Sean Sellers

October 24, 2025

SSOSoft for FIRS

Contents

1 Introduction to the Code

SSOSoft [1] is the full-package facility reduction code for instruments currently in operation at the Dunn Solar Telescope. The module, `ssosoft.spectral.firsCal` is designed to carry out all necessary calibrations to bring the raw, Level-0 data to Level-1.5 status, with an optional module, `ssosoft.spectral.inversionPrep` that can additionally prep the reduced data for spectropolarimetric inversion. The `firsCal` module depends on `ssosoft.spectral.spectraTools` and `ssosoft.spectral.polarimetryTools`, which contain some of the lower-level functions used in the calibration routines.

Reductions are performed by setting up a configuration file (see Section 9), and then, in a python session;

```
import ssosoft.spectral.firsCal as firs
fred = firs.SpinorCal("configfile.ini")
fre.firs_run_calibration()
```

At this point, the reduction process will begin. There are, at a minimum, two points at which user intervention is required. These are unavoidable, due to the configurable nature of the instrument. These interventions are perfomed through interactive matplotlib plots, with user selections providing code inputs.

Once properly-configured, the `firsCal` module will perform the following corrections/reduction steps:

1. Determination of average dark current
2. Construction of average lamp flat
3. Construction of average solar flat
4. Iterative removal of spectral lines from solar flat to create a gain table
5. Demodulation from 8 polarization states to Stokes-IQUV
6. Determination of the net Müller matrix of optical train (from the polcal optics to the camera)
7. Combines orthogonally polarized beams into single IQUV state
    a) Registration of the instrument hairlines
    b) Registration of a chosen spectral line for alignment. This step also accounts for spectral curvature, and straightens spectral lines along the slit.
8. Wavelength calibration via comparison to FTS atlas
9. Application of optical train Müller matrix, as well as telescope matrix

[1] https://github.com/sgsellers/SSOsoft

a) Currently uses the saved 2010 measurements. Measurements taken in 2014, 2016, and 2017 were never reduced, as scientists at the time asserted that the matrix was stable. I plan to measure the matrix once more in mid-2025, but until then, these measurements are the best we have.

10. Removal of $I \rightarrow QUV$ crosstalk
11. (Optional) Removal of $V \rightarrow QU$ crosstalks
12. (Optional) First-order analysis of polarization degree, velocity, and line-width of user-selected lines.
13. Packaging of reduced data into FITS format, with a file structure that's *almost* SOLARNET standard.
14. (Optional) creation of context movie (using same underlying functions as SPINOR).
15. (Optional) alignment of data with HMI context images (using same underlying functions as SPINOR).

2 General Data Properties

2.1 What is FIRS?

The **F**acility for **I**nfra**R**ed **S**pectropolarimeter is a liquid-crystal variable retarder-based spectropolarimeter that works mainly in the near-infrared, though most of the hardware exists for a visible arm as well (though not the LCVRs and the beamsplitters we have in inventory mean that there's not much of interest we can put there). It is capable of obtaining maps in the X/Y/lambda space by stepping the spectrograph across the region of interest. Typically, it is operated on the He I 10830 Å spectral window, which includes the photospheric Si I 10827 Å line, however, with an LCVR swap, it can be used on the 1565 nm iron lines, which are incredibly sensitive to magnetic fields (Lande g-factor of 3!). This is a less useful mode, but interested parties should contact the instrumental PI, Haosheng Lin at IfA for advice if this is a desired mode. Note though, that at that wavelength, your spatial resolution will have dropped by a factor of 50%. Your available light will actually have increased significantly (as you're using a lower order on the grating), to the point where the notes I have indicate that a factor-0.3 ND filter is required to keep from saturating the camera. Swapping between the two lines is not something that can be easily done multiple times in a day, as it requires a hardware swap in the light feed (prefilter and LCVRs at a minimum, DWDM and slit-unit possibly as well).

FIRS is technically capable of multi-slit operations with the addition of a DWDM (dense wavelength division multiplexing) filter and a multi-slit unit. DWDM filters were developed for fiber-optics communications and cost an arm and a leg, but they provide remarkable square filter functions with very high throughput, which allows multiple slits to be imaged as narrow windows on a single camera chip. We have a 1.3 nm bandpass DWDM for He I, but no suitable multi-slit units. Given the spectrograph configuration, three slits would be tight, but probably do-able with some slit-overlap, and two slits would fit comfortably. Our only dual-slit unit, however, has a narrower slit width, which would cut down the

light (and undo the purpose of the DWDM, which is faster raster times). I contacted National Aperture for a tri-slit unit quote, but they never got back to me. This would be a cheap (sub-$2k) and quick upgrade to FIRS if a unit could be acquired. It's possible that our 40 μm quad-slit unit would work, it really depends on where the overlap in spectral windows would fall and the science use case. If Si I and He I are not contaminated by the neighboring windows, it would be a great value-add for the instrument. I suspect, however, that the overlap would render the Si-line useless.

2.2 The FIRS visible arm

FIRS was originally intended to be used as a dual-channel spectrograph with two camera. The infrared arm was intended for either He I or Fe I, with the visible arm providing spectropolarimetry of Fe I 6302 Å. **The visible channel is no longer functional.** In the mid-2010s, the beamsplitter that fed FIRS began to degrade, with the coating flaking off the optic. The original beamsplitter design reflected 630 nm as well as 900+ nm, however, a direct replacement could not be sourced. Instead, the replacement beamsplitter reflects 450-550 nm, and 900+ nm. We also have some beamsplitters that transmit 400-900 nm, and reflect 900+ nm, but these do block Ca K, and their reflectance band isn't great in terms of efficiency. With a previous optical configuration, I tested whether a Mg I *b* channel would be possible, with no success. With the grating in-use on FIRS, these wavelengths are on the 100th+ order, which is just too dispersed and dim for use with typical exposure times. I suspect that 6302 Å was about as low as that grating can go, and that's at the 90th (I think) order. Even if a replacement beamsplitter that could send 6302 Å to FIRS could be sourced (I haven't found any), the LCVRs for that arm are long gone. I think Haosheng has them, but he doesn't recall where they ended up. If the user desires a visible spectropolarimeter, use SPINOR. The new modulator is highly efficient across the entire visible range, and almost completely fringe-free. It's also usually a bit more than twice as fast as FIRS, and can be run even quicker if the light path is optimized for SPINOR use.

The rest of the visible arm hardware is still there, and has been tested. It works, though the computer needs the network info updated to communicate with Coconut (computer that controls the infrared arm). I don't know if there's anything it could be used for, and the camera is ancient, but it is there.

2.3 Level-0 Data Properties

Level-0 FIRS data are stored in FITS files (though not named `.fits`). These files are not FITS-compliant, and astropy will throw a series of warnings when you try to open the files. It's mostly the LCVR voltage header cards. Each file corresponds to a single co-added exposure. For science scans, this is also a single slit position. The filenames are of the pattern `firs.2.YYYYMMDD.HHMMSS.SSSS.RRRR`. All files from the FIRS infrared arm start with `firs.2`; `firs.1` referred to the visible arm. which is no longer operational. The rest of the tags are the date, the time, and the last two tags refer to the exposure number in the sequence and the number of times the sequence was looped. Observation types (flat, dark, polcal, science, etc.) are not recorded anywhere except for the the COMMENT card in the individual file FITS headers.

