VGOS Data Processing Manual

John Barrett Roger Cappallo Brian Corey Geoffrey Crew
Pedro Elosegui Arthur Niell Chester Ruszczyk
Mike Titus

MIT Haystack Observatory

Version 1.11

Jan 14, 2021

Contents

List of Figures

List of Tables

1 Introduction

The processing of VGOS experiment data brings several complications which are not normally dealt with in legacy-S/X experiments. These difficulties primarily stem from the need to handle the much larger bandwidth of the new stations and the dual-linear-polarization signal chain [?]. The intent of this document is to provide a general description of a process which transforms raw station baseband data into data products which may be used for geodetic analysis. It is not intended to be a completely authoritative guide covering all possible situations which might be encountered during VGOS operations, but rather to serve as a framework on which further developments may be based. Additional details describing a complete and operational VGOS system can be found in Niell et al.[?].

2 Correlation

Once the stations have recorded a VGOS session, the first step in the data processing pipeline is the correlation of the VLBI data. This correlation is done using the DiFX[2] software correlator [?]. Since general use and operation of the DiFX software has been documented extensively elsewhere[3], we will concentrate on details which are specific to correlating VGOS data. It should be noted that the following instructions are intended to guide and augment the correlation process when handling VGOS experiments. However, it is expected that individual correlator sites may have their own specific set of procedures and conventions for running a DiFX correlation that may be somewhat different than what is described here.

2.1 Data file preparation

For the purpose of this document we will assume the VGOS stations have provided data on Mark-6 recording modules in the native scatter gather format. To make this data available to the correlator, it must first be mounted on the Mark-6 playback units. To do this, first make sure that the playback units are configured to run `cplane` on boot-up but that `dplane` has been disabled. Then insert and key-on the station data modules while watching the disk activity indicator LEDs to check that each disk is mounted. To ensure that the disks have been appropriately recognized by the host-bus-adapter cards, an `mstat?` query can be made in cplane.

```
da-client
Host: 127.0.0.1 port: 14242
>> mstat?1
<<  !mstat?0:0:1:1:GSFC%041/48008/4/8:8:8:-:48008:unmounted:unprotected:sg;
```

The above output demonstrates that in this case all eight of the module disks have been recognized (the number of discovered disks is indicated by the red arrow). Once it has been confirmed that all disks of a module are present, they may be mounted from within `da-client`. To mount the data of a station which provides data on a single module that has been inserted into the first slot, open da-client and type: `group=mount:<slot number>`, as follows:

```
da-client
Host: 127.0.0.1 port: 14242
>> group=mount:1
<<  !group=0:0:1;
>> mstat?1
<<  !mstat?0:0:1:1:GSFC%041/48008/4/8:8:8:15476:48008:closed:unprotected:sg;
```

Proceed similarly for the rest of the station modules. Note multiple complete grouping of modules can be mounted from different stations in the same Mark-6. For example, if a single module was used for recording, a Mark-6 can have up to four stations mounted. For a station which provides data on multiple modules, simply specify the slot where its modules reside when issuing the mount command. For example, if the station modules reside in slots 3 and 4, one would use: `group=mount:34`.

[2] Version 2.5.2 or later is recommended.

[3] `https://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/documentation`

Once the data has been mounted, a file list for each station may be constructed using the utility `vsum`, which is provided with the DiFX package. For example, to construct a list of all files residing in slot #1 for station Gs, issue the following command.

```
stdbuf -o0 vsum -s --mark6slot 1 | tee ./gs_file.list
```

The file list can then be used in the .v2d file (needed in the next step) to specify the location of the data to be correlated.

Stations which provide data via e-transfer must have the data buffered locally before use. The exact format and procedure for this process depends heavily on the details of the individual station and correlator and should be determined on a case-by-case basis. The correlator should specify the expected format of the data for e-transfer by a particular site (which may depend on whether the data is for a single fringe test or standard processing). Standard format requires a station to convert data from multiple thread-IDs into a single thread-ID containing all four bands and both polarizations. For fringe tests, stations are expected to format their data into four files, with each file containing a single thread-ID (frequency band).

2.2 Vex file preparation

In order to process VGOS experiments using the DiFX correlator, the operator must prepare a correlator-specific .vex file describing the session and convert this .vex file to DiFX's native input format using the vex2difx utility.

An initial .vex file associated with the VGOS session to be correlated may be obtained from the VGOS master schedule on the IVS website[4]. However, the .vex files available on the IVS website are incomplete and are primarily used to provide the scan scheduling information for the observing stations. They do not yet contain valid information on the station frequency setup ($FREQ), intermediate frequency ($IF), baseband converter ($BBC), or tracks ($TRACKS) sections needed for correlation. These sections must be completed and added by the correlator operator.

Information to construct the $IF, $BBC, and $FREQ sections of the correlator .vex file are described in a memo on VGOS frequency band and channel configuration [?]. The nominal VGOS $IF, $BBC, and $FREQ, configurations are shown in the appendix in listings 7, 9, and 10, respectively. It should be noted that care should be taken to procure the frequency setup information directly from each station joining the VGOS network to ensure that it is compatible with the standard VGOS settings, as not all stations have similar front-end or back-end equipment. The $TRACKS section should also be replaced by the correlator operator. For the time being all that is needed for this section is a simple place holder declaring use of the VDIF format, as shown in listing 8. The $TRACKS section of the .vex file will later be overridden by the precise format specified for each station in the `vex2difx` configuration (.v2d) file as shown in listing 6.

2.2.1 Clock model generation and sampler delay adjustment

Prior to the generation of the correlator input files using vex2difx, the correlator operator must also set up the Earth orientation parameters ($EOP) and station clock ($CLOCK) sections of the .vex file.

The earth rotation and orientation parameters $EOP section should be filled in according to the description in the vex 1.5b1 standard[5]. Data to fill in this section can be obtained via the IERS rapid service[6]. At least five days of data should be specified about the date of the VGOS session to provide an adequate range for interpolation. An example of the $EOP section to be appended to the vex file is given in listing 5.

The setup of the .vex file $CLOCK section is somewhat more complicated because of the four separate frequency bands used in VGOS. Since these four bands may be down-converted

[4]https://ivscc.gsfc.nasa.gov/sessions/vgos/2018/

[5]https://vlbi.org/vex/docs/vex%20parameter%20tables%2015b1.pdf

[6]http://maia.usno.navy.mil/

through different signal paths and digitized by separate samplers, but must be fringe fit together, care must be taken when setting the station clocks. An extensive discussion of this procedure can be found in [?], and further details will be available in future revisions of this document.

2.3 DiFX input conversion and running the correlation

Before running the DiFX correlation, the experiment .vex file must be converted into something DiFX understands using the utility vex2difx. This program also requires a configuration (.v2d) file which is needed to specify additional parameters needed for the correlation. Normally for VGOS sessions, the .v2d setup section should specify the number of spectral channels to be 128, and the integration period as 1 second as shown below. Historically, the choice of 128 spectral channels has been motivated by the sometimes large sampler delays at individual stations, which reduces the SNR if the number of lags (i.e., the size of the Fourier transform) is not increased to compensate. It is expected that this value maybe relaxed in the future for more efficient correlation.

```
SETUP default
{
  nChan = 128
  tInt=1
}
```

An example .v2d file for a simple correlation done directly on a Mark-6 is shown in listing 6. Correlator operators should adjust the header section to specify the machines, threads, etc. which match their local environment. In addition to local setup, the correlator operator should also take care to fill in the ANTENNA section appropriately, indicating the data location, file-list, sampling type, and phase-calibration frequency interval (5 MHz in VGOS). For stations which are operated with RDBE-G back-ends, the sampling is complex with four threads, and the format must be specified using the VDIFC format tag. However, the format specified for each station in the .v2d file will depend on how the data were delivered by the station and should be dealt with on a case-by-case basis. For stations which have e-transfered files that have been gathered from multiple threads, or do not use RDBE-G back-ends and have recorded with real (rather than complex) sampling, the format specifier must necessarily be adjusted accordingly. Consult the station operator and/or the DiFX wiki entry on `vex2difx`[7] to determine the proper format specifier if it is unknown.

After `vex2difx` has been run to generate the DiFX input files the correlation can be started with the script `startdifx` with the appropriate options. For example, to run with a pre-constructed machines file and force the generation of output even if it already exists, one would run:

```
startdifx -f -M <machines file> <input file>
```

2.4 Running difx2mark4

Once the correlation is complete, the session should be converted to the Mk4 data format in preparation for post-processing with HOPS using the utility `difx2mark4`. This can be done from the directory containing the DiFX setup and Swinburne files (*.vex, *.im, *.input, *.difx/) following the instructions on the DiFX wiki [8]. The user should specify the four-digit experiment number to be associated with this session at the time of conversion and also indicate a station code file to map the two-character station codes to the single character Mk4 IDs. Since all four frequency bands are to be fringe fit together, they should be merged using the '-b' option to specify that the 'X' band covers the entire VGOS frequency range of 2.3 to 14 GHz. An example usage of difx2mark4 is given below:

```
difx2mark4 -v -d -b X 2300 14000 -e <4-digit experiment code> -s <station code file>
```

[7] `https://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/vex2difx`

[8] `https://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/difx/difx2mark4`

The current VGOS station identification file contains the following mapping for stations which have so far participated in VGOS experiments: GGAO (Gs), Kokee 12m (K2), Westford (Wf), Wettzell (Ws), Ishioka (Is), Yebes (Yj), Onsala-northeast (Oe), and Onsala-southwest (Ow).

```
G Gs
H K2
E Wf
V Ws
I Is
Y Yj
S Oe
T Ow
```

3 Fringe-fitting and Post-processing

After the correlation is complete, accurate estimates of the multi-band delay and delay-rate residuals must be made from the visibility data for input into the geodetic database. The process of determining these residuals is collectively known as fringe-fitting, and for VGOS data this is performed by the program `fourfit`, which is distributed with the software package HOPS. Additional utilities are also provided within HOPS which assist in constructing a `fourfit` control file to guide the fringe-fitting process.

3.1 Post-processing software installation

The post-processing software requires a computer with a Linux operating system with at least 4GB of RAM. Debian Stretch (9.7) is recommended, as the following procedure has been tested with a fresh installation of this distribution. While it is expected that other Linux/Unix distributions may also be satisfactory, they have not been extensively tested and instructions contained herein may need to be modified accordingly[9]. On Debian-9.7, the basic software dependencies required to build and verify the installation of HOPS can easily be installed via the package manager with the following commands.

```
sudo apt-get update
sudo apt-get install build-essential bc rsync pkg-config gfortran libfftw3-dev
sudo apt-get install ghostscript ghostscript-x libx11-dev
```

The dependencies of the VGOS-specific Python packages and scripts that are bundled with HOPS may be installed by issuing:

```
sudo apt-get install python-future python-matplotlib python-numpy python-scipy python-progress
```

The package `python-progress` is optional.

