HOPS4 and Software Updates

J. Barrett on behalf of the Haystack Geodesy Team

MIT Haystack Observatory

TOW Correlator Workshop – May 9, 2025

Outline

MIT HAYSTACK OBSERVATORY

1. Introduction - background, history, motivation
2. The HOPS4 package, dependencies and installation
3. Notable difference and similarities with HOPS3
4. New capabilities and data formats
5. Comparison with HOPS3
6. On going work and future outlook

Background

HOPS - Haystack Observatory Post-processing System

MIT HAYSTACK OBSERVATORY

Collection of utilities which provide post-correlation data processing for VLBI data, including:

1. Importing data from a software correlator (DiFX)
2. Data flagging/removal of corrupted visibilities in time/frequency
3. Calibration and phase correction of data, as well as other manipulation
4. Fringe-fitting (solving for residual group-delay/delay-rate w.r.t to correlator model) on a per-scan, per-baseline basis
5. Data quality analysis and visualization for debugging station problems

Brief History

MIT HAYSTACK OBSERVATORY

1. HOPS has a multi-decade history as a post-processing/analysis package and has evolved continuously over time.
    1. Current C-code was written in the early 90's by Colin Lonsdale, Roger Cappallo and Cris Niell.
    2. Additional adaptations made over time to provide additional knobs for data treatment, support wider bandwidth, optimize SNR, and support software correlators such as DiFX
2. `fourfit` – primary component is this fringe-fitting tool.
    1. The basic algorithm was adopted from the FORTRAN routine `FRNGE`, developed by Alan Rogers in late 70's [1]
    2. Data flagging and calibration is configured via the `fourfit` control file.
3. Additional tools: `alist` & `aedit` (among others) – provide diagnostics for debugging, inspection and evaluation of data quality.

[1] `https://doi.org/10.1029/RS005i010p01239`

VGOS History/Upgrades

MIT HAYSTACK OBSERVATORY

1. Multi-tone phase cal. & sampler delays [2]
   1. Fits multiple tone phasors per-channel to extract phase and delay offset
   2. Resolves channel delay ambiguities of the same (physical) sampler
2. Ionospheric dTEC fitting
   1. Correction to compensate for ionospheric dTEC dispersion
   2. Introduced as outermost-loop of grid search – estimated simultaneously with group delay
3. Pol-product summation (pseudo-Stokes-I) [3]
   1. Maximize SNR, eliminate dependence on Δ parallactic angle
   2. Requires per-station, per-polarization phase/delay offset corrections
4. Python post-processing scripts – manage control file generation and diagnostics, parallel batch fringe fitting
5. Proxy-cable cal. for stations without hardware cable-cal.

[2] https://www.haystack.mit.edu/wp-content/uploads/2020/07/docs_hops_011_multitone_phasecal.pdf

[3] https://ivscc.gsfc.nasa.gov/publications/gm2014/019_Cappallo.pdf

Motivation to Upgrade HOPS

MIT HAYSTACK OBSERVATORY

1. Goal: Eliminate limitations while maintaining existing capabilities
2. Technical limitations:
   1. Relatively limited ability for dynamic memory allocation - fixed limit on the number of stations, APs, and channels
   2. Fully complex (amplitude and phase) band-pass corrections are not possible (only phase/delay corrections)
   3. Only a single per-scan/baseline (grid search) fringe-finding algorithm is available
   4. Currently no support for multi-processing, apart from single-program-multiple-data
3. Practical limitations:
   1. Plotting and results are not decoupled
   2. Custom data treatment (e.g. band-pass correction) is limited to that which is allowed by the control file parameters
   3. File and data formats (Mk4-types) are restrictive and not easily modified
   4. Required use of old/unmaintained libraries (e.g. PGPLOT)

Additional Motivation / Future needs

1. Primary objectives:
   1. Increase flexibility of data structures to support as-yet-unknown (meta)data collections (no c-structs)
   2. Enable the ability to implement multiple fringe-finding algorithms
   3. Support user-injected python plugins for data manipulation
   4. Enable fully complex (sub-channel) bandpass correction
   5. Support VEX 2.0
2. Secondary Objectives:
   1. Augment with SIMD parallelism (OpenCL/CUDA)
   2. Allow for multiple plotting backends

