Table of Contents

plateReader

Processes CSV files produced by the Medusa2012 program for 64 pad combi plates. An example file called “plateFile.csv” can be found in the shared Code folder.

1. From the Anaconda3 folder, open the python.exe application.
2. In the shell, enter the command “from DataAnalysis.plateReader import *”.
3. Enter the CSV file name without the .csv extension.
    - e.g. For the file “plateFile.csv”, just enter “plateFile”.
    - A CSV file in the “Data” folder on the desktop can be accessed through the file location: “C:\Users\McCalla Lab\Desktop\Data\plateFile”.
4. Enter the name of the file containing the masses of each sample on each channel. An example file showing the format of the mass file can be found in the shared Code folder called “massFile.csv”. Note that any other text in the file will be ignored and that the masses for channels 1-64 should be in columns C-BM respectively of row 2. The masses are also located in the same location as the masses in the exported file so the same file can be used for masses as the exported data.
    - The mass file needs to contain the masses in the same location as the masses in the exported data CSV file.
    - If no mass file is available, you can click “Enter” to pass the command without entering a file name. Masses will be requested as needed for each channel.
5. To use one of the available methods, type the method name with the desired parameters in the brackets and press Enter.
    - e.g. The method “capacity(channel, cycle)” can be used for channel 1 during cycle 2 to calculate charge and discharge capacities by entering “capacity(1, 2)”.
6. Methods:
    - newFile()
        - Load data from a new file.
    - setMass(channel, fileNum = 1)
        - Change the sample mass used on the input channel.
    - export(fileNum = 1)
        - Exports the masses, charge and discharge capacities, and resistances that have been calculated to a CSV file. An example export file can be found in the shared Code folder as “plateReader export.csv”.
    - phaseDiagramExport(fileNum = 1)
        - Takes a properly formatted phase diagram file (eg. phaseDiagramFile.csv) and exports the capacity values of each channel during each cycle to the same file. The phase diagram data must contain 64 rows of phase data with Channel 1 in row 2 and Channel 64 in row 65 and will throw an error otherwise.
    - capacity(channel, cycle = 1, fileNum = 1)
        - Calculate the capacity (mAh/g) of one channel during one cycle.
    - capacity64(cycle = 1, fileNum = 1)
        - Calculate the capacity (mAh/g) of all 64 channels during one cycle.
    - capacityAllCycles(channel, fileNum = 1)
        - Calculate the capacity (mAh/g) for all the cycles of a single channel.
    - allCapacities(fileNum = 1)

