Specpol Flow:
NormPlot2.4 User Guide and
Tutorial for Echelle Spectra Normalization

(Shaquann Seadrow, Colin Folsom)

2023

NormPlot2.4

This current version is written in Python3. This can be utilized as a data preparation step in the Specpol Flow pipeline.

User Guide

Prior to using the normalization GUI, the file containing the Stokes data should be converted from fits to .s format. You can use the normalization code from the terminal with the command

`python normPlot2.py <observation_file.s>`.

You should see the following window.

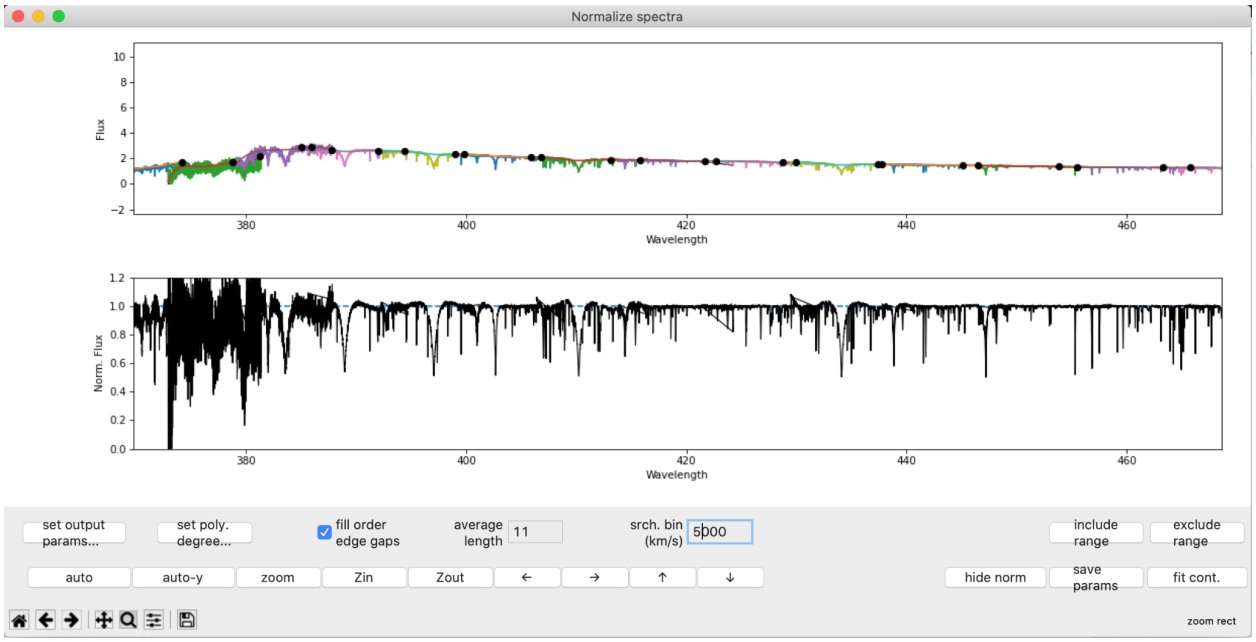


Figure 1)

The top panel of this window displays the spectral orders and polynomial fits for the black scatters points on the spectrum of each order. The black fitting points are computed by dividing

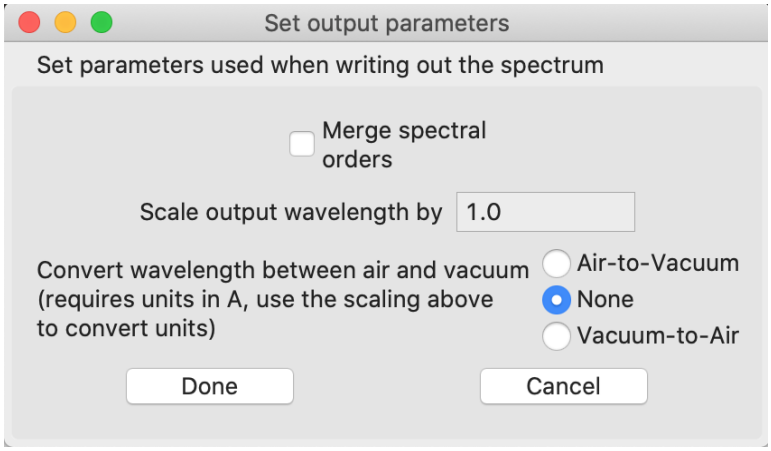
an order into consecutive search bins of equal width and extracting the **mean** flux within each bin. The panel below is a preview of the normalization product that would be output by the code. The axes are synced and allow the user to comparatively inspect a spectral order and the respective normalized spectrum within the same range of wavelength. The lower panel updates anytime the new fit is explicitly computed.

