Note on PyLongslit uncertainties

Kostas Valeckas, NBI, University of Copenhagen

February 2025

This is a note describing uncertainties in the PyLongslit[1] software. For now for internal use only.

Contents

1 General remarks

The error propagation in a pipeline like PyLongslit can get extremely complicated. This is due to the many pipeline steps both include modelling and direct calculations, sometimes with the errors being somewhat correlated. For the PyLongslit software, a simple, yet consistent approach is taken to give an error estimate of the pipeline products. We provide a transparent and complete description on how the errors are calculated, and we especially disclose where simplifications/approximations are made in error estimation.

1.1 Errors on fitted models

The pipeline relies heavily on fitting mathematical models to data. The models are adjustable from the user-site, meaning that a constant number of fit parameters can not be assumed. Performing analytical error estimation for these variable models, while possible, would be extremely complicated and would increase the code complexity marginally - and therefore also make the code more vulnerable to errors. Therefore, **a heavily simplified, but robust** error estimation is used for the values produced by a fitted model. For a fitted model f_{model}, the error on the value estimated by the model is:

[1] `https://github.com/KostasValeckas/PyLongslit/tree/develop`

$$\sigma_{f_{\text{model}}} = \text{RMS}(f_{model}(\mathbf{x}) - y(\mathbf{x})) \quad , \tag{1}$$

where RMS stands for the *root mean square* ($\sqrt{\frac{x_1^2+x_2^2+...+x_N^2}{N}}$), and the difference between $f_{\text{model}}(\mathbf{x})$ and $y(\mathbf{x})$ represents the residuals, where the input vector $\mathbf{x}$ are the independent and y are the dependent data-points used for model fitting. The RMS residual metric, in experience, gives a sufficient estimate of the order of magnitude of the error, but lacks nuance as it is independent of what input vector $\mathbf{x}$ is passed to the model. Therefore, the software is engineered to produce **quality assessment plots** for every fitted model, and the user should inspect these carefully to estimate the overall quality and systematic errors of the fitted model.

1.2 Errors on direct computations

For a value $f(\mathbf{x})$, the error is estimated by the **law of combination of errors** [1, eq. 4.14]:

$$\sigma_{f(\mathbf{x})} = \sqrt{\left(\frac{\partial f(\mathbf{x})}{\partial x_1}\right)^2 \sigma_{x_1}^2 + \left(\frac{\partial f(\mathbf{x})}{\partial x_2}\right)^2 \sigma_{x_2}^2 + ... + \left(\frac{\partial f(\mathbf{x})}{\partial x_N}\right)^2 \sigma_{x_N}^2} \quad , \tag{2}$$

where the uncertainties σ_{x_i} on the input vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, are assumed to be uncorrelated.

2 Calibration errors

2.1 Master bias

The master bias B_{master} is constructed by taking a median of every (x, y) pixel value from N stacked (potentially overscan subtracted[2]) bias frames B:

$$B_{\text{master}}(x, y) = \text{Median}(\{B_i(x, y), i = 1...N\}) \quad . \tag{3}$$

The error on the median can be estimated in two ways (choice is made by the user). The first, most straightforward way is to approximate the error by using an empirical formula [6, ch. 9.2],

$$\sigma_{B_{\text{master}}(x,y)} = \frac{1.2533 \ \sigma_{std(x,y)}}{\sqrt{N}} \quad , \tag{4}$$

where $\sigma_{std(x,y)}$ is the standard deviation of the pixel (x, y) in the bias stack. The issue with using this formula is that N must be large for it to guarantee correctness. If the number of raw bias frames is small ($N < 30$), the software will throw a warning about the error estimation on the master bias possibly not being correct. The user can instead choose to use a bootstrapping algorithm to estimate the errors. The algorithm runs as follows:

1. From the existing collection of bias frames, pick the same amount of frames by randomly picking one frame at a time, allowing the same frame to be chosen several times (random resampling).
2. Calculate the median of the sample.
   ——— after this is repeated for 1000 iterations ———
3. Calculate the standard deviation of all the resampled medians.

This should provide a somewhat correct error estimation on the median even for small N of bias frames. The trade-off is that since the detector-array sizes are easily in the order of $10^3 \cdot 10^3$, even 1000 re-samples takes several minutes to perform. The user can decide whether this is necessary for the given science case and instrument.

2.1.1 Ignoring errors from overscan subtraction

If the user decides to use overscan subtraction, this also introduces some additional errors in the bias frames. However, these errors are not used in the above described procedure. The reason for this is that for regular median estimation, errors do not change the outcome. A weighted median could be used, but this is deemed unnecessary. This would introduce additional complexity and computations - with very likely no noticeable impact on the results - as the regular (non-weighted) median is extremely robust to outliers as it is [1].