An additional requirement of HOPS is the program PGPlot (v5.2). This must be built and installed from source with a shared library made available. To retrieve the source code and install PGPlot to the canonical installation directory (`/usr/local/pgplot`), enter the following as the root user:

```
cd /usr/local/src/
wget ftp://ftp.astro.caltech.edu/pub/pgplot/pgplot5.2.tar.gz
tar -xzf ./pgplot5.2.tar.gz
mkdir /usr/local/pgplot
cd /usr/local/pgplot/
cp /usr/local/src/pgplot/drivers.list ./
```

Then edit the file `drivers.list` to indicate which device drivers you wish to build. At a minimum the resulting `drivers.list` file should contain the following (uncommented) lines, indicating that the PostScript and X-window drivers be built.

```
PSDRIV 1 /PS      PostScript printers, monochrome, landscape Std F77
PSDRIV 2 /VPS     Postscript printers, monochrome, portrait     Std F77
PSDRIV 3 /CPS     PostScript printers, color, landscape         Std F77
PSDRIV 4 /VCPS    PostScript printers, color, portrait          Std F77
XWDRIV 2 /XSERVE  Persistent window on X Window System          C
```

[9]For example, on Ubuntu 18.04, the default compiler gcc 7.3 produces broken shared libraries, and the default system installation of python is python3. To work around these differences on Ubuntu 18.04, install the clang compiler with `apt-get install clang`, and install the associated python pre-requisites using their python3-* equivalents. Then before the HOPS configure step, run `export CC=clang && export PYTHON=python3`.

Once this is done, generate the PGPlot make-file by running the following commands:

```
/usr/local/src/pgplot/makemake /usr/local/src/pgplot linux g77_gcc
sed -i -e 's/FCOMPL=g77/FCOMPL=gfortran/g' ./makefile
```

Next, build and install PGPlot and its associated libraries by executing:

```
make
make cpg
make clean
```

Finally, set the following environmental variables:

```
export PGPLOT_DIR=/usr/local/pgplot
export PGPLOT_DEV=/XSERVE
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/pgplot
```

If you wish to make these variables persistent across login sessions, add the above lines to the `.bashrc` script in your home directory.

Once the system prerequisites are satisfied, the HOPS software can be unpacked and installed as a normal user. To do this, retrieve and decompress the file `hops-3.21-2936.tar.gz`, create a build directory, and configure with the options specified below[10].

```
wget ftp://gemini.haystack.mit.edu/pub/hops/hops-3.21-2936.tar.gz
tar -xzf ./hops/hops-3.21-2936.tar.gz
mkdir build
cd build
../hops-3.21/configure --prefix=<path/to/install/location>
```

Then run `make` and `make install`:

```
make
make install
```

Finally, to use and validate the HOPS software, set the appropriate environmental variables and place the HOPS binaries in your path by running the HOPS setup script with the following:

```
source <path/to/install/location>/bin/hops.bash
```

Then to run a set of internal tests that verify that the installation was successful, run the following (from within the build directory):

```
export RUNCHOPSCHECK=1
make check
```

If the check passes, then the HOPS software has been compiled successfully. **There should be no alternate HOPS installation already in your path prior to running this setup script.** Once this is complete, you are ready to work with `fourfit` and the associated VGOS post-processing Python utilities.

3.2 Overview

The program `fourfit` transforms the visibility data into delay vs. delay rate space in order to search for the fringe peak; it can also simultaneously fit for the ionosphere differential total electron content (dTEC) and is responsible for applying the station multi-tone phase-calibration data to correct the raw visibility phases. `fourfit` accepts a control file which provides the user with an extensive number of configuration options that can be used to guide the fringe fitting. Since the process of constructing a control file that can appropriately direct `fourfit` to maximize the SNR on each baseline and minimize the error on the multi-band delay can be somewhat complicated (given the number of parameters involved), several auxiliary programs have been developed to aid in the configuration of the control file. In general, creating a well-calibrated `fourfit` control file is an iterative process which requires several steps before the final fringe fitting and database generation can be done.

The post-processing procedure can be broken down into the following steps:

1. Construct an initial `fourfit` control file using a priori data (e.g. the last experiment's control file).

[10]If you do not specify an installation prefix, then a default installation directory will be created at the same level as the `build` folder. However, the name of this directory will be system architecture dependent.

2. Generate a pseudo-Stokes-I fringe fitting `fourfit` control file:
   (a) Estimate the characteristic channel-by-channel phase offsets at each station with respect to the network reference station (typically Gs).
   (b) Estimate the phase and delay offsets between the X and Y polarizations at each station.
3. Perform session wide pseudo-Stokes-I fringe fitting and examine data quality.
4. Generate the proxy cable calibration files for stations which lack a dedicated cable delay measurement system and select appropriate bands/polarizations for inclusion in the vgosDb file.
5. Generate the VGOS database using vgosDBMake and append cable calibration corrections using vgosDbProcLogs.

The relation between these steps and the overall process is broadly summarized in the block diagram in figure 1.

To streamline the procedure of generating a control file for use in production pseudo-Stokes-I fringe fitting, several programs have been developed. These are `ffres2pcp.py`, `fourphase.py`, and `vgoscf_generate.py`. These auxiliary programs will be distributed with HOPS. However, they have not yet been incorporated into the standard release (currently HOPS v3.19) that is distributed with DiFX. Further information regarding the software release is forthcoming.

In typical operation, the input to this collection of programs is a complete session's worth of correlated data in Mk4 type-1 (corel) format along with an initial (a priori) `fourfit` control file. The program `ffres2pcp.py` is responsible for calculating the appropriate phase adjustments needed for individual channels, while `fourphase.py` is used to align the phases and delays between the two linear-polarization signal chains. The program `vgoscf_generate.py` is provided for convenience so that these two processes may be run together sequentially and is the normal means by which the user should generate a pseudo-Stoke-I mode control file for `fourfit`. By and large, `vgoscf_generate.py` completely automates the yellow (Post-Correlation) box shown in 1.

The final output of this process is a vgosDb with ionosphere-corrected and cable-delay corrected data.

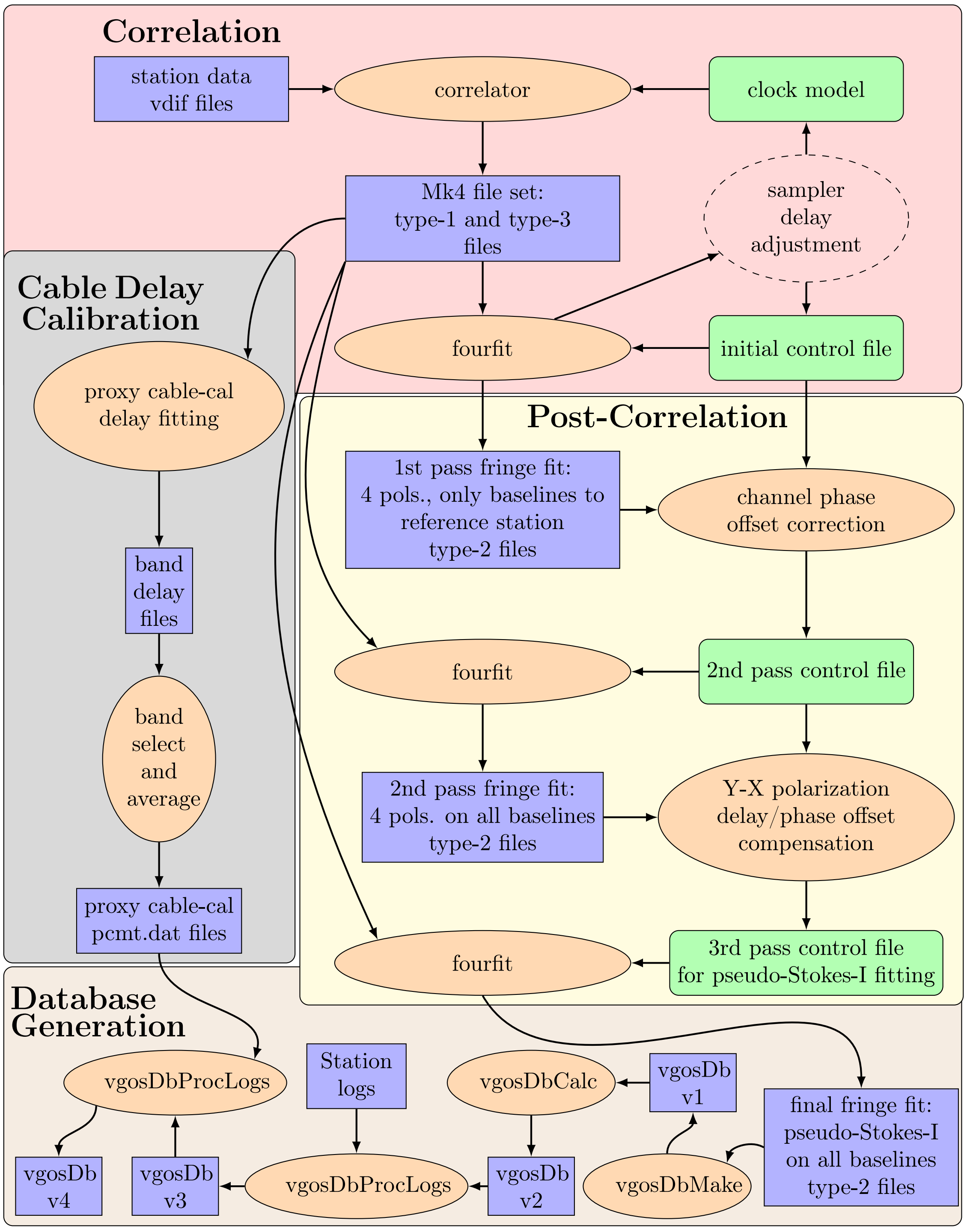
Correlation
station data vdif files
correlator
clock model
Mk4 file set: type-1 and type-3 files
sampler delay adjustment
Cable Delay Calibration
fourfit
initial control file
proxy cable-cal delay fitting
Post-Correlation
1st pass fringe fit: 4 pols., only baselines to reference station type-2 files
channel phase offset correction
band delay files
fourfit
2nd pass control file
band select and average
2nd pass fringe fit: 4 pols. on all baselines type-2 files
Y-X polarization delay/phase offset compensation
proxy cable-cal pcmt.dat files
fourfit
3rd pass control file for pseudo-Stokes-I fitting
Database Generation
vgosDbProcLogs
Station logs
vgosDbCalc
vgosDb v1
final fringe fit: pseudo-Stokes-I on all baselines type-2 files
vgosDb v4
vgosDb v3
vgosDbProcLogs
vgosDb v2
vgosDbMake

Figure 1: VGOS post-processing block diagram. The orange ellipses denote a program or script, green boxes are configuration files, and the blue boxes are data files. Dashed ellipses denote procedures which are currently not yet aided by automated scripting.

3.3 Initial control file

The initial control file should be provided as a best-guess estimate of the current state of the stations in the VGOS network. Ideally, it should provide estimates of the channel-by-channel phase offsets for each station, although this is not completely necessary, as these can be computed from the session data. However, at a minimum it should contain the basic VGOS fourfit configuration and station sampler delays; a complete example is given in the appendix, listing 4.

In general operation, when there have been no major hardware changes to any of the stations in the network, or no new stations joining the network, it is normally acceptable to reuse the production pseudo-Stokes-I control file from the most recent session as the a priori control file for the current session. Usually, the sampler delays at each station are fairly stable unless hardware changes have been made, so the sampler-delay adjustment procedure as described in [?] is not normally required. However if there has been any noticeable change, the sampler-delays will need to be corrected while constructing the clock model during the correlation step before proceeding.