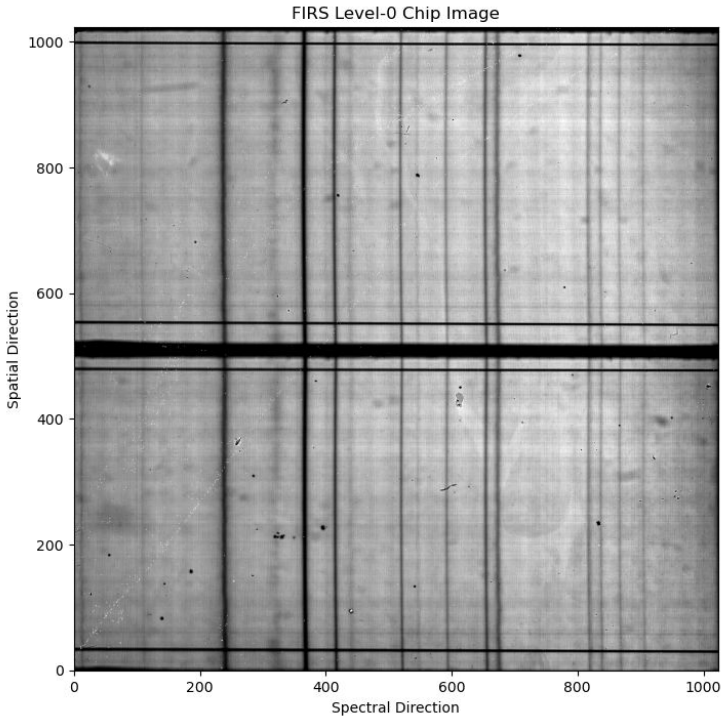


Figure 1: *FIRS Virgo Infrared Camera chip image during observation. Dark vertical lines are spectral lines, horizontal lines are hairline fiducials used in alignment. The most prominent spectral line here is a telluric line. The faint line directly to the left of the telluric line is He I 10830 Å.*

Obstype	Abbreviation	Typical Num. Files	Typical Coadds	Typical Exptime
Science	SCAN	> 1	5, 8, or 10	125 ms
Dark	DARK	16	Same as SCAN	125 ms
Solar Flat	SFLT	32	Same as SCAN	125 ms
Polcal	PCAL	36	Same as SCAN	125 ms
Line Grid	LGRD	Usually 200	1	125 ms
Target	TARG	Usually 200	1	125 ms
Pinhole	PHOL	Usually 200	1	125 ms
Lamp Flat	LFLT	32	2	1000 ms
Lamp Dark	LDRK	16	2	1000 ms

The observation type is usually abbreviated to a four-letter tag, wihch is entered by whichever observer is running FIRS that day. Since the tags are manually-set, the observations can sometimes be mis-tagged. Usually, you can tell what each sequence is by the number of exposures. Here are a list of common tags and their typical properties:

Since the files are often mis-tagged (or at least, often enough to have a fallback plan in place), the firsCal pipeline takes inspiration from the old IDL code, and writes a summary of all observing sequences it found into a text file. If something is mis-tagged, the user can inspect the file, change the tagging in the file, and re-run the code, at which point, if an observing summary file already exists, the values in that file supersede the values entered as FITS comments.

Within each file, the single data extension has the shape $(8, 1024, 1024)$, for 8 polarization states on a 1024×1024-pixel chip. Two orthogonally-polarized beams are imaged onto the same chip via a Wollaston

prism just before the focal plane. Unlike the SPINOR beamsplitter, the use of a Wollaston means that the two beams have the same orientation on the chip. Figure 1 shows an image of the FIRS chip. The sharp-eyed may note that there are a significant number of dust spots and hot pixels on the chip. These are usually removed fairly well through the dark and flat-fielding process.

2.4 Reconstructing a Stokes Vector from 8 polarization states

Unlike SPINOR, FIRS uses an LCVR-based modulation scheme. The LCVRs are pretty standard Meadowlark units, and have been in use for quite some time now. The LCVRs currently in use are optimized for He I 10830 Å. There is a spare set of infrared LCVRs in the FIRS optical cabinet that, according to Haosheng, were purchased by the NSO to be equally efficient for both He I 10830 and Fe I 15648, however, they were never calibrated or used. For the LCVRs currently in use, the 8 modulation states are redundant, with the same pattern repeated twice:

- **I:** $0.5 \times ++++$
- **Q:** $\frac{0.866}{2} \times --++$
- **U:** $\frac{0.866}{2} \times +-+-$
- **V:** $\frac{0.866}{2} \times +--+$

Apply the sign to each modulation state, sum the states, and normalize to reconstruct your vectors.

If the alternate LCVRs are to be used, you'll need to be able to modulate you input beam. There are linear polarizers both in the FIRS cabinet, and in the Chief Observer's office, as well as infrared retarders.

3 Darks, Flat-Fielding, and Gain Table Creation

FIRS takes two types of flat fields: lamp and Sun. These are used to construct gain tables, which are dark-corrected, normalized flat fields. They correct for large-scale and small-scale spatial inhomogeneities in the spectra. Sun flats are best for correcting things along the slit(s), but are insensitive to trends in the wavelength-direction. For this reason, lamp flats are also acquired. Ideally, both types of flats will receive a dedicated set of 16 dark frames taken with an identical exposure time. Sometimes, however, the observers will skip lamp darks if they're in a time crunch, as they can take quite a while, and often run up against the start of tours (and thereby tourists, who are also known as a source of stray light). If no dedicated lamp dark is available, the code will attempt to fake one using the dark rate from the solar dark. However, due to slight non-linearity in the detector, it can be pretty hit-or-miss whether the resulting correction is of high enough quality to use. Figure 2 shows an example of a decent lamp gain image.

Like SPINOR, FIRS primarily uses the solar gain for its corrections; the lamp gain is ultimately optional, but the solar gain is required. Again, like SPINOR, to use the Sun as a calibration source, the spectral lines must be iteratively removed. We use the same widget as SPINOR for this, so we can get a