[2] The user can choose whether or not to subtract overscan - as this option might not be relevant for some instruments.

2.2 Wavelength calibration

The wavelength calibration routine in PyLongslit produces a wavelength solution $\Lambda(x, \lambda)$, that maps a spacial pixel coordinate x and spectral pixel coordinate λ to a wavelength. The error estimate for the mapped wavelength is simply calculated by using equation (1):

$$\sigma_\Lambda = \mathrm{RMS}(\tilde{\Lambda}(\tilde{x}, \tilde{\lambda}) - \Lambda(\tilde{x}, \tilde{\lambda})) \quad , \tag{5}$$

where $\tilde{\Lambda}(x, \lambda)$ is the measured wavelength of identified calibration-lines at pixels $(\tilde{x}, \tilde{\lambda})$. The expression "measured wavelength" is however a bit misleading, as the wavelengths are measured by manual pattern-matching and then several fitting routines to refine the calibration-lamp/sky line wavelengths and positions, and therefore the measured wavelengths $\tilde{\Lambda}$ might carry a significant error of their own. This said, the wavelength measurement algorithm can be set to be very conservative (rejecting any fits that do not comply with user defined metrics), in order to minimize this error.

The wavelength calibration routine is by far the most complex routine in the whole pipeline, and the above described error estimation is extremely simplified in favour of robustness. We therefore advise the user to pay great attention to the quality assessment plots to asses the quality of the modelling.

2.3 Master flat-field

Firstly, a median master flat F_{master} is produced with the same procedure as with the master bias. The only difference is that the master bias and the dark current (if present) are subtracted from the raw flat-field frames before median estimation. The error introduced by this subtraction is ignored in the median estimation, with the same argumentation as given in chapter (2.1.1) for overscan.

Then, the spectral response (and possibly slit illumination profile - if user chooses) are normalized using fitted models M:

$$F_{\text{master (spectral normalized)}}(x, y) = \frac{F_{\text{master (not normalized)}}(x, y)}{M_{\text{spectral}}(x, y)} \quad , \tag{6}$$

$$F_{\text{master}}(x, y) = \begin{cases} \frac{F_{\text{master (spectral normalized)}}(x,y)}{M_{\text{illumination}}(x,y)}, & \text{with slit-illumination normalisation ,} \\ F_{\text{master (spectral normalized)}}(x, y), & \text{without slit-illumination normalisation ,} \end{cases} \tag{7}$$

For eq. (6), the error propagation is calculated using eq. (2) (omitting (x, y) in notation for readability):

$$\sigma_{F_{\text{master (spectral normalized)}}} = F_{\text{master (spectral normalized)}} \sqrt{\left(\frac{\sigma_{F_{\text{master (not normalized)}}}}{F_{\text{master (not normalized)}}}\right)^2 + \left(\frac{\sigma_{M_{\text{spectral}}}}{M_{\text{spectral}}}\right)^2} \quad , \tag{8}$$

and then, if slit-illumination normalization is performed (eq. (7)):

$$\sigma_{F_{\text{master}}} = F_{\text{master}} \sqrt{\left(\frac{\sigma_{F_{\text{master (spectral normalized)}}}}{F_{\text{master (spectral normalized)}}}\right)^2 + \left(\frac{\sigma_{M_{\text{illumination}}}}{M_{\text{illumination}}}\right)^2} \quad , \tag{9}$$

where the errors for the spectral response model $M_{spectral}$ and slit-illumination model $M_{illumination}$ are calculated from the residuals using eq. (1).

3 Reduction of the object frames

3.1 Initial error estimation of raw target frames

For raw target frames (both object and standard star), a similar approach is taken as in [5, p. 45 - 46] for initial error estimation. It is assumed that the raw frames primely consist of the detected light, the dark current (if any is present), and the bias detector level. Therefore, the initial estimate of the error of the object frames is as follows (using eq. (2) and also adding the read noise)):

$$\sigma_{initial} = \sqrt{\sigma^2_{Poisson} + \sigma^2_D + \sigma^2_O + \sigma^2_B + \sigma^2_{ReadNoise}} \quad , \tag{10}$$

where D stands for dark current, O stands for overscan and B stands for bias. $\sigma_{Poisson}$ is the photon shot noise calculated as the square root of detected light[5, p. 45]:

$$\sigma_{Poisson}(x,y) = \sqrt{C_{initial}(x,y) - D - O(x,y) - B(x,y)} \quad , \tag{11}$$

where $C_{initial}(x,y)$ are the total counts of the raw frame. For σ_D, since the software supports only subtracting a constant amount of dark current from every pixel (user inputs the dark current as a scalar in electrons/second, and then it gets converted into detector counts by the software), the error σ_D is calculated as [5, p. 45]:

$$\sigma_D = \sqrt{D} \quad . \tag{12}$$

σ_O is calculated as the error of mean of the overscan row/column depending on the detector and user input:

$$\sigma_O = \frac{\sigma_{std}}{\sqrt{N}} \quad , \tag{13}$$

where N is the number of pixels in the row/column of the overscan strip. The bias error σ_B estimation is described in chapter (2.1). The read noise $\sigma_{ReadNoise}$ is given as a scalar by user input (as electrons, and then converted automatically by the software to detector counts).