We provide controls for navigating the GUI plot:
Auto Scales, **Auto-y**, **Zoom in**, **Zoom out**, and panning buttons (**left, right, up, and down**).The python default below these buttons also work for navigating the frame **home position**, **revert step, redo step, pan, zoom, and snapshot.**

We provide different tools for deriving an optimal fit of the individual orders. Here, we will explain the other functions and features that can be utilized in the fitting process.

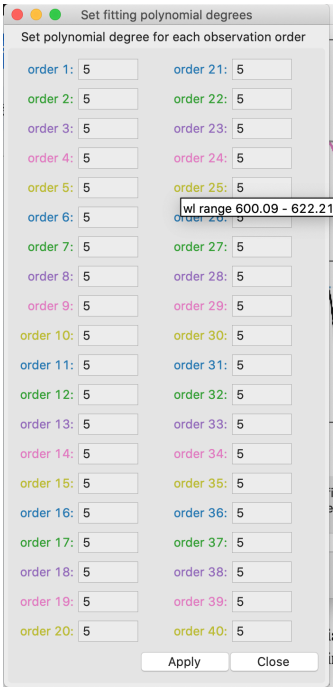
Fit cont.
Clicking this button will apply any parameter changes made and update the continuum fit. It will also update the final normalized spectrum.

Set output params



Here you can determine the structure of the spectrum in the output file. If you select **merge spectral orders** the final spectrum will be continuous rather than separate orders (if separate there will be wavelength overlap between consecutive orders). In the process of merging the orders, the program will chose the spectrum of the upper order. The lower panel in Figure 1 will change accordingly, so you will be able to preview the merged normalized spectrum. The wavelength **units** can also be scaled. If converted to angstroms (set the scale to 10), the wavelengths can be converted between air and vacuum wavelengths.

Set poly. degree.



Here the degree of the polynomial fits (of the black bin points) can be set for each individual order. The color of the text corresponds to the color by which that order is plotted. If you hover your mouse over one of the inserts, text will pop up indicating the wavelength range for that given order. This will help in searching for a specific order on the plot. After clicking apply and then **fit cont.**, you will see a change in the polynomial plotted over the order and in the normalized spectrum in the region corresponding to that order.

fill edge order gaps

average length

Within all the search bins, you can apply boxcar averaging to smoothen the data before extracting the point (mean flux). The input sets the width (ie. number of neighboring data points on the spectrum) that will be used in the averaging. This is most advantageous to correct for fit points that favor noise. This smoothening is only in effect when computing points for the fits of the orders, so you will not produce a smoothed normalized spectrum.

srch. bin (km/s)

This value sets the width of the search bins. Recall, that there is only one point extracted per bin. As you increase the width of the bins, you decrease the number of fit points for each order and vice versa.

exclude range

You can select specific ranges of wavelengths to be excluded from the fitting process. After clicking this option and going into either panel, the first click will set the starting edge of the exclusion range. A boxed box made of dashed lines will appear with one edge fixed on where the first click. The second vertical edge will follow the mouse in the panel, and all wavelengths within this box will be excluded. A second click will finalize this selection and after pressing fit. cont. you will see that portion of the spectrum change its color to black. Any fit points in that region will also disappear.

include range
You can reverse the exclusion and select

hide norm
You have the option to remove the lower panel (or return it) at your leisure. The upper panel will expand to the width of the full window. This feature can be useful for those doing the normalization of small computer screens, and would rather have fuller viewing of the original spectrum and fits.

Tutorial

This tutorial will formally introduce the reader to the procedure of normalizing echelle spectra using NormPlot2.1.py. This set of codes has been integrated into the SpecpolFlow pipeline as a preliminary step toward polarimetric analysis. As a formal introduction to spectra normalization, we are using observations of the star Psi 1 CMa which were obtained with Espadons. This is also the observation that is used in the example LSD and Mean Longitudinal Field tutorial, so feel free to test out your own normalized data through the rest of the pipeline 😁.

Before proceeding with the tutorial make sure your computer/machine is using python3. Confirm that your version of python has the following sub packages installed:

If you do not which to have the full Specpol Flow, make sure that you have the following:

Why do we normalize?

1) Converting from Fits to text.