When reusing the last session's control file, it is generally good practice to hand-edit the file in order to clean it up by stripping out old comments and removing any lines containing **pc_delay_offset_x/y** or **pc_phase_offset_x/y**. This is not strictly necessary when using the control file generation script `vgoscf_generate.py`, as it will comment out extraneous lines, but it helps to prevent the control files from becoming too cluttered.

3.4 Residual phase adjustment with `ffres2pcp.py`

To align the phases across all bands in preparation for combined polarization fourfit-ing, the characteristic phase residuals need to be removed across all channels for each station. This step is important because the non-linear phase structure in the response of the signal chain at each station (which remains uncorrected by the phase-calibration system) degrades the quality of the fringe-fit. This is because the phase model used by `fourfit` expects behavior which is linear in delay and delay rate, along with a single dTEC term which is inversely proportional to frequency. Any residual phase structure which cannot be compensated by this model not only reduces the fringe amplitude (and corresponding SNR) of whatever solution is found, but can also cause the appearance of multiple competing peaks within the (delay, delay-rate, dTEC) search space. When competing solutions have similar fringe amplitudes, random noise can push the fitting algorithm toward a particular solution in an arbitrary way and makes consistent determination of the dTEC more difficult from scan to scan, increasing the scatter in the multi-band delay measurements. Although the long-term behavior of these phase non-linearities demonstrates that they are relatively stable (changes are usually $< 20°$) over months-to-year-long time frames, the phase corrections should be re-calculated on a per-session basis to optimize SNR and to monitor station health.

Since absolute phase calibration isn't necessary, we choose a single station to serve as a phase reference for the other stations in the network. We will refer to this station as the network reference station. The choice is somewhat ad-hoc but is best served by a station which has a fairly flat bandpass and is also present in as many scans a possible. For this purpose, the X polarization feed at Goddard (GGAO) has been used as the network reference station for the VGOS network so far.

The determination of the residual phase adjustments for each channel is ideally done using many scans which have high SNR fringes on all four polarization products (XX, YY, XY, YX). Moreover, it is also important that the dTEC solution for each polarization product differs very little (< 1.0 TEC unit) from the mean dTEC. The process of calculating these phase adjustments is done using a script called `ffres2pcp.py`. This script is responsible for running `fourfit` where required, loading the resulting fringe files, locating optimal scans, and computing the mean phase residual corrections to be applied. The input to this script is the experiment Mk4 type-1 (corel) files and an initial (a priori) control file. The output of this script is a new control file which specifies the phase corrections (indicated by the keyword

pc_phases_x/y in the fourfit control file) to be applied to each channel at each station.

Running `ffres2pcp.py` is fairly straightforward and has only four required arguments. These are the initial control file to be used, the single-character Mk4 station code for the network reference station, a concatenated list of the remaining non-reference stations, and the path to the directory containing the Mk4 type-1 (corel) files to be processed. For example, the command shown below:

```
ffres2pcp.py ./cf_3659_firstpass G HEVI ./3659/
```

will run the ffres2pcp.py procedure for experiment 3659 using station G (X-polarization) as the network reference station, and the file `cf_3659_firstpass` as the initial control file. It will output a new control file (in this case, by default named `cf_3659_GEHIV_pcphases`) which will consist of the initial control appended with appropriate phase correction lines for the X and Y polarizations of stations E, H, I, V, as well as the Y polarization of station G. Currently, regardless of which station is specified as the network reference station, the X polarization of this station will be used as the phase reference. It should be noted that this is an arbitrary restriction which may be lifted in the future if needed.

There are several optional arguments which may be specified to modify the default behavior of `ffres2pcp.py`. Information on the optional arguments can be obtained at any time by calling `ffres2pcp.py -h` at the command line which displays the usage information. Further discussion of the configuration options can be found in section 3.7.

In order to keep track of the scans which were selected to be included in the mean phase corrections and to record how the phase corrections were calculated, this script also produces a report file in .json[11] format. This report file can be processed using the script `summarize_report.py` to generate a simple text summary and several plots to inform the user. The text summary consists of a table listing the a priori phase corrections, the new phase corrections, and the estimated error (standard deviation) of the phase corrections for each station and polarization. As an example, table 1 displays a portion of the data statistics for the X-polarization of station E from experiment #3659.

channel	a	b	c	d	e	f	g	h	···	y	z	A	B	C	D	E	F
a priori pc_phase	-4.4	-2.1	-0.1	-6.4	-1.7	10.1	17.6	19.5	···	18.8	13.3	12.1	5.1	6.2	7.2	13	14.6
delta pc_phase	-1.7	-0.3	1.5	-1.4	-0.2	-0.4	-0.6	1.7	···	1.3	-2.7	-2.7	-0.9	4.2	-1.6	0.6	4
result pc_phase	-6.1	-2.4	1.4	-7.8	-1.9	9.7	17	21.2	···	20.1	10.6	9.4	4.2	10.4	5.6	13.6	18.6
error	1.9	3	1.9	1.7	2.6	2.4	2.5	3.6	···	2.7	3.1	4.6	4.2	6.3	2.9	3.5	3.8

Table 1: An example data statistics table[12] for the channel-by-channel phase corrections for station E, X-polarization of experiment #3659. This table lists (in degrees) the a priori pc_phase, the difference (delta), resulting pc_phase, and estimated error (taken to be the std. dev.) calculated for this session for each of the 32 channels. The channels are referenced according to the `fourfit` channel label (e.g. a, b, ...F).

In addition, two types of plots will be generated for each station and polarization. These are a graphical display of the phase corrections as shown in figure 2, and a scatter plot of selected scans. The scatter plot displays the location (minimum SNR, maximum dTEC deviation) of each scan considered or chosen for inclusion in the calculation of the mean phase corrections. Those scans which were considered (remaining after the rough initial cuts) are shown as black dots, while those which were actually included in the mean are indicated by blue stars, as shown in figure 3.

[11] JavaScript object notation.

[12] Re-formatted from original to fit this page.

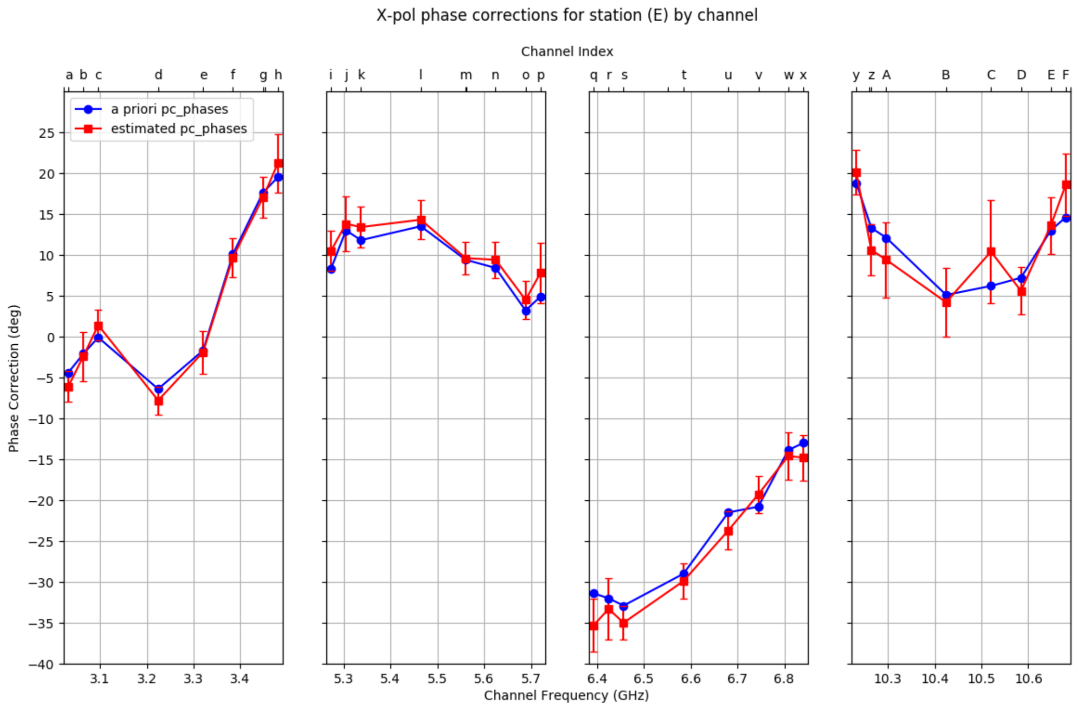

Figure 2: Example of the phase corrections for station E, X-polarization. The blue markers represent the a priori (expected) phase correction as estimated from the last session, while the red markers denote the newly computed phase corrections generated from the current session (vt8218). The top axis shows the `fourfit` channel labels, while the bottom displays the frequency in GHz.

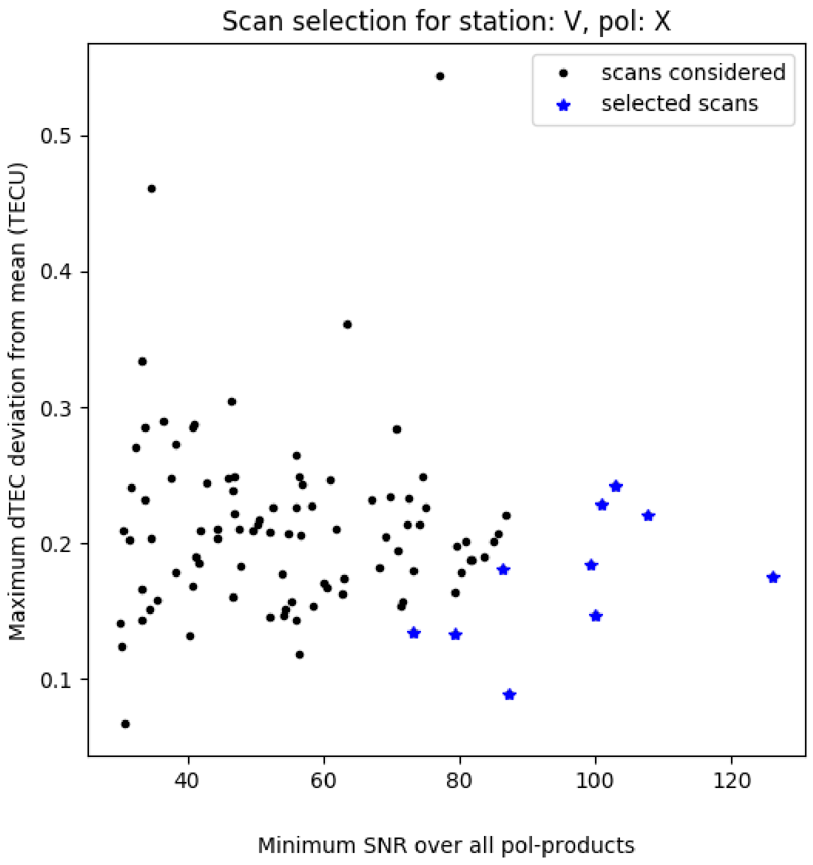

Figure 3: Plot of minimum SNR vs. maximum dTEC deviation from the mean (taken over all polarization products) for candidate scans on the GV baseline. The black dots represent the scans which passed the preliminary cuts and were considered; the blue stars indicate scans which were chosen to be included in the calculation of the mean channel-by-channel phase corrections. The selected scans are picked by attempting to maximize the minimum SNR found over the fringe solution of all four polarization products, while minimizing the maximum deviation from the mean of the dTEC solution among them. When there is one scan which satisfies this criterion, it is selected first, and then removed from the set of scans to consider. Then the best scan is found in the remaining scan set and so forth, up to some limit (the default limit is 10 scans). In the case where there is no single optimal scan, each scan is given a score S, and the scan with the maximum score is selected. This score is given by $S = min(\text{SNR})/mad(\text{dTEC}) + \eta$, where the minimum $min(\text{SNR})$ and the maximum-absolute-deviation $mad(\text{dTEC})$ are taken over the fringe solution of all four polarization products, and η has been empirically determined to be 0.1 TECU. It should be noted that this selection method is used to limit the number of scans only when the option (averaging-scan-limit) is active. If this option is disabled, all scans which pass the preliminary cuts on SNR and dTEC deviation will be used.

