This wavelength calibration library is used to find the polynomial that will convert a pixel spectrum into a wavelength spectrum with the most accuracy. It can be used on all channels of raw data, or one channel at a time. The intended use is to improve the calibration of future readings, but the found polynomials can also be used to recalibrate the existing data for more accurate analysis.

Results are in the format [c0, c1, c2, c3] which corresponds to the polynomial c3x³ + c2x² + c1x + c0

Contents

[Usage guide](#usage_guide)

* [On a single pixel spectrum reading](#_On_a_single)
* [On an already calibrated dataset](#_On_an_already)
* [Using a different reference spectrum](#_How_to_use)

[Available graphs](#available_graphs)

* [All calibrations](#_All_Calibrations)
* [Segmented linear calibration](#_Segmented_Linear_calibration)
* [Matched dips](#_Matched_Dips)
* [calibration](#_Calibration)
* [Calibrated spectrum](#_The_calibrated_spectrum)
* [Individual segment calibration](#_Individual_segment_calibration)
* [Tested calibration histogram](#_Tested_Calibration_Histogram)

[Adjustment parameters](#adjustment_parameters)

* [find\_peaks](#_find_peaks)
* [default\_calibration](#_default_calibration)
* [reference\_dip\_locations](#_reference_dip_locations)
* [num\_dips](#_num_dips)
* [centre\_point](#_centre_point)
* [dist\_from\_centre](#_dist_from_centre)
* [c0\_lim](#_c0_lim)
* [c1\_lim](#_c1_lim)
* [iterations](#_iterations)
* [init\_c0\_lim](#_init_c0_lim)
* [init\_c0](#_init_c0)
* [left\_cutoff](#_left_cutoff)
* [num\_reference\_dips](#_num_reference_dips)

[Text file formats](#text_file_formats)

* [Pixel spectrum reading](#_Pixel_spectrum_reading)
* [Reference spectrum](#_Reference_Spectrum)
* [Output file](#_output_file_1)
* [Wavelength calibration file](#_Wavelength_calibration_file)

[Documentation](#Documentation)

* [RawDataset](#_RawDataset)
  + [find\_all\_calibrations](#_find_all_calibrations)
  + [plot\_calibration](#_plot_calibration)
  + [plot\_calibration\_wide](#_plot_calibration_wide)

Usage Guide:

# On a single pixel spectrum reading

1. Read the .txt file containing the spectrum. The format that the data should be stored in can be found here

pixel\_dataset = hsr.read\_txt.read\_raw\_pixels(data\_file)

1. Create a [RawDataset](#_RawDataset) object, using the pixel dataset, and specify spectrum\_type=”pixel”.

dataset = hsr.spectrum.RawDataset(pixel\_dataset, spectrum\_type="pixel")

1. Run [dataset.find\_all\_calibrations](#_find_all_calibrations)(). You can just run it without any parameters, but you can control where the results are saved, what graphs are displayed during the calibration, and specifics on how the calibration is performed.

new\_cals = dataset.find\_all\_calibrations(file\_output=output\_file, plot\_summary=True)

1. The found calibrations are stored in the file given at file\_output, and are also stored in the new\_cals variable.

# On an already calibrated dataset

1. Fetch the data. It should be in a pandas DataFrame with one column for each channel, one row for each reading and the spectra should be stored in numpy float arrays. This is an example of loading from a database.

raw\_driver = hsr.RawDbDriver(db\_name)

raw\_data = raw\_driver.load()

1. Create a RawDataset object, using the raw data.

dataset = hsr.spectrum.RawDataset(raw\_data)

1. Run [dataset.find\_all\_calibrations](#_find_all_calibrations)() You can just run it without any parameters, but you can control where the results are saved, what graphs are displayed during the calibration, and specifics on how the calibration is performed.

new\_cals = dataset.find\_all\_calibrations(file\_output=output\_file, plot\_summary=True)

1. The found calibrations are stored in the file given at file\_output, and are also stored in the new\_cals variable.

# How to use a different reference spectrum.

The reference spectrum is critical to calibrating the spectrometer. Dips in the reference spectrum are matched to dips in the measured spectrum, and a calibration is generated from that. By default, a measurement generated by SMARTS is used. If you are using a calibration lamp, or have a more accurate solar measurement, you can use that as the reference for more detailed results.

To use a different reference spectrum, you can pass it into dataset.find\_all\_calibrations directly as an 801-float long numpy array, or pass in the file location where it is stored. This file must be either a [simple list](#_Simple_list) or a [csv reference file](#_CSV_reference_file)

The calibration software tries to match dips by default, but if you use a spectrum that has more peaks than dips, you can pass find\_peaks=True, it will match on peaks instead.

dataset.find\_all\_calibrations(reference\_filepath="spectra/TAOR\_reference.txt", find\_peaks=True)

Summary of Algorithm

1. A “weighted spectrum” is generated from the measured dataset. This is calculated by finding the wavelength and prominence of the n largest peaks and adding the prominence to a total for each wavelength. The result of this over a dataset is that the weighted spectrum will have a few tall and very narrow peaks(usually 1-2 nm wide) with the taller ones being more common and/or more prominent. The same process is applied to the reference spectrum, but as there is only 1 spectrum to look at, only the peaks’ prominence has any effect on the reference weighted spectrum.
2. Once there are 2 weighted spectra, they are split into segments and a linear calibration (c1\*measured + c0 = new\_measured) is applied to each measured segment to try and match the peaks together. The first segment covers the wavelengths from centre\_point-dist\_from\_centre to centre\_point+dist\_from\_centre. When finding a linear calibration for a segment, the program tries many combinations of c0 and c1 within c0\_lim and c1\_lim of the previous segment’s c0 and c1 values. The quality of the match between peaks in the measured spectrum and the reference spectrum is decided by multiplying the weighted spectrum after each linear calibration with the corresponding segment of the reference\_spectrum. This means that when dips are well matched, high values will be multiplied by high values, resulting in a high fit quality. The pair of c0 and c1 values that produce the highest fit quality are used as this segment’s linear calibration. The starting point for each segment’s linear calibration is the c0 and c1 values from the neighbouring segment, starting from centre\_point, and working out to the left and then to the right. The first segment that is calibrated uses c0\_init and c1\_init.
3. The weighted spectrum segments are then all combined together to form one weighted spectrum with several linear calibrations. Peaks in the reference spectrum are paired with peaks in the weighted spectrum, and then the linear calibration, and the existing calibration is reversed, so there is now pairs of pixels and wavelengths that are points in the measured spectrum, and the corresponding points in the reference spectrum. Now, it is possible to run a line fitting algorithm that will return a polynomial that will map the measured pixel dips onto the reference dips, and by extension, the measured pixel spectrum onto the reference spectrum.