The HOPS4 Package

MIT HAYSTACK OBSERVATORY

1. The HOPS4 package consists of three main components:
   1. The original HOPS3 C applications and libraries (3.26) - `fourfit3`, `alist3`, `fplot3` etc. These are provided along with all of the new code for convenience, comparison testing, and general utility[4].
   2. The new C++ applications and libraries - `fourfit4`, `alist4`, `fplot4`, etc.
   3. Python scripts and extensions (e.g. vgos scripting)
2. HOPS4 has no required dependencies on the old C-libraries (with the exception of mk4-data for import/export)
3. The new architecture supports the extension of the library/application code in two ways:
   1. Developer (compile-time, arbitrary) extensions to data-manipulation as a new derived class of the 'data operator' type
   2. User (run-time, limited) extensions via a python plug-in interface
4. C++ is the primary language and the build system has changed to CMake (instead of automake)

[4] symlinks with the original names are provided, e.g. fourfit → `fourfit3`

HOPS4 Installation

MIT HAYSTACK OBSERVATORY

1. HOPS4 beta[5] release now available on github: https://github.com/MITHaystack/HOPS
2. HOPS4 installation follows the familiar configure/make/install process, however, the configuration is now done via CMake
3. For example, on a fresh copy of ubuntu 22.04 to download and install:

```
#install the desired dependencies (HOPS4 & HOPS3):
sudo apt-get install build-essential cmake cmake-curses-gui python3-dev python3-
    pip wget jq libfftw3-dev pgplot5 libgfortran5 libfftw3-dev libx11-dev
    gnuplot binutils libxpm-dev ghostscript ghostscript-x
#then download the software and unpack it
wget https://github.com/MITHaystack/HOPS/archive/refs/tags/v4.0.0-beta2.tar.gz
tar -xzvf ./v4.0.0-beta2.tar.gz
cd ./HOPS-4.0.0-beta2
#configure the software
mkdir build
cd build/
ccmake ../ -DHOPS_PYPI_MANAGE_DEPS=ON #use OFF if numpy, matplotlib installed
#build and install
make -j8 && make install
```

[5] This release is intended for experimentation and familiarization only and is not meant for production use or processing!

HOPS4 Pre-requisites

MIT HAYSTACK OBSERVATORY

1. General philosophy is to minimize the required dependencies, and only build additional functionality if optional dependencies are present:
2. The absolute mininum HOPS4 requirements for the three primary HOPS4 applications (`fourfit4`, `alist4`, `fplot4`) are:
    1. build-essential (GCC, GNU make, etc)
    2. cmake, cmake-curses-gui
    3. python3-dev python3-pip†
3. FFTW3 is optional but highly recommended, as the native FFT implementation provided is slower
4. To install the original HOPS3 software as well, you will also need the typical HOPS3 dependencies:
    1. libfftw3-dev
    2. pgplot5, libgfortran5
    3. libx11-dev, gnuplot, binutils, libxpm-dev, ghostscript, ghostscript-x

HOPS4 Optional Dependencies

MIT HAYSTACK OBSERVATORY

1. If present, optional dependencies will enable additional functionality for HOPS4. These are
   1. DIFXIO: enables the library DiFXInterface and the application `difx2hops` (must have `DIFXROOT` defined when running `cmake`)
   2. MPI: allows `fourfit4` to be run in parallel using 'mpirun'
   3. CUDA: experimental - enables GPU based multi-band delay search
   4. OPENCL: experimental
   5. Doxygen/Sphinx: experimental documentation generation
2. Most of these optional dependencies must also be toggled 'ON' from the cmake interface in order to trigger the build of the desired functionality
3. Not a dependency, but useful tool for inspecting json files is jless[6].
4. † Not strictly optional, but for plotting using `fourfit4`, pip must install matplotlib, numpy and scipy. The flag HOPS_PYPY_MANAGE_DEPS toggles local install via pip. If you already have these packages, this should be set to OFF.