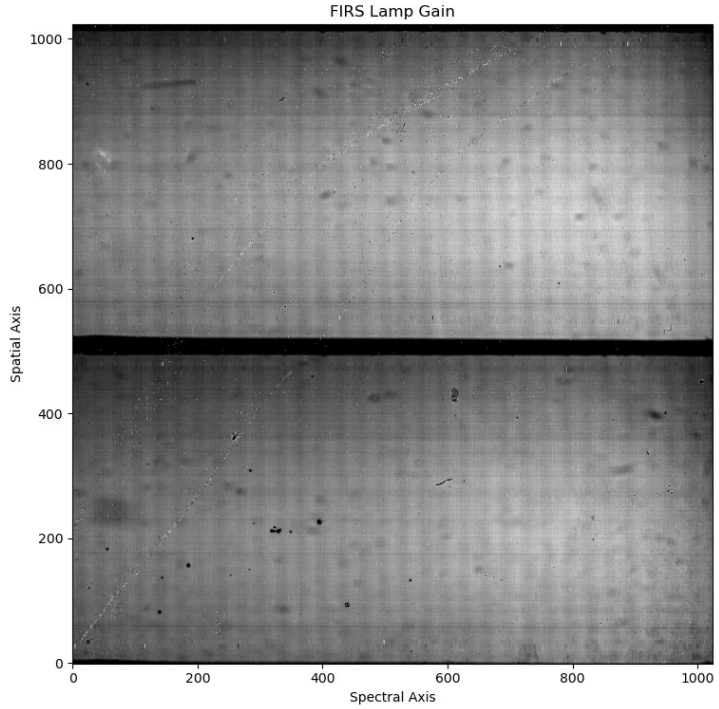


Figure 2: *FIRS Lamp Gain image*

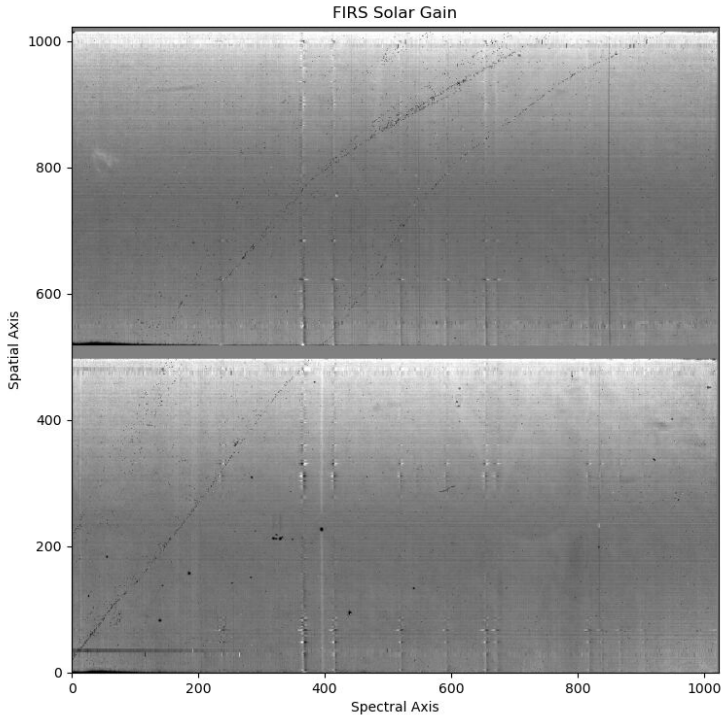


Figure 3: *FIRS gain table. There is some artifacting, however, the regions of interest are clear of the worst of the artifacts.*

wavelength calibration from the same selections. Typically, I select the Si I 10827 Å line and whatever strong solar line I can see on the opposite side of the spectral window. An example of the solar gain table can be found in Figure 3.

3.1 Beam, Slit, and Hairline Detection

See the SPINOR manual for a full description of how this works. FIRS uses the same functions, but with an additional call to try and detect a number of slit windows defined by the user. For all archival data, the number of slits is 1, but again, it would be good to have the ability to run a multi-slit unit in the future. Part of why the FIRS code is so overwrought and difficult to read is that it is trying to track an arbitrary number of slits as well as two beams and four hairlines per slit. I did not, however, have any multislit data to use in testing, so much of that functionality is untested, and I suspect that the automatic detection will not work if the slit windows overlap. The rest of why it's so overwrought is that I wrote it in a week and then debugged it in an airport. It was not the most robust development cycle, if I'm being honest.

Regardless, it has the same failover state as the SPINOR implementation, so if it fails to find the beams correctly, there is the capability for user selection.

As always, the positions of the hairlines are used to both align the two beams, and interpolate over the hairlines for the gain table creation step.

4 Polarization Calibrations

Like SPINOR (I'll be saying that a lot), the FIRS polcals are a combination of a baseline telescope matrix and the matrix of the Coudé table. The baseline telescope matrix defaults to the 2010 measurements. I was able to get a set of measurements in August 2025 from 460-1000 nm, but the reductions have proved trickier than what I'm able to do in the limited time between August and my departure from DST. The jupyter notebooks I was using to try the fitting is in the `/sunspot/solarchive/testing` directory if anyone wants to take a crack at it. The weather conditions during the attempt were less than ideal, with significant amounts of thin cloud, which may be part of the issue. It was the literal only window we got in a week and a half.

The Coudé table matrix is measured daily using the Port 4 calibration optics. Unlike SPINOR, which takes both clear frames without the polarizer and retarder in the beam, as well as series with only one of the calibration optics, the FIRS polcal starts with both optics in the beam. The linear polarizer remains at a fixed angle while the retarder is rotated in 10-degree increments through a full rotation. Once one series is completed, a second sequence is taken with a second polarizer position. The polarizer positions are 0° and 45°. This generally does a pretty good job. I've had the observers do a sequence of measurements with various angles of the polarizer to see whether it would be worth trying to include additional measurements in the sequence. The matrices I got for the optical train were just about identical

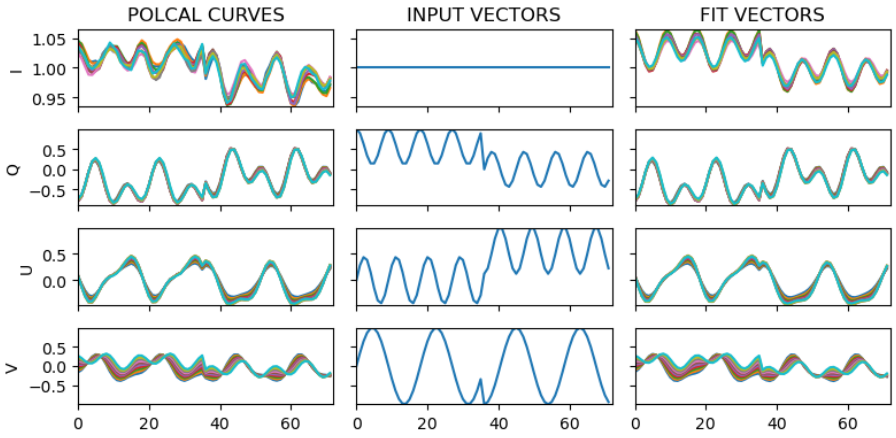


Figure 4: *Good Coudé table polarization calibration. Left column are the observed Stokes vectors for each overlapping section on the y-axis. Middle column shows what the Stokes vectors should after the calibration optics, but before going through the optical train. Right column shows the Stokes vectors resulting from the best-fit Müller matrix.*

to the typical series, so it's probably fine as-is, and there are plenty of other instrumental deficiencies that could be addressed more readily. The old IDL pipeline only used one of the two polarizer angles, and it's safe to assume that Christian Beck knows better than I do. Figure 4 shows an example polcal from 2025-06-13.

Since the FIRS polcals are taken without a clear frame, the normalization I use is slightly different, and the FIRS polcals have a bad tendancy to raise warnings about an invalid Müller matrix as a result. You can ignore these, but do keep an eye on the efficiencies. Fringes from the LCVRs may cause weird efficiency spikes if they're clipping the range used for polcals. You can adjust the `slitDivisions` or `polcalClipThreshold` parameter in the config file if the efficiency are too high (> 0.866) or low (< 0.1).

As a reminder, polcals should remain consistent for quite some time, so long as the optical bench setup is not changed, so if the observers were unable to get a good polcal, or if the polcal they got was through clouds, it is often better to grab the reduced polcal from a different day, and allow the code to load and apply those matrices.