Available graphs

There are a few options for graphs:

Plot\_total\_output: after all channels have a calibration, plots

* All Calibrations: plot of all the new calibrations.

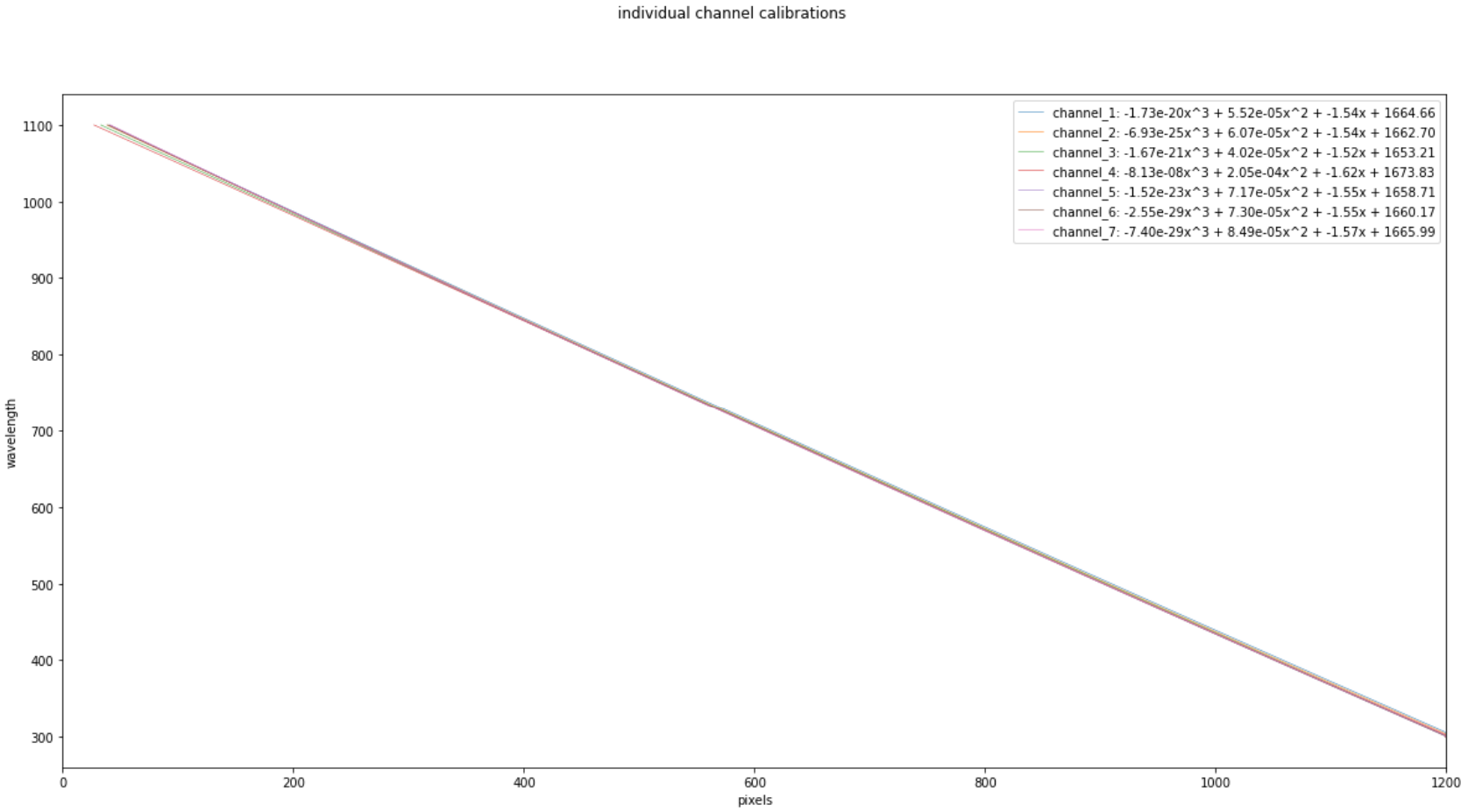
Plot\_individual\_results: for each channel, plots

* The Segmented Linear calibration, a combination of each segment of the spectrum, calibrated with the best linear calibration for that segment
* The matched dips: a plot of pixel\_spectrum and reference spectrum, with the dips that will be used to generate the calibration plotted as dots on the corresponding spectrum
* Calibration: plots the original and the recalculated calibration polynomial, with the chosen dips also plotted
* Calibrated spectrum: plots the original calibration, the recalibrated spectrum, and if provided, the reference spectrum. Plots the first spectrum in the dataset.

plot\_debug: for each segment in each channel, plots

* Individual Segment Calibration: Weighted windowed reference spectrum and the weighted measured spectrum with the found linear calibration applied to it.
* Tested calibration histogram: plots all the tested stretch and shift values, with the colour of the plot representing the fit quality.

# All Calibrations



X-axis: pixels.