[6] https://jless.io/

HOPS4 CMake Interface

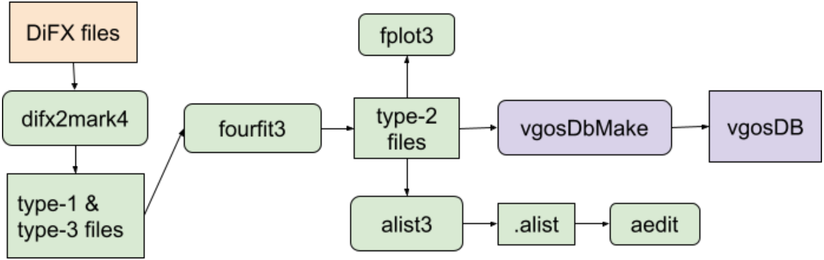
MIT
HAYSTACK
OBSERVATORY

```
barrettj@curie: ~/work/projects/hops-git/build
                                                                 Page 1 of 2
 BASH_PROGRAM                     /usr/bin/bash
 BC_PROGRAM                       /usr/bin/bc
 BUILD_DOXYGEN_REF                OFF
 BUILD_LATEX_DOCS                 OFF
 BUILD_SPHINX_DOCS                OFF
 CMAKE_BUILD_TYPE                 RelWithDebInfo
 CMAKE_INSTALL_PREFIX             /home/barrettj/work/projects/hops-git/x86_64-4.0.0
 CPGPLOT_LIBRARY                  /usr/lib/libcpgplot.so
 ENABLE_BUILD_D2M4                OFF
 ENABLE_COLOR_MSG                 ON
 ENABLE_DEBUG_MSG                 ON
 ENABLE_EXTRA_VERBOSE_MSG         OFF
 ENABLE_SNAPSHOTS                 OFF
 ENABLE_STEPWISE_CHECK            OFF
 EXTRA_WARNINGS                   OFF
 GFORTRAN_LIB                     /lib/x86_64-linux-gnu/libgfortran.so.5
 GS_EXE                           /usr/bin/gs
 HOPS3_DISABLE_WARNINGS           ON
 HOPS3_PYTHON_EXTRAS              ON
 HOPS3_USE_ADHOC_FLAGGING         ON
 HOPS_BUILD_EXTRA_CONTAINERS      OFF
 HOPS_CACHED_TEST_DATADIR         /home/barrettj/work/projects/hops-git/x86_64-4.0.0/data/test_data
 HOPS_ENABLE_DEV_TODO             OFF
 HOPS_ENABLE_TEST                 OFF
 HOPS_IS_HOPS4                    OFF
 HOPS_PYPI_MANAGE_DEPS            OFF
 HOPS_USE_CUDA                    OFF
 HOPS_USE_DIFXIO                  ON
 HOPS_USE_FFTW3                   ON
 HOPS_USE_MPI                     ON
 HOPS_USE_OPENCL                  OFF
 HOPS_USE_PYBIND11                ON

BASH_PROGRAM: Path to a program.
Keys: [enter] Edit an entry [d] Delete an entry                                         CMake Version 3.22.1
      [l] Show log output   [c] Configure
      [h] Help              [q] Quit without generating
      [t] Toggle advanced mode (currently off)
```

Overview of HOPS3/HOPS4 workflow

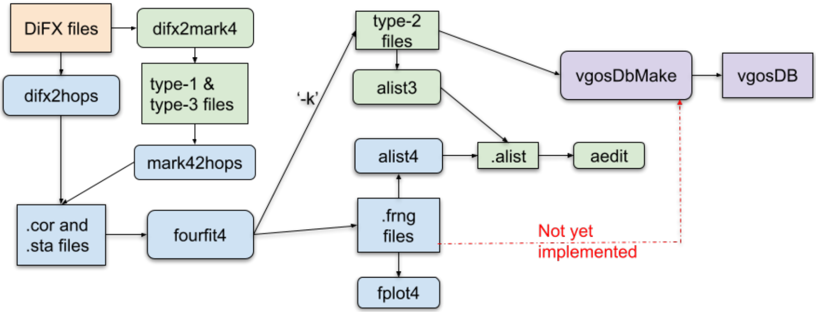
MIT HAYSTACK OBSERVATORY



1. Typical HOPS3 workflow:
    - difx2mark4 used to convert DiFX output to mark4 type-1 & type-3 files
    - `fourfit3` used to fringe-fit, and create type-2 files
    - vgosDbMake consumes the type-2 files and produces the database
    - `alist3` & `fplot3` can be used to inspect the fringe output

Overview of HOPS3/HOPS4 workflow

MIT HAYSTACK OBSERVATORY



1. Typical HOPS4 workflow:
   - difx2hops used to convert DiFX output to (correlator) .cor and (station) .sta files
   - `fourfit4` used to fringe-fit, can create .frng files or '-k' option to create type-2 files
   - vgosDbMake can consume the type-2 files and produce the database
   - `alist4` & `fplot4` can be used to inspect the .frng files
   - Export of .frng files directly to vgosDbMake is not yet available

Overview of software/tools in HOPS4

MIT HAYSTACK OBSERVATORY

HOPS4 Tool	HOPS3 equivalent	Description
fourfit4	fourfit3	fringe fitting tool
fplot4	fplot3	fringe plot tool
alist4	alist3	fringe summary
difx2hops	difx2mark4	creation of input files from DiFX
hops2json	CorAsc2	binary file inspection
hops2keys	CorAsc2	binary file summary
mark42hops		conversion of mark4 files to hops4
vex2json		conversion of vex to json
json2vex		conversion of json to vex
	aedit	data inspection
	adump	alist data extraction
	snratio	correlator report tool

Currently, HOPS4 tools for the equivalent functionality of: `fourmer`, `average`, `fringex`, and `search` are not yet available.

