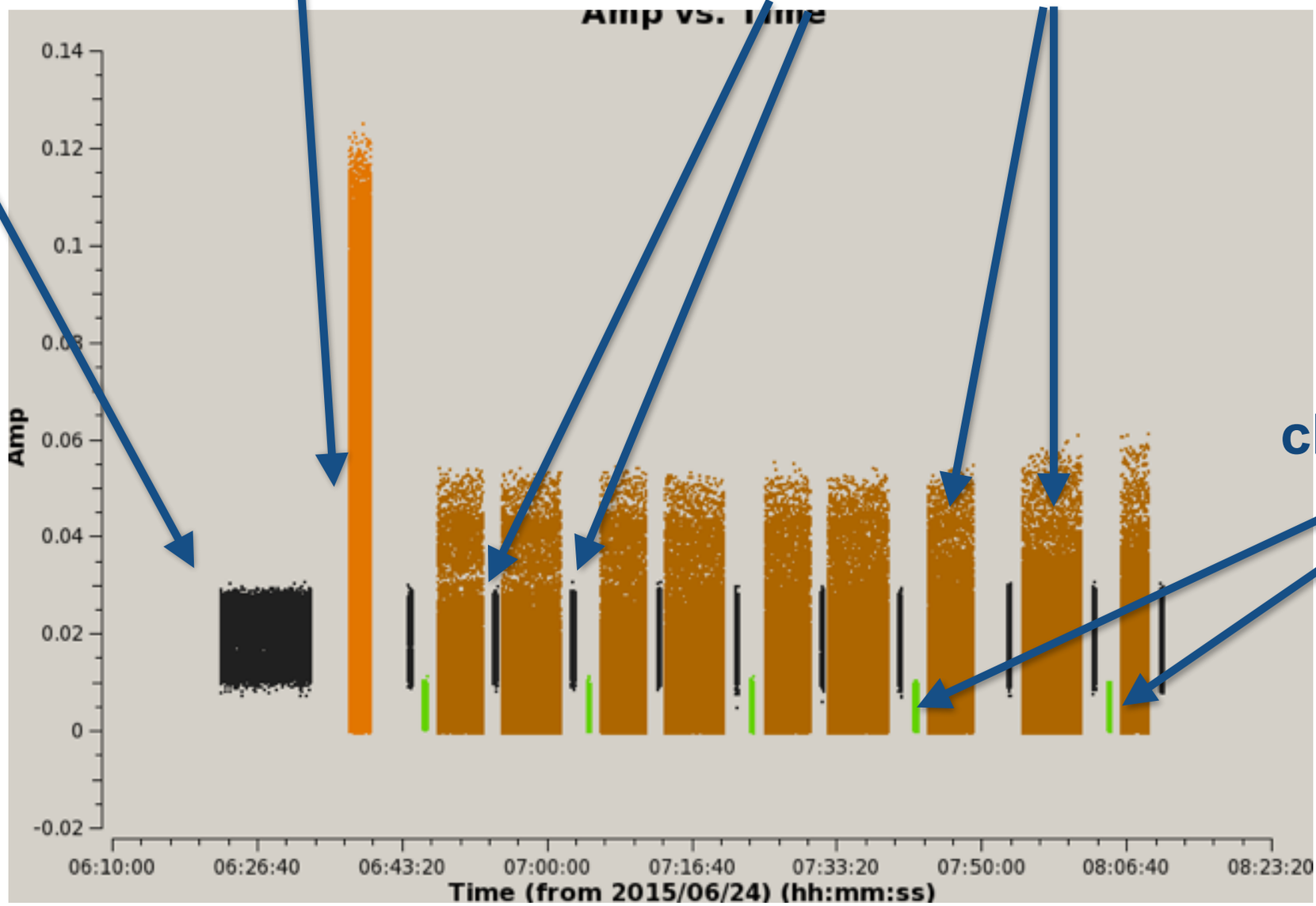
```
# ALMA Data Reduction Script

# Calibration

thesteps = []
step_title = {0: 'Import of the ASDM',
              1: 'Fix of SYSCAL table times',
              2: 'listobs',
              3: 'A priori flagging',
              4: 'Generation and time averaging of the WVR cal table',
              5: 'Generation of the Tsys cal table',
              6: 'Generation of the antenna position cal table',
              7: 'Application of the WVR, Tsys and antpos cal tables',
              8: 'Split out science SPWs and time average',
              9: 'Listobs, and save original flags',
              10: 'Initial flagging',
              11: 'Putting a model for the flux calibrator(s)',
              12: 'Save flags before bandpass cal',
              13: 'Bandpass calibration',
              14: 'Save flags before gain cal',
              15: 'Gain calibration',
              16: 'Save flags before applycal',
              17: 'Application of the bandpass and gain cal tables',
              18: 'Split out corrected column',
              19: 'Save flags after applycal'}

if 'applyonly' not in globals(): applyonly = False
try:
  print 'List of steps to be executed ...', mysteps
  thesteps = mysteps
except:
  print 'global variable mysteps not set.'
if (thesteps==[]):
  thesteps = range(0,len(step_title))
  print 'Executing all steps: ', thesteps
```

PIPELINE



Calibration - The data

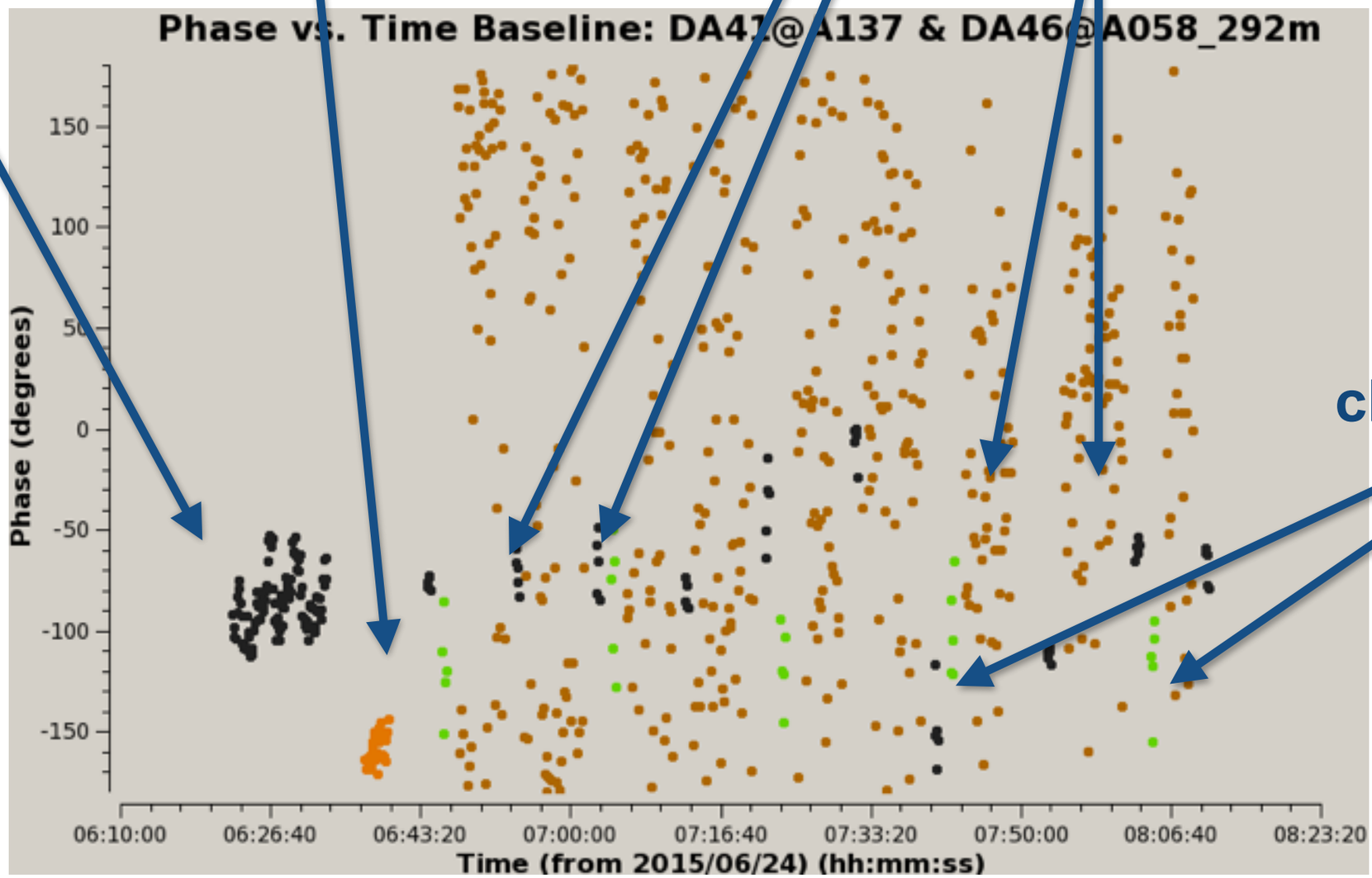
Bandpass - Flux Cal - [Gain Cal - Source] x repeat



Plots : plotms(vis) - set colorize to ‘field’ & channel averaging to 999999 - yaxis = ‘amp’
weblog - hif_applycal (after calibration only)

Calibration - The data

Bandpass - Flux Cal - [Gain Cal - Source] x repeat



check source

Plots : plotms(vis) - set colorize to ‘field’ & channel averaging to 999999 - yaxis = ‘phase’, iteraxis=‘baseline’ weblog - hif_applycal (after calibration only)

Calibration - Bandpass

- **The Source**
  - **strong source with enough S/N per channel** (caution - v.high S/N if your target is strong in continuum and you search for a weak line - check with your Contact Scientist)
  - **observed 5-10 minutes at start of observing run**
- **The Steps**
  - ***1*** **- must 'phase-up' - simple phase calibration on selected narrow range of channels to correct for decorrelation with time**
  - ***2*** **- apply phase solution 'on-the-fly' and correct for the frequency response**

Calibration - Bandpass (1)

MANUAL

```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ap_pre_bandpass',
  field = '0', # J1733-1304
  spw = '0:1536~2304,1:1536~2304,2:1536~2304,3:1536~2304',
  scan = '1,2,',
  solint = 'int',
  refant = 'DA59',
  calmode = 'p')
```

- **gaincal - creates antenna based solutions to solve phases**
  - **caltable - table that will hold solutions**
  - **spw - select centre of bandwidth (dependent on width)**
  - **calmode - p for phase only**
  - **refant - reference antenna - centrally located**

PIPELINE

13. hifa_bandpass
14. hifa_spwphaseup
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Weblog

WHY? :Phase up - solve phases with time !
Plots : in 'QA' (or 'calibration') directory / weblog '*ap_pre_bandpass.plots'

Calibration - Bandpass (1)

MANUAL

```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ap_pre_bandpass',
  field = '0', # J1733-130
  spw = '0:1536~2304,1:15
  scan = '1,2,',
  solint = 'int',
  refant = 'DA59',
  calmode = 'p')
```

solint = ‘int’
INT is the integration time
Usually 6 sec

- **gaincal - creates antenna based solutions to solve phases**
  - **caltable - table that will hold solutions**
  - **spw - select centre of bandwidth (dependent on width)**
  - **calmode - p for phase only**
  - **refant - reference antenna - centrally located**

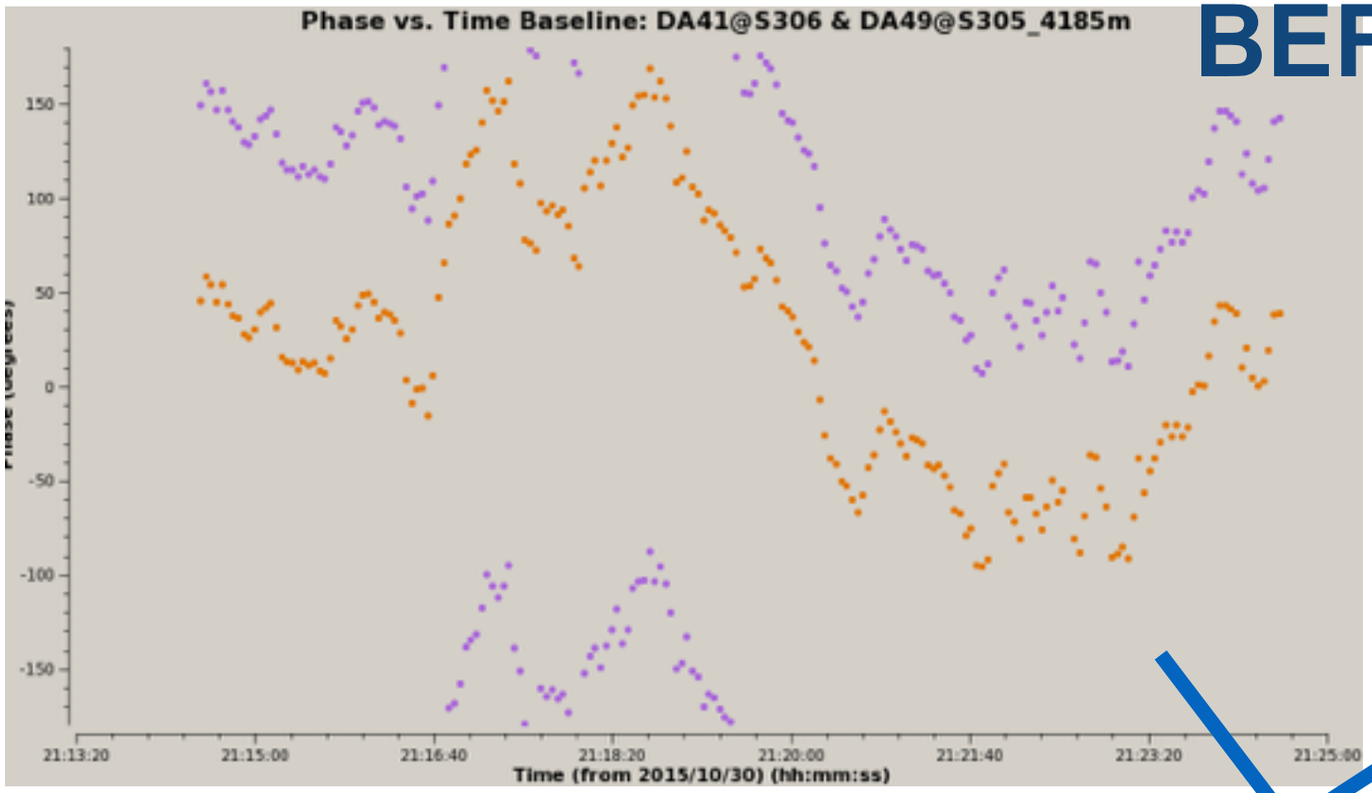
PIPELINE

13. hifa_bandpass
14. hifa_spwphaseup
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Weblog

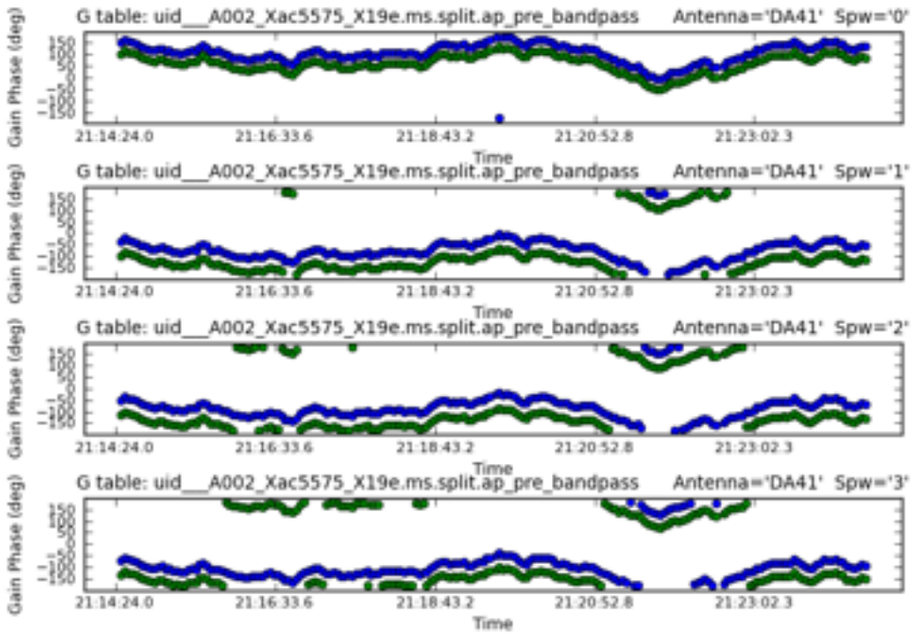
WHY? :Phase up - solve phases with time !
Plots : in ‘QA’ (or ‘calibration’) directory / weblog ‘*ap_pre_bandpass.plots’

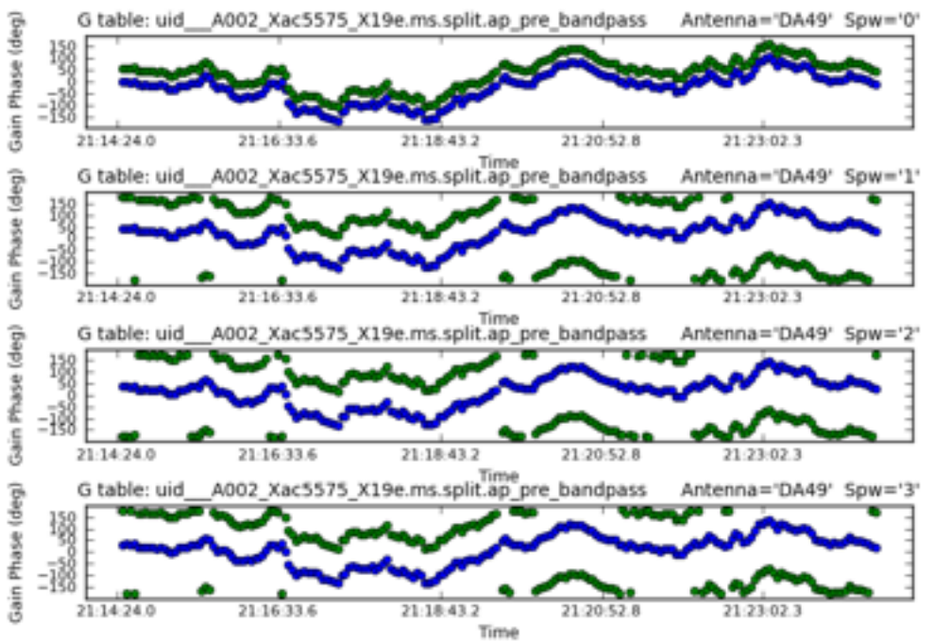
Calibration - Bandpass (1)



