**How Code Works – 17/11/21**

**Naming Convention and Data Format**

The files, regardless of spectrometer, must have the following naming format to be read by the code.

PG1000\_AuBare\_Air(1)\_\_Comments

This section will be labelled as the Sensor – Anything you want can go in this space aside from underscores.

This section will be extracted and labelled as the solution. The (1) is optional. If you want to use the PCA function which will be explained later, there must be multiple readings of each solution. In this case, the code will look for files such as ‘PG1000\_AuBare\_Air(1)\_\_Comments, ‘PG1000\_AuBare\_Air(2)\_\_Comments’, ‘PG1000\_AuBare\_Air(3)\_\_Comments’ in order to run PCA. If the PCA function is off then this will not matter.

You can put anything you want in this section. This is just somewhere to put any comments you may want to remember when taking a reading. **NOTE** – it is a double underscore between the solution and comments section.

How the data is structured within the ‘run’ folder (see Setup). This will depend on whether you are using the Stellanet or Optosky Spectrometer.

Optosky: Just text files within the folder.

StellaNet: Within the run folder, there must be a folder for each .EP file. These folders must have exactly the same name as the .EP file that is inside them. The .EP files are extracted within these folders.

Graphical user interface, application, table

Description automatically generated Graphical user interface, application

Description automatically generated

**Setup**

I would recommend downloading Anaconda to use Spyder. I have never used any other development environment for python. This is where the code is run from.

Graphical user interface, application, table, Excel

Description automatically generated

You should have your folders set up like this. The code won’t work if it doesn’t have a ‘Figures’ folder and ‘CSVFiles’ folder to export data and figures too.

Once you have set up these folders, you can open the ‘MainCode\_17\_11\_21.py’ file in Spyder. At the top of the script, there will be a bunch of statements beginning with ‘import’ and ‘from’ – ignore these. Below this – you will see the options below.

A screenshot of a computer

Description automatically generated

The **DataSet** section must contain the name of the folder with your files.

The options are turned off by typing ‘True’ for on and ‘False’ for off.

Here is what they all do:

**Spectrawiz\_Select and Optosky\_Select** one of these should be True and one False depending on which spectrometer you are using.

**Smooth and SmoothPoints** – If true, smoothing will be applied to the data with the value in SmoothPoints dictating how many points are averaged in the smooth as defined by:

**OverlaySensors, OverlaySamples and AllPlot** – These options will automatically plot files with the same sensor name (**OverlaySensors**) or sample name (**OverlaySamples**). **AllPlot** is the one I most commonly use and will just plot all the files that have been imported onto the same graph.

Examples:

Graphical user interface, application, table

Description automatically generated

For the above dataset, if OverlaySensors, OverlaySamples and AllPlot are all on, we will end up with a plot of ‘PG1828\_AuGLU’, ‘PG1828\_AuMPA’, ‘PG1828\_AuGLU’,’Water(1)’, ’Water(2)’, ’Water(3)’, ‘Creatine (1)’, ‘Creatine (2)’, ‘Creatine (3)’ and a plot with every line on it.

Chart, line chart

Description automatically generatedChart, histogram

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

These graphs are automatically saved in the subfolder ‘OverlayedPlots’ within the ‘Figures’ folder.

**AllGraphs** – This will plot every averaged graph with the minima labelled and an error given.

Chart

Description automatically generated

These graphs will be automatically saved in the ‘Averaged\_Data\_with\_Minima’ folder. Note – this will create a LOT of graphs if you are running a lot of data. The error is defined by:

**Do\_PCA\_Q** – This will only work if the formatting requirements discussed in the ‘Naming Convention and Data Format’ section have been met. This uses the pandas module to do PCA on the data. I have not had this cross checked with the stats Will does and I mainly use it as a quick check for myself. If you want to use it go ahead. It will give you the graphs below but they will NOT automatically save.

Chart, scatter chart

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

**Minima Finding**

In previous versions of code there were multiple methods of doing this. This is the minima finding I have settled on. With our current, nice peaks – this works just fine. Some steps in this are a bit redundant now but they are legacy steps from asymmetrical fitting when fitting was much harder.

*Preprocessing*

First, the data is trimmed. For Optosky, this means removing the first value which is often 0 or even a large negative number. For Stellanet, this means cutting at 460nm:900nm.

*Minima Finding*

For each reading within a file a minima is calculated. For StellaNet, this is every TRM file within an EP file. For Optosky, this is for each reading with the TXT file generated.

This is done by first, taking the literal minima of the data and cutting around this minima this cut is done by fitting the entire spectra with a high order polynomial and cutting at the second derivates of the polynomial function. Once this cut has taken place, the remaining peak is levelled out as to not bias any following steps (see beautiful image).

Diagram

Description automatically generated

Following this the remaining section of the peak is heavily smoothed (30 point smooth) and the literal minima is taken from this smoothed section.

*Output*

With both stellanet and optosky data, the minima for each peak is calculated as above (i.e. if you have an episodic file) and the averaged minimas are plotted on an average graph of all the peaks with the associated error as seen above. The average minima for each reading is then arranged and exported on the CSV file.

**FWHM Calculations**

In addition to minima fitting the code also spits average out FWHM and height values for every reading. This isn’t particularly well optimised but it gives an idea of the FWHM of each peak

Step 1.

The first 100 wavelengths (excluding the very first wavelength as it is often at intensity 0) have their intensities averaged and this is taken as a baseline. This is not ideal as sometimes this section of the graph can be sloping but it is at least consistent

Step 2.

The literal minima is taken and the difference between the baseline and the minima is taken as the height. From this, we can find the half height and then simply find the difference in wavelength values of the two points that cross this line to get the FWHM.

The FWHM is also exported onto the CSV file along with the height of the peak as given with this method.

**Troubleshooting**

Some common issues – by far the most common issue is that a sample contains a name that can be found within another name. For example ‘PG1000\_AuBare\_Ethanol1\_\_Comments’ and ‘PG1000\_AuBare\_Ethanol10\_\_Comments’ will throw up an error or give unexpected outputs. This is because when the code searches for ‘Ethanol1’ it will find it in both strings. This is the most important thing to keep an eye out for as it may not give an error – just incorrect results.

Another error that will halt the code is if the code cannot find a minima for one of your readings. Maybe this could happen if you were making trimers and they weren’t giving you any minima at all 😊