Notable differences & similarities

MIT HAYSTACK OBSERVATORY

1. Majority of existing control file features and syntax is supported by `fourfit4`[7]. Most HOPS4 programs provide a '–help' option.
2. Fringe plots in HOPS4 are generated using python instead of PGPLOT, & will (soon) support all of the existing content and data
3. To convert data from difx to HOPS4 format, use the utility difx2hops, which supports most of the same syntax as difx2mark4
4. `fourfit4` requires input files in the HOPS4 (.cor) format, but can generate either Mark4 output (type_2xx) files or HOP4 (.frng) files. Note: mark4 files generated with `fourfit4` will not contain fringe plots!
5. Existing VGOS post-processing scripts can be used with `fourfit4`, but you must set the environmental variable `HOPS_VPAL_FRINGE_FITTER=fourfit4` in order to use it, as the default is `fourfit3`.

[7] With the exception of the deprecated features listed on next slide

Deprecated keywords and features

MIT HAYSTACK OBSERVATORY

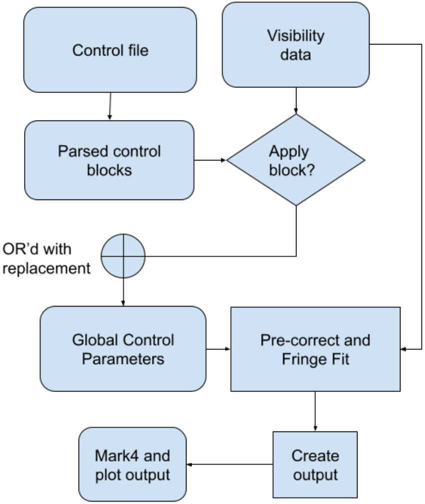
1. Mark4 hardware correlator specific features:
   - index
   - max_parity
   - x_crc and y_crc
   - x_slip_sync and y_slip_sync
   - use_samples
2. Frequency switching related:
   - switched
   - gates
   - period
3. Other:
   - ra_offset – was never implemented
   - dec_offset – was never implemented
   - pc_freqs
   - interpolator – default has been 'simultaneous' algorithm for some time, 'iterative' has been removed
   - fmatch_bw_pct

Furthermore pc_mode 'ap_by_ap' and 'normal' have not been implemented, only 'multitone' and 'manual' are currently allowed.

`fourfit3` architecture

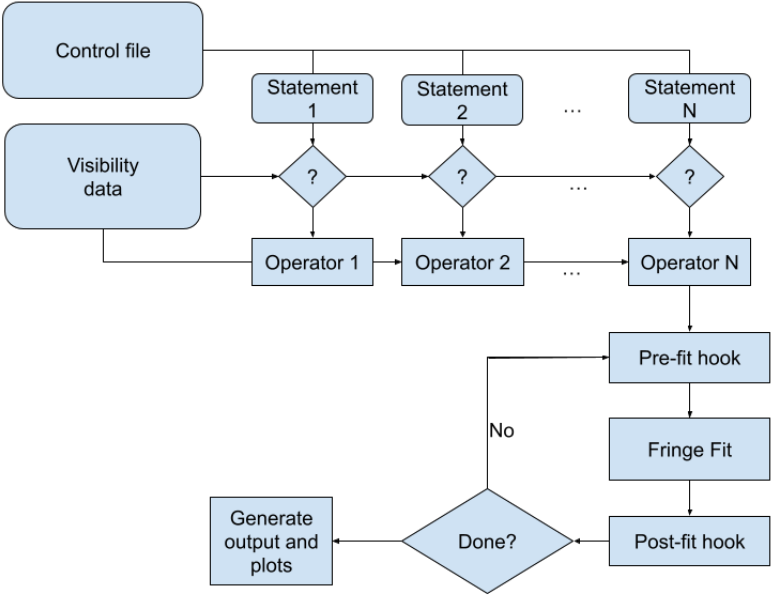
MIT HAYSTACK OBSERVATORY

1. Monolithic application
2. Control parameters combined (or'd) into single global control structure
3. Global parameters govern the operation during pre-correction/fitting
4. One-pass, create output and plot



