On the Organization of the HSG Analysis Python Library

1 Introduction

This will give a brief foray into the dark underbelly of hsganalysis.py. The file itself will contain all of the variable information in the docstrings. This will go deeper into how it actually functions, what things to call when.

2 The classes

There are three super classes: CCD, PMT, and Spectrum. CCD currently handles any spectrum data from the EMCCD. The PMT class handles data from the PMT detector. Spectrum combines CCD and PMT classes to make whole spectra.

2.1 CCD class

As covered previously, the CCD class handles data taken with the Andor EMCCD. The general processing guide is to initialize the CCD instance, guess the sideband locations and determine their strengths, fit those sidebands to gaussians (or any arbitrary function), and finally to save the processed spectrum and the fit results.

2.1.1 Attributes

2.1.2 Methods

There are many methods in the class, only some of which need to be called by the user.
Magic methods:

1. `__init__(fname)` - Initializes the CCD object.
    - Inputs: `fname` - The whole path to the spectrum file. This file contains a dictionary with all of the important experimental descriptions and parameters.
    - Internals:
        (a) `self.fname` - the path that the imported file came from.
        (b) `self.parameters` - dictionary with all of the important experimental details
        (c) `self.description` - string with a brief description of the spectrum
        (d) `self.raw_data` - np.array of size 1600×2. The first column is in nm, the second column has the raw counts from the CCD.
        (e) `self.hsg_data` - np.array of size 1600×2. The first column is in eV, the second column contains the counts normalized by the number of FEL pulses.
        (f) `self.dark_stdev` - standard deviation of the dark counts as calculated during the experiment divided by the number of FEL counts.
        (g) `self.addenda` - a list of the things added to this object to make it what it is, including constant offsets and other spectra.
        (h) `self.subtrahenda` - a list of spectra subtracted from this object.

- Outputs: `None`

2. `__add__` - Adds a number (float or int) or another spectrum. It's unclear at the moment whether this should be `raw_data` or `hsg_data`. It seems to work currently, though, because of an extra method, `image_normalize`.
   - Inputs: `other` - A number or another spectrum. Currently adds `hsg_data` if other is a CCD object.
   - Internals:
     (a) `ret.addenda` - the two addendas are concatenated.
     (b) `ret.subtrahenda` - the two subtrahendas are concatenated.
     (c) `ret.parameters['fel_pulses']` - these are added together as well.
   - Outputs: a new CCD object that is the sum of self and other.
3. `__sub__` - Subtracts a number (float or int) or another spectrum. It's unclear at the moment whether this should be `raw_data` or `hsg_data`. It seems to work currently, though, because of an extra method, `image_normalize`.
   - Inputs: `other` - A number or another spectrum. Currently subtracts `hsg_data` if other is a CCD object.
   - Internals:
     (a) `ret.addenda` - the old addenda is concatenated with the new subtrahenda.
     (b) `ret.subtrahenda` - the old subtrahenda is concatenated with the new addenda.
   - Outputs: a new CCD object that is the sum of self and other.
4. `__str__` - prints the `description` of the file.
   - Inputs: nothing
   - Internals: nothing
   - Outputs: `self.description`.

Normal methods:

1. `add_std_error` -
   - Inputs:
   - Internals:
   - Outputs:
2. `calc_approx_sb_order` -
   - Inputs:
   - Internals:
   - Outputs:
3. `image_normalize` -

- Inputs:
- Internals:
- Outputs:

4. `guess_better` -
    - Inputs:
    - Internals:
    - Outputs:
5. `guess_sidebands` -
    - Inputs:
    - Internals:
    - Outputs:
6. `fit_sidebands` -
    - Inputs:
    - Internals:
    - Outputs:
7. `fit_sidebands_for_NIR_freq` -
    - Inputs:
    - Internals:
    - Outputs:
8. `save_processing` -
    - Inputs:
    - Internals:
    - Outputs:
9. `stitch_spectra` - This does nothing.

2.2 PMT class

This class imports and handles data from the PMT.

2.2.1 Attributes

2.2.2 Methods

Magic methods:

1. `__init__` - Initializes the PMT object. Requires all of the tiny files to be in the same folder
   - Inputs: `folder_path` - the complete path of the folder that contains all of the tiny PMT data files.
   - Internals:
     (a)
   - Outputs:

Normal methods:

1. `fit_sidebands` -
   - Inputs:
   - Internals:
   - Outputs:
2. `save_processing` -
   - Inputs:
   - Internals:
   - Outputs:

2.3 Spectrum class

This class takes PMT and CCD class data and combines them.

2.3.1 Attributes

2.3.2 Methods

Magic methods:

1. `__init__` - Initializes the Spectrum object. At the moment it requires a PMT and a CCD object. It would be cool if it could handle two CCD objects (hint hint).
   - Inputs:
     (a) `PMT_spectrum`
     (b) `CCD_spectrum`
   - Internals:
     (a) `self.pmt_results`
     (b) `self.ccd_results`
     (c) `self.parameters`

(d) `self.pmt_dict`
(e) `self.full_dict`

- Outputs: `None`

Normal methods:

1. `plot_prep` - This makes an np.array, `self.full_results`, that is identical to the `sb_results` np.array in other objects.
   - Inputs: Nothing
   - Internals: `self.full_results`
   - Outputs: `None`
2. `add_sidebands` - This method takes another CCD object and adds `sb_results` to the current `self.full_results` attribute in the dumbest way possible.
   - Inputs: CCD_spectrum - a CCD object that is part of the same spectrum making up this object.
   - Internals:
     (a) `self.ccd2_results` - copied from `CCD_spectrum.sb_results`.
     (b) `self.ccd2_dict` - keys are sb order, values are the results from before.
     (c) `self.full_dict` - gets stitched with `self.ccd2_dict` to make a complete thing. No averaging, no nothing.
   - Outputs: `None`
3. `save_processing` - This saves things like the other save-y things.
   - Inputs:
     (a) `file_name` - the base of the file name that will be saved.
     (b) `folder_str` - the folder that the file will be saved in. It will create the folder if necessary.
     (c) `marker=''` - an extra piece of the file name to be saved. Can be a parameter or something.
     (d) `index=''` - indexer that has to be external because numpy's saving system sucks.
   - Internals:
     (a) `spectra_fname` - `file_name + '_' + marker + '_' + str(index) + '.txt'`
     (b) `fit_fname` - `file_name + '_' + marker + '_' + str(index) + '_fits.txt'`
     (c) `parameter_str` - the json string created from `self.parameters`
     (d) `origin_import_fits` - header created to be read by Origin.
   - Outputs: `None`