Lancaster University

GUIDE BOOK

Entropy profiling

Table of contents

1. Introduction

The entropy profiling for the study of Lithium Ion batteries is used in several experiments all over the world. It exists different ways to proceed the measurements. In this hand book, one of the experimental sets up possible is presented. It is made by Michael Mercer and it is based on the research of Osswald and Al. in the article “Fast and accurate measurements of entropy profiles of commercial Lithium-ion cells”.

It concerns cycling and temperature depended open circuit voltage (OCV) measurements of cylindrical or graphite half-cells (also called “entropy measurements”).

In this handbook, the principles of the entropy profiling, the realisation of a test plan in Basytec software, and treatment of the data are explained.

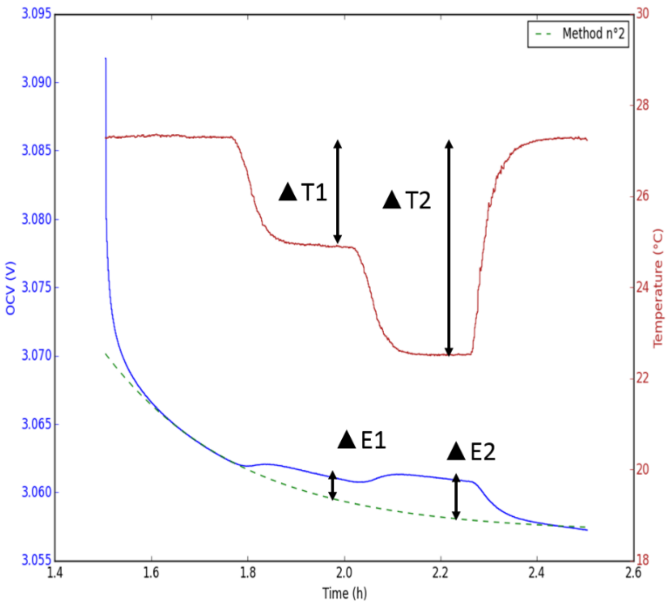
2. Principles of the entropy profiling and methodology

The entropy profiling is based on the dependence of the entropy coefficient on the variation of OCV and temperature, as it is underlined by the formula below (calculation based on the Gibbs energy):

$$\left(\frac{\partial E_{EMF}(x)}{\partial T}\right)_{p,x} = -\frac{1}{nF}\left(\frac{\partial}{\partial x}S(x)\right)_{p,x} = -\frac{1}{nF}\Delta_r S(x)$$

Therefore, the purpose of the experiment is to impose temperature variation to the battery for each state of charge (SOC) or depth of discharge (DOD) and measure the OCV. The temperature is controlled by the means of a thermal bath in which the cells are plunged. The high resolution OCV is measured with high voltage resolution setup.

The plot below shows the OCV evolution during one state of charge, more precisely during the relaxation part after imposing the charge/discharge current (OCV measured in blue and temperature in red):

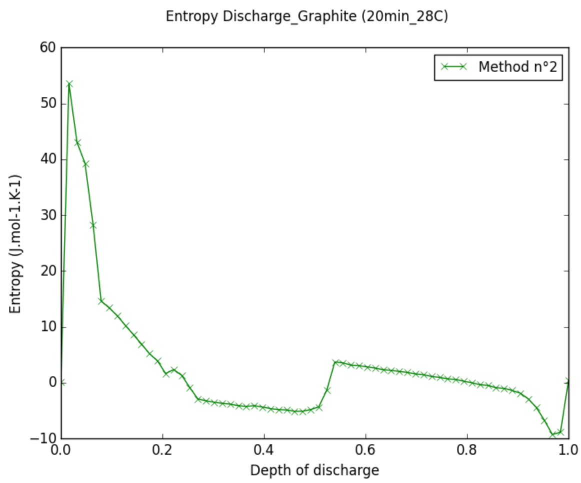


It is possible to impose different temperature levels, which induce a voltage variation (blue curve). This voltage variation can be obtained by the difference of the measured OCV (blue) and the extrapolated curve (green). This last curve represents the voltage if there was not any temperature variation. The methodology used is explained the last part of this handbook.