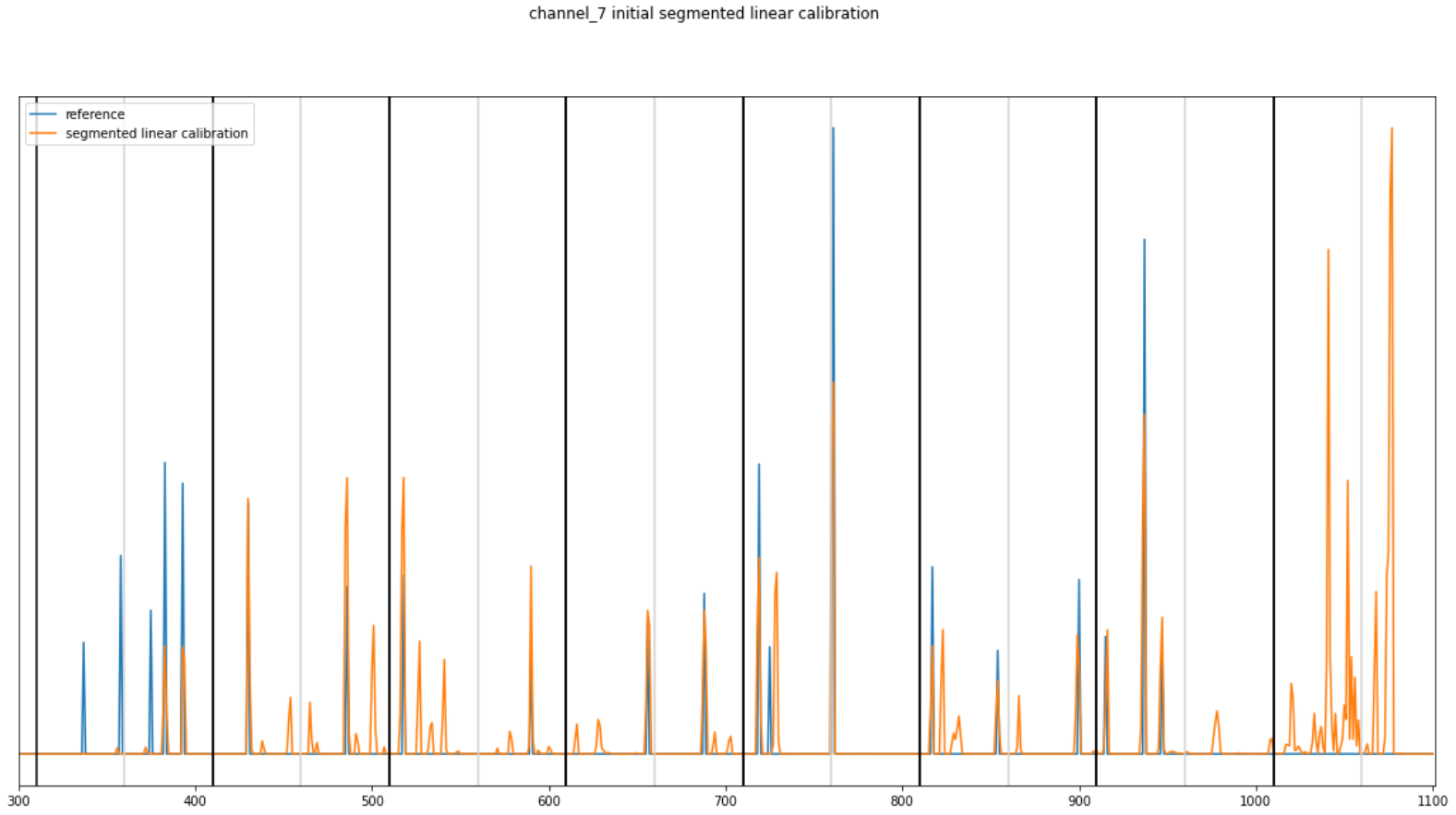
Y\_axis: wavelength.

Coloured lines: each line represents one channel’s calibration.

Can be used to find channels that are more miscalibrated than the rest

To plot it directly, see the documentation [here](#_plot_calibration)

# Segmented Linear calibration



Blue graph: the weighted spectrum generated from the reference spectrum.

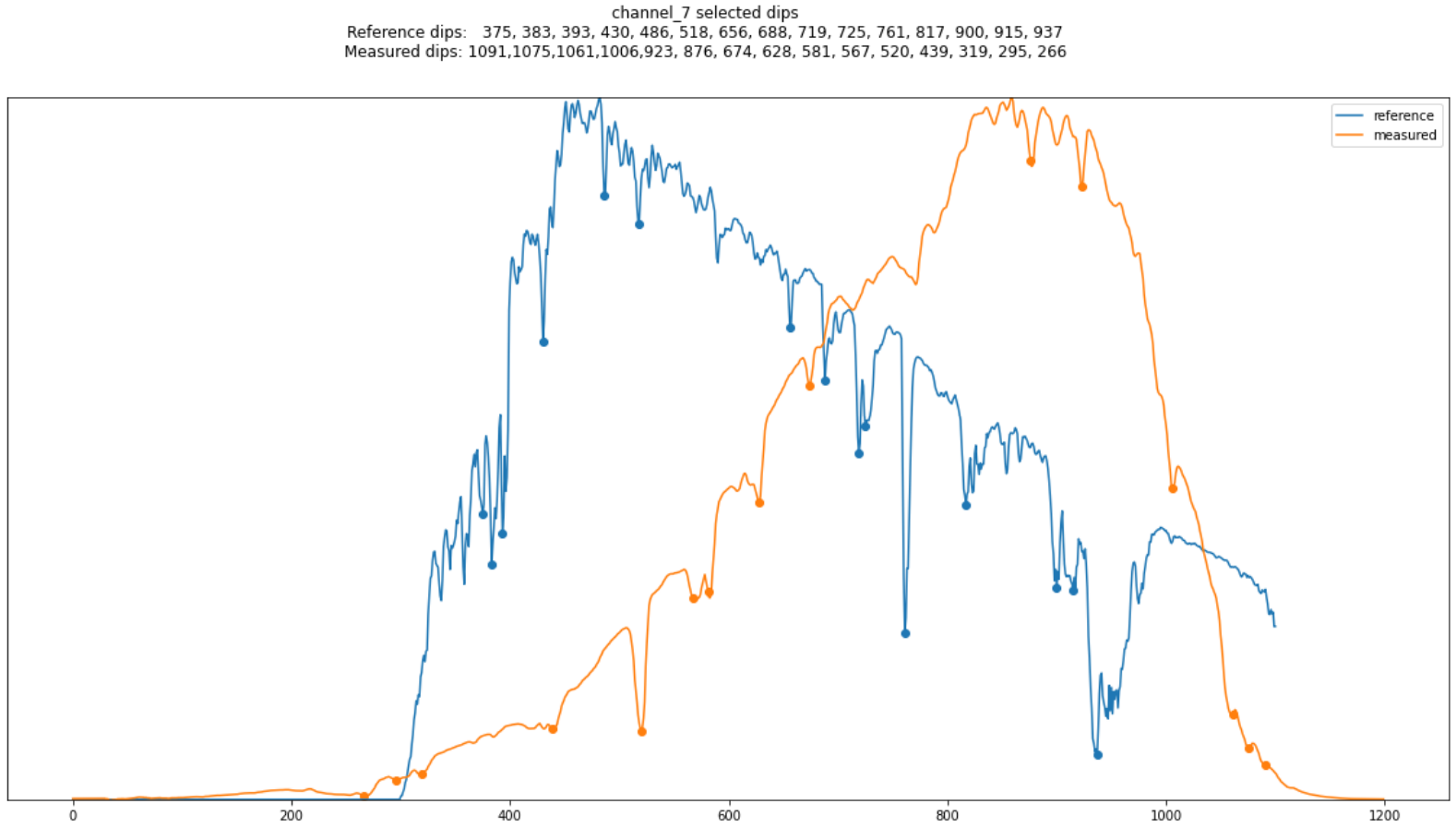
Orange graph: the weighted spectrum generated from the measured dataset, with each segment’s calibration applied to each segment.