5 Fringe Correction

FIRS uses LCVRs for polarization modulation, which can and do exhibit strong polarimetric fringes. Unfortunately, since the He I line is a wide, chromospheric line, and its polarization signals (paricularly Hanle-effect signals) tend to be low-frequency and low-amplitude. Worse, they are of a very similar frequency and amplitude as the fringe signal. Trying to isolate the fringes from the data frames is a difficult task, and requires a lot more care and attention than can be applied to every dataset in the archive. Nevertheless, we try. During reductions, I noticed that the fringe positions are mostly static. So what the code currently does is grab the flat map closest in time to the science map, and processes it as

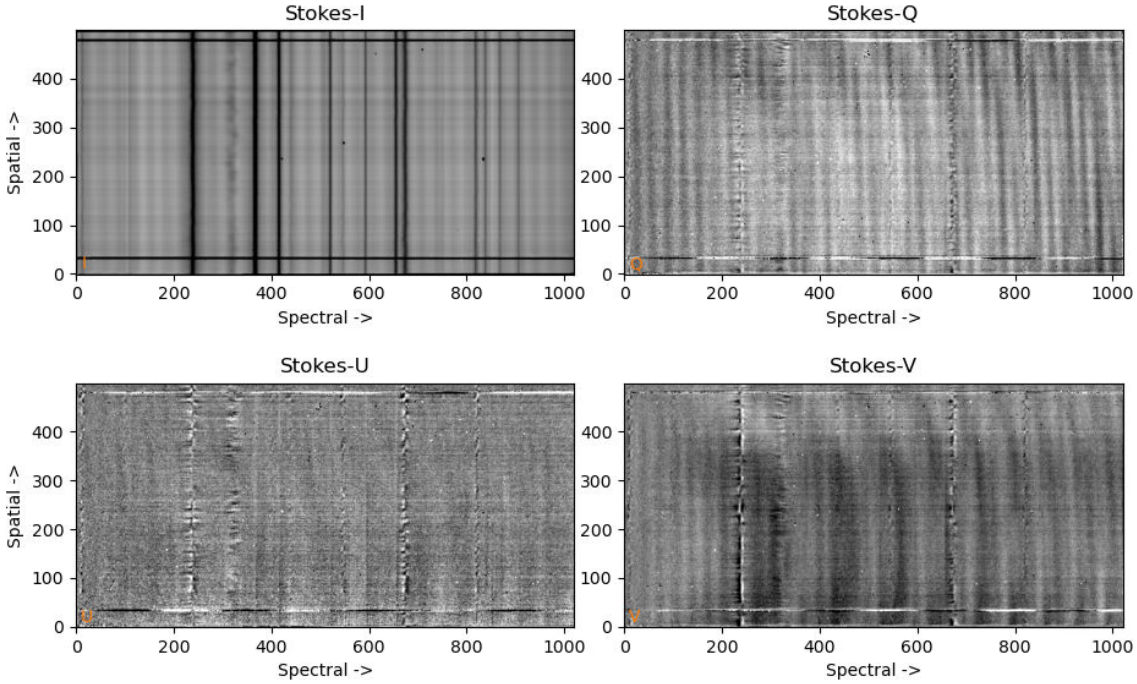


Figure 5: *FIRS Data with fringe correction applied. "Hey", you might be saying. "That looks pretty bad, fringe-wise", you'll opine. Well, here's what it looked like before...*

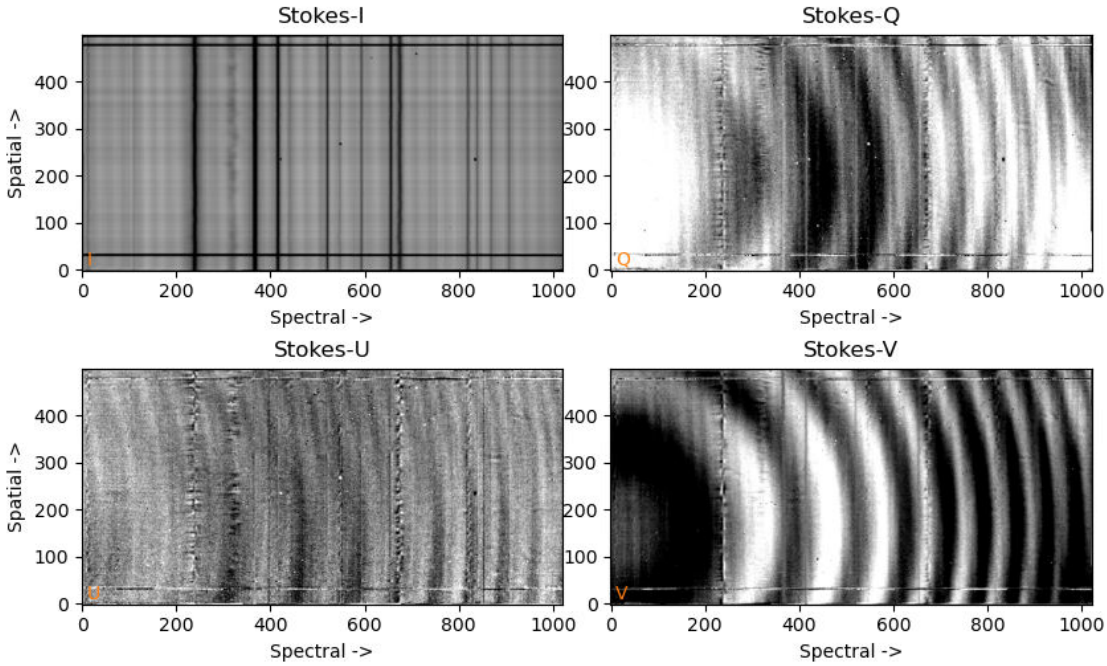
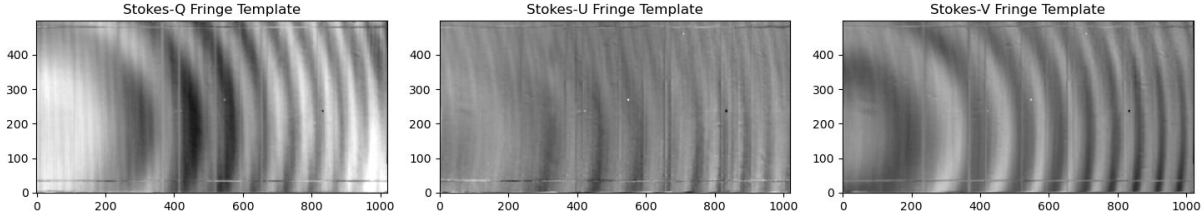


Figure 6: *That's the same data, scaled to the same colorbar extents. "Oh", you'll say. "I see", you'll concede. Here's the template used in the defringe process.*



if it were a science scan. Since there should be no significant polarization signals in the flat field, we can get a pretty good idea of the fringe pattern from the average flat field polarization data.

We smooth the average flat field images to try and get rid of any residual hairline signals or other defects, from there, the fringe template is saved, and can be used as a subtractive additional correction to the data maps. This works reasonably alright, but not perfectly. It tends to perform better at shorter wavelengths in the window (i.e., around He I).

I do have the IDL source code for a principal component analysis-based defringe scheme from Roberto Casini (see Casini et. al., 2012). This would work better (but slower), and might be able to rescue some of the older SPINOR data before the modulator swap. I never got around to rewriting it for Python, but it would be a good future upgrade.

The fringe correction can be turned off entirely by setting the `fringeMethod` config file keyword to any value besides "flat". It would be entirely valid to leave it turned off and rely on the "fringe" pseudo-atmosphere included in the newer version of Hazel for the correction. I've never tried it, but the telluric atmospheres are easy enough.

Check out Figure 6 to see what the correction looks like and its effect on the data.