3.5 Polarization phase/delay offset calibration with `fourphase.py`

Before pseudo-Stokes-I mode fringe fitting can be performed, the phase and delay offsets between the X and Y polarizations must also be determined. This is necessary in order to coherently add the visibilities and maximize the SNR in the resulting combined fringe. These offsets can only be determined after the channel-by-channel phases adjustments have been made because of the difficulties an un-corrected station bandpass can cause when fitting the dTEC.

The script `fourphase.py` is used to compute the overall delay and phase offsets between the X and Y polarizations of each station. This script takes the same arguments as `ffres2pcp.py`, with the exception that the control file to be used should be the control file which is the results of running `ffres2pcp.py`. It is called as in the following example:

```
fourphase.py ./cf_3659_pcphases G HEVI ./3659/
```

which will run the phase/delay offset extraction for the stations G, H, E, V, and I for experiment #3659 using the control generated by `ffres2pcp.py`. This script also supports several options which are described in section 3.7. A .json report file is also produced by `fourphase.py` which can be processed with `summarize_report.py` to provide the user with feedback on how the calculation was performed. The text output produced from the .json file consists of a summary of the configuration used to run `fourphase.py` as well as a brief table of the data statistics, an example of which is shown in table 2.

station	N total	N used	N cut	mean delay (ns)	delay error (ns)	median delay(ns)	median abs deviation (ns)	mean phase (deg)	phase error (deg)
I	146	137	9	-0.064	0.003	-0.064	0.003	41.203	5.32
H	168	161	7	0.143	0.003	0.143	0.003	-50.661	4.287
E	194	179	15	0.722	0.003	0.722	0.003	117.547	6.091
G	120	111	9	1.681	0.003	1.681	0.002	30.962	3.857
V	196	177	19	-0.059	0.002	-0.059	0.002	-132.253	4.955

Table 2: An example data statistics table (re-formatted) on station Y-X phase and delay offsets extracted from `fourphase.py` report. For each station, the number of scan-baselines (total, used, cut) when computing the Y-X phase/delay offsets is shown in the first three columns, respectively, followed by the mean phase and delay offset values and errors (estimated as the std. dev).

In addition to the data statistics table, a histogram of the calculated Y-X phase and delay offset corrections for each station will be created as shown in figure 4. Also indicated on these histograms are a Gaussian fit, the resulting mean value, and the mean value calculated as if no outlier values were cut.

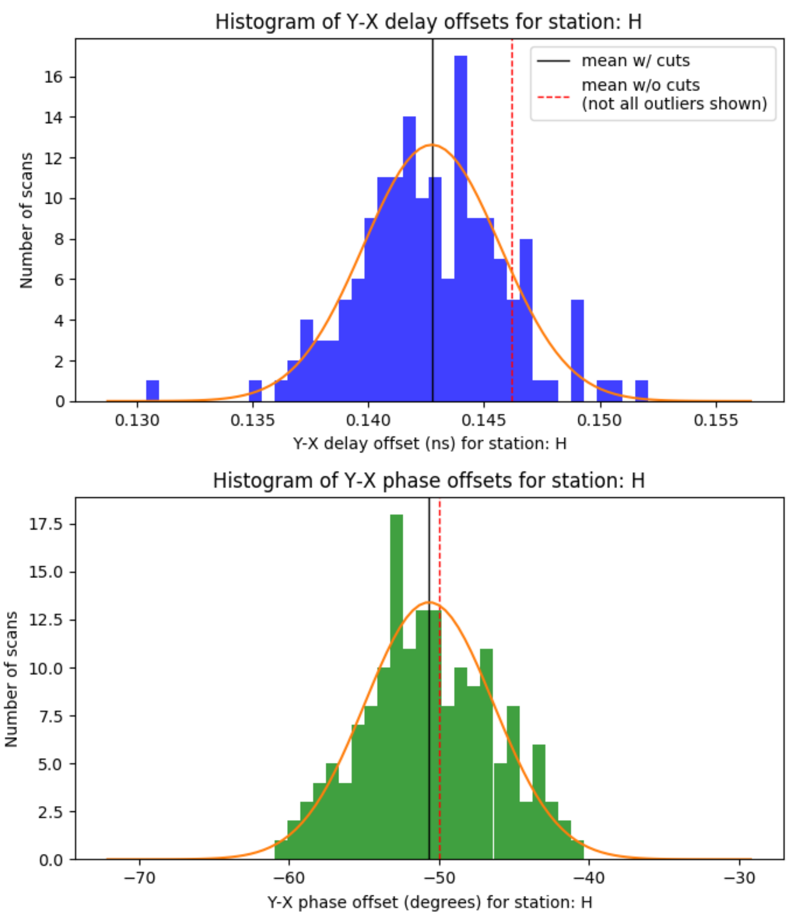

Figure 4: Histogram of the Y-X delay and phase offsets for station H, experiment #3659.

3.6 Generating a pseudo-Stokes-I mode control file with `vgoscf_generate.py`

In the course of processing a normal VGOS experiment, the script `ffres2pcp.py` and the script `fourphase.py` are not typically run individually. Instead they are run in combination though the convenience script `vgoscf_generate.py`. This script takes the options and arguments described in section 3.7, with the usage description displayed by the `-help` option shown in listing 1. The output is the final control file suitable for pseudo-Stokes-I fringe fitting. During the course of executing this script, two .json report files will be generated, one for the `ffres2pcp` process and another for the `fourphase.py` process. These report files may be individually digested by the script `summarize_report.py` to produce summary reports and plots to inform the user as described previously. Note that the default behavior of these scripts is to place the generated control files (and fringe-files) underneath a *scratch* folder[13] within the current experiment directory. Once the desired control file has been copied out of the *scratch* directory, the entire directory may be removed. This is done in order simplify the removal of extraneous fringe files from the experiment directory, as the intermediate fringe files created during this process should not be incorporated into the experiment database. However, if the user desires, this default behavior can be disabled such that the control files and intermediate

[13]Each time one of these scripts is run a new directory will be created under the *scratch* folder, where the results will be placed. The name of this directory is generated based on the current time in order to avoid reuse.

option	short hand	parameter type	Description
--num-proc	-n	integer	Maximum number of concurrent fourfit processes to run
--snr-min	-s	float	Minimal allowable SNR for any selected scan-baseline
--quality-limit	-q	integer	Minimal allowable quality code for any selected scan-baseline
--dtec-threshold	-d	float	Maximum allowable difference between dTEC solutions for different polarization products on the same scan-baseline
--begin-scan	-b	DDD-HHMM	Limit scans under consideration to be after this scan (specified by day-of-year, hour and minute)
--end-scan	-e	DDD-HHMM	Limit scans under consideration to be before this scan (specified by day-of-year, hour and minute)
--cut-sigma-factor	-c	float	Cut phase/delay residual values which are more than a multiple of the standard deviation from the mean
--without-scratch	-w	N/A	Allow fourfit processes to create files directly in the experiment directory instead of a scratch folder
--progress	-p	N/A	Monitor progress with simple command line display
--output-filename	-o	string	Override the default name for the generated control file
--averaging-scan-limit	-a	integer	Limit the number of scans used when computing the mean phase/delay corrections
--toggle-run-info	-t	N/A	Disable the addition of information about how this program was called to the control file

Table 3: Description of available configuration options for control file generation scripts.

fringe files will be place within the current experiment directory.

3.7 Configuration options of `ffres2pcp.py`, `fourphase.py`, and `vgoscf_generate.py`

Collectively, the programs `ffres2pcp.py`, `fourphase.py`, and `vgoscf_generate.py` all take the same arguments and options. The required arguments are the control file to be modified, the network reference station code, a list of the other non-reference stations, and the experiment data directory containing the Mk4 type-1 (corel) and/or type-2 (fringe) files. The additional options are described in table 3.

Listing 1: Arguments and options of the script vgoscf_generate.py

```
usage: vgoscf_generate.py [-h] [-v VERBOSITY] [-n NUM_PROC] [-s SNR_MIN]
                          [-q QUALITY_LOWER_LIMIT] [-d DTEC_THRESH]
                          [-b BEGIN_SCAN_LIMIT] [-e END_SCAN_LIMIT]
                          [-c SIGMA_CUT_FACTOR] [-w] [-p] [-o OUTPUT_FILENAME]
                          [-a AVERAGING_SCAN_LIMIT] [-t]
                          control_file network_reference_station stations
                          data_directory

utility for constructing a control file suitable for pseudo-Stokes-I fringe
fitting of VGOS sessions

positional arguments:
  control_file        the control file to be applied to all scans
  network_reference_station
                      single character code of station used as network
                      reference
  stations            concatenated string of single codes of non-network-
                      reference stations of interest
  data_directory      relative path to directory containing experiment or
                      scan data

optional arguments:
  -h, --help          show this help message and exit
  -v VERBOSITY, --verbosity VERBOSITY
                      verbosity level: 0 (least verbose) to 3 (most
                      verbose), default=2.
  -n NUM_PROC, --num-proc NUM_PROC
                      number of concurrent fourfit jobs to run, default=1
  -s SNR_MIN, --snr-min SNR_MIN
                      set minimum allowed snr threshold, default=30.
  -q QUALITY_LOWER_LIMIT, --quality-limit QUALITY_LOWER_LIMIT
                      set the lower limit on fringe quality (inclusive),
                      default=6.
  -d DTEC_THRESH, --dtec-threshold DTEC_THRESH
                      set maximum allowed difference in dTEC, default=1.0
  -b BEGIN_SCAN_LIMIT, --begin-scan BEGIN_SCAN_LIMIT
                      limit the earliest scan to be used in the calibration,
                      e.g. 244-1719
  -e END_SCAN_LIMIT, --end-scan END_SCAN_LIMIT
                      limit the latest scan to be used in the calibration,
                      e.g. 244-2345
  -c SIGMA_CUT_FACTOR, --cut-sigma-factor SIGMA_CUT_FACTOR
                      cut phase/delay values which are more than
                      cut_factor*sigma away from the mean value, default=3.0
                      (use 0.0 to disable).
  -w, --without-scratch
                      disable use of scratch directory and work directly in
                      experiment directory
  -p, --progress      monitor process with progress indicator
  -o OUTPUT_FILENAME, --output-filename OUTPUT_FILENAME
                      specify the name of the generated control file
  -a AVERAGING_SCAN_LIMIT, --averaging-scan-limit AVERAGING_SCAN_LIMIT
                      limit the number of scans used in averaging, use 0 to
                      disable, default=10
  -t, --toggle-run-info
                      do not append control file with information about how
                      this program was called
```

3.8 Pseudo-Stokes-I polarization fringe-fitting

Once a control file suitable for pseudo-Stokes-I mode fringe fitting has been generated, the entire experiment directory may be fringe-fit using the utility `batch_fourfit.py`. This script is a simple wrapper around `fourfit` which allows several processes to run simultaneously in order to expedite processing. It provides a modicum of parallelization by simply spawning several processes at once until all scans and baselines have been fringe fit. However, it is restricted to run on a single computer and can therefore provide only a small speed-up factor limited by the number of CPU cores and memory of the workstation.

