

Methods of VGOS post-processing

Dan Hoak (for the Haystack correlation team)

TOW Correlator Workshop May 4-5 2023



VGOS post-processing: Haystack environment

Experience is based on VR / VO sessions from the past 2 years

Basic outline:

- Mike Titus correlates and generates mk4 files
- John/Dan run postproc scripts and generate control file, script output, diagnostic plots
- Mike runs the final pseudo-Stokes lxy batch_fourfit job and builds the vgosDB
- John/Dan run proxy cc scripts to build PCMT files, add to v4 vgosDB
- Dhiman/Arthur check the vgosDB with nuSolve
- Mike sends the report and vgosDB to Sergei, who makes final checks
- If everything works, submit report

For the post-processing, only about 1-in-5 experiments are “turn-the-crank”

- This is after considerable (sometimes heroic) work by Mike to solve problems at the correlation stage!
- VGOS correlation & processing is an ongoing R&D effort!

VGOS post-processing: Haystack environment

We typically run the post-processing scripts on a 12-core machine running Ubuntu 22.04.

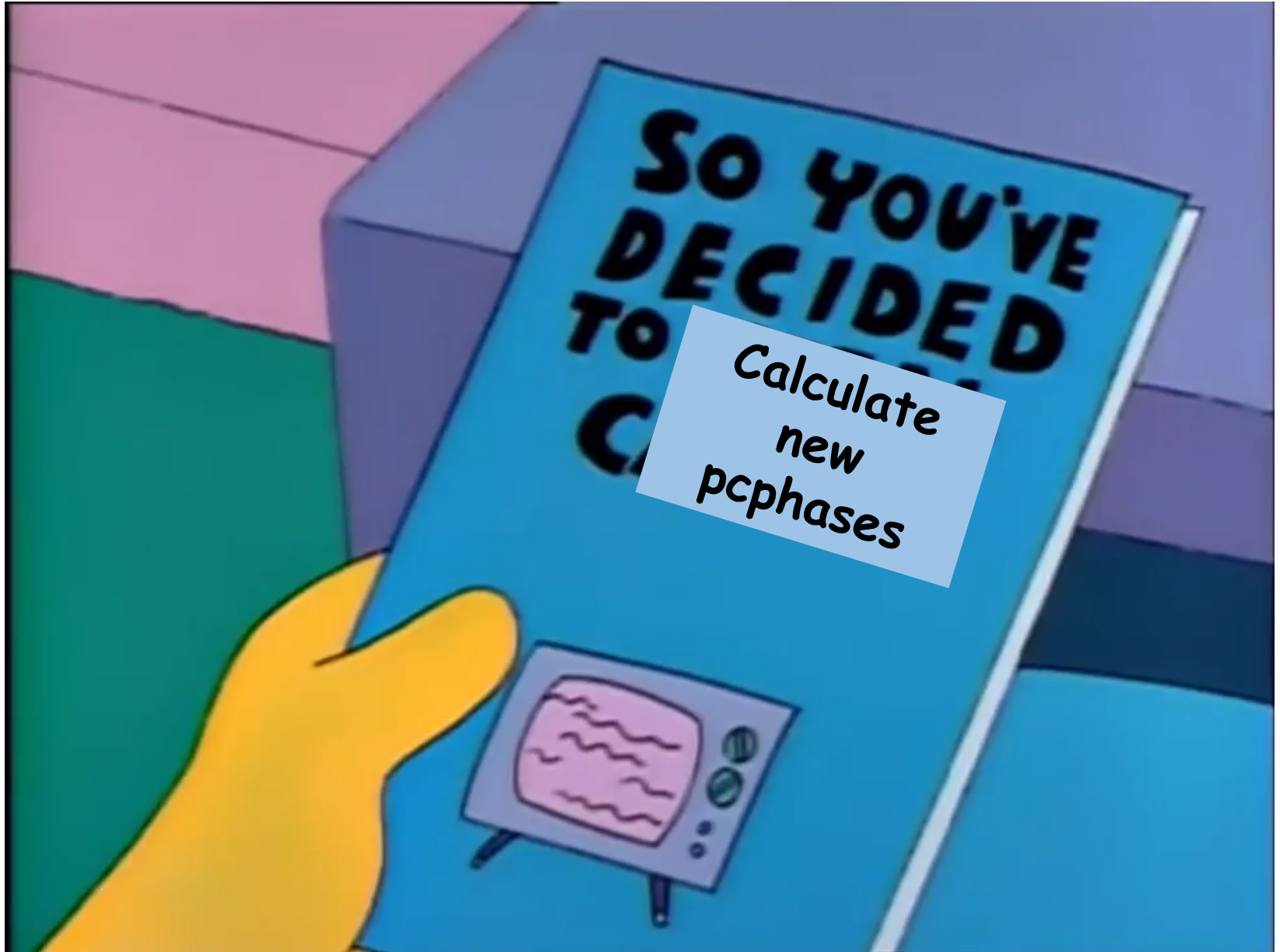
- python 3.10 (backward compatible to 2.7.5)
- numpy 1.21 (backward compatible to 1.7.1)
- matplotlib 3.5.1 (backward compatible to 1.2.0)

For a 24-hr VGOS session with 8-11 stations and 1872 scans:

- **ffres2pcp** can take 6-8 hours
- **fourphase** can take 12-18 hours
- **batch_fourfit** can take 6 hours

Can finish the post-processing in ~3-4 days.

Running **vgoscf_generate** seldom works when the network includes Australia; hard to find a reference station for **ffres2pcp** that has good coverage with Europe, North America and Australia. I almost always run the two scripts separately.



First I edit the initial control file:

- Remove any ignored channels
- Remove old y-x delay/offsets
- Remove pcphases for stations not in the experiment
- Edit ionospheric search as needed:
`ion_npts 75`
`ion_win -100.0 100.0`

(this is good enough for
ffres2pcp and fourphase)

And then give it a try:

```
~/geodesy/3825$ ffres2pcp.py cf_3825_MEHLNSV_dh1 M EHLNSV .  
-n 16 -s 12 -a 30 -d 5 -q 5 -p
```

First I edit the initial control file:

- Remove any ignored channels
- Remove old y-x delay/offsets
- Remove pcphases for stations not in the experiment
- Edit ionospheric search as needed:

```
ion_npts 75
ion_win -100.0 100.0
```

(this is good enough for
ffres2pcp and fourphase)

And then give it a try:

```
~/geodesy/3825$ ffres2pcp.py cf_3825 MEHLNSV_dh1 M EHLNSV .
-n 16 -s 12 -a 30 -d 5 -q 5 -p
```

The SNR for cross-hands on short baselines (GE) can be low, and tag-along stations might have very few scans available; don't waste them!

The difference in dTEC among pol-prods for long baselines can be large

First I edit the initial control file:

- Remove any ignored channels
- Remove old y-x delay/offsets
- Remove pcphases for stations not in the experiment
- Edit ionospheric search as needed:
`ion_npts 75`
`ion_win -100.0 100.0`
(this is good enough for
ffres2pcp and fourphase)

And then give it a try:

```
~/geodesy/3825$ ffres2pcp.py cf_3825_MEHLNSV_dh1 M EHLNSV .  
-n 16 -s 12 -a 30 -d 5 -q 5 -p
```

This will generate a sub-directory:

```
~/geodesy/3825/scratch/YYYYMMDD-HHMMSS/3825
```

...make sym-links to everything in the top experiment directory, and start generating type-2 fringe files. I often rename the YYYYMMDD folder to 'ffres1'.

ffres2pcp will do a bunch of things:

- Figures out which baselines are available from the reference station / remote stations you selected
 - It does this by scanning for type-1 mk4 files (correl files), and only checks the two characters in the filenames, so it's fast
- Looks for root files and assembles a list of scans
- Searches for pre-existing fringe files with the right baselines, pol-products, and control file
 - This is slow, because it has to check the control file hash in the fringe file and check the pol-product
 - Can take a long time if there are many fringe files to open!
- Once it has a list of scans with correl files for the selected baselines, it runs **fourfit** to fringe XX, XY, YX, YY

Finding baselines

```
Baselines:  ['MV', 'MN', 'ME', 'LM', 'MS', 'HM']
```

Finding root files

```
Found 3182 root files.
```

```
4639 fringe files after baseline selection
```

```
Found 4639 previously generated fringe files
```

```
0 fringe files remaining after control file hash filter
```

```
load_and_batch_fourfit: will run a total of 18556 fourfit processes
```



VGOS post-processing: **ffres2pcp**

Ok, go work on something else while it runs...

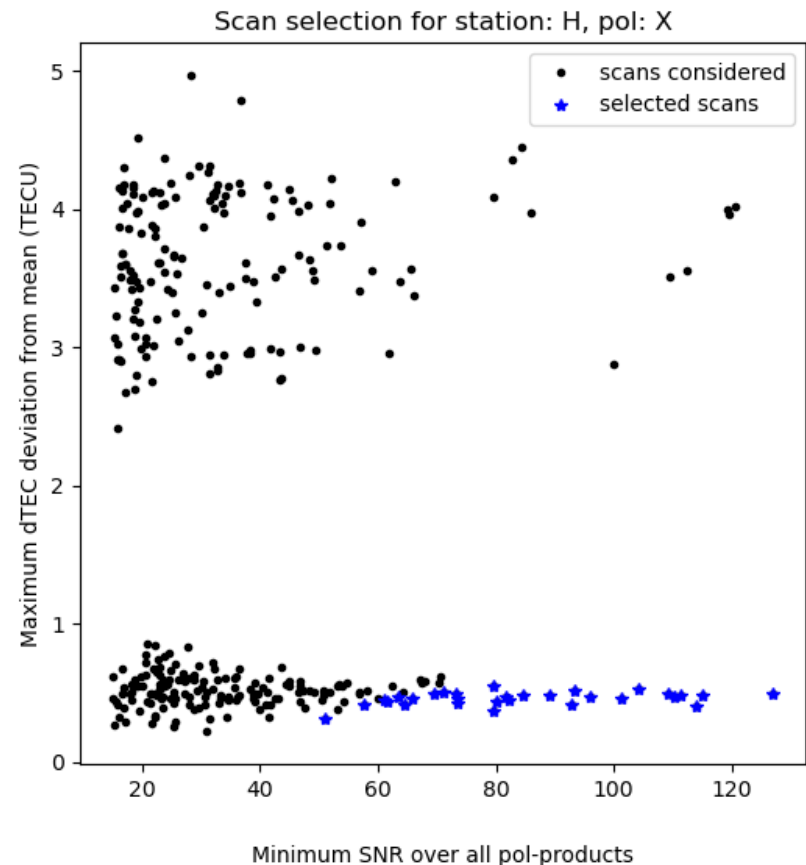
Once the fourfit jobs are finished, **ffres2pcp** will open the fringe files, and select good scans for each scan/baseline based on minimum SNR among the pol-prods, the qcodes, and the difference in dTEC among the pol-prods.

When it's finished, use **summarize_report.py** to parse the output `ffres2pcp-report.json` file and generate diagnostic plots.

Check the scan selections and the changes to the pcphases!

This is what success looks like 

Note that `ffres2pcp` will use a Pareto front calculation to only choose the scans with the best SNR and dTEC deviation. So you can relax the dTEC constraint if you want to make sure you get scans. You can always rerun with tighter constraints!



VGOS post-processing: **ffres2pcp**

If you need to kill the job, you can restart it without losing your previous work!

- ffres2pcp will check for previously generated fringe files that match the baselines and control file
- But, remember to use the pre-existing scratch directory!

```
~/geodesy/3825/scratch/YYYYMMDD-HHMMSS/3825$ ffres2pcp.py  
cf_3825_MEHLNSV_dh1 M EHLNSV . -n 16 -s 12 -a 30 -d 5 -q 5 -p -w
```

Use the `-w` flag to run ffres2pcp without creating a scratch directory (it will refuse to run in a directory with 'scratch' or 'prepass' in the path)

If you need to kill the job, you can restart it without losing your previous work!

- `ffres2pcp` will check for previously generated fringe files that match the baselines and control file
- But, remember to use the pre-existing scratch directory!

```
~/geodesy/3825/scratch/YYYYMMDD-HHMMSS/3825$ ffres2pcp.py  
cf_3825_MEHLNSV_dh1 M EHLNSV . -n 16 -s 12 -a 30 -d 5 -q 5 -p -w
```

This will pick up the fringe-fitting jobs where you left off.

