HOPS MSRI Development Notes

Geoff Crew and John Barrett
MIT Haystack Observatory

Version 0.1, August 13, 2019

Contents

1 Introduction

1.1 EHT MSRI Project

The Event Horizon Telescope (EHT) has launched an MSRI (Mid-Scale Research Initiative) project which looks to develop the technologies needed for a second-generation Event Horizon Telescope (EHT). This project is looking at all parts of the existing system and looking to scale it up to a larger and more capable array. A significant component of the telescope is the software needed to properly operate, reduce and analyze the data taken by the member telescopes. It is unfortunately true that the development and maintenance of such critical software is historically a side effect of other programmatics—*i.e.* it is easier to get resources to build something than it is to obtain resources to properly analyze the data from it. In this case, however, the MSRI project is unusual in that it specifically allocate resources at Haystack to address this and move the current collection of EHT-required software into the 21-st century. This document is a start on organizing the thoughts in concert with the accepted MSRI proposal.

The next generation EHT is looking to support $\sim$20 stations, with wider bandwidth (128 Gbps has been mentioned—meaning four dual-polarization, 4 GHz bands), although support for greater bit depth is potentially also of interest should recording media be available. The EHT to date has had an annual cycle of observing; but the ability to make more observations per year has been mentioned. Without a substantial increase in analytic support manpower, all of this implies that processing with HOPS must be made, smarter, more automatic, more robust and easier to use.

At the same time it is critically important to recognize that the existing HOPS (Haystack Observatory Postprocessing System) framework is required by the geodetic community for current operations as well as planned development to their generation of geodetic stations ([1]). At the same time, the geodetic analyses struggle from some of the same constraints the EHT is facing. Thus the new design and implementation plan must repect the geodetic needs—at the end of the process, there are not likely to be resources to support divergent HOPS packages. In this document we out the basic plan for development of the package and identify the work that will transpire under the MSRI program.

1.2 History of HOPS

At the heart of the VLBI technique is the correlation of the raw station data using either dedicated hardware or software to find the correlated signal from the cosmic source. The correlation is manifest as an interference fringe that changes in an expected way as the Earth rotates. This is a simple, but (computationally) expensive process that requires good, but nevertheless approximate, models in order to obtain useful a useful fringe. Thus some post-correlation processing software is required to analyze the fringes to obtain scientifically useful results.

The current Haystack Observatory Postprocessing system (HOPS) was born from the efforts of Alan Rogers in the late 70's with a program called FRNGE which was written in Fortran and designed to be efficient on an HP-21MX (later renamed HP-1000) minicomputer. With improvements in hardware and software, a rewrite of the toolset was launched in the early 90's by Colin Lonsdale, Roger Cappallo and Cris Niell as driven by the needs of the geodetic community. The basic algorithms were adopted from FRNGE; but there was a complete rewrite of the code into (K&R) C and substantial revisions of the i/o, control and file structures resulting in the framework of the current HOPS system. This was followed by a substantial effort in the early-mid 00's to develop tools for optimizing SNR and deriving correction factors for data with imperfect coherence, based on analysis of amplitude with coherent averaging time. While there is no definitive, published, HOPS reference in the literature, the Mark 4 Correlator paper ([2]) touches upon the basic implementation available by this time. Further evolution in the late 00's was provoked by the re-emergence of software correlation (DiFX, [3], [4]), and in the 10's by the needs of EHT-scale mm-VLBI which brings us to HOPS in its current form.

Acknowledging its geodetic heritage, HOPS was optimized for precision on per-baseline delay and delay-rate measurements which are the raw material for the geodetic analysis programs. Consequently, it is somewhat light on support for some routine calibration processes found in some other astronomical software packages (e.g. AIPS or CASA). Nevertheless, it provides a good framework for the reduction and analysis of mm-VLBI data, where the vagaries of atmospheric effects require ever more specialized processing to harvest significant astronomical results.

For the needs of the EHT Campaigns of 2017 it was decided to augment the existing HOPS package with some python-based packages in order to create a pipeline for the initial reduction of data. (See [5], and [6], [7], and [8]). The EHT also looked at data reduction with other packages. There were initial surveys of options in 2015 (Leiden workshop) and 2016 (Nijmegen) which led to a focus on HOPS, AIPS and CASA as the three viable options to pursue for 2017. The Calibration and Error Analysis working group of the EHT was able to demonstrate consistent results between the three packages; ultimately production processing via HOPS for the EHT was the winning solution. In this continued development of HOPS, we shall assume that HOPS alone must be capable of the full analysis; but we should also be mindful that options to move the data to AIPS or CASA must at some level exist.

1.3 The Basic Plan

So our charge from the MSRI project is to update the existing HOPS and EHT Pipeline system and produce a better organized, more useable and flexible system for the next decades. A significant constraint is that HOPS is currently the critical analysis package for geodetic use. This is especially true with the introduction of the VGOS system (Reference...). Since we will not ultimately have resources to support multiple systems, it is a de facto requirement that the next-generation HOPS system work seamlessly with all of its user community (*i.e.* EHT/astronomy and geodetic). Given all the validation work that went into the HOPS/EHT Pipeline for the 2017 data analysis, this is not a real restriction. The EHT will demand the same results from the old as well as the new HOPS.

This document will evolve into a full development plan once in the months ahead.