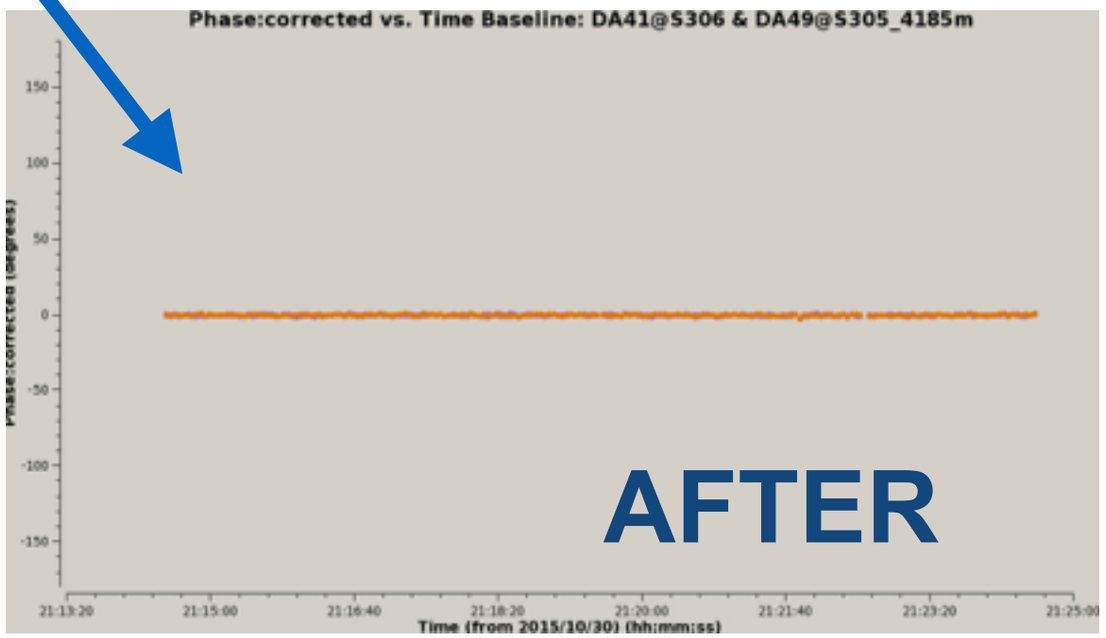
BEFORE

DA41 - Soln.





DA49 - Soln.



AFTER

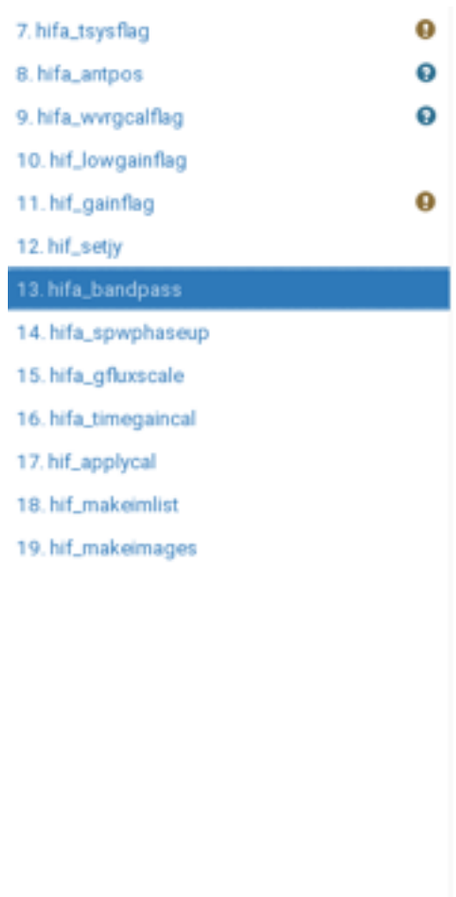
Calibration - Bandpass (2)

MANUAL

```
bandpass(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch',
  field = '0', # J1733-1304
  scan = '1,2,',
  solint = 'inf,4MHz',
  combine = 'scan',
  refant = 'DA59',
  solnorm = True,
  bandtype = 'B',
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.ap_pre_bandpass')
```

- **bandpass** - creates antenna based solutions for amp & phase with Freq.
  - **caltable** - table that will hold solutions
  - **combine** - scan - all data needs to be combined for each SPW
  - **bandtype** - B for bandpass. BP for B-poly if you bandpass is noisy - use with care

PIPELINE



Weblog

WHY? :Solve for the frequency dependence
Plots : in 'QA' (or 'calibration') directory / weblog
'*bandpass.plots' & '*bandpass_smoothXXXX.plots'

Calibration - Bandpass (2)

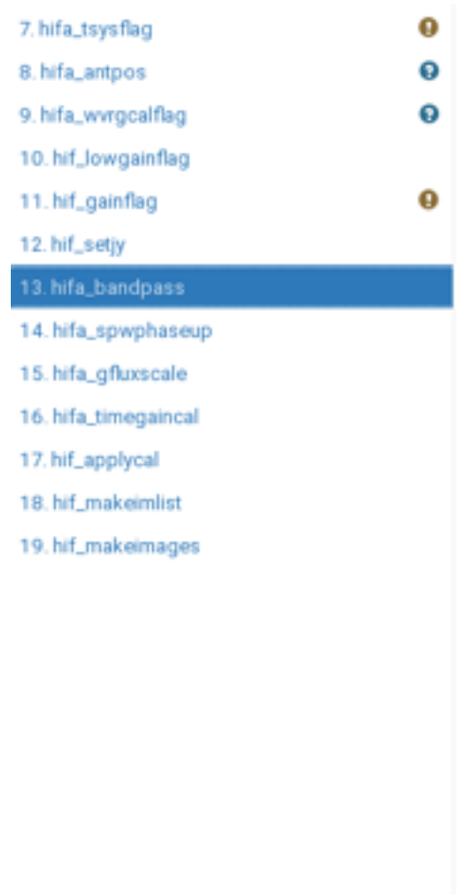
MANUAL

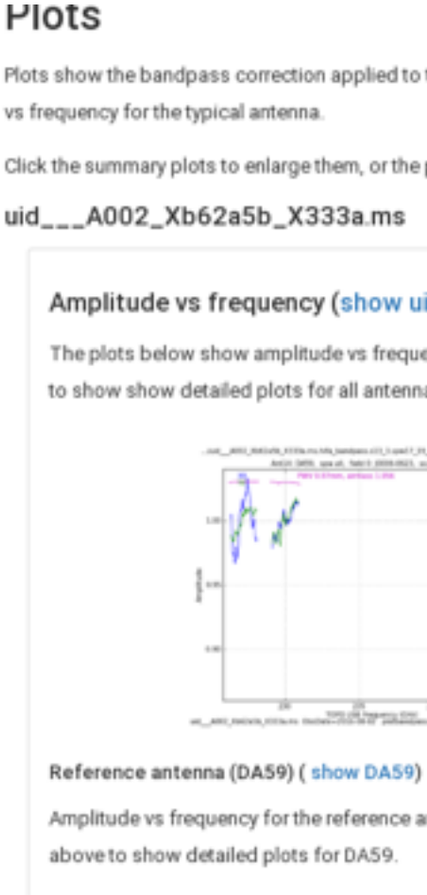
```
bandpass(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch',
  field = '0', # J1733-1304
  scan = '1,2,',
  solint = 'inf,4MHz',
  combine = 'scan',
  refant = 'DA59',
  solnorm = True,
  bandtype = 'B',
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.ap_pre_bandpass')
```

- **solint - inf to average in time 4MHz or XXch for averaging in frequency**

- **bandpass - creates antenna based solutions for amp & phase with Freq.**
  - **caltable - table that will hold solutions**
  - **combine - scan - all data needs to be combined for each SPW**
  - **bandtype - B for bandpass. BP for B-poly if you bandpass is noisy - use with care**

PIPELINE



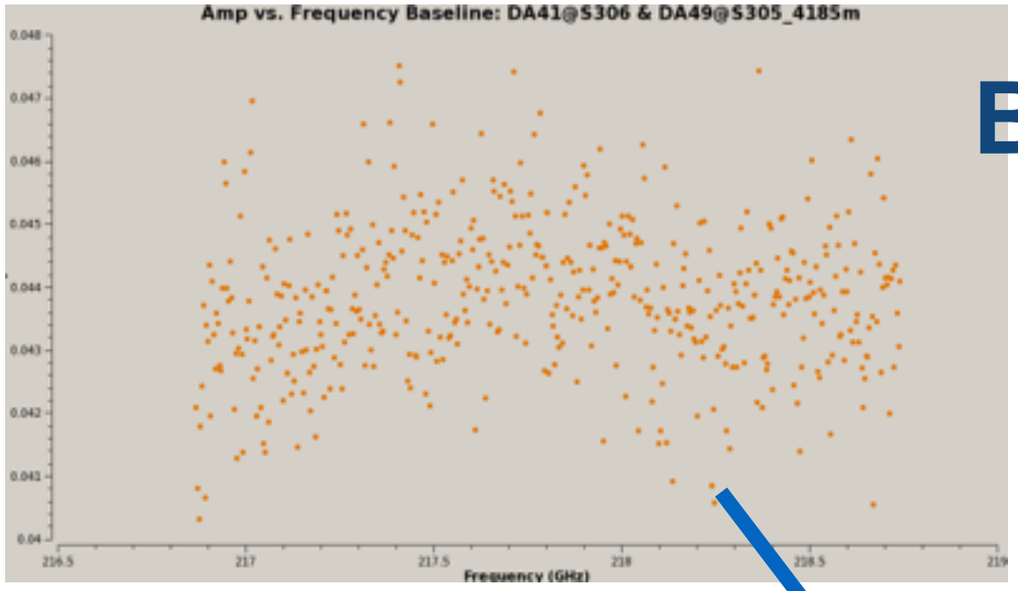


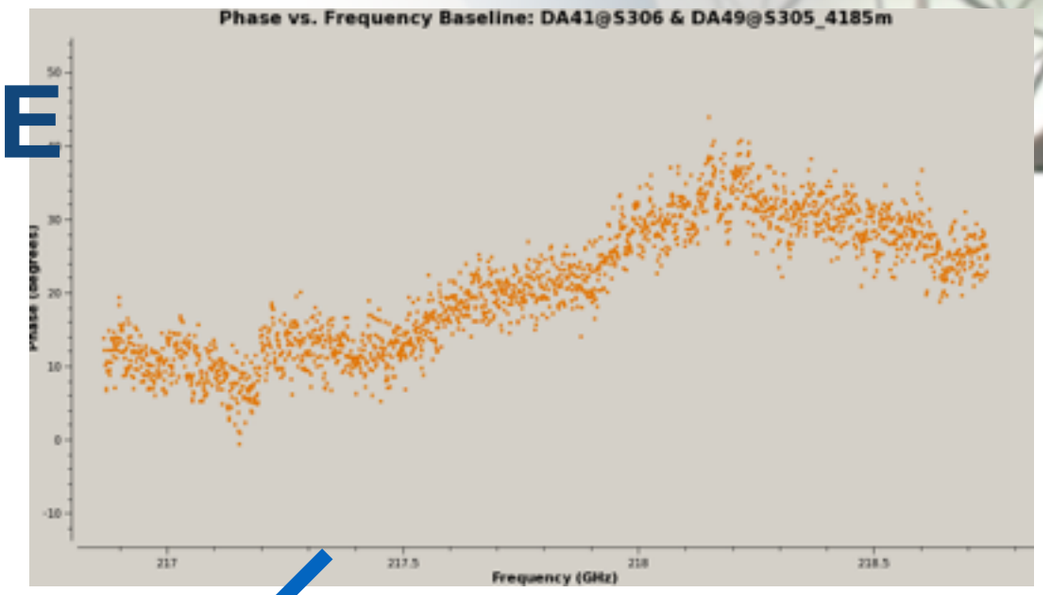
Weblog

WHY? :Solve for the frequency dependence
Plots : in 'QA' (or 'calibration') directory / weblog
'*bandpass.plots' & '*bandpass_smoothXXXX.plots'

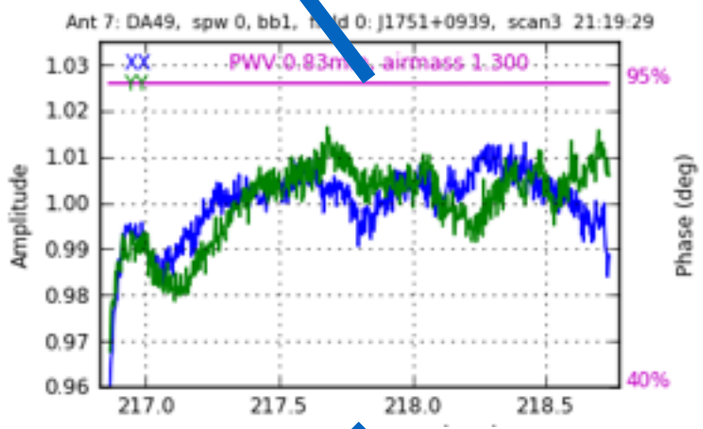
Calibration - Bandpass (2)

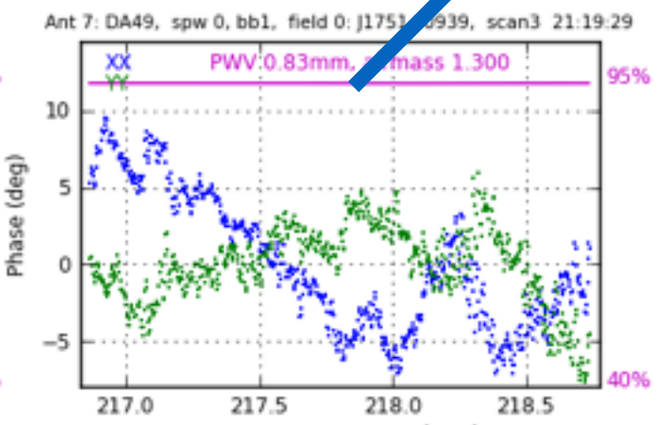
BEFORE





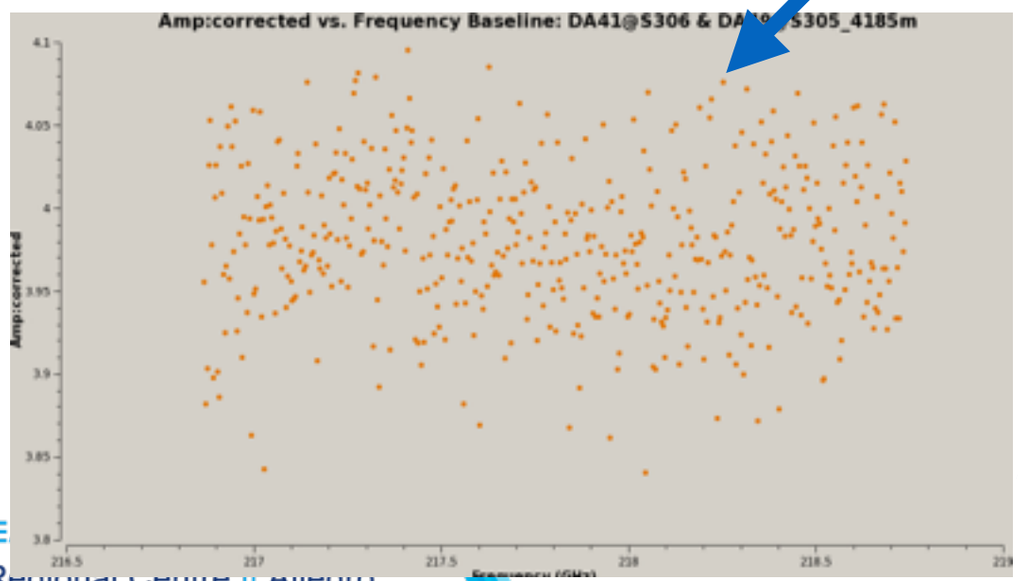
AMP.