fourfit4 architecture

MIT HAYSTACK OBSERVATORY

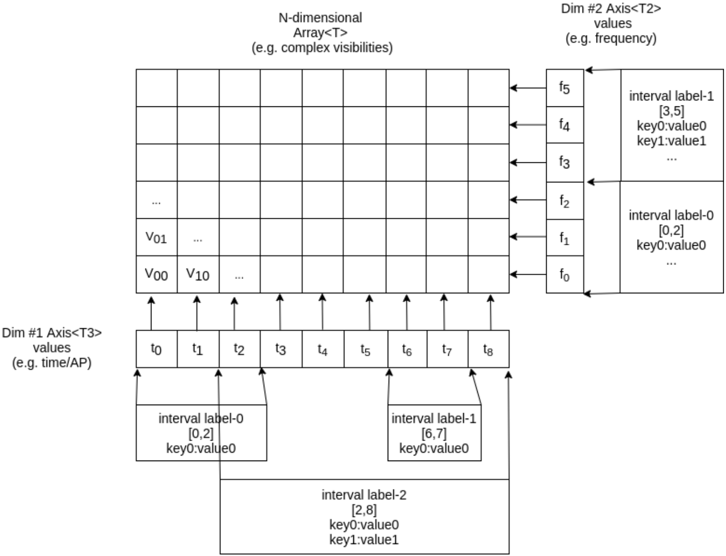


HOPS4 Data Containers

1. ND-array template class with data-type and rank parameters + axes, uses STL-style iterators
2. Axes are templated on coordinate type (float, string, etc.)
3. Axis interval labeling of array can be labeled with type-agnostic key:value pairs for data selection (json)
4. I/O library allows for per-object retrieval (unlike the Mk4, no need to read the entire file byte-by-byte to extract a single object)
5. Encompasses nearly all of the data objects in use by `fourfit4` (visibilities, weights, pcal, station model, etc.)
6. Used in-memory as well as for the .cor, .sta, and .frng files
7. Where needed, heterogenous data (nested mixtures of strings and numerical data) is stored as a json object in CBOR format
8. Utility hops2json is provided to allow text-based inspection of these data files (like CorAsc2)

HOPS4 Data Containers

MIT HAYSTACK OBSERVATORY



HOPS4 Data Operators

MIT HAYSTACK OBSERVATORY

1. Abstracts away the interface for data manipulation, and isolates each discrete operation from the rest of the program
2. Operator class only needs to define initialization and execution functions
3. Operators are built upon configuration and inserted in initialization/execution queue when/where desired by category (e.g. calibration)
4. Only pay for what you use
5. Categories: labeling, selection, flagging, calibration

Operator interface:

```
class MHO_Operator
{
    public:
      MHO_Operator(){};
      virtual ~MHO_Operator(){};
      virtual bool Initialize() = 0;
      virtual bool Execute() = 0;
}
```

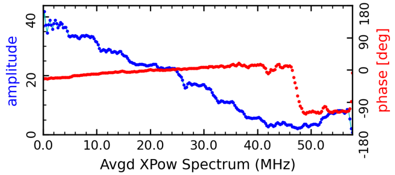
Plugin mechanism

MIT HAYSTACK OBSERVATORY

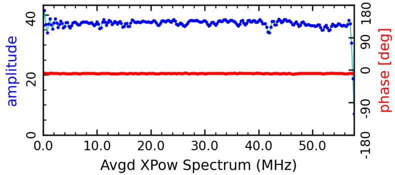
1. One specific type of data operator is a 'python plugin'
2. The Python interface is supported via the header-only pybind11 library
3. Bindings to container classes allow in-memory or file data to be manipulated by user python scripts
4. User python code can be injected at runtime into the fringe fitter for custom/experimental calculations with full access to in-memory parameters and visibility/weight data
5. Data access on python side is via an interface which allows retrieval by object name or UUID. ND-array data is exposed as a numpy array, and heterogenous data is exposed as a Python dictionary
6. Some restrictions/limitations:
    1. Python plugin operations are only applied at specify hooks/locations (e.g. calibration or pre/post-fit)
    2. Data can be accessed and modified, but can not be resized/reshaped/deleted

Example user plugin

1. Fixing intra-channel phase jumps – discrete changes in phase at fixed locations in each channel
2. Fixing this in HOPS3 would have required a devising an entirely new custom control file feature
3. Implemented in a single (short) python function.



