MIT HAYSTACK OBSERVATORY

VGOS Processing Demonstration

VGOS Correlation Workshop

MIT Haystack Observatory

May 10, 2019

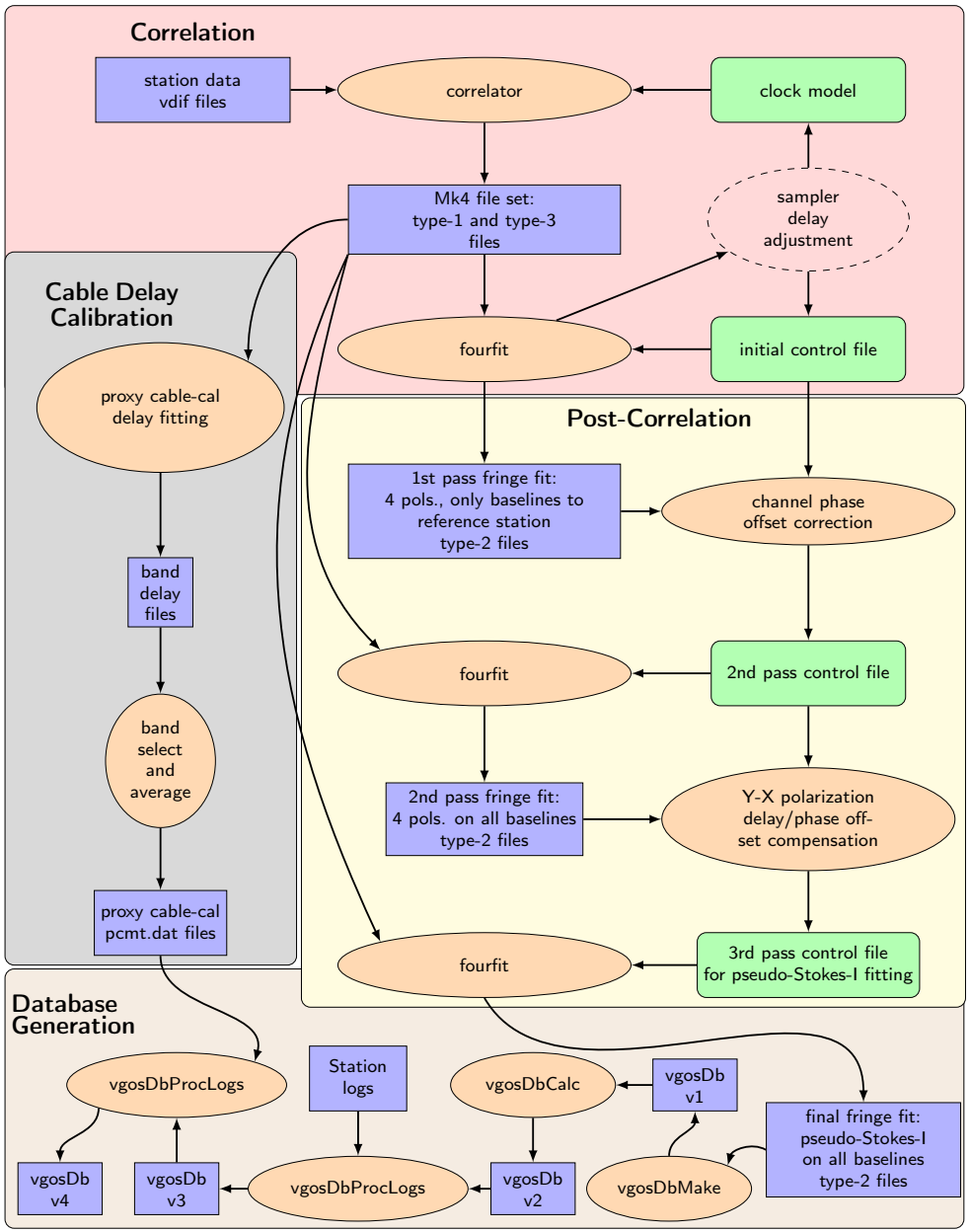
Outline

MIT HAYSTACK OBSERVATORY

- We will follow the general process outlined in the VGOS data processing checklist.
- See page 27 of the manual handout.
- Due to time constraints we will work with some pre-processed data and/or smaller than normal data sets at various steps.
- Feel free to interrupt with questions at any time!