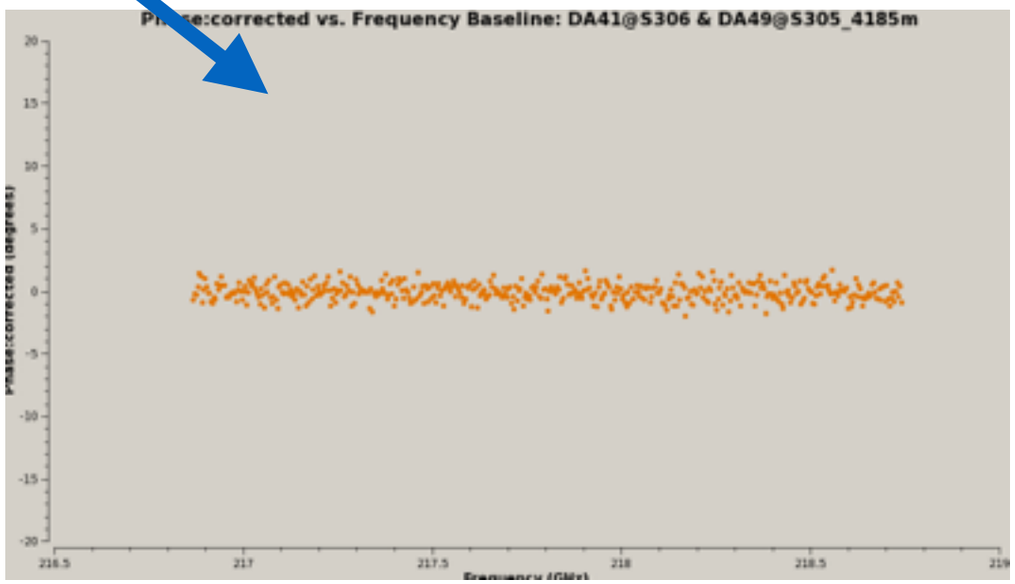




PHASE

AFTER





Calibration - Flux

- **The Source**
  - **Solar System Object (SSO) - caution too resolved is an issue**
  - **QSO known/monitored source**
- **The Steps**
  - ***1* - use 'setjy' to set the flux scaling to refer to later**
  - ***GC1*- phase up all sources required for flux scaling to solve for decorrelation - flux cal, BP cal, gain cal (caution with resolved sources!!!)**
  - ***GC2*- apply the phase-up 'on-the-fly' and solve the amplitudes**
  - **2 - scale the amplitude gains according to the output flux of the flux calibrator - bootstrapping !**

Gain cal steps

Calibration - Flux (1)

MANUAL

SSO

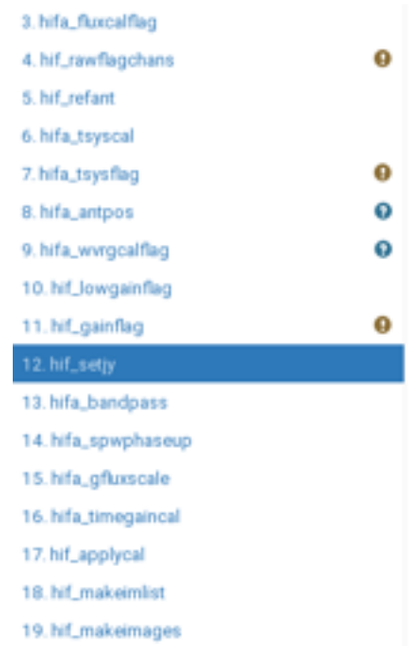
```
setjy(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  field = '2', # Ceres
  spw = '0,1,2,3',
  standard = 'Butler-JPL-Horizons 2012')
```

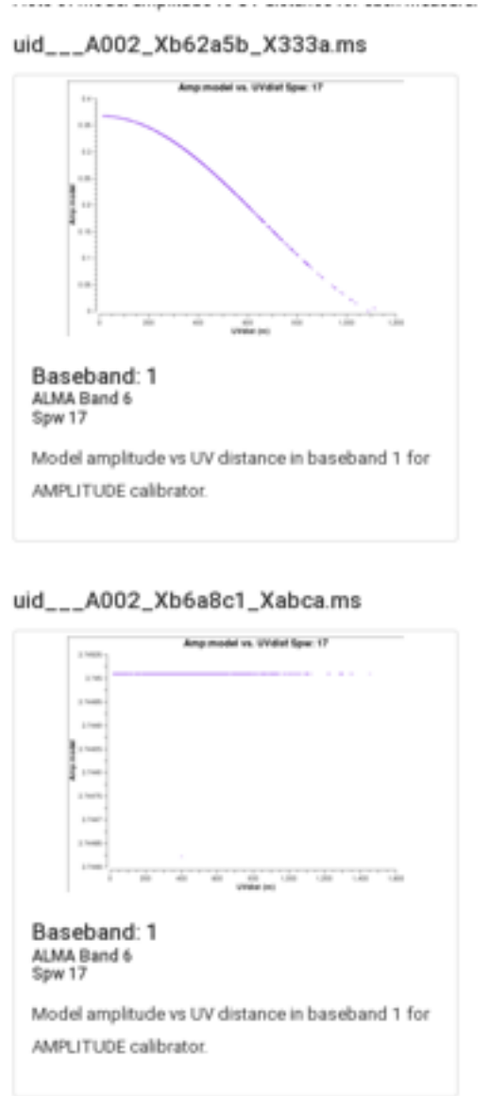
QSO

```
setjy(vis = 'uid___A002_Xac5575_X19e.ms.split',
  standard = 'manual',
  field = 'J1751+0939',
  fluxdensity = [4.04889627574, 0, 0, 0],
  spix = -0.476483659328,
  reffreq = '226.451094779GHz')
```

- **setjy - sets the flux scaling for the flux calibrator source**
- **standard - the model OR manual**

PIPELINE





Weblog

WHY? :Setting the correct flux scale to use later
Plots : in ‘QA’ directory / weblog

Calibration - Flux (1)

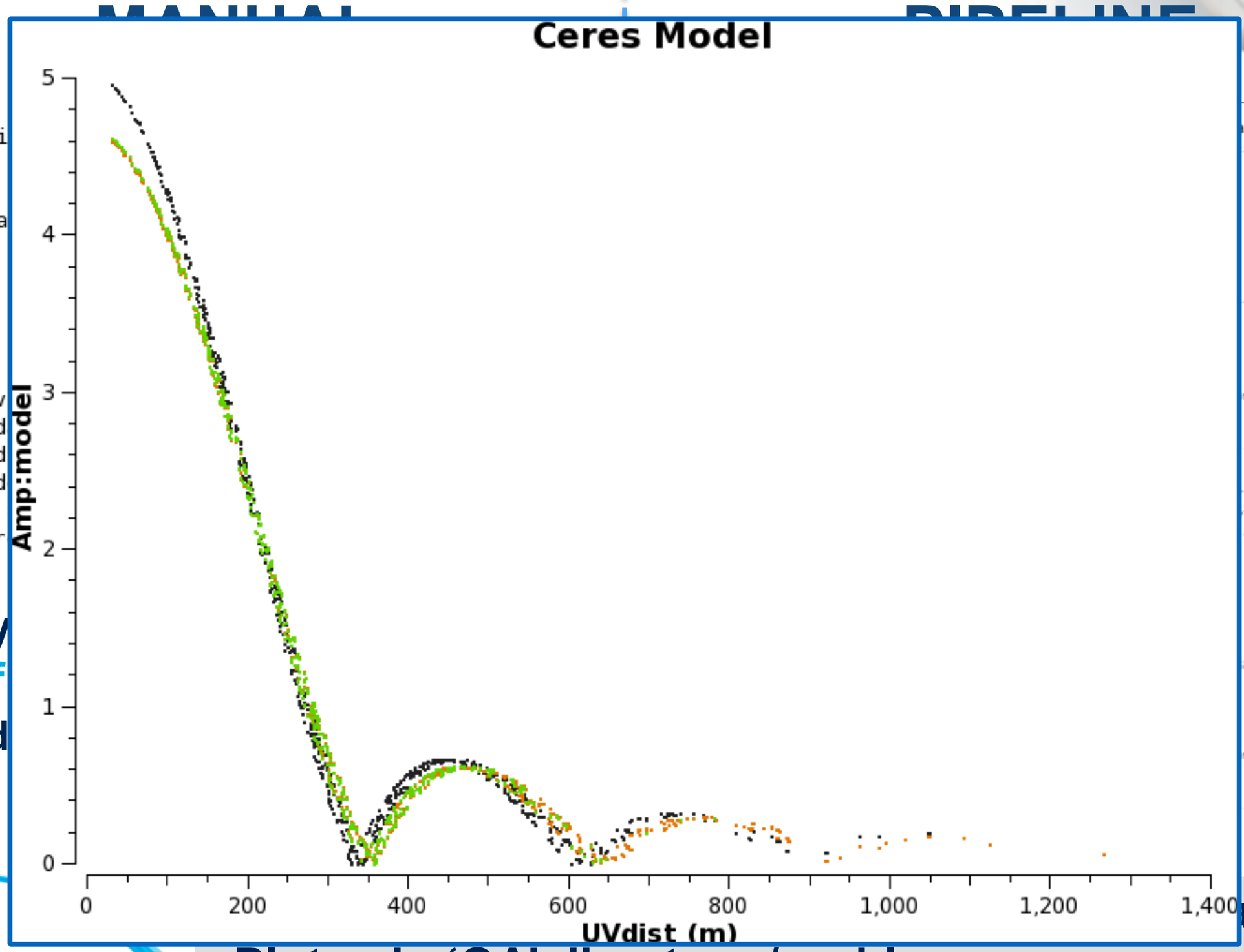
SSO

```
setjy(vi
  field
  spw =
  standa
```

QSO

```
setjy(v
  stand
  field
  fluxd
  spix
  reffr
```

- **setjy**
  the f
- **stand**



Plots : in ‘QA’ directory / weblog

Calibration - Gains

- **The Source**
  - **QSO point source - known amp and phase visibilities**
- **What?**
  - **Short term phases**
  - **Long term amplitudes**
  - **~Long term phases - per visit to phase calibrator**
- **The steps**
  - ***1*** **- 'phase-up' to solve for decorrelation due to rapid phase changes cause be atmosphere -** ***required to get the correct amplitudes**
  - ***2*** **- Slow variation of amplitude with time - solved with 'phase-up' solutions applied 'on-the-fly'**
  - ***3*** **- Phase variations due to atmospheric changes are monitored by the gain calibrator and interpolated to the source**

Calibration - Gains (1) - phase ‘int’

MANUAL

```
gaincal(vis = 'uid___A002_Xac5575_X19e.ms.split',
  caltable = 'uid___A002_Xac5575_X19e.ms.split.phase_int',
  field = '0,1,2', # J1751+0939,J1924+1540  - J1922+1530
  solint = 'int',
  refant = 'DV10',
  gaintype = 'G',
  calmode = 'p',
  gaintable = 'uid___A002_Xac5575_X19e.ms.split.bandpass')
```

- gaincal - creates the antenna based phase solutions
  - field - ALL calibrator sources
  - solint - ‘int’ should be selected for best phase solutions
  - gaintable - apply bandpass OTF

PIPELINE

14. hifa_spwphaseup
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Weblog

WHY? :Correct for all phase variations in time
Plots : in ‘QA’ directory / weblog
‘*split.phase_int.plots’

Calibration - Gains (1) - phase ‘int’

MANUAL

```
gaincal(vis = 'uid___A002_Xac5575_X19e.ms.split',
  caltable = 'uid___A002_Xac5575_X19e.ms.split.phase_int',
  field = '0,1,2', # J1751+0939,J1924+1540  - J1922+1530
  solint = 'int',
  refant = 'DV10',
  gaintype = 'G',
  calmode = 'p',
  gaintable = 'uid___A002_Xac5575_X19e.ms.split.bandpass')
```

- **gaincal - creates the antenna based phase solutions**
  - **field - ALL calibrator sources**
  - **solint - ‘int’ should be selected for best phase solutions**
  - **gaintable - apply bandpass OTF**

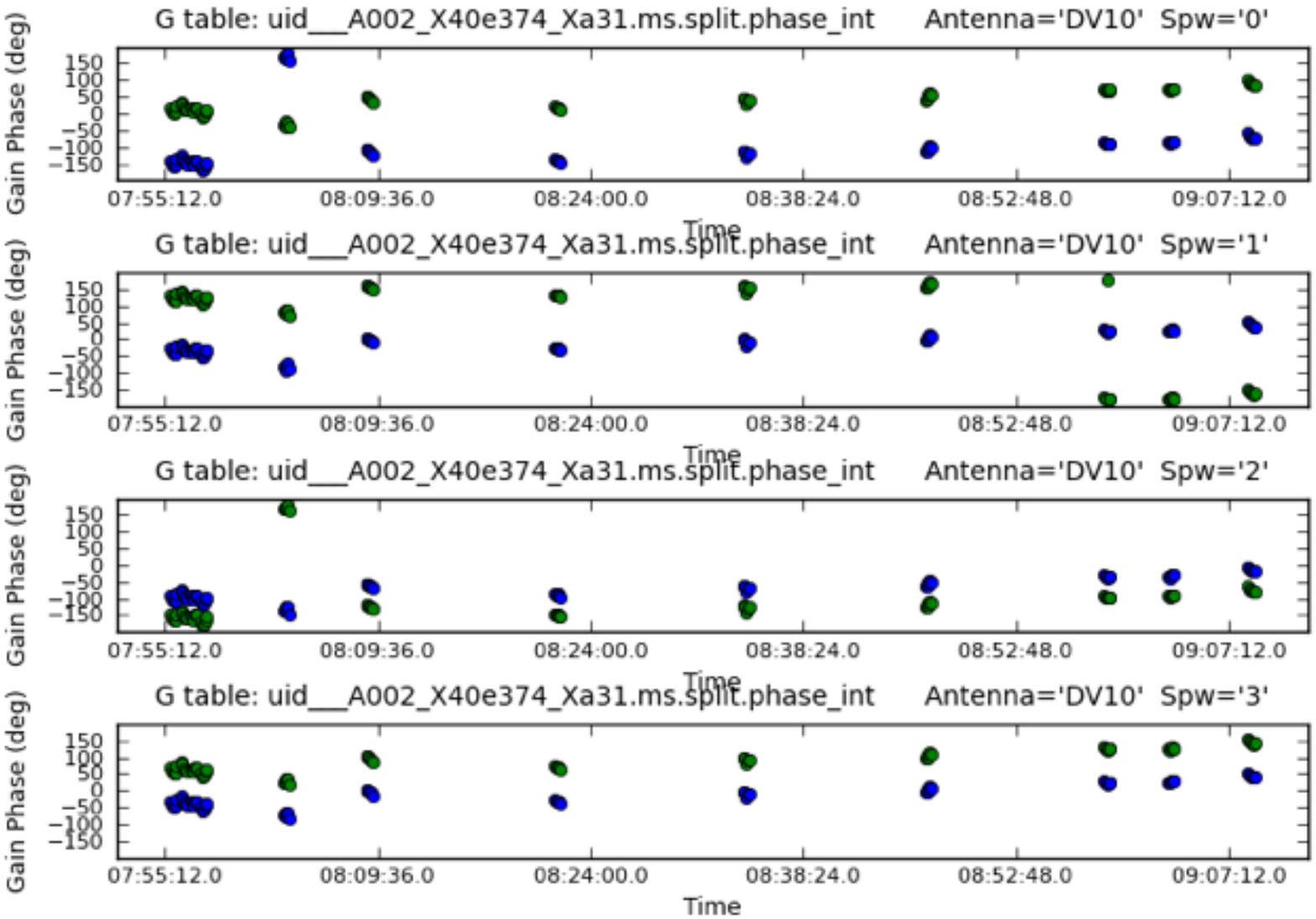
PIPELINE

14. hifa_spwphaseup
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Weblog

If one (or more) of your calibrators is weak (HF data) this can be increased or multiple gain tables are required, be aware this could cause fluxes to have a larger uncertainty unless atmosphere is stable

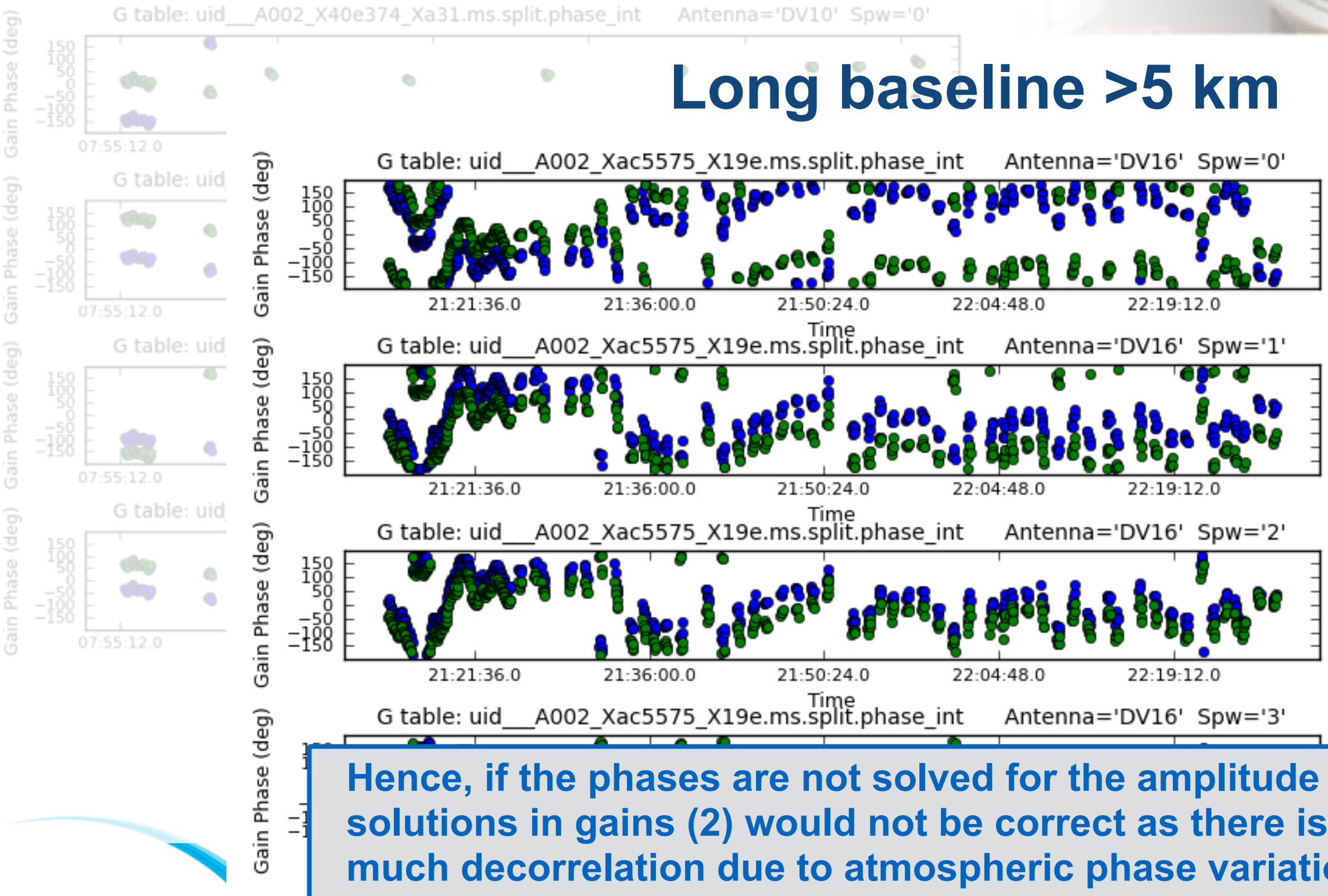
Calibration - Gains (1) - phase ‘int’