Let's proceed by converting the format for the data files from .fits to the .s text format. The data file needs to be converted in order to be used in NormPlot2.py. These processed data files (can also be found on CADC archives) are formatted as: i.fits contains spectroscopic data (usually 4

spectra), and **p.fits contains the polarimetric data.** We want to convert the p.fits files and use the Stokes I spectrum the normalization code. For this tutorial, we will use the ESPaDOns observation, **2378200p.fits**.

ESPaDOns p.fits files have 24, and we are going to use the file-conversion script **convert-espadons-fits.py** to extract the first 12 columns. These particular columns are the normalized (0-5) and unnormalized (6-11) polarimetry products with the automatic wavelength correction applied. The columns are [0] Wavelength, [1] Intensity, [2] Stokes, [3] Null 1, [4] Null 2, [5] Errorbar, and the latter 6 follow an identical sequence.

In the command line, write the command

```
python convert-espadons-fits.py  inputfile-p.fits
```

where you will replace the inputfile-p.fits with the name of our Espadon file **2378200p.fits**. The script will return **2378200pn.s** (normalized data) and **2378200pu.s** (unnormalized data) in the same directory as the fits file.

We want to note that this is a specific example, and the reformatting of .fits files from other polarimeters is necessary in order to use this normalization code. For example, a similar conversion can be applied to SPIRou products, p.fits, t.fits, and e.fits, using the convert-spirou-fits-s0.2.py**.**

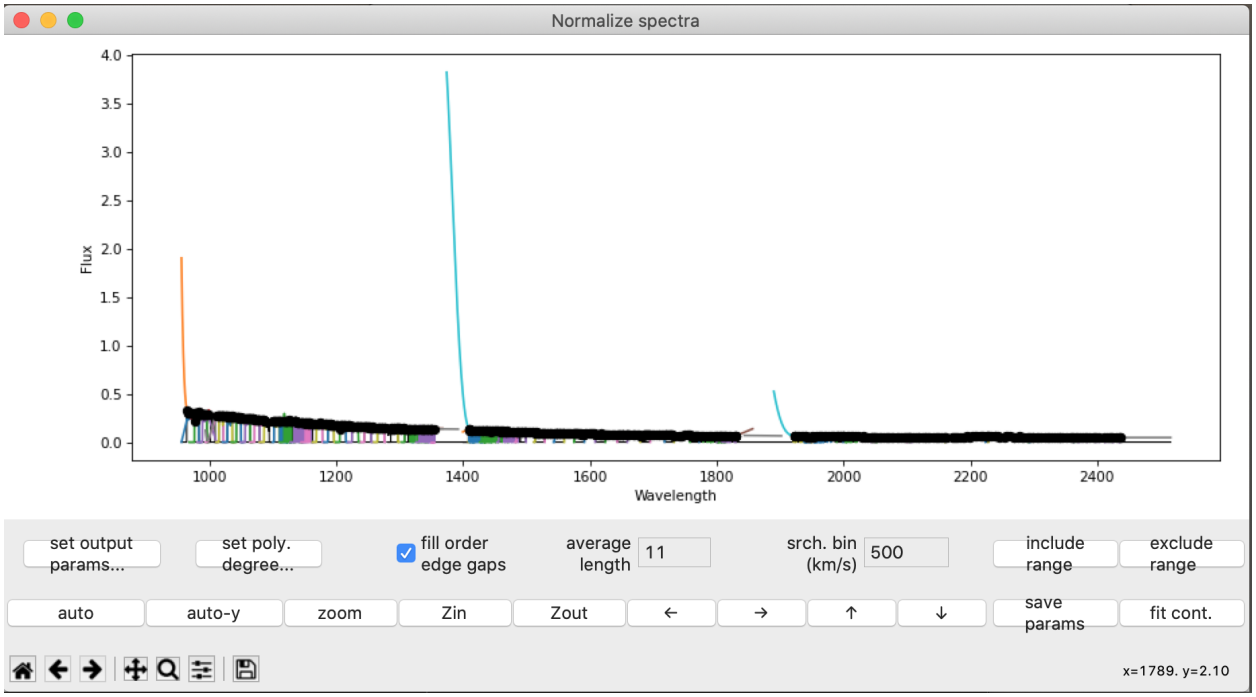
2) Normalization UI

Our goal in this process is to fit a curve onto the star's continuum in every order. From there we can normalize the continuum. This creates a 'common line' (a flat horizontal line at 1) about which we can consistently measure the properties of spectral lines.

