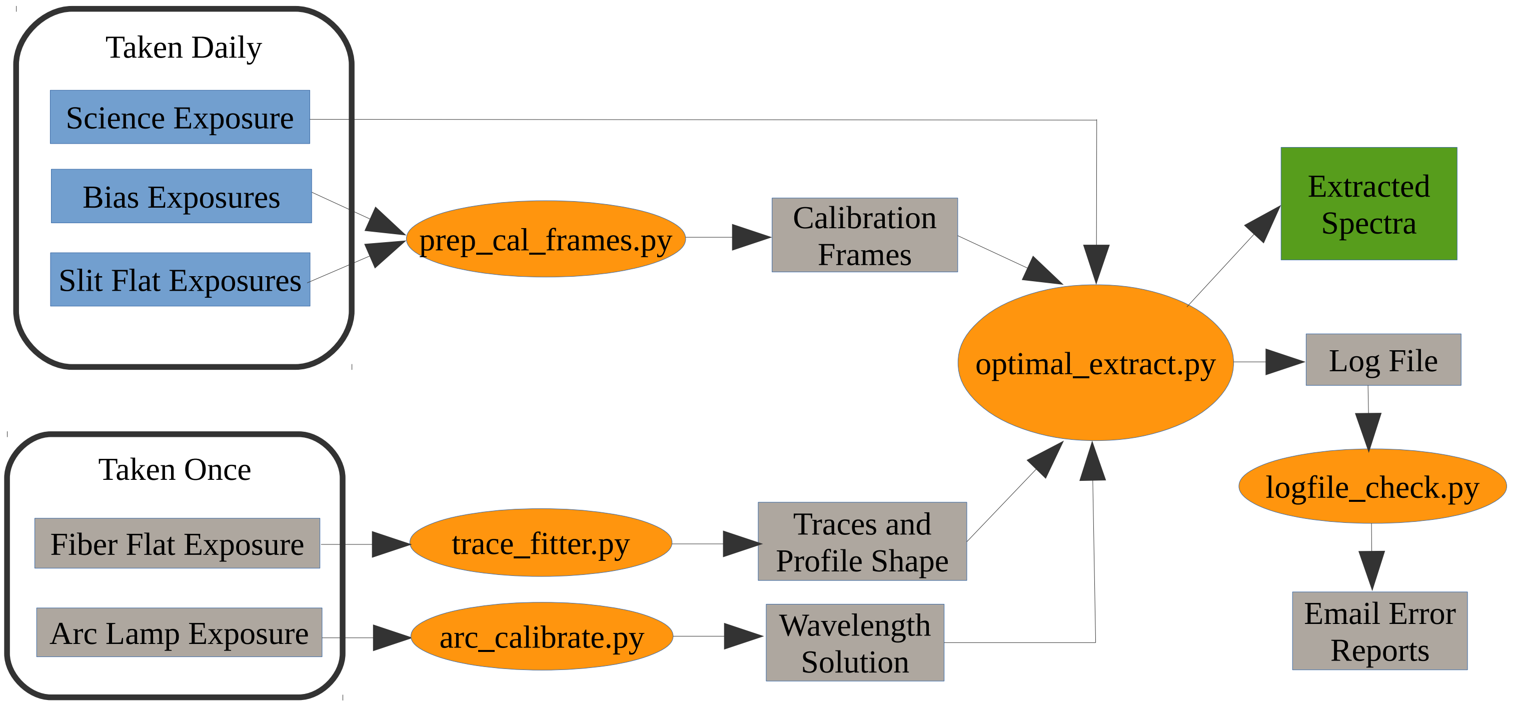
MINERVA Raw Reduction/Spectral Extraction Documentation (Optimal Extraction Version)

Purpose: Convert raw CCD frames to extracted spectra with wavelength solution. Includes calibration (bias subtraction, slit flat normalization, and scattered light subtraction)

Schematic:



Inputs:

Calibration Data

- Arc frames (during setup - two per telescope, one with, one without iodine)
- Fiber flats (during setup - several per telescope)
- Bias frames (daily)
- Dark frames (daily – not currently used)
- Slit flats (daily)

Science Data (All the same from point of view of code)

- Daytime Sky
- B-Stars
- Target Stars

Code:

Run rarely, only if new fiber flats or arc frames are taken

- trace_fitter.py – Finds a trace solution and profile shape from fiber flats
- arc_calibrate.py – Finds the wavelength solution from the arc frames

Core functions for optimal extraction

- special.py – general functions (Gaussian profile, χ^2 fitting, etc.)
- minerva_utils.py – MINERVA specific functions, includes trace fitting and optimal extraction code

- prep_cal_frames.py – Stacks bias and slit flat frames for use in extraction
- optimal_extract.py – Code to run optimal extraction. Calls heavily on functions in minerva_utils
- logfile_check.py – Examines log files generated by optimal_extract.py and sends error reports (via email) on low exposures or unprocessed data

Spectro-Perfectionism code (not in production)

- SP_extract.py – Code for testing 2D PSF and extraction
- bsplines.py – Generates functions related to splines in support of 2D PSF
- psf_utils.py – Special functions for fitting 2D PSF
- fit_2D_psf.py – Code to find Gauss-Hermite 2D PSFs. Output of this is used in SP_extract.py

Outputs:

Extracted Spectrum - .fits file

Four HDU extensions

1. Photon Counts (proxy for flux)
2. Wavelength
3. Inverse Variance
4. Pixel Mask

Each extension contains a 3D array:

1. Axis 1 = pixel position (along trace)
2. Axis 2 = order
3. Axis 3 = telescope

A few notes. These are large files, ~7.3MB. Mask could be merged with inverse variance to save space. Also, wavelength is the solution from the arc frames. This is *identical* in all extracted spectra since (right now) I only have one arc frame. In the future, this may not be true, but space could be saved by making a wavelength solution fits and adding a header in each extracted spectrum to show which wavelength file should be used.

Installation

Code is on github under MinervaCollaboration/minerva-pipeline/spectral_extraction

Clone this onto the desired machine under $$$/minerva-pipeline/spectral_extraction (this will also clone $$$/minerva-pipeline/minerva_dopcode – the radial velocity pipeline)

To run, the following environmental variables must be set:
MINERVA_DIR – points toward $$$/minerva-pipeline/spectral_extraction/python
MINERVA_DATA_DIR – points toward wherever the data is stored (right now must have a local copy on your machine). Under this directory, data should be organized as:

nYYYYMMDD/nYYYYMMDD.exp_ID.exp_num[####].fits

Example:

n20160216/n20160216.HR2209.0017.fits
n20160216/n20160216.slitFlat.0009.fits

MINERVA_REDUX_DIR – points toward wherever you would like the reduced (extracted) data to be stored. Code will automatically save it with the same name, appended with a .proc.

Example:

n20160216/n20160216.HR2209.0017.proc.fits

MINERVA_SIM_DIR – points toward wherever you choose to store support files needed for extraction. These include arc lamplines, order estimates, iodine FTS spectrum, etc.

I like to set a MINERVA_ROOT, then put the other directories underneath:
MINERVA_ROOT/
 -data
 -sim
 -redux
 -software/master/minerva_pipeline/…

Load necessary sim files (I think I can just move these into github so they come automatically with the clone)

- table1.dat (HARPS vacuum reference for ThAr lines, with only wavelength range relevant to MINERVA)
- arc_order_estimates.csv (file I made manually with five estimated pixel position and corresponding wavelengths for the orders that show up on the MINERVA ccds. These get the automatic fitter close enough to lock on to the correct lines)
- [Optional] Iodine.fits (fits file of iodine FTS spectrum. I'm not using this right now (but I think the dopcode is))

I also like to set a MINERVA_TAG_DIR, which can be used for tagged versions (e.g. v1.0.0). In this way you can make updates, but save earlier versions. Then you can roll back if something breaks in a new version. This would be something like $$$/v1.0.0/minerva-pipeline/spectral_extraction/python

Operation:

Generate wavelength solution, traces, and profiles (to be used in all subsequent extractions). **These only need to be run the first time you load the code on a new system, or if new arc frames or fiber flats are collected.** Run the following commands (If you like, you can change these to executables).