Correlation

MIT HAYSTACK OBSERVATORY



Step 1: Mount/transfer station data

MIT HAYSTACK OBSERVATORY

- Due to time/logistics constraints we won't demonstrate mounting/verifying Mark6 data here.
- We also won't consider E-transfer, since setup is unique to each correlator, and must be arranged in communication with each respective station.

Step 2: Generate the file lists

MIT HAYSTACK OBSERVATORY

- DiFX needs a list of the recorded files for each station and their approximate start/stop times in order to locate the right chunk of data.
- This information can be constructed with the vsum tool. For example:

```
stdbuf -o0 vsum -s --mark6slot 1 | tee ./vt9105_wf.filelist
```

- Depending on amount of data this can take quite some time.
- Note: the `-mark6slot` feature was only made available recently in the DiFX trunk[1].

[1] It's not included with DiFX 2.5.2, which we will otherwise be using throughout.

Steps 3 & 4: Grab the .vex or .skd file

MIT HAYSTACK OBSERVATORY

- To configure DiFX we need the session .vex file.
- For this example we are working with a recent session: vt9105, on April 15, 2019.
- At the moment, there is no .vex file publicly available for this session, so we'll grab the .skd file:

```
wget ftp://cddis.gsfc.nasa.gov/vlbi/ivsdata/aux/2019/vt9105/vt9105.skd
```

- Then convert to .vex with sked:

```
sked ./vt9105.skd
> VWC vt9105.vex.obs
> quit
```

Steps 3 & 4: Construct correlator .vex file

MIT HAYSTACK OBSERVATORY

- As the .vex file doesn't contain all the data we need to run DiFX we need to add VGOS specific setup data.
- We use a template file that contains information which changes infrequently.
- The template file should be updated whenever a new station joins, or the frequency/hardware setup at a site changes, etc.
- Now strip the useful bits out of the observational .vex file and combine it with the template.

```
cp ../prep/template.vex ./template.vex.tmp
source ./tweak_vex.sh ./vt9105.vex.obs ./vt9105.vex.corr
```

- Set correlator specific information; `exper_num`, `exper_nominal_start`, `exper_nominal_stop`.
- Prepare and insert the $EOP and $CLOCK sections.

Step 5: Check for CalcServer and $DIFX_MACHINES

MIT HAYSTACK OBSERVATORY

- If you have a static correlator configuration, make sure $DIFX_MACHINES is set and configured.
- A global machines file is not strictly necessary if the machine setup is defined in the .v2d file. This is the path we will follow in this exercise.

- Check that the $CALC_SERVER is setup and running:

```
echo $CALC_SERVER
checkCalcServer $CALC_SERVER
```

Step 6: Configure .v2d file

MIT HAYSTACK OBSERVATORY

- DiFX needs additional information to fully configure the correlation process.
- This is provided in the .v2d file.
- The .v2d file provides information detailing with:
- The job configuration:
  - vex file
  - start/stop times, singleScan
  - nChan, tInt
  - machines, nCore, nThread
- Additional antenna information:
  - data location (machine)
  - filelist
  - format/sampling, phaseCalInt (5MHz)
- Note that in our example .v2d file we've limited the time-range of the correlation to just the first two scans.

Step 7: vex2difx

MIT HAYSTACK OBSERVATORY

- Run vex2difx to generate the DiFX configuration for this correlation.

```
vex2difx h.v2d
```

- This will generate the DiFX .input, .machines, .threads, .flag files and a .joblist file.

