Sean Sellers

October 13, 2025

SSOSoft for SPINOR

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.spinorCal` 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 `spinorCal` 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 8), and then, in a python session;

```
import ssosoft.spectral.spinorCal as spin
sred = spin.SpinorCal("CAMERA", "configfile.ini")
sred.spinor_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 `spinorCal` 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.

2 General Data Properties

2.1 What is SPINOR?

The **SP**ectropolarimeter for **IN**frared and **O**ptical **R**egions is a spectropolarimeter that is the evolution of the old Advanced Stokes Polarimeter (ASP), which operated at the DST in the 1980s. It is built on top of the Horizontal Spectrograph platform, with the addition of a rotating modulator and a polarizing beamsplitter. It is capable of obtaining maps in the X/Y/lambda space by stepping the spectrograph across the region of interest. It can be thought of as three separate components:

1. The Spectrograph component consists of the entrance slit, the rastering box, and the grating. The entrance slit is adjustable via micrometer, and isolates all but a narrow stripe of the region of interest in the telescope's FOV. This slit image is collimated to place an image of the telescope entrance pupil on the spectrograph grating. This grating is interchangeable, but mostly we use a grating with 308.57 lines/mm and a blaze angle of 52 degrees. The slit unit and collimating lens are contained in a white box, with a worm gear that allows it to step the slit across the telescope FOV. Just behind the slit is the SPINOR polarizing beamsplitter, which separates the beam into spatially-displaced orthogonally polarized components.
2. The Modulator component is the rotating modulator. This sits and spins in a housing that is usually kept directly in front of the spectrograph slit. By sampling at 8 different rotation angles between 0 and 180 degrees, we can reconstruct the full Stokes vector. See Lites 1987 for the mathematical discussion of *why* this works. The current modulator[2] was provided to us by Roberto Casini (HAO) and David Harrington (NSO) as a replacement for the original modulator, which had issues with polarimetric fringes. This modulator has an approximate retardance of 100–130 degrees (close to a 3/8-wave plate), but as you'll know if you read that Lites paper, the degree of retardance does not actually matter. The rotating mount can also be installed in the port 4 "gray box" to allow the AO

[2] If you'd like details on the modulator, it is number G18126, as described in this paper: DOI 10.1117/1.JATIS.6.3.038001

to compensate for modulator-induced beam wobble. This has not been much of a factor since the new modulator was installed, however.

3. The Camera component(s) are the set of up to four cameras compatible with SPINOR. There are two CCDs manufactured by Sarnoff, with 16 μm pixels in a 1024×512 chip size, and two InGaAs cameras manufactured by FLIR, with 25 μm pixels in a 640×512 chip size. These are synced to the modulator rotation.

2.2 Level-0 Data Properties

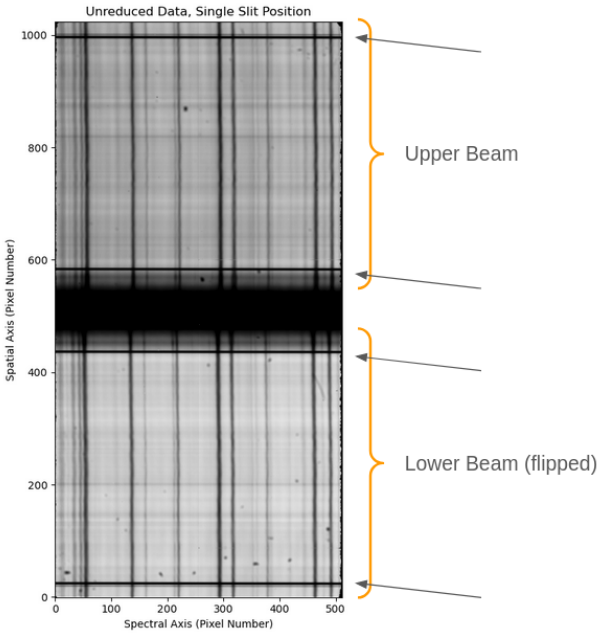

Figure 1: *Sarnoff Camera chip image during observation. Dark vertical lines are spectral lines, horizontal lines marked with arrows are hairline fiducials used in alignment.*

Level-0 SPINOR data are stored in FITS files following standard FITS convention. Multiple steps (slit positions) during observations are combined into a single file, up to a file size of 2GB, due to antiquated file size limits on some of the older computers. That means that a map may be split over multiple files, unfortunately. In a given FITS file, the 0th extension contains header information only, with subsequent extensions containing each the data at a single slit position. The shape of the data is (8, ny, nx), where each of the 8 images in the extension correspond to the 8 modulation states observed during a half revolution of the modulator. The size of each image, ny and nx, are set by the camera in use. Both orthogonally polarized beams are imaged on the same chip, displaced in the spatial direction. See Figure 1 for an example of what that looks like.

Level-0 files are named with the file format `YYMMDD.HHMMSS.0.cccXX.c-hrt.TYPE.iiii.fits`. Here, "YYMMDD" is the date in year/month/day format and "HHMMSS" is the time. Unfortunately, the time is local, not UTC. UTC times are contained in the file headers. "XX" denotes the camera computer used to connect the camera to the SPINOR utility. That's important for the operators, and mostly irrelevant for you. "TYPE" is the observation type, and can be "map" for science observations and fiducial maps (air force target, line grid, etc), "lamp.flat" for flats taken with the calibration lamp, "solar.flat" for flats taken on the sun, and "cal" for polcal observations taken using the port-4 polcal optics. Finally, "iiii" is the number of that observation taken on the day. The counter is independent for observation type, and remains the same for files that were split by the 2GB file size limit. The fact that number remains the same is, in fact, the only easy way to tell that a file was split.

A summary of Level-0 FITS headers can be found in the appendix.

2.3 Reconstructing a Stokes Vector from 8 polarization states

When we installed the modulator, we did not install it with the fast axis at a particular rotation. The actual orientation is not important so long as the rotation and strobe sequence is repeatable. I did some work to nail down the orientation, and back out the modulation sequence. This sequence could change for any of the following reasons, off the top of my head:

- The modulator was not epoxied into its mount, as we were unsure whether it would be a permanent solution. It is friction fit only, and has not slipped in the last few months, but it may in the future.
- The home and reference positions of the rotator can be reset from several of the DST GUIs. They shouldn't be, but changing it would be simple as a misclick or someone messing around.
- If the modulator mount is ever reversed or move on the table, the sequence could be reversed.

Determining a new modulation sequence from scratch would be relatively simple, and can either be done the "easy" way or the "correct" way. The easy way requires no additional observations to be made, just access to historical polcals. Essentially, you can determine the average difference between the upper and lower beams for each modulation state for a given polcal state, and compare to a historical polcal. The curves will be offset if the modulation scheme is different. You can apply this offset to the old modulation scheme and move on with your life. The correct way would be to take a quick set of SPINOR data at a quiet position on the solar disk while adding some optics in front of the modulator, specifically, a linear polarizer and a quarter-wave plate, both of which can be found in the DST inventory. Using these optics, you can generate light that is purely Q, U, or V polarized. Refer to the Lites paper on how the modulation state curves would look, and then determine your modulation sequence.