(a) Pre-corrected average xpower spectrum



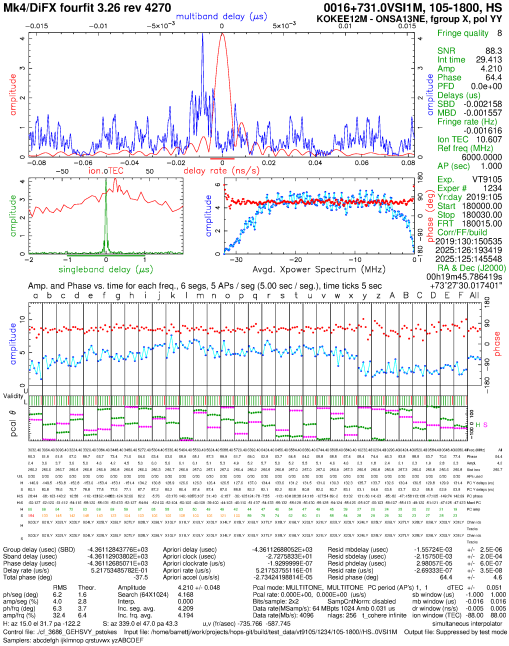
(b) Post-corrected average xpower spectrum

Other possible plugins...

1. Station pol-swap relabeling ($X \rightarrow Y$)
2. Auto G-code correction (weak channel cut)

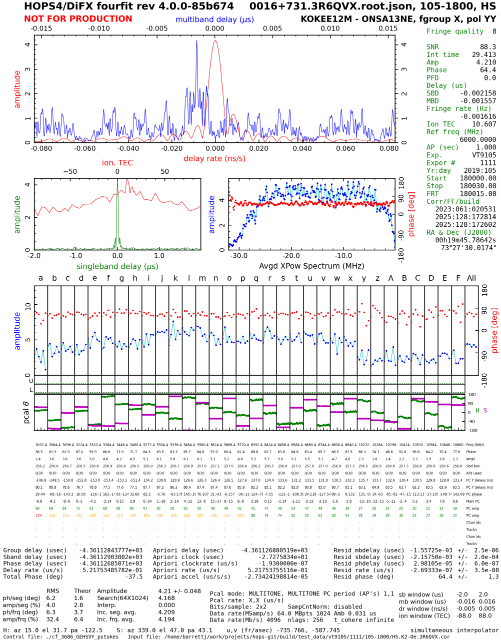
Plotting in HOPS3

1. Diagnostic plot generated for each fringe
2. Plotting is is done with PGPLOT → postscript
3. Plotting and mk4 data output intertwined



Plotting in HOPS4

1. Fringe results/output files are now independent of plotting
2. New plotting backend is matplotlib
3. Can reformat/-zoom/explore data in more detail, will allow further customization.



Numerical comparison

MIT HAYSTACK OBSERVATORY

1. Before routine use of new code, we need to validate HOPS4 behavior against HOPS3 (testing and debugging is on-going)
2. One straightforward comparison we've done so far is to simply examine the Pseudo-Stokes-I Mk4 output of a VGOS session (VR2404):
    1. Use original production control file*
    2. Only compare `fourfit4` vs `fourfit3` (not testing scripting e.g. vgoscf_generate.py)
    3. Fringe fitting was run via batch_fourfit.py (ionex TEC file in use)
    4. `fourfit4` was passed the '-k' option to force it to produce Mk4 format output
    5. Scans with SNR<10 were cut, as well as scans with G or H codes, and short baselines (Oe-Ow or Ws-Wn)
    6. Compare differences of a few select quantities (q-codes, residual mbd, dTEC, phase, etc.)

First look: alist Q-code summary

MIT HAYSTACK OBSERVATORY

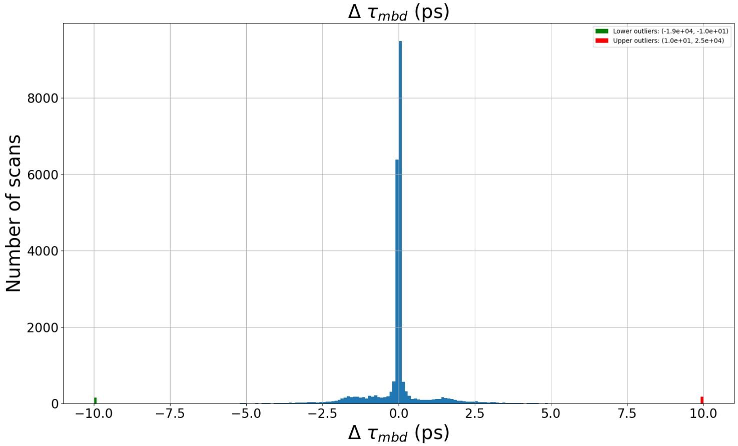
1. Rough fringe quality overview as evaluated by `fourfit3` vs. `fourfit4` for VR2404