    - Calculates the charge and discharge capacities (mAh/g) for all the cycles of every channel
- averageVoltages(startTime, endTime, fileNum = 1)
    - Calculate the average voltage (V) for every channel between the desired start and end times.
- resistances(cycle = 1, fileNum = 1)
    - Calculate the resistances (Ω) of each channel during the specified cycle with standard deviation and variance.
- allResistances(fileNum = 1)
    - Calculates the resistance (Ω) on every channel during each cycle with the mean, standard deviation and variance of resistances in each cycle
- plotCurrentVsVolts(channel, cycle = 1, xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot current (μA) vs voltage (V) for the specified channel and cycle between the minimum and maximum voltage values (vMin and vMax).
- plotCurrentVsVolts64(cycle = 1, xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot current (μA) vs voltage (V) for all 64 channels during one cycle between the minimum and maximum voltage values (vMin and vMax).
- plotAllCyclesCurrentVsVolts(channel, xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot current (μA) vs voltage (V) for one channel during every cycle between the minimum and maximum voltage values (vMin and vMax).
- plotAllCyclesCurrentVsVolts64(xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot current (μA) vs voltage (V) for all 64 channels during every cycle between the minimum and maximum voltage values (vMin and vMax).
- plotAllFilesCurrentVsVolts(channel, cycle = 1, xMin = 3.1, xMax = 4.5)
    - Plot current (μA) vs voltage (V) for the specified channel and cycle for every loaded file between the minimum and maximum voltage values (vMin and vMax).
- plotAllFilesCurrentVsVolts64(cycle = 1, xMin = 3.1, xMax = 4.5)
    - Plot current (μA) vs voltage (V) for all 64 channels on every loaded file during one cycle between the minimum and maximum voltage values (vMin and vMax).
- plotAllFilesAllCyclesCurrentVsVolts(channel, xMin = 3.1, xMax = 4.5)
    - Plot current (μA) vs voltage (V) for one channel during every cycle of every loaded file between the minimum and maximum voltage values (vMin and vMax).
- plotAllFilesAllCyclesCurrentVsVolts64(xMin = 3.1, xMax = 4.5)
    - Plot current (μA) vs voltage (V) for all 64 channels of every loaded file during every cycle between the minimum and maximum voltage values (vMin and vMax).
- plotNormalizedCurrentVsVolts(channel, cycle = 1, xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot normalized current (mA/g) vs voltage (V) for the input channel and cycle between the minimum and maximum voltage values (vMin and vMax) for the input file.
- plotNormalizedCurrentVsVolts64(cycle = 1, xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot normalized current (mA/g) vs voltage (V) for all 64 channels for the input cycle between the minimum and maximum voltages for the input file number.
- plotAllCyclesNormalizedCurrentVsVolts(channel, xMin = 3.1, xMax = 4.5, fileNum = 1)

    - Plot normalized current (mA/g) vs voltage (V) for the input channel between the minimum and maximum voltage values (vMin and vMax) furing every cycle for the input file.
- plotAllCyclesNormalizedCurrentVsVolts64(xMin = 3.1, xMax = 4.5, fileNum = 1)
    - Plot normalized current (mA/g) vs voltage (V) for all 64 channels between the minimum and maximum voltage values (vMin and vMax) for every cycle of the input file.
- plotAllFilesNormalizedCurrentVsVolts(channel, cycle = 1, xMin = 3.1, xMax = 4.5)
    - Plot normalized current (mA/g) vs voltage (V) for the input channel and cycle between the minimum and maximum voltage values (vMin and vMax) for every loaded file.
- plotAllFilesNormalizedCurrentVsVolts64(cycle = 1, xMin = 3.1, xMax = 4.5)
    - Plot normalized current (mA/g) vs voltage (V) for all 64 channels for the input cycle between the minimum and maximum voltages for every loaded file.
- plotAllFilesAllCyclesNormalizedCurrentVsVolts(channel, xMin = 3.1, xMax = 4.5)
    - Plot normalized current (mA/g) vs voltage (V) for the input channel between the minimum and maximum voltage values (vMin and vMax) furing every cycle for every loaded file.
- plotAllFilesAllCyclesNormalizedCurrentVsVolts64(xMin = 3.1, xMax = 4.5)
    - Plot normalized current (mA/g) vs voltage (V) for all 64 channels between the minimum and maximum voltage values (vMin and vMax) for every cycle of every loaded file.
- plotVoltsVsCurrent(channel, cycle = 1, fileNum = 1)
    - Plot voltage (V) vs current (μA) for a single channel during one cycle.
- plotVoltsVsCurrent64(cycle = 1, fileNum = 1)
    - Plot voltage (V) vs current (μA) for all 64 channels during one cycle.
- plotAllCyclesVoltsVsCurrent(channel, fileNum = 1)
    - Plot voltage (V) vs current (μA) for one channel during every cycle.
- plotAllCyclesVoltsVsCurrent64(fileNum = 1)
    - Plot voltage (V) vs current (μA) for all 64 channels during every cycle.
- plotVoltsVsTime(cycle = 1, fileNum = 1)
    - Plot voltage (V) vs time (hr) for one cycle.
- plotAllCyclesVoltsVsTime(fileNum = 1)
    - Plot voltage (V) vs time (hr) for every cycle.
- plotCurrentVsTime(channel, cycle = 1, fileNum = 1)
    - Plot current (μA) vs time (hr) for one channel during one cycle.
- plotAllCyclesCurrentVsTime(channel, fileNum = 1)
    - Plot current (μA) vs time (hr) for all cycles on one channel.
- plotVoltsVsCapacity(channel, cycle = 1, fileNum = 1)
    - Plot voltage (V) vs charge and discharge capacity (mAh/g) for one channel during one cycle.
- plotCapacityVsCycle(channel, fileNum = 1)
    - Plot the charge and discharge capacity (mAh/g) of the input channel vs cycle number.
- plotChargeCapacityVsCycle(channel, fileNum = 1)