You can also rerun with different SNR, qcode, or ddTEC limits, if you need to relax them.

If `ffres2pcp` fails to find any acceptable fringe files for a particular baseline:

- Check that fringe files were produced; for example, if `ffres2pcp` failed to generate `pcphases` for Hobart:

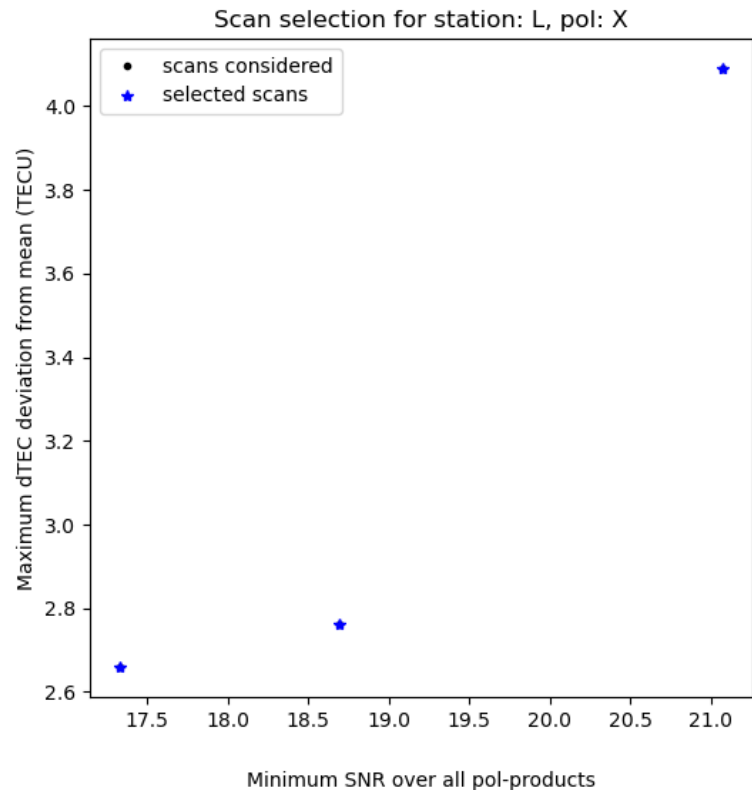
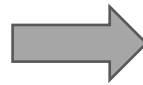
```
$ ls */LM.X.*  
...  
$ fplot */LM.X.*
```

- Do the fringe files make sense?
- Do you need to expand the `ffres2pcp` limits?
- If there are no fringe files, try running **fourfit** by hand for a few scans

Be careful expanding the ddTEC limit:

- Varying dTECs between the pol-products can bias the pcphases
- If you really can't find enough scans for a particular station, maybe try using a different reference station? (Eg, for Hobart or Katherine, use Kokee instead of GGAO/MGO)
- This will run a bunch of **fourfit** jobs, but only for those baselines

Not enough scans



VGOS post-processing: **ffres2pcp**

Be careful expanding the ddTEC limit:

- Varying dTECs between the pol-products can bias the pcphases
- If you really can't find enough scans for a particular station, maybe try using a different reference station? (Eg, for Hobart or Katherine, use Kokee instead of GGAO/MGO)
- This will run a bunch of **fourfit** jobs, but only for those baselines

```
~/geodesy/3825/scratch/YYYYMMDD-HHMMSS/3825$ ffres2pcp.py  
cf_3825_MEHLNSV_pcp H L . -n 16 -s 10 -a 30 -d 5 -q 5 -p -w
```

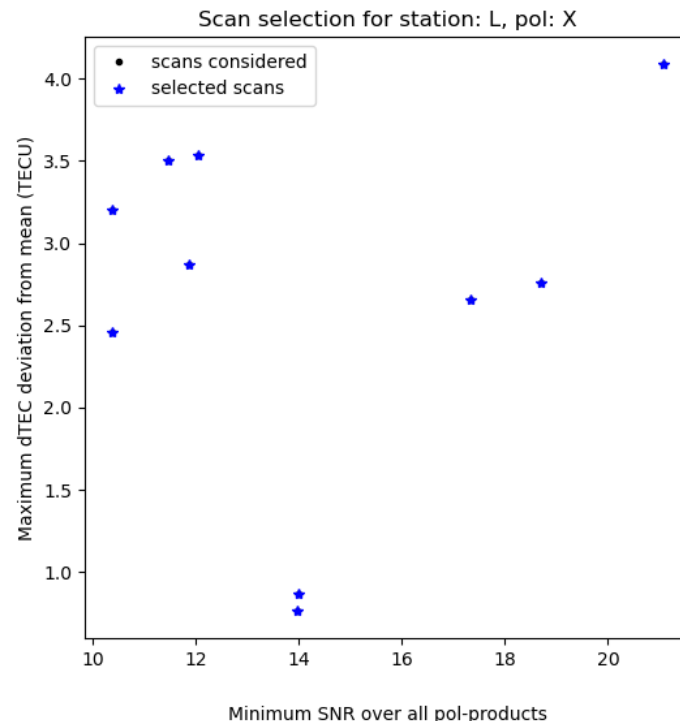


Use the control file generated by the first run of pcphases to take advantage of the updated pcphases for Kokee!

10 scans; better!



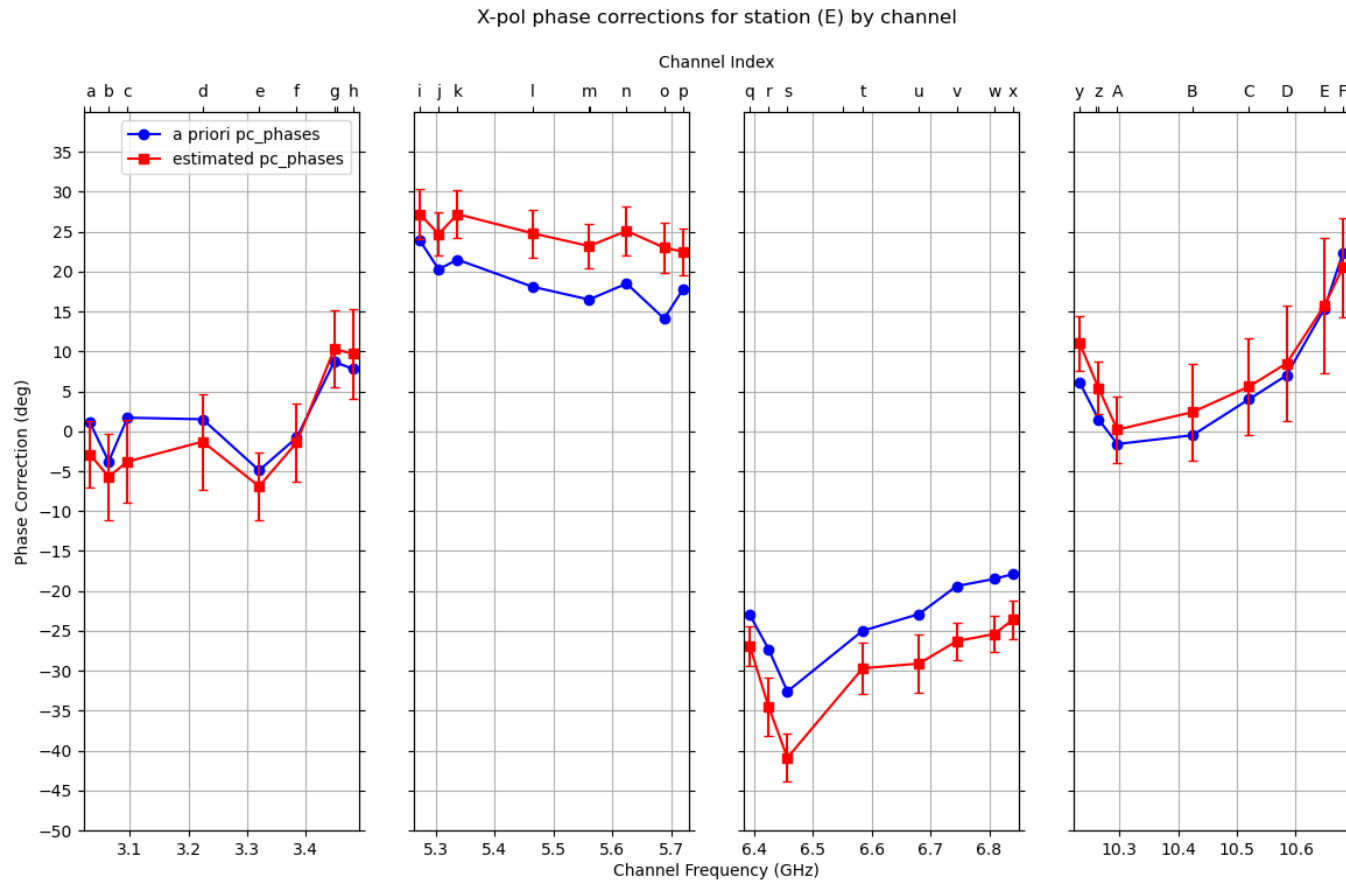
Remember to copy-paste the Hobart pcphases from cf_HL_pcp into cf_MEHLNSV_pcp.



VGOS post-processing: **ffres2pcp**

Once you have pcphases for each station, check the comparison plots with the a priori phases.

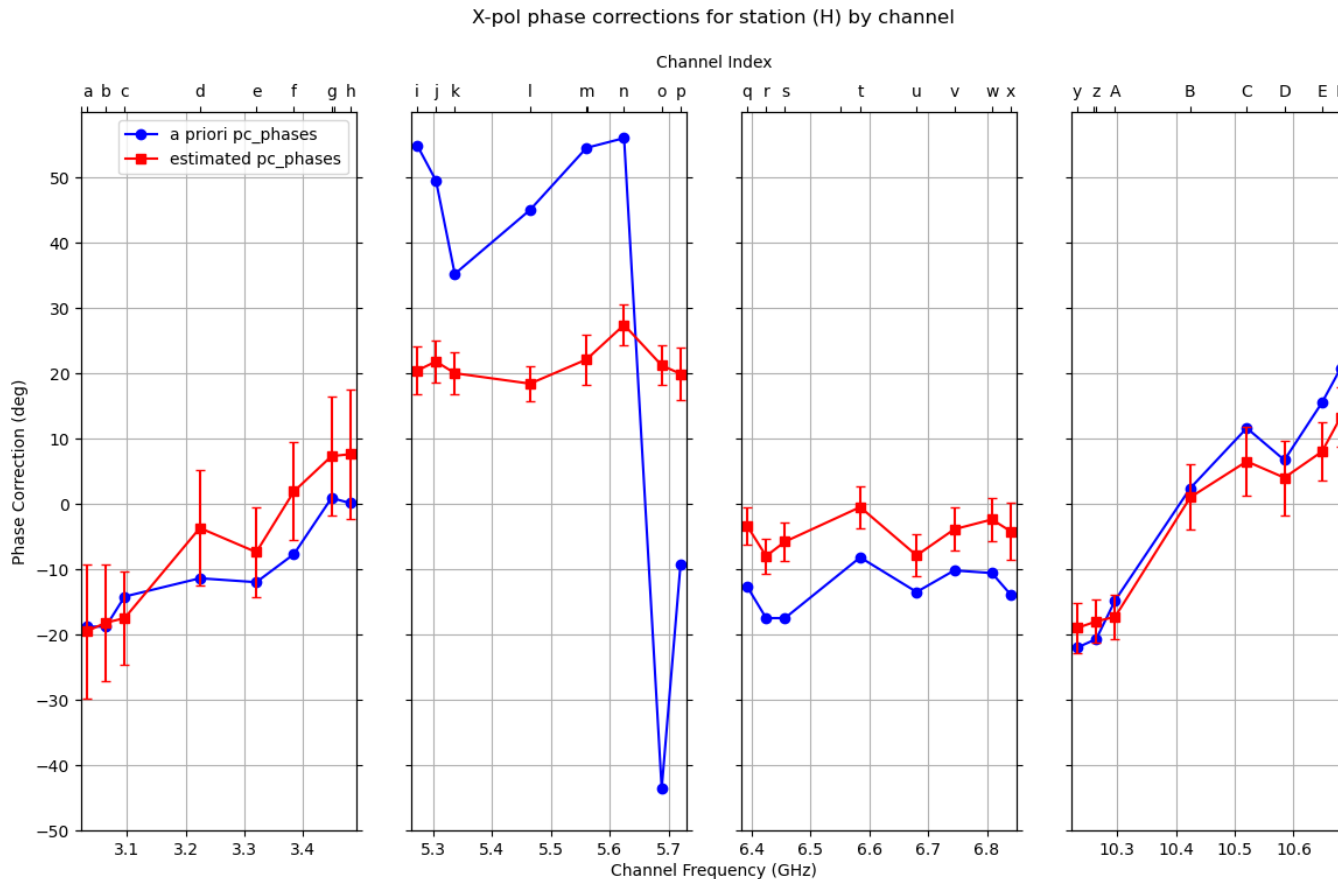
This looks good:



VGOS post-processing: **ffres2pcp**

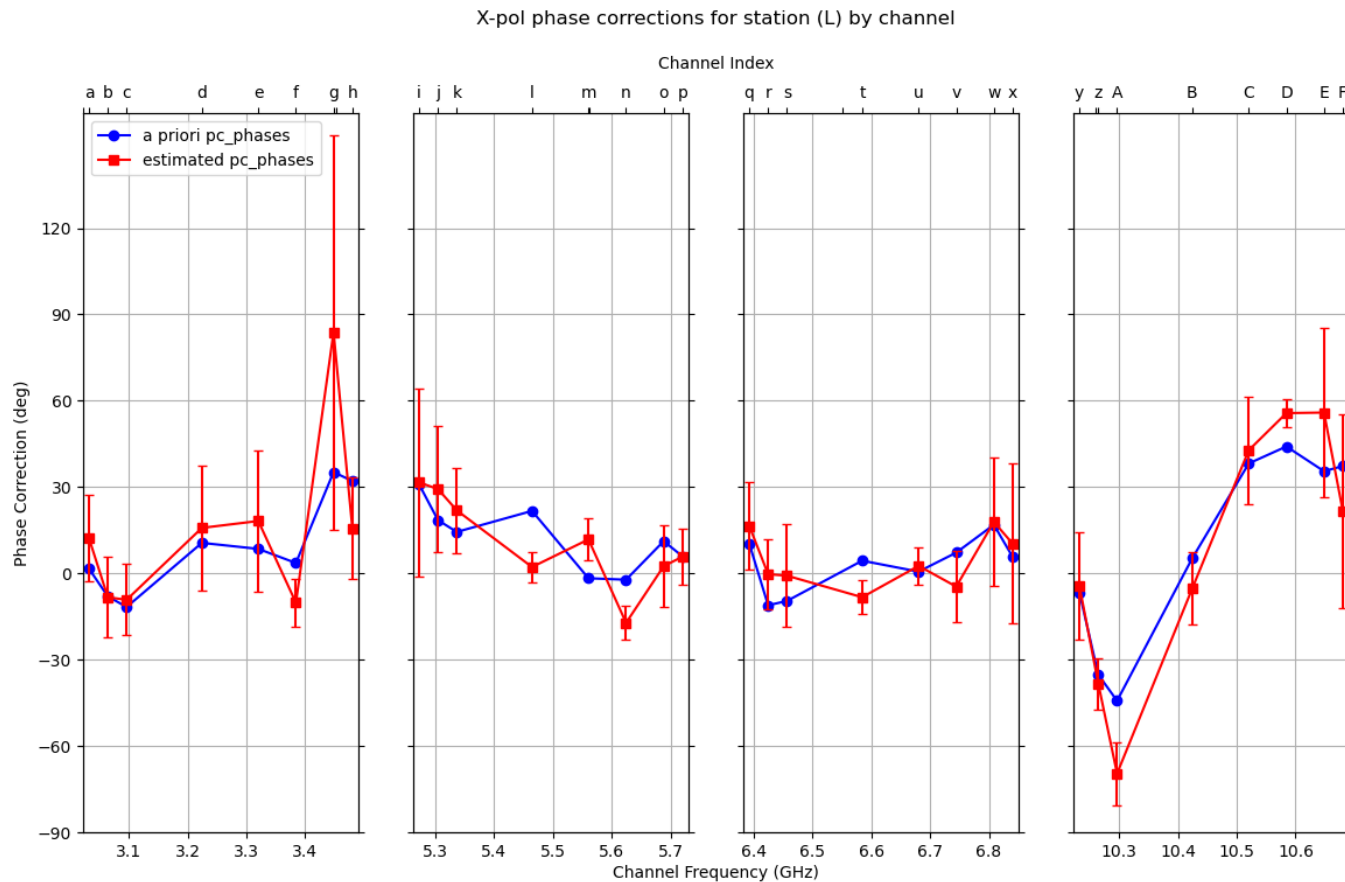
Once you have pcphases for each station, check the comparison plots with the a priori phases.

This is fine, Kokee fixed their band-B R2DBE:



Once you have pcp_hases for each station, check the comparison plots with the a priori phases.

Hobart has a channel with RFI:



I nearly always have to use Kokee for baselines to Australia; there just aren't enough scans to GGAO/MGO.

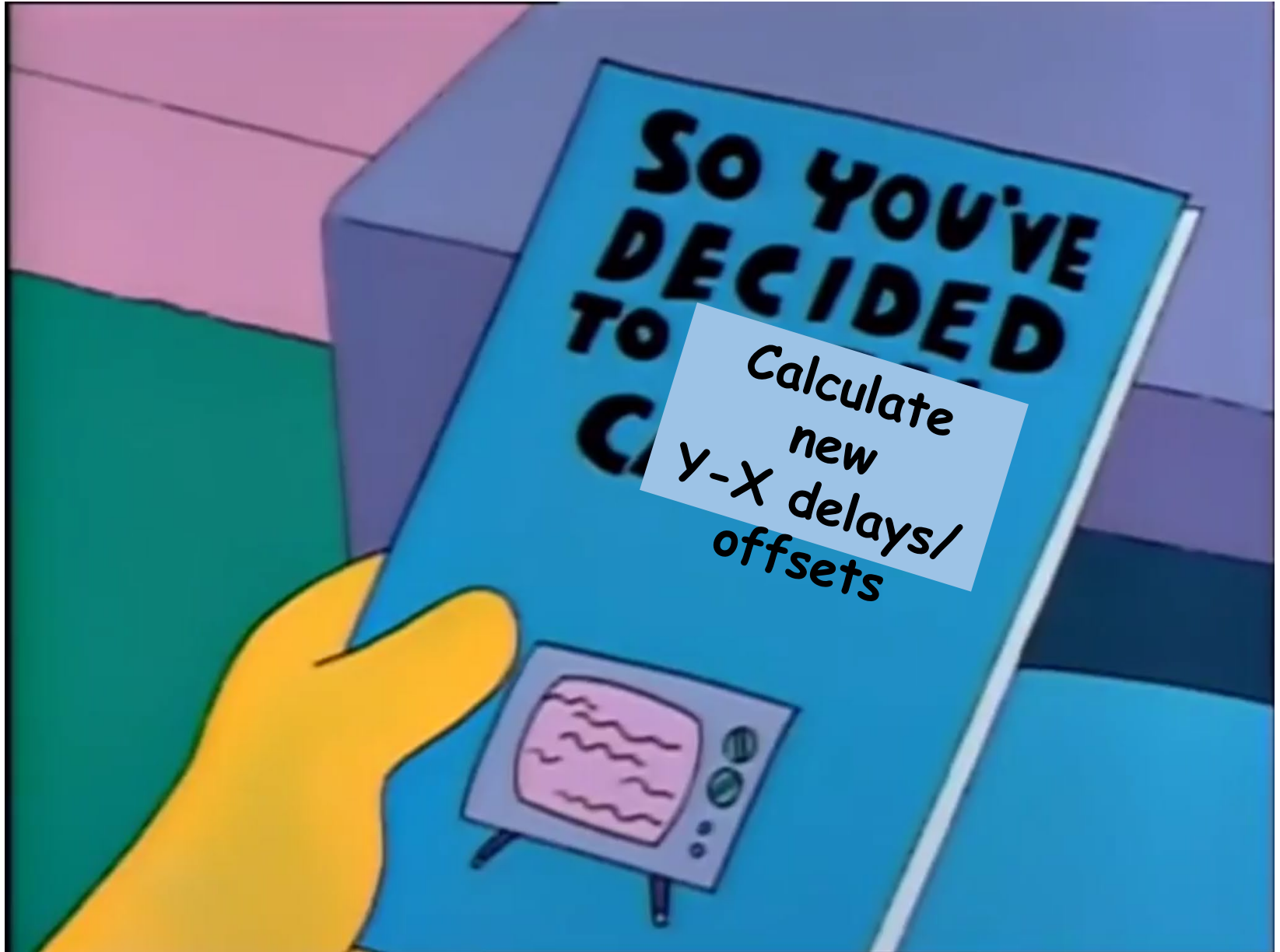
The dTEC threshold always has to be expanded for those stations.

ffres2pcp doesn't typically crash, but if it has to check many, many fringe files for the right control file hash it can take a long time before it starts running **fourfit** jobs.

(Caveat: this week we discovered a bug that blew up the allocated memory if there were too many good scans!)

Always a good idea to check the **pcphases** control file for sane values, no line breaks, etc. You can also use it to **fourfit** a few scans (any baseline) by hand and check that the residual phases are flat!

What's your experience?



Once I've generated a good set of pcphases, I move back to the top directory and run **fourphase.py**

- **fourphase** will go through much of the same routine as **ffres2pcp**: create a scratch directory, check for baselines, assemble a list of scans, check for previously generated fringe files with the correct pol-products and control file hash, and then fire off a bunch of **fourfit** jobs
- It uses all the baselines, so the selection of reference station is not critical
- It takes a long time

```
~/geodesy/3825$ fourphase.py cf_3825_MEHLNSV_pcphases M EHLNSV .  
                -n 16 -s 15 -a 30 -d 5 -p -b 314-1000
```

Once I've generated a good set of pcphases, I move back to the top directory and run **fourphase.py**

- **fourphase** will go through much of the same routine as **ffres2pcp**: create a scratch directory, check for baselines, assemble a list of scans, check for previously generated fringe files with the correct pol-products and control file hash, and then fire off a bunch of **fourfit** jobs
- It uses all the baselines, so the selection of reference station is not critical
- It takes a long time

```
~/geodesy/3825$ fourphase.py cf_3825_MEHLNSV pcphases M EHLNSV .  
-n 16 -s 15 -a 30 -d 5 -p -b 314-1000
```

Protip: if you have a 24hr experiment, there are plenty of scans, you can limit the number you fringe-fit using the `begin_scan_limit` and `end_scan_limit`.

Here, Hobart joined at 314-1211, so I only used the last third of the session (still 22k jobs).

Once I've generated a good set of pcphases, I move back to the top directory and run **fourphase.py**

- **fourphase** will go through much of the same routine as **ffres2pcp**: create a scratch directory, check for baselines, assemble a list of scans, check for previously generated fringe files with the correct pol-products and control file hash, and then fire off a bunch of **fourfit** jobs
- It uses all the baselines, so the selection of reference station is not critical
- It takes a long time

```
~/geodesy/3825$ fourphase.py cf_3825_MEHLNSV_pcphases M EHLNSV .  
                -n 16 -s 15 -a 30 -d 5 -p -b 314-1000
```

fourphase will:

- fringe every scan/baseline in the requested range
- select good scans based on the SNR, ddTEC parameters (-a parameter)
- rerun the fringe search with a modified control file (“ion-search”)
 - (this step is probably redundant)
- then fix the ion-search for each scan/baseline at the weighted mean of dTEC and fringe the four pol-prods again
- calculate the y-x delay and phase offset for that scan/baseline using the mean of [YY-XY,YX-XX] multiband delay or residual phase (reverse for remote station)
- then average the results from the set of good scans

VGOS post-processing: **fourphase**

Again, the SNR and differential dTEC limits are important for some baselines. Unfortunately there aren't plots available check the SNR-ddTEC parameter space like in ffres2pcp (but it should be more or less the same).

We're working to improve the logfile and error messaging in fourphase to make it more helpful when it fails to find good scans for a particular station.