You can notice that only 2 temperatures levels are necessary to get an entropy coefficient. However, by increasing this number of temperature levels, the accuracy of the entropy value increases since the final entropy coefficient is the average from all the entropy coefficient resulting from each temperature variation. For the SOC n° i , the entropy value is (p: number of temperature levels):

$$S_i = \frac{1}{p-1}\sum_{j=1}^{p-1} S_j \quad = \quad \frac{1}{p-1}\sum_{j=1}^{p-1} F\,.\frac{\Delta E_j}{\Delta T_j}$$

Once the different entropy coefficient for each SOC are calculated, the resulting entropy profiling (below discharge profile of a graphite half-cell) is:



3. Experimental plan

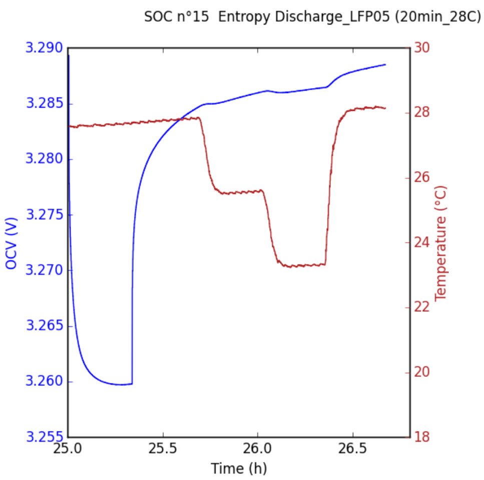
You can find below one kind of plan for the Basytec cycler. For the first SOC, you do not need to start it with a phase of charge and discharge, but then each SOC have discharge/charge phase where the imposed current depends on the time you want between each temperature level and the number of SOCs.

$$I = \frac{Nominal\ Capacity}{x} \qquad x = \frac{N_{SOC} \cdot t_{step}}{60}$$

You will notice that it is very important to write a line in the test for each temperature level. Indeed, the code for the data processing to get entropic coefficients is based on those line changes.

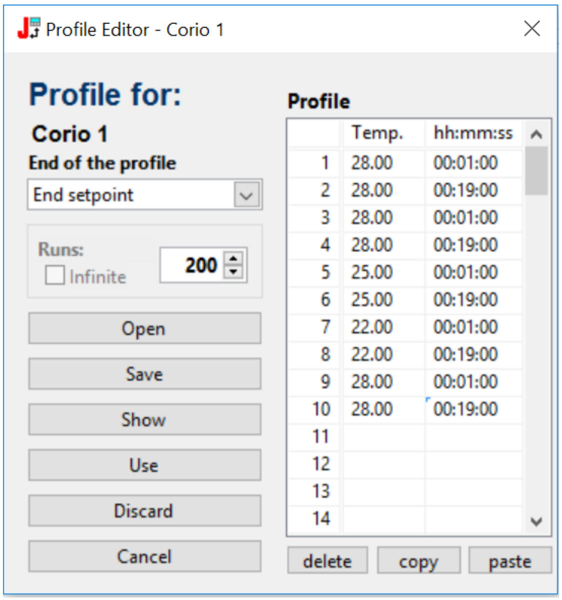
	Level	Label	Command	Parameter	Termination	Action	Registration	Comment
1			Start		U>1UBatMax&t>1s U<1UBatMin&t>1s I>1IBatMax&t>1s I<1IBatMin&t>1s			
2			Pause		t>15s			
3			pause		t>20min		t=1s	C/10 30 steps
4			pause		t>20min		t=1s	28 C
5			pause		t>20min		t=1s	25 C
6			pause		t>20min		t=1s	22 C
7			pause		t>20min		t=1s	28 C
8			Cycle-start		U<2.5V			
9			discharge	I=150mA	t>20min		t=1s	C/10 30 steps
10			pause		t>20min		t=1s	28 C
11			pause		t>20min		t=1s	25 C
12			pause		t>20min		t=1s	22 C
13			pause		t>20min		t=1s	28 C
14			Cycle-end	Count=50				
15			Stop					

Basytec test plan (step time of 20 min)



One loop of the cycle

You can find below the test plan of the thermal bath.