        - Plot the charge capacity (mAh/g) for one channel for every cycle.
    - plotDischargeCapacityVsCycle(channel, fileNum = 1)
        - Plot the discharge capacity (mAh/g) for one channel for every cycle.

cellReader

Processes CSV files produced by Neware for Swagelok cells. An example cell file can be found in the shared Code folder called “cellFile.csv”.

1. From the Anaconda3 folder, open the python.exe application.
2. In the shell, enter the command “from DataAnalysis.cellReader import *”.
3. Enter the CSV file name without the .csv extension.
    - e.g. For the file “cellFile.csv”, just enter “cellFile”.
    - A CSV file in the “Data” folder on the desktop can be accessed through the file location: “C:\Users\McCalla Lab\Desktop\Data\cellFile”.
4. To use one of the available methods, type the method name with the desired parameters in the brackets and press Enter.
    - e.g. The method “plotVoltsVsCapacity(cycle, fileNum = 1)” can be used to plot Voltage vs Capacity for cycle 2 of file 1 by entering “plotVoltsVsCapacity(2)”.
    - File 1 is the default file number. To plot another file, enter the file number in the method call.
        - “plotVoltsVsCapacity(1, 2)” will plot the data for cycle 1 of file 2.
5. Methods:
    - newFile()
        - Load data from a new file.
    - clear()
        - Clear all stored data.
    - setMass(fileNum = 1)
        - Changes the mass stored for the input file number (fileNum) then recalculates the specific capacities and energies of the cell with the new sample mass
    - cellInfo(fileNum = 1)
        - Print the charge and discharge capacities (mAh), specific charge and discharge capacities (mAh/g), charge and discharge energies (Wh), and specific charge and discharge energies (Wh/g) of the input file number (fileNum).
    - plotVoltsVsCapacity(cycle , fileNum = 1)
        - Plots the voltage (V) vs specific capacity (mAh/g) for the input file number during the input cycle number.
    - plotAllCyclesVoltsVsCapacity(fileNum = 1)
        - Plots voltage (V) vs specific capacity (mAh/g) for every cycle of the input file number.
    - plotConnectedVoltsVsCapacity(fileNum = 1)
        - Plots voltage (V) vs specific capacity (mAh/g) for every cycle of the input file number as a continuous plot where each charge capacity is shifted positive by the last discharge capacity in the previous cycle and each discharge capacity is shifted positive by the last charge capacity in the previous cycle.
    - plotVoltsVsCapacityCycleWindow(cycle1, cycle2, fileNum = 1)