If fourphase fails for a particular station:

```
INFO:vpal.fourphase_lib:station: G had 362 total data points for determining delay/phase offsets of which 5 were cut
INFO:vpal.fourphase_lib:station: E had 554 total data points for determining delay/phase offsets of which 3 were cut
INFO:vpal.fourphase_lib:station: H had 300 total data points for determining delay/phase offsets of which 2 were cut
INFO:vpal.fourphase_lib:station: L had 12 total data points for determining delay/phase offsets of which 0 were cut
INFO:vpal.fourphase_lib:station: N had 348 total data points for determining delay/phase offsets of which 12 were cut
INFO:vpal.fourphase_lib:station: P had 0 total data points for determining delay/phase offsets of which 0 were cut
INFO:vpal.fourphase_lib:station: S had 568 total data points for determining delay/phase offsets of which 3 were cut
INFO:vpal.fourphase_lib:station: T had 640 total data points for determining delay/phase offsets of which 4 were cut
INFO:vpal.fourphase_lib:station: Y had 640 total data points for determining delay/phase offsets of which 19 were cut
ERROR:vpal.fourphase_lib>Error: station P has no useable delay offset data to compute mean Y-X delay offset.
ERROR:vpal.fourphase_lib>Error: station P has no useable delay offset data to compute std. dev. of Y-X delay offset.
ERROR:vpal.fourphase_lib>Error: station P has no useable phase offset data to compute mean Y-X phase offset.
ERROR:vpal.fourphase_lib>Error: station P has no useable phase offset data to compute std. dev of Y-X phase offset.
```

VGOS post-processing: **fourphase**

Again, the SNR and differential dTEC limits are important for some baselines. Unfortunately there aren't plots available check the SNR-ddTEC parameter space like in ffres2pcp (but it should be more or less the same).

We're working to improve the logfile and error messaging in fourphase to make it more helpful when it fails to find good scans for a particular station.

(until recently there was a bug in the config handling, the `-d` parameter was not being used)

If fourphase fails for a particular station:

```
INFO:vpal.fourphase_lib:station: G had 362 total data points for determining delay/phase offsets of which 5 were cut
INFO:vpal.fourphase_lib:station: E had 554 total data points for determining delay/phase offsets of which 3 were cut
INFO:vpal.fourphase_lib:station: H had 300 total data points for determining delay/phase offsets of which 2 were cut
INFO:vpal.fourphase_lib:station: L had 12 total data points for determining delay/phase offsets of which 0 were cut
INFO:vpal.fourphase_lib:station: N had 348 total data points for determining delay/phase offsets of which 12 were cut
INFO:vpal.fourphase_lib:station: P had 0 total data points for determining delay/phase offsets of which 0 were cut
INFO:vpal.fourphase_lib:station: S had 568 total data points for determining delay/phase offsets of which 3 were cut
INFO:vpal.fourphase_lib:station: T had 640 total data points for determining delay/phase offsets of which 4 were cut
INFO:vpal.fourphase_lib:station: Y had 640 total data points for determining delay/phase offsets of which 19 were cut
ERROR:vpal.fourphase_lib:Error: station P has no useable delay offset data to compute mean Y-X delay offset.
ERROR:vpal.fourphase_lib:Error: station P has no useable delay offset data to compute std. dev. of Y-X delay offset.
ERROR:vpal.fourphase_lib:Error: station P has no useable phase offset data to compute mean Y-X phase offset.
ERROR:vpal.fourphase_lib:Error: station P has no useable phase offset data to compute std. dev of Y-X phase offset.
```

- Check for fringe files along some expected baselines; for Katherine (P), check for HP baselines: `$ ls */HP.X.*`
- What is the min SNR among the pol-products? What is the spread of dTECs?
- Did it generate any “ion-fixed-HP” control files? `$ ls */*ion-fixed-HP`
- If not, it never found scans within the specified min-SNR, max-ddTEC, qcode parameters.

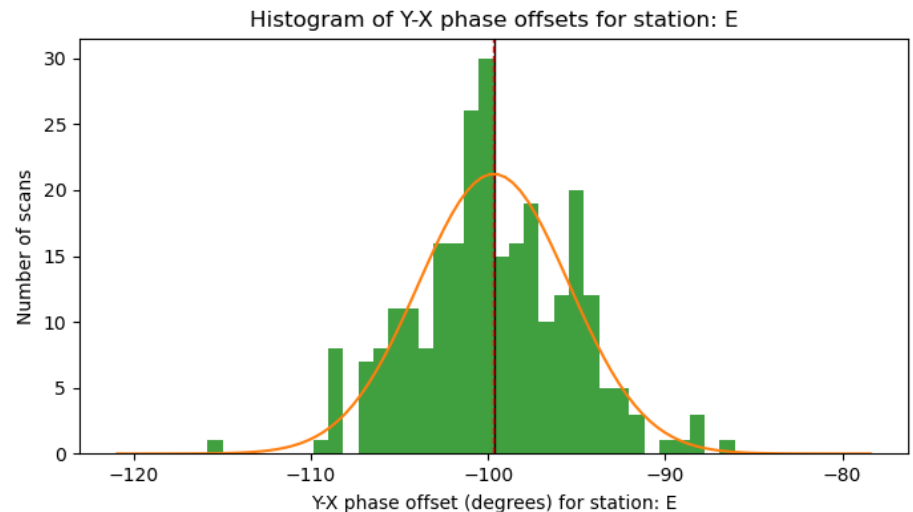
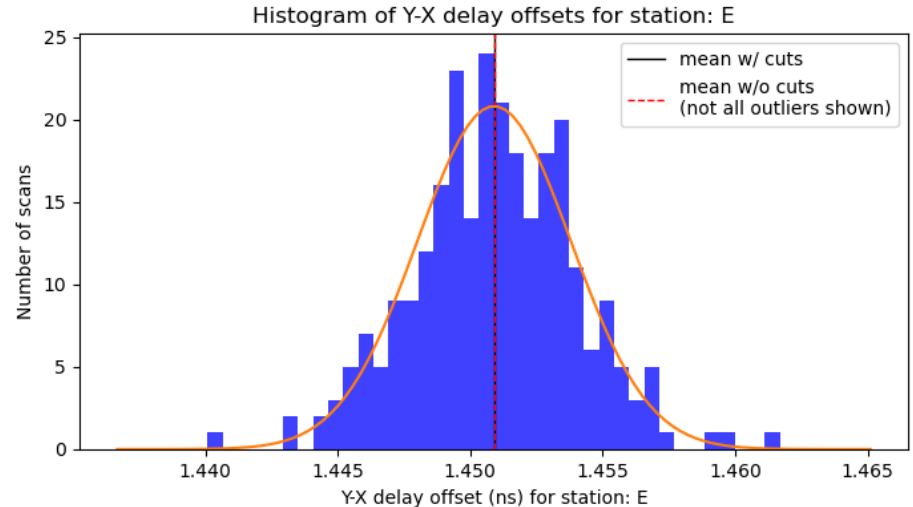
VGOS post-processing: **fourphase**

Ok, hopefully now **fourphase** has finished, the report json file is generated, all stations have some scans, and you can run **summarize_report.py** to check the plots.

VGOS post-processing: **fourphase**

Ok, hopefully now **fourphase** has finished, the report json file is generated, all stations have some scans, and you can run **summarize_report.py** to check the plots.

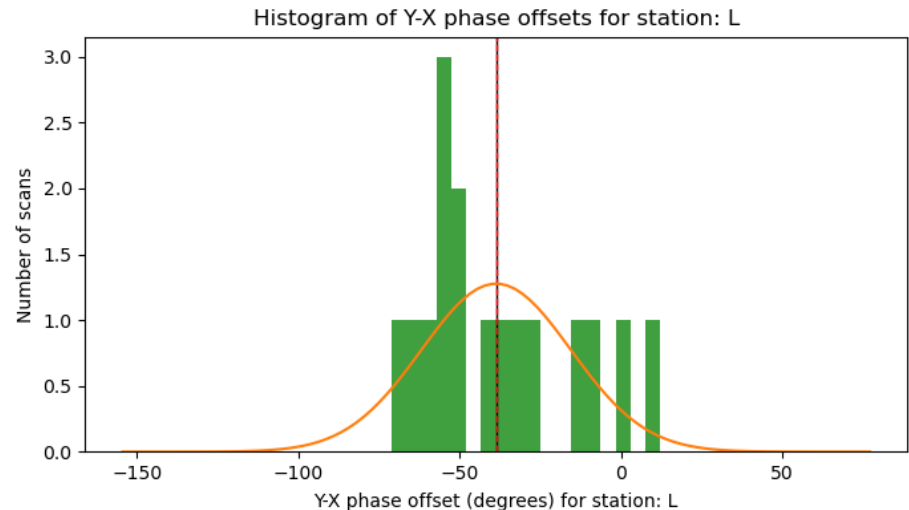
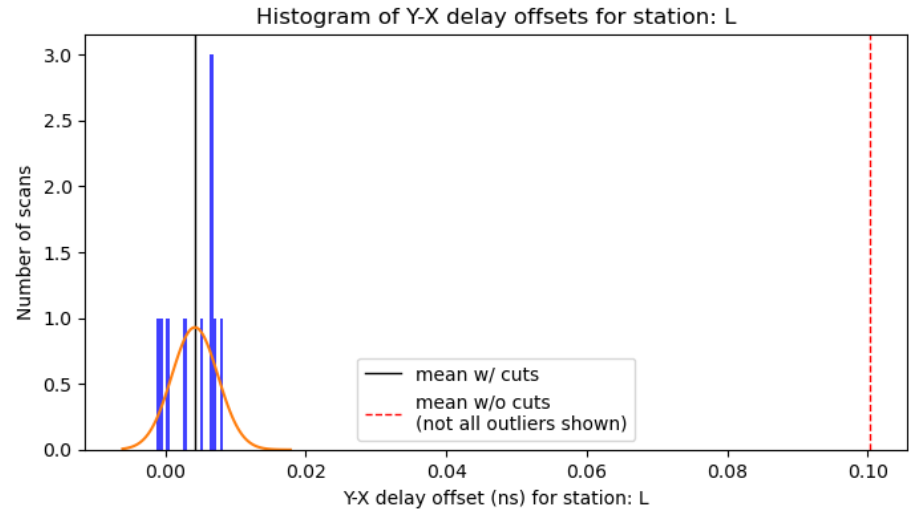
This is good!



VGOS post-processing: **fourphase**

Ok, hopefully now **fourphase** has finished, the report json file is generated, all stations have some scans, and you can run **summarize_report.py** to check the plots.

This is...ok



VGOS post-processing: **fourphase**

Ok, hopefully now **fourphase** has finished, the report json file is generated, all stations have some scans, and you can run **summarize_report.py** to check the plots.

Compare the initial control file, or one from a recent experiment, with the pstokes file generated by fourphase.

- Have the `pc_delay_y` and `pc_phase_offset_y` parameters changed?