3.2 Error estimation of the reduced frames

The calibrated counts $C_{calibrated}$ for the object frames are obtained by subtracting the dark current D, frame overscan level O, detector bias B, and then dividing by the master flat F (with D and/or O being 0 for some instruments):

$$C_{calibrated}(x,y) = \frac{C_{initial}(x,y) - D - O(x,y) - B(x,y)}{F(x,y)} \quad . \tag{14}$$

Propagating the errors using eg. (2):

$$\sigma_{C_{calibrated}} = \frac{1}{F}\sqrt{\sigma^2_{initial} + \sigma^2_D + \sigma^2_O + \sigma^2_B + C^2_{calibrated}\,\sigma^2_F} \quad , \tag{15}$$

where $\sigma_{initial}, \sigma_D, \sigma_O, \sigma_B$ are described in the previous chapter, and σ_F is described in chapter (2.3).

The errors for the dark current and the detector bias get double-counted - once when estimating the initial error of the raw science frame, and once when estimating the error of the reduced frame. This approach is taken from [5, p. 45-47], and the argument for this is that since we "count" the detector counts twice, we have to account for the errors twice.

3.3 Cosmic-ray elimination

The user can choose to deploy the Astro-SCRAPPY[3] cosmic ray detection Python package [2, 4] after the frame reduction. Errors for the pixels corresponding to the detected cosmic rays are assigned a value that is the mean error of the whole error image.

4 Sky-subtraction errors

Sky-subtraction in the software is optional, and can be performed in several ways - by **A-B dithering background subtraction**, by **polynomial fitting** to acquire a sky-model, or both (executed in the order they are listed in). For A-B dithering background subtraction, both frames are to be reduced beforehand, so both frames A and B have an error σ_A and σ_B decided by eq. (15). The error of the background subtracted image σ_{sub} is then decided by eq. (2):

$$\sigma_{sub} = \sqrt{\sigma^2_A + \sigma^2_B} \quad . \tag{16}$$

[3] `https://github.com/astropy/astroscrappy?tab=readme-ov-file`

For polynomial sky-estimation, a polynomial background fit is performed in the spacial direction of every spectral pixel λ_{pix}. The error for the sky estimation for the spectral pixel $\sigma_{\lambda_{pix}}$ is decided as the RMS of the residuals (eq. (1)). The sky-model is constructed by stacking the individual fits through the spectral axis to map the whole detector. The error image of the sky-model σ_{sky} is likewise acquired by stacking the errors for every spectral pixel $\sigma_{\lambda_{pix}}$ through the spectral axis. The model is then subtracted from the frame. If the frame has an initial error σ (either from eq. (15) or eq. (16)), then the error on the frame with the subtracted sky model σ_{sub} is again:

$$\sigma_{sub} = \sqrt{\sigma^2 + \sigma_{sky}^2} \quad . \tag{17}$$

5 Extraction errors

5.1 Horne optimal extraction

The primary extraction algorithm is the Horne optimal extraction algorithm [3]. The error for the extracted spectrum at wavelength λ in detector counts C_{1d} is estimated as:

$$\sigma_{C_{1d}}(\lambda) = \sqrt{\left(\sum_x \frac{P(x,\lambda)^2}{\sigma_{C_{calibrated}(x,y)^2}} \right)^{-1}} \quad , \tag{18}$$

where x is a spacial pixel(s) corresponding to wavelength λ, $P(x,\lambda)$ is a Gaussian weight for the object aperture, and $\sigma_{C_{calibrated}}$ is described in eq. (15). For details about this equation, please refer to [3].

5.2 Simple photometry extraction

Secondary extraction option is by simply performing a photometry extraction on the estimated object aperture. For this, an out-of-the-box solution is used from `photutils.aperture.RectangularAperture`, with the method `do_photometry`. For the details on how the error gets estimated on the simple extraction, please refer to the method documentation at `https://photutils.readthedocs.io/en/2.0.2/api/photutils.aperture.RectangularAperture.html#photutils.aperture.RectangularAperture.do_photometry`.