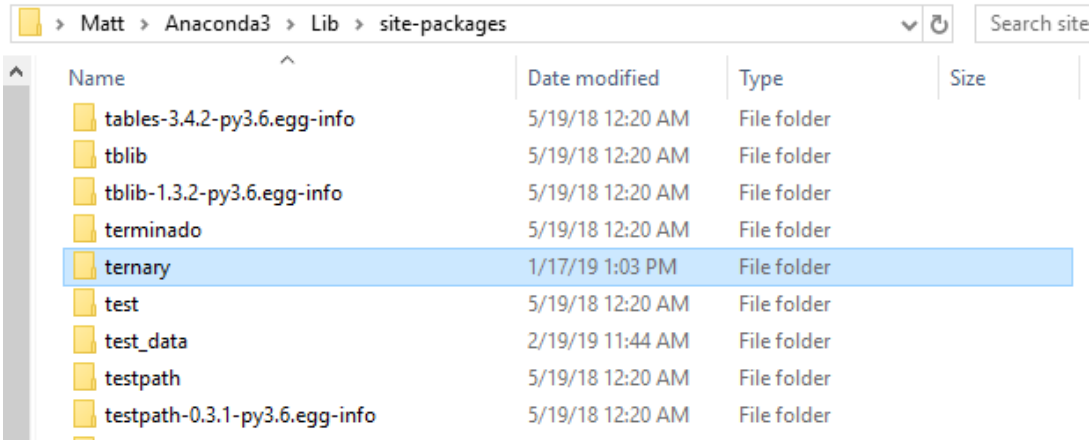
    - Plots voltage (V) vs capacity (mAh/g) between the first input cycle (cycle1) and the second input cycle (cycle2) for the designated file number (fileNum), with each cycle at a different color.
- plotAllFilesVoltsVsCapacity(cycle = 1)
    - Plots the voltage (V) vs the specific capacity (mAh/g) of all the files that have been loaded for the specified cycle, with each file as a different color.
- plotVoltageWindowCycleWindow(vMin, vMax, cycle1, cycle2, fileNum = 1)
    - Plots voltage (V) vs capacity (mAh/g) between the first cycle (cycle1) and the second cycle (cycle2) for voltages between the minimum (vMin) and maximum (vMax) voltage, with each cycle as a different color.
- plotVoltageWindow(vMin, vMax, cycle = 1, fileNum = 1)
    - Plots the voltage (V) vs the specific capacity (mAh/g) of the specified fileNum for the input cycle number between the minimum (vMin) and maximum (vMax) voltages.
- plotAllCyclesVoltageWindow(vMin, vMax, fileNum = 1)
    - Plots voltage (V) vs capacity (mAh/g) for the voltages between the minimum (vMin) and maximum voltages (vMax) for all cycles.
- plotAllFilesVoltageWindow(vMin, vMax, cycle = 1)
    - Plots the voltage (V) vs the specific capacity (mAh/g) for the input cycle number between the minimum (vMin) and maximum (vMax) voltages, with each file as a different color.
- plotdQVsVolts(boxcar = 5, cycle = 1 , fileNum = 1)
    - Plots the derivative of specific capacity (mAh/Vg) as a function of voltage for the input cycle of the input file with the input boxcar average.
- plotdQCycleWindow(cycle1, cycle2, boxcar = 5, fileNum = 1)
    - Plots the derivative of specific capacity (mAh/Vg) between the first input cycle (cycle1) and second input cycle (cycle2) for the specified file number (fileNum) with the input boxcar average.
- plotAllCyclesdQVsVolts(boxcar = 5, fileNum = 1)
    - Plots the derivative of specific capacity (mAh/Vg) for every cycle of the input file number with the input boxcar average.
- plotAllFilesdQVsVolts(boxcar = 5, cycle = 1)
    - Plots the derivative of specific capacity (mAh/Vg) for the input cycle for every loaded file, with each file as a different color with the input boxcar average.
- plotCapacityVsCycle(fileNum = 1, legend = False)
    - Plots specific charge and discharge capacities (mAh/g) vs cycle number for the input file number.
- plotChargeCapacityVsCycle(fileNum = 1)
    - Plots specific charge capacity (mAh/g) vs cycle number for the input file number.
- plotDischargeCapacityVsCycle(fileNum = 1)
    - Plots specific discharge capacity (mAh/g) vs cycle number for the input file number.
- plotAllFilesCapacityVsCycle(legend = False)
    - Plots specific charge and discharge capacities (mAh/g) for every loaded file vs cycle number, with each file as a different color.
- plotCapacityVsCycleWindow(cycle1, cycle2, fileNum = 1, legend = False)