Quality code:	G	H	0	1	2	3	4	5	6	7	8	9	?	Total
HOPS4	1249	1307	1940	2	3	22	62	308	740	2313	6893	16415	0	31254
HOPS3	1291	239	3081	0	5	10	40	159	630	2198	7108	16503	0	31264
Difference	-42	1068	-1141	2	-2	12	22	149	110	115	-215	-88	0	-10

1. Behavior is for the most part quite similar, small fraction ($\sim 1\%$) of scans are shifted to lower quality codes (9,8 $\rightarrow$ 7,6,5)
2. Handful of scans (10) are missing? – we need to identify the origin of this.
3. Main qualitative difference is that `fourfit4` prefers to assign an H-code over a quality code of 0 or a G-code, this is not entirely unexpected due to known `fourfit3` behavior.

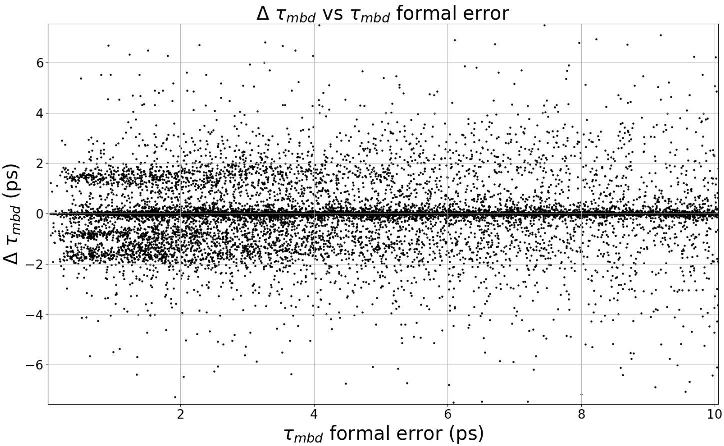
Difference in residual multi-band delay

MIT HAYSTACK OBSERVATORY



Majority of results are within $\pm 2.5ps$, but there are some outliers and some curious 'side lobes'. Why?

$\Delta\tau_{mbd}$ vs. formal error



Difference in τ_{mbd} vs. the formal error as reported by `fourfit3` (zoomed in view). Outliers not shown. Notice the curious banding in this plot.

$\Delta\tau_{mbd}$ vs. formal error cont.

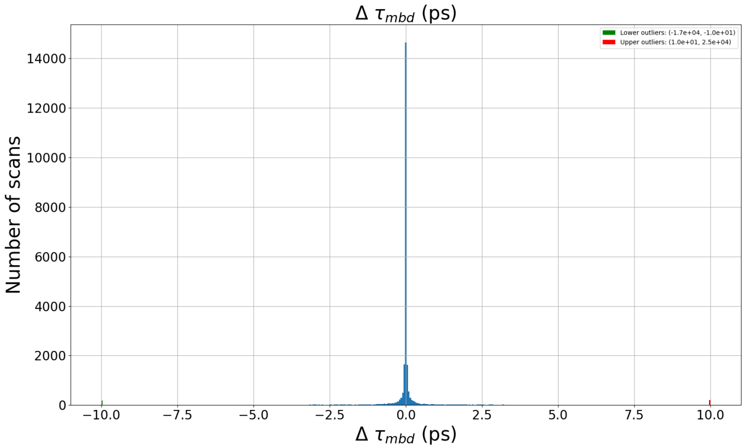
MIT HAYSTACK OBSERVATORY

Two things discovered which contributed to the observed banding:

- RFI spike in channel (r) at station Ow which lead to divergent behavior between `fourfit3/4`.
- There was a bug in the window caching mechanism in the SDB/dTEC search in `fourfit4`

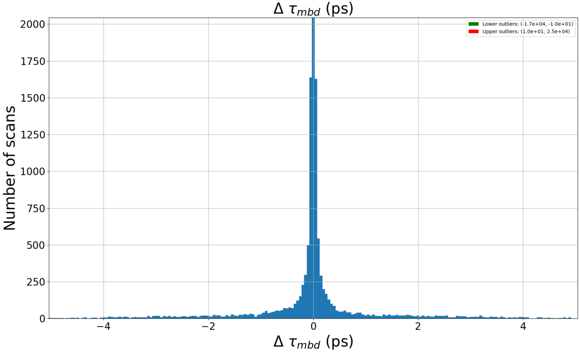
$\Delta\tau_{mbd}$ distribution

MIT HAYSTACK



$\Delta\tau_{mbd}$ distribution after removing channel 'r' from Ow and fixing window bug.