6 Main Calibration Loop

With the gain tables created, the polarization matrix determined, and optionally, a fringe template generated, it's time for the main calibration loop. The general flow of this section is:

1. Load a Level-0 file into memory.
2. Apply dark and gain table corrections.
3. Demodulate the eight polarization states into IQUV.
4. Cut out the two orthogonally-polarized beams and however many slits were in use.
5. Use the hairline fiducials to sub-pixel align each beam and slit to a common position on the y-axis.
6. Use the spectral line selected during gain table correction to sub-pixel align each beam and slit to a common position on the x-axis.
7. Combine the two beams for each slit.
8. Apply telescope and Coudé table polarization corrections.
9. $I \rightarrow QUV$ crosstalk.
10. Redistribute Stokes-Q/U to correct for the parallactic angle.
11. Subtract of fringes if using a template.
12. Perform residual crosstalk corrections for $V \rightarrow Q$, $V \rightarrow U$, and $Q/U \rightarrow V$ (optional).

13. Perform basic analysis on spectral regions of interest (unlike SPINOR, default positions are included in the code, but the user can override this to choose ranges).
14. Create or update monitoring plots.
15. Move to the next slit position.

6.1 Beam and Slit Alignment

The code that aligns the two orthogonally-polarized beams for each slit is the same as used for SPINOR. It's a bit more reliable, since the configuration of the instrument rarely changes, and you rarely have situations where the beams aren't centered on the camera. Each slit, if there are multiple, is aligned to the same position as well, so the rasters line up. This does assume that the hairlines are perfectly straight across the entire slit unit. Since the hairlines are quite literally taped on, that means that if anyone ever changes the slit unit, it would behoove them to grab a micrometer, and make absolutely certain the hairline wire is level the entire way across. Otherwise, you'll have gaps in the structures between slits.

6.2 $I \rightarrow QUV$ Crosstalk Correction

The strongest component of crosstalk (mixing of polarization states) arises from Stokes-I. Ideally, the Stokes-QUV profiles should horizontal, centered around 0, with positive and negative polarities. Crosstalk from Stokes-I causes the continuum in Stokes-QUV to be other than 0. Traditionally, Stokes-$I \rightarrow QUV$ crosstalk is corrected by choosing a section of continuum, and determining a scale factor:

$$\alpha = \frac{\langle QUV(\lambda_{cont})\rangle}{\langle I(\lambda_{cont})\rangle} \tag{1}$$

And subtracting off Stokes-I as scaled by that value:

$$QUV(\lambda)_{corr} = QUV(\lambda)_{uncorr} - \alpha I(\lambda) \tag{2}$$

This approach is often imperfect, and leaves residuals. Most often, this is in the form of a variation in the Stokes-QUV continuum along the wavelength axis. As such, this correction is not the standard in SSOsoft, but can be enabled by setting the `crosstalkContinuum=x0,x1`, where `x0` and `x1` are the pixel coordinates of a section of continuum.

The default behavior, however, is to perform a linear fit:

$$QUV(\lambda)_{corr} = QUV(\lambda) - (m\lambda I(\lambda) + bI(\lambda)) \tag{3}$$

Where m and b are chosen such that a polynomial fit to $QUV(\lambda)_{corr}$ is minimized to 0. Essentially, the fit tries to get QUV centered around 0. This has worked quite well so far, but to my knowledge it is not a published or standardized calibration step, hence allowing the user to select the standard behavior.

6.3 Internal Crosstalk Correction

Internal crosstalk refers to the polarization state mixing caused by improper calibration of the telescope or optical train. In practice, it shows up strongest as contamination in Stokes-Q and Stokes-U from Stokes-V. The shape of the Stokes profiles is a dead giveaway —for typical Zeeman splitting, Stokes-Q and U should be symmetric around the line core, with a shape similar to the second derivative of the Stokes-I profile, while Stokes-V should be antisymmetric, with a shape similar to the first derivative.

Correcting this is a subject of ongoing debate. One method I've seen used in the past relies on the fact that sunspots have different structures in different polarities. One can compute raster images of sunspots in QUV, and fit the parameter α:

$$QU(x,y)_{corr} = QU(x,y)_{uncorr} - \alpha V(x,y) \tag{4}$$

Such that the linear correlation between $QU(x,y)_{corr}$ and $V(x,y)$ is minimized.

This method has the significant drawback that it only works in regions with extended polarized structures that can be safely assumed to be significantly differently-structured in different polarization states. SSOsoft has implemented a dual-phase experimental correction to try and break this limitation. Phase 1 is similar to the above, but applied to a single raster image, $QU(\lambda,y)$ and $V(\lambda,y)$. Phase 2 takes single profiles, $QU(\lambda)$ and $V(\lambda)$, and computes the cosine similarity between these vectors:

$$cos(\theta) = \frac{\mathbf{V}(\lambda)\cdot\mathbf{QU}(\lambda)_{\mathbf{corr}}}{||\mathbf{V}(\lambda)||\ ||\mathbf{QU}(\lambda)_{\mathbf{corr}}||} \tag{5}$$

Where $QU(\lambda)_{corr} = QU(\lambda) - \alpha V(\lambda)$, and α is chosen to minimize $|cos(\theta)|$ (as a cosine similarity of 0 denotes two orthogonal vectors). Essentially, it tries to make QU unique from V. I will be the first to admit that it's still a bit touchy, and the limits probably need to be reined in a bit. Internal crosstalks can be toggled with the `v2q`, `v2u`, and `u2v` parameters in the config file. A value of `False` will turn off the correction entirely, while `True` performs only phase 1 (a single correction value per slit position), and `full` performs both phase 1 and phase 2 (a correction value for every profile). All crosstalk values (including $I \rightarrow QUV$) are saved in a FIT file to the calibration directory, with the filename `CAMERA_MAP_X_CROSSTALKS.fits`, where `CAMERA` is the facility camera used (Sarnoff1/2, FLIR1/2), and `X` is the xth map reduced. Within this file, extension 0 is the header, while extensions 1 onward correspond to each crosstalk correction applied.

New as of Summer 2025: The above corrective method is a decent first start, and may still be useful, however, I think I've found a better solution. The above method assumes no $QU \rightarrow V$ crosstalk, and discards the excess V signal. This has the effect of artifically lowering the magnitude of V, as well as leaving in residual contamination. The new method assumes that the uncorrected signal can be modeled as a simple linear retarder, with some retardance value δ at an angle θ. These parameters are fit by minimizing similarity between QU_{corr} and V_{orig}. This basically reshuffles everything into the correct polarization state losslessly. It works quite well for photospheric data. It can be toggles with the

`internalCrosstalk` parameter in the config file. A value of `False` will turn off the correction entirely, while `True` fits a single retarder per slit position, and `full` fits a retarder for every profile. Try `full` first, and if too much Q/U is getting shuffled into V, switch to `True`. If it's still iffy, turn it off. This setting completely supersedes the `v2q`, `v2u`, and `u2v` keywords, and turns them off if `internalCrosstalk` is set to anything other than `False`.

Unlike SPINOR, which uses (almost) the entire spectral range for the crosstalk fitting, the presence of residual fringes in FIRS neccessitates that we be a bit more selective with our crosstalk ranges. By default, we choose a small window around the Si I line, and use that smaller range to try and determine the crosstalk, assuming that the fringes won't be as impactful on this range. The default central position is 10827.089 Å, with a ±2 Å wide window, but that can be overridden in the config file with the `crosstalkWavelength` and `crosstalkRange` keywords. Values provided to those keywords must be in angstrom.

6.4 Analysis and Plotting

Like SPINOR, the FIRS package will both produce and update plots as the calibration proceeds, and then perform some basic analysis on a few select spectral ranges. Unlike SPINOR, since FIRS almost always observes the He I window, I've included three default ranges around the Si I 1082.8 nm line, the He I 1082.9 nm line, and a shared window for the remaining two components of the He I 1083.0 nm line. Setting the `analysisRanges` keyword to "choose" will restore the behaviour of the SPINOR pipeline, and allow the user to select different ranges. This may be useful in datasets with flares, where the default ranges will not full encapsulate the behaviour of the much faster-moving plasma.