    - Plots capacity (mAh/g) vs the cycle number for the cycles between cycle1 and cycle2 inclusive.
- plotAllFilesCapacityVsCycleWindow(cycle1, cycle2, legend = False)
    - Plots capacity (mAh/g) vs the cycle number for the cycles between cycle1 and cycle2 inclusive for all loaded files.
- plotEnergyVsCycle(fileNum = 1)
    - Plots specific charge and discharge energies (Wh/g) vs cycle number for the input file number.
- plotChargeEnergyVsCycle(fileNum = 1)
    - Plots specific charge energy (Wh/g) vs cycle number for the input file number.
- plotDischargeEnergyVsCycle(fileNum = 1)
    - Plots specific discharge energy (Wh/g) vs cycle number for the input file number.
- plotAllFilesEnergyVsCycle()
    - Plots specific charge and discharge energies (Wh/g) for every loaded file vs cycle number, with each file as a different color.
- plotCapacityVsRate()
    - Plots the first charge and discharge capacity (mAh/g) of the input cells vs the current rate (mA/g). This function can only plot 7 different rates at a time so cells must be input in sets of 7. A filler cell called “blank.csv” can be found in the “Code” folder if you do not have a full set of 7 cells to input into the program. This can take up to 10 sets of 7 cells. The last set of 7 cells should all contain the desired rates to be plotted on the x-axis. Each set of 7 is plotted in a different color.
- plotDischargeVsRate()
    - Plots the first 10 discharge capacities (mAh/g) of the input cells vs the current rate used (mA/g). If a file has less than 10 discharge cycles, the missing data will be filled by zeros. This function can only plot 7 different rates at a time so cells must be input in sets of 7. A filler cell called “blank.csv” can be found in the “Code” folder if you do not have a full set of 7 cells to input into the program. This can take up to 10 sets of 7 cells. The last set of 7 cells should all contain the desired rates to be plotted on the x-axis. Each set of 7 is plotted as a different color.
- plot(fileNum = 1)
    - Runs plotConnectedVoltsVsCapacity(fileNum), plotCapacityVsCycle(fileNum), and plotAllFilesCapacityVsCycle().
- overlay(boxcar = 5, cycle = 1)
    - Runs plotAllFilesVoltsVsCapacity(cycle) and plotAllFilesdQVsVolts(boxcar, cycle).
- rate()
    - Runs plotCapacityVsRate() and plotDischargeVsRate().

phaseDiagram

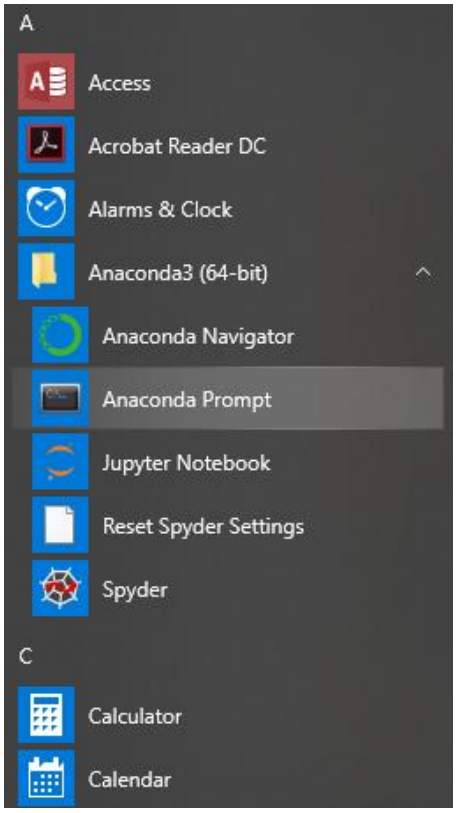
Processes ternary phase data to generate a ternary phase diagram. An example of the phase diagram file format can be found in the shared Code folder called “phaseDiagramFile.csv”. Note the axis labels are read from the column labels of the input CSV file.