2 Re-Design Considerations

2.1 Outline of Discussion Topics

This section contains an outline to help organize thought. For clarity, the discussion is shifted to paragraphs of a subsequent subsection.

1. Software features and design (3)
   - (a) General Architecture (3.1)
     - i. Language choice: C/C++ with python (3.1.1)
     - ii. Build system and version control (3.1.2)
     - iii. Options for parallel processing (3.1.3)
     - iv. Interactivity vs. batch processing (3.1.4)
     - v. External package dependencies (3.2)
   - (b) Imports from Correlator Output (3.3)
     - i. DiFX (Swin) output (3.3.1)
     - ii. File conversion: difx2mark4, difx2fits, etc. (3.3.2)
   - (c) Exports to subsequent analyses (3.4)
     - i. Export to imaging (UVFITS) (3.4.1)
     - ii. Export to geodetic reductions (CALCSOLVE) 3.4.2
   - (d) HOPS file specifications (3.5)
     - i. Mark4 file types 3.5.2
     - ii. python wrappers (mk4) 3.5.3
     - iii. alist format 3.5.4
     - iv. fourfit control file 3.5.5
     - v. vex2xml and vex2.0 3.5.6
   - (e) New Objects (3.6)
     - i. equivalent to above
   - (f) Algorithm specification (3.7)
     - i. baseline-specific delay/delay-rate fitting (3.7.1)
     - ii. global-fringe fitting (3.7.2)

iii. spectral line fringing (3.7.3)
iv. vgos ionospheric fitting (3.7.4)
v. coherence fitting (3.7.5)
vi. weak fringe searching (3.7.6)
vii. pulsar folding/searching (3.7.7)
viii. space-based problems (3.7.8)
ix. Polconvert (3.7.9)

(g) Calibration specification (3.8)

i. phase calibrations (EHT v Geodesy) (3.8.1)
ii. derivation of manual phase cals (3.8.2)
iii. application of a-priori phase-cal. (3.8.3)
iv. instrumental bandpass-correction (3.8.4)
v. atmospheric phase correction (3.8.5)
vi. polarization dependent corrections (3.8.6)
vii. ionospheric corrections (3.8.7)
viii. source structure (3.8.8

(h) Infrastructure (3.9)

i. output messaging (3.9.1)
ii. data utilities (3.9.2)
iii. performance monitoring and profiling (3.9.3)
iv. averaging (3.9.4)

(i) Data inspection and visualization (3.10)

i. inspection (corAsc) and import from ascii (3.10.1)
ii. What do we do with the fourfit plot? (3.10.2)
iii. Interactive tools like aedit? (3.10.3)
iv. Alternate visualization options? (3.10.4)

(j) New Libraries (3.11)

i. equivalent to above

(k) New Programs and scripts (3.12)

i. equivalent to above

2. Development schedule (4)

(a) Pre-requisites (4.1)
(b) Re-use of existing code? (4.2)
(c) Unit test coverage (4.3)
(d) Testing/validation (4.4)
(e) Other considerations (3.1)

i. What can be worked on in parallel? (4.5.1)
ii. What must be done sequentially? (4.5.2)
iii. What dependencies exist between modules? (4.5.3)
iv. What other resources may be used (4.5.4)
v. What parts must be supported by the geodetic team (4.5.5)
vi. Clarity Descope options (4.5.6)

2.2 Inputs and Resources

The MSRI award to Haystack calls for approximately one FTE of effort spread across 4 years. In principal additional resources for testing, validation and interfacing with other resources in the EHT will be available. (At the very least, the Calibration and Analysis working group [C&E-WG] will be involved in specification and testing.)

2.3 Proposed Timeline

The preliminary proposal timeline called for the following general schedule:

Q01 Obtain input for feature definitions/requests
Q02 Define software requirements
Q03 Selection of architecture and file system, begin porting algorithms
Q07 Review progress
Q10 Finish porting algorithms
Q12 Verification of new software package agains old
Q14 Validation on new wideband data
Q16 Final release

The work planned for the first two quarters will almost certainly result in some modifications of the general schedule, so we do not plan to fix the schedule at this point. We anticipate that the general framework for the new HOPS will remain (we have been discussing this for several years), but some of the priorities and features may very well require adjustment by the start of Q3.

3 Commentary on Software Features and Design

The following sections provide some background and initial design discussions for the topics listed in the outline.

3.1 General Architecture

3.1.1 Detail on language choice

Several aspects need to be taken into account when deciding on a choice of programming language for this project. Namely, some of these are:

1. Availability of software developer expertise.
2. The inherent performance attainable with a specific language.
3. Availability of high performance open source utility libraries for math, I/O, etc.
4. The primary language of the existing code base (C).
5. The accessibility and ease of extensibility of the project by users with varying levels of experience.

Obtaining a reasonable balance between these considerations is difficult with a single language. Therefore it may be desirable to consider a multi-language project, wherein the base computation layer is handled within C/C++, but additional data manipulation can be done via optional Python plugins embedded within the appplication or independently by external Python scripts which have access to some of the underlying application libraries. C++ is a reasonable choice given current personnel, and the possibility of reuse of portions of the existing code base in C. It also allows for the use of a wide variety of open source libraries, bot C and C++ (not least of which is the built in standard library which provides access to a wide collection of basic data types (strings, vectors, maps, etc) and algorithms (searching, sorting, etc) which reduces the required amount of maintenance of internal code and reliance on external libraries. Further augmenting C/C++ libraries with inter-language communication to Python can be done via a wide variety of mature tools (ctypes, boost.Python, SWIG, etc.), and may increases the ease at which outside users can augment the software. Adopting C++ would allow a easier path to memory management (currently handled rather painfully in the existing HOPS code base).