1. python $MINERVA_TAG_DIR/arc_calibrate.py
2. python $MINERVA_TAG_DIR/trace_fitter.py

These will make the files:

- $MINERVA_REDUX_DIR/{date of arc exposure}/wavelength_soln_T[1-4].fits
- $MINERVA_REDUX_DIR/flat_trace/trace_{profile shape}.fits

Set up crontab (or similar) to run daily:

1. Download data from minerva_main (unless already on minerva_main) to MINERVA_DATA_DIR. If not on minerva_main, set up a shell script for a daily rsync (example is included in docs folder)
2. python $MINERVA_TAG_DIR/prep_cal_frames.py
3. python $MINERVA_TAG_DIR/optimal_extract.py -f {science exposure, including full path}
4. python $MINERVA_TAG_DIR/logfile_check.py
5. Upload reduced data from $MINERVA_REDUX_DIR to minerva_main (again, unless already

on minerva_main)

Step 3 (the extraction) is the most time consuming, order 5 mins per frame. Depending on your system and preferences, these can be run in a batch, parallel, or similar to increase speed. You may need to create a shell script wrapper to run all exposures for the day.

Right now I use the follwing shell scripts (stored in the docs folder):

- load_minerva_data.sh – rsync data from minerva_main
- extract – search for files to extract and generate list of optimal_extract.py calls
- launch_minerva_batch – launch extraction in parallel using slurm
- send_proc_data.sh – rsync reduced data to minerva_main

The scripts and needs will vary depending upon the system. When on minerva_main (default), the first and last are unnecessary.

Details

(Note, all orientations described here are with reference to ccd frames with traces running horizontally, left to right in increasing wavelength. Bluest orders are on top, reddest on bottom)

Trace fitting

I use the high SNR fiber flats to find the traces. I start with a cross section of column 1 and find all of the peaks. I then discard a couple traces near the edges because they do not stay on the ccd for their entire length (and are outside of the range of the Iodine spectrum, so they wouldn't be used in the RV pipeline anyway). I then have a list of fiber numbers and their corresponding peak pixel location.

From there, I step by 20 pixels (step size can be adjusted) to the right, find the next peaks, and continue until the entire ccd is covered. At each step, I fit a modified gaussian profile (described in the extraction section) to the region around the peak. From this I get a more precise center as well as the height, width, and power parameters. From these positions, I run a 12th order polynomial fit (within each trace/fiber) for the peak vertical ('y') position as a function of the horizontal ('x') pixel position. The fit is repeated for the height, width, and power coefficients (essentially saving not only the trace position, but also the trace profile as a function of pixel).

The fiber flats appear insensitive to cosmic rays and other disruptions so this procedure tends to be robust. The coefficients are saved into $MINERVA_REDUX_DIR/flat_trace/trace_gaussian.fits

Optimal Extraction

Here I follow the optimal extraction algorithm detailed in Horne, 1986. For the model, I use a modified Gaussian profile as a function of pixel x:

$$I(x)=h\cdot\exp\left(\frac{|x-xc|^{p}}{2\sigma}\right) \quad (1)$$

Where the parameters are:

- h = height (relative intensity, function of sigma, plus photon counts)
- xc = center pixel position
- σ = width parameter of Gaussian
- p = power parameter (2 for true Gaussian)

I have tried many other profiles including Moffat, Bsplines, and the modified Gaussian with Lorentzian wings. I get the best performance with the profile in Equation 1, but the residuals are still higher than I would like. If you can find a better profile fitting method, that would be great, but the current profile performs well.

Before starting an extraction, I re-fit the traces (since the trace positions can drift relative to the fiber flat). I start with the estimates from the trace fitting, then repeat the procedure, just with science data. I fit this with a spline for more precise centering (although polynomials may also be used). During the trace refinement, the other profile shape parameters (sigma, power) are held constant.

In the extraction loop, I run the extraction once on each pixel column. To do this, I generate a profile matrix with the shape found in trace fitting centered around the refined trace center. This creates a relatively sparse matrix with a normalized profile in each column. I also include a twelve term polynomial for the background level across the column. This makes a 2052x(116+12) matrix **P**. I take the data from that column as matrix **d.** Finally, I generate the inverse variance at each point, which I assume to be independent of adjacent pixels. Inverse variance is given by (including slit_flat renormalization):

$$invar(x) = \frac{1}{\frac{|data(x)|}{slit\,flat(x)} + \left(\frac{readnoise}{slit\,flat(x)}\right)^2} \quad (2)$$

I then fit (linearly, in a least-squares sense) to find the coefficients **c** that solve the matrix equation.

$$Pc = d + N \quad (3)$$

The noise matrix, **N**, is a diagonal matrix with 1/sqrt(invar) on the diagonel. This algorithm accounts for fiber-to-fiber cross talk, which is small, but present in MINERVA exposures. With the fitted values, I reconstruct a model and find the relative height (photon counts), and re-estimate the inverse variance. The height, the extracted counts, is given by the first 116 elements of **c**. The last 12, the background terms, are not saved.