At installation, the demodulation sequence for the new modulator was

- **I:** $+ + + + + + + +$
- **Q:** $- - + + - - + +$
- **U:** $+ - - + + - - +$
- **V:** $+ - - - - + + +$

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

3 Darks, Flat-Fielding, and Gain Table Creation

SPINOR takes two types of flat fields: lamp and Sun (there's a laser option, but no laser implemented). 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, but are insensitive to trends in the wavelength-direction, to include fringes, for reasons

that will be described later. For this reason, lamp flats are also acquired. Both sets of flats take 4 dark frames at the beginning and end of the flat series, for a total of 64 darks per flat file (4 at the start, 4 at the end, 8 modulation states per dark frame). In general, the darks acquired during lamp flats are used only for the lamp flats, as they are susceptible to stray light contamination to a greater degree than Sun flats.

Darks are subtractive corrections, while gains are multiplicative. To apply dark and gain corrections, you:

- Subtract the average dark frame from your data frame
- Divide the dark-corrected data frame by your lamp gain (which is the mean-normalized, dark subtracted lamp flat)
- Divide the lamp-gain-corrected, dark-corrected data frame by the solar gain.

3.1 Lamp Flats

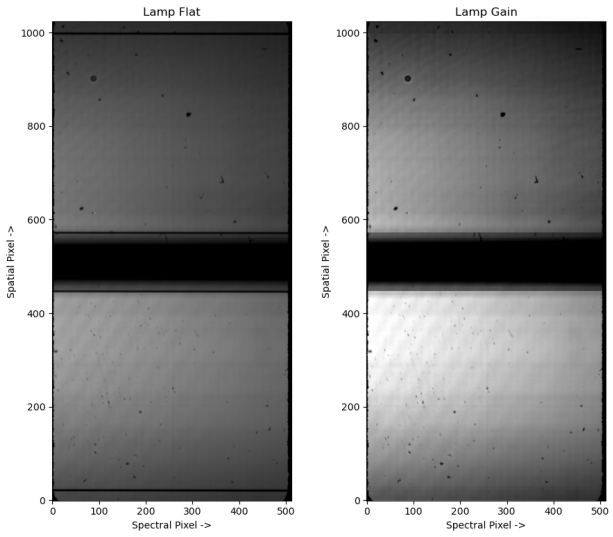

Figure 2: ***Left:*** *Dark-corrected average lamp flat.* ***Right:*** *Processed lamp gain*

Figure 2 shows the dark-corrected lamp flat and minimally-processed lamp gain for a section of spectrum around 5250 Å with the Sarnoff 1 camera. Notice the speckles visible on the upper beam, and the fringes on the lower. The speckles are dirt on the window in front of the camera chip, while the fringes are from interference within the window itself. The window is something like $\approx$ 2 mm thick on the Sarnoff cameras. This causes fringing in the intensity image. This fringing is worse with increasing wavelength. At, say, 8542 Å it is quite significant, and can really only be corrected by a lamp flat. Even that will not fully correct for the fringes, thanks to the alignment between beams and the wobble of the DST main light feed.

Speaking of the DST wobble, you will also notice that the hairlines present in the left-hand image are absent in the right. During the lamp gain creation, the hairlines are automatically detected and interpolated over to remove them from the final image. This is done because the hairlines are formed at the entrance to the spectrograph, and the wobble in the DST beam[3] will cause them to move around on

[3] The DST beam wobble is a vexatious feature of the telescope that causes misalignments as the table rotates. When changing the rotation of the telescope, the beam will have to be realigned and the AO re-calibrated. This is caused by bending of the mirror support rods in the turret, and is quite possibly unsolvable with our current staffing levels.

the camera chip over the course of an observing day. To avoid strong residuals in the final product, we interpolate over the hairlines.

Depending on the illumination of the lamp, this may be the first place in which the pipeline user encounters a need for intervention.

3.2 Beam, Slit, and Hairline Detection

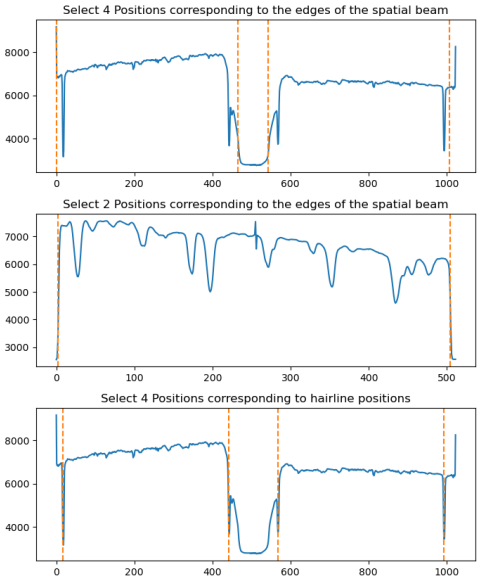

Figure 3: ***Top:*** *Manual selection of beam edges.* ***Middle:*** *Manual selection of slit edges.* ***Bottom:*** *Manual selection of hairlines.*

I'll be the first to say it; the beam, slit, and hairline detection function is somewhat overwrought. It works by intensity thresholding and masking to try and detect the expected number of beams, hairlines, and slits (though this last is relevant only for FIRS). It starts by making a mask of 1's and 0's based on an initial intensity threshold. It then takes the gradient of that profile in the X and Y direction to try and detect the edges of the mask, as well as temporary dips. Dips are labeled, while beam edges are labeled as such. The function is also recursive, and will adjust the intensity threshold of the mask if no good solution is found.

This method works *alright*, but it can fail, particularly in cases where the signal is low, or the hairlines are very close to a beam edge. When this happens, rather than completely fail and stop the reduction process, it will prompt the user to make selections for the beam edges, slit edges, and hairline centers. It does this by creating a plot for each of these selections, and prompting the user to make a number of selections based on how many beams, slit, and hairlines are expected. For something like shown in Figure 2, it will create three windows, prompt the user to select four beam edges (for the top and bottom of each beam), two slit edges (for the left and right boundaries of the chip), and four hairlines (two per beam, though FLIR may have only one per beam). An example of this can be seen in Figure 3. As the data progresses through the pipeline, this sequence may have to be repeated up to four times:

- Once during lamp gain creation to detect and erase hairlines
- Once (the main time) during solar gain creation to select the beam edges and hairlines.
- Once during the solar gain creation (again) to detect and erase hairlines.
- Once during the first step of the final map reduction to mark the hairlines for the final alignment.

The lamp gain image is saved to the calibration directory, with a filename `CAMERA_LAMPGAIN.fits`, where `CAMERA` is the facility camera used. Within this file, the lamp gain image is in the 0th data extension.