Now let's use the normalization GUI on our data. The main file for executing the normalization is normPlot2.py. This normalization code utilizes the text format spectra that would have been produced in the first steps (it was built around LibreESPRIT reduced data). On the backend it runs off of fittingFuncs2.py (has functions for the normalization), and guiMainWin2.py and guiControls2.py (functions for the UI). You can run the code in the command line by

```
python normPlot2.py <observation_file.s>
```

The files you want to execute should be in the same working directory as normplot.py. Once executed, a window will pop up and you will see the spectra separated by order with polynomial fit (these are fits to the continuum per order and they will vary in color based on their order).



In the bottom left corner there are the conventional commands that allow you to navigate this python interface (panning, zooming, returning to default, saving).
Then they are also added above it.

The fit points: Each order is broken down into a set of consecutive search bins with respect to wavelength. They are equal in width (in velocity. The black dots are the median flux in those bins which are placed at the bins' centers (x-axis wise). Collectively the black dots within an order are the points that a polynomial function maps in order to fit the continuum of that order. The optimal outcome of this process is generating black dots representative of the continuum flux so that when we normalize it is near-uniformly 1. In other words, you want a flat line at f = 1 with various spectra lines added onto it.

From here we will refer to the buttons addressed in the User Guide. The format of this guide will address each item we must give attention to in the normalization process.

A few things that ideally will become second nature while doing this tutorial:

- If you want to see the ranges of orders, you can open the polynomial degree panel. When you hover the mouse over a specific order, a prompt will appear indicating the

wavelength range of the order. Also, colors in the panel match the plot-colors of their respective orders.

- To capture more of the depth of the line, when they appear small , Use python default zoom to create an enclosed area that is the width of the window, but a much smaller height about the spectral line.
- Python's default panning, and zoom button get overridden by the include and exclude region functions. Make sure they are not on when you're exploring the spectra.
- To see the outcome of any changes you've enacted either panel, you have to click fit. cont.

For now we will keep the default 500 km/s search bin width.
We will keep

1) **Telluric/ Noise Dominated orders:** Let's get through some of the less rigorous things first. In some instances your observations may have have orders that have very low signal to noise ratio or are saturated with telluric lines (sharps lines created from absorption by the atmosphere). It can be difficult to try to apply the full set of fitting procedures on these orders to get them flat. One of the most direct fixes to this is to reduce the order of the polynomial degree.
   Click **set poly. Degree**
   Change 1, 2, 38, 39, 40 to lower values. 3/2 might be a good value.

2) **Removing Spectral lines from the fit:** Remember we essentially make a common line that the spectra lines extend from, so we must do our best to make sure they aren't contributing to the fits. This means we don't want any of our fit points resting on spectral lines. Wherever we see fit point on spectral lines us we use the **exclude range** function to remove those line from being options to place fit points. For example, let's remove this line at ~383 nm.
   If your exclude region is too big, you can use include to regions to bring back portions of the spectrum that you would like back.
   Now there are a few other lines we need to tackle at: 397, 410, 434 (H_gamma) , 438, 486 (H_beta), 646, 656(H_alpha), 850, 860, 866, 875, 810, 901, 905???

   One advantage of this is that if we adjust the width of your search bins later, you will not have to worry about new points appearing on the same line.
   Note: Some instances you will have spectral lines that exist on the overlapping edge of two orders like H_alpha and H_beta.
   What is the fix?

For some lines this can be managed by having the fill order edge gap on. Once the line is excluded, the program will try to complete the fit using the nearest fit point from the neighboring order.

3) **Fit Points on Telluric Lines:** You may occasionally find some fit points being placed on telluric lines. We exclude those regions in an order as well (especially if you choose to adjust the bin widths later on). For example let's exclude the region about 759 nm.

4) **Fit Points on Noise:** You may occasionally find a fit point resting in the noise level on the continuum. To try to correct this we can adjust the **average length** and increase it to a slightly larger number. What's happening here is there's a small window of width that we specify that is moving across the spectrum one data point at a time, and averaging the flux at each step. This essentially smoothens the spectrum contained in that bin (this does not directly affect the actual data we use) and lessens the impact of the noise when the height of our fit point is calculated.

You can use merge the product so it keeps the spectrum of the upper order, or you can keep it unmerged and use your own discretion in how you use the overlapping regions.

Finally save in output and make a copy of the file should you need to revisit the normalization later.

You can also use saved input parameters from the command line, run the code with python normPlot2.py -h for some extra information about that.