Convenient Fringe-Fit Data Storage in Python Dictionaries for VO2187 Experiment

Leonid Benkevitch, May 13, 2025

Motivation

In Very Long Baseline Interferometry (VLBI), the correlated and fringe-fit data are stored in the Mark4 format, a 2-level directory tree. The top directory name is a 4-digit number, below it contains directories named <doy>-<time>[<letter>]. Each of these directories holds the data files for a scan: the cross- and auto-correlated data, and the fringe-fit data. For example:

2187

├── 187-1803b

├── 0458-020.3HJQAB

├── E..3HJQAB

├── EE..3HJQAB

├── G..3HJQAB

├── GE..3HJQAB

├── GE.X.6.3HJQAB

├── HE..3HJQAB

├── HE.X.2.3HJQAB  
 . . . . . .

The file names only have the baseline letters. The fringe-fit file names also have “.X.”. In order to access the information stored in the files the package HOPS is used. For example, here is how to access data items in a single file "/home/benkev/Work/2187/scratch/Lin\_I/2187":

$ cd /home/benkev/Work/2187/scratch/Lin\_I/

In Python or IPython:

from vpal import fringe\_file\_manipulation as ffm

f\_obj = ffm.FringeFileHandle()

f\_obj.load("GE.X.6.3HJQAB")  
src = f\_obj.source # Celestial source

phase = f\_obj.resid\_phas # Residual phase

dtec = f\_obj.dtec # Differential Total Electron Content

ttag = f\_obj.time\_tag # Time or measurement, seconds

mbdelay = f\_obj.mbdelay\*1e6 # Multiband delay, us

sbdelay = f\_obj.sbdelay\*1e6 # Single-band delay, us

snr = f\_obj.snr # Signal to noise ratio

The data in a single file are for a specific polarization correlation product (for linear polarization, one of XX, XY, YX, or YY). It can be found with the following code:

import hopstestb as ht

pp\_list = \

ht.get\_file\_polarization\_product\_provisional("GE.X.6.3HJQAB")

pp = pp\_list[0] # Polarization Product

With such an organization of information, selecting multiple data that meet several criteria (for example, the time interval of scanning a particular source for baselines that make up a triangular closure) becomes quite non-trivial. In addition, a lot of time is spent opening multiple files.

The data needed for a particular analysis can be extracted from all of the fringe-fit files only once. The extracted data should be stored in data structures that provide convenient access. For example, it should be possible to access the whole data cluster related to a celestial source, or a time tag, or a baseline, or a baseline triangle.

Python has a built-in dictionary type currently implemented as a hash table. “Multidimensional” dictionaries (or dictionaries of subdictionaries of subsubdictionaries …) are the perfect containers for storing the fringe-fit Mark4 data and for accessing it conveniently.