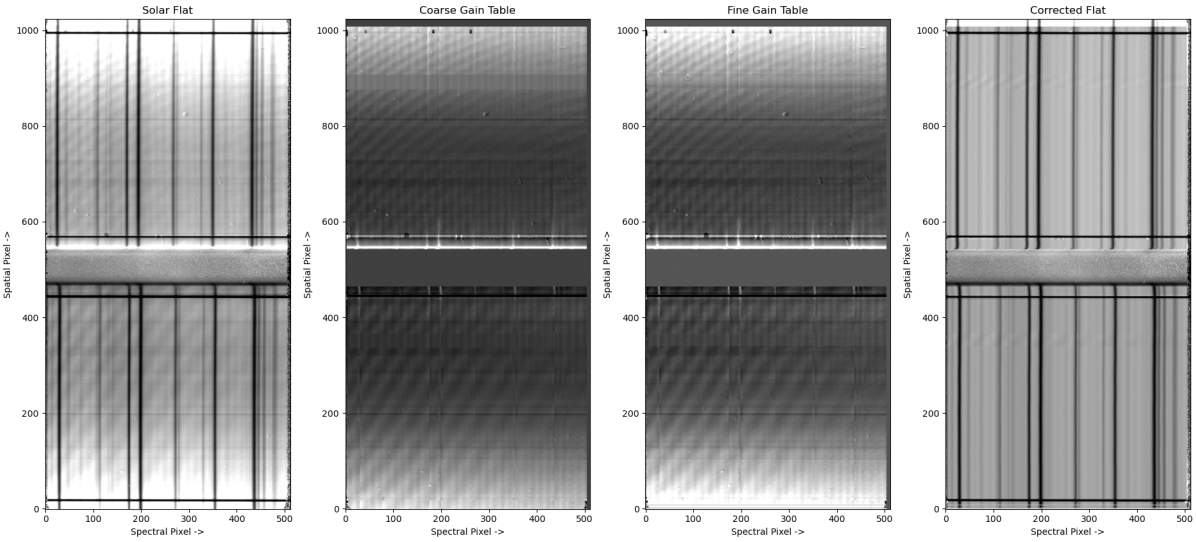

Figure 4: *Stages of solar gain creations.* ***Left*** *is the solar flat after dark and lamp gain correction.* ***Second*** *is the coarse gain table, and* ***Third*** *is the fine gain table. If the instrument is well-aligned, these should be very similar. There are differences between the two, but they are minimal.* ***Right*** *is the solar flat again, now with the gain correction applied. Note that the intensity variations across the field are suppressed relative to panel 1.*

3.3 Solar Flats and Solar Gain

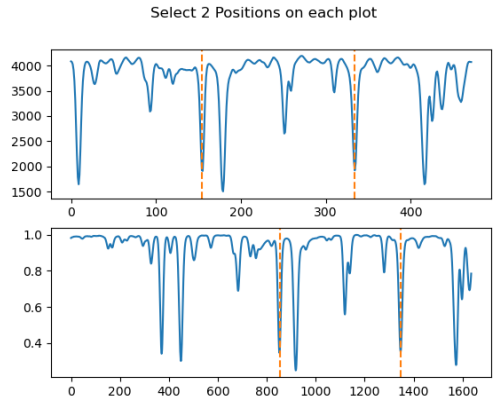

Figure 5: *Example of line selection for SPINOR from the FTS atlas at 5250 Å.* ***Top:*** *SPINOR average spectrum with lines selected.* ***Bottom:*** *FTS atlas with the same two lines selected. The lines selected are Fe I 5250 Å and Fe I 5253 Å. The exact line centers will be determined by the code, so there is no need for the user to be exact in the selection.*

How do you create a flat field that's evenly illuminated with high signal in observing-like conditions for a solar telescope? You use the Sun. For the DST, flat-fields are taken by setting the AO deformable mirror to "unflat" (i.e., sending a repeating pattern over it to blur the image) and randomly dithering the telescope at or near a quiet region of disk center. This works very well for imagers, but the positions and presence of spectral lines are not affected by this process, meaning that for SPINOR and FIRS, additional processing is require. See Figure 4 for an example of a solar flat before and after processing. One cannot simply divide the science image by a simple solar flat, as the spectral lines themselves are the object of study.

Rather, an iterative method is used to remove the spectral profile while preserving the pixel-by-pixel sensitivities of the detector. The locations of the two beams, edges of the slit, and hairline positions are detected from the flat image first. Then, the spectral line to be used for detrending the solar spectrum must be selected. Since SPINOR is a configurable instrument, it is impractical for the code to have an

automatic selection method. Instead, this is (under ideal conditions) the first time that interaction is required from the used. A window will be produced with an average SPINOR spectrum in the top panel, and the FTS reference spectrum in the bottom panel. The user will have to select two lines from the top panel, and two lines from the bottom panel. An example of this can be found in Figure 5. This selection actually performs three different corrections: the selection of a line for gain table creation, determining whether the spectral image is flipped, and the wavelength grid registration, which will be updated throughout the reduction process. To this end, the wavelengths selected by the user should be either both of solar origin, or both of terrestrial (i.e., telluric) origin. Two solar lines should be used if the user does not want to have to detrend the motions of the earth and Sun in the final product (i.e., a relative wavelength scale). If the user desires an absolute wavelength scale, two telluric lines should be selected. The spectral flip routine is necessary as the exact placement of lenses on the table may cause the spectrum to be flipped. The code is pretty good at figuring out whether the spectrum is flipped, so long as the same two lines are selected in any order and there's at least *something* in between them. So don't choose two adjacent lines.

For a good gain table, the first spectral line selected should be relatively unblended and prominent. For the 8542 Åwindow, this can cause problems, as the weakness of the photospheric lines in the wings are too weak to be good gain table lines. Instead, the core of 8542 should be used for this purpose.

Once the lines are selected, the code creates a coarse and a fine gain table. The general way this is done is by taking an average spectral profile, then using the selected line core to determine the subpixel shift between the average profile and each profile along the slit. This done, the average profile can be divided from the singular profile along the slit. For the coarse gain table, a single average profile is used. For the fine gain table, a median filter is applied to the flat field in the y-direction, and the local average of this image is used. For the default SPINOR pipeline, the surrounding 12 pixels are used for this profile. There are some scenarios in which the coarse gain table may be a better correction than the fine. There is no switch in the SPINOR pipeline to use the coarse gain table instead of the fine. If this behaviour is desired, I would recommend you open the FITS file containing the gain tables in "update" mode, and replace the "GAIN" extension with the data found in the "COARSE-GAIN" extension.

This process is the reason that spectral fringes may remain in the final data product. As you may have already guessed, this shift and divide method will inevitably preserve variations in the x-direction to a greater or lesser extent. Use of a lamp flat can mitigate, but not eliminate these residuals. Fortunately, the linear combination of polarization states into the final Stokes images removes these fringes in Stokes-QUV (though polarization fringes will remain; this has not been a major issue with SPINOR since the new modulator was installed in October 2025). The remnants in Stokes-I will cause issues with velocity and temperature determination. I hope to have a solution based on Roberto Casini's PCA method working eventually, but that's further down the line.

The solar gain tables are save to the calibration directory with the filename `CAMERA_X_SOLARGAIN.fits`, where `X` denotes the xth solar gain created. The code attempts to grab a flat field that is closest in time to the science map it is reducing. Within this file, the 0th extension contains a header with information

that's important to the code. Subsequent extensions contain the solar flat image, dark image, coarse gain image, fine gain image, beam edge positions, hairline positions, slit edge positions (edges in the x-direction on the chip), and shifts between the upper and lower beams.

4 Polarization Calibrations

Every optical element has an associated Müller matrix. Flat mirrors, particularly at 45-degrees, are a particularly strong polarizer. Optical windows, particularly those being stressed by vacuum, act as retarders. Oxidation on mirror surfaces will also s- and p-polarization states unevenly. The location of your observatory will cause Stokes-Q/U mixing based on the observer's latitude and the altitude of the Sun.