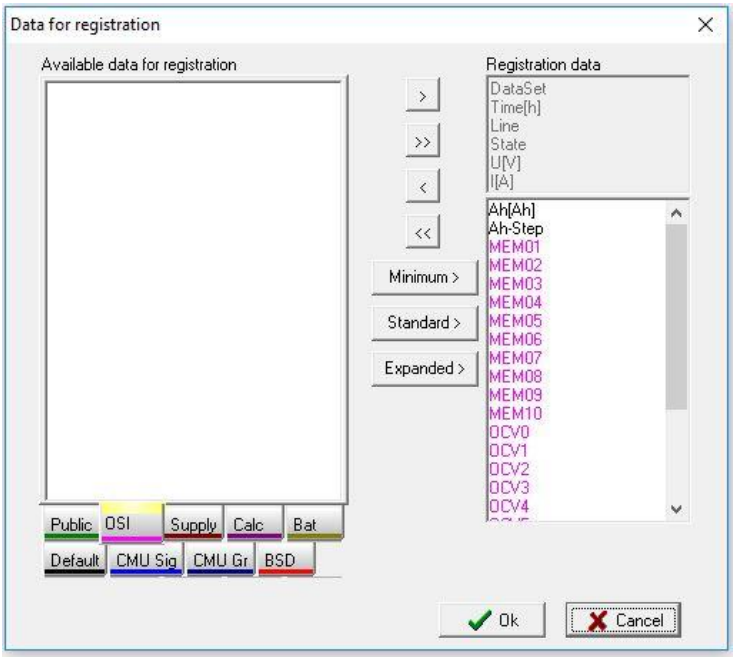


Thermal bath test plan (step time of 20 min)

4. Conduct an experiment

4.1. Prepare the experiment

⚠ For each plan you create, the registration format must be checked. When the plan tab is opened click on *Registration format* and double click on the different names of the keysight channels (MEM and OCV) the OSI Available data to make them visible in the data table and save them.



4.2. Start the experiment

First launch the Basytec test plan. Then you can start the bath test plan.

⚠ The two plans must be synchronized. It is impossible to click on the start button in the same time. So, it is recommended to add an additional time (15 seconds for example) in the basytec test plan at the beginning (cf Basytec test plan above) to have the time to start the temperature plan.

5. Data extraction and data treatments

5.1. Extract the data

Once the experiment is done, you just have to extract the data from the plan database in a text file or CSV file (with a parameter: coma between values).

5.2. Python code interface

To treat the data, you can use the delivered python code. This code is composed of 3 files:

- *Class_ method*: all the classes and their methods
- *Database*: database of batteries, channel, group of battery, group of experiences
- *Main*: workspace

You can find below, the structure of the object-oriented code:

```
class Battery
    def __init__
        attr name
        attr nominal_capacity
        attr mass
        attr Hioki_R
        attr RPT_file
        attr impedance_file
    def RPT_plot
    def RPT_capacity
    def get_impedance
class Battery_group
    def __init__
        attr battery_list
    def capacity_list_std
    def Discharge_cap_plot
    def Discharge_cap_weight_plot
    def get_mean_impedance
    def Nyquist_impedance_plot
    def comparaison_impedance_plot
class Channel
    def __init__
        attr name
        attr thermo
        attr OCV
class Experiment_group
    def __init__
        attr experiment_list
        attr experiment_type
    def temperature_plot
    def entropy_plot
```

```
class Experiment
    def __init__
        attr name
        attr experiment_type
        attr setup
        attr battery
        attr channel
        attr time_step
        attr number_temperature_level
        attr temp_ref
        attr Tsteps
        attr df_basytec
        attr df_basytec
        attr title
        attr title
        attr SOC_relax_list
        attr df_entropy_data
    def max_capacity
    def DataFrame_T_expected
    def OCV_temperature_plot
    def bestfit_entropy_matlab_plot
    def method_entropy_matlab_plot
    def rawdata_entropy_matlab_plot
    def entropy_coefficient
        ## Block 1 : Constant and parameters
        ## Block 2 : Conversion mV to V, °C to kelvin, for the bassytec file
        ## Block 3 : Split the data in different SOC (each SOC a dataFrame)
        ## Block 4 :Split the data of each SOC to keep only the relaxation part and save
        ## Block 5 :For each SOC relax...
        ## Block 5.1: Voltage reference
        ## Block 5.2: Capacity reference
        ## Block 5.3: Get the temperature levels, voltage levels_rawdata, delta_tempe
        ## Block 5.4 : Data extraction for the fitting methods
        ## Block 5.5 : Voltage fitting and get deltaE=voltage_rawdata-estimated volt_c
        def func1
        def func3
        ## Block 5.6 : Get MSE (Mean Square error) for each method and SOC
        ## Block 5.7: Get delta_E  for specific soc and temperature level, and for each n
        ## Block 5.8: Get the entropy=-nF*delta_E/delta_T for specific soc, method and
        ## Block 5.9 : Entropy = mean(entropy_levels)  and enthalpy
        ## Block 5.10: Select the best fit for the SOC
        ## Update of the SOC_relax_list with all the entropy data
        ## Block 6: CSV export
        ## Return: SOC_relax_list,df_entropy_data
    def entropy_plot
    def SOC_plot
    def SOC_relax_fit_plot
    def enthalpy_plot
```

This code includes the data treatment for a Reference Performant Test, which gives all the characteristics of a commercial cell and allows the comparison between the constructor's datasheet and the actual data.

