Clock Searching

version 4.0

10 november 2005

This document is a revision of the *Clock Searching v3.0* guide from 23 sep 02. It goes over steps for clock searching an experiment that has been prepared previously (see the *Pre-correlation* guide). There are many ways to get the post-correlation delay/rate residuals into the `$CLOCK` section of the VEX file(s) used to control the production correlation. This guide describes the most straightforward method based on existing software, one that renders the intricacies of sign conventions & units more or less transparent to the clock-searcher. Discussion of these topics can be found in an earlier version of the *Clock Searching* guide, which can be found in ∼`campbell/public-html/clksrch3.ps.gz`. Here, we give a single formula which is handy for converting between [μs/s] & [mHz] when dealing with delay-rates:

$$\mathrm{Rate}_{[\mathrm{mHz}]} \quad = \quad \mathrm{Rate}_{[}\mu\mathrm{s/s}] \times \nu_{[\mathrm{GHz}]} \times 10^{6}$$

0. for the remorselessly impatient

Below is a checklist for clock-searching, with references to paragraphs in the main text.

α) Before you start to correlate:

Why clock search? . . . . [**1.a**] p.2

Selecting scans, correlation parameters, *etc.* . . . . [**1.b**] p.3

β) Correlation:

Confirming fringes in `fringe_display` . . . . [**2.a**] p.7

No fringes to some station(s)? . . . . [**2.b**] p.9

γ) Determining clock offsets/rates:

Making the Measurement Set . . . . [**3.a**] p.11

`jfrnge.g` . . . . [**3.b**] p.12

other plots from the 1st clock-search scan . . . . [**3.c**] p.17

δ) Applying the clock offsets/rates:

`updateclock.pl` . . . . [**4**] p.(18)

ϵ) Running a confirmation scan:

Fringes centered in `fringe_display`? . . . . [**5.a**] p.(20)

Plots to document successful clock-searching . . . . . . [**5.b**] p.(20)

Fixing swapped pols in the `vix`-file . . . . [**5.c**] p.(21)

The rest of this guide attempts to conform to a convention in which:

`Typewriter text font` signifies verbatim parts of a path, program input, *etc.*

Italic font signifies parts of a path, program input, *etc.* that may vary (usually experiment-, job-, or disk-related).

{braces} signify a task or parameter that may not be required, depending on your situation.

The computers you will be using are referred to as C^3 (currently `jaw0`, for correlator control), D^3 (currently `juw26`, where raw correlated data live), and E^3 (currently `juw27`, where the bulk of the post-correlation operations take place). Unless otherwise specified, everything in here assumes you have logged in as user `jops`.

1. Before You Start to Correlate.

a. Why Clock-Search?

The goal of clock searching is to find the clock and clock-rate terms for each station that center the fringes for all baselines as well as possible throughout the experiment, and which reduce the delay-rates to near zero (a typically goal is $\leq$5 mHz). Centered fringes mean flat phase as a function of frequency; zeroed delay-rate means flat phase as a function of time. Both characteristics together will provide more successful vector averaging in subsequent stages of the post-correlation review without having to fringe the data (*cf.* the *Post-Correlation Review* guide). Of course, the typical case of having only two adjustable parameters per station (clock offset, rate) to fix over the whole experiment can not replace fringing later on in AIPS, but the confidence to use vector averaging in our pre-fringe review plots, without losing too much coherence to phase wrapping over frequency or time, helps us retain sensitivity to weaker sources. Of course, we center fringes on a baseline basis, so you only need to deterimine these clock terms for $(N_{\mathrm{sta}} - 1)$ stations with respect to a reference station, whose clock terms are fixed. Typically, you'd pick the strongest station (Ef/Eb) for the reference station to get the highest possible SNRs, unless you know a priori that there's a major problem with that station. (*N.B.*: Ef/Eb can have noticeable differences — 10's of ns — in clock offsets among different BBC's.)

The "original" estimates for the clock parameters are derived from the stations' GPS data, which is daily observations of the difference of GPS and station's formatter output, maser, or other such time signal (the order of the differencing isn't necessarily consistent everywhere, although by now for stations that participate regularly, `log2vex` knows what the sign conventions are, such that the initial VEX file should have a consistent definition for the sign of the rates). Unless there's an obvious problem in the GPS model (*cf.* ¶1.b), the initial GPS-derived clock offsets usually require $< 2\,\mu$s adjustments and the GPS-derived clock rates are usually well under the 5 mHz guide-line, when performing the clock-searching with respect to a fixed reference station. We are compiling a history of clock-search adjustments per station, with a future goal of getting a closer *a priori* estimate. However, to date we have not worried about getting an absolute calibration of the network as a whole, such that the time-stamps of the correlated data would rigorously be UTCs at the fixed points on all the antennas.

Clock searching will most likely be the first time a scan from an experiment get correlated. As such, it provides the first glimpse into the experiment's data. Thus in the course of clock searching, we also have the opportunity to:

- ensure that the polarizations weren't swapped at any station (*i.e.*, are the parallel-hand fringes considerably stronger than the cross-pol fringes?),
- check whether phase-cal was turned on or not (except for spectral-line observations, the clock-search scan(s) will likely use more lags, and hence have higher spectral resolution, and any subsequent production correlation),
- check sampler-stats for the stations, and look for other other station-based problems (*e.g.*, low-power BBCs, maser casualty, *etc.*).

Of course, NMEs and other experiments from the session that have already been correlated could also point to such problems, but individual experiments always have the opportunity to show a unique set of problems (and seem to relish punishing complacency...).

b. Selecting Scan(s), Correlation parameters, *etc.*

As with most things in life, the hardest (and most significant to making your life easier subsequently) part of clock searching is the thinking before you even start. Included in this general area include which source(s)/scan(s) should I use, what's my back-up plan if station problems prevent using the originally identified source/scan(s), how much do I trust the *a priori* GPS clock model, and what correlator parameters do I want to use? Some thoughts for addressing such questions:

i — Ideally, the PI would have scheduled at least one scan of a fringe-finder source that is visible to all stations, and to which all stations can slew to without losing excessive time on source. This would allow you to use just one scan to find all the clocks. Unfortunately, this is not always the case, especially for globals with low-declination targets. Even if there is such a scan, station problems during that time might also leave you with a situation in which you don't have any scan in which all stations successfully observed. If this is indeed the case, then it will take multiple scans to find all the clocks, with at least one station in common between each set of two scans. You can then optimize the selection of scans to the sub-set of stations participating in each scan. Using sub-sets of stations may also allow you to get higher values for the correlation parameters ($N_{\rm lag}, N_{\rm pol}, N_{\rm sb}$) than you could have used if using all stations. Tools to help navigate through this include:

- ◦ the `sum`-file from sched, to get per-station source elevations & late-on-source times. Depending on what the PI included in the `sumitem` upon running sched, you may need to remake the `sum`-file again. In principle, a `sum`-file with the appropriate entries should already exist in the experiment folder by the time you do clock-searching. Watch out for stations that come on source rather late in the scan (ideally, would have $\gtrsim$5 min on source, to help get an idea of any residual clock-rate) and stations with low elevations (which may be unavoidable in globals).
- ◦ a list of calibrator/fringe-finder sources, with flux-densities or u-v distribution plots. The USNO and VLBA web pages provide tools to check out the images and/or u-v distributions for various sources (as S & X bands):

  `rorf.usno.navy.mil/rrfid.shtml`

  `www.vlba.nrao.edu/astro/calib/index.html`

  These pages are also linked from the EVN "VLBI Links" page. A text version of the VLBA Calibrator List resides on `jaw0`: `/expr/vlbaCalib.txt`.
- ◦ station feedback (from the EVN/TOG web page) and comments (derived by `log2vex` from the field-system logs; should be a copy in the experiment folder) provide an overview of possible problems at the stations that may affect their ability to observe the scan(s) you'd prefer to use. The field-system logs themselves provide more detailed information; these live on `jaw0`, under the path `/jaw0_1/jops/Ftpfiles/logexp_date/`*EXP_yyyymmdd*`/`*expss*`.log`, where *ss* in the filename is the lower-case 2-letter station abbreviation.

ii — If all stations can be clock-searched at once, the choice of a reference station isn't terribly critical at this pre-correlation stage; all stations will be in the data and you can make this decision when running `jfrnge` or analyzing its output (*cf.* ¶3.b). If you don't have any scan in which all stations observed successfully, then you need to pick at least one station in common to both scans you will use. After finding all the clocks from the first scan, one of these stations will become the reference station for the second scan, to ensure that all the clocks are centered with respect to each other consistently. If you need more than two scans, then the above scheme is needed for each pair of scans, but there's no absolute need for a station to be in all scans. In principle, using the most sensitive station as the reference lets you get away with using fainter sources — with a true fringe-finder, station sensitivity won't be a significant factor.

iii — The initial clock parameters in the VEX file are derived from a linear fit to 5 days' worth of (daily) GPS data as provided by the stations. `log2vex` can make plots of the GPS data around the time of the experiment, along with a red line segment representing the linear fit. The fitting process currently disregards jumps in the GPS data, so it's possible that the linear fit will have a bias in offset, rate, or both. If this is the case, there should be a copy of the plot in the experiment folder. If it's obvious from the plot that the days surrounding your experiment are well-fit by a straight line, then you could simply make a manual estimate of the clock parameters by interpolating the two surrounding days to the reference time for the clock model (the "`clock_early_epoch`" in the `$CLOCK` section of the VEX file). It may also prove informative to plot the values reported in the `gps-fmout` (or vice-versa for some stations) lines in field-system logs. If present, these measurements occur much more frequently than once a day, and thus can give you a feel for the (formatter) clock stability through the experiment. The operational system currently can't handle multiple sets of clock parameters for a single station; if there is a major jump known *a priori*, then two multiple VEX files, each with different clock parameters, would be required.

iv — In selecting correlation parameters, the key factor is the width of the fringe-search window (more precisely, the delay-search window). If the clock-difference of two stations from its *a priori* value is larger than the (half-)width of the delay-search window, then you won't see the fringe. Of course, there may be other reasons why you don't see fringes during clock searching (problem at the station, off source, source characteristics, *etc.* The absense of fringes gives you no real clues as to the underlying cause — that's why it's important to pre-select scans/stations so that you can rule out certain causes with some confidence, and focus on others.

The width of the delay-search window is a function of the subband bandwidth ($BW_{\rm sb}$) and the number of lags: each lag is the equivalent of $1/2BW_{\rm sb}$ seconds (this also holds valid for how we currently handle oversampled data), so the delay-search window extends $\pm N_{\rm lags}/4BW_{\rm sb}$ μs, if $BW_{\rm sb}$ is in MHz. Thus, the more uncertain the *a priori* clock parameters, the more lags you would want to use. At the same time, you would also want to use 4 pols to be able to check for swapped-pols, and all subbands to check on the sampler stats. The correlator itself has finite capacity, where (for local validity):

$$N_{\rm lag} \quad \leq \quad \frac{262144}{N_{\rm sta}^2 \cdot N_{\rm sb} \cdot N_{\rm pol}}$$

In this equation, $N_{\rm sta}$ is "granular" in units of 4: for 5–8 stations, use 8, for 9–12 use 12, and for 13–16 use 16. A more detailed derivation & discussion of this equation can be found in ¶3.1 of the *Field-of-View Calculations* document on the wiki, or on the EVN web site `www.evlbi.org/user_guide/fov/`). Note that you can run out of lags pretty quickly: for a 1Gb/s experiment with $N_{\rm sta} \geq 9$ stations, you can only get 32 lags when doing cross-pols — only ± 0.5 μs. Nowadays, the initial clock offsets are typically within 2 μs; if there has been another experiment observed close to the one you're clock-searching that itself has already been clock searched, you could "borrow" those parameters as your *a priori* values. In principle, this is as simple as cut-n-pasting the other experiment's `$CLOCKS` section; however, this would throw off keeping track of the clock-parameter adjustments made in clock searching, so it would be better to add new lines manually with the other experiment's post-clock-searched parameters. There's currently no script to do this. Of course, once you've found some stations' clocks, you can cut back on N_{sta} by re-searching only those stations still not found (the key is to reduce N_{sta} from > 8 to ≤ 8). If you really have to, you can also cut back N_{pol} in a dual-polarization experiment (from 4 to 2), N_{sb} by commenting out some subbands in the `$FREQ` and `$TRACK` sections of the VEX file, and if you're really desparate, change to `validity = global` in the `runjob.pl` profile (this will provide another factor of 2 in correlator capacity, at the risk of encountering the problem of mutually tape-frame headers correlating with each other when the baseline *a priori* delay corresponds to an integral number of tape frames (*cf.* the *Operational Impact of Delay/Rate Problems* guide on the wiki).