Step 8: Run DiFX

MIT HAYSTACK OBSERVATORY

- Make sure you start up erromon2 in another window to monitor the messages and catch any errors.

```
errormon2 5
```

- Then go ahead and fire up DiFX with the joblist created by vex2difx:

```
startdifx -f -n h.joblist
```

Step 9: Convert to difx2mark4

MIT HAYSTACK OBSERVATORY

- Once DiFX is finished, we'll need to convert the raw DiFX output to the Mk4 data format for the post-processing stage.
- Make sure that there is a station-codes file present.
- Note that in contrast to SX, the visibility data is treated as a single band (X) for VGOS.

```
difx2mark4 -v -d -b X 2300 14000 -e 1234 -s station.codes
```

Step 10: Fourfit

MIT HAYSTACK OBSERVATORY

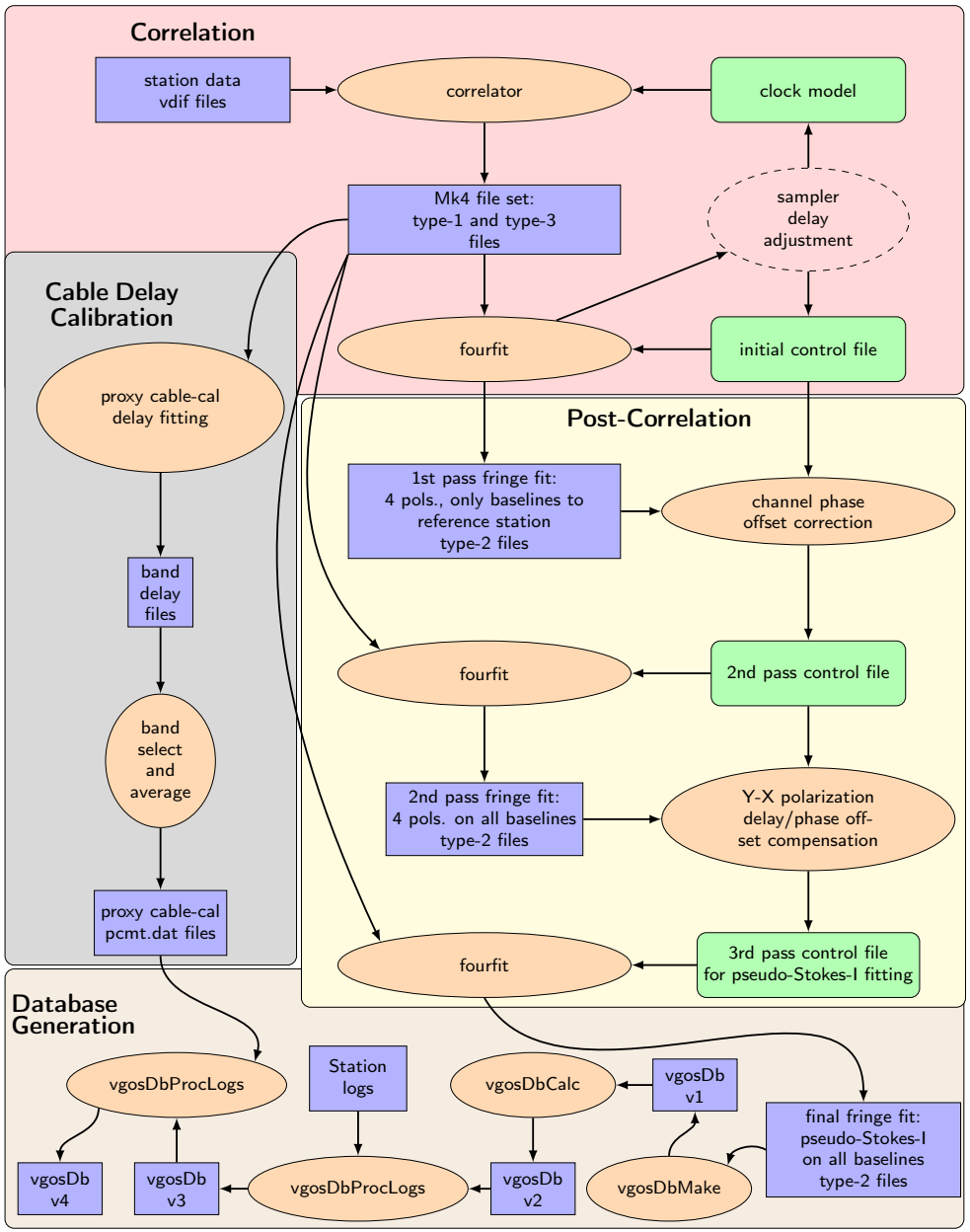
- Prepare an a-priori fourfit control file.
- For this purpose we can often just use the production control file from the last experiment.
- Now inspect the output with fourfit, for example:

```
fourfit -pt -c ./cf_3686_GEHSVY_pstokes -b GV -P I ./1234/105-1800
```