Black lines: the borders of each segment.

Grey lines: the centre point of each segment.

If the correct calibration is not found, this graph will help to find where the error is. If this graph shows corresponding dips matched closely together, then the error is probably somewhere else. If the dips are incorrectly matched, or not matched at all, then “plot\_debug=True” may be helpful.

# Matched Dips



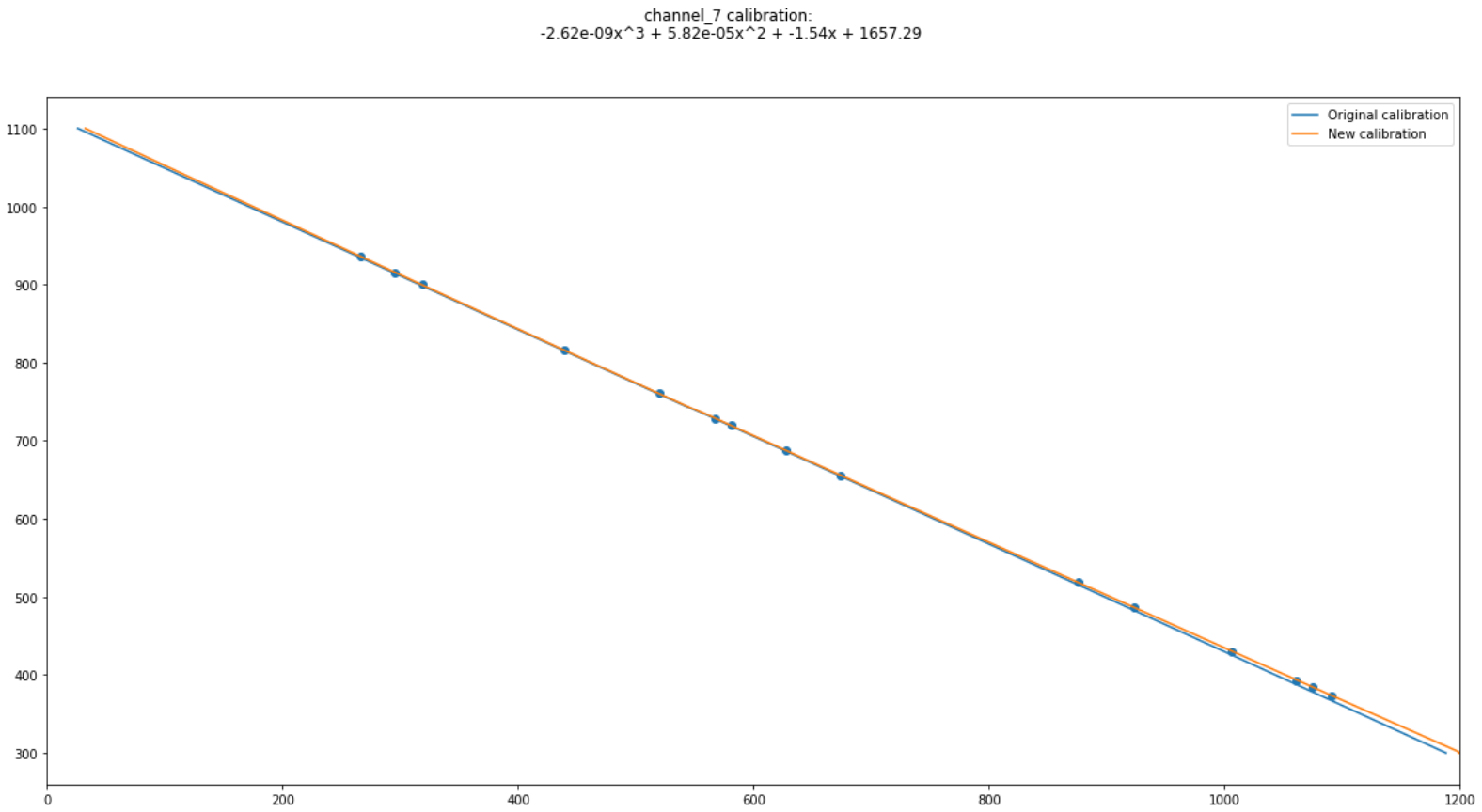
Blue line: the reference spectrum, with the locations of the found dips superimposed.

Orange line: the first measured spectrum, with the locations of the found dips superimposed.

This graph can show when the found dips don't match the actual position of the dips, or if they are being mismatched. Shows the dips that will be used to calculate the calibration.