6 Flux-calibration errors

When flux-calibrating the extracted 1d-spectrum from detector counts, a sensitivity function needs to be obtained first. This function is obtained by fitting a model to an array of sensitivity points $S_{points}(\lambda)$. These points are obtained by diving a flux-calibrated spectrum of a standard star (that is in the wanted units) with the obtained 1d-spectrum in counts pr. second of the same star. In PyLongslit default units:

$$S_{points}(\lambda) \left[\frac{erg/cm^2/\mathring{A}}{counts} \right] = \frac{\text{1d-spec flux-calibrated spectrum } \left[erg/s/cm^2/\mathring{A} \right]}{\text{1d-spec observed spectrum i counts pr. second } \left[counts/s \right]} \quad . \tag{19}$$

Fitting a model to these points gives a conversion factor $S(\lambda)$ between observed counts pr. second $C_{1d}(\lambda)/s$ to flux in physical units $Flux(\lambda)$:

$$Flux(\lambda) = \frac{C_{1d}(\lambda)}{\text{exposure time}} \; S(\lambda) \quad . \tag{20}$$

In the software, the fit for $S(\lambda)$ is performed in (10-base) log-space ($S_{log}(\lambda)$). This is because the observed 1d-standard star spectrum in counts will still have some artifacts such as absorption lines from the sky, and these might corrupt the fit. Fitting in logarithmic space scales these artifacts down. The error of the fit in logspace is calculated using eq. (1):

$$\sigma_{S_{log}} = \text{RMS}(\log\left(S_{points}(\tilde{\lambda})\right) - S_{log}(\tilde{\lambda})) \quad , \tag{21}$$

where $\tilde{\lambda}$ are the wavelengths at which the sensitivity points were evaluated.

To convert from $S_{log}(\lambda)$ to $S(\lambda)$, the following expression is used:

$$S(\lambda) = 10^{S_{log}(\lambda)} \quad , \tag{22}$$

and using this expression with eq. (2) results in following error:

$$\sigma_{S(\lambda)} = |\ln(10)\ 10^{S_{log}(\lambda)}\ \sigma_{S_{log}}| \quad . \tag{23}$$

When flux calibrating a 1d-spectrum that is in counts using eq. (2) and eq. (20), and assuming the error on the exposure time is negligible, the error propagates to:

$$\sigma_{Flux(\lambda)} = \sqrt{\left(\frac{S(\lambda)}{\text{exposure time}}\sigma_{C_{1d}}\right)^2 + \left(\frac{C_{1d}(\lambda)}{\text{exposure time}}\sigma_{S(\lambda)}\right)^2} \quad , \tag{24}$$

where $\sigma_{C_{1d}}$ is defined in eq. (18) or in chapter (5.2), depending on what extraction algorithm is chosen by the user.

7 Spectrum-Combining errors

PyLongslit has a routine for combination of flux-calibrated 1d-spectra. The spectra are combined as a weighted mean $Flux_{\mu}(\lambda)$, where the errors from eq. (24) are used as weights [1, p. 54]:

$$Flux_{\mu}(\lambda) = \frac{\sum_i \frac{Flux_i(lambda)}{\sigma^2_{i_{Flux(\lambda)}}}}{\sum_i \frac{1}{\sigma^2_{i_{Flux(\lambda)}}}} \quad , \tag{25}$$

with the error:

$$\sigma_{Flux_{\mu}(\lambda)} = \sqrt{\frac{1}{\sum_i \frac{1}{\sigma^2_{i_{Flux(\lambda)}}}}} \tag{26}$$

References

[1] R. Barlow. *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences*. Manchester physics series. Wiley, 1995. URL: https://books.google.dk/books?id=rvxunQEACAAJ.

[2] Pieter G. van Dokkum. "Cosmic-Ray Rejection by Laplacian Edge Detection". In: *Publications of the Astronomical Society of the Pacific* 113.789 (Nov. 2001), pp. 1420–1427. ISSN: 1538-3873. DOI: 10.1086/323894. URL: http://dx.doi.org/10.1086/323894.

[3] K. Horne. "AN OPTIMAL EXTRACTION ALGORITHM FOR CCD SPECTROSCOPY." In: *Publications of the Astronomical Society of the Pacific* 98.604 (June 1986), p. 609. DOI: 10.1086/131801. URL: https://dx.doi.org/10.1086/131801.

[4] Curtis McCully, Steve Crawford, Gabor Kovacs, et al. *astropy/astroscrappy: v1.0.5 Zenodo Release*. Version v1.0.5. Nov. 2018. DOI: 10.5281/zenodo.1482019. URL: https://doi.org/10.5281/zenodo.1482019.

[5] James Burnell Richard Berry. *The Handbook of Astronomical Image Processing*. 2nd ed. Willmann-Bell, 2005. ISBN: 0943396824; 9780943396828.

[6] Dennis O Wackerly, William Mendenhall, and Richard L Scheaffer. *Mathematical statistics with applications, international edition*. 7th ed. Florence, KY: Brooks/Cole, Oct. 2007.