

# OCD Analysis Software (v 1.0)

Nick Charron - Huang Biophysics Lab

Rice University. 2017.

This software is designed to plot and process spectra obtained from the J-810 Jasco CD spectrometer in the Huang lab. All measurement files must be in ASCII format (\*.txt files). This format can be chosen when saving the measured spectra in the SPECTRA MANAGER software provided by Jasco. File reading routines are currently optimized for J-810 ASCII output, though these can be modified to read any consistent text-based spectral data file.

The software is designed to work as a collection of tools, so that it may be used interactively. To start an interactive python session, type the following command at your shell or command prompt:

```
>_ python
Python 3.6.3 (default, Oct  4 2017, 06:09:15)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.37)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>_
```

Depending on your distribution and machine details, the output above may not be exactly the same (This output was generated on a 2015 Apple MPB with a brew install of python 3.6). Next, the tools must be imported:

```
>>> from ocd import *
```

This also imports the following python modules: numpy as np, and matplotlib.pyplot as plt. Loading a spectrum is done through the creation of an instance of the class ocd\_spec. Providing a filename loads the corresponding spectral information (wavelength and CD signal). Optionally the class may be called without a filename to create an empty spectrum (no spectral information):

```
>>> myspec = ocd_spec("scan_01.txt") #class with filename
>>> emptyspec = ocd_spec()           #empty class
```

You can also use file dialogs to select the file from your current directory. On Windows systems, this functionality is supplied by the wxPython-Phoenix package, which interfaces with the system default file manager and pipes the selected filename to the creation of an ocd\_spec instance. On Linux and OSX systems, the same process is achieved through the use of pycurses and its lightweight text user interface.

```
>>> myspec = fs()
```

Once a spectrum is loaded, you have access to the following functions and variables:

**ocd\_spec.wl** : returns a 1D numpy array of wavelengths used in the scan.

**ocd\_spec.cd** : returns a 1D numpy array of CD signals measured.

**ocd\_spec.load(string)** : loads wl and cd data from specified filename.

**ocd\_spec.name** : returns the name of the spectrum.

**ocd\_spec.graph()** : produces a simple graph of the spectrum.

**ocd\_spec.renorm(float)** : globally rescales the CD signal by specified factor.

**ocd\_spec.rm\_baseline(float)** : returns an instance of `ocd_spec` with a constant baseline subtracted from the CD signal.

**ocd\_spec.trim(wl1, wl2)** : returns an instance of `ocd_spec` trimmed to the wavelength range specified by `wl1` and `wl2`.

**ocd\_spec.filter()** : returns an instance of `ocd_spec` that has a Kaiser-filtered CD signal.

**avg\_signal( [spec1, spec2, ...] )** : returns an `ocd_spec` class with an average CD signal from `spec1, spec2, ...`. The spectra must be passed to this function as a python list.

**mult\_graph(specs,types,colors,title)** : produces a graph of multiple spectra with custom style options. Specs, types, and colors must all be lists of the same size. A legend will automatically be generated using the values of `spec.name`.

**graph\_series(specs,title,cmap)** : produces a graph of a sequence of experiments - useful for showing spectral changes dependent on concentration. The color map can be chosen from the matplotlib package, though the default is `Reds()`. A legend will automatically be generated using the values of `spec.name`.

In this way, multiple spectra can be graphed, rescaled, filtered, corrected, and analyzed interactively at the user's discretion. Custom graphs beyond the scope of the simple **mult\_graph()** function (advanced color schemes, plotstyles, line types, etc.) can be made by directly calling functions from the `pyplot` module:

```
>>> fig = plt.figure("Custom Plot")
>>> plt.plot(myspec.wl, myspec.cd, 'b--',linewidth=2)
>>> plt.xlim(220,240)
>>> plt.title("230 nm Peak - A State")
>>> plt.ylabel("CD [mdeg]")
>>> plt.xlabel("Wavelength [nm]")
>>> plt.show()
```