Note that Python 2 is no longer supported, so to be clear, all new development will be Python 3.

3.1.2 Build system version control

We have a working SVN repository and the system is currently built with autotools. Do we want to continue using SVN or move to git? Do we want to continue using autotools or migrate to cmake? We can consider both, but there is no rush or a compelling reason to do either, except perhaps for long term maintenance.

3.1.3 Options for parallel processing

The existing fringe-fitting process is largely a data-parallel process operating on individual baselines with no inter-process communication. This lends itself easily to simple parallelism using multiple independent processes (SPMD), which has been exploited [5] to deal with the EHT data volume. However, this approach eliminates the ability to simultaneously fit for global or station-based parameters and requires multiple iterations in order to apply successive calibration/corrections. Therefore, if some calibration tasks are to be done simultaneously with fringe-fringe fitting, this will require both a substantial architectural change from the current fitting algorithm, but also necessarily reduce the degree of (simple) parallelism available. To accommodate this, some parallel processing will have to be addressed within the application. There several architectural options from which to choose to provide support for differing levels of parallelism. A limited table of these options is detailed in the following table:

Name	Classification	Hardware Scalability	Effort
OpenCL/CUDA	SIMD	Single machine, CPU/GPU	Low-Medium
pthreads	MIMD	Single machine (CPU)	Medium
c++11 threads	MIMD	Single machine (CPU)	Medium
OpenMP	SIMD	Single machine (CPU)	Low-Medium
OpenMPI	MIMD	Multiple machines (CPU)	High

Note that a proper re-design of the data types and low-level algorithmic codes should make it relatively straightforward to develop various parallel versions of the fringe fitter or analysis tools, but this should probably be done only after profiling a single-threaded version of the code, to determine where the most crucial bottlenecks are.

3.1.4 Interactivity vs. batch processing

The existing HOPS processing is batch-oriented for fringing, but interactive at the `aedit` stage. . . .

Batch-oriented fringe production works well at the later stages (after tuning) but at the initial analysis stages, some flexibility is needed. This is particularly true as the amount of information contained in broadband fringing greatly exceeds what will fit on one page. And expansion to multiple pages places more burden on the analyst—too much to look at.

In the analysis stage (e.g. `aedit`) some degree of sophistication is needed in the selection and display of data. Again, the volumes of data to be understood will require smarter software.

3.2 External package dependencies

In order to leverage existing open source software some decisions should be made relatively early on about which external (maintained by outside unrelated entities) should be required. For example the current library used by HOPS for executing FFT's is FFTW3, which is required and not optional, likewise, for creating fringe plots the package pgplot is required and not optional. Wherever possible requirements should be made optional in order to keep complexity low and reduce reliance on external maintainers, but to make the best use of existing resources some minimal subset needs to be decided upon.

3.3 Imports from Correlator Output

3.3.1 DiFX Outputs

At present DiFX is the de-facto correlator for the EHT. However, there are other correlator codes (e.g. SFXC or CorrelX) that may be useful in the future. Also the so-called Swinbourne output format from DiFX may evolve. Thus we need a define a suitably generic model for the correlation output that will be adaptable to possible future development.

In general this framework includes inputs to the correlation (e.g. VEX and V2D) as well as correlation setup files (e.g. calc).

3.3.2 File conversions

The current HOPS pathway from DiFX is difx2mark4 which is very specific to DiFX and (the current) HOPS. For analyis in AIPS or CASA, the difx2fits program is currently used.

Aside from file input, there is generally a desire to migrate data between the HOPS world and the AIPS/CASA world. Thus it is sensible to provide pathways from the new HOPS file structure to these other tools.

3.4 Exports to Subsequent Analyses

3.4.1 UVFITS

The current C&E-WG processing pipeline ends with the generation of the UV flavor of FITS (UVFITS). That tool should properly be updated to work with the new fileset and added to the HOPS tool collection.

3.4.2 CALCSOLVE

The geodetic analysts current take delivery of HOPS fringe results for import into their CALCSOLVE program which solves for Earth Orientation Parameters and other geodetic products. The current delivery format is native HOPS mk4 file format. This capability must be preserved until they can adopt their input library to use the new HOPS.

3.5 HOPS file specifications

3.5.1 File Types

HOPS makes use of a filesystem, and every file consists of representations of objects. A uniform plan for migrating these from what we currently have to something more sensible is a project that decomposes into the various types discussed in the next sections. This design task consists of all of the general considerations before the details of the subsequent tasks.

3.5.2 Mark4 file types

The existing "Mark4" filesystem was (believe it or not) an improvement of the previous "Mark3" filesystem which was closely wedded to (a now extremely outdated) filesystem for an HP filesystem. The modularity of the Mark4 data types is not particularly in question. However the binary format is bigendian and wedded to the C structures and other capabilities of the UNIX system that was available at the time (SunOS, a precursor of Solaris). Since that time a number of viable archival data formats have been created and are in general use. A migration of the Mark4 types to an HDF5-based fileformat is considered to be the desirable option at this point. The libraries from that package are well tested and essentially ready to use.