Old:

```
if station L
  pc_delay_x 0.0
  pc_delay_y 0.008 * (ns) estimated error is +/- 0.004
  pc_phase_offset_x 0.0
  pc_phase_offset_y -53.2 * (deg) estimated error is +/- 3.8
```

New:

```
if station L
  pc_delay_x 0.0
  pc_delay_y 0.004 * (ns) estimated error is +/- 0.003
  pc_phase_offset_x 0.0
  pc_phase_offset_y -38.7 * (deg) estimated error is +/- 23.2
```

VGOS post-processing: **fourphase**

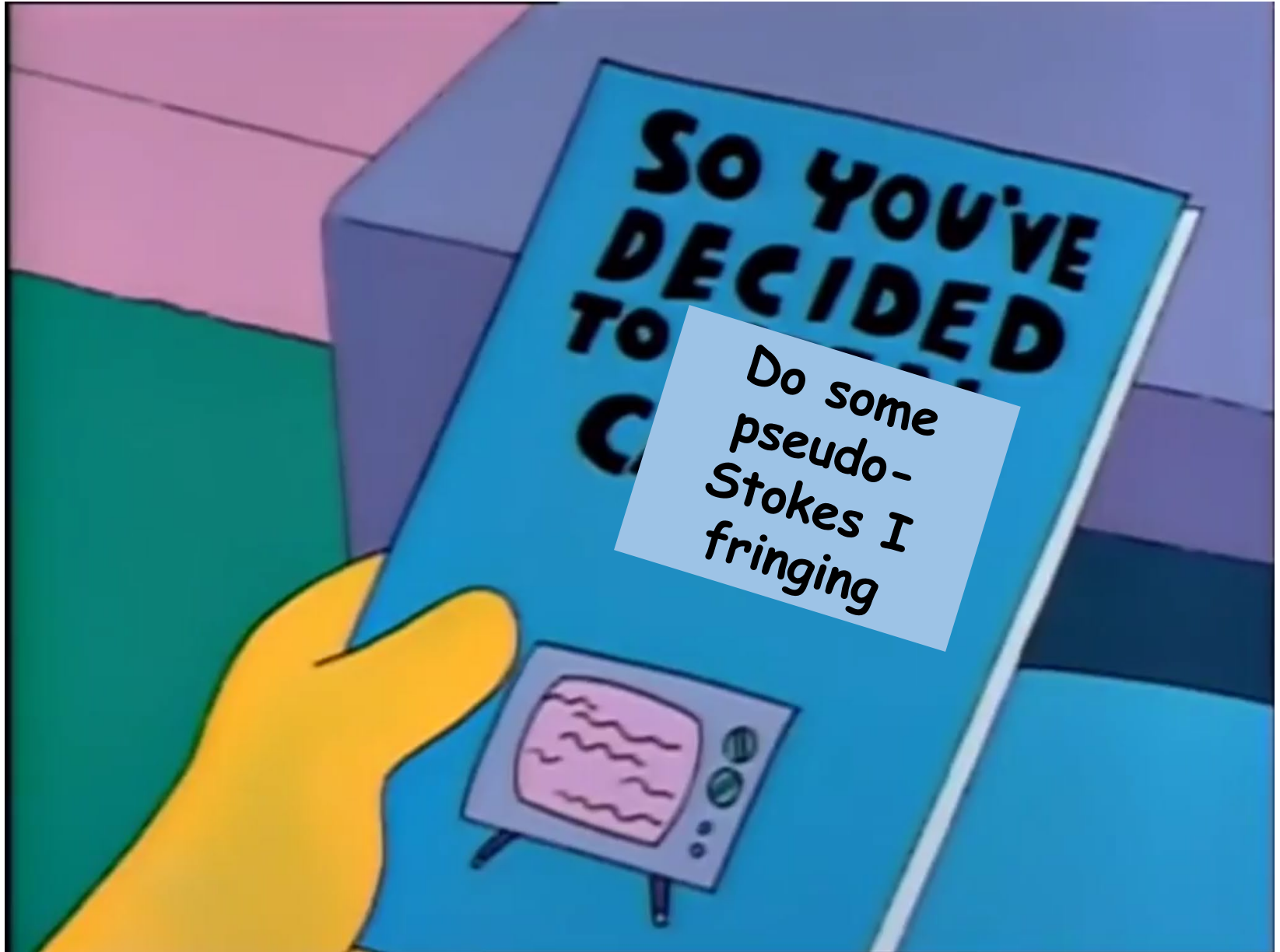
Ok, hopefully now **fourphase** has finished, the report json file is generated, all stations have some scans, and you can run **summarize_report.py** to check the plots.

Compare the initial control file, or one from a recent experiment, with the pstokes file generated by fourphase.

- Have the `pc_delay_y` and `pc_phase_offset_y` parameters changed?

Lots of room for improvement in this script! But it is critical to get a large distribution of y-x delays/offsets so the linear polarizations can be summed coherently into pseudo-Stokes-I. (Note: we don't correct for amplitude, only delay and phase!)

What kind of problems have you encountered?



VGOS post-processing: **batch_fourfit**

This one is usually pretty easy (the IONEX predictions are your friend!):

```
~/geodesy/3825$ get_ionex_dtec_bounds.py /data-sc03/difxoper/vr2206/  
/data-sc16/geodesy/ionex/ vr2206_ionex_tec.json  
  
~/geodesy/3825$ batch_fourfit.py cf_3825_MEHLNSV_pstokes MEHLNSV I .  
-n 16 -p -t vr2206_ionex_tec.json
```

When it's done, check the:

- phase residuals (and range of dTECs for each baseline)
- multiband delay for each baseline
- G and H codes

For phase residuals, I typically remove the channels if:

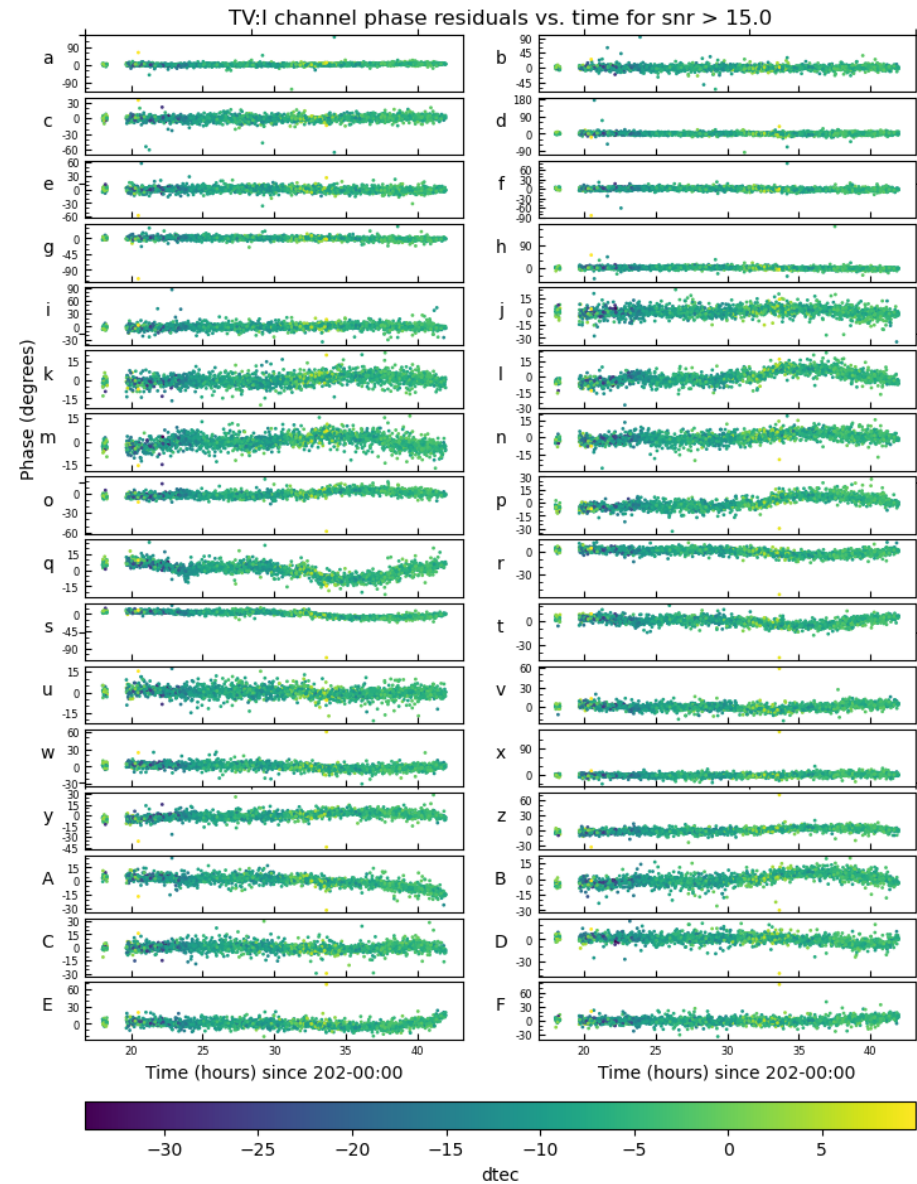
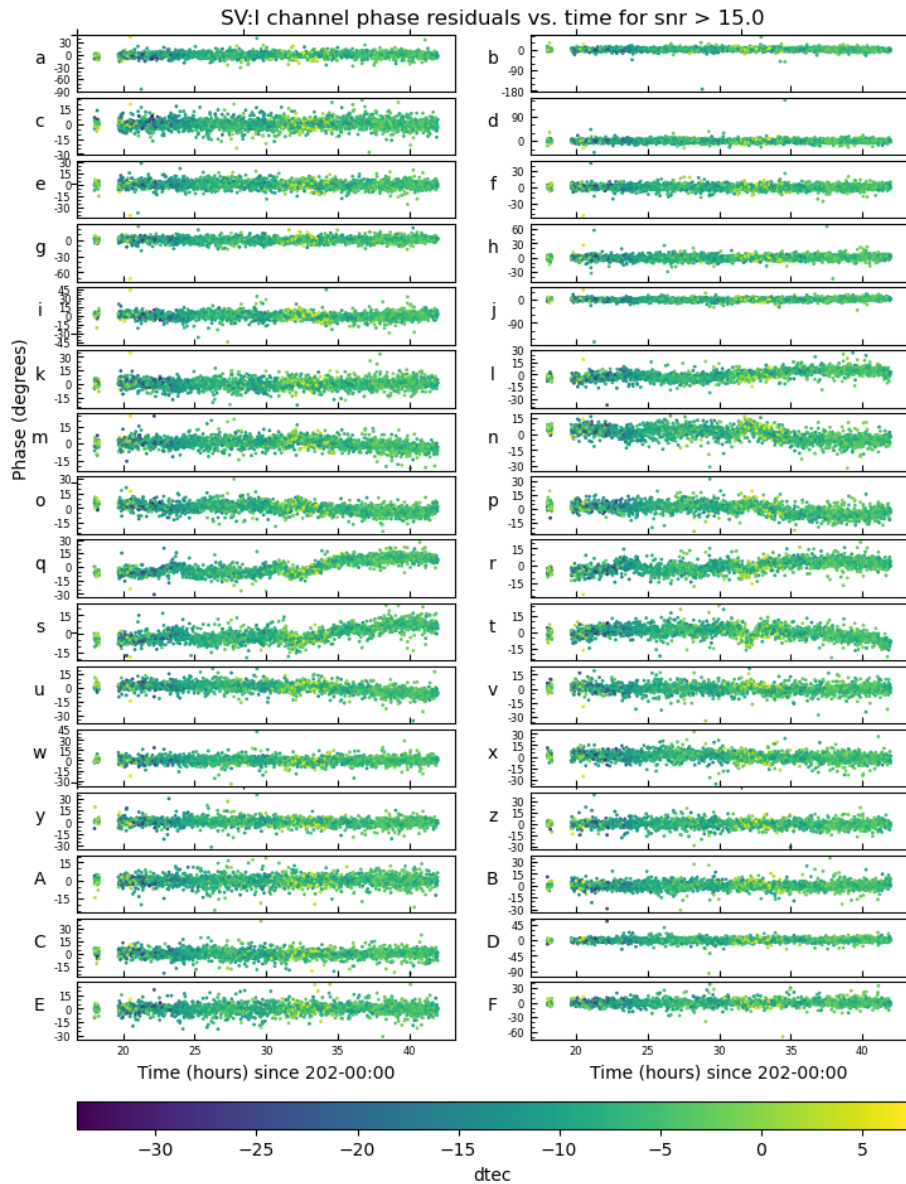
- Mean phase residual is > 10 deg
- A clear outlier in variance
- Clearly bimodal (indicative of RFI)
- Bands A & D are important for ionospheric search, so think twice

G,H codes: greater than 10% of scans is too many. Figure out why!

Multiband delay should be flat & not close to the limits of the search range.