- Note that in the above example, the -PI parameter is specifying the pseudo Stokes-I fringe-fit.
- It is also important to examine the other individual polarization products XX, YY, XY, YX, when searching for possible issues with the data.

Post-correlation

MIT HAYSTACK OBSERVATORY



Step 11: Construct pseudo Stokes-I control file

MIT HAYSTACK OBSERVATORY

- For convince this is most easily done using the script `vgoscf_generate.py`.
- To run this, we need to specify the following arguments:
  - an initial control file
  - the network-reference station
  - list the remaining stations
  - data directory
- There are additional options to control data selection and behavior of the script.

```
vgoscf_generate.py -n 12 -p -w ./cf_1234_firstpass G EHSVY ./
```

- The above command will generate a pseudo Stokes-I control file for this experiment.
- Alternatively, we can also use the scripts `ffres2pcp.py` and then `fourphase.py` in series to perform the same task.
- This is sometimes necessary if there are various data issues with one or more stations, and hand editing of the intermediate control file is needed.
- The default behavior of these is to dump all output (except logs) into a scratch folder. This is to simplify clean up later.

Step 12: Quick quality check - `pc_phases_x/y`

MIT HAYSTACK OBSERVATORY

- Check the resulting channel-by-channel phase corrections for each station.
- Absent any hardware changes at a station, these typically do not change much from session to session.
- Typical changes are ($<$ 20 degrees).
- To generate a summary and associated plots, run:

```
summarize_report.py ./ffres2pcp-report-*
```

- Inspect the plots for each polarization of each station.
- Large deviations from a-priori values or error-bars indicate data issues which should be inspected in more detail.

Step 12: Quick quality check - `pc_delay_x/y` and `pc_phase_offset_x/y`

MIT HAYSTACK OBSERVATORY

- Also check the station Y-X polarization delay and phase offset corrections.
- Absent hardware changes, these typically do not vary much from session to session
- Changes to delays are normally (< 5 ps), and phase offset (< 5 degrees).

```
summarize_report.py ./fourphase-report-*
```

- Inspect the plots for each polarization of each station.
- Large deviations from the previous session or non-Gaussian distributions indicate data issues which should be inspected in more detail.

Step 13: Pseudo Stokes-I fringe fitting

MIT HAYSTACK OBSERVATORY

- Once we have a control file with the necessary phase and delay corrections, all data should be fringe-fit in pseudo Stokes-I mode.
- The convenience script `batch_fourfit.py` is provided for rudimentary parallelization on a single machine, but it is not optimized.
- Before running, ensure there are no left over fringe files generated by other control files!

```
batch_fourfit.py -n 12 -p ./cf_1234_GHEVSY_pstokes GHEVSY I ./
```