~short baseline <1 km

Calibration - Gains (1) - phase 'int'

Long baseline >5 km



Hence, if the phases are not solved for the amplitude solutions in gains (2) would not be correct as there is much decorrelation due to atmospheric phase variations (worse for longer baselines and higher frequencies)

Calibration - Gains (2) - amp ‘inf’

MANUAL

```
gaincal(vis = 'uid___A002_Xac5575_X19e.ms.split',
  caltable = 'uid___A002_Xac5575_X19e.ms.split.ampli_inf',
  field = '0,1,2', # J1751+0939,J1924+1540  J1922+1530
  solint = 'inf',
  refant = 'DV10',
  gaintype = 'T',
  calmode = 'a',
  gaintable = ['uid___A002_Xac5575_X19e.ms.split.bandpass',
5_X19e.ms.split.phase_int'])
```

- **gaincal - creates the antenna based amp solutions**
  - **field - ALL calibrator sources**
  - **solint - ‘inf’ i.e. per scan/visit to the gain cal**
  - **gaintable - apply bandpass AND the ‘int’ phase-up solutions OTF**

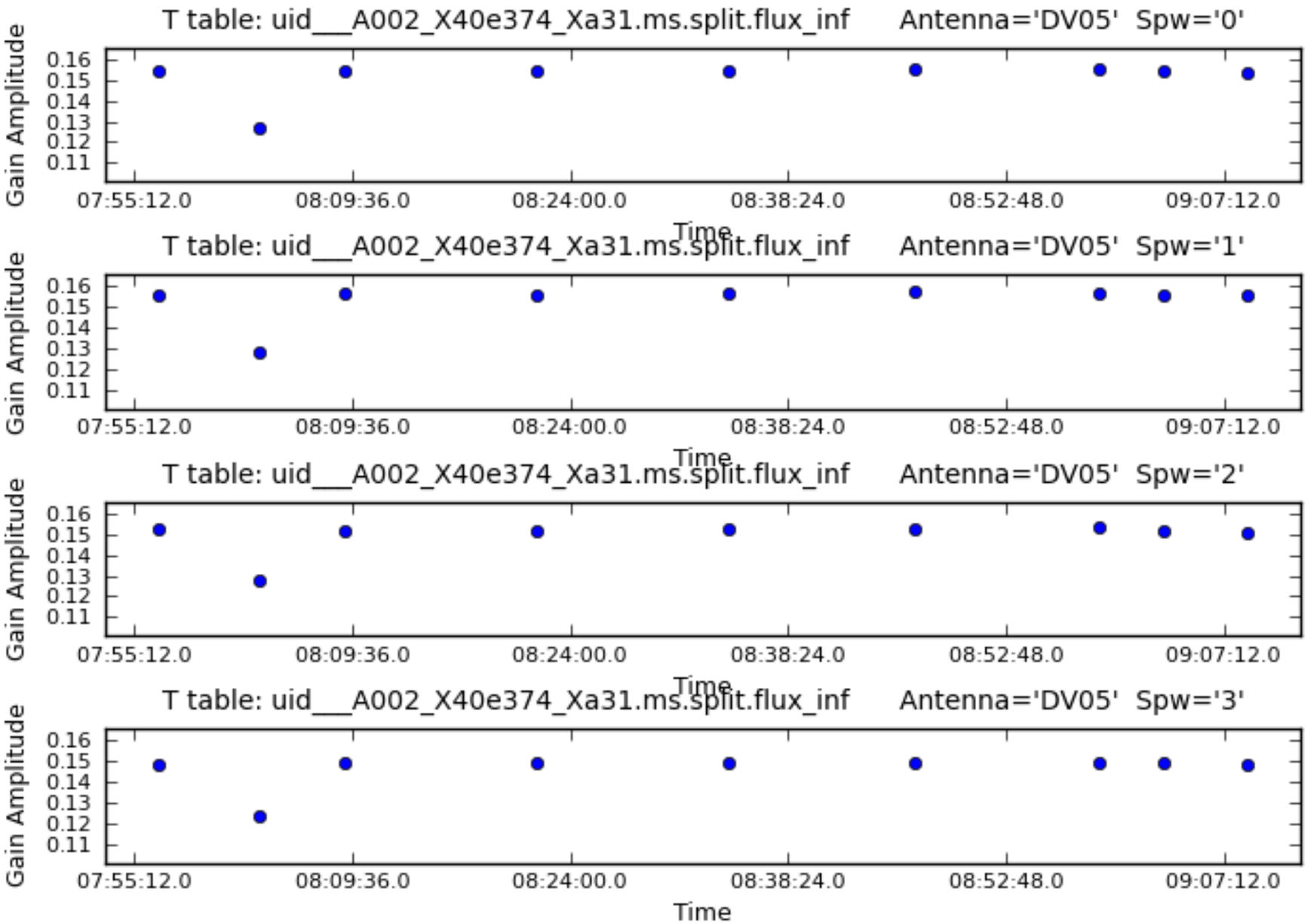
PIPELINE

15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Weblog

WHY? :Correct for slow amplitude variations in time
Plots : in ‘QA’ directory / weblog
‘*split.ampli_int.plots’ OR ‘*split.flux_inf.plots’

Calibration - Gains (2) - amp 'inf'



Solutions ONLY every visit to the gain calibrator

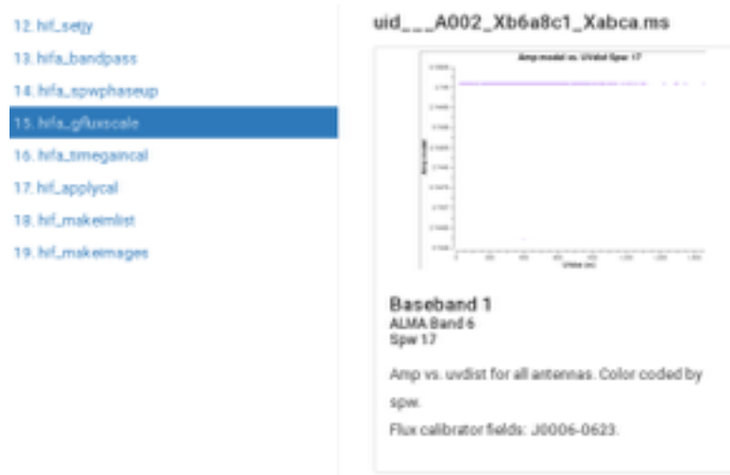
Calibration - Flux (2)

MANUAL

```
fluxscaleDict = fluxscale(vis = 'uid___A002_Xac5575_X19e.ms.split',
  caltable = 'uid___A002_Xac5575_X19e.ms.split.ampli_inf',
  fluxtable = 'uid___A002_Xac5575_X19e.ms.split.flux_inf',
  reference = '0') # J1751+0939
```

- **fluxscale - compares the input amplitude gain table and the model in setjy**
- **caltable - the long term amplitude gains previously solved**
- **fluxtable - the new output gain table with correct gains to scale fluxes**
- **reference - the source used with the setjy model earlier**

PIPELINE



WHY? :Setting the correct gains for flux scaling
Plots : in ‘QA’ directory / weblog
‘*split.flux_inf.plots’
File : ‘*split.fluxscale’

Calibration - Flux (2)

MANUAL

```
fluxscaleDict = fluxscale(vis = 'uid___A002_Xac5575_X19e.ms.split',
  caltable = 'uid___A002_Xac5575_X19e.ms.split.ampli_inf',
  fluxtable = 'uid___A002_Xac5575_X19e.ms.split.flux_inf',
  reference = '0') # J1751+0939
```

- fluxscale - compares the input amplitude gain table and the model in setjy
- caltable - the long term amplitude gains previously solved
- fluxtable - the new output gain table with correct gains to scale fluxes
- reference -

If one (or more) of your calibrators is weak (HF data) the flux reported could 'appear' higher some SPWs in particular narrow ones - as amplitude errors can ONLY be positive - can also map other SPW solutions

Plots : in 'QA' directory / weblog
'*split.flux_inf.plots'
File : '*split.fluxscale'

PIPELINE

Calibration - Gains (3) - phase ‘inf’

MANUAL

```
os.system('rm -rf uid___A002_Xa44acb_Xadb.ms.split.phase_inf')
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.phase_inf',
  field = '0,2,3', # J1733-1304,Ceres,J1832-2039
  solint = 'inf',
  refant = 'DA59',
  gaintype = 'G',
  calmode = 'p',
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch')
```

- **gaincal - creates the antenna based phase solutions**
  - **field - ALL calibrator sources**
  - **solint - ‘inf’ i.e. per scan/visit to the gain cal**
  - **gaintable - apply bandpass**

PIPELINE

16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

16. Gain Calibration

This task creates gain solutions for each measurement

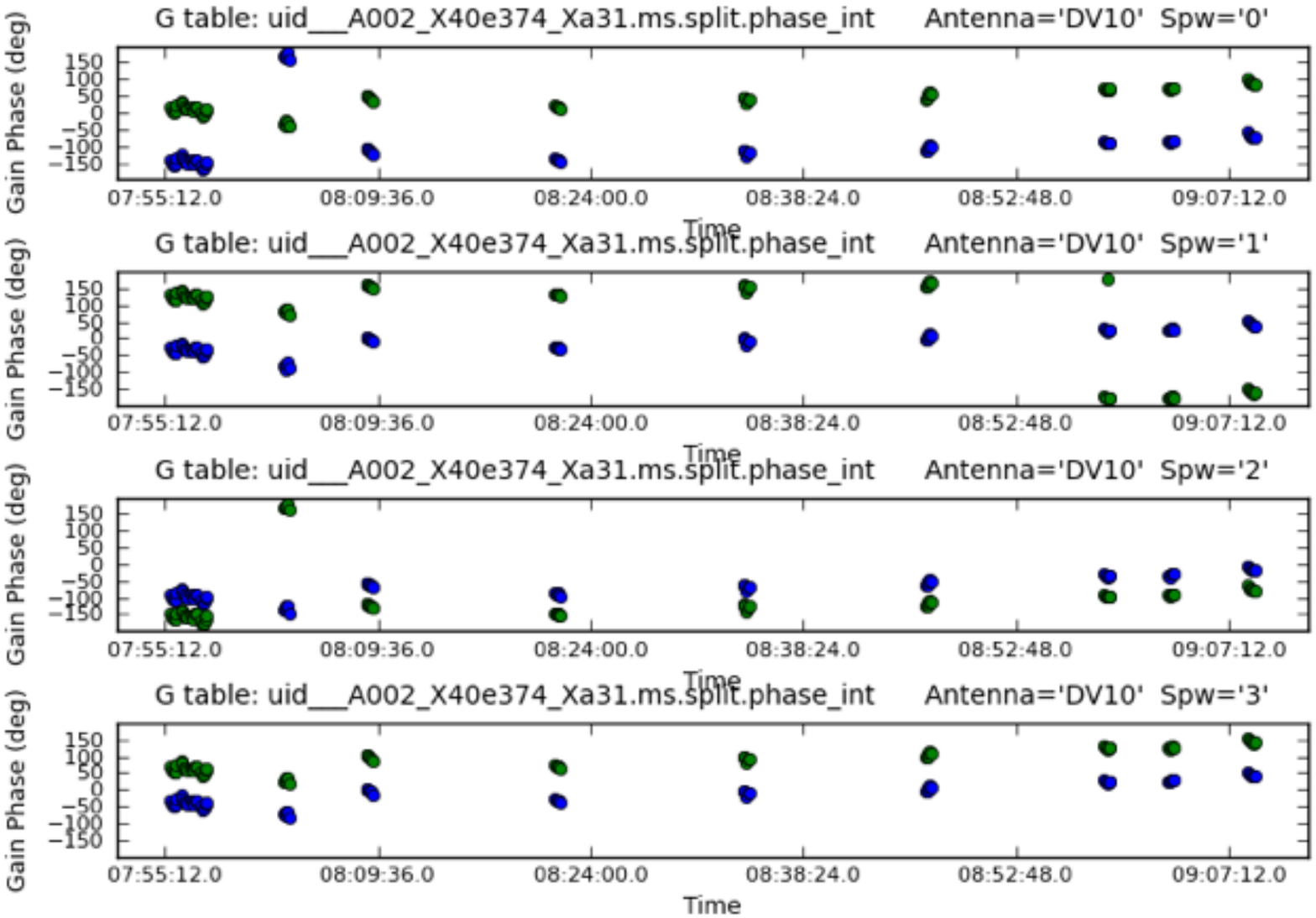
- Plots
  - Phase vs time
  - Phase structure
  - Amplitude vs time
- Diagnostic plots
  - Phase vs time
  - Amplitude vs time

Weblog

WHY? :Correct for phase variations in time (per scan) to apply to the science target
Plots : in ‘QA’ directory / weblog ‘*split.phase_inf.plots’

Calibration - Gains (3) - phase ‘inf’