2. Initial Correlation.

The process of correlating for clock-searching is exactly the same as for production correlation. You should have a `runfile` file from which `runjob.pl` can extract the correlation parameters, but if this doesn't already exist, `runjob.pl` can create it. By using a separate *Klok* profile (or some similar same), you can easily distinguish between clock-searching jobs and production jobs making your `lis`-files later on (*cf.* the *Post-Correlation Review* guide, ¶1.d, 2.a,*iii*). Also, a safety feature of `runjob.pl` prevents you from changing the correlation parameters of profiles that contain the (case-insensitive) string "`prod`" on the first job has been run for such a profile. In clock-searching, you'll want to change the parameters, so avoid using any such profiles for your clock-searching jobs. For the initial clock-searching job(s), you'd typically want:

- as many lags as you can get (to maximize the chance of having the fringe in the delay-search window, and also to provide higher spectral resolution for checking whether phase-cal tones are present)
- cross-pols (to be able to check for crossed-pols)
- low integration times aren't usually important, unless you have some reason to suspect severely bad *a priori* rates (lower t_{int} widens the rate-search window). Higher t_{int} cuts down the size of the output data/MS to `jfrnge` later (¶3). Typically, $t_{\text{int}} = 2$–4 s is adequate.

Also, since the `vix`-file was most likely made shortly after the experiment ended, it may not have the most up-to-date EOPs. You can quickly fix this by running `log2vex` for the experiment, clicking on the `Eop` button, and saving the result to a file. Then just cut-n-paste this file over the existing `$EOP` section at the end of the `vix`-file. If there are multiple `vix`-files, make sure to do this for all of them. Do not change the `$EOP` section once clock-searching is done.

a. Visual fringe confirmation.

i — As the correlation is ongoing, we have the opportunity to see the lag-space correlation functions for each integration as they come out of the correlator. In the `fringe-display` interface window (initially iconized along the left-side side of the screen, you can enter "`help`" at the prompt for a list of all commands. Generally, the first command should be:

`update` N_{sec}

where N_{sec} would be the integration period (provides the most rapid useful refreshing of the plot). The syntax for plotting a fringe spectrum is:

`plot` M `intf` S_1-$S_2/SB/Pol$

The spaces shown above are not mandatory. M is the (1-based) plot window to use, S_1 & S_2 are the DPU numbers of the stations of the baseline to plot, *SB* is the subband to plot (a 0-based integer), and *Pol* is the polarization specification, either "`00`", "`11`", "`01`", or "`10`". Here, "`0`" represents LCP and "`1`" represents RCP. Thus, in an RCP-only experiment, there will only be polarization 11. The order of the baseline in the plot will come out in VEX- file order (specifically, the order in which the stations appear in the `$STATIONS` section of the VEX file used to control the job), regardless of the order you type in for S_1-S_2, or of the stations' SUs. For all but *Pol*=`10`, the first station in the VEX file will be the first station of the baseline; for *Pol*=`10`, it's just the opposite. Thus is there is a noticeable residual offset, you would see the fringe shifted in one direction relative to the center for LL, RR, & LR, but shifted in the other direction for RL. This is of no operational concern if you follow the procedures in this guide (the actual clock-parameter adjustments will be determined later in a way that automatically takes care of the sign conventions for baseline-order & delay-offset direction). An annoying aspect of the `fringe_display` window is that the `Up-Arrow` key doesn't return the previous command, so although well over half the keystrokes will remain the same, you still have to type them all in for each new plot.

The top plot in fringe-display window shows the real (red) and imaginary (green) components of the correlation function. The bottom plot shows the amplitude (quadrature sum of real & imaginary), plus an annotation giving the lag corresponding to the parabolically-interpolated maximum of the correlation-function amplitude.

ii — Cycle through baselines to a station as the correlation procedes, checking for fringes in the `fringe_display` plots. At this stage, it doesn't really matter which SB or pol you look at; once you see fringes for these station(s), you know the *a priori* clock parameters were good enough to get into the fringe-search window, and that the quantitative clock offset/rate adjustment determinations in ¶3 will work. It's generally best to get through all baselines quickly, worrying about the ones you don't see fringes to later (*cf.* ¶2.b). If you finish all baselines quickly enough, and if If the scan lasts long enough, let it run for $\gtrsim$5 min; this would provide a reasonable (time) baseline upon which to estimate a rate-offest. If the scan lasts $\lesssim$8 min, might as well let it run to completion. I generally write down the location of the fringe peak for each station to the reference in the operator logbook, to leave a record of fringes that have already been found.

b. No fringes to some station(s)?

The problem with not seeing any fringes on a baseline is that that piece of empirical imformation doesn't necessarily tell you why there aren't fringes. The possible problems are many; below are some things to keep in mind before panicking:

- check the `sum`-file from sched to make sure the station isn't just late getting on-source, or that this source/scan was indeed above the horizon.
- Avoid SB/pols where you know the station in question has an equipment problem (dead BBC, not enough front-end bandwidth, *etc.*). Cm has only $\lesssim$16 MHz available bandwidth over the micro-wave link back to Jb, so in a wide-band experiment may fringes in only a couple subbands. Try starting from the middle (SB3,4 if $N_{\mathrm{sb}} = 8$).
- If the problem is with Wb, Jb, or the VLA, check to make sure you have the proper position (in the `$SITE` section of the VEX file) & clock (in the `$CLOCKS` section of the VEX file) for the specific antenna/array used in observing. For the VLA, `log2vex` should detect whether the array or a single-dish observed, and if the latter, which pad location was used. Thus the position should be okay, and the clock should be adjusted reasonably well (the array should have an *a priori* clock offset of ~235–240 μs). For Wb, there's no automated was of knowing whether the array or a single-dish (Wb7) was used; you'll have to rely on the station feedback. The array should have an *a priori* clock offset of over 1000 μs. It is possible for single-dish observations to be routed through the adding-box, such that they would look like array observations to the correlator. For Jb, the thing to check is whether the antenna that observed is indeed the one that was scheduled (it's not unheard of for one to be swapped for the other after the PI has scheduled due to maintenance/repair work). The difference between the appropriate *a priori* clock offsets for Jb_1 & Jb_2 isn't enough to worry excessively about (there's only one set of GPS data for `log2vex` to use anyway). If you do correlate with the station position of the wrong antenna/configuration, you may still see a fringe, which at first glance may appear fine. But later plots (*cf.* ¶3.c, ¶5.b) should show high residual rates and/or delays (depending on the baseline projection with respect to the source).
- a station can suffer a "integral-second formatter clock jump", which would place the fringe far outside the delay-search window. However, there is usually a comment about this in the field-system log or station feedback; I don't recall an instance where we successfully found a fringe by blindly trying to "fix" such a formatter jump when the station didn't explicitly mention the occurrence of one. The way to fix these is to add or subtract 1000000 μs from the existing *a priori* clock offset value in the VEX file.
- N_{lag} may not be big enough to get the fringe into the delay-search window for a station S using the reference station R, but is for S to some other station (at this stage, you just want to confirm that fringes exist — you'll have a chance to quantify the delay/rate residuals, to all baselines if necessary, in ¶3.b). This