Step 14: Inspect the output

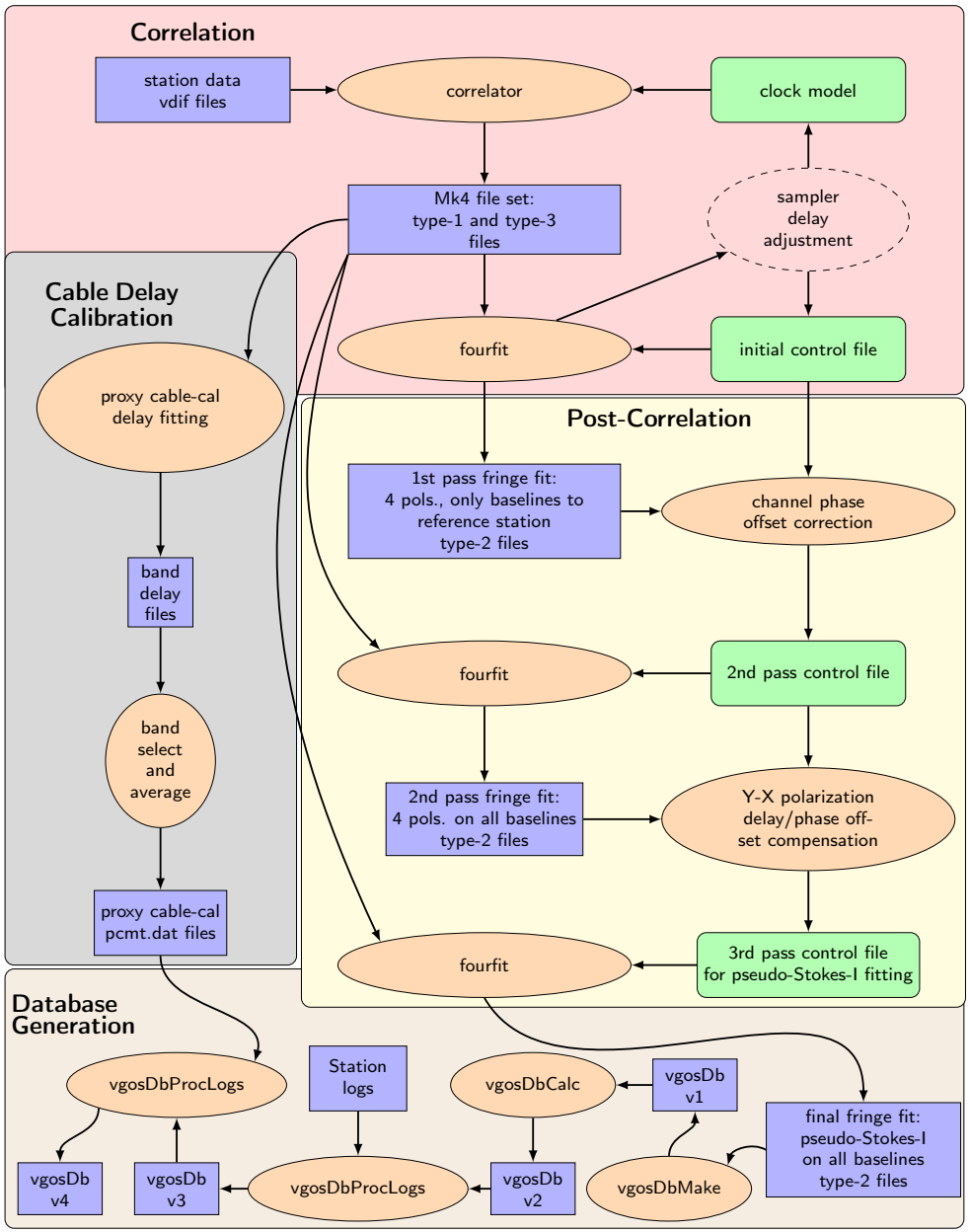
MIT HAYSTACK OBSERVATORY

- Once the pseudo Stokes-I fringe-fitting is done, we need to check the data quality.
- One simple but very helpful check, is to examine the residual phase of each channel over the duration of the experiment.
- These phase residual plots can be created with the script `phase_resid.py`.
- The following script will reuse any pre-existing fringe-files matching the given control file, when creating the plot.

```
phase_resid.py -n 12 -p -z dtec ./cf_1234_GHEVSY_pstokes GV I ./
```