7 Alignment and Registration

One of the last steps in reducing a file is to attepmt a correction for the plate scale and pointing information. The DST solar coordinates are Not Accurate. They drift over time, and even at the best of times are only accurate to a handful of arcsec. Additionally, the plate scales included in the Level-1 files without registration are calculated from the lenses in the optical path, not all of which are achromats, and may have slightly different focal lengths than advertised. After the initial Level-1 file is written, if the `solarAlign` keyword is set, the code will attempt to correct each Level-1 file for both issues.

7.1 Plate Scale Adjustment

The plate scale adjustment is carried out using the linegrid raster files, which should be taken at the end of each day of observing. With the grid slide at the prime focal plane, FIRS is set to raster densely across the grid image. The grid lines are 1 mm apart, and the telescope prime plate scale in radians is the inverse of the primary focal length, so for DST, in arcsec/mm, the primary plate scale is $\frac{206264.806[''/rad]}{54864[mm]} \approx 3.76\ ''/mm$. Since there are no (significant) refractive focusing elements in the optical train before

this point, this value is unaffected by chromaticism that would alter the plate scale as a function of wavelength[2].

The grid rasters are used to update the plate scale of the FIRS Virgo camera via a simple peak-finding algorithm (the `scipy.signal.find_peaks` one, specifically). I hardcoded in the parameters for this function call based on some minor testing, and it isn't perfect. If the values it finds are more than 10% from the calculated plate scale, it skips updating entirely.

7.2 Solar Alignment

Alignment to solar coordinates uses the `ssosoft.tools.alignment_tools` module, which leverages the functionality of Sunpy Maps to do interpolation, image resizing, and coordinate tracking. First, a single image for each raster is constructed by averaging along the spectral axis. This gets you a photospheric-ish image (with non-square pixels). The code then fetches an HMI map corresponding to roughly the midpoint of the raster, and reprojects a section of it to the FIRS camera plate scale, orientation, and coordinates.

From there, an iterative cross-correlation is carried out. The first pass aims to get relatively close in the event that the coordinates are very far ($20^{'}$ or more) off. Successive iterations fine-tune the first by preforming the cross correlation on smaller central subwindows (apodisation segments). This will raise a number of Sunpy warnings about incomplete WCS information. Ignore them.

If the `verifyPointingUpdate` keyword is set to True in the config file, this will spawn a plot overlaying contours of HMI on the FIRS 2D raster image, and prompt the user to confirm that they're happy with the alignment. Proceeding from here will update the header coordinate values and WCS to the new grid. If the alignment is rejection, or if it is attempted on a very small raster image (about 40 steps or less), it will throw an error, and not continue with the next file. You'll have to restart the pipeline, and proceed past the reduced file.

If the initial coordinate guess is particularly far off, the code may not be able to find a decent alignment. Generally, if the actual center point of a given map is outside the bounds of the map coordinates, no amount of alignment will be able to fix it, and manual intervention will be required. The code uses the Stokes-I HDU for its coordinates, so what I generally do in this scenario is pull up Helioviewer data from the approximate observing time, find the X/Y coordinates that correspond to the approximate center of the map, and do the following:

```
import astropy.io.fits as fits
with fits.open("reduced_file.fits", mode="update") as hdul:
    hdul[1].header["CRVAL1"] = xcd-from-Helioviewer
    hdul[1].header["CRVAL2"] = ycd-from-Helioviewer
```

[2] As an aside, flexure of the entrance and exit windows can cause them to act as slight focusing elements. This would cause issues with the plate scale (and AO performance), but in practice, the effect is small enough to not matter. I'd estimate it to be on the order of $10^{-2} - 10^{-3}$.

```
hdul.flush()
```

and re-run the code. Continue past the prompts to re-make the maps (so the main cal loop doesn't run again), and see whether the new alignment works better.

Note that alignment in this way will only work when a region with structure is observed, e.g., a sunspot. I have managed to align quiet-Sun data, but it is a manual, labor-intensive process that is outside the scope of this pipeline. Contact me for advice if you're dead-set on trying.

8 Level-1 Data Structure

The reduced Level-1 data are stored in FITS standard, with detailed header information. The user can set the filename in the config file, but the code expects there to be the three following Python string-formatting tags in order: `{:s}`, `{:s}`, and `{:03d}`. These format to the YYYYMMDD formatted date, the HHMMSS formatted time at the start of exposure, and the number of steps in the map, respectively.

The Level-1 FITS files contain 7 extension, in the following order:

- **Ext 0:** Header only, no data
- **Ext 1:** Stokes-I data in the shape (ny, nx, nλ). Extension named `STOKES-I`
- **Ext 2:** Stokes-Q data in the shape (ny, nx, nλ). Extension named `STOKES-Q`
- **Ext 3:** Stokes-U data in the shape (ny, nx, nλ). Extension named `STOKES-U`
- **Ext 4:** Stokes-V data in the shape (ny, nx, nλ). Extension named `STOKES-V`
- **Ext 5:** Wavelength array in the shape (nλ). Extension named `lambda-coordinate`
- **Ext 6:** Metadata values in the format of a record array. Extension named `METADATA`, with the following five keys in the record:
  - `T_ELAPSED`: Time elapsed since start of exposure, at each slit step exposure start.
  - `TEL_SOLX`: The (approximate) x-coordinate at the center of the *telescope's* field-of-view. This may differ from SPINOR's FOV.
  - `TEL_SOLY`: Same as above, but y-coordinate.
  - `LIGHTLVL`: The DST guider tracks the amount of light incident upon it. This is a unitless quantity, but it's a good way to tell if there's significant cloud coverage or haze, as it will cause the light level to drop quickly. Typically, it's between 3.5 and 5.5 during observations.
  - `TELESCIN`: Astmospheric scintillation value from the telescope's Seykora scintillation monitor. The quantity is in arcseconds of seeing perturbation at a height of 70 m above the telescope. Lower is crisper seeing.

I made the choice to break the different Stokes parameters into different extensions for a few reasons:

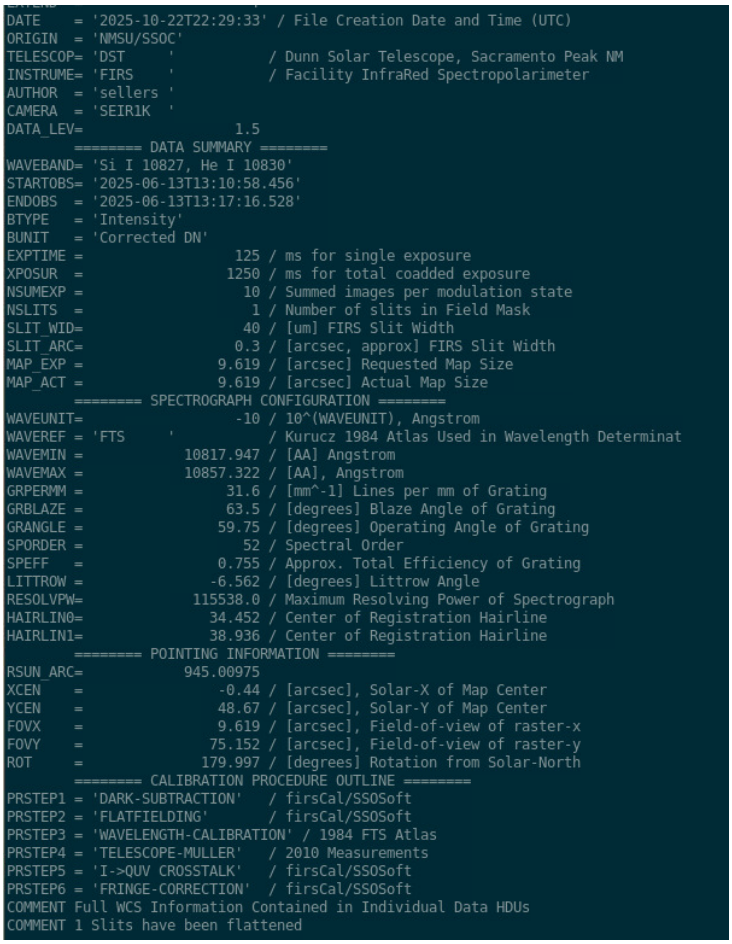


Figure 7: *Level-1 File, extension 0 header with human-readable observational metadata.*