The DST, like most solar telescopes (except THEMIS) is a very strong polarizer, and its optical train on the Coudé table further is also a strong polarizer. By my count, as of June 2025, there are about 25 optical elements between SPINOR and the sky that are not part of the polarimeter. The net effect of the optical train is to mix polarization states, visible as "crosstalk" in the reduced Stokes vectors. The optical train can be broken in to two components, and a net Müller matrix determined for both components. Component 1 is the telescope itself, consisting of the entrance window, turret flat mirrors, primary mirror, and exit window. Component 2 is everything after the exit window. That's everything on the Coudé table and the spectrograph itself. So your final Stokes vector becomes:

$$\overrightarrow{S}_{Observed} = \mathbf{M}_{Coude}\mathbf{M}_{Tower}\overrightarrow{S}_{Solar} \tag{1}$$

To recover the original Stokes vector, you simply multiply the observed vector by the inverse of the telescope and Coudé table matrices. Determining the matrices of these components is as simple as placing some polarizing object directly in front of the thing you're trying to calibrate.

4.1 The Telescope Matrix

As I said, it's as simple as putting a polarizer in front of the object you're trying to calibrate. So to calibrate the telescope, you must simply place a polarizer in front of the telescope, the entrance window of which is 40 meters off the ground and 76 cm wide, accessible only via the movable work platform by trained personnel. Also, the telescope matrix varies throughout the day, based on the position of the Sun in the sky (since the turret rotates to track it, and doing so changes the angles of the flat mirrors) and the rotation of the Coudé table.

We do have a window polarizer that sits on top of the entrance window, but using it for calibrations requires a minimal staffing level that the DST lacked from 2017–2024. Prior to 2017, calibration measurements were carried out sporadically, and reduced for use even more sporadically. The last good telescope matrix we have was performed in 2010. The last one that I know measurements exist for was carried out in 2016, but I have not seen that data. According to NSO scientists I've spoken to, the consensus about

the 2016 measurements were that there was no significant change relative to 2010. I hope to perform measurements at some point in 2025, now that we have the minimum required staff, but that has not been done at time of writing.

The 2010 measurements are in IDL .sav format, and can be opened in Python via `scipy.io.readsav`. The reduction code requires that you have a copy of the file, and point the code to it in the `SHARED` section of the config file. Contact me for a copy of the matrix.

4.2 The Coudé Table Matrix

Fortunately, this one is quite a bit easier to determine. The DST has a large polarizer and retarder mounted on rotating stages just above the exit window of the telescope. They can be staged into the beam, and rotated to modulate incoming light to some known Stokes vector. By performing a sequence of rotations, the Coudé table matrix can be fit from the known input and measured output Stokes vectors. This is a relatively quick process (minutes) and is typically performed at the end of an observing day. There are factors that can cause a bad polcal to be obtained (typically instrument faults or thin clouds during the calibration). Fortunately, polcals are typically stable on the order of days or weeks **provided there are no major optical setup changes**. A good polcal for data taken at 6302Å can be seen in Figure 6. The mean matrix for this polcal was:

$$\begin{bmatrix} 1 & 0.34 & -0.04 & 0.01 \\ 0.07 & 0.21 & -0.16 & -0.26 \\ 0.14 & 0.37 & 0.06 & 0.16 \\ 0.02 & 0.09 & 0.24 & -0.03 \end{bmatrix} \tag{2}$$

This is fairly typical for the complex optical train at the DST. The efficiencies for the individual Stokes components are: $Q_{eff} = 0.17$, $U_{eff} = 0.16$, $V_{eff} = 0.12$, $QUV_{eff} = 0.27$. Generally, the efficiencies should be between 0.1–0.866. The efficiencies here are quite low (and certainly not ideal). This is due to a combinations of the polcal method, which attempts additional retardance corrections and skews the efficiencies low, and the choice of beamsplitters in use at the DST[4].

If you start seeing efficiencies outside the good range, or if you get a warning about an unphysical Müller matrix, there's a good chance something has gone wrong. This can happen for a few reasons, but the most common has to do with the limits placed in the code. In order to avoid hot pixels or other defects caused by combining the two beams, the polcal section omits pixel values outside some fraction of the frame mean. If too many pixels are omitted, usually because some strong defect has skewed the mean, you may end up with spurious results. The default range is $[0.5, 1.5]$, and can be widened with the `polcalClipThreshold` keyword in the SpinorCal config file. You can also reduce the number of sections the images are split into with the `slitDivisions` keyword, or try turning on the hot pixel removal

[4] Beamsplitters are expensive, often have to be custom-made, and rarely perform up to manufacturer specs. This is because the larger market is more concerned with laser line splitting, and broadband performance is a secondary consideration.

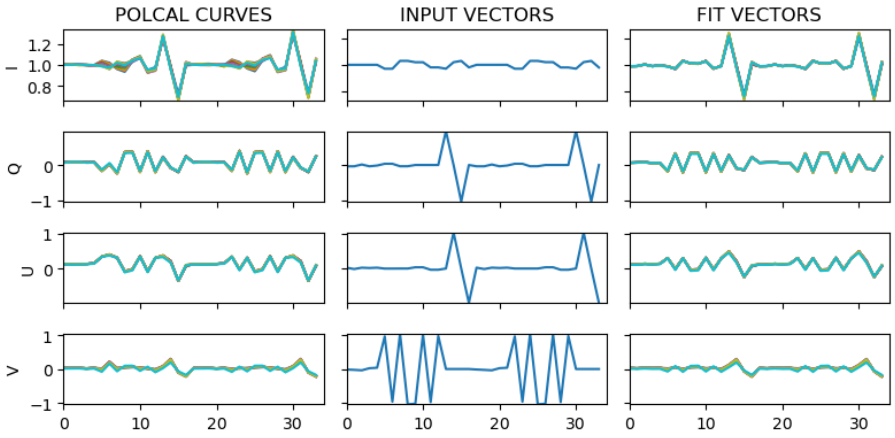

Figure 6: *Good Coudé table polarization calibration. Left column are the observed Stokes vectors for each overlapping section on the y-axis. Middle column shows what the Stokes vectors should after the calibration optics, but before going through the optical train. Right column shows the Stokes vectors resulting from the best-fit Müller matrix.*

function with `despike`. If all else fails, see if the problem persists across a few days. The polcal results are written to a FITS file, and if another day yields a better polcal, you can just drop that file into the reduction directory, and restart the code. It'll load the results from the file, and proceed with the calibration loop from there.

Reduced polcal matrices are saved to the calibration direction with the filename `CAMERA_POLCAL.fits`, where `CAMERA` is the facility camera used (Sarnoff1/2, FLIR1/2). Within the polcal file, extension 0 is empty, while extensions 1–6 contain the input Stokes vectors, observed Stokes vectors, Coudé table matrices, χ^2 values for the matrices, the (0,0) value of each matrix (since we normalize that parameter to 1), and the inverted Coudé table matrices, respectively.

5 Main Calibration Loop

With the gain tables created and the polarization matrix determined, it's time for the main calibration loop. This is by far the slowest part of the code, as well as the section most likely to go wrong. The general flow of this section is:

1. Load a single slit position 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.
5. Use the hairline fiducials to sub-pixel align the two beams along their y-axis.
6. Use the spectral line selected during gain table correction to sub-pixel align the two beams along their x-axis.

7. Combine the two beams.
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. Perform residual crosstalk corrections for $V \rightarrow Q$, $V \rightarrow U$, and $U \rightarrow V$ (optional).
12. Perform basic analysis on user-defined spectral regions of interest.
13. Create or update monitoring plots.
14. Move to the next slit position.

I would recommend not looking too closely at code for the main calibration loop. The number of moving parts and limited development time means it's a bit messier than I'd like. There are several locations where the code may prompt the user for intervention, usually when something has gone wrong. Before they can be aligned, the beams are corrected for dark and gain, then cut out, using the beam edges defined in the gain table creation step.