Proxy cable-calibration

MIT HAYSTACK OBSERVATORY



Step 15: Generate proxy cable-calibration files

MIT HAYSTACK OBSERVATORY

- Some stations without hardware cable-calibration need a 'proxy' cable-delay calibration.
- This is estimated from the phase-calibration tone data.
- To generate the delay estimates for each station use the script `pcc_generate.py`.

```
pcc_generate.py -v 3 -f G ./
```

- Once complete we can review the proxy cable-delay trend plots which were generated.

Step 16: Run `pcc_select.py`

MIT HAYSTACK OBSERVATORY

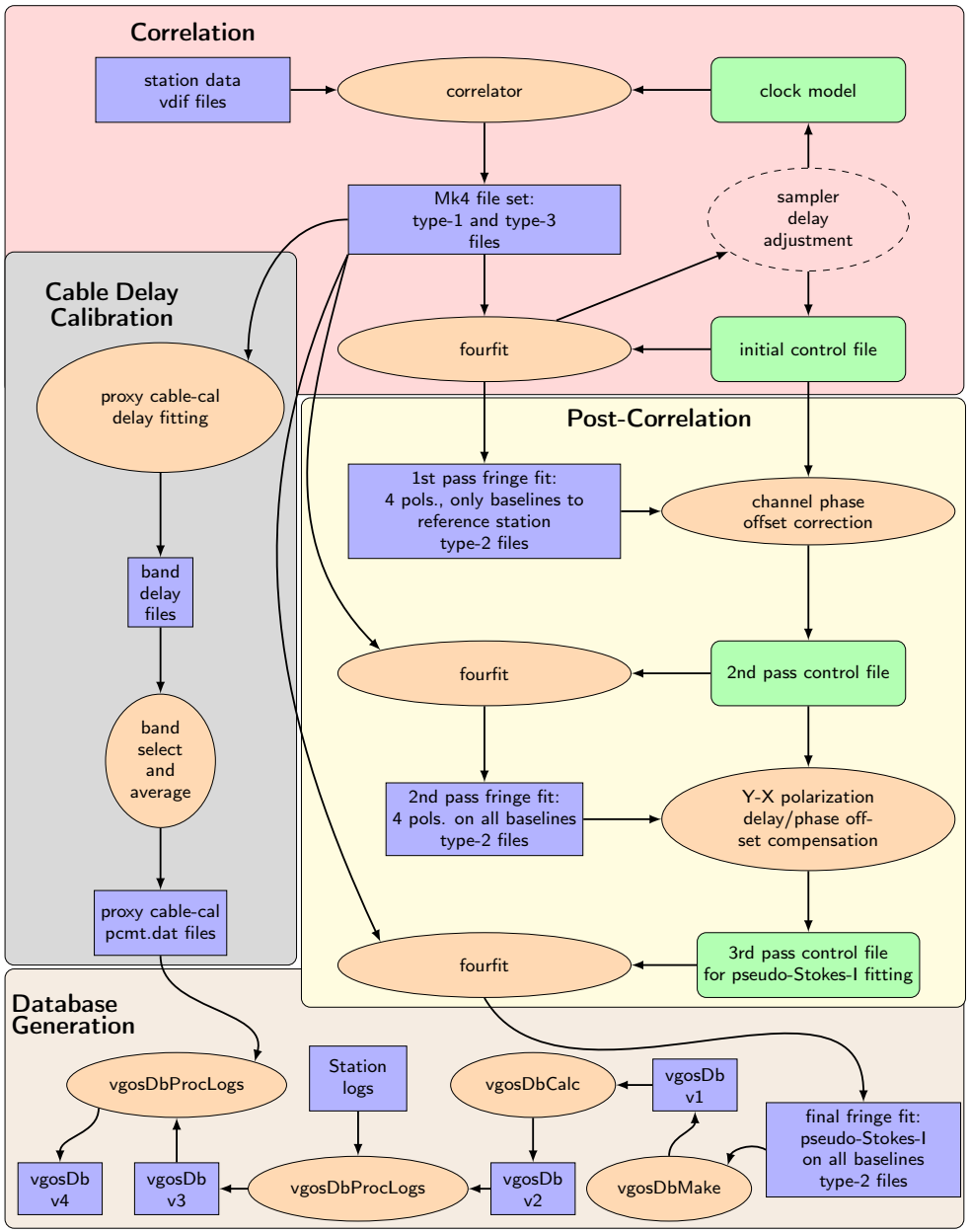
- Select the band-polarizations to use when computing the mean 'proxy' cable delay. The mean value for each scan will provide the cable-delay correction supplied to the database.
- In following example, we are selecting both polarizations (X,Y) and the three bands B,C, and D for station G.

```
pcc_select.py -e vt9105 -d ./ -s G:BCD:XY
```