1. FIRS data files can be quite large. Most modern FITS frameworks are capable of reading a single extension into memory, which is useful for working with FIRS data on weaker machines. Especially compared to SPINOR, FIRS maps can be 4+ GiB easily
2. Personally, I find 3D cubes easier to work with than 4D. Easier to remember which indices go where.
3. Simplifies the headers of each individual extension. The headers for each individual extension are nearly-identical, so there is some duplication, but the more common solar data frameworks in Python (sunpy, ndimage) should like this format more than a combined format.
4. I haven't tested it, but it should make it easier to use this data product with legacy tools like CRISPEX.

As always with FITS files, it behooves the user to check the headers. I've done my best to make them as complete as possible. See Figure 7 for an example of the extension-0 header, and Figure 8 for an example of the extensions 1–4 headers.

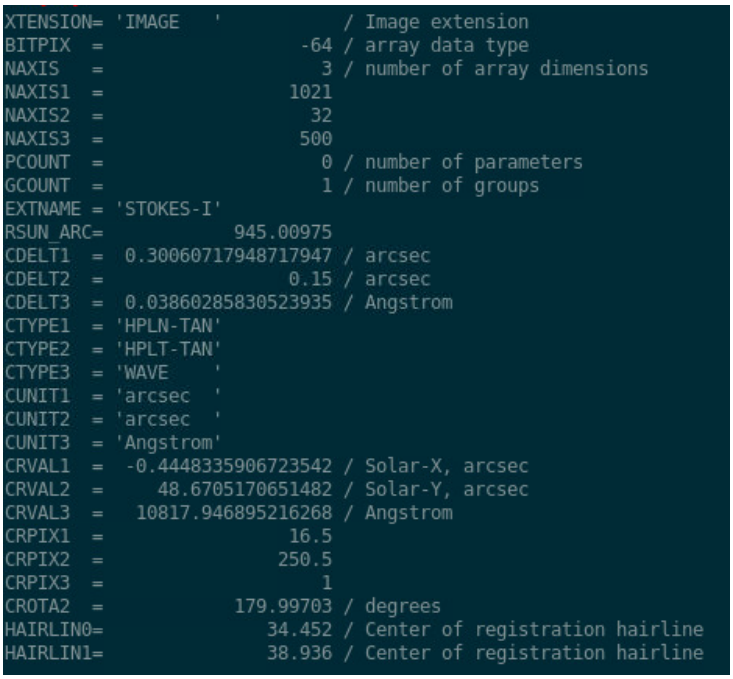


Figure 8: *Level-1 File, extension 1 header with FITS-compliant observational metadata.*

9 Allowed Config File Keywords (and what they do)

Unlike SPINOR, the FIRS config file has only one section: `FIRS` (two if you count `INVERSIONS`). This was done to allow SPINOR and FIRS to share one config file if necessary. Many of the parameters are shared between SPINOR and FIRS, but we'll break them out here as well. The sample config file in SSOSoft is well-commented, so make sure to read through it.

9.1 Required

These are the *only* required keywords. These will enable a very minimal run of firsCal with default values and no extra corrections.

- rawFileDirectory: path, location of Level-0 FIRS spectropolarimetric data.
- reducedFileDirectory: path, location to place Level-1 FIRS data.
- reducedFilePattern: str, Python-format-able string with 3 format spaces. The first two should be `{:s}` tags to insert the date and time, the third should be `{:03d}` to place the number of steps. Can be set to larger values that 3 to zero-pad the step numbers. My default filename is `{:s}_{:s}_firs_10830_level1_{:03d}_steps.fits`.
- reducedParameterMapPattern: str, Python-format-able string with 5 format spaces. The first two, again, are `{:s}` tags to insert date and time. The third formats to the wavelength band used. For default analysis ranges, this is, e.g., `HeI_10830`. The latter two should be float-formatting tags, e.g., `{:0.2f}` to insert the wavelength range used for parameter maps.
- tMatrixFile: path, Location and filename of the telescope polarization matrix file.

9.2 Verbosity

Controls how much the code bugs you as it runs.

- plot: bool, default False. If true, produces overview plots while the code runs.
- saveFigs: bool, default False. If true, saves those overview plots in the reduced directory.
- verbose: bool, default False. If true, spawns quite a bit of extra text to the terminal as an overview of where the code is.

9.3 Crosstalk

Controls how the code corrects for crosstalk.

- crosstalkContinuum: list of indices, not set by default. If given, sets the pixel indices used for the legacy I→QUV crosstalk method. I do not recommend using it.
- internalCrosstalk: False, True, or full, default False. Sets corrections for newest retarder-based crosstalk scheme. True tells the code to do a single fit per slit position, full tells it to do a fit per profile. Generally, "full" is too twitchy for these data. It works well for 7699, but not 10827.
- v2q: False, True, or full, default False. Sets the legacy internal crosstalk correction in the V→Q direction. The internalCrosstalk keyword overrides this regardless of what it is set to.
- v2u: Same as v2q, but in the V→U direction.
- q2v: Same as v2q, but in the reverse direction.
- u2v: Same as v2u, but in the reverse direction.
- crosstalkWavelength: float, default 10827.089. Wavelength of the center of the window used in internal crosstalk fitting. Defaults to the Si I line.
- crosstalkRange: float, default 2. Range in Å to either side of crosstalkWavelength for the fit window. Default is ±2 Å, for 4 Å total.

9.4 Data Cleanup Keywords

Optional keywords to enable or disable a few other general cleanup procedures.

- despike: bool, default False. Enables a median-filter based despike procedure when set to True. I usually keep it on.
- defringeMethod: str, default "flat". If set to flat, performs the flat-based defringe method. If set to anything else, skips that step entirely.
- spectralTransmission: bool, default False. Sometimes, when no lamp flat is available, there's significant changes in the brightness of the spectrum in wavelength. This attempts to flatten out the spectrum.

9.5 Spectral Location and Optical Path Keywords

These can change where the code assumes you are spectrally. If FIRS is used with something other than the usual He I window, you'll need to alter these. The optical path keywords are mainly there if Haosheng ever decides to try mxSpec again on the UBF port. The pipeline may be adaptable to mxSpec data. I'm honestly not sure.