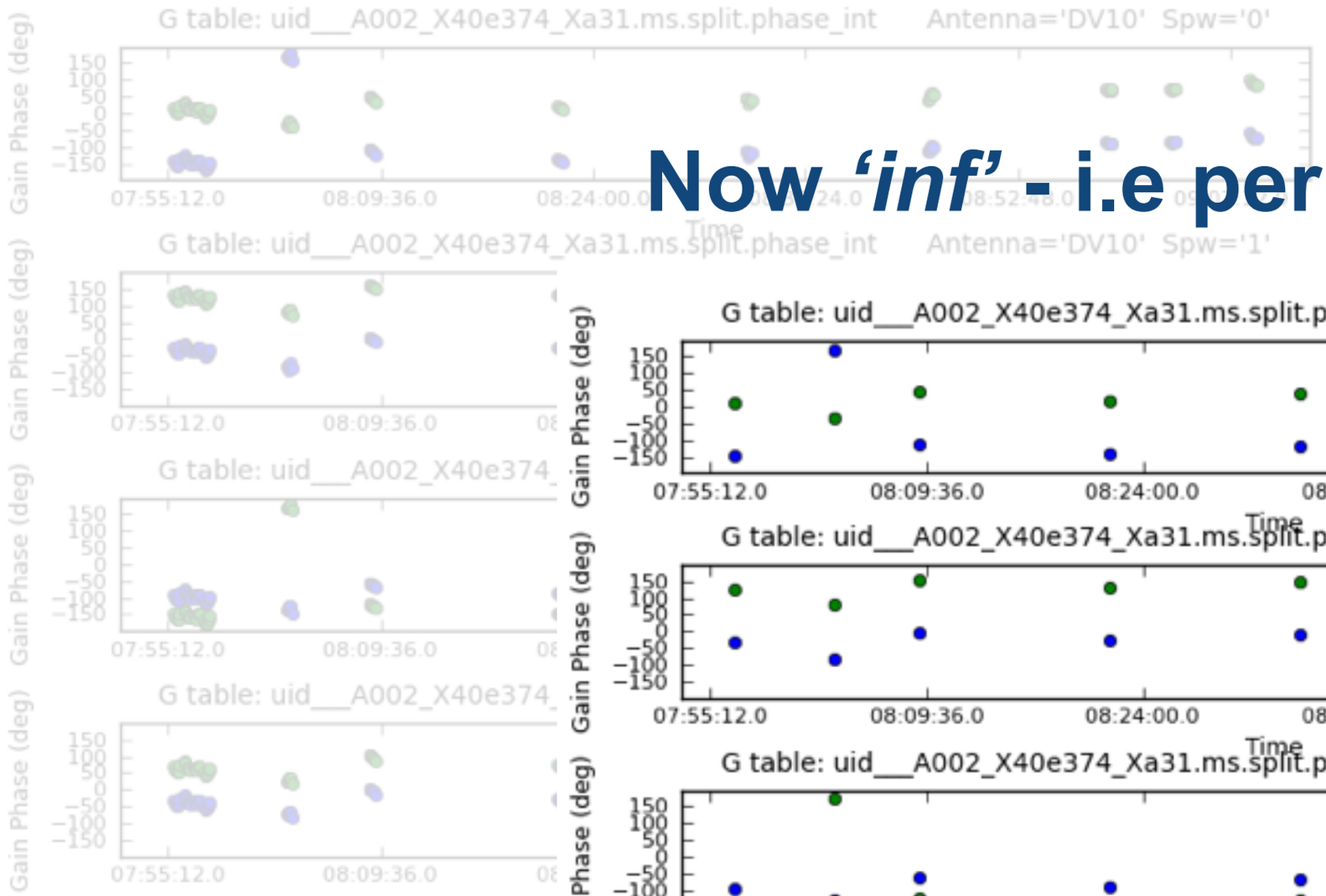
Recall - ‘int’ - integration time



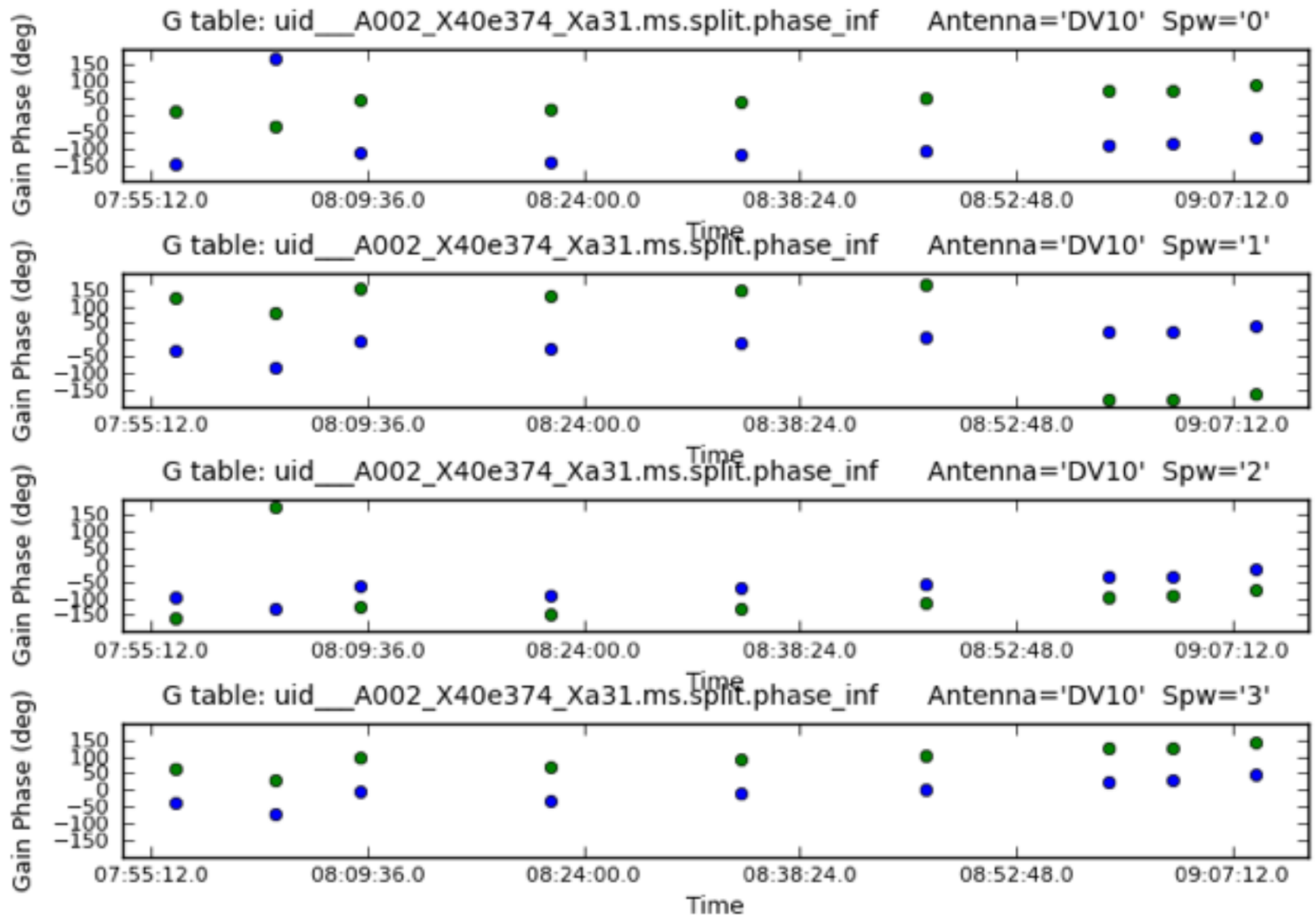
~short baseline <1 km

Calibration - Gains (3) - phase ‘inf’

Recall - ‘int’ - integration time



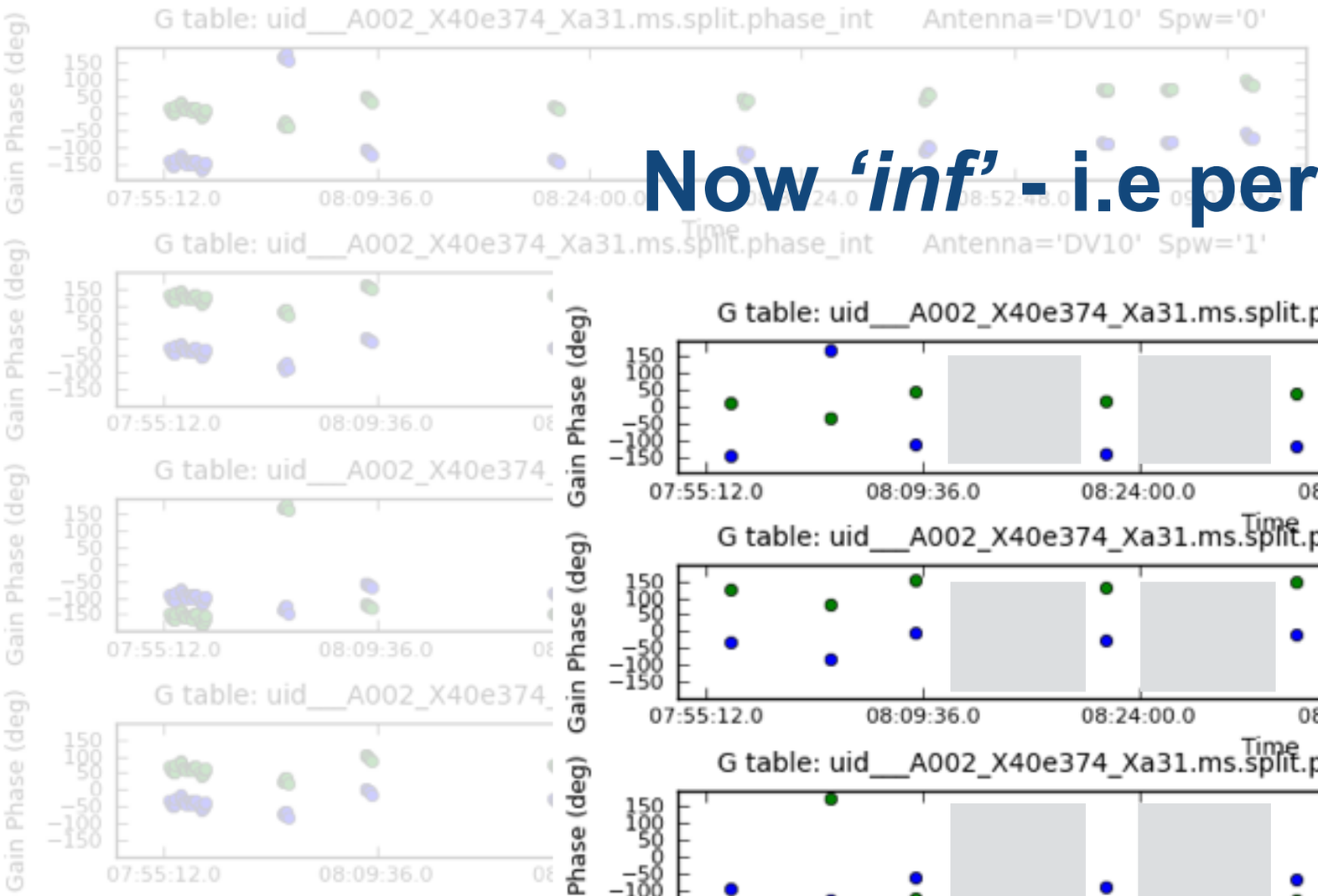
Now *‘inf’* - i.e per gain can scan



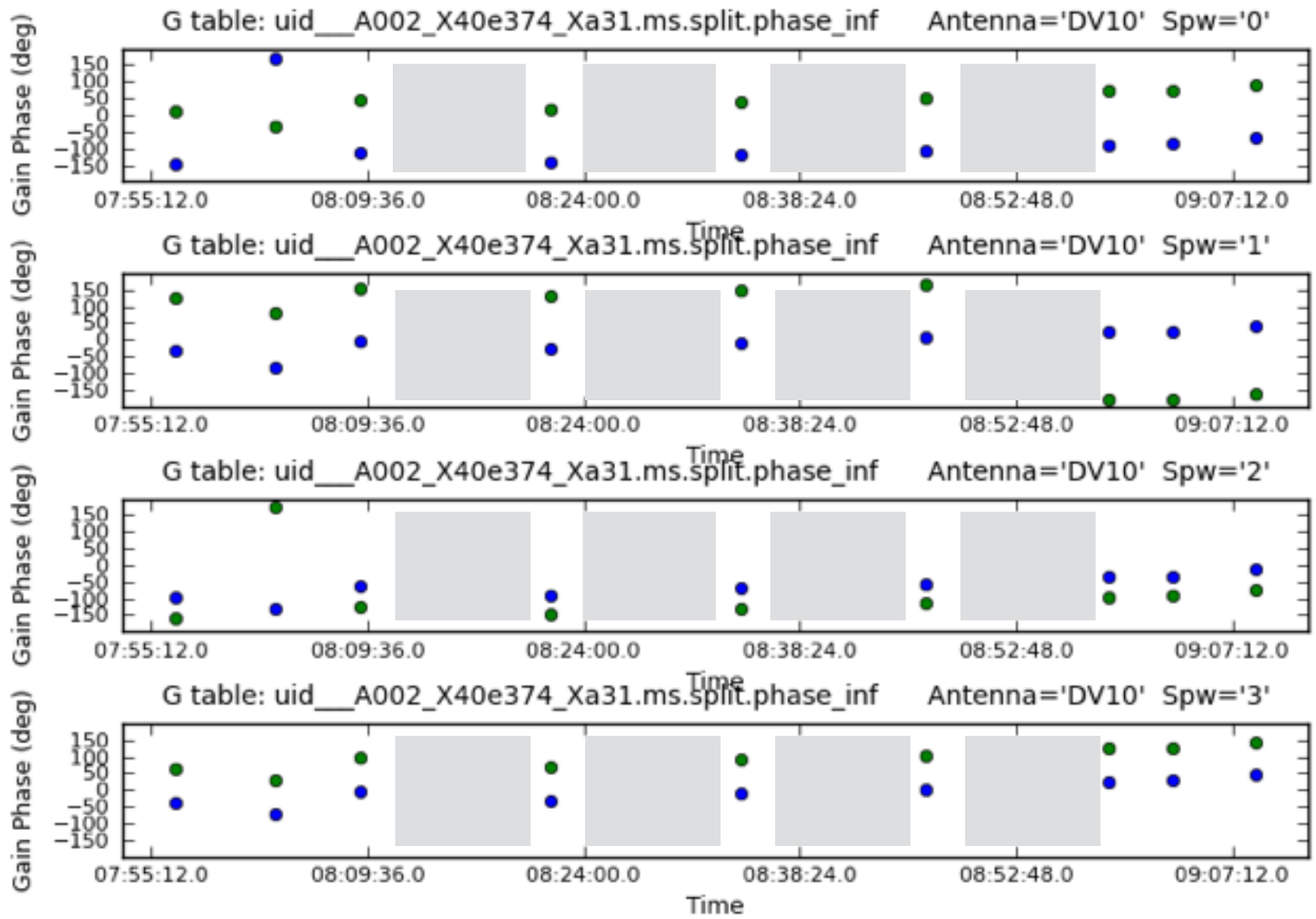
~short ba

Calibration - Gains (3) - phase ‘inf’

Recall - ‘int’



Now *‘inf’* - i.e per gain can scan



~short ba

Calibration - tables

Bandpass

QSO -freq. response

→ ‘*ap_pre.bandpass’ (applied OTF)

‘*split.bandpass’ OR
‘*split.bandpass_smooth20ch’

Flux

planet/QSO

setjy (task) →

‘*split.flux_inf’

Temporal Gains

WVR system

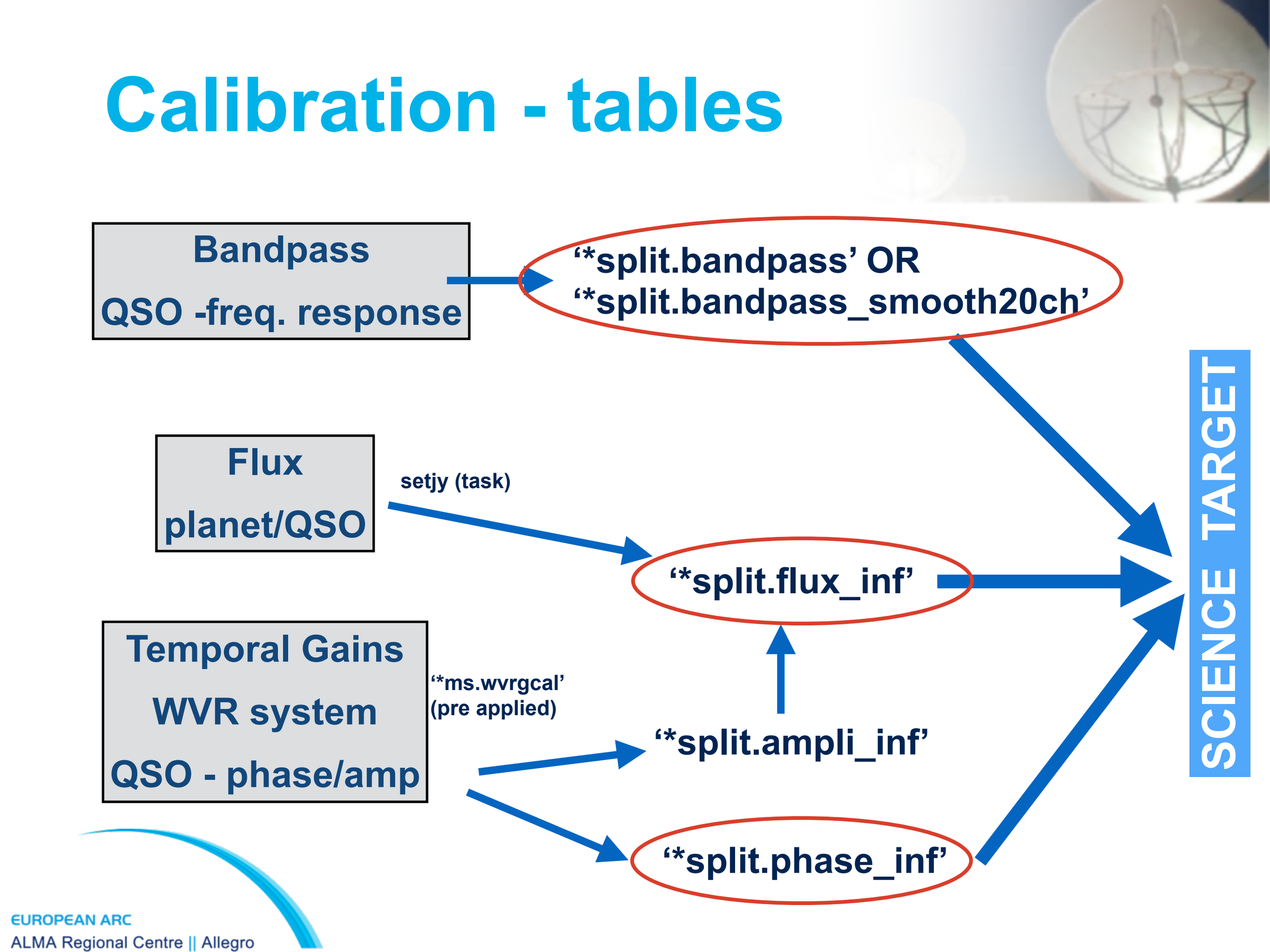
QSO - phase/amp

‘*ms.wvrgcal’
(pre applied)

‘*split.ampli_inf’
‘*split.bandpass’ (applied OTF)
‘*split.phase_int’ (applied OTF)

‘*split.phase_inf’
‘*split.bandpass’ (applied OTF)

Calibration - tables



Calibration - Application

MANUAL

```
for i in ['0', '2', '3']: # Bandpass, Ceres,J1832-2039
  applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
    field = str(i),
    gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
      'uid___A002_Xa42f75_X83e.ms.split.phase_int',
        'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
    gainfield = ['', i, i],
    interp = 'linear,linear',
    calwt = T,
    flagbackup = F)
```

```
applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
  field = '1,4',
  gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
   'uid___A002_Xa42f75_X83e.ms.split.phase_inf',
 'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
  gainfield = ['', '0', '0'], # J1733-1304
  interp = 'linear,linear',
  calwt = T,
  flagbackup = F)
```

- **applycal - applies the calibration tables**

PIPELINE

17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

Contents

- Applied calibrations
- Flagged data after calibration application
- Plots
  - Calibrated amplitude vs frequency
  - Calibrated phase vs frequency
  - Calibrated amplitude vs UV distance
  - Calibrated amplitude vs time
  - Calibrated phase vs time
  - (Corrected amplitude / model) vs antenn
  - (Corrected amplitude / model) vs UV dis
  - Science target: calibrated amplitude vs
  - Science target: calibrated phase vs freq
  - Science target: calibrated amplitude vs

Weblog

WHY? :Apply all required tables
Plots : use plotms(), QA directory / weblog

Calibration - Application (1)

MANUAL

```
for i in ['0', '2', '3']: # Bandpass, Ceres,J1832-2039
  applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
    field = str(i),
    gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
      'uid___A002_Xa42f75_X83e.ms.split.phase_int',
        'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
    gainfield = ['', i, i],
    interp = 'linear,linear',
    calwt = T,
    flagbackup = F)
```