5.1 Beam Alignment

In order for the instrument to be useful, the two orthogonally-polarized beams have to be aligned with subpixel precision. The code to do this uses the same shift method used to create the gain table. This works as long as the alignment features are relatively narrow along the axis that you're aligning, and present at all/most positions along the other axis.

5.1.1 Vertical Alignment

Aligning the beams vertically uses a hairline fiducial (see horizontal lines in Figure 1). The profile of the hairline is narrow in Y, and present across the entire X-axis. The code will, by default, use the hairline positions found during gain table creation as a starting position, then attempt to find the hairline core in that region. Once it has found a good hairline, it will determine the subpixel shifts in the y-direction, and iteratively fit order-1, followed by order-2 polynomials to those shifts, to give a y-shift model at each x-position. This can fail under certain circumstances, most commonly if the science data were taken at the solar limb, or if the science data and the flat fields were taken several hours apart.

When the slit is partially off the solar limb, one of the hairlines will disappear for the very simple reason that there's no continuum emission off the limb. The code works just fine with a single hairline, but depending on which end of the slit is off-limb, it may require the user to select the hairline used in alignment via a popup widget. If the science data and the flat fields are several hours apart, the hairlines may have moved relative to where they were during the flat field. This is down to the DST beam wobble. Again. Usually, the code is pretty good at figuring out the new position, but sometimes it can fail if the hairline was particularly close to the chip edge. If you see a double image in the hairlines on the live plot,

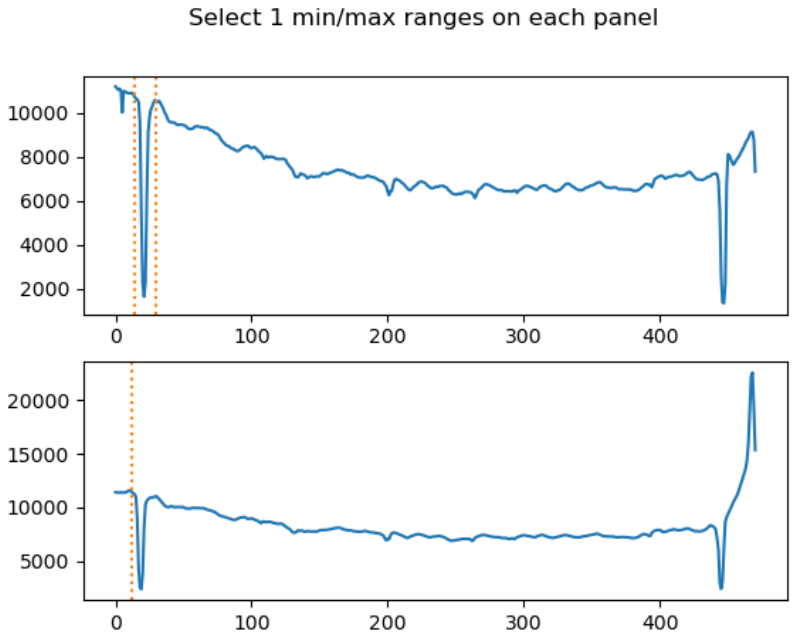

Figure 7: *Widget used to manually select hairline ranges. Each panel is the intensity profile, averaged along the x-axis for the lower and upper beams respectively. Hairlines show up as narrow dips in the profile. There are two in this cross-section, one around pixel 20, and another around pixel 450. There may only be a single hairline visible. The user is prompted to select two positions on each panel corresponding to the minimum and maximum of the range around the hairline. This example shows three of the four selections already made. At this point, the user would select the right-hand side of the hairline in the lower panel. This prompts the plot to close, and save these positions.*

that's a pretty good indicator that the code made its best guess, and it wasn't very good. If the code fails, or if you set the `hairSelect=True` switch in the config file, the code will produce the popup plot shown in Figure 7. Simply click twice in each panel to select the lower and upper range of a hairline, and the code will proceed with these values.

5.1.2 Horizontal Alignment

Aligning the beams horizontally uses a spectral line, which can be tricky. Ideally, this would use a telluric line, as these are constant, thin, and show no (or no significant) line-of-sight velocities. This is not always possible, and frequently a solar spectral line must be used. Using a spectral line can cause issues with the 8542Å window, particularly in flaring regions. The same method is used for both the horizontal and vertical alignment, which can cause problems if there are significant flows across the SPINOR field-of-view, or if the spectral line goes into emission (as 8542 is prone to). Unlike the hairline alignment, the code cannot detect a failure easily. The only way to turn on manual selection is to notice a misalignment or severe deformations in the shape of the spectrum and turn the manual selection on with `alignSelect=True` in the config file. The widget produced will behave the same as the hairline widget. See Figure 8 for an example.

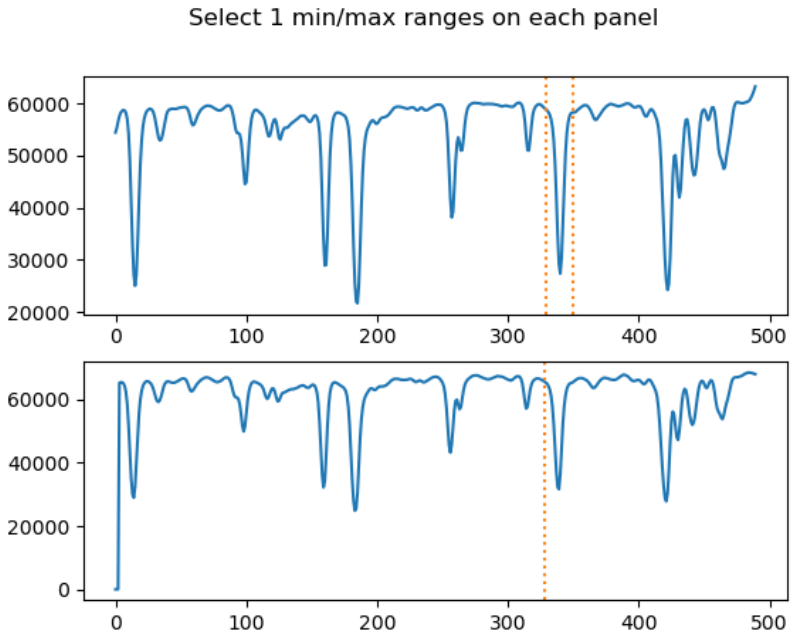

Figure 8: *Manual selection of spectral line for alignment. Shown is the average spectrum in the upper and lower beams. The user can select the minimum and maximum range around a spectral line. Ideally, this should be a prominent, narrow line. In this example, the line selected is the Fe I 5253.5Å line. There are no telluric lines available in this window, so this line is chosen as there are no nearby lines that may throw off the algorithm.*

5.2 $I \rightarrow QUV$ 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:

$$\alpha = \frac{\langle QUV(\lambda_{cont})\rangle}{\langle I(\lambda_{cont})\rangle} \tag{3}$$

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

$$QUV(\lambda)_{corr} = QUV(\lambda)_{uncorr} - \alpha I(\lambda) \tag{4}$$

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:

$$QUV(\lambda)_{corr} = QUV(\lambda) - (m\lambda I(\lambda) + bI(\lambda)) \tag{5}$$

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.