3.5.3 Python Wrappers

One of the first steps of the C&E-WG was to create Python wrappers for the Mark4 types so that some manipulations could be done directly in Python. This is now considered an essential feature and it should be natively supported by **(new)HOPS**.

3.5.4 Alist

Following fringing, for the data analysis stage, HOPS provided a "oneline" fringe format (an aline) which could be manipulated by several tools (e.g. `alist` and `aedit`). The amount of information that is desirable to be captured on the aline has grown to the point of incomprehensibility, but the need for such a thing remains. The precise solution is likely a topic of study as there are a number of alternatives.

3.5.5 Fourfit Control File

The fringing process may be controlled (almost) equivalently on the command line or via a "control" file. This file has a peculiar format supporting a limited set of logic contructs for setting various parameters. In a package making specific use of Python, it is sensible do discard the existing control file machinery in favor of a native Python format and conventions for command line adjustments to it.

3.5.6 VEX file parsing

The VLBI Experiments are specified in a rather arcane VEX file which is currently stuck at verion 1.5. The community has identified a version 2.0, but it has not actually been implemented. For use at ALMA, we developed and recently added to HOPS a VEX2XML parser that allows the VEX file to be parsed to XML and then standard XML libraries may be used to extract information. We propose to adopt 2.0 features as they become useful to the EHT, and work through the VEX2XML parser tool.

3.6 New Objects

The mark4 fileset stores data in number types: the 100 series for correlation products, the 200 series for fringes and the 300 series for station calibrations. The list grew in a somewhat ad hoc fashion and was decidedly driving by the needs of analyzing data from the old hardware correlator. The list of types needs to be reviewed and re-organized for the modern era.

3.7 Algorithmic specifications

3.7.1 Baseline-based delay/delay-rate fitting

At the heart of fourfit is the baseline-based delay/delay-rate fitting. Fourfit expresses as both a so-called "single-band delay" (SBD, an average over the recorded channels) as well as the "multi-band delay" (MBD, over the full band). This algorithm must of course be preserved, but it should be noted that it may be decomposed into the data calculations and the fitting program. The current algorithm also allows for an ionospheric fitter. These parts must be decomposed and put in library functions for more flexible use.

3.7.2 Global Fringe Fitting

In particular, given a saner organization for the fitting, it should be possible to provide the (more conventional, as provided in AIPS and CASA) global fringe fitter which assigns results on a station basis. This requires a(t least one) least-squares fitting technique to be implemented.

3.7.3 Spectral Line Fringing

There are developments in mmVLBI that should allow work with spectral line (i.e. narrow) sources; support for this capability should be in place in the **(new)HOPS**.

3.7.4 Ionospheric Fitting

VGOS currently fits for the total electron content (TEC) of the ionosphere as part of the high-delay precision fits. This is likely not needed at the high frequencies the EHT uses, but it definitely must be preserved.

3.7.5 Coherence Fitting

The current HOPS tool `cofit` allows one to identify the shortest averaging interval with the maximum SNR. The algorithm and plotting artifacts should be continue to be supported as the tool is still useful.

3.7.6 Weak Fringe Searching

The current HOPS tool `search` allows one to examine the two-dimensional delay and delay-rate space to ascertain whether weak peaks are likely to be real or not. The tool is still of some use and should be retained.

3.7.7 Pulsar Folding/Searching

Support for folding data on a known pulsar period, or for searching for pulsar periodicities in data was at one point contemplated for HOPS. It may be useful to consider this in the new architecture.

3.7.8 Space Based Support

If there is a future possibility of space based radio telescopes participating in the EHT network, this may require some additional features in the fringe fitting software, such as compensating for higher order delay residual terms (delay-acceleration) in addition to the linear delay/delay-rate model used for ground based stations.

3.7.9 PolConvert

The ALMA observatory uses a linear, rather than a circular polarization basis. The current practice is to follow the correlation process (into a mixed basis) by a polarization conversion step (using the tool `PolConvert`). In principle this step could be carried out in the post-processing stage within **(new)HOPS**. The architecture should support this.

3.8 Calibration Specification

3.8.1 Phase Calibration

The EHT generally has used manual phase cals (no other alternative). The geodetic sites generally have a pulsed tone phase cal system. Depending on whether a pulsed system can be developed for the EHT, it may in principal need to use both methods.

3.8.2 Manual Phase Calibration

Lacking a pulse phase cal system at all of its observatories, the EHT generally relies on a manual phase cal procedure. There are several scripts that have been written to estimate the manual phase cals (at each station) on one bright scan and then place these phases into the control file. Likewise, delays may be estimated from a single scan and placed in the control file. Scripts to continue to do this must be supported in the **(new)HOPS** architecture.

3.8.3 Pulse Phase Calibration

Pulse phase calibration is a method of compensating for the time varying phase and delay response of a telescope's receiving system. This is done by injecting a train of sharp pulses (a Dirac comb) which are synchronized with the site's reference clock. This train of pulses is equivalent to set of equally separated tones in frequency space. Since the tones are known to be in phase at the point of injection, it is then possible to monitor the frequency dependent phase changes made to the incoming signal by the receiving system by tracking the accumulated phase of each tone. This technique allows for the elimination of the (possibly time-varying) dispersion introduced by the receiving system.