- **applycal - applies the calibration tables**
  - **gaintable - order to apply, BP, phase, flux**
  - **gainfield - which field to use in the gaintable**
  - **interp - the interpolation mode to use**

Calibration - Application (1)

MANUAL

```
for i in ['0', '2', '3']: # Bandpass, Ceres,J1832-2039
  applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
    field = str(i),
    gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
      'uid___A002_Xa42f75_X83e.ms.split.phase_int',
        'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
    gainfield = ['', i, i],
    interp = 'linear,linear',
    calwt = T,
    flagbackup = F)
```

field - these are the bandpass, flux cal, check source

ORDER !! - which field is used from which table

- **calwt** - true for correct weight for CASA >4.3.1, for correct ACA + 12m merging

- **applycal** - applies the calibration tables
  - **gaintable** - order to apply, BP, phase, flux
  - **gainfield** - which field to use in the gaintable
  - **interp** - the interpolation mode to use

For all calibrators EXCEPT phase cal, the 'int' phase solution is applied from itself

Calibration - Application (2)

MANUAL

- **applycal - apply the calibration tables - *to the science target***

```
applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
  field = '1,4',
  gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
   'uid___A002_Xa42f75_X83e.ms.split.phase_inf',
 'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
  gainfield = ['', '1', '1'], # J1733-1304
  interp = 'linear,linear',
  calwt = T,
  flagbackup = F)
```

Calibration - Application (2)

MANUAL

- **applycal - apply the calibration tables - *to the science target***

```
applycal(vis = 'uid___A002_Xa42f75_X83e.ms.split',
  field = '1,4',
  gaintable = ['uid___A002_Xa42f75_X83e.ms.split.bandpass_smooth20ch',
   'uid___A002_Xa42f75_X83e.ms.split.phase_inf',
 'uid___A002_Xa42f75_X83e.ms.split.flux_inf'],
  gainfield = ['', '1', '1'], # J1733-1304
  interp = 'linear,linear',
  calwt = T,
  flagbackup = F)
```

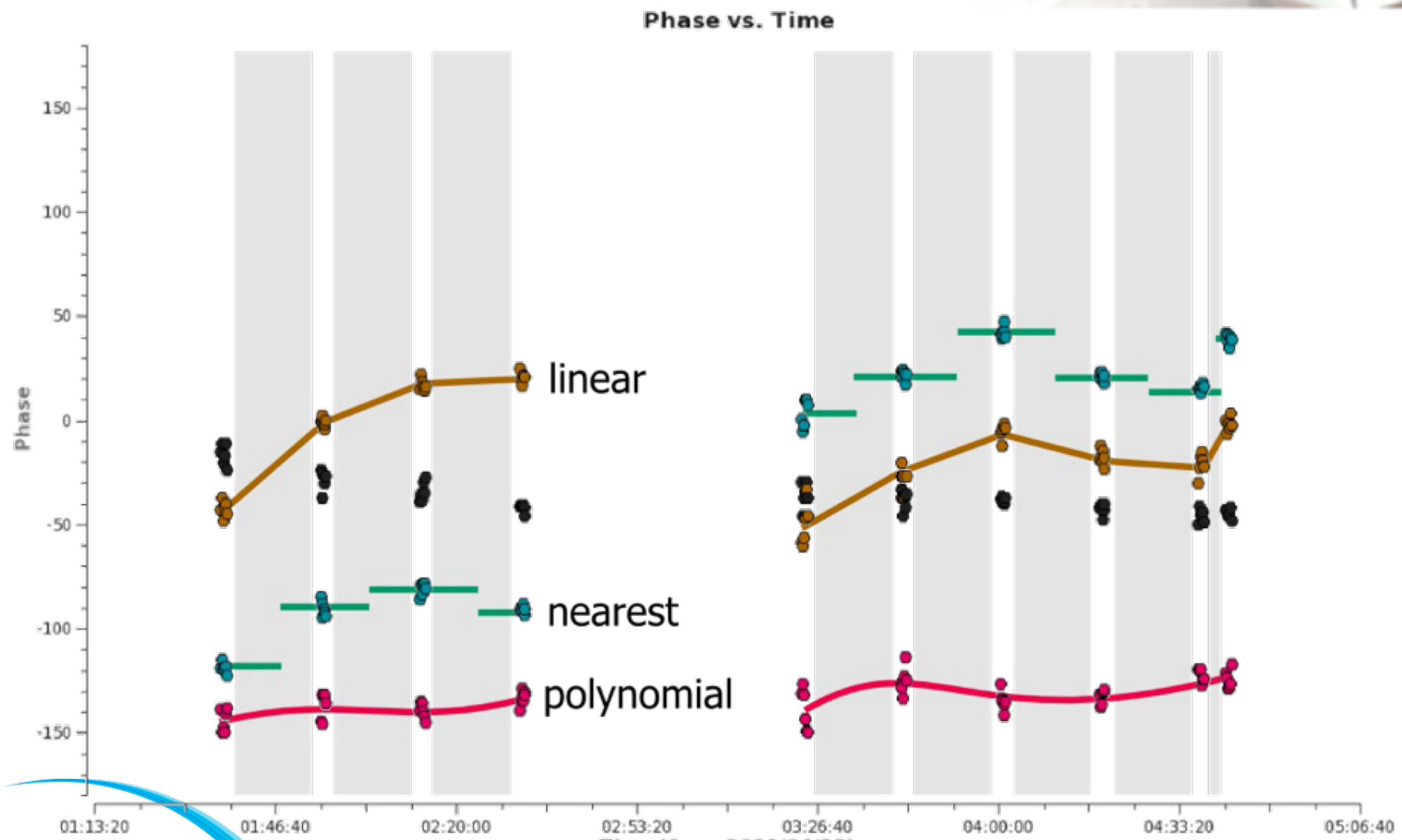
field - gain cal & science target

inf phase solution - can only interpolate phases

gainfield - solutions for field '1' are applied to BOTH '1' and '4'

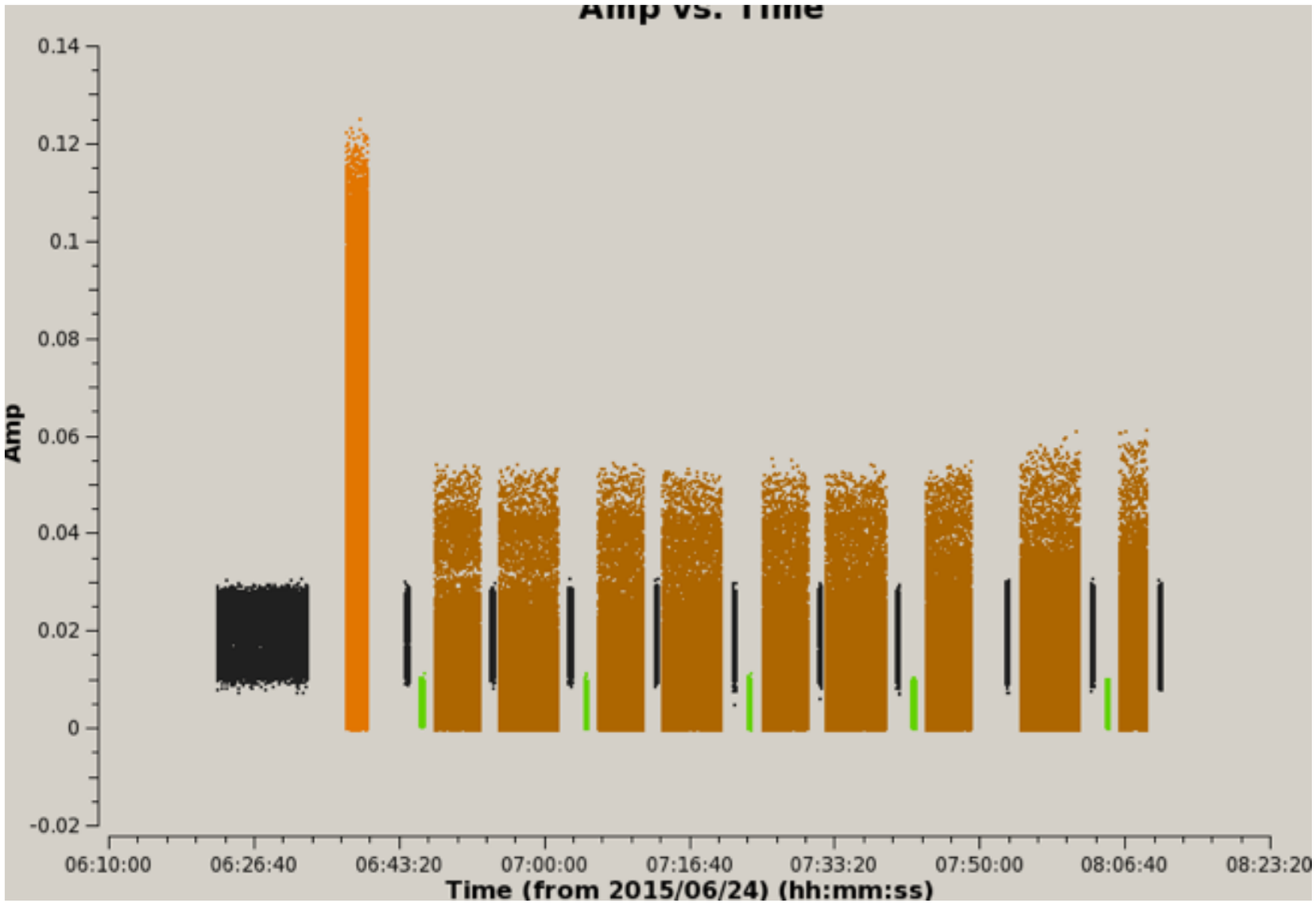
The gaincal and the science target have the solutions from the gaincal applied to them

Calibration - Application (2)



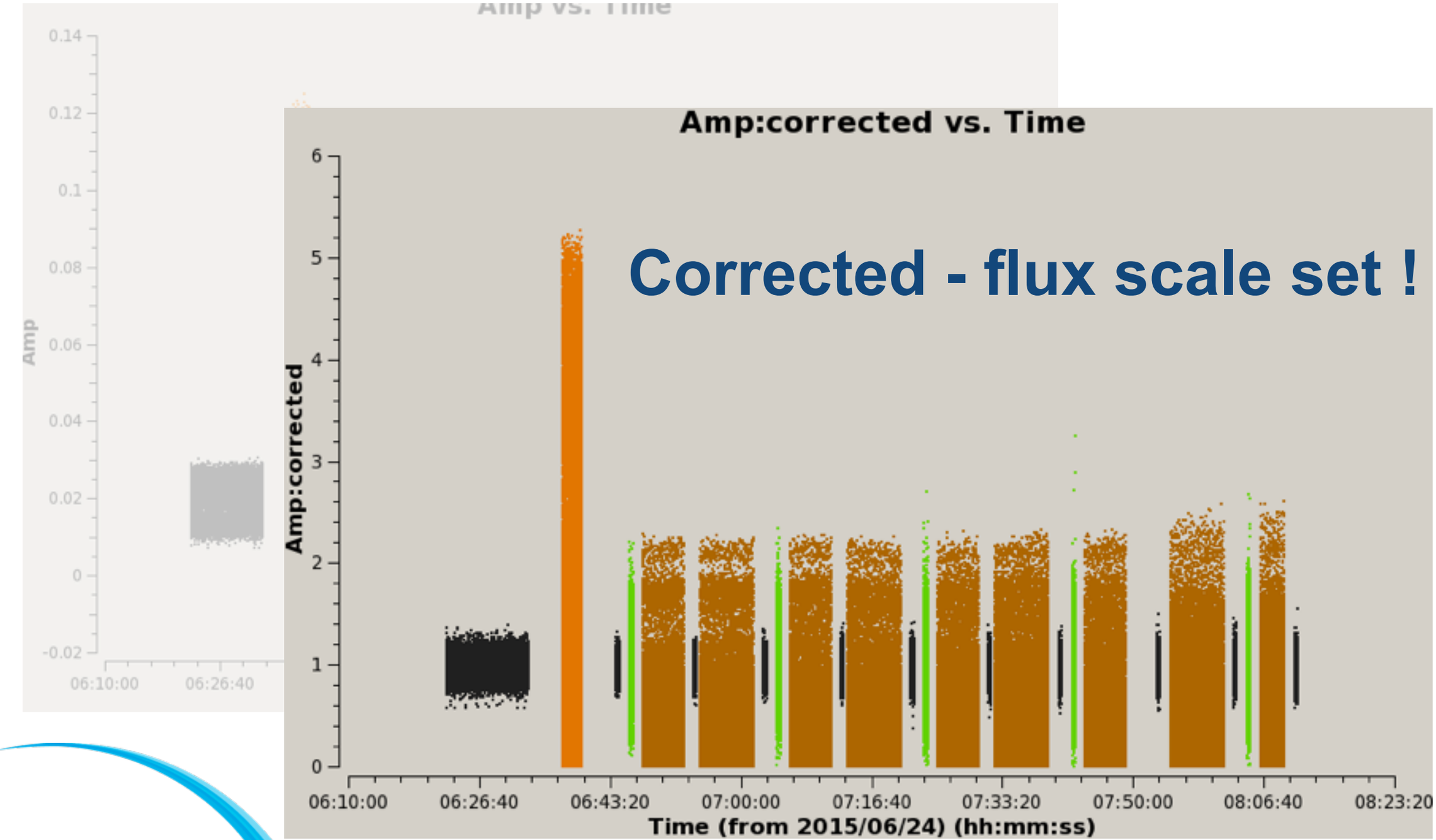
Calibration - The data (amp)

Recall - ‘raw’ data



Calibration - The data (amp)

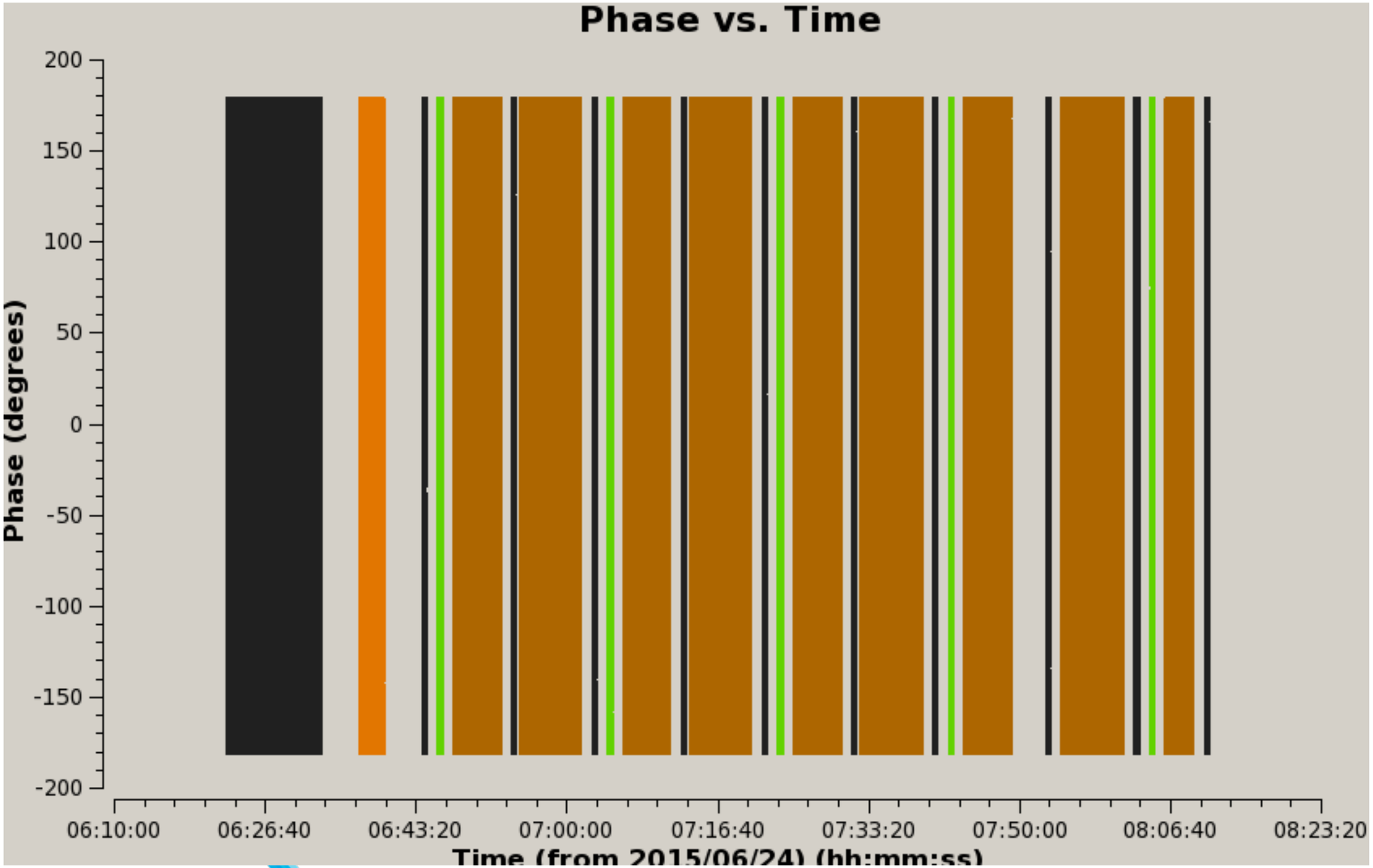
Recall - ‘raw’ data



Corrected - flux scale set !