To run this program, the python-ternary library must be installed in Anaconda3. If this has been installed, a folder called “ternary” can be found in “Ananconda3\Lib\site-packages”.

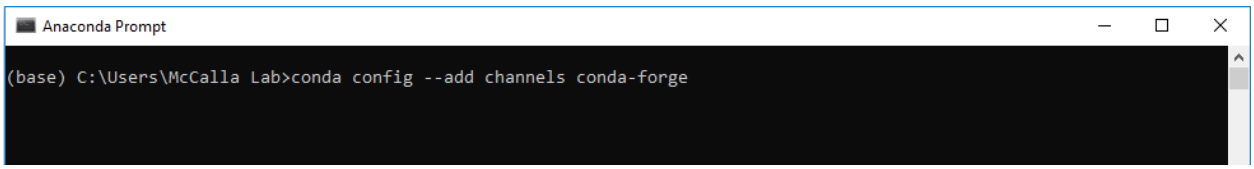


Otherwise, the python-ternary library can be installed through the following steps:

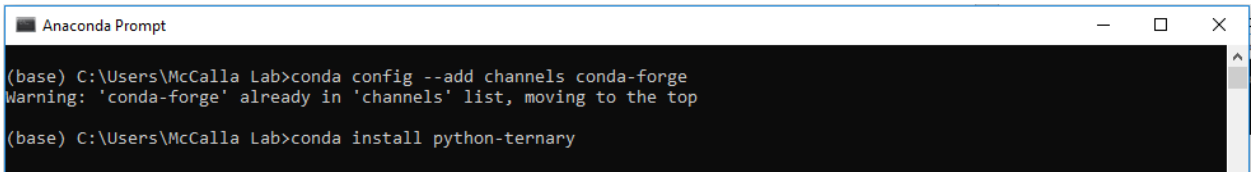
1. In the Start Menu, open the Anaconda Prompt found in the Anaconda3 folder.



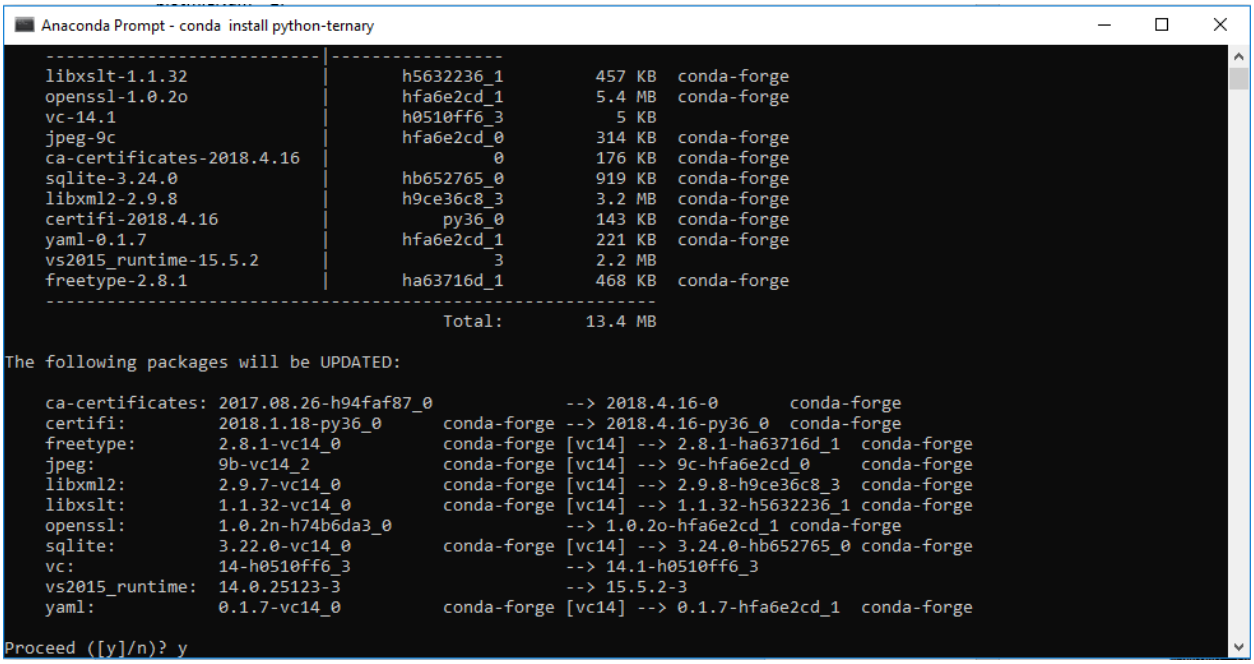
2. In the Anaconda Prompt, type “conda config --add channels conda-forge” then press Enter.