By default, this script will fringe fit every scan within the directory specified, for every baseline possible given the list of stations (except for auto-correlations unless otherwise indicated). The behavior of this script can be modified following the usage options which are given below.

```
usage: batch_fourfit.py [-h] [-n NUM_PROC] [-b BEGIN_SCAN_LIMIT]
                        [-e END_SCAN_LIMIT] [-a] [-d] [-p]
                        control_file stations pol_products
                        experiment_directory

simple utility for running fourfit over multiple scans with the same control
file

positional arguments:
  control_file        the control file to be applied to all scans
  stations            concatenated string of single character codes for all
                      stations to be fringe fit
  pol_products        comma separated list of polarization-products to be
                      fringe fit
  experiment_directory relative path to directory containing experiment data

optional arguments:
  -h, --help          show this help message and exit
  -n NUM_PROC, --num-proc NUM_PROC
                      number of concurrent fourfit jobs to run, default=1
  -b BEGIN_SCAN_LIMIT, --begin-scan BEGIN_SCAN_LIMIT
                      limit the earliest scan to be used e.g 244-1719
  -e END_SCAN_LIMIT, --end-scan END_SCAN_LIMIT
                      limit the latest scan to be used, e.g. 244-2345
  -a, --auto-corrs    enable auto-correlations
  -d, --disable-cf-caching
                      disable type_222 control file record caching in fringe
                      files (caching is on by default).
  -p, --progress      monitor process with progress indicator
```

After pseudo-Stoke-I mode fringe fitting is completed, some sense of the raw data quality can be obtained through the use of the utility `phase_resid.py`. This program plots the channel phase residuals (with respect to the overall fringe phase) as a function of time for all baselines and scans in the experiment. An example plot of the channel-by-channel fringe phases (in this case, colored by the dTEC solution) is shown in figure 5. While the quantity of information in the plots generated by `phase_resid.py` can be quite large (nominally 32 plots per baseline for pseudo-Stoke-I mode alone), in some cases problems (which can otherwise be somewhat difficult to appreciate in individual fourfit plots) can be spotted quite easily by sharp jumps or splits in the phase residuals of one or more channels.

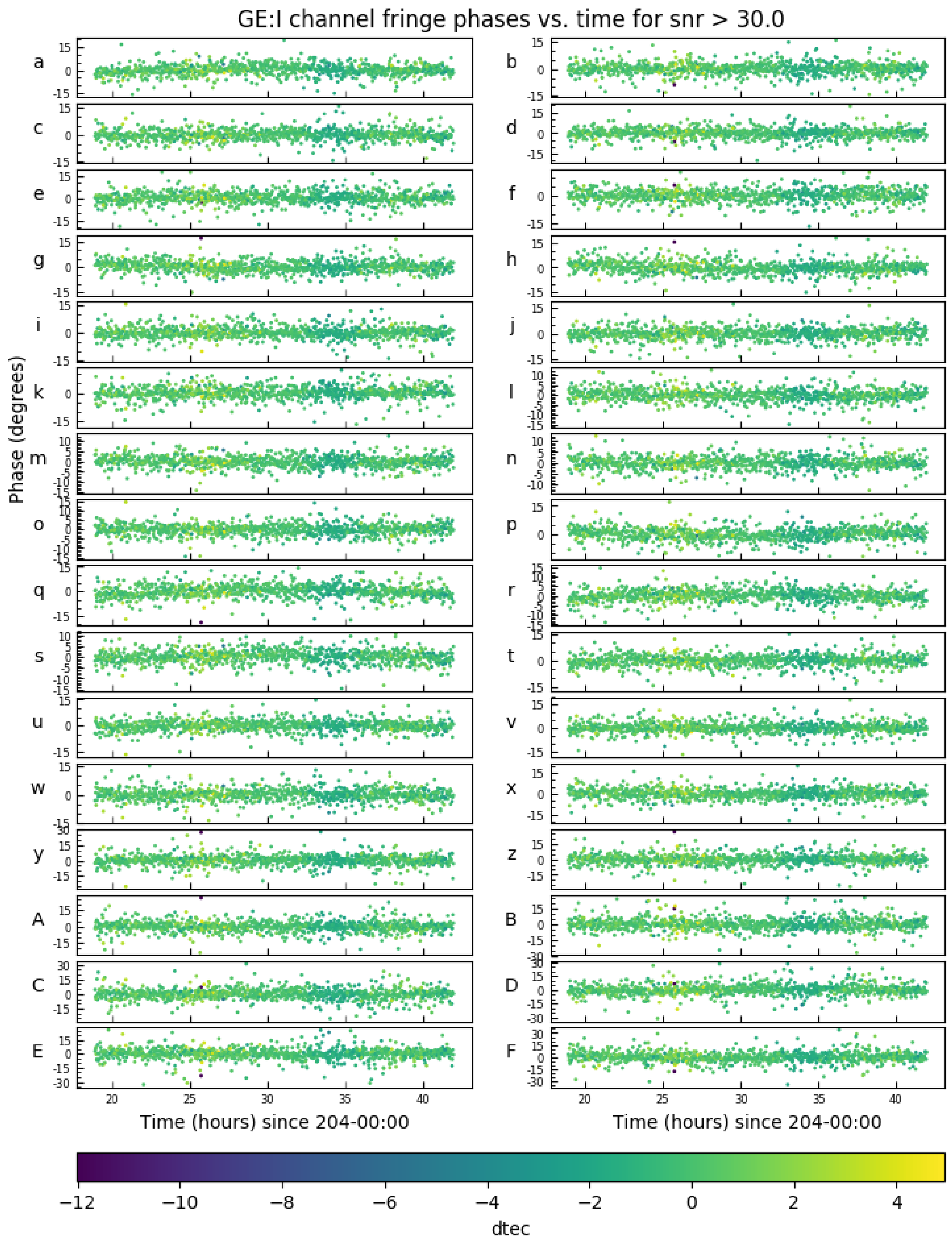

Figure 5: Channel-by-channel phase residuals for all scans with SNR > 30 on GE baseline for session vt8204. Color-coded according to the baseline dTEC.

3.9 Proxy cable calibration

Before the geodetic database is created, any available (non-station-log-file) cable calibration data should be collected first. For stations which do not have a dedicated cable delay measurement system (CDMS), a stop-gap proxy cable calibration method has been developed to provide an estimated cable delay correction. This method relies on the injected phase calibration tones to extract delays for the signal chain of each band. Individually, the proxy-cable delays as calculated by this method for each band contain the contribution of the reference clock uplink-cable delay (which we would like to measure) in addition to the delay due to the signal chain between the phase-calibration injection point and each individual sampler. Therefore, an average of the individual band delays is taken on a per-scan basis to provide an estimate of the antenna-orientation-dependent delay variation of the uplink-cable supplying the reference signal to the phase calibration generator.

To process the phase calibration data available in the station data (Mk4 type-3) files and calculate the proxy cable delays, the script `pcc_generate.py` may be used. The output of this script is a simple text (.dat) file containing the calculated differential delay for each scan for each band and polarization. In addition to these .dat files, if the `-f, --figures` option is specified, a plot will be generated for each band-polarization that has been fit, as well as a summary plot for each station. The individual band-polarization plots show the time trend of the fit solution's delay values, the phase at DC, and phase RMS. An example is shown in figure 6. The summary plot for each station shows the time trend of the delay for each band and polarization over the course of the session along with the mean delay value of bands BCD[14]. An example summary plot is shown in figure 7. It should be noted that extracting the short-time behavior of the proxy-cable delay is the main objective, while the overall general trend of the delay during a session is of relatively low importance because it is expected to be removed by the clock variation model during geodetic analysis.

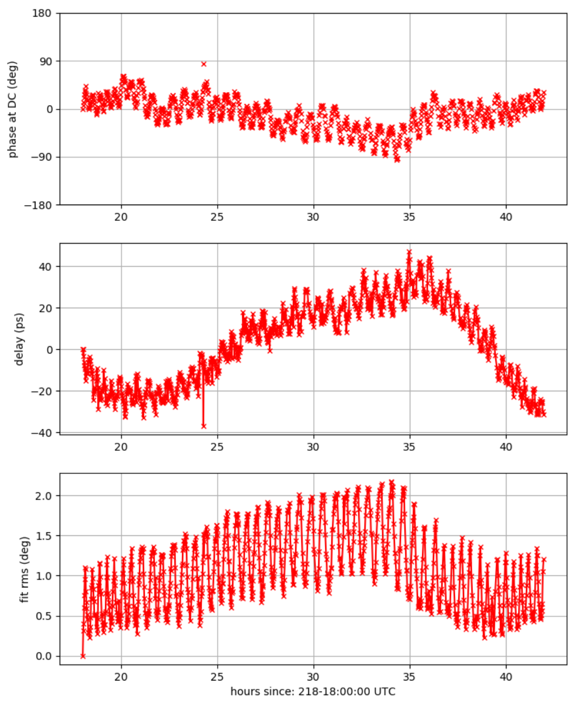

Figure 6: The band B, Y-polarization, multitone proxy delay fit data for station E, experiment vt8218.

[14]Note that band A is excluded from the mean by default because at several stations it is transported from the front-end to the sampler over co-axial cable rather than optical fiber unlike the other three bands.

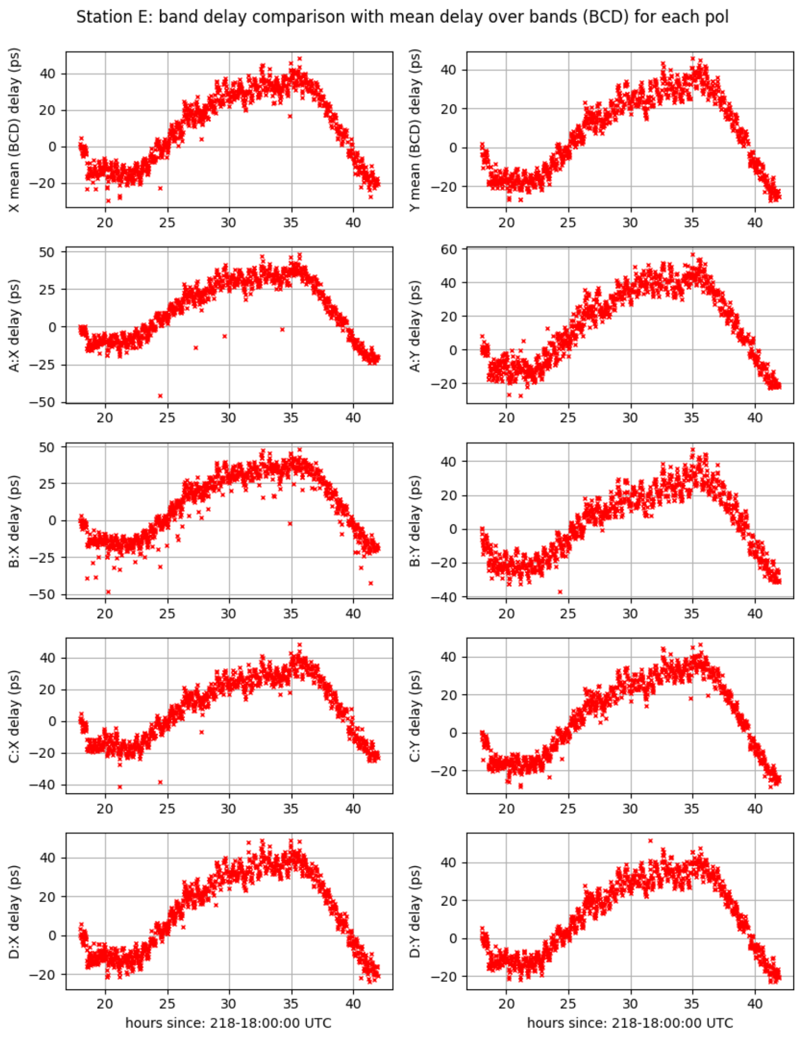

Figure 7: The summary plot for station E, displaying the mean delay of bands (BCD) for both polarizations along with the individual delay trends for each band-polarization, for experiment vt8218.