especially may be the case if the fringes for most stations to R lie noticeably off-center, all to the same side (thus, would have more centered fringes with respect to each other) — S's fringe may lie further out in the same direction. Try again plotting S in a baseline with the station whose fringe to R was farthest towards the edge. If the fringes on R–`*` were more scattered, try S to the two stations whose fringes to R were farthest out in either direction.

- Sometimes the PI's scheduling combined with station problems forces you to try a marginal source for clock searching. It's always possible that the source was too weak on a baseline to provide a good fringe in just a single integration. You may still be able to detect the source in `jfrnge.g`, which performs a 2-D fringe-search over the whole scan. As you will do this step anyway (*cf.* ¶3.b) don't worry too much until then.

3. Determining Adjustments to Clock Offsets / Rates.

By making a (small) Measurement Set from your clock-search scan, you can look at in AIPS++ to quantify the residual clock offsets and rates on various baselines. The principal advantage of clock searching via AIPS++ is the ability to look at all subband/polarizations on any/all baselines simultaneously, without the need for rapid typing to pull up all the fringe-display plots individually. Further, you can easily get a hard-copy record (print-outs & plots), determine rate corrections, and have the possibility to use fringe-finder sources that aren't bright enough to give good fringes in a single integration period. There are some problems that using AIPS++ will not solve: unexpected clock/formatter-offset jumps, improper station BBC patching, *etc.* — not seeing expected fringes in the fringe-display plots to specific stations may likely be symptomatic of a more serious problem than can be solved by adding ~100 or so integration periods together off-line. Since ~March 2003, we've had a (rough) fringe-fitting program that outputs residual delays & rates with the appropriate signs and units to be fed directly back into the VEX file via `updateclock.pl` (¶4.a).

a. Making the Measurement Set.

In general, you'll want to follow the procedure in the *Post-Correlation Review* guide (¶2.a–c), but since you'll be making a Measurement Set from just a single subjob, there are some short-cuts you can take.

i — Set up a working directory for the experiment on E^3 (currently `juw27`).

ii — Get the data for the subjob (no real need to bother with a `lis`-file:

```
getdata.pl -proj EXP
```

`getdata.pl` will detect that there isn't a top-level VEX file yet, and thus will ask you whether you want to get one from `jaw0`. Enter "`y`", and then will provide a selection of files it thinks might be the VEX file you want. Enter the filename that you want for the top-level VEX file (`getdata.pl` will ftp it over, and take care of any renaming required — by not running `j2ms2` via a `lis`-file, the root of the top-level VEX filename needs to be the same as the subdirectory name it lives in... usually *EXP*). You will then be shown a list of jobs available on D^3; the default is the most recent job, which when clock-searching will most likely be the one you want. Just enter return to get the default job (or enter a different *jobID* to get another one).

iii — Make the Measurement Set via `j2ms2`:

```
j2ms2   -o MSname   jobID/SJ
```

where the subjob will almost certainly be "`1`". Remeber when running `j2ms2` in this “manual” fashion, you need to have the top-level VEX file have the name *dirname*`.vix`, where “*dirname*” is the name of the working directory (see the *Post-Correlation Review* guide, ¶2.c.*i*). This top-level VEX file should be the one used for correlation of your clock-searching scan. If you retrieved the proper VEX file via `getdata.pl`, all this is taken care of for you automatically.

iv — Start glish in the standard way: `glish -l logger.g`. Load the data into a variable & make an initial weight plot to check out the correlation quality:

- `mssum('`*MSname*`')` — note the stations that participated in the subjob. Make a vector, let’s say `gdsta`, of these stations’ (0-based) integer-IDs. As an example of specifying a vector in glish: `gdsta := [0,2,5:9,11]` (the “`:`” denotes an inclusive range of integers).
- *a* `:= readms('`*MSname*`', gdsta)` — to load the data associated with all baselines/autocorrelations included in `gdsta` into a variable, to make subsequent operations faster (no need to re-access the disk).
- `tplot(`a`,F,F,F,F,[1,2],'weight',T, yplot=`N_{sta}`,xplot=1,`
  `plotline=T,plottime=F)`

The `tplot` provides the weightplot of the parallel-hand pols (for a single-pol experiment, replace the “`[1,2]`” with another “`F`”). No output file is specified, so the plot will go to the screen. The `yplot` & `xplot` parameters can be adjusted as desired ($N_{\text{sta}}/2, 2$) could be appropriate for large N_{sta}). The parameter `plotline=T` connects the individual weight points to improve the contrast of the weight curves (in a plot that would have only a small number of integrations, ∼50–100). The parameter `plottime=F` makes the x-axis be integration-number rather than time. This may prove useful in `jfrnge.g` to specify a start/end integration to use.

b. `jfrnge.g`

`jfrnge.g` is a basic, baseline-based fringe-fitting program in glish. The main engine is simply a 2-dimensional Fourier transform; ∼90% of the code is for checking the various alternative input formats and making output reports/plots. There are lots of options, but for clock-searching most of the parameters can be left at their default values. There’s a detailed description of the syntax and all the individual parameters on the wiki. Here’s how a standard clock-search `jfrnge` run would go (assuming that you’ve just completed the step in ¶3.a.*iii* above), using just the most “common” parameters.

i — Syntax. Given that the variable *a* from the above `readms` call still exists, here’s how to use `jfrnge` to determing the clock-adjustment parameters:

```
include 'jfrnge.g'
```

`jfrnge(`a`, {`Ant_{ref}`}, weight=`w`, {initvis=`N_1`, ntimes=`N_t`})`

The parameter $Ant_{\rm ref}$ provides the means to look at all baselines to a reference antenna. The antenna may be passed either as an integer or as the 2-letter abbreviation string. If this is left out, then `jfrnge` will operate on all baselines. (Actually, this parameter can be a vector of antennas, and fulfills the same role as `ANTENNA` does in AIPS. The order of the first three parameters after the initial "MS" parameter is `antenna, baseline, ants`, as oppposed to the `ants, antenna, baseline` order in `readms` — *cf.* ¶3.h.*iv* in the *Post-Correlation Review* guide.)