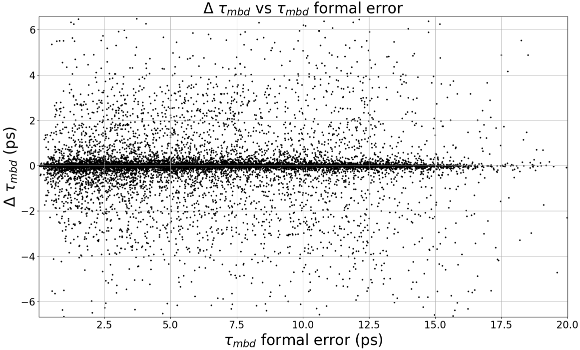
$\Delta\tau_{mbd}$ distribution (zoomed)



Zoom into distribution of $\Delta\tau_{mbd}$.

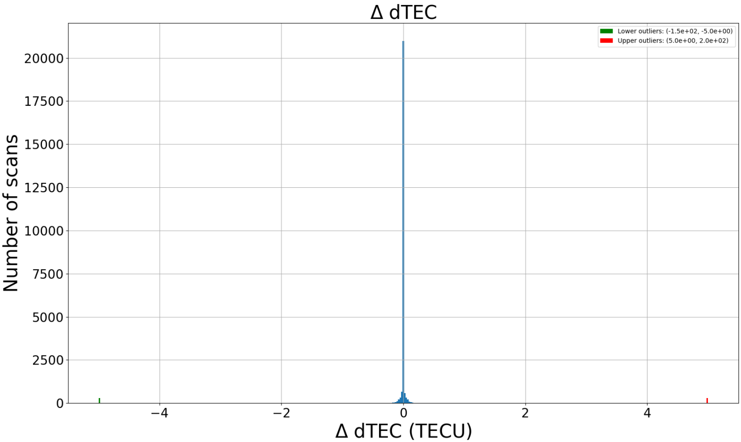
Difference in residual multi-band delay

MIT HAYSTACK OBSERVATORY



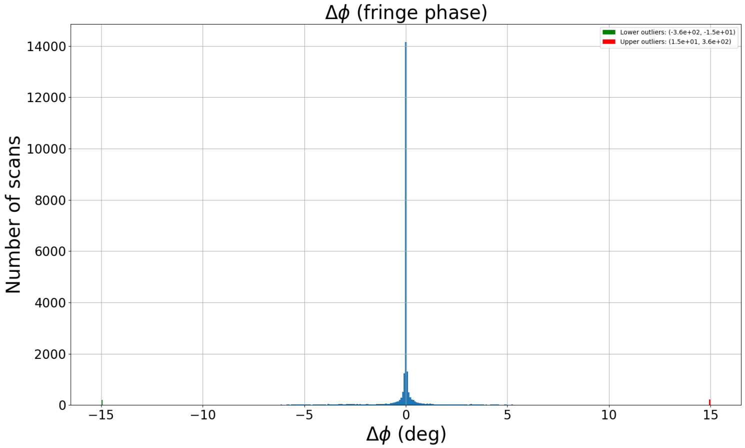
Bands eliminated, but still need to determine the cause of the remaining difference in τ_{mbd} seen in the small fraction of scans which are the source of the scatter in this plot.

Difference in dTEC



Difference in residual phase

MIT
HAYSTACK
OBSERVATORY



Majority of results are within $\pm 5^\circ$.

Comparison test conclusions

MIT HAYSTACK OBSERVATORY

1. A large majority of results are effectively the same between the two implementations especially for the group delay, however, there is a small fraction of outliers
2. These outliers need to be understood, as they may indicate non-equivalent treatment of the data between the two implementations.
3. It is also possible that some degree of numerical instability in the fringe-fitting algorithm may be present which could cause small initial differences to become magnified
4. Where possible these differences should be eliminated
5. The net effect of any remaining differences between the results of `fourfit4` and `fourfit3` on the final geodetic results still needs to be explored and quantified

Future outlook

MIT HAYSTACK OBSERVATORY

1. The build out of full (existing) capabilities is nearing completion
2. HOPS3 isn't going away (existing code is captured and will be distributed alongside HOPS4)
3. New capabilities are ready to be explored – alternative fringe fitting algorithms, ionospheric and source structure correction techniques, etc.
4. Lots of on-going testing and verification to continue.
5. A **lot** of work on documentation needs to be done!

Thanks for listening!