CLS-LIME – SOP

Setup

1. Have XRF data (timed line scans .dat files) in a subfolder named “XRF”. Have Echem files (.txt file exported from ECLab) in subfolder named “Echem”.
   1. Echem file should be exported WITHOUT a header, with “Elapsed time” box checked and all columns included.
2. Download latest version of python. I use version 3.10, but the most recent versions should work.

General

1. There is a single class object, which contains multiple functions (functions start with “def” and have a list of arguments defined).
2. “self” just means it is referencing the class object. It is used liberally in my code but ideally you should have multiple class objects.
3. The class object begins with a function called “\_\_init\_\_”. This is where I feed all the important parameters. When adapting the script to a new file, you will need to change many values here.
   1. Timestart/end: Select time (in minutes) that you want to plot. This is based on the Echem start time.
4. The functions will run in order that they are called. This is done at the bottom of the script. The order there should be correct. For the functions that you don’t want to run, put a “#” in front of it which will make the compiler ignore that line of code.

Script

1. Get\_echem\_files
   1. Imports echem file from a .txt file
   2. Finds the import columns (time, voltage, current etc.)
   3. Converts units as necessary (time to minutes etc.)
   4. Can work on multiple files. If there are multiple files, it will concatenate them and the time will be continuous and relative to the first file start time.
      1. If your files start with non-zero time and that information is important, and you don’t want to start at 0 mins, you need to take out this line of code:
      2. self.dfEchem['time/s'] = self.dfEchem['time/s'] - self.dfEchem['time/s'][0] #Normalize to first timepoint
   5. If it works successfully, it will print “Upload Echem Complete”. Errors occur if another text file got in the folder by accident or if the parsing line bugs out.
2. Plot\_echem
   1. Will plot the entire voltage vs capacity plot. Good way to make sure it imported things properly. Also useful for echem figures.
   2. Errors will occur if it tries to find a column that doesn’t exist. Make sure the column name changes made in “get\_echem\_files” corresponds to the column names you are calling.
   3. If no data pops up (blank), try printing df\_Echem which holds all the echem data. It could have not imported properly for some reason. If not, make sure that the range of current, potential and time you choose are appropriate. You can zoom in and out to “find” your data if this happens.
3. Get\_xrf\_times (NOT IMPORTANT UNLESS YOU HAVE MULTIPLE XRF FILES)
   1. This will import your timelinedscan file ONLY TO GET THE TIMESTAMP
   2. It starts by finding the timestamp for the file, and converting that to an amount of minutes
   3. Next files will have a start timestamp relative to the first file
4. Concatenate\_xrf
   1. This will import your xrf file (or files)
   2. If there are multiple timedlinescans, it will concatenate them and plot them all together relative to the start time of the first one. Here it will also adjust the time based on the Echem start time delay that you set manually.
      1. You set this time difference in \_\_init\_\_
      2. If echem was started 1 min before XRF, self.startotalminutesechem = 1
5. Plot\_XRF
   1. Important params in \_\_init\_\_
      1. Time slice is based on self.timestart and self.timeend
      2. Self.Steps must be equal to the number of steps in the scan. Check XRF file for this value
      3. Self.stepdistance will give you distances in microns
      4. Self.reduce changes the number of profiles to plot. 2 plots every 2nd curve, 3 plots every 3rd curve etc etc
      5. Self.plotstart and self.plotstop determine what positions to plot
      6. Self.numberofaverages will take n number of curves and average them for one point calib. If you only have one good curve at OCV, this will unfortunately need to be 1. The more, the better.
      7. Self.calibrationconcentration concentration of electrolyte
   2. Will make the classic concentration profile plot ([Li+] vs position)
   3. Will do the concentration calibration either from the beginning of the time slice OR from time = 0
      1. averagey = self.one\_point\_calibration()  
         #averagey = self.one\_point\_calibration\_from\_start()
      2. Choose one ^
6. Heatmap
   1. Important parameters in \_\_init\_\_
      1. Same as for plot XRF
      2. Time
   2. Calibration to OCV is done separately here
      1. #####Do a 1-point calibration for the heatmap#####  
         for i in range(0, len(heatmapDF)):   
          averageAtOCV = numpy.average(heatmapDF.iloc[i,0:0]) #0:x is the curves to be used for normalization
   3. Vmin and vmax control lower and upper concentration bounds