Note: as the spectrum is taken from the first measurement, and the dips are found from the whole dataset, there may be slight discrepancies.

# Calibration



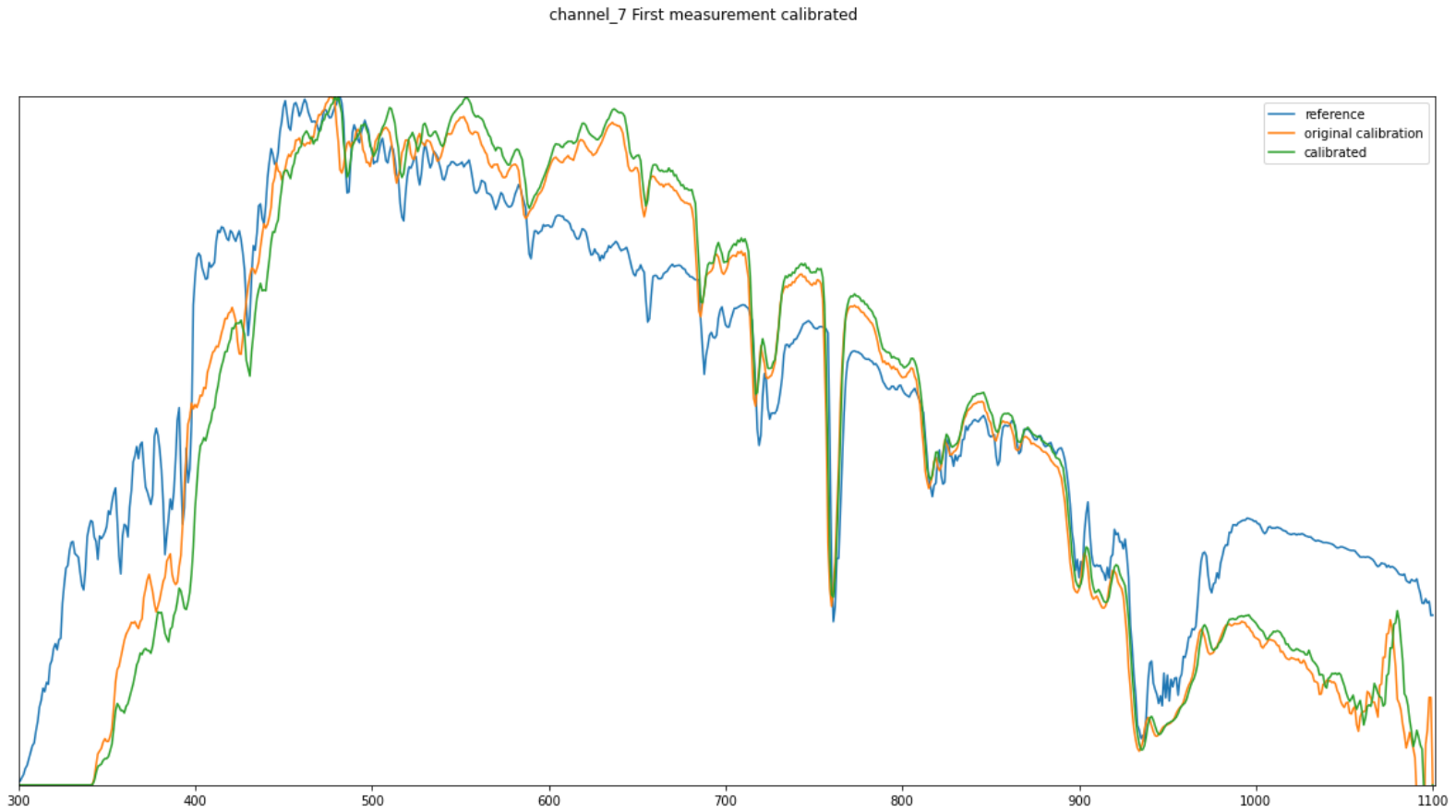
Blue line: the original calibration polynomial.

Orange line: the newly found calibration polynomial.

Blue dots: the reference dip locations against the matched dip location.

Plots the found calibration and the old calibration.

# The calibrated spectrum



Blue line: the reference spectrum.