3.9.1 Running `pcc_generate.py`

The usage and options for this script are shown in listing 2. The default behavior of this script is to process all data for the listed stations in the given directory, and it may be run as soon as the station data files are available (after `difx2mark4` is run), even if the fringe fitting is not yet complete. For each station the earliest scan available is used as the session phase reference. The phase calibration phases of subsequent scans are then differenced with respect to this reference scan and a linear delay function is fit to the residual phases of each band (A, B, C, D) and polarization (X, Y) separately. In the example below:

```
pcc_generate.py -o ./pcc_test -b B,C,D -p X,Y -e -f -v 3 GEH ./
```

the command issued will calculate the delays for bands B, C, and D, and both polarizations for stations G, E and H. The station data files will be taken from the current directory and output will be placed in `./pcc_test`. The options `-e -f -v 3` indicate that progress will be estimated, plots will be generated, and verbosity will be set to the highest level. If no bands and polarizations are specified, then all available will be fit.

Listing 2: Usage and arguments for the script `pcc_generate.py`

```
usage: pcc_generate.py [-h] [-o ODIR] [-v VERBOSITY] [-f] [-t TRIM_LENGTH]
                       [-d DIAGNOSTICS] [-r REF_SCAN] [-b BANDS] [-p POLS]
                       [-i] [-y] [-n] [-e] [-c SIGMA_CUT_FACTOR]
                       stations data_directory

utility for generating proxy-cable calibration delays from phase-cal tones

positional arguments:
  stations              concatenated string of single codes of stations of
                        interest
  data_directory        relative path to directory containing experiment or
                        scan data

optional arguments:
  -h, --help            show this help message and exit
  -o ODIR, --outputdir ODIR
                        set output directory name, overrides default location:
                        <exp_dir>/pcc_datfiles.
  -v VERBOSITY, --verbosity VERBOSITY
                        verbosity level: 0 (least verbose) to 3 (most
                        verbose), default=0.
  -f, --figures         enable standard figures, default=False.
  -t TRIM_LENGTH, --trim-length TRIM_LENGTH
                        length of time to trim from the start of each scan
                        (seconds), default=2.
  -d DIAGNOSTICS, --diagnostics DIAGNOSTICS
                        enable p-cal (1) and fit (2) diagnostics plots,
                        default=0 (disabled)
  -r REF_SCAN, --reference-scan REF_SCAN
                        specify phase reference scan (defaults to first scan)
  -b BANDS, --bands BANDS
                        list of frequency bands to process: e.g.
                        --bands=A,B,C,D default is all.
  -p POLS, --pols POLS list of polarizations to include: e.g. --pols=X,Y,
                        default=X,Y default is all
  -i, --include-headers
                        include a header line in the dat files, default=False.
  -y, --yes             reply yes to all prompts, default=False.
  -n, --no              reply no to all prompts, default=False.
  -e, --estimate-progress
                        estimate progress, default=False.
  -c SIGMA_CUT_FACTOR, --cut-sigma-factor SIGMA_CUT_FACTOR
                        cut tones with residual phase values which are more
                        than cut_factor*sigma, default=3.0 (use 0.0 to
                        disable).
```

3.9.2 Running `pcc_select.py`

As generated, the .dat files produced by `pcc_generate.py` are not suitable to be directly incorporated into the database via `vgosDbProcLogs`. They must first be selected and averaged by the script `pcc_select.py`. The reason for this two-stage process is that some bands/polarizations may generate erroneous values which should not be incorporated into the proxy-cable delay correction. Typically, reasons for rejecting a particular band/polarization are large jumps in the delay trend, many failed fits or large outlier values, and/or large disagreement in the delay trend with the other band/polarizations. Currently, rejection criteria for a particular band/polarization have not been quantitatively formulated, but this will be addressed in future versions of this document.

To run `pcc_select.py` the user must specify the experiment name, the directory containing the .dat files produced by `pcc_generate.py`, and for each station: the bands and polarizations which are to be selected and averaged together. For example, in the command below:

```
pcc_select.py -e vt8190 -d ./pcc_test/ -o ./pcmt_test -s G:BCD:XY E:BCD:Y H:BD:XY
```

the delays from bands B, C, and D for both polarizations will be averaged for station G, for station E the same bands will be averaged but the X-polarization excluded, while for station H only the delays from bands B, D, and both polarizations will be averaged into the output. Individual band-polarizations may be selected for averaging, but if this option is chosen then all band-polarization combinations that are to be averaged must be listed explicitly. A full description of the usage options is given in listing 3.

Listing 3: Usage and arguments for the script `pcc_select.py`

```
usage: pcc_select.py [-h] -e EXPERIMENT_NAME -d DAT_DIRECTORY
                     [-o OUTPUT_DIRECTORY] [-y] [-n] -s
                     [STATION_BANDS_POLS [STATION_BANDS_POLS ...]]

required arguments:
  -e EXPERIMENT_NAME, --experiment EXPERIMENT_NAME
                        Experiment name, for example: vt7226
  -d DAT_DIRECTORY, --dat-dir DAT_DIRECTORY
                        Directory containing per-band .dat files
  -s [STATION_BANDS_POLS [STATION_BANDS_POLS ...]], --select [STATION_BANDS_POLS [STATION_BANDS_POLS ...]]
                        Space deliminated lists of selected stations, bands,
                        and polarizations. Stations must be specified with
                        single character code. allowable values for bands: A,
                        B, C, D allowable values for polarizations: X, Y. The
                        following two examples are equivalent Example 1:
                        (select bands/pols collectively): G:BCD:XY E:BC:XY
                        V:BC:X Y:BCD:Y Example 2: (list individiual bands-pols
                        separately): G:BX,BY,CX,CY,DX,DY E:BX,BY,CX,CY V:BX,CX
                        Y:BY,CY,DY

optional arguments:
  -o OUTPUT_DIRECTORY, --output-dir OUTPUT_DIRECTORY
                        Specify the output directory, default is DAT_DIRECTORY
  -y, --yes             reply yes to all prompts, default=False.
  -n, --no              reply no to all prompts, default=False.
```

4 Database creation and modification with vgosDbMake and vgosDbProcLogs

Once the experiment has been fringe-fit in pseudo-Stokes-I mode, and any desired proxy-cable calibration files have been prepared, a geodetic database may be generated for output to the analysis centers. This is done using the utility `vgosDbMake` distributed with the `nuSolve` software [?]. We will not cover the installation and configuration of this software as it is beyond the scope of this document, but it is expected that individual correlator sites will have determined their own directory structure for the generation and archival of experiment database files with the vgosDb utilities.

The basic usage of the `vgosDbMake` utility follows the form:

```
vgosDbMake <options> <path-to-fringe-files>
```

For a complete description of the available options of `vgosDbMake`, we refer the reader to the user guide distributed with the software [?]. A simple example demonstrating the creation of a database named `19JAN07VG` from the fringe file data from experiment 3678 is as follows:

```
vgosDbMake -d "19JAN07VG" /data/geodesy/3678
```

Note that when vgosDbMake is run it is important that the only fringe files present in the experiment directory are those from the final pass pseudo-Stokes-I fringe fitting. If a correlator report is to be appended to the database, it should be added at this time via the '-t' option.

After `vgosDbMake` has been run and a version-1 database file has been created, it is now time to run vgosDbCalc. Assuming the a priori model information has been configured and updated, then running vgosDbCalc is straightforward and can be done quite simply by pointing it to the version-1 database wrapper file. For example, the following command:

```
vgosDbCalc /data/vgos/vgosDb/2019/19JAN07VG/19JAN07VG_V001_iMIT_kall.wrp
```

will augment the database with the theoretical delay model information and generate a version-2 database wrapper.

Next the station log information should be attached to the database. The station log files may be obtained either directly from the station operator, through CDDIS, or the IVS website[15]. Once the station log files are available and have been placed in the appropriate directory [16], appending them to the database can be done with the utility vgosDbProcLogs. For example, to append the station log data for the session 19JAN07VG, the following command would be run on the version-2 database wrapper:

```
vgosDbProcLogs -k log /media/barrettj/data/vgos/vgosDb/2019/19JAN07VG/19JAN07_V002_iMIT_kall.wrp
```

Note that in addition to met-data, this command will also import the station cable calibration data for those stations which propagate this data into the station log file. For those stations which either do not have a cable calibration unit or have one which is temporarily malfunctioning, a second pass with vgosDbProcLogs is necessary to attach the proxy cable calibration data. To perform this second pass, collect the `pcmt` files which were generated by `pcc_select.py` and place them in the same directory as the station logs. Then to simultaneously zero out any previously existing cable calibration and insert the proxy data for the selected stations run:

```
 vgosDbProcLogs -zc -s <station1> ... -s <stationN> -k pcmt </path/to/database/wrapper/file>
```

As an example, the following command:

```
vgosDbProcLogs -zc -s KOKEE12M -s WETTZ13S -s RAEGYEB -s GGAO12M -k pcmt /data/19JAN07VG/19
     JAN07_V003_iMIT_kall.wrp
```

will zero out any existing cable calibration information for the stations: KOKEE12M, WETTZ13S, RAEGYEB, and GGAO12M and replace it with the delay values found in the proxy cable calibration pcmt files. This will generate a version-4 database wrapper file. At this point the basic VGOS database creation is complete. If desired, further editing of the database may be done interactively though `nuSolve` but it is recommended that this should be done in consultation with the analysis center.

[15]`https://ivscc.gsfc.nasa.gov/sessions/`

[16]The location is correlator-site specific, but is usually determined at the time when the nuSolve software is installed/configured.