Exclusion of low-weight/bad integrations noticed in the `tplot` weight(t) plot above can be handled with the parameter(s) `weight` and/or `initvis, ntimes`. The former has its usual effect: any integration with a weight $< w$ does not contribute to the `jfrnge` calculation. However, there is one new wrinkle here — `jfrnge` doesn't handle "gaps" in the data well when computing the delay rate (see the next section for the effect on the output print-out). With the advent of disks, this is becoming less & less important from a practical standpoint, so we've never really put much effort into fixing this. However, you can use `initvis` & `ntimes` to limit the data considered to a block of continuously high-weight times. The unit for both of these parameters is integration-number, which is where the `plottime=F` parameter in `tplot` above comes in handy (you can read integration number right off the x-axis). `initvis` is the integration-number for the first integration to use (default = 1); `ntimes` gives how many integrations to use starting from `initvis` (default = all). Note that in the usual case of ∼perfect playback with disks, only `initvis` with the integration-number of the first good-weight integration of the last station to synch would be required (*i.e.*, no `ntimes`, no `weight`).

ii — Output. As run above, `jfrnge` will make an output listing with a filename *EXP*`.`*MSname*`.clkfrng`. This file can be printed out comfortably with 126 characters per line (*e.g.*, `a2ps -Pxrxjive -R --columns=1 -l126` *filename*). The output file provides, for each baseline/SB/pol, one row per `jfrnge`'d "scan" (in the standard clock-searching usage, there is only 1 scan). This row contains the central time of the scan, the total number of integrations & the number of "good" integrations contributing to the solution, the resulting (residual) delay and rate estimates (with statistical uncertainties), a rough SNR estimate, and the center frequency of the SB. In this output, delays are in [μs] and rates in [mHz]. At the end of each baseline, there is a summary line containing the weighted average of delay and rate. This line has the proper sign convention and units for direct input into the VEX file(s) via `updateclock.pl` (*cf.* ¶4). Listing 1 is an extract of a `clkfrng`-file.

- If $N_{\rm gd} < 0.8N_{\rm vis}$, then each bsln/SB/pol will have two rows in which `jfrnge` treats the "missing" data in the gap differently: first, ignoring the gap (thus passing a non-uniformly sampled time-axis to the 2-D Fourier transform), and second, inserting $0 + 0i$ into all frequency points of all integrations in the gap. The second line appends a comment "`0-pad`". If the two delay/rate solutions are sufficiently different ($\Delta\tau > 10^{-3}$ lag, $\Delta\dot{\tau} > 0.1$ mHz; these limits fairly ad-hoc), then neither contributes to the weighted average for the baseline.

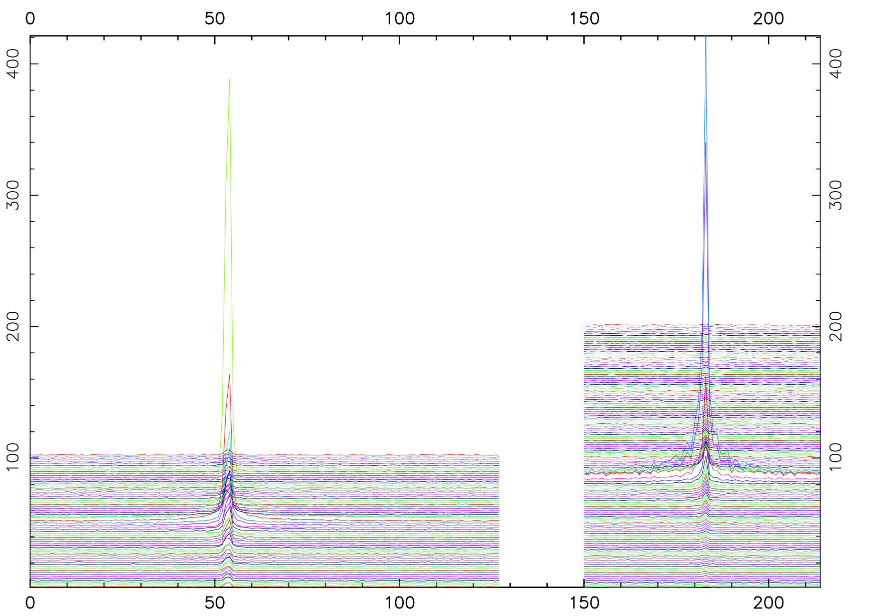
```
   JFRNGE for EB030, clk135
Delays in [us],  Rates in [mHz];  sign of Delay adjusted for updateclock compatibility
   Source    d/hh:mm:ss.sss  Nvis  Ngd       Delay      sig.Dly        Phs.Rt    sig.Phs.Rt    SNR     Freq.ctr
Tr-Ur
*** sub0,  LL
      OQ208  0/20:03:19.125    75   71      0.00879us   0.00112us     0.4724mHz   0.0157mHz     122.75    1654.490MHz
*** sub0,  RR
      OQ208  0/20:03:19.125    75   71      0.00998us   0.00116us     0.4540mHz   0.0163mHz     118.59    1654.490MHz
*** sub0,  LR
      OQ208  0/20:03:19.125    75   71      0.00760us   0.00291us     0.3827mHz   0.0407mHz      47.31    1654.490MHz
*** sub0,  RL
      OQ208  0/20:03:19.125    75   71      0.00757us   0.00449us     0.3788mHz   0.0627mHz      30.73    1654.490MHz
*** sub1,  LL
      OQ208  0/20:03:19.125    75   71      0.00991us   0.00112us     0.4532mHz   0.0157mHz     122.99    1662.490MHz
*** sub1,  RR
      OQ208  0/20:03:19.125    75   71      0.01324us   0.00123us     0.5639mHz   0.0172mHz     111.98    1662.490MHz
*** sub1,  LR
      OQ208  0/20:03:19.125    75   71      0.00644us   0.00291us     0.6460mHz   0.0407mHz      47.32    1662.490MHz
*** sub1,  RL
      OQ208  0/20:03:19.125    75   71      0.02285us   0.00468us     0.3483mHz   0.0655mHz      29.43    1662.490MHz
updateclock:  Tr-Ur average SBD/DR:  0.01025 +/- 0.00055 us     0.2905 +/-  0.0046 ps/s
```