5.3 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:

$$QU(x,y)_{corr} = QU(x,y)_{uncorr} - \alpha V(x,y) \tag{6}$$

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:

$$cos(\theta) = \frac{\mathbf{V}(\lambda)\cdot\mathbf{QU}(\lambda)_{\mathbf{corr}}}{||\mathbf{V}(\lambda)||\ ||\mathbf{QU}(\lambda)_{\mathbf{corr}}||} \tag{7}$$

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

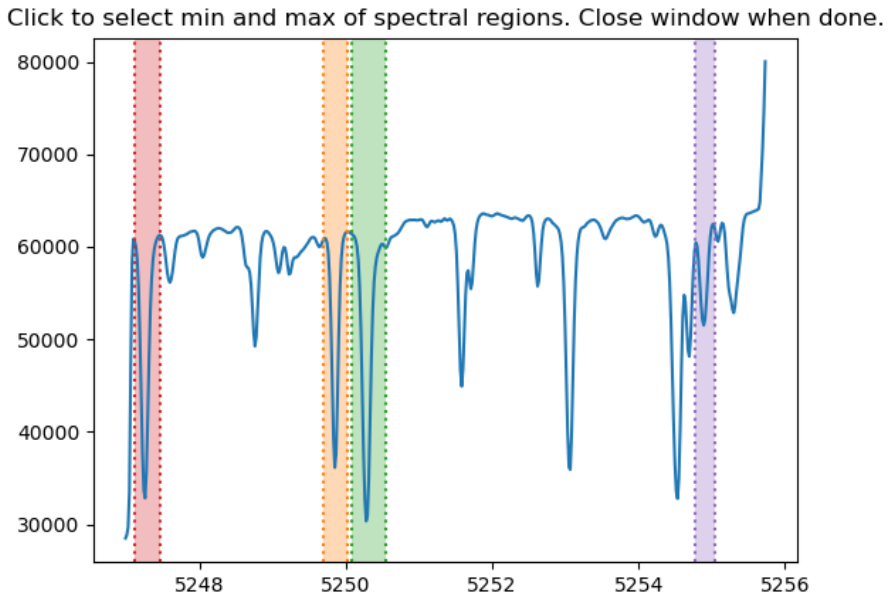

Figure 9: *Line selection widget. The user can make an arbitrary number of selections. The best guess at a wavelength scale (in Angstrom) is provided as the x-axis for reference. If you do not have a large manual of the solar spectrum available at your desk,* `https://bass2000.obspm.fr/solar_spect.php` *is quite handy. If you set the range similar to what's shown on the plot, and click on line cores, it will tell you what each line is, if you do not know. The above regions highlight (left-to-right) Cr I, Fe I, Fe I, and Mn I.*

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, but I've recently found it to be mostly unnecessary with chromospheric 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`.

5.4 Analysis and Plotting

For the purposes of monitoring the quality of corrections and performing some low-level analysis of any lines of interest, the code will prompt you to select regions of interest once it has performed a correction for the 0th slit position of the 0th map observed in sequence. It does this with the widget shown in Figure 9. This window will stay open and block the code until you're done selecting regions of interest. If you don't need analysis, you can simply close this window, or select an arbitrary number of regions. Each region you select will result in another live plot being spawned at the next step, assuming you have plotting turned on, so be warned.

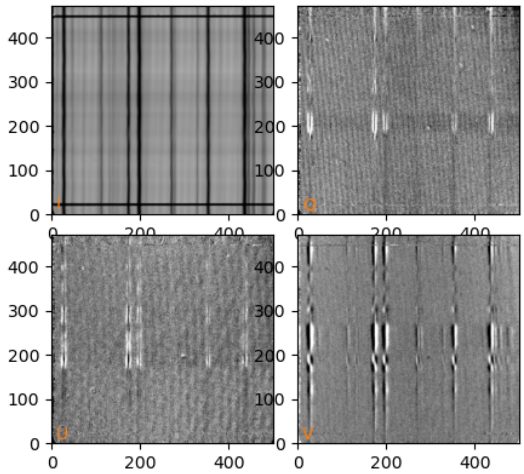

Figure 10: *Reduced Stokes-IQUV images for a single slit position. This is after the full reduction sequence, including full $V \rightarrow QU$ crosstalk.*

Each line selected will be passed through an analysis routine after the main calibration loop is completed. This analysis carries out an estimation of velocity and line-width (from moment analysis), and some rough parameters of polarization degree.

During the main calibration loop, several live plots can be enabled for monitoring of corrections. This is a useful feature for fine-tuning the method used for crosstalk correction, monitoring if there's any issues, etc. If plotting is enabled, the following plots are created:

- The reduced per-slit IQUV images. These are the images in λ/Y space. Figure 10 shows an example of this plot.
- If the user selects any lines for analysis, each line selected will create and fill in a live plot of the (X/Y) field map in Stokes-IQUV. Since the X-axis is being filled in during the calibration loop, these plots will fill left-to-right as the calibration continues. Stokes-I is shown in the line core intensity, which may be structurally different than continuum intensity. The other panels show the integrated polarization degree. Since Stokes-Q/U are symmetric around the line core, while Stokes-V is not, the integrated quantity shown differs between Q/U and V. Q/U shows the net degree of polarization when integrated, while V shows the excess positive/negative polarization. This could lead the the user to (mistakenly) draw the conclusion that V is weaker or less structured. Figure 11 shows an example of this plot, partially filled in.
- If any internal crosstalks are being corrected, the final plot will display the crosstalk value. If the per-slit images look suspect, and the crosstalk values are large, it's a good sign to the user that the method may need to be adjusted. See Figure 12

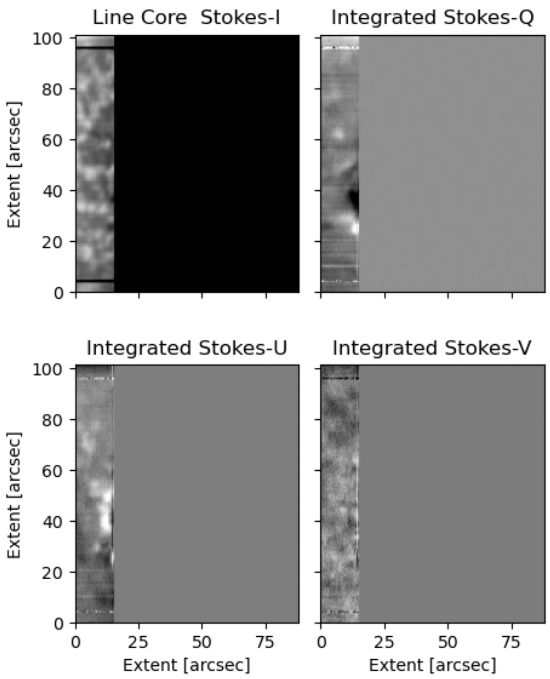

Figure 11: *Field maps in the selected Fe I 5250Å line. These are filled in as the calibration loop continues.*

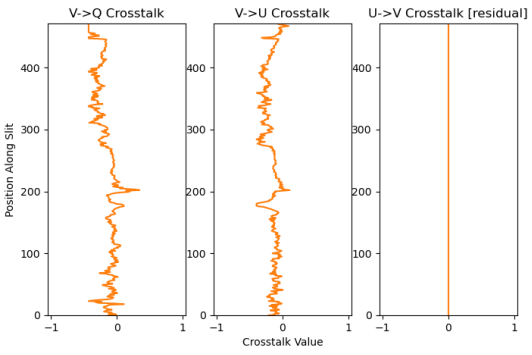

Figure 12: *Degree of internal crosstalk correction, as applied to the images in Figure 10*

6 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.