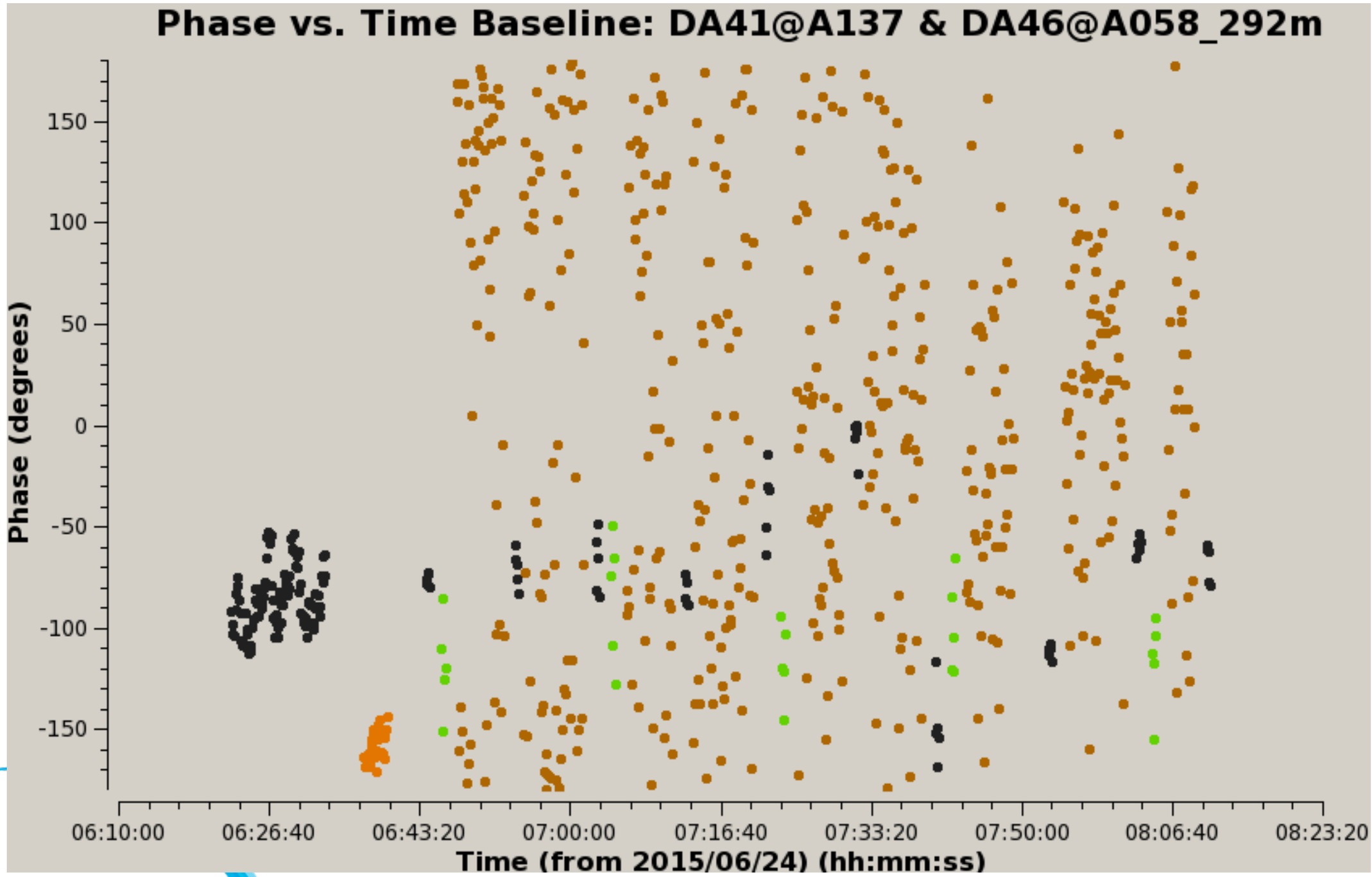
Calibration - The data (phase)

Recall - ‘raw’ data - all baselines - phases ‘everywhere’



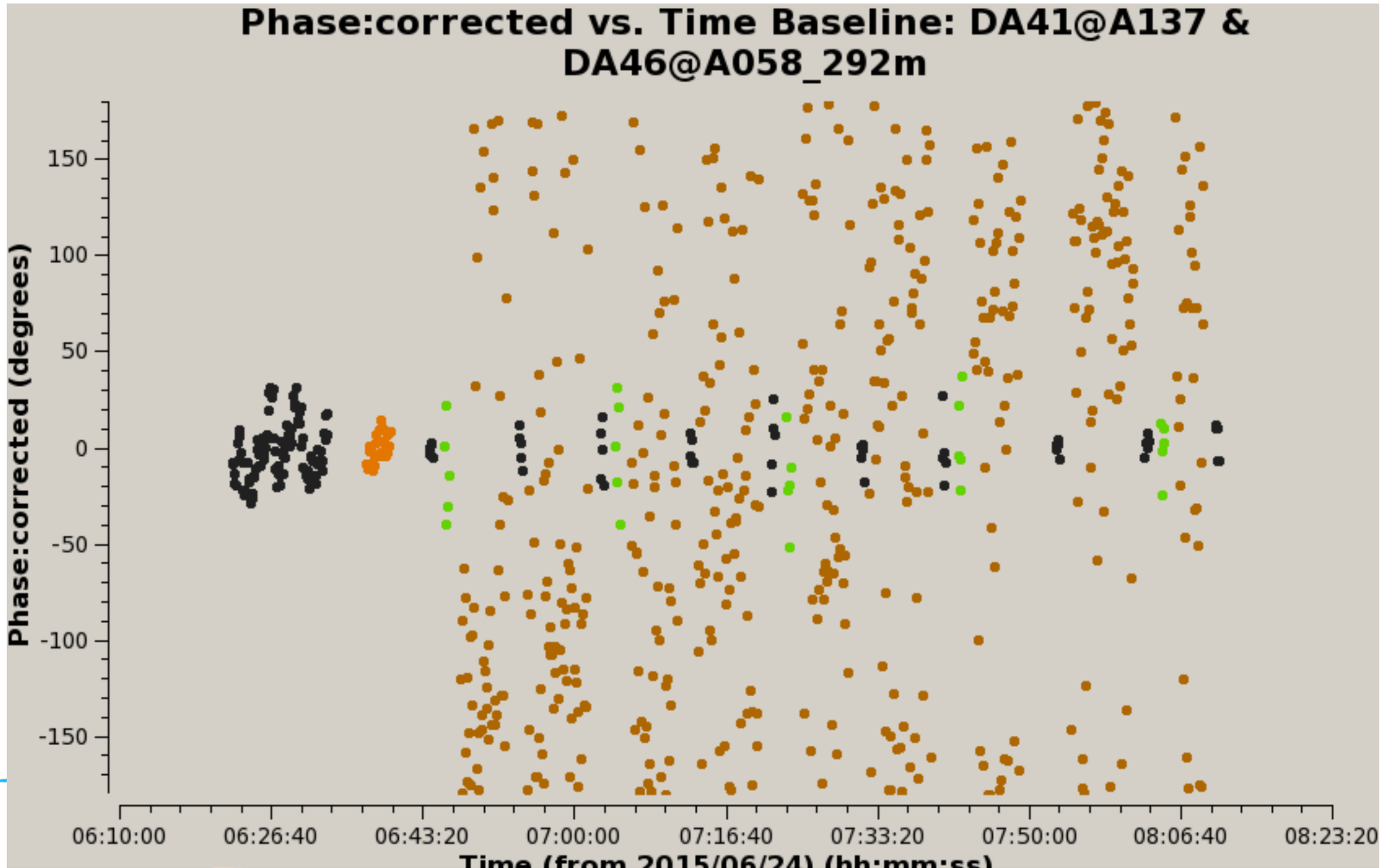
Calibration - The data (phase)

Recall - ‘raw’ data - single baseline - non-zero calibrator phases



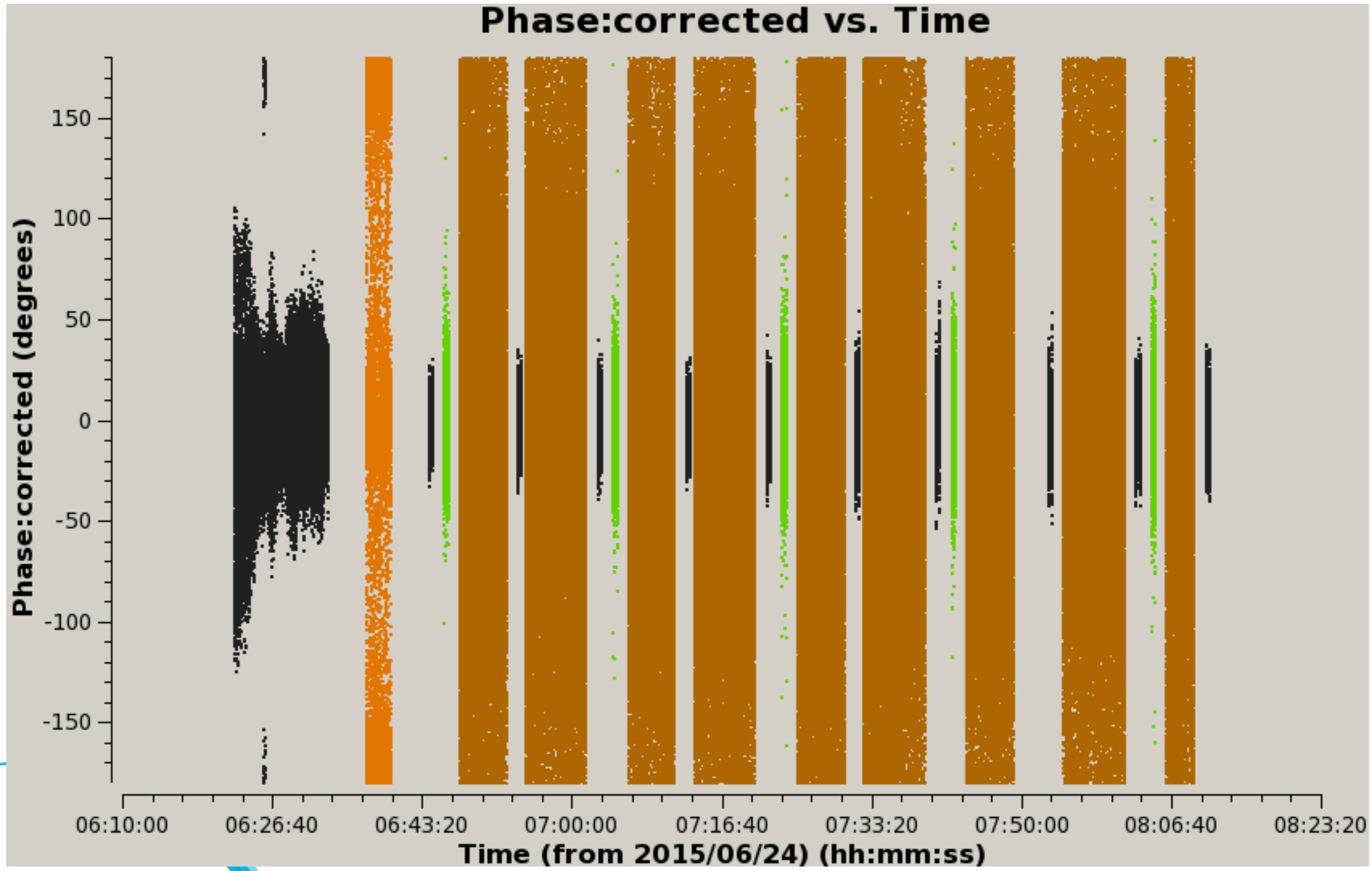
Calibration - The data (phase)

Recall - corrected data - single baseline - calibrators ~ zero phase



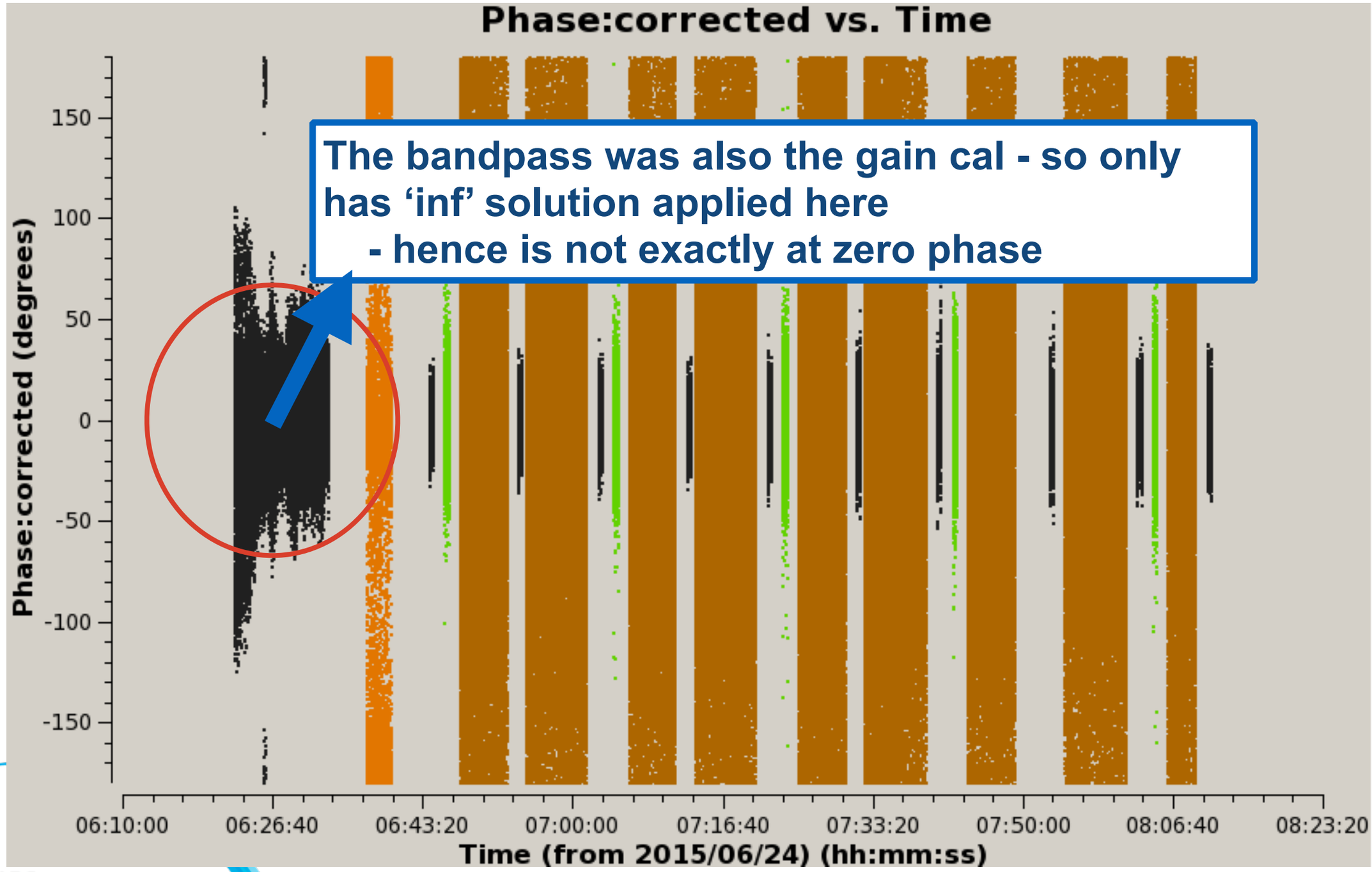
Calibration - The data (phase)

Recall - corrected data - ALL baseline - calibrators ~ zero phase



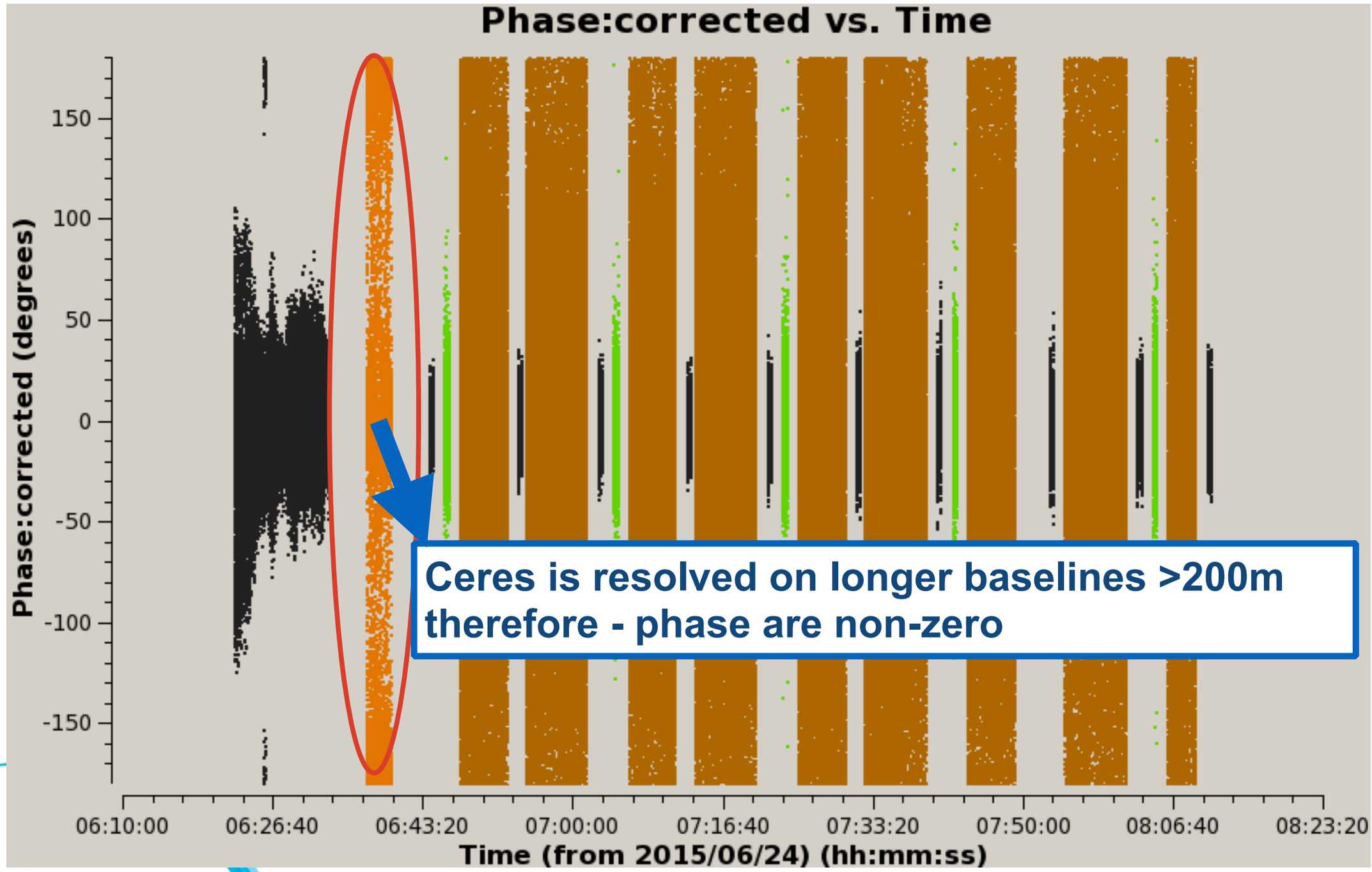
Calibration - The data (phase)

Recall - corrected data - ALL baseline - calibrators ~ zero phase



Calibration - The data (phase)

Recall - corrected data - ALL baseline - calibrators ~ zero phase



Calibration - caveats

- Solar system object as flux cal - more steps
- Narrow and wideband mixed (e.g. 2GHz and <250MHz) - SPW mapping
- Low SNR issues on gain cals - SPW combination
- Lots of narrow SPW - Bandwidth Switching

Calibration - COMPLETE!!