Listing 1: `clkfrng`-file extract (from EB030).

iii — The goal of clock searching is to get the residual delays down to $\lesssim 0.1$ lag and the residual rates down to $\leq$ 3–5 mHz. The delay correction is usually quite straightforward; differences among the instrumental delays in different BBCs motivates the use of the weighted mean to determine the (single) clock-adjustment to add to the VEX file. If there is an apparent rate adjustment for a station, this requires a little more thinking before accepting. Does the GPS-model plot from `log2vex` suggest that the *a priori* rate estimate is off? Be suspicious if it doesn't — the GPS-derived rate is usually more than good enough; there's no sense of instrumental offset as there is for the *a priori* delay. Do you see the same residual rate for all baselines to the station in question (check with another `jfrnge` run with a different $Ant_{\rm ref}$; easy to do if a still in memory). If not, it's most certainly an artefact. If the high rate-residual has passed all these hurdles, and does indeed appear intrinsic to the station, you also have to watch out for the possibility that it's induced because of the source's low elevation as seen from the station (using easy-to-calculate geometries — stations on equator, $\delta_{\rm src} = 0°$, isotropic constant zenith delays of 7.5 ns — a baseline whose stations have elevations of 90° and 10° could see a tropospheric delay-rate of up to ~18 ps/s) or even source-structure gradients as seen by the baselines. Especially in globals, if you can find a scan with all stations participating it's likely that the outer ones (*e.g.*, Chinese, Mauna Kea) will have lower elevations.

An unstable frequency-standard (*e.g.*, maser acting up, or already shifted over to a Rb standard) can also induce what looks like a high fringe-rate over short periods (another reason for having run a 5–8 min scan for the clock-search, if possible). You can check whether `jfrnge` has found a well-defined peak in the fringe-rate spectrum via its on-screen (`gplot1d`) plots. Figure 1 shows two examples of such plots from N05C2.

Here is the `jfrnge.g` syntax for generating plots for the baseline Ant_1–Ant_2:



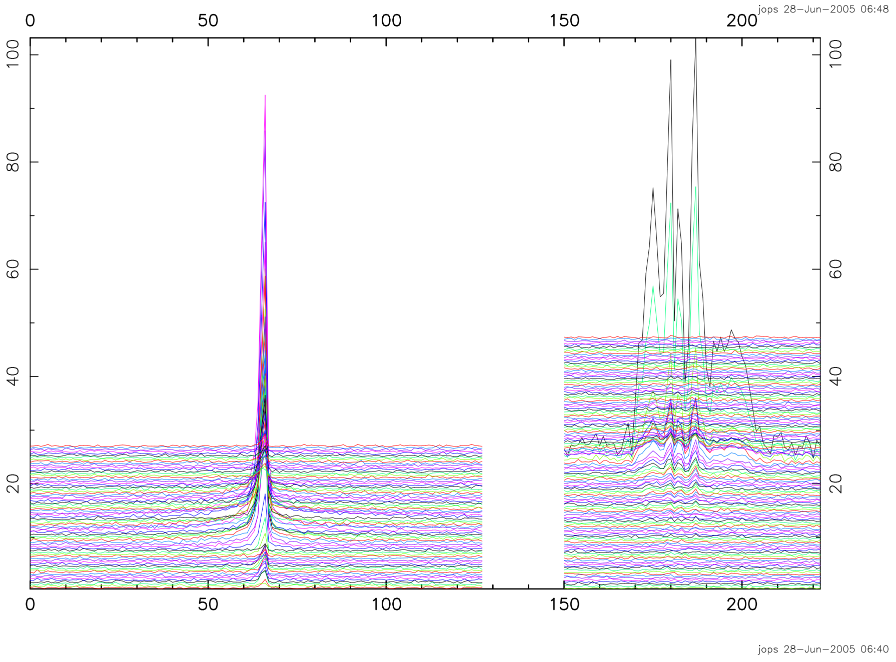


Figure 1: `jfrnge` plots from N05C2: top Ef–Wb, SB0/LL; bottom Ef–Tr, SB0/LL.

`jfrnge(`a`, `Ant_1`, `Ant_2`, weight=`w`, updtclk=F, doplot=T)`

The antennas can be specified either as an integer-ID or a 2-letter abbreviation (in single-quotes). The `doplot=T` parameter results in plots being made (default = `F`). The `updtclk=F` parameter omits making an output file (unless you also specifically include `outfile=`*OUTPUTfilename*). If your experiment has multiple subbands/pols, you'll then be prompted whether you want a plot for each SB/pol, or whether you want to limit the plotting to 1 SB/pol (you always have the option of specifying a subset of SB/pol in the command line with the `subband=` & `pol=` parameters). When a plot comes to screen, it stays until you take some action at the `-->` prompt. There are currently 3 actions (a letter followed by a carriage return):

- `s` — stop plotting from here on
- `p` — save current plot to a postscript file (will be prompted for the filename)
- anything else (including just return) — go on to next plot.

Each plot comprises 2 parts. In the left-hand part, delay is along the x-axis and rate appears to run into the page. In the right-hand part, rate is along the x-axis, and delay appears to run into the page (*i.e.*, it's as if you're looking at the left-hand part, then "walk around" the plot 90° clockwise and look at it again). The units on the x-axis are not scaled to [s] or [s/s]. The top plot of Figure 1 shows Ef–Wb, illustrating a normal good fringe — sharply peaked in both delay & rate. The bottom plot shows Ef–Tr, with a poorly-localized rate peak (in this experiment, Tr had lost its H-maser, and was using a Rb frequency-standard). For such a plot, it is clear that you can't place much confidence in the rate estimate.

The 2-part `jfrnge.g` plots in the `gplot1d` windows are clearly rather crude. I've made little effort in making a proper shaded-surface plot. However, if you need a pretty 2-D fringe plot, there is a variant of `jfrnge.g` that saves its output, and associated IDL scripts that plot it. An example is in Figure 2 (one of the "pseudo"-fringes from the same-night Mp–Pa fringe check from the actual Huygens-descent observations).