Typically this calibration process is applied to the data in two steps after it is recorded. The first step is the extraction of the phase calibration tone phases. This is primarily done by the correlator by calculating the in-phase and quadrature components of the stations' signals at each of the discrete tone frequencies. The second step is done during the fringe-fitting process, where each stations tone-phase data is applied to correct the correlated signal. When there are many tone available across the correlated bandwidth, this process also allows for the correction of instrumental delays (higher order terms are not currently considered). However, as is often the case, the pulse calibration data is not perfect and it can be significantly contaminated by RFI or weak tones. The current implementation of (multiple tone) phase-calibration in fourfit is fairly robust, and admits some ability to mask bad tone data. However, there is ample room for improvement in the automatic flagging of bad phase calibration data, which would be especially useful in the case where the data quality varies across both the time and frequency domain.

3.8.4 Instrumental Bandpass Calibration

It should be possible in **(new)HOPS** to perform an instrumental bandpass correction as is available in other packages. Here one or more scans on a bright calibrator may be used to establish the per-station deviation of the bandpass from the optimal flat response. These derived bandpass solutions may then be applied to adjust the correlation amplitude as a function of frequency. Such a bandpass correction may also be fully complex (i.e. phase adjustments as well).

3.8.5 Atmospheric Phase Calibration

The currently implmentation of fourfit allows from some limited ability to handle atmospheric phase calibration. Currently, this is done via the introduction of ad-hoc phases, which are applied independently from the formal fitting procedure for delay/delay-rate. In the existing EHT calibration pipeline, these ad-hoc phases are estimated (after data-flagging and some bandpass correction have been applied) from data residuals after the fringe-fitting. The smoothed phase corrections estimated from the residuals must then be exported to a ad-hoc phase file in order to be applied on the next pass. This process (while generic) is time consuming and requires the presence of a reference station with good SNR to the majority of stations in the network. Therefore, it desirable to have a dedicated algorithm to estimate and apply atmospheric phase calibration within the fringe-fitting process itself.

3.8.6 Polarization Calibration Corrections

In the current EHT analysis, network calibration techniques are used to self-calibrate the array for polarization work. This is an area to discuss with the other working groups what form of support in **(new)HOPS** would be most effective. In addition, good estimates of station-based polarization calibration parameters also enables the coherent summation (Stokes-I) of individual polarization-products resulting in higher SNR observations. This is also of great interest for geodetic observations, and important for observations between stations with mixed polarization types (circular-linear).

3.8.7 Ionopheric Calibration Corrections

While the ionosphere does not typically play much of a roll at the high frequencies at which the EHT is observing (230, 345 GHz), knowledge of the ionosphere is crucial for obtaining good results during geodetic observations that occur at lower frequencies (2-14 GHz) where the dispersion it causes is much stronger. Currently, fitting for the(line-of-site) differential total electron content (ΔTEC) of the ionosphere can currently be done from geodetic VLBI observations themselves during fringe-fitting on a single baseline. However, the current implementation could be strengthened if the fringe-fitting architecture were extended to allow for the possibility of simultaneously fitting for the line-of-site TEC associated with station, as a station (rather than baseline) based quantity. This would help in the reduction of non-closing Δ TEC errors, and which can currently only be done in an ad-hoc manner. Additionally, fitting for the ΔTEC is sensitive to residual instrumental phase effects which can be difficult to detect unless examining data from multiple single baselines/scans.

3.8.8 Source Structure Corrections

One explicitly geodetic feature that may be desirable in the **(new)HOPS**is the possibility to introduce phase and delay corrections to compensate for source structure given an image model of a non-point like radio source. Having this ability would eliminate a significant source of systematic error in geodetic delay observables, and allow for a larger catalog of usable (bright) sources for geodetic observations.

3.9 Infrastructure

3.9.1 Messaging

HOPS currently passes commentary to the user via a uniform messaging service. This allows the user to turn on verbosity or run completely silently. This should be preserved, but due to the potential volume of such messages, the control should become finer grained—i.e. the user should be able to specify what portions of code are making comments.

3.9.2 Utilities

HOPS has a few utilities for date or geometry calculations—these should be reviewed and clearly supported.

3.9.3 Performance

HOPS has a performance monitoring system built into the existing code. This should be retained and augmented by more sophisticated profiling tools.

3.9.4 Averaging

Time averaging of data was added ad hoc to several of the tools; we propose to build this capability in directly with native support.

3.10 Data Inspection and Visualization

3.10.1 To/From ASCII

Every bit of data that HOPS works with should be available in a human-readable form. So every object in the new design should support methods to export to or import from some human readable format.

3.10.2 The Fourfit Plot

The current fringe fitter, `fourfit` produces a single page summary of the fringe result for every baseline. In the high-bandwidth modern era, one is hard pressed to capture everything in a readable format. A single-page format is still desirable, as is the ability to page through multiple plots (i.e. as can be done with `fplot`), but some controls over the information that is displayed is probably desirable. Thus in **(new)HOPS**, there will still be a standard page, but one may also create custom formats which may be more useful for some experiment setups.

3.10.3 Interactive Tools

The post-fringe processing in HOPS begins with `alist` which provides one line summaries of every fringe. These "alist" files may then be used to make data-quality selections or support inspection (`fplot` of errant fringes). This capability must be retained, but it can almost certainly be better implemented in Python, which would make it more extensible.