Now ready for imaging

EUROPEAN ARC
ALMA Regional Centre || Allegro

Calibration - Gains (1) SSO

MANUAL

```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.phase_short_int',
  field = '2', # Ceres
  selectdata = T,
  antenna = 'DA46,DA49,DA53,DA55,DA59,DA63,DV04,DV08,DV18,DV19,PM02,PM04&',
  solint = 'int',
  refant = 'DA59',
  gaintype = 'G',
  calmode = 'p',
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass')
```

- **antenna - select the antennas close OR**
- **uvrange - select a short uvrange (e.g. <200m)**

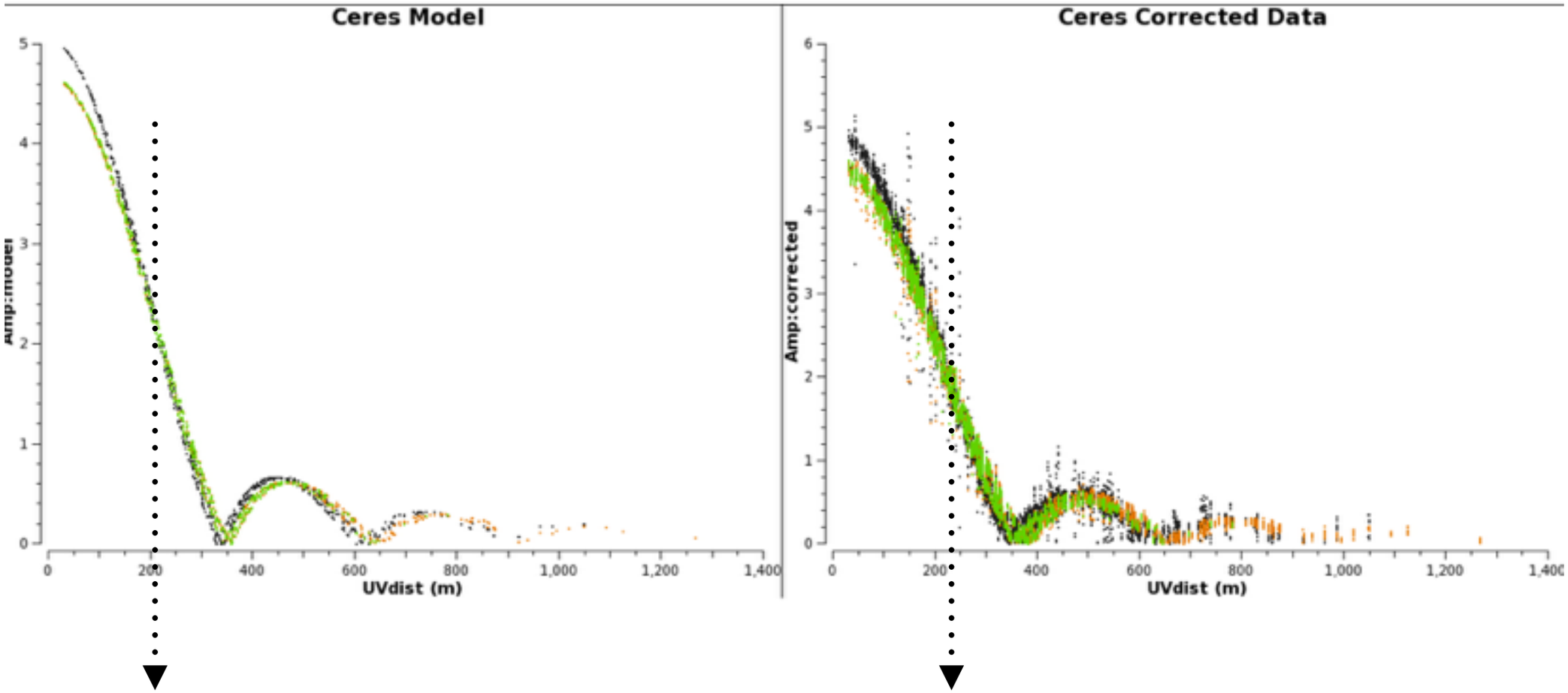
```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.phase_short_int',
  field = '0,3', # J1733-1304,J1832-2039
  selectdata = T,
  solint = 'int',
  refant = 'DA59',
  gaintype = 'G',
  calmode = 'p',
  append = T,
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass')
```

PIPELINE

14. hifa_spwphaseup
15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

WHY? :Correct for all phase variations in time - short baselines - SSO is unresolved - 2 STEPS!!
Plots : in ‘QA’ directory / weblog ‘*split.phase_short_int.plots’

Calibration - Gains (1) SSO



ONLY baselines <200m used to bootstrap flux value to other calibrators - i.e. assume a point source such that a phase-up can be accomplished (i.e. phase_short)
Plots: QA directory

Calibration - Gains (2) SSO

MANUAL

```
os.system('rm -rf uid___A002_Xa44acb_Xadb.ms.split.ampli_short_inf')
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ampli_short_inf',
  field = '0,2,3', # J1733-1304,Ceres,J1832-2039
  selectdata = T,
  solint = 'inf',
  refant = 'DA59',
  gaintype = 'T',
  calmode = 'a',
  gaintable = ['uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch',
'uid___A002_Xa44acb_Xadb.ms.split.phase_short_int'])
```

- **gaincal - creates the antenna based amp solutions**
  - **field - ALL calibrator sources**
  - **solint - 'inf' i.e. per scan/visit to the gain cal**
  - **gaintable - apply bandpass AND the 'int' SHORT phase-up solutions OTF**

PIPELINE

15. hifa_gfluxscale
16. hifa_timegaincal
17. hif_applycal
18. hif_makeimlist
19. hif_makeimages

WHY? :Correct for slow amplitude variations in time
Plots : In some cases in 'QA' directory / weblog '*split.ampli_short_int.plots'

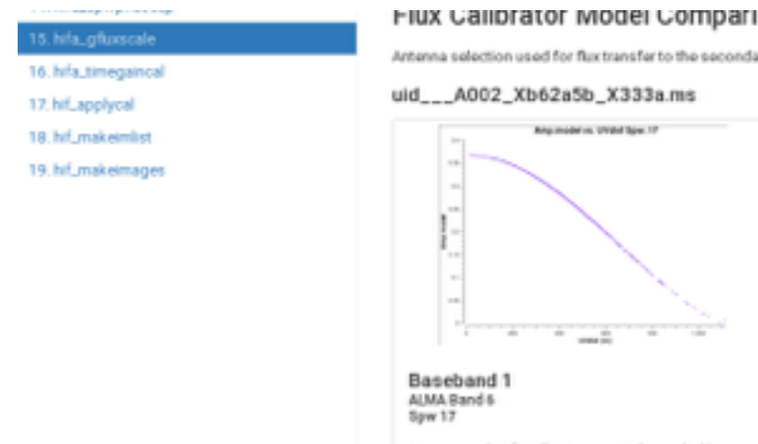
Calibration - Flux (2) SSO

MANUAL

```
fluxscaleDict = fluxscale(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ampli_short_inf',
  fluxtable = 'uid___A002_Xa44acb_Xadb.ms.split.flux_short_inf',
  reference = '2') # Ceres
```

```
f = open('uid___A002_Xa44acb_Xadb.ms.split.fluxscale')
fc = f.readlines()
f.close()

  for phaseCalName in ['J1733-1304']:
    for i in range(len(fc)):
      if fc[i].find('Flux density for '+phaseCalName) != -1 and re.search('in Sp
*?\))? is: [0-9]+\.[0-9]+', fc[i], re.DOTALL|re.IGNORECASE) != None:
        line = (re.search('in SpW=[0-9]+(?: \(.*?\))? is: [0-9]+\.[0-9]+', fc[i],
IGNORECASE)).group(0)
        spwId = (line.split('='))[1].split()[0]
        flux = float((line.split(':'))[1].split()[0])
        setjy(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
          field = phaseCalName.replace(';','*;').split(';')[0],
          spw = spwId,
          standard = 'manual',
          fluxdensity = [flux,0,0,0])
```

PIPELINE



WHY? :Setting the correct gains for flux scaling - then recalibrate all baselines with bootstrapped flux
Plots : in ‘QA’ directory / weblog - ‘*split.flux_inf.plots’

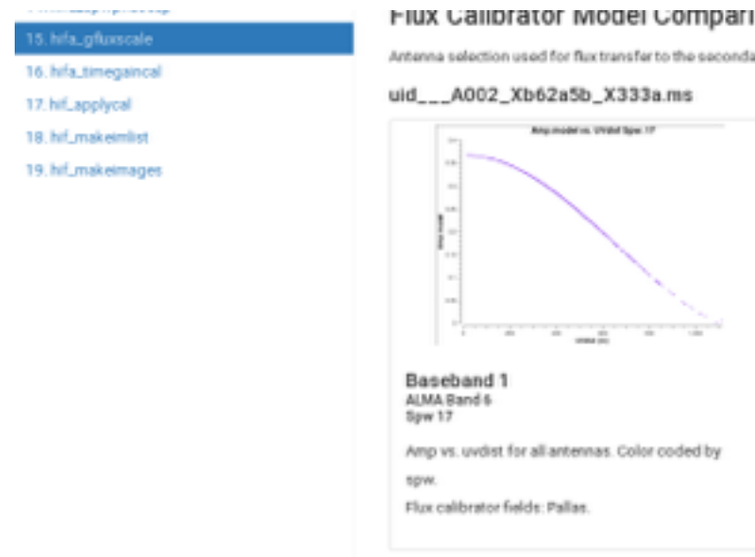
Calibration - Flux (2) SSO

MANUAL

```
fluxscaleDict = fluxscale(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ampli_short_inf',
  fluxtable = 'uid___A002_Xa44acb_Xadb.ms.split.flux_short_inf',
  reference = '2') # Ceres
```

```
f = open('uid___A002_Xa44acb_Xadb.ms.split.fluxscale')
fc = f.readlines()
f.close()

  for phaseCalName in ['J1733-1304']:
    for i in range(len(fc)):
      if fc[i].find('Flux density for '+phaseCalName) != -1 and re.search('in Sp\
*?\))? is: [0-9]+\.[0-9]+', fc[i], re.DOTALL|re.IGNORECASE) != None:
        line = (re.search('in SpW=[0-9]+(?: \(.*?\))? is: [0-9]+\.[0-9]+', fc[i],
IGNORECASE)).group(0)
        spwId = (line.split('='))[1].split()[0]
        flux = float((line.split(':'))[1].split()[0])
        setjy(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
          field = phaseCalName.replace(';','*;').split(';')[0],
          spw = spwId,
          standard = 'manual',
          fluxdensity = [flux,0,0,0])
```

PIPELINE



- **loop - searches the *fluxscale file for the flux of another calibrator - gain cal**

WHY? :Setting the correct gains for flux scaling - then recalibrate all baselines with bootstrapped flux
Plots : in 'QA' directory / weblog - '*split.flux_inf.plots'

Calibration - Flux (2) SSO

MANUAL

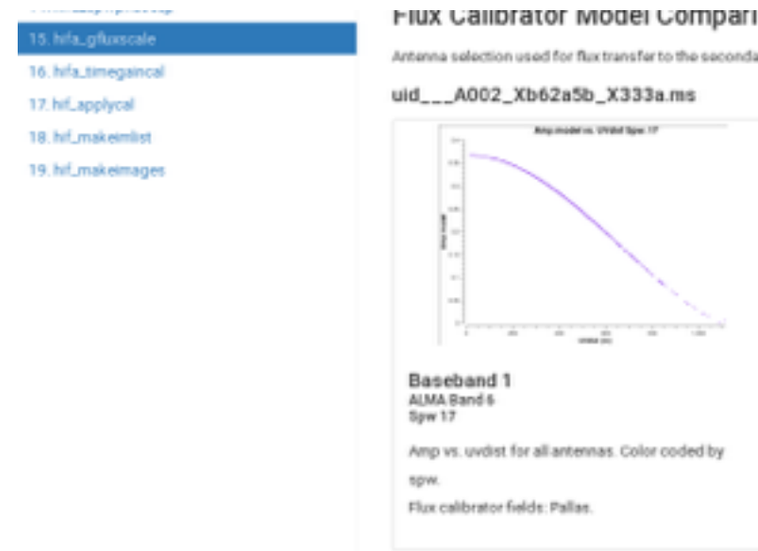
```
fluxscaleDict = fluxscale(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.ampli_short_inf',
  fluxtable = 'uid___A002_Xa44acb_Xadb.ms.split.flux_short_inf',
  reference = '2') # Ceres
```

```
f = open('uid___A002_Xa44acb_Xadb.ms.split.fluxscale')
fc = f.readlines()
f.close()
```

```
  for phaseCalName in ['J1733-1304']:
    for i in range(len(fc)):
      if fc[i].find('Flux density for '+phaseCalName) != -1 and re.search('in Sp\
*?\))? is: [0-9]+\.[0-9]+', fc[i], re.DOTALL|re.IGNORECASE) != None:
        line = (re.search('in SpW=[0-9]+(?: \(.*?\))? is: [0-9]+\.[0-9]+', fc[i],
IGNORECASE)).group(0)
        spwId = (line.split('='))[1].split()[0]
        flux = float((line.split(':'))[1].split()[0])
        setjy(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
          field = phaseCalName.replace(';','*;').split(';')[0],
          spw = spwId,
          standard = 'manual',
          fluxdensity = [flux,0,0,0])
```

- **setjy - called again to set gain cal bootstrapped flux**

PIPELINE



- **loop - searches the *fluxscale file for the flux of another calibrator - gain cal**

WHY? :Setting the correct gains for flux scaling - then recalibrate all baselines with bootstrapped flux
Plots : in ‘QA’ directory / weblog - ‘*split.flux_inf.plots’

Calibration - Redo-Gains/Flux - SSO

MANUAL

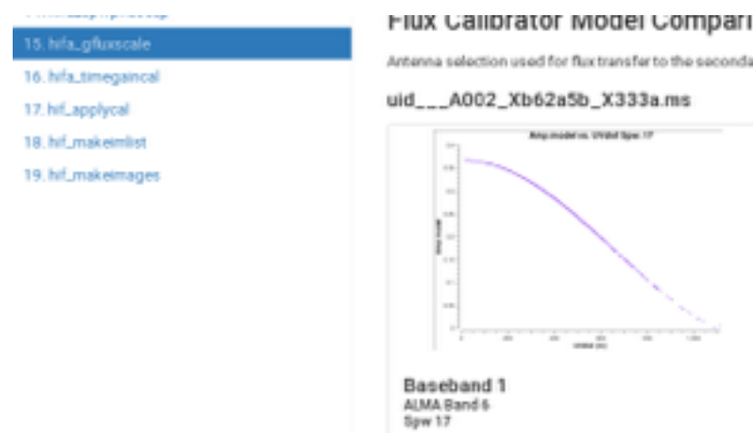
```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.phase_int',
  field = '0,2,3', # J1733-1304,Ceres,J1832-2039
  solint = 'int',
  refant = 'DA59',
  gaintype = 'G',
  calmode = 'p',
  gaintable = 'uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch')
```

- **gaincal - creates the antenna based phase solutions - ALL baselines**

```
gaincal(vis = 'uid___A002_Xa44acb_Xadb.ms.split',
  caltable = 'uid___A002_Xa44acb_Xadb.ms.split.flux_inf',
  field = '0,2,3', # J1733-1304,Ceres,J1832-2039
  solint = 'inf',
  refant = 'DA59',
  gaintype = 'T',
  calmode = 'a',
  gaintable = ['uid___A002_Xa44acb_Xadb.ms.split.bandpass_smooth20ch',
'uid___A002_Xa44acb_Xadb.ms.split.phase_int'])
```

- **gaincal - creates the antenna based amp solutions - ALL baselines**

PIPELINE



WHY? : Resolve phase-up, then re-solve amplitudes for ALL baselines after flux bootstrap from resolved SSO