5 VGOS data processing checklist

The complete processing chain can largely be broken down into the following steps:

1. ☐ Collect and mount the Mark-6 disk modules from each station. Make e-transfered data available to the correlator file system.
2. ☐ Generate a file list for each station using the DiFX utility program `vsum`.
3. ☐ Obtain the session .vex file and modify it to include the frequency setup information. Modify as needed for each station.
4. ☐ Following the procedure outlined in [?], set up the clocks section of the correlator .vex file, and adjust station sampler delays if needed.
5. ☐ Configure the machines file for the correlator, and ensure the CALC server is running and accessible.
6. ☐ Configure the .v2d file, paying particular attention to the data format provided by each station.
7. ☐ Convert the vex file to DiFX input using vex2difx.
8. ☐ Run the correlation using startdifx to generate the delay model and launch mpifxcorr.
9. ☐ Once correlation is finished, convert the Swinburne files to Mk4 format using difx2mark4. Correct any station ID labels and ensure that all channels are grouped into one band (X).
10. ☐ Prepare an a priori fourfit control file for this session.
11. ☐ Through either `vgoscf_generate.py` or a combination of `ffres2pcp.py`/`fourphase.py`, construct the pseudo-Stokes-I fourfit control file.
12. ☐ Use `summarize_report.py` to inspect the results of this process.
13. ☐ Fourfit all scan data in the session using the pseudo-Stokes-I fourfit control file. If desired, this can be expedited with the program `batch_fourfit.py`.
14. ☐ Use `phase_resid.py` to inspect the channel phases residuals for each scan/station/channel.
15. ☐ Use `pcc_generate.py` to produce proxy-cable calibration files for those stations which lack a dedicated cable delay calibration system.
16. ☐ Inspect the delay trend plots produced by `pcc_generate.py` to determine which bands and polarizations should be used to compute the delay corrections. Use `pcc_select.py` to select and average the appropriate files.
17. ☐ Generate the version-1 geodetic database using vgosDbMake.
18. ☐ Generate the version-2 geodetic database using vgosDbCalc.
19. ☐ Append the station log file data with vgosDbProcLogs to create the version-3 database. If necessary, insert the proxy cable calibration corrections with a second pass of vgosDbProcLogs generating the version-4 database.
20. ☐ Deliver the final database to the analysis center.

6 Example control file

Listing 4: Example initial control file.

```
*=========================================================
* Example initial control file
*=========================================================
dr_win -5.e-6 5.e-6      *specify delay rate window
pc_mode multitone        *specify that phase-cal will be given as multi-tone data
pc_period 1              *integration period for phase-cal tones
ion_smooth true          *smooth the dTEC fit function to avoid spurious peaks
mbd_anchor sbd           *use single-band delay to guide multi-band delay fit
samplers 4 abcdefgh ijklmnop qrstuvwx yzABCDEF    *group channels of each sampler
ref_freq 6000.0          *specify the reference frequency
weak_channel 0.1         *lower the weak channel G code threshold
pc_amp_hcode .001        *lower the phase-cal amplitude for H codes
ion_npts 45              *set number of coarse dTEC evaulation points
ion_win -88.0 88.0       *set wide search window for dTEC
pc_tonemask cdejnprwBC 2 16 16 1 16 16 16 2 16 16 *Set tone mask for n*100 MHz and 3090 MHz

*=========================================================
* Station sampler delays
*=========================================================
if station G
  sampler_delay_x -140 180 180 180
  sampler_delay_y -140 180 180 180

if station E
  sampler_delay_x -20 -20 -20 -20
  sampler_delay_y -20 -20 -20 -20

if station H
  sampler_delay_x -200 76 76 76
  sampler_delay_y -200 76 76 76

if station V
  sampler_delay_x 20  10  20  20
  sampler_delay_y 20  10  20  20

if station I
  sampler_delay_x 15  -15  -45  -50
  sampler_delay_y 15  -15  -45  -50

*=========================================================
* A priori channel-by-channel phase corrections, G:X is phase reference
*=========================================================
if station G
  pc_phases_y abcdefghijklmnopqrstuvwxyzABCDEF -2.5 -0.4 1.6 2.4 -0.8 0.5 2.5 2.9 -1.4 2.4 2.7 0.5 1.2 1.3 1.0
        -0.5 -0.7 -1.2 0.3 -1.4 0.4 0.1 -0.9 -1.8 -3.6 -6.0 -4.2 -1.1 -2.1 -3.6 0.4 -0.7

if station H
  pc_phases_x abcdefghijklmnopqrstuvwxyzABCDEF -10.2 -10.3 -9.3 -0.5 -0.1 2.0 6.1 5.0 -0.4 0.1 3.4 2.0 10.1
       12.4 10.2 9.5 0.8 -6.0 -5.4 -1.6 -3.1 -0.8 -3.3 -5.2 -16.5 -11.4 -12.5 -4.3 -1.6 1.7 1.4 2.9
  pc_phases_y abcdefghijklmnopqrstuvwxyzABCDEF -6.9 -6.4 -4.4 0.1 -4.6 -0.9 2.8 3.3 0.3 -0.2 0.7 4.4 11.4 14.9
        11.6 11.8 -3.2 -4.6 -1.9 -3.9 -4.7 -2.5 -1.0 -4.9 -16.5 -12.3 -9.5 -1.5 4.4 2.8 14.1 16.9

if station E
  pc_phases_x abcdefghijklmnopqrstuvwxyzABCDEF -4.4 -2.1 -0.1 -6.4 -1.7 10.1 17.6 19.5 8.3 13.0 11.8 13.5 9.4
       8.4 3.2 4.9 -31.4 -32.0 -32.9 -29.0 -21.5 -20.8 -13.9 -13.0 18.8 13.3 12.1 5.1 6.2 7.2 13.0 14.6
  pc_phases_y abcdefghijklmnopqrstuvwxyzABCDEF -13.2 -2.1 2.5 -30.2 -18.1 8.1 24.4 31.6 5.9 10.7 19.1 21.9
       24.8 13.9 7.1 12.8 -18.8 -27.0 -27.8 -31.1 -23.6 -23.2 -13.6 -12.3 13.3 8.6 6.5 5.5 10.6 3.6 3.5 6.5

if station V
  pc_phases_x abcdefghijklmnopqrstuvwxyzABCDEF -46.2 -37.2 -26.2 2.6 23.4 32.2 33.9 27.3 -2.5 3.4 5.4 7.8 12.6
        20.9 28.4 22.6 -8.8 -8.9 -9.4 -12.6 -15.1 -23.9 -24.5 -26.0 -0.5 1.9 6.9 5.8 -7.0 -4.5 9.7 11.5
  pc_phases_y abcdefghijklmnopqrstuvwxyzABCDEF -43.1 -33.7 -24.3 6.1 26.3 36.8 36.4 30.1 -9.6 -7.9 -3.2 -3.5
       2.9 8.2 14.2 15.6 -2.7 -1.6 -4.6 -9.2 -11.9 -19.4 -23.0 -26.5 -4.0 -5.2 -1.5 0.9 9.5 9.9 23.7 13.6

if station I
  pc_phases_x abcdefghijklmnopqrstuvwxyzABCDEF -11.6 3.6 7.6 5.0 -6.0 -3.0 -2.1 16.4 -3.3 3.4 1.1 -5.9 -7.0
       -1.6 1.2 6.4 0.7 2.0 0.9 -0.8 -2.0 -3.5 -2.9 0.8 -0.3 -3.3 -7.0 4.6 4.0 -4.4 2.9 3.7
  pc_phases_y abcdefghijklmnopqrstuvwxyzABCDEF -4.3 -1.5 -2.1 -8.7 0.7 2.2 6.5 18.2 -8.9 -3.8 0.4 -7.3 -3.7
       -3.6 -0.1 6.6 7.1 6.0 4.4 0.4 -1.0 -3.2 -2.8 1.9 -4.0 -5.8 -6.5 -0.1 2.4 0.7 4.7 4.5
```

7 Appendix: VGOS correlator .v2d and .vex file setup examples

Listing 5: An example earth orientation parameter ($EOP) section of the session .vex file.

```
 *----------------------- begin $EOP           ----------------------*
$EOP;
 def EOP003;
  TAI-UTC = 37 sec;
  A1-TAI = 0.03439 sec;
  eop_ref_epoch = 2018y203d;
  num_eop_points = 5;
  eop_interval = 24 hr;
* I E R S Rapid Service from 26 Jul 2018 ser7
  ut1-utc =   0.069047 sec : 0.069073 sec : 0.069190 sec : 0.069337 sec : 0.069516 sec;
  x_wobble =  0.19271 asec : 0.19377 asec : 0.19477 asec : 0.19549 asec : 0.19588 asec;
  y_wobble =  0.41361 asec : 0.41277 asec : 0.41221 asec : 0.41182 asec : 0.41136 asec;
 enddef;
```

Listing 6: Example .v2d file configuration.

```
vex = vt8204.vex.obs
mjdStart = 2018y204d18h00m00s
mjdStop = 2018y205d18h00m00s;
antennas = Gs,K2
startSeries = 1005
singleScan = True
machines = Mark6-4082
nCore = 6
nThread = 2

SETUP default
{
  nChan = 128
  tInt=1
}

ANTENNA Gs
{
  machine = Mark6-4082
  format = VDIFC/0:1:2:3/8224/2
  mark6filelist = vt8204_gs.filelist
  sampling = COMPLEX
  phaseCalInt = 5
}

ANTENNA K2
{
  machine = Mark6-4082
  format = VDIFC/0:1:2:3/8224/2
  mark6filelist = vt8204_k2.filelist
  sampling = COMPLEX
  phaseCalInt = 5
}
```

Listing 7: Example VGOS $IF .vex configuration.

```
*----------------------- begin $IF            ----------------------*
$IF;
  def VGOS_std;
    if_def = &IF_1N : 3N : X : 8080.0 MHz : U : 5 MHz : 0 Hz;
    if_def = &IF_3N : 3N : Y : 8080.0 MHz : U : 5 MHz : 0 Hz;
  enddef;
*----------------------- end $IF              ----------------------*
```

Listing 8: Example VGOS $TRACKS .vex configuration.

```
*----------------------- begin $TRACKS        ----------------------*
$TRACKS;
  def VDIF_format;
    track_frame_format = VDIF;
  enddef;
*----------------------- end $TRACKS          ----------------------*
```