3.10.4 Alternative Visualization

Additional visualizations of data should be provided as these are often useful for understanding subtle issues with data. Again, a Python layer providing access to the underlying **(new)HOPS** data formats would probably be the most elegant solution.

3.11 New Libraries

The existing algorithms should be encoded into new library methods that have clear inputs, outputs and side-effects. (In the current HOPS, there are many side-effects to global variables, so the methods cannot be directly reused as coded.)

3.12 New Programs

With the new architecture in place, versions of the existing programs (e.g. `fourfit`, `alist` and `aedit`) should be provided along with new, improved tools for the desired new capabilities.

4 Development Schedule

4.1 Pre-requisites

Probably the most critical thing is to evaluate whether there are any "gotchas" in using HDF5 (the proposed new standard). Some decisions about external package dependencies must also be made.

4.2 Re-use of existing code

Many portions of the existing HOPS code can and should be incorporated into the new software. In order to do so, some reorganization will be needed. This will consist of identifying and isolating useful functions in the existing code base, so they can be used independently and compiled into separate utility libraries. These utility libraries can than be linked in as needed.

4.3 Unit test coverage

During development a clear set of unit tests for various functions and libraries will need to be developed concurrently with the software. The purpose of this is two-fold. First, the implementation of unit test cases allows one to validate the correct operation of individual compontents, and secondly, it allows a rapid identification of problems and bugs that may be introduced during development before they become problematic. The unit tests for each component will necessarily be unique to each item they are testing, but should be operable on the test component independently with a minimum number of dependencies.

4.4 Verification and Validation

Apart from the unit testing of individual software components, it is critically important to verify the correction operation of the complete software package. This should be done using a combination of synthetic and real data. Use of synthetic data is desirable since it should be possible to make concrete calculations about the output of the software, whereas it is also imperative to test the performance of the software on read data which may exhibit pathologies that are not possible to replicate artificially. Validation should also done in comparison to the original HOPS software, to ensure no functionality is lost or degraded.

4.5 Other Considerations

4.5.1 What can be worked on in parallel?

Once the general design is in place, many code modules may be worked on in parallel by independent developers provided well posed unit tests and interfaces are defined and implemented.

4.5.2 What must be sequential?

The main architectural decisions must be made early, the programs that put all the pieces together will come later.

4.5.3 What are the module dependencies?

The algorithmic modules depend on the data access modules, but the internal data representation should not rely on having any particular data access (I/O) library installed. Whenever possible dependencies on external packages and libraries should be made optional, but not necessarily in a feature preserving way. For example, it is natural that in order to access data in HDF5 files, an HDF5 library must be available during compilation, but the lack of such a library should not keep the user from manipulating Mark4 or other file types so long as their prerequisites are met. Likewise, this should be done for other libraries when possible, e.g. visualization, where it should be possible to run the fringe fitting procedure with no visualization package available at all if desired.

4.5.4 What other resources may be used?

We expect to have access to some geodetic resources to support maintenance of traditional HOPS capabilities into the new package. We expect that EHT members from other working groups and other institutions than Haystack will be available to test aspects of the new package as they become available. Should other developers wish to provide additional modules we would be open to that.

4.5.5 What parts must be supported by the geodetic team?

If the EHT does not adopt a hardware phase cal system, the geodetic team would have to provide support for this feature/port from existing code.

4.5.6 Clarity on descope options

more to come

4.6 Detailed Timeline

In this section we provide some tables that provide inputs for Gantt-style charting. However, in view of the fact that the first months of the project call for a refinement of the plan in terms of features, requirements and specifications…such a chart must at best be considered a best effort today and almost certainly to be revised within the first year of development.

Since the manpower is likely to be distributed across parts of multiple individuals (at least one of whom is to be hired), this version considers 1 FTE doing the complete job.

The meanings of the columns are as follows:

N(umber) is an arbitrary line label for tracking predecessors and successors

Ref(erence) should be to the detailed commentary of Sec 3 or equivalently, Sec 2.1

Type the type of task, see below

Topic should be some short title for the task

Sta(rt) should (at this point refer to quarter in which the work starts

Eff(ort) is a number of man-weeks (5 work days)

Pre(decessors) should indicate any predecessor tasks

Suc(cessors) should indicate any tasks which depend on this

G(eodetic indicates a geodetic task supported by other resources

Comments as needed

The items in the tables are either tasks (which involve a time estimate to complete the work), or a milestone to represent conclusion of a stage. For example, the design process generally concludes with a specification prior to implementation, and completion of a set of tasks concludes with a review of the verification or validation results. Thus we use this shorthand (for readability)

(C)onsultation with partners regarding feature requests and or usability considerations and other requirements

(D)esign conducting the design study, trade-offs and convergion to a plan

(S)pecification details of the code element, inputs, outputs, algorithms, methods

(I)mplementation actual coding

(T)esting unit testing for code modules

(Ve)rification refers to verifying that the code does what specification called for

(Va)lidation refers to validating the results of the code against previous work

(R)eview review of testing, verification or validation

The following tables assign ∼142 man-weeks of effort which leaves ∼50 man-weeks of margin which is appropriate at this level of task specification (especially as features not in our minds might yet be proposed). References to quarter for work (Q01 through Q16) are only approximate.