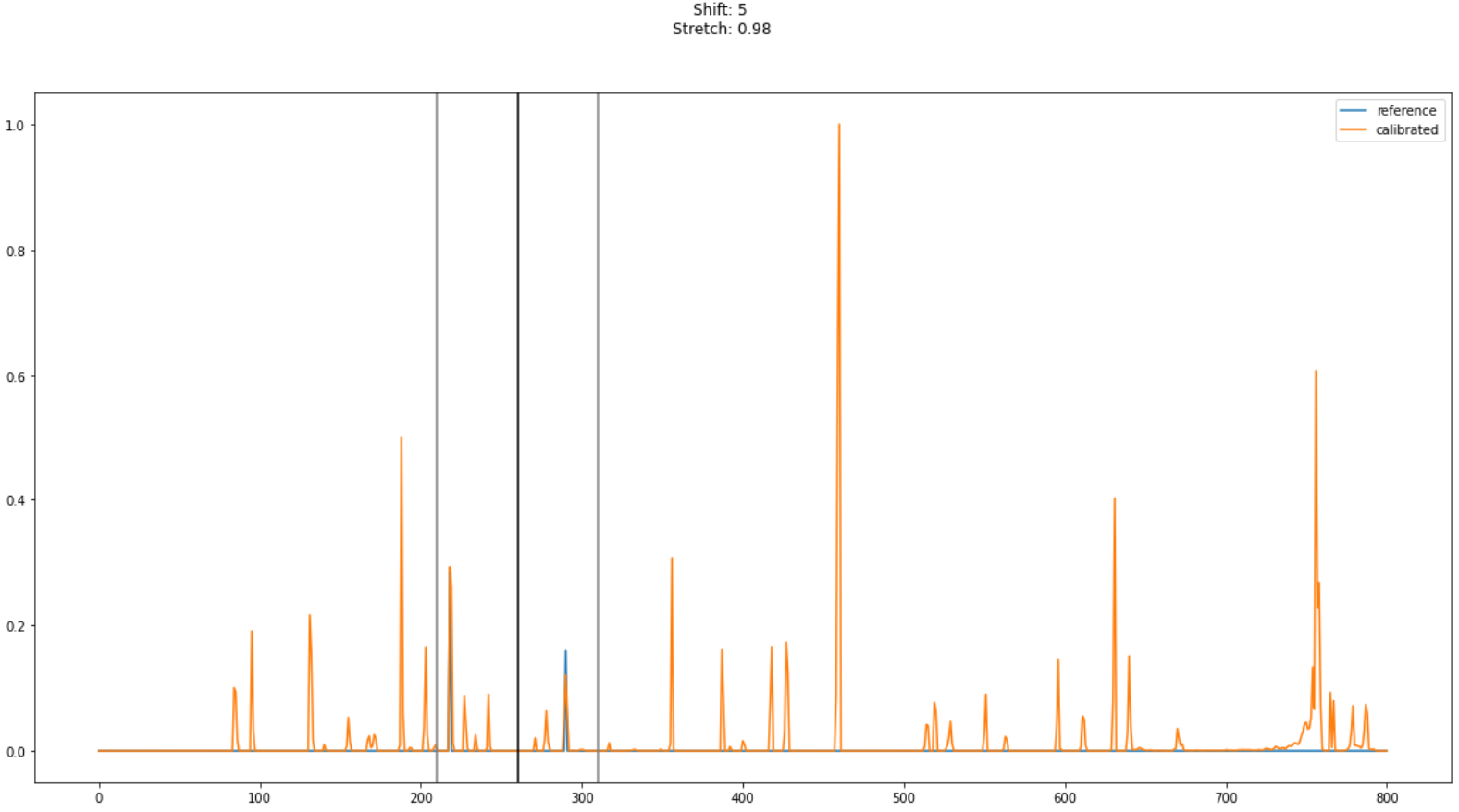
Orange line: the first measured spectrum with the original calibration.

Green line: the first measured spectrum with the new calibration applied.

This is the simplest check to see by eye if the calibration was found successfully and how accurate it is.

Note: as the spectrum is taken from the first measurement, and the calibration is calculated from the whole dataset, there may be slight discrepancies.

# Individual segment calibration

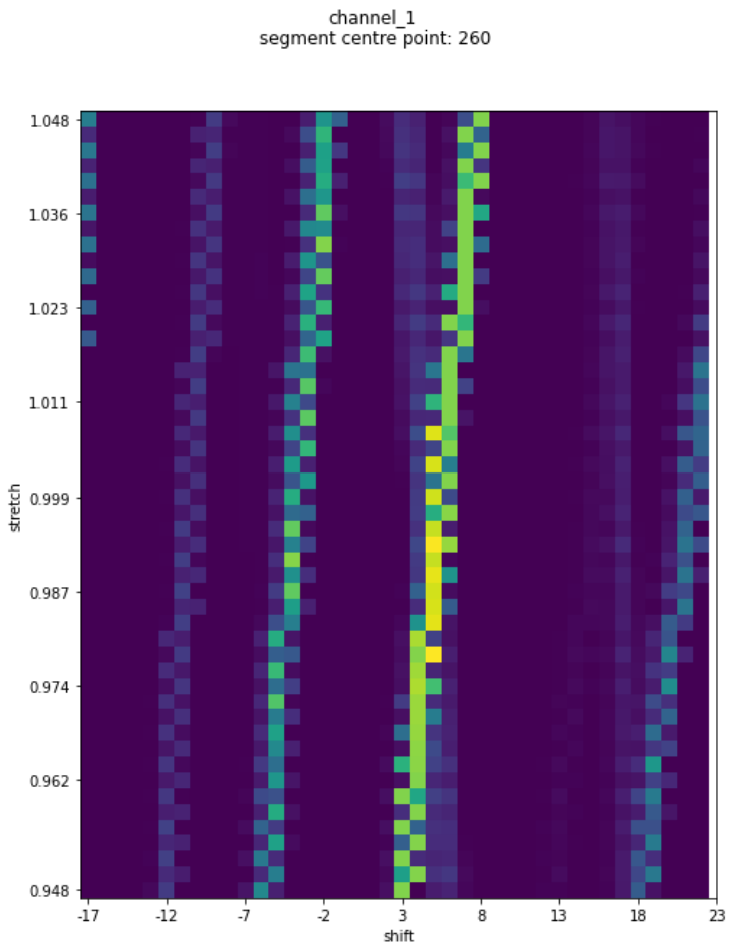


Blue line: the reference weighted dips spectrum, cropped to only include the current segment.

Orange line: the measured weighted dips spectrum, with the stated stretch and shift applied, around the centre point of the segment.

Useful for checking if the correct peaks are matched up and haven’t been confused by tiny peaks.

# Tested Calibration Histogram



X-axis: how much the spectrum has been shifted to the left/right.

Y-axis: how much the spectrum has been stretched.

Colour: fit quality, the sum of shifted and stretched spectrum multiplied by cropped reference.

This graph shows all the different shift and stretch values that were tried, with their results, this is useful for understanding which specific linear segment calibration was chosen, and also helpful when choosing c0 and c1 limits, if a bright peak far away from the initial calibration is being chosen, that could be a large dip from a different segment getting matched, and it is best to decrease the limits. Similarly, if there is no bright peak in the graph but rising intensity towards the edges, it could be worth increasing the c0 and c1 limits.

Note: the initial calibration is taken from the previous segment, which is why for most plots the axes aren't centred around 0/1.

Adjustment parameters

If the default parameters don't correctly calibrate the spectrum, they can be adjusted. Here is an overview of what each parameter does. Most parameters control how the weighted\_spectrum is created, and how the segmented linear calibration is carried out, as that is the part of the calibration that most often fails.

The key parameters to include are:

* Find\_peaks if you are calibrating against a peaky spectrum
* Centre\_point if the most prominent dip is not around 760.