Listing 9: Example VGOS $BBC .vex configuration.

```
*----------------------- begin $BBC           ----------------------*
$BBC;
  def VGOS_std;
    BBC_assign = &BBC01 : 01 : &IF_1N;
    BBC_assign = &BBC02 : 02 : &IF_1N;
    BBC_assign = &BBC03 : 03 : &IF_1N;
    BBC_assign = &BBC04 : 04 : &IF_1N;
    BBC_assign = &BBC05 : 05 : &IF_1N;
    BBC_assign = &BBC06 : 06 : &IF_1N;
    BBC_assign = &BBC07 : 07 : &IF_1N;
    BBC_assign = &BBC08 : 08 : &IF_1N;
    BBC_assign = &BBC09 : 09 : &IF_3N;
    BBC_assign = &BBC10 : 10 : &IF_3N;
    BBC_assign = &BBC11 : 11 : &IF_3N;
    BBC_assign = &BBC12 : 12 : &IF_3N;
    BBC_assign = &BBC13 : 13 : &IF_3N;
    BBC_assign = &BBC14 : 14 : &IF_3N;
    BBC_assign = &BBC15 : 15 : &IF_3N;
    BBC_assign = &BBC16 : 16 : &IF_3N;
    BBC_assign = &BBC17 : 01 : &IF_1N;
    BBC_assign = &BBC18 : 02 : &IF_1N;
    BBC_assign = &BBC19 : 03 : &IF_1N;
    BBC_assign = &BBC20 : 04 : &IF_1N;
    BBC_assign = &BBC21 : 05 : &IF_1N;
    BBC_assign = &BBC22 : 06 : &IF_1N;
    BBC_assign = &BBC23 : 07 : &IF_1N;
    BBC_assign = &BBC24 : 08 : &IF_1N;
    BBC_assign = &BBC25 : 09 : &IF_3N;
    BBC_assign = &BBC26 : 10 : &IF_3N;
    BBC_assign = &BBC27 : 11 : &IF_3N;
    BBC_assign = &BBC28 : 12 : &IF_3N;
    BBC_assign = &BBC29 : 13 : &IF_3N;
    BBC_assign = &BBC30 : 14 : &IF_3N;
    BBC_assign = &BBC31 : 15 : &IF_3N;
    BBC_assign = &BBC32 : 16 : &IF_3N;
    BBC_assign = &BBC33 : 01 : &IF_1N;
    BBC_assign = &BBC34 : 02 : &IF_1N;
    BBC_assign = &BBC35 : 03 : &IF_1N;
    BBC_assign = &BBC36 : 04 : &IF_1N;
    BBC_assign = &BBC37 : 05 : &IF_1N;
    BBC_assign = &BBC38 : 06 : &IF_1N;
    BBC_assign = &BBC39 : 07 : &IF_1N;
    BBC_assign = &BBC40 : 08 : &IF_1N;
    BBC_assign = &BBC41 : 09 : &IF_3N;
    BBC_assign = &BBC42 : 10 : &IF_3N;
    BBC_assign = &BBC43 : 11 : &IF_3N;
    BBC_assign = &BBC44 : 12 : &IF_3N;
    BBC_assign = &BBC45 : 13 : &IF_3N;
    BBC_assign = &BBC46 : 14 : &IF_3N;
    BBC_assign = &BBC47 : 15 : &IF_3N;
    BBC_assign = &BBC48 : 16 : &IF_3N;
    BBC_assign = &BBC49 : 01 : &IF_1N;
    BBC_assign = &BBC50 : 02 : &IF_1N;
    BBC_assign = &BBC51 : 03 : &IF_1N;
    BBC_assign = &BBC52 : 04 : &IF_1N;
    BBC_assign = &BBC53 : 05 : &IF_1N;
    BBC_assign = &BBC54 : 06 : &IF_1N;
    BBC_assign = &BBC55 : 07 : &IF_1N;
    BBC_assign = &BBC56 : 08 : &IF_1N;
    BBC_assign = &BBC57 : 09 : &IF_3N;
    BBC_assign = &BBC58 : 10 : &IF_3N;
    BBC_assign = &BBC59 : 11 : &IF_3N;
    BBC_assign = &BBC60 : 12 : &IF_3N;
    BBC_assign = &BBC61 : 13 : &IF_3N;
    BBC_assign = &BBC62 : 14 : &IF_3N;
    BBC_assign = &BBC63 : 15 : &IF_3N;
    BBC_assign = &BBC64 : 16 : &IF_3N;
  enddef;
*----------------------- end $BBC             ----------------------*
```

Listing 10: Example VGOS $FREQ .vex configuration.

```
*----------------------- begin $FREQ          ----------------------*
$FREQ;
  def VGOS_std;
    chan_def = &X : 3480.40 MHz : L : 32.000 MHz : &Ch01 : &BBC01 : &L_cal;
    chan_def = &X : 3448.40 MHz : L : 32.000 MHz : &Ch02 : &BBC02 : &L_cal;
    chan_def = &X : 3384.40 MHz : L : 32.000 MHz : &Ch03 : &BBC03 : &L_cal;
    chan_def = &X : 3320.40 MHz : L : 32.000 MHz : &Ch04 : &BBC04 : &L_cal;
    chan_def = &X : 3224.40 MHz : L : 32.000 MHz : &Ch05 : &BBC05 : &L_cal;
    chan_def = &X : 3096.40 MHz : L : 32.000 MHz : &Ch06 : &BBC06 : &L_cal;
    chan_def = &X : 3064.40 MHz : L : 32.000 MHz : &Ch07 : &BBC07 : &L_cal;
    chan_def = &X : 3032.40 MHz : L : 32.000 MHz : &Ch08 : &BBC08 : &L_cal;
    chan_def = &X : 3480.40 MHz : L : 32.000 MHz : &Ch09 : &BBC09 : &L_cal;
    chan_def = &X : 3448.40 MHz : L : 32.000 MHz : &Ch10 : &BBC10 : &L_cal;
    chan_def = &X : 3384.40 MHz : L : 32.000 MHz : &Ch11 : &BBC11 : &L_cal;
    chan_def = &X : 3320.40 MHz : L : 32.000 MHz : &Ch12 : &BBC12 : &L_cal;
    chan_def = &X : 3224.40 MHz : L : 32.000 MHz : &Ch13 : &BBC13 : &L_cal;
    chan_def = &X : 3096.40 MHz : L : 32.000 MHz : &Ch14 : &BBC14 : &L_cal;
    chan_def = &X : 3064.40 MHz : L : 32.000 MHz : &Ch15 : &BBC15 : &L_cal;
    chan_def = &X : 3032.40 MHz : L : 32.000 MHz : &Ch16 : &BBC16 : &L_cal;
    chan_def = &X : 5720.40 MHz : L : 32.000 MHz : &Ch17 : &BBC17 : &L_cal;
    chan_def = &X : 5688.40 MHz : L : 32.000 MHz : &Ch18 : &BBC18 : &L_cal;
    chan_def = &X : 5624.40 MHz : L : 32.000 MHz : &Ch19 : &BBC19 : &L_cal;
    chan_def = &X : 5560.40 MHz : L : 32.000 MHz : &Ch20 : &BBC20 : &L_cal;
    chan_def = &X : 5464.40 MHz : L : 32.000 MHz : &Ch21 : &BBC21 : &L_cal;
    chan_def = &X : 5336.40 MHz : L : 32.000 MHz : &Ch22 : &BBC22 : &L_cal;
    chan_def = &X : 5304.40 MHz : L : 32.000 MHz : &Ch23 : &BBC23 : &L_cal;
    chan_def = &X : 5272.40 MHz : L : 32.000 MHz : &Ch24 : &BBC24 : &L_cal;
    chan_def = &X : 5720.40 MHz : L : 32.000 MHz : &Ch25 : &BBC25 : &L_cal;
    chan_def = &X : 5688.40 MHz : L : 32.000 MHz : &Ch26 : &BBC26 : &L_cal;
    chan_def = &X : 5624.40 MHz : L : 32.000 MHz : &Ch27 : &BBC27 : &L_cal;
    chan_def = &X : 5560.40 MHz : L : 32.000 MHz : &Ch28 : &BBC28 : &L_cal;
    chan_def = &X : 5464.40 MHz : L : 32.000 MHz : &Ch29 : &BBC29 : &L_cal;
    chan_def = &X : 5336.40 MHz : L : 32.000 MHz : &Ch30 : &BBC30 : &L_cal;
    chan_def = &X : 5304.40 MHz : L : 32.000 MHz : &Ch31 : &BBC31 : &L_cal;
    chan_def = &X : 5272.40 MHz : L : 32.000 MHz : &Ch32 : &BBC32 : &L_cal;
    chan_def = &X : 6840.40 MHz : L : 32.000 MHz : &Ch33 : &BBC33 : &L_cal;
    chan_def = &X : 6808.40 MHz : L : 32.000 MHz : &Ch34 : &BBC34 : &L_cal;
    chan_def = &X : 6744.40 MHz : L : 32.000 MHz : &Ch35 : &BBC35 : &L_cal;
    chan_def = &X : 6680.40 MHz : L : 32.000 MHz : &Ch36 : &BBC36 : &L_cal;
    chan_def = &X : 6584.40 MHz : L : 32.000 MHz : &Ch37 : &BBC37 : &L_cal;
    chan_def = &X : 6456.40 MHz : L : 32.000 MHz : &Ch38 : &BBC38 : &L_cal;
    chan_def = &X : 6424.40 MHz : L : 32.000 MHz : &Ch39 : &BBC39 : &L_cal;
    chan_def = &X : 6392.40 MHz : L : 32.000 MHz : &Ch40 : &BBC40 : &L_cal;
    chan_def = &X : 6840.40 MHz : L : 32.000 MHz : &Ch41 : &BBC41 : &L_cal;
    chan_def = &X : 6808.40 MHz : L : 32.000 MHz : &Ch42 : &BBC42 : &L_cal;
    chan_def = &X : 6744.40 MHz : L : 32.000 MHz : &Ch43 : &BBC43 : &L_cal;
    chan_def = &X : 6680.40 MHz : L : 32.000 MHz : &Ch44 : &BBC44 : &L_cal;
    chan_def = &X : 6584.40 MHz : L : 32.000 MHz : &Ch45 : &BBC45 : &L_cal;
    chan_def = &X : 6456.40 MHz : L : 32.000 MHz : &Ch46 : &BBC46 : &L_cal;
    chan_def = &X : 6424.40 MHz : L : 32.000 MHz : &Ch47 : &BBC47 : &L_cal;
    chan_def = &X : 6392.40 MHz : L : 32.000 MHz : &Ch48 : &BBC48 : &L_cal;
    chan_def = &X : 10680.40 MHz : L : 32.000 MHz : &Ch49 : &BBC49 : &L_cal;
    chan_def = &X : 10648.40 MHz : L : 32.000 MHz : &Ch50 : &BBC50 : &L_cal;
    chan_def = &X : 10584.40 MHz : L : 32.000 MHz : &Ch51 : &BBC51 : &L_cal;
    chan_def = &X : 10520.40 MHz : L : 32.000 MHz : &Ch52 : &BBC52 : &L_cal;
    chan_def = &X : 10424.40 MHz : L : 32.000 MHz : &Ch53 : &BBC53 : &L_cal;
    chan_def = &X : 10296.40 MHz : L : 32.000 MHz : &Ch54 : &BBC54 : &L_cal;
    chan_def = &X : 10264.40 MHz : L : 32.000 MHz : &Ch55 : &BBC55 : &L_cal;
    chan_def = &X : 10232.40 MHz : L : 32.000 MHz : &Ch56 : &BBC56 : &L_cal;
    chan_def = &X : 10680.40 MHz : L : 32.000 MHz : &Ch57 : &BBC57 : &L_cal;
    chan_def = &X : 10648.40 MHz : L : 32.000 MHz : &Ch58 : &BBC58 : &L_cal;
    chan_def = &X : 10584.40 MHz : L : 32.000 MHz : &Ch59 : &BBC59 : &L_cal;
    chan_def = &X : 10520.40 MHz : L : 32.000 MHz : &Ch60 : &BBC60 : &L_cal;
    chan_def = &X : 10424.40 MHz : L : 32.000 MHz : &Ch61 : &BBC61 : &L_cal;
    chan_def = &X : 10296.40 MHz : L : 32.000 MHz : &Ch62 : &BBC62 : &L_cal;
    chan_def = &X : 10264.40 MHz : L : 32.000 MHz : &Ch63 : &BBC63 : &L_cal;
    chan_def = &X : 10232.40 MHz : L : 32.000 MHz : &Ch64 : &BBC64 : &L_cal;
    sample_rate = 64.0 Ms/sec;
  enddef;
*----------------------- end $FREQ            ----------------------*
```