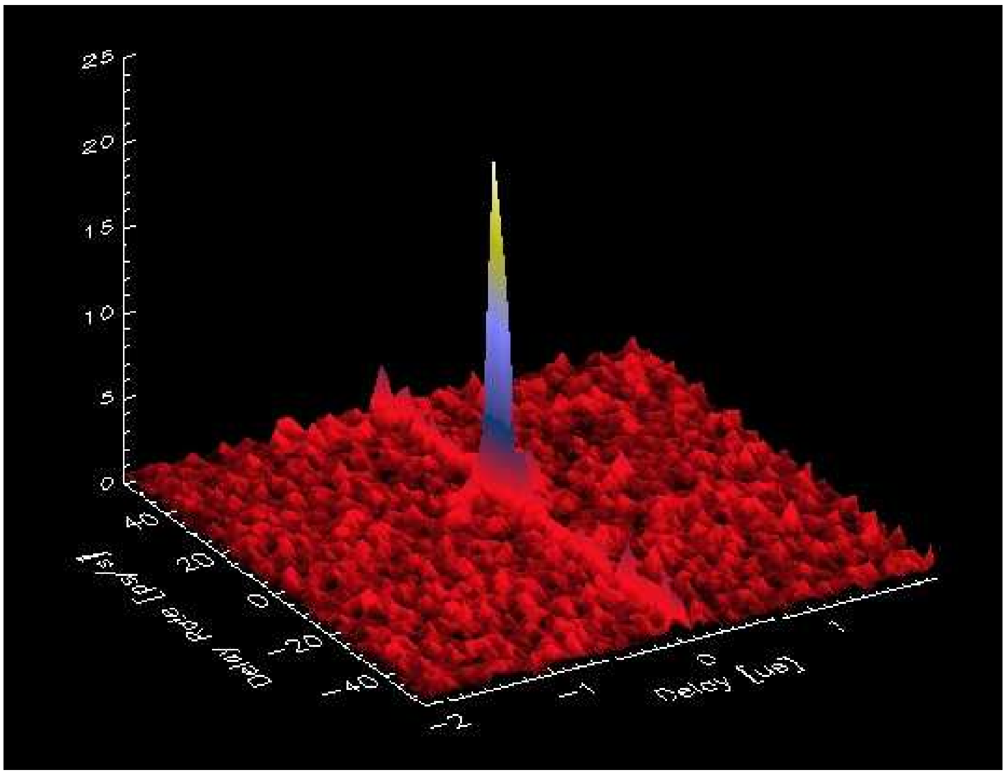


Figure 2: An example of a prettier fringe-plot possible with `jfrnge`-variant output.

c. Other useful plots

While you're clock-searching, you're in the best position to make a couple other plots that will make subsequent reduction and/or correlation.

i — `fplot` autocorrelation plots.

Unless you're doing a spectral-line experiment, it's likely that clock-searching will use more lags (& thus have higher spectral resolution) than will production correlation. Thus autocorrelation bandpass plots from clock-searching will provide the clearest evidence of whether phase-cal was on in the participating stations. Use:

```
a := readms('MSname', auto=T)

fplot(a, F,F,F,F,[1,2],'real',T, yplot=N_sta,xplot=N_sb,
  weight=w, multisub=F, outfile='PLOTfilename')
```

For $N_{\rm sb}$ = 8 (or $N_{\rm sb} > N_{\rm sta}$), you could add `landscape=T`. For a large number of stations, you could make a multi-page plot by making `yplot` some fraction of $N_{\rm sta}$.

If you're clock-searching with $\geq$ 256 lags, the phase-cal tones should be fairly evident. Stations without phase-cal tones wouldn't need to have their delay-rate∼0 events flagged by `plyflg.g` (via including them in the `noDR` array of stations). If you have a station that appears not to have phase-cal tones, check a couple of its delay-rate∼0 events found by `plyrecon.g` to see if the characteristic amplitude-spikes exist. This process is described more fully in the *Post-Correlation Review* guide, ¶3.f.*i*–*ii*.

ii — `tplot` for sampler-stats.

The effect of the distribution of "high"- & "low"-bits in the 2-bit sampled data is discussed in the *Post-Correlation Review* guide, ¶3.e. What is useful to learn during clock-searching is whether any station has a very low fraction of high-bits (f_h) in all its SB/pol channels, say $\lesssim$ 10%. If this is true for a station, treating that station as 1-bit sampled (*i.e.*, ignoring the magnitude bits) would actually result in a senstivity gain, when the effects of the 2-bit van Vleck correction are taken into account. The first step is to make a plot of f_h (assuming the variable *a* still contains the result of the `readms` from the previous sub-section):

```
tplot(a, F,F,F,F,[1,2],'samp',T, yplot=N_sta,xplot=N_sb,
  weight=0.1, multisub=F, outfile='PLOTfilename')
```

The same sort of page-layout considerations from the previous sub-section also pertain here. The *y*-axis range for all plots will be from –3% to 103%. Ideally, all channels from all stations will lie close to ∼36.4%. Stations that have all (or most?) of their channels near or below the 10% level are candiates for treatment as 1-bit sampled stations. (The parenthetical "or most" in the preceding sentence is there because our choice is to treat the station as either 2-bit or 1-bit sampled — there's no current option to treat individual channels differently for the same station.)

You can make a station appear to be 1-bit sampled through a fairly straightforward modification to the `vix`-file. This is why it's important to check this during clock-searching — so that the production `vix`-file can be adapted if necessary. Once the correlation is done, the damage would be done. The `vix`-file modification entails making a new `$TRACKS` section for the station(s) with too-low f_h, as illustrated in Listing 2 **(soon to appear)**

Listing 2: `vix`-file modifications to treat a station as 1-bit sampled, in case of $f_h \lesssim 10\%$.

4. Applying the Clock Offset / Rate Adjustments.

The `jfrnge.g` output file (default name = *EXP*.*MSname*.`clkfrng` contains the clock offset & rate adjustments to make to the *a priori* clock model in the production `vix`-file(s). Back in the `expr/`*EXP* directory on C^3, you run `updateclock.pl` to enter these adjustments directly, without having to worry about the sign-convention or units. `updateclock.pl` will automatically edit the `$CLOCK` section of the `vix`-file to include the updated clock parameters, saving a backup copy of the unupdated version of the `vix`-file. If for whatever reason you will be using multiple `vix`-files for production-mode correlation (*e.g.*, continuum/line, sub-sets of `$SCHED` in time, *etc.*), make sure that you copy the updated `$CLOCK` section from the first one that you correct to all the others (less risky, and faster, than running `updateclock.pl` multiple times).