## find\_peaks

default: False

whether your spectrum has more peaks or dips. Daylight spectra have much more defined dips from absorption, but calibration lamps may have very well-defined peaks, in which case find\_peaks = True is essential.

## default\_calibration

default: [1623, -1.4, 0, 0]

if using pixel data, the default calibration is applied so that the spectrum is similar enough to the reference to find the correct calibration. The closer this default calibration is to the actual calibration, the better. If a calibration for a similar instrument is known, it can be helpful to pass it here, then decrease c0\_lim and c1\_lim.

## reference\_dip\_locations

Default: None

If the automatically found reference dips are inaccurate, or poorly spaced along the spectrum, a list of wavelengths corresponding to the desired dips can be passed.

## num\_dips

Default:15

The number of largest dips to select from the reference spectrum and match to the measured spectrum when finding the polynomial. A good calibration can be found from a small number of dips but generally, the more accurately matched dips are used to find the polynomial, the better, as the calibration is consistently accurate throughout the spectrum, rather than being accurate where the dips are then drifting in the gaps between them

If reference\_dips\_locations is provided, this parameter is ignored, and the number of reference\_dips is used instead.

**The following parameters are all related to how the weighted spectrum is generated and how the linear calibration is carried out.**

## centre\_point

Default: 760

This is the centre of the first segment that will be calibrated. The first segment is the hardest to match, so it should contain the largest and most distinct dips and have the largest close to the centre. The default is 760, to match the daylight O2 dip.

There can sometimes be errors when there are reference dips on either side of the border between segments when there are also not many large reference dips in the rest of either of the segments. When this is the case, the same measured dip can be used by both segments. If this happens, alter the centre\_point and dist\_from\_centre values, or limit c0 and c1.

## dist\_from\_centre

Default: 50

This is how far the edges of each segment should extend from the middle of the segment. Segments should be wide enough that there are enough reference dips to find an accurate linear calibration, but narrow enough that there is a linear calibration that can match all or most of the dips.

It is also important to make sure that all the biggest reference dips are included within the linear calibration, as if dist\_from\_centre is large, e.g: 100, the first spectrum could start at 390 (as segments with a centre point outside the spectrum are not calculated) which would mean that reference dips around 380 would not be calculated accurately and the calibration would fail.

## c0\_lim

Default: 20

When finding a linear calibration for each segment, this parameter controls how many c0 values are tested.

Smaller c0\_lim values help make sure a segment’s calibration does not include dips from outside the segment, but if it is too low, it could exclude some data on particularly miscalibrated spectra. This value doesn't apply to the centre segment, which has its own parameter: init\_c0\_lim.

The limits for testing are the c0 calibration from the previously calculated segment +- c0\_lim.

Only integer c0 values are tested.

## c1\_lim

Default: 0.05

Similar to c0\_lim, controls the range of c1 values tested when finding linear calibrations for segments

The parameter “iterations” controls the number of values tested within the c1\_lim range.

## iterations

Default:30

Defines the number of c1 values to attempt for each c0 value, within the limits provided from c1\_lim. Higher values here will lead to more accuracy but will take longer to execute.

## init\_c0\_lim

Default: 20

The c0 limit for the centre segment. Use this parameter if the spectrum is currently very miscalibrated and the main dip is not being matched, or in conjunction with init\_c0.

## init\_c0

Default: 0

The starting value for c0 in the central segment. If the linear calibration for the first segment is not being accurately found, decrease init\_c0\_lim and set the initial value for c0 by eye (it is easiest to use [the calibrated spectrum graph](#2pqxx1lldw4) and compare original/default calibration). A single value can’t be accurately used for all channels, as they are usually all miscalibrated by different amounts, so it is best to keep init\_c0\_lim above zero, the best value will depend on how much the channels differ and how accurate init\_c0 is.

## left\_cutoff

Default: 360

The minimum wavelength to try and match between the reference and measured spectrum. SMARTS has data down to 300 nm whereas the spectrum is only measured down to 350 nm. This parameter prevents attempting to match dips that don't exist in the measured spectrum.

## num\_reference\_dips

Default: 30

The number of dips to use when creating the weighted\_reference spectrum (blue line in the segmented linear calibration graph). Selects the largest dips, so decreasing this value can be used to filter out smaller dips so the segment calibration doesn't get confused.

Increase this value if there aren't enough reference dips to get a good segmented calibration.

Text file formats and methods

## Pixel spectrum reading

This file contains the raw pixel data from one reading, over all channels. It has 2 lines of metadata, then the rest is a csv file using tabs as delimiters.

The first important line is the column headers. These are tab separated strings, one for each column. The first column is the pixel, and the rest are each a channel.

Each row after that contains the data for one pixel, as tab separated values

There is data for pixel 0-1200 (inclusive)

Example file:

A screenshot of a computer screen

Description automatically generated

### read\_raw\_pixels

hsr.read\_txt.read\_raw\_pixels(file\_path=None, skiprows=2)

reads a [pixel spectrum reading txt file](#_Pixel_spectrum_reading) into a DataFrame that can be used to create a RawDataset

Parameters:

* file\_path(str): the file that will be read, if a directory, the list of files that will be read, if None, the current working directory will be used.
* Skiprows(int): how many rows of metadata are at the top of the file before the column headings.

## Reference Spectrum

The reference spectrum is key to the calibration, as this is the spectrum that the measurements will be calibrated to. There are 2 different valid formats this can come in, a simple list, or a csv file with columns for wavelength and average intensity.

## Simple list

A text file with 801 lines, each line containing a single float. These represent the intensity across the spectrum with the first reading corresponding to 300nm and the last reading corresponding to 1100nm.

A number with black circles

Description automatically generated with medium confidence

This file can be read using the below function

### read\_simple\_file

hsr.spectrum.spectrumUtils.read\_simple\_file(filepath)

reads a simple file with one float per line.

Parameters:

* filepath(str): filepath to the file that will be read.

Returns:

* a numpy array of floats that represents a spectrum values 300-1100.

## CSV reference file

A tab delimited csv file with 2 columns: wavelength and average. Wavelength contains the wavelength of each value, and average contains the intensity of the spectrum at that point.