VGOS post-processing: phase residuals

The Onsals often have some time-dependent modulation of the phases in band C,D

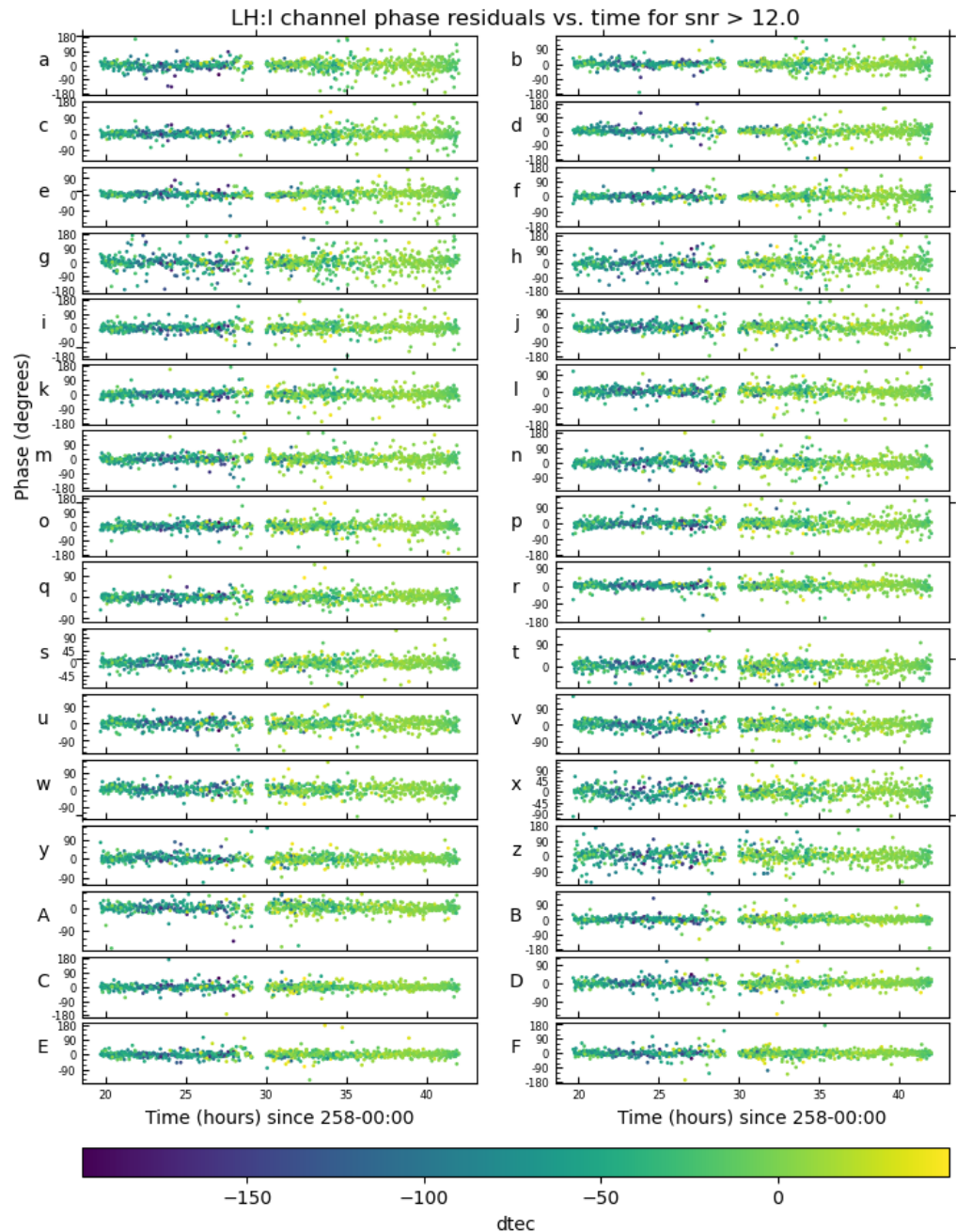


VGOS post-processing: phase residuals

Hobart channels g,h are often noisy; dTEC range is large.

If you see an interesting feature, inspect a few fringe plots! Easy to select fringe plots for a particular baseline by hour with **fplot** and scroll through:

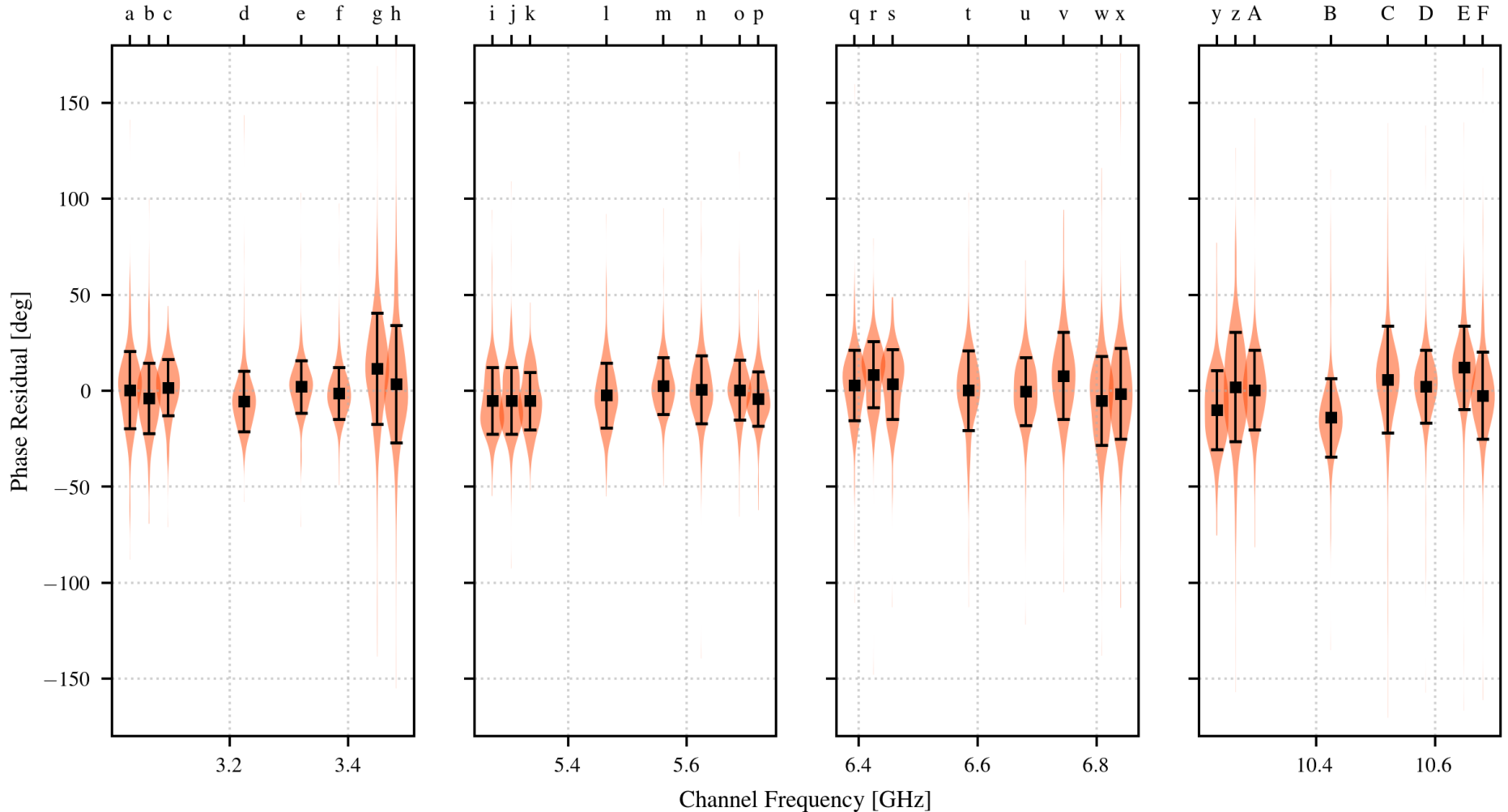
```
$ fplot 351-12*/LH.X.*
```



VGOS post-processing: phase residuals

Hobart can also have channels in band D with large mean residuals. There are no short baselines to Hobart, so the scatter of each channel is large. Makes it hard to diagnose problems!

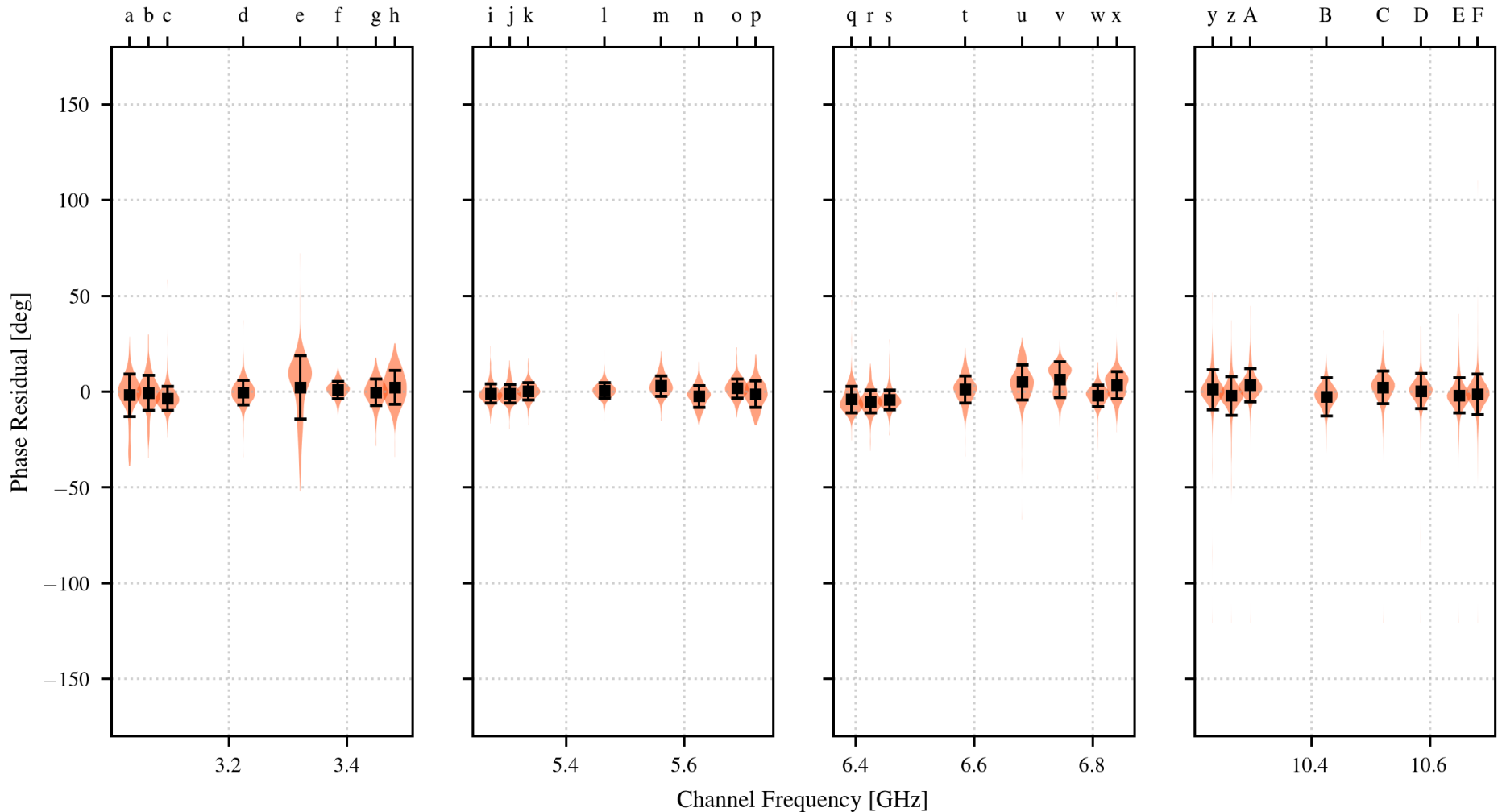
Channel-by-channel phase residuals for station L in experiment 3825



VGOS post-processing: phase residuals

NyAlesund has short baselines to use for comparison, so the variance is smaller, but they do have a few problem channels.

Channel-by-channel phase residuals for station N in experiment 3826



audit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

```
$ alist -o alist.test .  
alist: Successfully wrote 34916 A-file  
lines to file alist.test  
  
$ aedit  
audit> read alist.test  
audit> summ 2
```

First, build an alist file for the experiment. This command will sweep up all the fringe files in the directory (it takes some time).

Start aedit

Read the alist file

Print a summary of the data in the file

VGOS post-processing: aedit tools

audit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

SUMMARY OF UNFLAGGED DATA IN MEMORY

Total number of unflagged fringe records = 34916

Earliest scan: 122-348-180000

Latest scan: 122-349-175848

Earliest procdte: 123-077-2124

Latest procdte: 123-096-2218

Stations present: EGHLNPSTY

Baselines present: GT SY TE TY EY GY GS GE SE NT NS NY LH HP LP HY LS

PS LT PT GH NE HE HN HT HS LN PN PY PE LE LY GP GL

Frequencies present: X

Polarizations present: YY XY YX XX

SNR extrema: 5.216 740.6

Experiments present: 3826

Sources present: 0003-066 0017+200 0035-252 0059+581 0109+224 0119+115

0131-522 0133+476 0202+319 0215+015 0235+164 0322+222 0332-403

0454-234 0458-020 0537-441 0552+398 0556+238 0606-223 0613+570

...