6.1 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, SPINOR 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[5].

The grid rasters are used to update the plate scale of the SPINOR cameras 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.

6.2 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 SPINOR 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.

[5] 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 is on the order of $10^{-2} - 10^{-3}$.

If the `verifyPointingUpdate` keyword is set to True in the config file, this will spawn a plot overlaying contours of HMI on the SPINOR 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.

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.

7 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.

```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                    8 / array data type
NAXIS   =                    0 / number of array dimensions
EXTEND  =                    T
DATE    = '2025-06-09T19:00:10' / File Creation Date and Time (UTC)
ORIGIN  = 'NMSU/SSOC'
TELESCOP= 'DST     '           / Dunn Solar Telescope, Sacramento Peak NM
INSTRUME= 'SPINOR  '           / SPectropolarimetry of INfrared and Optical Regi
AUTHOR  = 'sellers '
CAMERA  = 'Sarnoff #1, 1024x512, 12-bit'
DATA_LEV=                  1.5
        ======== DATA SUMMARY ========
STARTOBS= '2025-05-28T14:50:07.659'
ENDOBS  = '2025-05-28T14:50:08.059'
BTYPE   = 'Intensity'
BUNIT   = 'Corrected DN'
EXPTIME =                 50.0 / ms for single exposure
XPOSUR  =                  400 / ms for total coadded exposure
NSUMEXP =                    8 / Summed images per modulation state
SLIT_WID=                 40.5 / [um] HSG Slit Width
SLIT_ARC=                  0.3 / [arcsec, approx] HSG Slit Width
MAP_EXP =              87.607 / [arcsec] Requested Map Size
MAP_ACT =              137.71 / [arcsec] Actual Map Size
        ======== SPECTROGRAPH CONFIGURATION ========
WAVEUNIT=                  -10 / 10^(WAVEUNIT), Angstrom
WAVEREF = 'FTS     '           / Kurucz 1984 Atlas Used in Wavelength Determinat
WAVEMIN =            5247.065 / [AA] Angstrom
WAVEMAX =             5256.15 / [AA], Angstrom
GRPERMM =              308.57 / [mm^-1] Lines per mm of Grating
GRBLAZE =                52.0 / [degrees] Blaze Angle of Grating
GRANGLE =                56.0 / [degreed] Operating Angle of Grating
SPORDER =                  10 / Spectral Order
SPEFF   =               0.677 / Approx. Total Efficiency of Grating
LITTROW =               3.725 / [degrees] Littrow Angle
RESOLVPW=           166120.0 / Maximum Resolving Power of Spectrograph
HAIRLIN0=              21.985 / Center of registration hairline
HAIRLIN1=              446.985 / Center of registration hairline
        ======== POINTING INFORMATION ========
RSUN_ARC=           947.02125
XCEN    =              -678.94 / [arcsec], Solar-X of Map Center
YCEN    =               147.14 / [arcsec], Solar-Y of Map Center
FOVX    =               137.71 / [arcsec], Field-of-view of raster-x
FOVY    =              101.276 / [arcsec], Field-of-view of raster-y
ROT     =              180.023 / [degrees] Rotation from Solar-North
        ======== CALIBRATION PROCEDURE OUTLINE ========
PRSTEP1 = 'DARK-SUBTRACTION'   / spinorCal/SSOSoft
PRSTEP2 = 'FLATFIELDING'       / spinorCal/SSOSoft
PRSTEP3 = 'WAVELENGTH-CALIBRATION' / FTS Atlas
PRSTEP4 = 'TELESCOPE-MULLER'   / 2010 Measurements
PRSTEP5 = 'SPECTROGRAPH-MULLER' / spinorCal/SSOSoft
PRSTEP6 = 'I->QUV CROSSTALK'   / spinorCal/SSOSoft
PRSTEP7 = 'V->Q CROSSTALK'     / spinorCal/SSOSoft
COMMENT Full WCS Information Contained in Individual Data HDUs
```

Figure 13: *Level-1 File, extension 0 header with human-readable observational metadata.*

- `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:

1. SPINOR 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 SPINOR data on weaker machines.
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 13 for an example of the extension-0 header, and Figure 14 for an example of the extensions 1–4 headers.

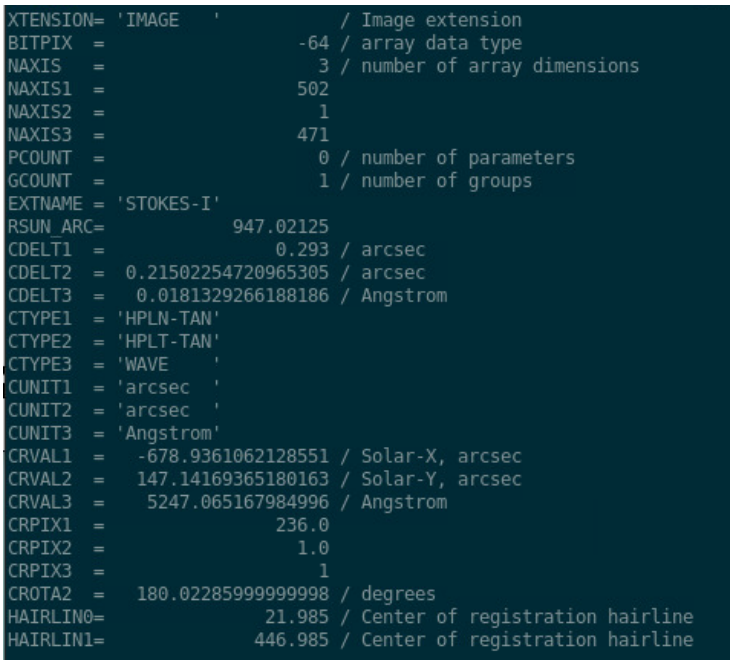

Figure 14: *Level-1 File, extension 1 header with FITS-compliant observational metadata.*

8 Allowed Config File Keywords (and what they do)

8.1 SHARED

- tMatrixFile [required]: path, location and filename of telescope matrix file. Currently, only the 2010 measurements are supported, unless I can fix the 2025-08 measurements.
- contextMovieDirectory [optional]: path, location to place context movies. Required in the channel attribute `contextMovie` is set to True.
- gratingRules [optional]: float, lines per mm of grating used. If not provided, defaults to 308.57. Since 2025-06, it should be set to 110.
- blazeAngle [optional]: float, blaze angle of the grating used. If not provided, defaults to 52. Since 2025-06, it should be set to 64.
- qModulationPattern [optional]: comma-separated ints, linear combination used to reconstruct Q. If the modulator is rotated in its mount, or the home position is reset, this will have to be changed. Defaults to -1,-1,1,1,-1,-1,1,1
- uModulationPattern [optional]: comma-separated ints, linear combination used to reconstruct U. If the modulator is rotated in its mount, or the home position is reset, this will have to be changed. Defaults to 1,-1,-1,1,1,-1,-1,1
- vModulationPattern [optional]: comma-separated ints, linear combination used to reconstruct V. If the modulator is rotated in its mount, or the home position is reset, this will have to be changed. Defaults to 1,-1,-1,-1,-1,1,1,1

- basePlateScale [optional]: float, arcsec per mm of the prime focal plane. Should only be set if some very strange relay optics are placed before SPINOR. Defaults to 3.76
- telescopeCollimator [optional]: float, focal length in mm of the post-focal collimating optic. Should only be set if SPINOR is moved to a different DST port, or some very strange relay optics are used. Defaults to 1559.