A number of numbers on a white background

Description automatically generated

This file can be read with the below function:

hsr.spectrum.spectrumUtils.read\_reference\_file(filepath)

### read\_reference\_file

reads a csv file containing a reference spectrum

Parameters:

* filepath(str): filepath to the file that will be read.

Returns:

* a numpy array of floats that represents a spectrum values 300-1100.

## output file

the format that the results will be stored in. contains the new calibrations found by dataset.find\_all\_calibrations(). It a tab delimited cs file with one row for each channel’s calibration, and an average of the above channels. The columns are:

* the words “WavelengthPoly”. This is to match with the existing calibration file, to make it easier to copy and paste into it from this file.
* An integer, that represents the channel name.
* C0, a float, using E to represent scientific notation
* C1, as above
* C2, as above
* C3, as above

Where c0, c1, c2, and c3 represent the polynomial c3x^3 + c2x^2 + c1x + c0

## Wavelength calibration file

The format that the current channels’ calibration is stored in. it is a tab delimeted cvs file with several different rows. The only rows that are used in this context are those starting with WavelengthPoly, the rest are ignored. Those rows are in the same format as the [Output file](#_output_file)

A number and numbers on a white background

Description automatically generated

This file, and an output file can be read using the following function:

### read\_wavelength\_polys

hsr.spectrum.spectrumUtils.read\_reference\_file(filepath)

reads the calibration for each channel from a calibration file or an output file.

Parameters:

* filepath(str): filepath to the file that will be read.

Returns:

* dict that contains the calibrations for each channel. Keys are the channel names in the form: “channel\_n” and values are lists of floats [c0, c1, c2, c3] which represents the polynomial c3x^3 + c2x^2 +c1x + c0

Documentation

## RawDataset

hsr.spectrum.RawDataset(dataset, calibrations=None, calibrations\_filepath=None,

amplitude\_calibration=None,

amplitude\_calibration\_filepath=None,

spectrum\_type=”wavelength”)

Class containing raw data that will get calibrated. Can contain wavelength data or pixel data

Parameters:

* dataset(pandas.DataFrame) the data containing the intensity at each wavelength or pixel position. Columns are “pc\_time\_end\_measurement”, which is the time at which the reading was taken as a String, this can have a value of None on single reading datasets. Each other column is the name of a channel. Index is a simple increasing integer index. Spectra are stored as 1D numpy arrays in the corresponding timestamp and channel
* calibrations(dict): dict containing the existing calibration for each channel, this is used do undo the calibration before finding a new one. Keys are strings of the channel names, values are lists of 4 floats [c0, c1, c2,c3] corresponding to the polynomial c3x^3 + c2x^2 + c1x +c0
* calibrations\_filepath(str): file path to a file containing the existing calibrations, no need to pass this parameter if you have already passed the calibrations above. File format can be found [here](#_Wavelength_calibration_file)
* amplitude\_calibration(numpy.array): array of 801 floats, corresponding to the wavelengths 300-1100. For each spectrum, each value in the spectrum will be multiplied by the value in this array. This will have no impact on the accuracy of the calibration, but it can help the plotted spectra to more closely match the reference spectrum, if one is provided.
* amplitude\_calibration\_filepath(str): file path to a file containing 801 floats, that will be read and stored as a numpy array and used as above. If neither amplitude\_calibration or amplitude\_calibration\_filepath is provided, an array of ones will be used, which has no effect on the spectrum.
* spectrum\_type(str): “wavelength” or “pixel”. The type of spectrum that is being calibrated.

Attributes:

* dataset(pandas.DataFrame): see above
* calibrations(dict): see above
* amplitude\_calibration(numpy.array): see above
* spectrum\_type(str): see above

Methods:

* [find\_all\_calibrations](#_find_all_calibrations)
* [plot\_calibration](#_plot_calibration)
* [plot\_calibration\_wide](#_read_wavelength_polys)

### find\_all\_calibrations

dataset.find\_all\_calibrations(file\_output=None,

reference\_spectrum=None,

reference\_filepath=None,

plot\_total\_output=True,

plot\_individual\_results=False,

plot\_debug=False,

\*\*kwargs)

Finds the correct calibration for each channel and saves and plots the output.

Parameters:

* file\_output(str): for outputting results, which text file to use. leave as None to not output.
* reference\_spectrum(numpy.array): a spectrum in wavelength space that the measured data will be matched to. Consisting of 801 floats. Note: if this reference\_spectrum is from a calibration lamp and has more peaks, pass in the keyword argument find\_peaks=True
* reference\_filepath(str): file path to a file containing a reference spectrum, as above. If neither reference\_spectrum or reference\_filepath is passed, a default spectrum from SMARTS will be used.
* plot\_total\_output(bool): plots the calibration for each channel on one graph.
* plot\_individual\_results(bool): plots the results from each channel, passes plot\_results to each calibration
* plot\_debug(bool): plots the result from each segment of each channel, passed to each calibration
* you can also pass all the [adjustment parameters](#adjustment_parameters) here

Returns:

* new\_calibrations(dict): the new polynomials that will calibrate the spectra. Keys are strings of the channel names, values are lists of 4 floats [c0, c1, c2,c3] corresponding to the polynomial c3x^3 + c2x^2 + c1x +c0

### plot\_calibration

dataset.plot\_calibrations(polys, names=None)

A graph with lines and numbers

Description automatically generated with medium confidence

plots a line graph of pixel against wavelength for each channel. This can be used on the new calibrations or the old ones.

parameters:

* polys(list of lists or dict): the polynomials that will be plotted
* names(list of str): list of names to display on the legend

### plot\_calibration\_wide

dataset.plot\_calibrations(polys, names=None, title=””)

plots a line graph of pixel against wavelength for each channel, over a large range of values. This can be used on the new calibrations or the old ones, and can show how the different calibrations vary, where they might look identical on the smaller scale. 2 vertical lines are plotted to show the range that the normal pixel values cover.

parameters:

* polys(list of lists or dict): the polynomials that will be plotted
* names(list of str): list of names to display on the legend
* title(str): the title of the plot