1929+226 1954-388 2008-159 2052-474 2059+034 2113+293 2126-158

2149+056 2214+241 2227-088 2229+695 2255-282 2309+454 2319+317

2325+093 2329-384 3C274 3C371 3C418 CTA26 NRAO150

OJ287

Quality code summary:

A	B	C	D	E	F	G	H	0	1	2	3	4	5	6	7	8	9	?
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

0	0	0	10	0	0	945	2760	2959	1	1	7	20	166	727	3050	9058	15212	0
---	---	---	----	---	---	-----	------	------	---	---	---	----	-----	-----	------	------	-------	---

There are 0 flagged records present

VGOS post-processing: aedit tools

audit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

SUMMARY OF UNFLAGGED DATA IN MEMORY

Total number of unflagged fringe records = 34916

Earliest scan: 122-348-180000

Latest scan: 122-349-175848

Earliest procdates: 123-077-2124

Latest procdates: 123-096-2218

Stations present: EGHLNPSTY

Baselines present: GT SY TE TY EY GY GS GE SE NT NS NY LH HP LP HY LS

PS LT PT GH NE HE HN HT HS LN PN PY

Frequencies present: X

Polarizations present: YY XY YX XX

SNR extrema: 5.216 740.6

Experiments present: 3826

Sources present: 0003-066 0017+200 0035-252 0059+581 0109+224 0119+115

0131-522 0133+476 0202+319 0215+015 0235+164 0322+222 0332-403

0454-234 0458-020 0537-441 0552+398 0556+238 0606-223 0613+570

...

1929+226 1954-388 2008-159 2052-474 2059+034 2113+293 2126-158

2149+056 2214+241 2227-088 2229+695 2255-282 2309+454 2319+317

2325+093 2329-384 3C274 3C371 3C418 CTA26 NRAO150

OJ287

Quality code summary:

A B C D E F G H 0 1 2 3 4 5 6 7 8 9 ?

0 0 0 10 0 0 945 2760 2959 1 1 7 20 166 727 3050 9058 15212 0

There are 0 flagged records present

There are four polarization products in the alist records; Stokes I is actually recorded as YX

VGOS post-processing: aedit tools

audit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

SUMMARY OF UNFLAGGED DATA IN MEMORY

Total number of unflagged fringe records = 34916

Earliest scan: 122-348-180000

Latest scan: 122-349-175848

Earliest procdates: 123-077-2124

Latest procdates: 123-096-2218

Stations present: EGHLNPSTY

Baselines present: GT SY TE TY EY GY GS

PS LT PT GH NE HE HN HT HS LN PN PY PE LE LY GP GL

Frequencies present: X

Polarizations present: YY XY YX XX

SNR extrema: 5.216 740.6

Experiments present: 3826

Sources present: 0003-066 0017+200 0035-252 0059+581 0109+224 0119+115

0131-522 0133+476 0202+319 0215+015 0235+164 0322+222 0332-403

0454-234 0458-020 0537-441 0552+398 0556+238 0606-223 0613+570

1929+226 1954-388 2008-159 2052-474 2059+034 2113+293 2126-158

2149+056 2214+241 2227-088 2229+695 2255-282 2309+454 2319+317

2325+093 2329-384 3C274 3C371 3C418 CTA26 NRAO150

OJ287

Quality code summary:

A B C D E F G H 0 1 2 3 4 5 6 7 8 9 ?

0 0 0 10 0 0 945 2760 2959 1 1 7 20 166 727 3050 9058 15212 0

There are 0 flagged records present

We can distinguish different groups of fringe files with the processing date. We did the Stokes I batch job last, so let's filter on procdates from DOY 096.

audit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

```
$ alist -o alist.test .  
alist: Successfully wrote 34916 A-file  
lines to file alist.test
```

```
$ aedit  
audit> read alist.test
```

```
audit> summ 2
```

```
audit> procrange 23096-000000 23096-  
235959
```

```
audit> ed in
```

```
audit> summ 2
```

First, build an alist file for the experiment. This command will sweep up all the fringe files in the directory (it takes some time).

Start aedit

Read the alist file

This will print a statement about how many records have been read from the alist file.

Print a summary of the data in the file

Select files with a procdatetime between 2023-096-00:00:00 and 2023-096-23:59:59

Edit the inputs, or remove files that are flagged as outside of the selected procrange. This will print a statement about how many records are left.

Print another summary

VGOS post-processing: aedit tools

aedit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

SUMMARY OF UNFLAGGED DATA IN MEMORY

Total number of unflagged fringe records = 4824

```
Earliest scan:      122-348-180000
Latest scan:       122-349-175848
Earliest procdates: 123-096-1746
Latest procdates:  123-096-2218
Stations present:  EGHLNPSTY
Baselines present:  TE SY TY GS GT GE SE EY GY NT NS NY LH HP LP HY PS
                   LS PT LT GH NE HN HT HS HE PN LN PY PE LE LY GP GL
Frequencies present: X
Polarizations present: YX
SNR extrema:       5.911 / 40.6
Experiments present: 3826
Sources present:   0003-066 0017+200 0035-252 0059+581 0109+224 0119+115
                   0131-522 0133+476 0202+319 0215+015 0235+164 0322+222 0332-403
                   0454-234 0458-020 0537-441 0552+398 0556+238 0606-223 0613+570
                   1639-062 1741-038 1749+096 1751+288 1806+456 1846+322 1908-201
                   1929+226 1954-388 2008-159 2052-474 2059+034 2113+293 2126-158
                   2149+056 2214+241 2227-088 2229+695 2255-282 2309+454 2319+317
                   2325+093 2329-384 3C274 3C371 3C418 CTA26 NRAO150
                   OJ287
Quality code summary:
  A B C D E F G   H   0 1 2 3 4 5 6 7 8 9  ?
  0 0 0 2 0 0 168 469 7 0 0 1 6 15 87 443 2436 1190 0
```

Now we have a set of 4824 files with a narrow range of procdates and a single polarization (which is actually Stokes I)

123-096-1746
123-096-2218

YX

There are 30092 flagged records present

VGOS post-processing: aedit tools

aedit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

```
$ alist -o alist.test .  
alist: Successfully wrote 34916 A-file  
lines to file alist.test
```

```
$ aedit  
aedit> read alist.test
```

```
aedit> summ 2
```

```
aedit> procrange 23096-000000 23096-  
235959
```

```
aedit> ed in
```

```
aedit> summ 2
```

```
aedit> sort freq
```

```
aedit> sort base
```

```
aedit> sort time
```

```
aedit> write alist.test.Ixy
```

First, build an alist file for the experiment. This command will sweep up all the fringe files in the directory (it takes some time).

Start aedit

Read the alist file

This will print a statement about how many records have been read from the alist file.

Print a summary of the data in the file

Select files with a procdatetime between 2023-096-00:00:00 and 2023-096-23:59:59

Edit the inputs, or remove files that are flagged as outside of the selected procrange. This will print a statement about how many records are left.

Print another summary

Save an alist file with the Ixy fringe records, sorted by frequency, baseline, and time.

aedit is a useful HOPS tool for inspecting fringe properties. Here's a quick example for plotting multiband delays for the final pseudo-Stokes fringe results:

```
aedit> dev /xw
aedit> axis mbd
aedit> grid 2 5
aedit> qcode G H 1-9
aedit> ed in

aedit> plot
```

Select a device to plot (xwin)

Choose the yaxis value (multiband delay)

Set up a grid of 2 columns, 5 rows

Select qcodes G, H, and 1-9 (no nondetections)

Edit inputs to apply the flags

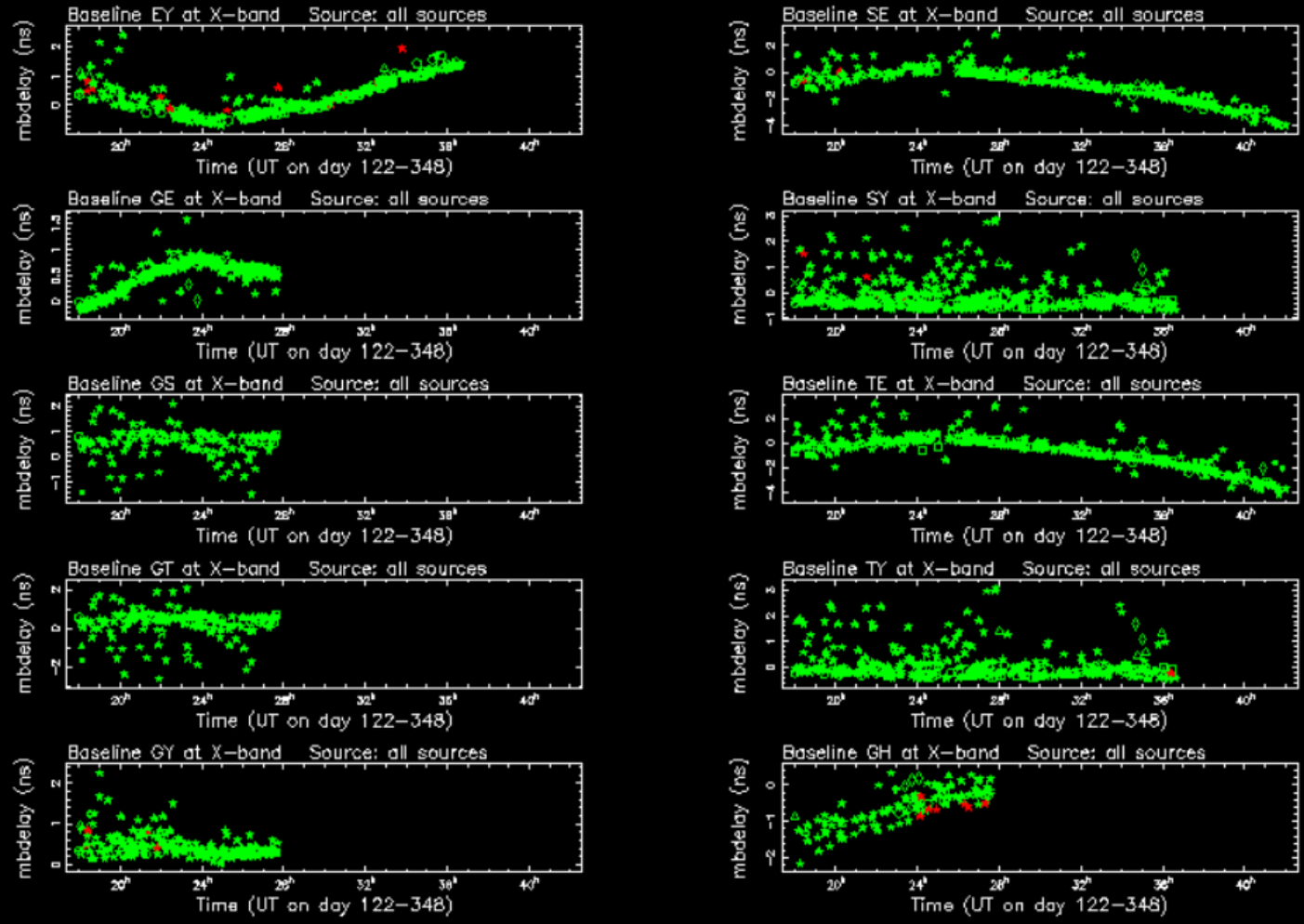
Plot!