initial planning

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
000	3.1.1	CD	language choices	Q01	1.0	none	013	N	discussion with users
001	3.1.2	CD	build system	Q01	1.0	none	013	N	discussion with users
002	3.1.3	CD	parallel support	Q01	1.0	none	013	N	discussion with users
003	3.1.4	CD	interactivity	Q01	1.0	none	013	N	discussion with users
004	3.2	CD	external packages	Q01	1.0	none	013	N	discussion with users
010	3.6	DS	new objects creation	Q01	2.0	none	013	N	work out object plan
011	3.11	DS	new library creation	Q01	2.0	none	013	N	work out library plan
012	3.12	DS	new program creation	Q01	2.0	none	013	N	progs & scripts
013	3.1	R	review general plan	Q01	0.0	000-012	many	N	new features as well

correlator input/file exchanges

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
020	3.3.1	DS	difx import	Q02	1.0	013	021	N	`difx2mark4`
021	3.3.1	I	difx import	Q04	4.0	020	022	N	recoding
022	3.3.1	T	difx import	Q05	1.0	021	300	N	unit test only
030	3.3.2	DS	file exchange	Q03	1.0	013	031	N	
031	3.3.2	I	file exchange	Q08	4.0	030,300	032	N	coding
032	3.3.2	T	file exchange	Q09	1.0	031	400	N	unit test only
040	3.3	VV	import/export tests	Q12	1.0	400	041	N	`difx2mark4` equiv.
041	3.3	VV	import/export tests	Q13	1.0	040,500	600	N	`difx2fits` equiv.

outputs to analysis

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
050	3.4.1	DS	uvfits	Q02	1.0	013	051	N	
051	3.4.1	I	uvfits	Q08	4.0	050,300	052	N	recoding
052	3.4.1	T	uvfits	Q09	1.0	051	400	N	unit test only
053	3.4.1	VV	uvfits	Q12	1.0	052,500	600	N	with imaging WG
060	3.4.2	DS	calcsolve	Q02	1.0	013	061	Y	(implicit)
061	3.4.2	I	calcsolve	Q08	1.0	060,300	062	Y	recoding
062	3.4.2	T	calcsolve	Q09	1.0	061	400	Y	unit test only
063	3.4.2	VV	calcsolve	Q12	1.0	062,500	600	Y	with GSFC

object implementation

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
070	3.5.2	DS	mk4 types	Q02	1.0	013	071	N	
071	3.5.2	I	mk4 types	Q04	4.0	070	072	N	recoding
072	3.5.2	T	mk4 types	Q05	1.0	071	300	N	unit test only
073	3.5.3	DS	python wrappers	Q02	1.0	013	074	N	
074	3.5.3	I	python wrappers	Q04	2.0	073	075	N	recoding
075	3.5.3	T	python wrappers	Q05	1.0	074	300	N	unit test only
080	3.5.4	DS	alist	Q02	1.0	013	081	N	`alist`
081	3.5.4	I	alist	Q04	2.0	080	082	N	recoding
082	3.5.4	T	alist	Q05	1.0	081	300	N	unit test only
083	3.5.5	DS	control file	Q02	1.0	013	084	N	
084	3.5.5	I	control file	Q04	2.0	083	085	N	coding
085	3.5.5	T	control file	Q05	1.0	084	300	N	unit test only
090	3.5.6	DS	update vex2xml	Q02	1.0	013	091	N	`vex2xml`
091	3.5.6	I	update vex2xml	Q04	2.0	090	092	N	coding
092	3.5.6	T	update vex2xml	Q05	1.0	091	300	N	unit test only
095	3.5	VV	old HOPS regression	Q12	1.0	400	600	N	data exchange tests

algorithmic implementation

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
100	3.7.1	DS	baseline fringing	Q02	1.0	013	101	N	
101	3.7.1	I	baseline fringing	Q04	8.0	100	102	N	recoding
102	3.7.1	T	baseline fringing	Q05	1.0	101	300	N	unit tests only
103	3.7.3	DS	spectral line fits	Q03	1.0	013	300	N	design only
104	3.7.4	DS	ionospheric fits	Q03	1.0	013	300	Y	
105	3.7.4	I	ionospheric fits	Q08	4.0	104,300	106	Y	recoding
106	3.7.4	T	ionospheric fits	Q09	1.0	105	400	Y	unit tests only
110	3.7.5	DS	coherence fits	Q03	1.0	013	111	N	
111	3.7.5	I	coherence fits	Q08	4.0	110,300	112	N	recoding
112	3.7.5	T	coherence fits	Q09	1.0	111	400	N	unit tests only
113	3.7.6	DS	weak fringe search	Q03	1.0	013	114	N	
114	3.7.6	I	weak fringe search	Q08	4.0	113,300	115	N	recoding
115	3.7.6	T	weak fringe search	Q09	1.0	114	400	N	unit tests only
120	3.7.7	DS	pulsar fold/search	Q03	1.0	013	300	N	design only
121	3.7.8	DS	support for space	Q03	1.0	013	300	N	design only
122	3.7.9	DS	polconversion	Q03	1.0	013	300	N	design only