3. In the next line, type “conda install python-ternary” then press Enter.



4. When prompted to proceed, type “y” then press Enter.



5. Once the program finishes running and the “(base) C:\Users\McCalla Lab>” prompt reappears, the phaseDiagram program is ready to be used.

To use the phaseDiagram program:

1. From the Anaconda3 folder, open the python.exe application.
2. In the shell, enter the command “from DataAnalysis.phaseDiagram import *”.
3. Call newFile(dataColumn = ‘A’, dataLabel = ‘’) then enter the CSV file name without the .csv extension.
    - e.g. For the file “csvexample.csv”, just enter “csvexample”.
    - A CSV file in the “Data” folder on the desktop can be accessed through the file location: “C:\Users\McCalla Lab\Desktop\Data\csvexample”.
4. To use one of the available methods, type the method name and press Enter.
    - e.g. Type “plotDiagram()” to generate the ternary phase diagram.
5. Methods:
    - newFile(dataColumn = ‘A’, dataLabel = ‘’)
        - Clears all previously stored data then loads data from the new input file. dataColumn represents the column of data in the CSV file that will be used to generate a color bar. dataColumn values can be upper- or lower-case letters corresponding to the

column of data as seen in excel or can be an integer value for the column number, with 1 corresponding to the first column. dataColumn values A-C, a-c, or 0-3 will not generate any color bar data as those columns contain the phase data. Letter values should be surrounded by apostrophes (eg. 'A' or 'a') and integers should have no surrounding characters. dataLabel will change the label shown for the color bar from the column header to the input dataLabel value. Alternatively, the column headers in the CSV file can be changed to the desired value, removing the need of inputting a dataLabel value. Any dataLabel label should also be surrounded by apostrophes.

- plotDiagram()
  - Uses data from the input files to generate the ternary phase diagram.

pxrd

pxrd plots the PXRD and refinement data exported from Rietica and converted into a CSV file or just the x ad y data contained in columns 1 and 2 of CSV file. An example of the CSV file format is in the shared Code folder, called "xrdFile.csv".

1. From the Anaconda3 folder, open the python.exe application.
2. In the shell, enter the command "from DataAnalysis.pxrd import *"
3. Enter the CSV file name without the .csv extension.
   - e.g. For the file "xrdFile.csv", just enter "xrdFile"
   - A CSV file in the "Data" folder on the desktop can be accessed through the file location: "C:\Users\McCalla Lab\Desktop\Data\xrdFile".
4. To use one of the available methods, type the method name and press Enter.
   - e.g. Type "plotXRD()" to plot the input XRD data.
5. Methods:
   - newFile()
     - Import new PXRD data from the file path. To import all PXRD data from a data path, do not enter a file name and just click enter. All the files in the data path must match the correct PXRD CSV file format for the data to be imported correctly.
   - clear()
     - Clear all imported file data
   - plotXRD(fileNum = 1)
     - Plot the PXRD data for fileNum with the fit if the data was exported from Rietica or just the spectrum if the file was converted to a CSV from the raw data. The default file number is 1 if no fileNum is entered.
   - plotXRDError(fileNum = 1)
     - Plot the PXRD for fileNum with the fit and difference of the data and the fit. Only exported files from Rietica can be plotted.
   - plotAllXRD()
     - Plot all the loaded PXRD data as a vertically stacked plot.