VGOS post-processing: aedit tools

aedit
for pl
aedit
aedit
aedit
aedit
aedit

PGPLOT Window 1

AEDIT plot - Expt 3B26, Freq X



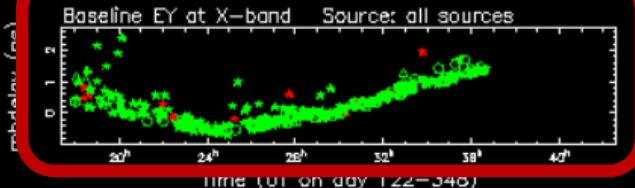
Symbol key: \circ = 0059+581, \times = CTA26, \square = 0955+476, \triangle = 1406-076, \diamond = 2008-159, \star = 2113+293
 \blacktriangle = 2126-158, \blacklozenge = 1749+096, \blacksquare = 1806+456, \blackstar = 1128+385, \star = the rest, \circ = 1751+288

VGOS post-processing: aedit tools

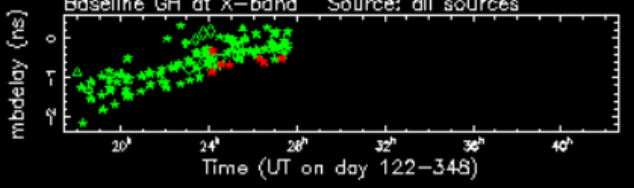
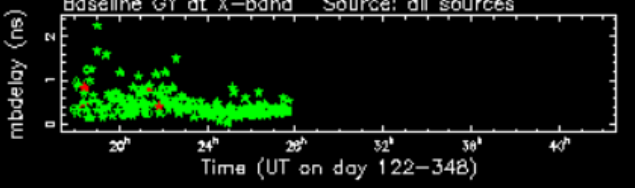
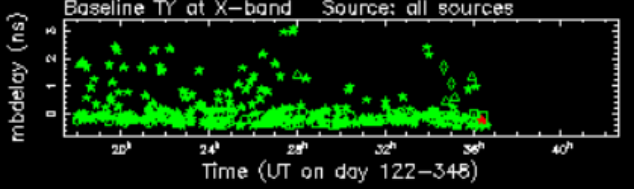
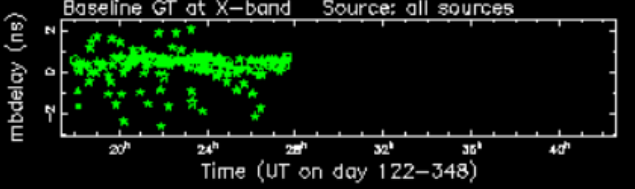
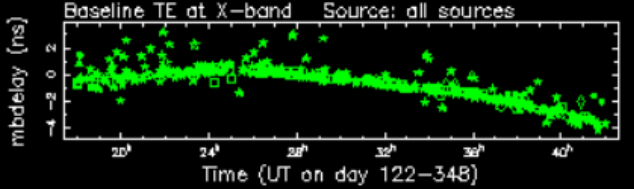
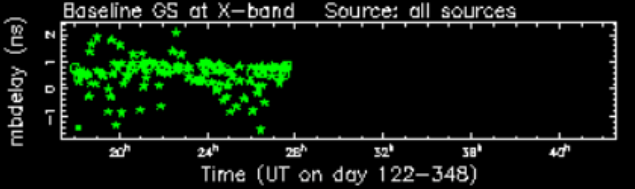
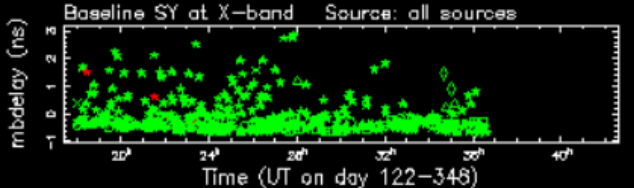
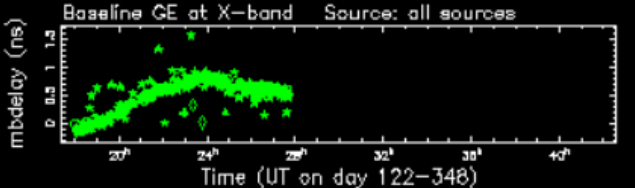
aedit
for pl
aedit
aedit
aedit
aedit
aedit

PGPLOT Window 1

AEDIT plot - Expt 3B26, Freq X



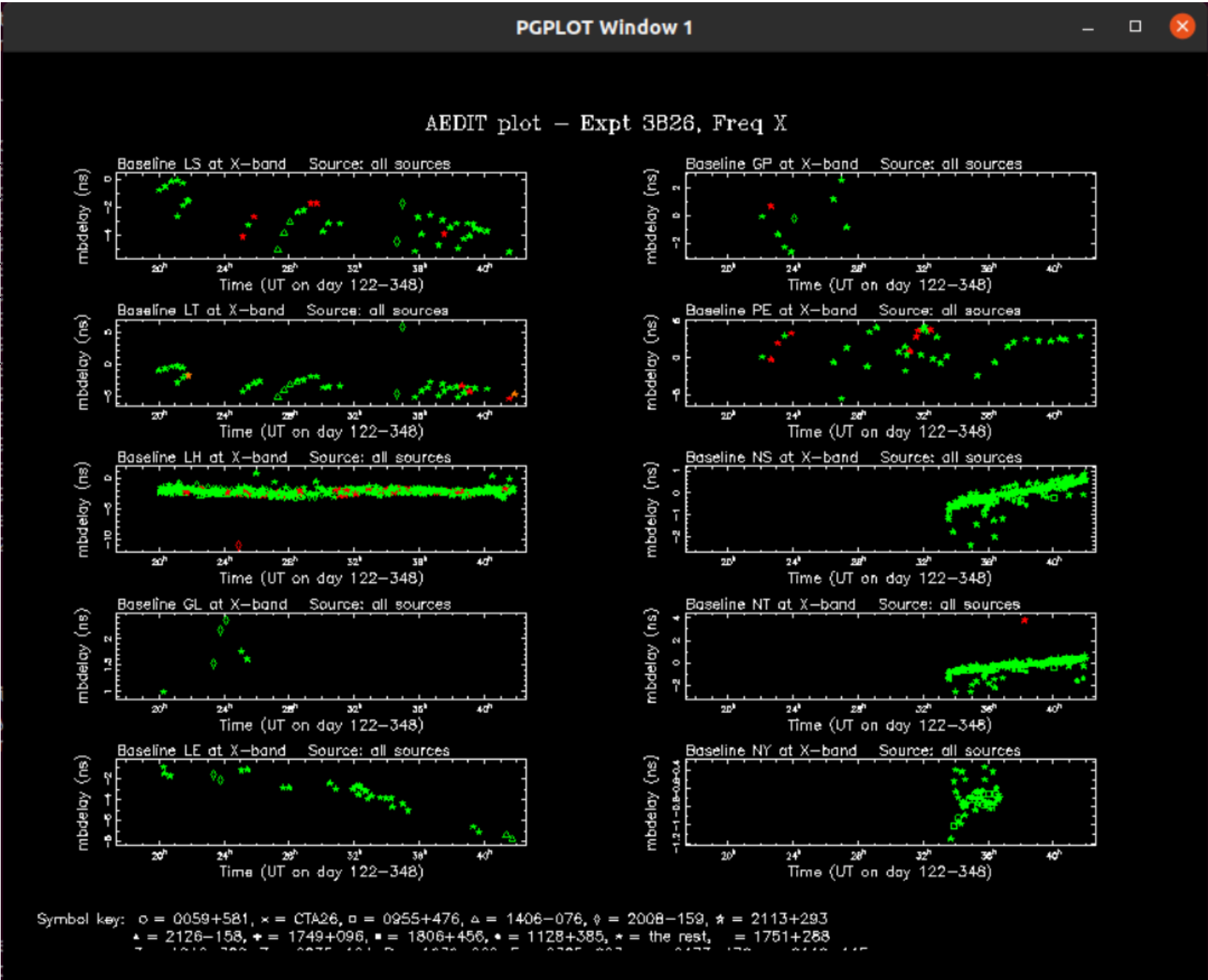
Smoothly varying MBD between [-2,2] nsec on the EY baseline



Symbol key: \circ = 0059+581, \times = CTA26, \square = 0955+476, \triangle = 1406-076, \diamond = 2008-159, \star = 2113+293
 \blacktriangle = 2126-158, \blackstar = 1749+096, \blacksquare = 1806+456, \blacklozenge = 1128+385, \blackstar = the rest, \blacktriangle = 1751+288

VGOS post-processing: aedit tools

aedit
for pl
aedi
aedi
aedi
aedi
aedi

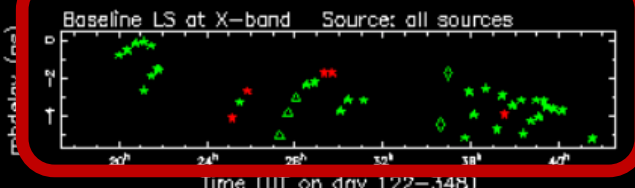


VGOS post-processing: aedit tools

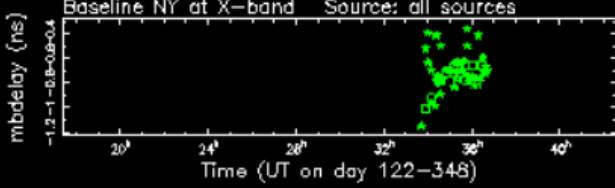
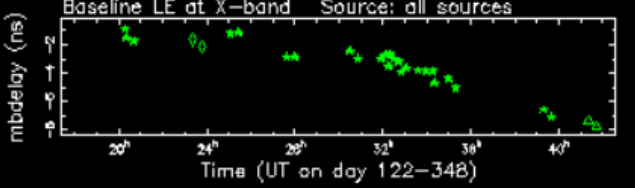
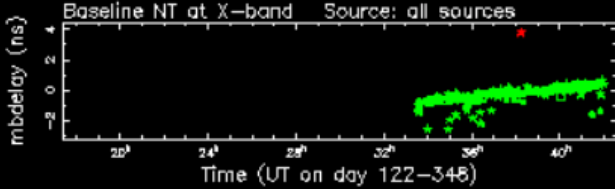
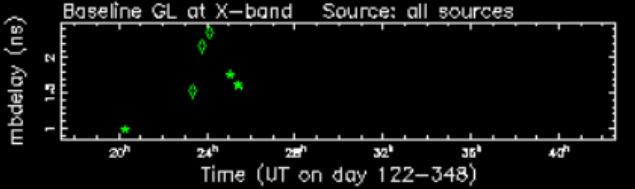
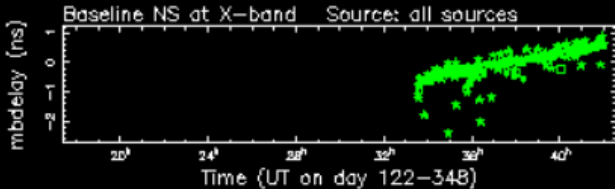
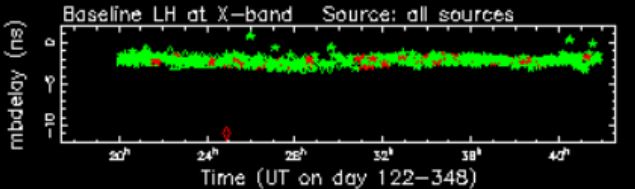
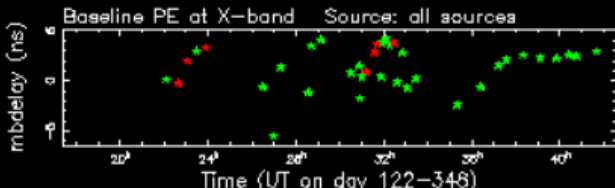
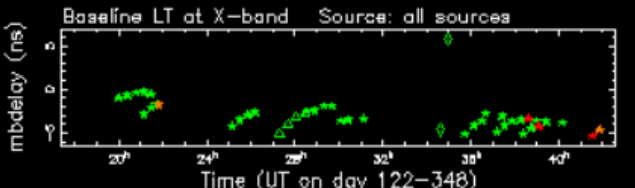
aedit
for pl
aedi
aedi
aedi
aedi
aedi

PGPLOT Window 1

AEDIT plot - Expt 3B26, Freq X



LS baseline has strange curves, probably the Hobart position is wrong



Symbol key: o = 0059+581, x = CTA26, □ = 0955+476, △ = 1406-076, ◇ = 2008-159, ☆ = 2113+293
 ▲ = 2126-158, + = 1749+096, ■ = 1806+456, ◆ = 1128+385, * = the rest, = 1751+268

VGOS post-processing: done

You've built the control file for the experiment!

Now it's off to a final **batch_fourfit** job and building the vgosDB.

Anything else?

VGOS post-processing: looking forward

The VGOS network is growing:

Currently: GEHILMNSTVY (11 stations, 55 possible baselines)

+ P (Katherine)

+ Santa Maria, Fortaleza, AuScope #3...

...very soon we'll have 15 stations and 105 possible baselines!

Need to use:

- Smarter/narrower ionospheric search methods, to speed things up
- Calibration scans for pcphases and y-x delay/offset, to reduce the number of **fourfit** calls

Getting help: hops-dev@mit.edu