- The output of the above command is the file `vt9105gs.pcmt.BCD.XY.dat`.
- The `.pcmt` files can then be digested by vgosDbProcLogs.

Database generation

MIT HAYSTACK OBSERVATORY



Step 17: Generate the vgosDb output

MIT HAYSTACK OBSERVATORY

- Once we have done the final fringe-fitting of a session, we can create a vgosDb database.
- This is quite straightforward with vgosDbMake.
- Appending a correlator report is standard practice and should created and attached at this time.

```
cd $VGOSDB
vgosDbMake -t $VGOSDB/vt9105.corr -d "19APR15VG" $TOWDATA
```

Step 18: Add delay model with vgosDbCalc

MIT HAYSTACK OBSERVATORY

- Then we need to insert the delay model information.
- This is also quite straightforward (just make sure the CALC server is running)
- vgosDbCalc operates on the version-1 database wrapper file.

```
vgosDbCalc ./19APR15VG/19APR15VG_V001_iMHC_kall.wrp
```

Step 19: Append the station log data

MIT HAYSTACK OBSERVATORY

- We need to add the station log data with vgosDbProcLogs.
- Retrieve the station logs if you haven't already, and place them in the appropriate folder.
- The first pass operates on the version-2 database wrapper file.

```
cd $STATION_LOGS
mkdir -p ./2019/vt9105 && cd ./2019/vt9105
wget ftp://cddis.gsfc.nasa.gov/vlbi/ivsdata/aux/2019/vt9105/*.log
cd $VGOSDB
vgosDbProcLogs -k log ./19APR15VG/19APR15VG_V002_iMHC_kall.wrp
```

- Next copy any .pcmt files you've generated into the station logs folder.
- Run vgosDbProcLogs a second time on the version-3 wrapper. This imports the proxy cable-cal. data for the station(s) which need it.
- Currently, this is typically done for the stations KOKEE12M, WETTZ13S, RAEGYEB, and GGAO12M.
- The following example is only GGAO12M.

```
cp $TOWDATA/pcc_datfiles/*pcmt* $STATION_LOGS/2019/vt9105/
vgosDbProcLogs -zc -s GGAO12M -k pcmt ./19APR15VG/19APR15VG_V003_iMHC_kall.wrp
```

Step 20: You're done!

MIT HAYSTACK OBSERVATORY

- At this point you have a database which is ready for geodetic analysis.
- Arrange to make this database available to the analysis center.

Things to note:

- While this demonstration was done in a linear way, this isn't necessarily typical.
- Some steps, such as setting clocks (which we have not touched on) are iterative.
- In addition, various issues with data may cause you to have to return to previous steps and fix things before proceeding.
- Checking the data quality at intermediate stages as you are going is essential.