Changing the File Path for All Programs

If you enter the file path where all data will be stored, it is not necessary to enter the path when prompted to enter the file name. If the path is correct, you will only need the name of the desired file to be processed. i.e. Ignore step 3 bullet point 2 for the plateReader, cellReader, phaseDiagram, and pxrd instructions.

To avoid entering any file paths, you can save a copy of the CSV file you want processed in the Anaconda3 folder.

1. In Anaconda3, open the DataAnalysis folder.
2. Open cellReader.py, plateReader.py, phaseDiagram.py, or pxrd.py in Notepad or WordPad.
3. On line 6 of cellReader.py, plateReader.py, pxrd.py and of phaseDiagram.py, enter the file path in the empty brackets of “filepath = '' ”
    - cellReader location

```
import csv
from itertools import islice
import matplotlib.pyplot as plt
import numpy as np

filepath = ''

# Empty global lists to store the cycle numbers, voltages, currents, capacities,
# energies, specific capacities and energies, and charge and discharge capacities,
# specific capacities, energies, specific energies, and capacitances for each cycles
# in chronological order
```

    - plateReader location

```
import csv
from itertools import islice
import matplotlib.pyplot as plt
import numpy as np

filepath = ''

# Empty global lists to store the cycle numbers, times, and voltages in chronological order and
# charge and discharge capacities and resistances
cycles = []
times = []
voltages = []
chgCapacities = []
dchgCapacities = []
resistance = []
meanResistances = []
stdevResistances = []
```

    - phaseDiagram location

```
import csv
import ternary
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

filepath = ''

# Empty global lists to store the fractions of each of the three elements and their respective capacities.
# The element names are then stored in the global xName, yName, and zName strings
xVal = []
yVal = []
zVal = []
capacities = []
xName = ''
yName = ''
zName = ''
```

- pxrd location

```
import csv
import os
from itertools import islice
import matplotlib.pyplot as plt

filepath = ''

# Lists to store the x and y data to be fit
xData = []
yData = []
yFit= []
numFiles = 0
```

4. After entering the path of your data folder, the line should look like this: "filepath = 'C:\\Users\\McCalla Lab\\Desktop\\Data\\' " for files stored in a "Data" folder on the desktop
    - ***Important*** Make sure there are two backslashes between subsequent folders in the file location, as shown in the above example. Backslashes are interpreted as an escape character by Python so a single backslash will not result in the correct file path and the program will not be able to process your file.
    - Any data you want processed should be in the folder you entered as the file path. Entering another file path in the terminal to access a file in a different folder when running the program will not process your file.
    - To process "csvexample.csv" in the "Data" folder, enter "csvexample" when prompted for the file name.

```
import csv
from itertools import islice
import matplotlib.pyplot as plt
import numpy as np

filepath = 'C:\\Users\\McCalla Lab\\Desktop\\Data\\'

# Empty global lists to store the cycle numbers, voltages, currents, capacities,
# energies, specific capacities and energies, and charge and discharge capacities,
# specific capacities, energies, specific energies, and capacitances for each cycles
# in chronological order
```

5. Save the file and close the text editor.
6. cellReader, plateReader, phaseDiagram, or pxrd can now be ready to be used as normal from the python.exe terminal.

Anaconda3

https://www.anaconda.com/distribution/#download-section

- Provides terminal used to run scripts
- Contains math and plotting libraries required for functions in scripts
- Note: python-ternary library needs to be added separately to Anaconda3, following instructions listed previously

Github

https://github.com/matthewburigana/DataAnalysis

- Used for version control
- Contains history of all new versions of scripts since January 2019