- slitCameraLens [optional]: float, focal length in mm of the lens that projects the field image onto the slit unit. Should only be used if the SPINOR feed optics are changed. I would not recommend doing that, as the current lens is a very achromatic triplet design, and more of our facility lenses are not achromats. Defaults to 780 mm.
- spectrographCollimator [optional]: float, focal length in mm of the spectrograph collimator. This can be changed, but it's a laborious process. Again, not recommended. Defaults to 3040 mm.
- telescopeLatitude [optional]: float, latitude of telescope site for parralactic correction of Q/U. I don't know why you'd ever change it, but it defaults to 32.786 degrees.

8.2 CAMERAS

The camera headers should have the camera names. For SPINOR, this is SARNOFF1, SARNOFF2, FLIR1, or FLIR2. No other cameras (in recent memory) have been added the the telescope CCC systems for use with SPINOR. These camera names are important for telling the code the pixel size. The Sarnoffs have 16 μm pixels, while the Flirs have 25 μm pixels. The reduction code also names a few of the ancillary calibration results with the camera name.

Each camera config section must contain the following required keywords:

- rawFileDirectory [required]: path, location of Level-0 SPINOR camera files.
- reducedFileDirectory [required]: path, location to place reduced files for this camera.
- reducedFilePattern [required]: 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}_spinor_6302_level1_{:03d}_steps.fits`.
- reducedParameterMapPattern [required]: Python-format-able string with 4 format spaces. The first two, again, are `{:s}` tags to insert date and time. The latter two should be float-formatting tags, e.g., `{:0.2f}` to insert the wavelength range used for parameter maps.
- centralWavelength [required]: int, approximate central wavelength. This is used to start the wavelength calibration, and is used in naming some of the ancillary files, as well as setting FITS header keywords.
- spectralOrder [required]: int, the order of the spectrum observed. This differs by line and grating. There are functions to calculate the most likely spectral order for a given line/grating combination

in the `ssosoft.spectral.spectraTools` module, but I never got around to integrating it. It will always be the same for a given line/grating combo, so it's easy enough to pull it from previous config files.

The following keywords are optional, but may be required for quality reductions in come cases:

- internalCrosstalk [optional]: str or bool, default False. Controls the newest version of the crosstalk correction. If set to "full", performs profile-by-profile correction. True does a single correction for the frame. False turns it off entirely. I keep it True or full for photospheric data (Fe I 6302, Fe I 5250, K I 7699), and False for chromospheric data.
- residualCrosstalk [optional]: bool, by default False. If set to True, does a second pass for $I \rightarrow QUV$ crosstalk. Use if you're get strong positive signals in Q/U that look like the line profile inverted. I always keep it on. It tends not to mess too much up, but if you're seeing line profiles residuals and it is on, it may be overcorrecting.
- despike [optional]: bool, default False. If set to True, does a median-filtering despike for hot pixels and other noise. FLIR needs this on, and the Sarnoffs may benefit as well. They're all old cameras. The algorithm isn't a super-robust one, but it does okay.
- verbose [optional]: bool, default False. If set to True, the code talks to you quite a bit more. I usually keep it on so I can track where I'm at in reductions.
- plot [optional]: bool, default False. If set to True, will spawn additional plots, which include the gain tables, polcal curves, and the live plotting of the reduced data while the main cal loop runs. If set to False, the code will still spawn any required interactive plots (since it needs those user inputs). I like to keep it on, particularly to watch the main cal loop, so I can keep an eye on the crosstalks.
- savePlot [optional]: bool, default False. If set to True, saves some of the plots its spawns when `plot` is set to True. Again, I usually keep it on.
- contextMovie [optional]: bool, default False. If set to True, writes a movie for each reduced file which shows the raster images and the individual slit images. I keep this on as well.
- contextMovieDirectory [optional]: path, default same as reducedFileDirectory. If set, in conjunction with `contextMovie` this will place the final context movie in the indicated directory. I usually set it to the daily `level1/context_movies` directory, along with the ROSA/HARDcam movies.
- solarAlign [optiona]: bool, default False. If set to True, attempts a coordinate correction to HMI. If you're doing fast rasters or quiet-sun observations, you'll want to set it to False.
- verifyPointingUpdate [optional]: bool, default False. If set to True in conjunction with `solarAlign`, it spawns a plot so you can check the alignment quality. I recommend keeping it on. Note that `plot` can be set to False, and this window will still spawn.

- hairSelect [optional]: bool, default False. If True, you can manually select the alignment hairline during the main cal loop. If the raster is near-limb, partially off-limb, or misaligned vertically on the chip, you may need to set this.
- alignSelect [optional]: bool, default False. If True, you can manually select the spectral line used for beam alignment. By default, the code uses the first line selected in the wavelength calibration/gain table creation. If this line happens to be a chromospheric line, the code may not align correctly, particularly if you're in a flaring region during the main scan.
- totalHairlines [optional]: int, default 4. If the top or bottom hairline falls very close to the edge of the chip, you may need to set this to 2, so the code looks for the middle pair of hairlines.
- hairlineWidth [optional]: int, default 3. Width (in pixels) of a hairline. You shouldn't have to change this, but if your raster is poorly-focused, you may need to increase it for the masking during gain table creation.
- polcalProcessing [optional]: bool, default True. If set to True, corrects the polcal images for lamp and solar gain before constructing Stokes vectors. Turn off if you're processing a polcal from a different day from the raw Level-0 file. In general, you should avoid doing that. If you don't have a polcal from that day, it's usually better to grab the reduced polcal from an adjacent day than to process it from raw data.
- calRetardance [optional]: float, default 90. If set, uses the provided value for the calibrator retardance. It should be around 90, but it's a bit less in the IR, and a bit more in the green/blue. Keeping it at 90 will make your polcals a bit less accurate, but the retardance isn't super well-known.
- slitDivisions [optional]: int, by default 10. During polcal creation, the image is split into this number of slices along the vertical axis. If your polcal vectors are weird, or you're getting a lot of invalid Müller matrices, you may want to decrease this number.
- polcalClipThreshold [optional]: comma-separated floats, default 0.5, 1.5. Allows the user to override the fractional clipping that selects the regions in each nslice for continuum-ey points. If your polcals are coming up mostly NaN, or you're getting warnings about all-NaN slices being encountered, you may try widening this range. This comes up a decent amount when the Liberty reflector is used to send Hα to HARDcam while SPINOR is observing 6302. For some reason, that beamsplitter *ruins* the linear polarization signal.
- cameraLens [optional]: float, default 1700. Focal length of the final lens in the SPINOR optical train before the camera. We almost always use 1700 mm lenses, but if that changes, you can update this value per camera.
- v2q, v2u, u2v [optional, deprecated]: bool/str, default False. Switches for the older internal crosstalk corrections. The internalCrosstalk keyword overrides these, but if that is set to False, these can be turned on with either "True" or "full".

- solarFlatFile, lampFlatFile, polcalFile, scienceFile [optional, deprecated]: str, default none. The pipeline automatically grabs what it thinks of as the best versions of these files. These were meant to override that behaviour, but I'm 98% sure it's broken at the moment. If you don't want to use certain files, I'd recommend renaming them in the Level-0 directory