The syntax for the usual operational execution of `updateclock.pl` is as follows (including no parameters returns help):

`updateclock.pl -scan` M_{scan} `-ref` *Ant*   *VEXfilename*

Here M_{scan} is the scan number (from the VEX file) being used (really only important if you're adjusting one or more station clock-rates, but it's best to get into the habit of supplying it) and *Ant* is the 2- letter abbreviation for the reference station. It's really important to include the `-ref` parameter to make sure the signs of the clock corrections work out properly. There will follow some prompts:

- You're first prompted for the "Mounted tapes". The default is all stations in the VEX file. You can enter a list of 2-letter abbreviations for the stations you plan to adjust — you won't be asked for clock corrections on baselines to the `-ref`-station for stations you leave out. You can just hit return here to accept all stations, and enter no adjustment (the default) for the stations you don't want to touch. This latter tactic is faster if you're omitting just a few stations out of many.

- The program then starts to loop through the baselines from all the stations you've selected above to the `-ref`-station (showing the proper `vix`-file order). It first echoes the current values the (non-`ref`) station's clock model. You are then prompted to enter 3 adjustments (per station):

    - `Scan epoch` (default = middle of scan from the `vix`-file)
    - `Residual delay` [μs] (default = 0)
    - `Residual rate` [ps/s] (default = 0)

  To accept the defaults (in [ ]), just hit return. The default `Scan epoch` should be okay. For the `Residual delay` and `Residual rate`, type in the value(s) in the "`updateclock:` Ant_1-Ant_2 `average SBD/DR`" line at the end of each baseline's section of the `jfrnge.g` output file. It's probably not worth entering a `Residual delay` $\lesssim$ a tenth of a lag (one lag = $1/2BW_{\mathrm{sb}}\,\mu$s), nor `Residual rates` less than ~5 mHz (and even then, check to make sure the residual rate estimate makes sense in light of other information — the *a priori* GPS-model plot, the quality of the `jfrnge.g` rate peak, *etc.*). `updateclock.pl` will take care of the units and any necessary sign swaps due to station/baseline order. If a station has no rate correction, just hit return to accept the "0" default. The tricky bit that sometimes catches me out is the initial query for `Scan epoch`; if you enter the delay offset here, `updateclock.pl` will fall over because it's not in the proper format (& there's apparently no format-checking/re-query in place). Should this occur, you'll have to start all over (hopefully, you don't do this in your 15th station...)

- If you entered a non-zero `Residual delay` or `Residual rate`, `updateclock.pl` will print out what the station's updated clock-model parameters would be, before going onto the next baseline. At the end of all baselines to the `-ref` station, you'll be prompted whether you really want to apply the corrections you've just entered (if you've made a correction to any station). If you type "`n`", then you won't change the `vix`-file.

i — Make sure you run `updateclock.pl` separately for each scan (when using it to incorporate residual rates). Also, make sure each run of `updateclock.pl` has only one reference station. If you need to use multiple runs to get all the stations, make sure you enter the appropriate `-scan` & `-ref` in the `updateclock.pl` command line. Of course, once you've updated the clocks for the first subset of stations, any of those can serve as the reference station in the second subset, and the "new" stations will be automatically tied into the "old" stations.

5. The Confirmation Scan.

Once you've updated the clock parameters for all the stations (that need updating), you're almost ready to start production. Just another check to make sure everything was entered properly…

a. Re-run the clock-search scan(s)

With the updated clocks, the fringes should be centered in the `fringe_display` windows (at lag = $N_{\rm lag}/2+1$). You can make a quick visual check of this as the scan is running. You probably don't need as many lags as you did in the original clock-searching; try to keep cross-pols, however. Stochastic residuals of a couple tenths of a lag aren't too worrying (at least at the higher 8 MHz or 16 MHz $BW_{\rm sb}$). Let the job run ∼4–5 min. Large residuals may be the sign that you missed a sign or dropped a post-decimal-point "0" from a residual delay in the `jfrnge.g` output file.

b. Final Plots Documenting Successful Clock Searching

Go through the standard Measurement-Set creation steps for this job. The goal for this MS is to produce plots that document the clock-searching has indeed left the clocks centered & the rates zeroed. There are three plots you want to make:

```
a := readms('MSname', gdsta)
(1) tplot(a,F,F,F,F,[1,2],'phas',weight=w, yplot=Ny,xplot=Nx,
      globalscale=T, outfile='PLOTfilename')
(2) fplot(a,F,F,F,F,F,'amp',weight=w, yplot=Ny,xplot=Nx,
      outfile='PLOTfilename')
(3) fplot(a,F,F,F,F,F,'phas',weight=w, yplot=Ny,xplot=Nx,
      plotline=F,globalscale=T, outfile='PLOTfilename')
```

In order, these show:

(1) phase *vs* time, with all the parallel-hand polarizations of all subbands placed in the same plot for all baselines that can be formed by the array of stations

gdsta. As a general rule, $N_y = 12$ & $N_x = 3$ produces a reasonably clear plot (no more than 4 pages for $N_{\mathrm{sta}} = 16$). Flat $\varphi(t)$ shows that the clock-rates are set well (if you have *sim*1 wrap over the duration of your clock-search scan for some baseline(s), then the vector-averaged plots of amp & phase *vs* frequency across the band will suffer decorrelation).

(2) amplitude *vs* frequency, with the subbands plotted side-by-side, separated by dotted vertical lines, for each baseline. All the polarizations are shown, with the polarization color-coding the same for all subbands (red = LL, blue = RR, cyan = LR, green = RL — unless you only have RR, which in that case would be red). In these plots, check that no station has swapped polarizations (in which case the cross-hand polarizations would be stronger than the parallel-hand polarizations for all baselines to a given station). See ¶5.c for how to unswap the polarizations in the `vix`-file. You can also check for any dead channels, and see if that makes sense with the station feedback.

(3) phase *vs* frequency, with the subbands arranged as above. Here, flat $\varphi(\nu)$ shows that the clock-offset parameters are set properly (an end-to-end phase difference of 180° would signify being 1 lag off in the fringe-centering). Some stations will have obvious S-shaped curvature across the band (Wb, Cm near the outer reaches of its 16 MHz micro-wave link); there's not much you can do about this during clock searching.)

Print the plots out to `tek8200`, and put them in the experiment folder, along with the `jfrnge.g` output file(s) and earlier plots (*e.g.*, the autocorrelation plot checking for phase-cal tones). If there was anything unusual (*e.g.*, swapped pols, dead tracks), make notes in the session notebook that should be between C^3's monitor and disk-tower.

c. Fixing Swapped Pols in the `vix`-file

(soon to appear)

Listing 3: `vix`-file modifications to fix swapped polarizations at a station.