Then I go through a loop to reject cosmic rays. This check the residuals between data and model at each column and rejects any that are too large (and positive). There is some tuning required to set a sensible threshold. I found 20 counts to give a good result. Overzealous rejection will tend to flag good points and causes more problems than cautious rejection. Note that if the model residuals can be decreased for a good profile, cosmic ray rejection will be more effective. Once the cosmic ray rejection is finished, final amplitude and inverse variance values are calculated.

This is repeated for every horizontal ('x') pixel along each of the 112 “good” traces (28 good fibers x 4 telescopes).

Final values are saved into an array as detailed above in **Outputs.**

Spectro-Perfectionism

I will be brief on detail for now as Spectro-Perfectionism (hereafter, SP) is not used in production. This should give an orientation and, if I carve out the time, more details will be included in a forthcoming paper.

SP extraction solves the same matrix equation as (3), but the profile is two-dimensional, instead of one-dimensional. This, in principle, more accurately accounts for the true CCD PSF. It comes with several complications, as you might imagine. There are two main steps in SP extraction.

1. Determine the PSF
2. Extract the spectrum

Note that step 1 is analogous to finding the cross-dispersion profile from fiber flats for optimal extraction.

Determine the PSF

This is the most challenging aspect of SP extraction. The PSF must be found empirically (unless you are really great calculating it from first principles). This can be done several ways, but the easiest is to fit a model to unresolved emission lines across a trace. For MINERVA, this type of frame comes only from the ThAr arc frames. It should be possible to find the profile from Iodine absorption lines, but I have not yet attempted this approach.

ThAr lines are unresolved, but problematic because their amplitudes vary wildly (affecting SNR) and because many lines overlap. In fit_2D_psf.py I call functions that search the ThAr frames for lines that are:

1. Above a minimum signal threshold
2. More than 12 pixels from the nearest known centroid (note, this does not catch every line overlap)

To these I fit a 2D modified Gaussian profile multiplied by Hermite polynomials with order <=4 (see Pandey thesis, University of Utah for a thorough description. I only modify this by allowing the power to vary from 2, as with the 1D cross-dispersion profile). The parameters and coefficients are constrained to vary slowly, as a second order polynomial, across the order.

The results are saved in a separate file for each trace (116 total, 112 useable).

The fitting is reasonably good, but not precise and this is the largest source of error in the extracted spectrum.

Extract the Spectrum

Building the matrix **P** is more complex than the optimal case, but otherwise, it should proceed the same in principle. The problem is that **P** is too big in most cases (including MINERVAs) to *store*, let alone to use in calculations.

This is overcome by imposing the (quite reasonable) assumption that the PSF is local. That is, incident light at a particular wavelength lands almost entirely within a small radius, about 10 pixels, of a centroid. This means that the CCD can be divided into a series of smaller boxes. Within these, the traces are extracted quite quickly. The edges are unreliable, but more than 10 pixels from an edge the results are valid. Hundred of these overlapping boxes are extracted, then stitched back into an extracted spectrum.

The entire process takes ~30 minutes on my PC with no parallelization. Slower than optimal, but still a tractable time for a supercomputing system.

The other point of note is that the “raw” SP extracted spectrum exhibits extreme ringing due to covariance in neighboring points. This is smoothed with a “reconvolution matrix” **R** (see Bolton & Schlegel 2010 for details). For comparison to stellar templates in later code, the same reconvolution matrix must be used. Thus **R** is saved (there is one **R** per box) for later use in the pipeline. Right now, this stage is in testing (on a low priority…). Applying **R** is analogous to applying an “instrument profile” post optimal-extraction. The benefit of SP over optimal is that **R** is calculated during extraction.