- centralWavelength: float, default 10830. Not actually the central wavelength, but the identifier wavelength in Angstrom. Changes the range used for wavelength calibration and also unsets a bunch of defaults if it's set to anything other than 10830.
- spectralOrder: int, default 52. The Fe I 1565 nm line is on the 36th order of the same grating. Unless you want to use FIRS as a coronagraph (not recommended, as the DST does not have an occulting disk), this parameter will probably be one of those two numbers.
- gratingRules: float, default 31.6. Grating constant in lines per millimeter. Change only if the FIRS grating has been swapped. There's another grating in the FIRS cabinet that gives (if I recall correctly) slightly higher dispersion, but very few that offer lower dispersion.
- blazeAngle: float, default 63.5. Blaze angle of the grating; only change if you swapped the FIRS grating.
- gratingAngle: float, default 59.75. Incident angle of the FIRS grating. This isn't explicitly set in the observer's GUI, like it is with SPINOR, nor is it in the FIRS Level-0 headers. But for He I and Fe I, it's 59.75 degrees. Change if you change the grating or try to use FIRS for some other set of lines.
- pixelSize: float, default 20. Size of the sensor pixels in micron. I really don't think we'll be getting a new camera, but you can change the pixel size if we do.
- slitCameraLens: float, default 780. Focal length of the lens that images the field onto the slit unit. Again, that part of the optical path hasn't been changed in years, but the switch is there.
- DSTCollimator: float, default 1559. Focal length of the collimator element after the prime focal plane. On Port 4 it's a 1559 mm mirror.
- spectrographCollimator: float, default 1524. Focal length of the collimator mirror that places the pupil image on the grating. For FIRS, it's also the camera lens (with a pair of beam extenders).
- cameraLens: float, default 1524. Unless you change those beam extenders for some reason. In which case, you'll need to figure out the effective focal length of the camera lens and enter it here.
- telescopePlateScale: float, default 3.76. Plate scale at the primary focal plane in arcsec per millimeter. I guess if you're using this pipeline for a different telescope and don't want to change the hardcoded defaults?
- analysisRanges: str, default "default". If set to "choose", allows the user to select the ranges used in the parameter map analysis and plotting.

9.6 Polcal Tweaks

These will affect the polarization calibration module.

- slitDivisions: int, default 10. Number of overlapping vertical sections to split the combined beam into for creating a polcal. Decreasing can help in situations with low signal or signal clipping.
- polcalClipThreshold: comma-separated float, default 0.5, 1.5. Default range to clip polcal signals to, expressed as a fraction of the mean intensity of the beam.
- polcalProcessing: bool, default True. Set to False if you don't want to apply the gain tables to your polcal. If the polcal is being processed from scratch from a different day (instead of using that day's reduced polcal), you may need to turn this off.
- calRetardance: float, default 83. Retardance of the Port4 calibration retarder. See Figure 9 for Christian Beck's summary of the measurements made of the optic retardance. Generally, if something indicates Christian measured it at any point, believe his measurements above all others. I'd fight god if there was a note from Christian Beck with instructions.

9.7 Additional Code Tweaks

These keywords alter some low-level behaviours of the code. Mostly, you shouldn't need these.

- totalHairlines: int, default 4. Number of hairlines that should be visible across two given beams. Generally, there'll be 4: two upper, and two lower. If you're off-limb, or if FIRS was misaligned, you may need to set it to 2.
- intensityThreshold: float, default 0.5. Fractional intensity that the code uses to start searching for beams and slits. The code that does this is recursive now, so it runs the gamut trying to get the correct number of slits and beams, so you shouldn't have to set this.
- hairlineWidth: float, default 3. Width of a hairline in pixels for removal from flats and filtering beam selection results. You shouldn't need to mess with this unless you install some very thick hairlines.
- alignSelect and hairSelect: Not implemented. See the SPINOR Cal manual, these would allow the user to select the alignment regions used. I never implemented it for firsCal because I didn't want to think about how it would interact with multi-slits. If you ever need the functionality, contact me. I can hack it together pretty quick probably.

9.8 Multi-Slit Enabling Keywords

If you ever do install a multi-slit unit, there are the keywords to make the code deal with it. Keep in mind that it is completely untested.

- nSlits: int, default 1. Number of slit windows in the unit.

- slitWidth: float, default 40. Width of a single slit in microns.
- slitSpacing: flat, default 0. Spacing between slit centers in mm for field sizing and combinations.

9.9 Registration and Tracking

These govern the tag-on procedures that happen after reductions. It's usually useful to keep these on, but they mostly default to off.

- contextMovie: bool, default False. If True, produces a movie of the data reductions across the raster. Always good to have.
- contextMovieDirectory: path, default same as reducedFileDirectory. Tells the code where to save the context movie if one is generated. I usually put all the context movies in a single directory in `level1/context_movies`.
- solarAlign: bool, default False. If True, attempts to align the raster with solar coordinates using the same methods as SPINOR. Remember, if it doesn't work and the raster does have identifiable structure, it means the initial coordinates were very far off and need to be manually bumped.
- verifyPointingUpdate: bool, default True if solarAlign is True. Allows the user to reject alignment updates.

9.10 Polarization Modulation Keywords

Most of these are not going to come up unless you start swapping LCVRs to go between He I and Fe I.

- siteLatitude: float, default 32.786. I don't know why you would ever set this instead of changing the hardcoded value. It's used for redistributing Q and U for parallactic corrections.
- qModulationPattern: comma-separated int, default -1,-1,1,1. Use to alter the demodulation pattern used in the code. The pattern denotes the linear combination of polarization states, which is then multiplied by the normalization factor.
- uModulationPattern: comma-separated int, default 1,-1,1,-1
- vModulationPattern: comma-separated int, default 1,-1,-1,1
- polNorm: comma-separated float, default 0.25,0.433,0.433,0.433. Normalization factors used in Stokes vector reconstruction.

Table 1: Retardance measurements for the DST ICU.

Source	589	630	656	854	1083	1565
Elmore/design	86	83	82	82	93	111
Beck/FIRS	–	88	–	–	82	116
Beck/SPINOR	–	–	–	88	83	105
Sarah/Tom(?)/FIRS	–	77	–	–	87	102
Ali/IBIS	87	86	84	83	–	–

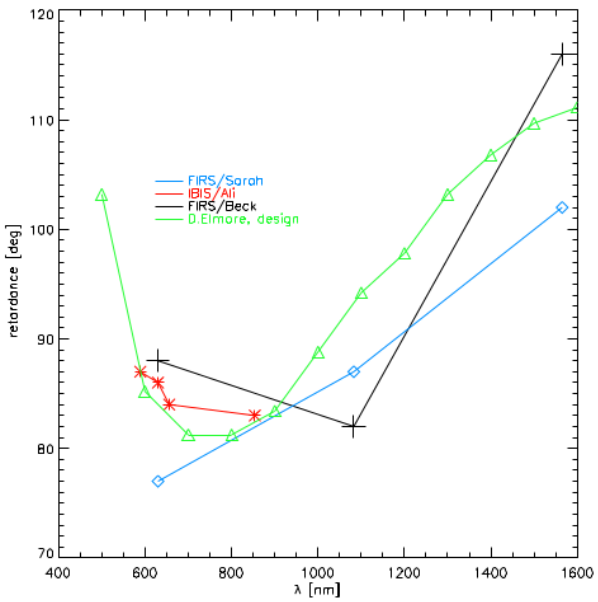


Figure 7: Retardance measurements for the DST ICU. *Black line/crosses*: FIRS data, Beck. *Red line/asterisks*: IBIS data, Ali. *Blue line/diamonds*: FIRS data, Sarah (or Tom ?). *Green line/triangles*: possible design, David.

Figure 9: *Summary of previous measurements of the Port 4 retarder. These are quite old at this point. It may be worth re-measuring with the new SPINOR modulator at some point.*