calibration implementation

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
130	3.8.1	DS	phase calibration	Q02	1.0	013	131,140	N	general design
131	3.8.2	DS	man phase cals	Q02	1.0	013	132	N	
132	3.8.2	I	man phase cals	Q04	4.0	131	133	N	recoding
133	3.8.2	T	man phase cals	Q05	1.0	132	300	N	unit tests
134	3.8.2	VV	man phase solving	Q05	1.0	400	500	N	validate autosolving
140	3.8.3	DS	pulse phase cals	Q03	1.0	013	141	Y	
141	3.8.3	I	pulse phase cals	Q08	8.0	140,300	142	Y	recoding
142	3.8.3	T	pulse phase cals	Q09	1.0	141	400	Y	unit tests
143	3.8.3	VV	pulse phase cals	Q12	1.0	500	600	Y	validate with VGOS
150	3.8.4	DS	bandpass cals	Q03	1.0	013	151	N	
151	3.8.4	I	bandpass cals	Q08	4.0	150,300	152	N	coding new functionality
152	3.8.4	T	bandpass cals	Q09	1.0	151	400	N	unit tests
160	3.8.5	DS	atmospheric cals	Q03	1.0	013	161	N	
161	3.8.5	I	atmospheric cals	Q08	4.0	160,300	162	N	coding new functionality
162	3.8.5	T	atmospheric cals	Q09	1.0	161	400	N	unit tests
170	3.8.6	DS	polarization cals	Q03	1.0	013	300	N	design only
180	3.8.7	DS	ionospheric cals	Q03	1.0	013	300	Y	design only
190	3.8.8	DS	source structure	Q03	1.0	013	300	Y	design only

processing support

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
200	3.9.1	DS	processing messages	Q02	1.0	013	201	N	
201	3.9.1	I	processing messages	Q04	2.0	200	202	N	recoding
202	3.9.1	T	processing messages	Q05	1.0	201	300	N	unit tests
210	3.9.2	DS	utility library	Q02	1.0	013	211	N	
211	3.9.2	I	utility library	Q04	2.0	210	212	N	recoding
212	3.9.2	T	utility library	Q05	1.0	211	300	N	unit tests
220	3.9.3	DS	performance support	Q02	1.0	013	221	N	
221	3.9.3	I	performance support	Q04	2.0	220	222	N	recoding
222	3.9.3	T	performance support	Q05	1.0	221	300	N	unit tests
225	3.9.4	DS	averaging support	Q02	1.0	013	300	N	implicit

user interfaces

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
230	3.10.1	DS	ascii inspection	Q02	1.0	013	231	N	`corAsc2`
231	3.10.1	I	ascii inspection	Q04	2.0	230	232	N	coding
232	3.10.1	T	ascii inspection	Q05	1.0	231	300	N	unit tests
240	3.10.2	DS	plotting inspection	Q02	1.0	013	241	N	`fplot`
241	3.10.2	I	plotting inspection	Q04	4.0	240	242	N	coding
242	3.10.2	T	plotting inspection	Q05	1.0	241	300	N	unit tests
250	3.10.3	DS	data selection	Q03	1.0	013	251	N	`aedit`
251	3.10.3	I	data selection	Q08	6.0	250,300	252	N	coding
252	3.10.3	T	data selection	Q09	1.0	251	400	N	unit tests
260	3.10.4	DS	other visualizations	Q03	1.0	013	300	N	design only

programmatics

N.	Ref.	Type	Topic	Sta.	Eff.	Pre.	Suc.	G	Comments
300	4	R	review	Q07	0.0	many	many	N	mid-course review
400	4	R	review	Q12	0.0	many	many	N	pre-verification review
500	4	R	review	Q14	0.0	many	many	N	pre-validation review
600	4	R	delivery	Q16	0.0	many	none	N	final MSRI review/delivery

5. References

[1] A. Niell et al. “Demonstration of a Broadband Very Long Baseline Interferometer System: A New Instrument for High-Precision Space Geodesy”. In: *Radio Science* 53.10 (Oct. 2018), pp. 1269–1291. ISSN: 00486604. DOI: 10.1029/2018RS006617. URL: http://doi.wiley.com/10.1029/2018RS006617.

[2] AR Whitney et al. “Mark 4 VLBI correlator: Architecture and algorithms”. In: *Radio Science* 39.1 (2004), pp. 1–24.

[3] Adam T Deller et al. “DiFX: a software correlator for very long baseline interferometry using multiprocessor computing environments”. In: *Publications of the Astronomical Society of the Pacific* 119.853 (2007), p. 318.

[4] Adam T Deller et al. “DiFX-2: a more flexible, efficient, robust, and powerful software correlator”. In: *Publications of the Astronomical Society of the Pacific* 123.901 (2011), p. 275.

[5] Lindy Blackburn et al. “EHT-HOPS pipeline for millimeter VLBI data reduction”. In: *The Astrophysical Journal Letters* (2019).

[6] Kazunori Akiyama et al. “First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole”. In: *The Astrophysical Journal Letters* 875.1 (2019), p. L1.

[7] Kazunori Akiyama et al. “First M87 Event Horizon Telescope Results. II. Array and Instrumentation”. In: *The Astrophysical Journal Letters* 875.1 (2019), p. L2.

[8] Kazunori Akiyama et al. “First M87 Event Horizon Telescope Results. III. Data Processing and Calibration”. In: *The Astrophysical Journal Letters* 875.1 (2019), p. L3.