Here the procedure of execution, all the details of the constructors and methods are in the documentation calling *help()*:

1) Execute *Class_ method*
2) In *Main*:

- Create the battery or copy and paste those ones you need from the *Database*
- If RPT procedure: you can create a group of battery, execute and call the methods you want
- If high voltage resolution experiment /entropy experiment:
    - Create the Channel of the experiment or copy and paste from the *Database*

- create the Experiment or copy and paste from the *Database*. You have to precise the path of the txt file with all the extracted data from Basytec software, in the constructor.
- Execute and in the process of construction, thanks to the *Experience.entropy_coefficient()* method all the entropy data will be saved as an attribute of the experiment and in a CSV file. All the SOCs data will be saved as an attribute of the experiment (list of dataFrame) and in different CSV files.
- Call the method you want to get the different plots

5.3. Experience.entropy_coefficient() method

This function is the heart of the code isolating Isolate the SOCs and calculate the entropy coefficient. Its documentation is:

Return

SOC_relax_list : list of dataFrame
List of of dataFrame where a dataFrame gathers all the data from a state of charge during the relaxation part (including the estimated voltage data from the fitting
and the voltage difference between the estimation and the raw data
df_entropy_data: dataFrame
DataFrame containing all the data from the entropy profiling (Capacity, voltage reference, best fitting method, entropy coefficient etc)

Save

SOC_relax_list : CSV
save the different SOCs in different CSV files
df_entropy_data: CSV
save all the data from the entropy profiling in a CSV file

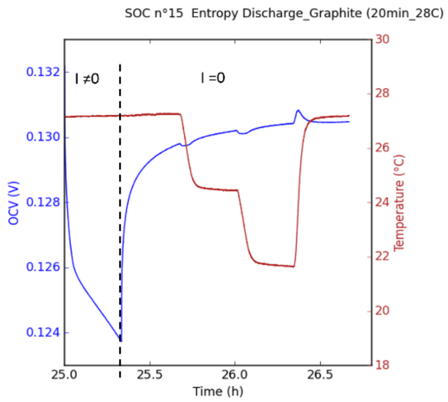
In the code, the notations can be different but, in this guide, they are:

- i : SOC_relax n°i
- k: temperature_level n°k
- p: number of temperature levels
- n: number of SOC
- m: method n°m
- l : number of method

You will find below different parts of the code explained:

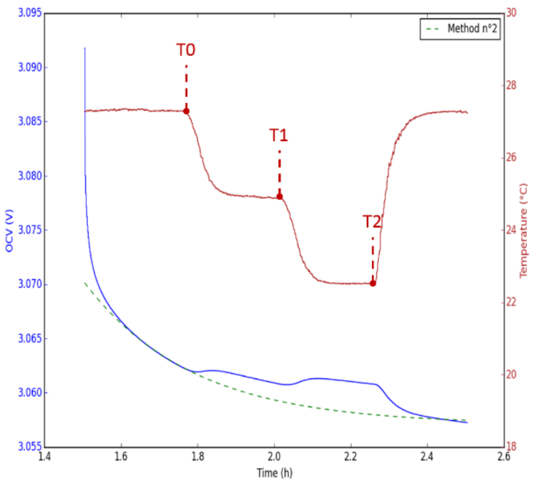
- **Block 1: Constant and parameters**
- **Block 2: Conversion mV to V and °C to kelvin**
- **Block 3: Split the data in different SOC (each SOC a dataFrame)**

  A SOC includes the part where the current is different from 0.



- **Block 4: Split the data of each SOC to keep only the relaxation part and save the indexes of the different temperature levels**

  The relaxation part corresponds to the part where I=0 A.
  Those indexes of temperature levels correspond to the last points before the temperature changes in the SOC. They are put forward in the graph below:

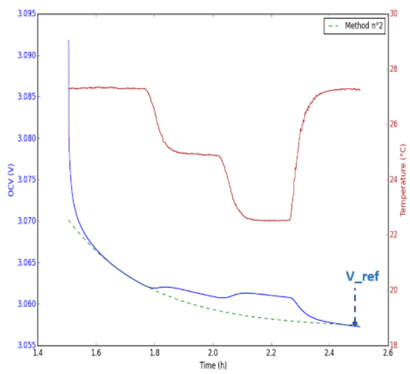


$$SOC\ temp\ index\ list = [\ temp\ index_0\ , \dots , temp\ index_i\ , \dots\ temp\ index_{n-1}]$$

The steps of the blocks 5. are in a loop, and they are executed for each SOC n°i

- **Block 5.1: Voltage reference**

For each SOC, a voltage reference is saved and can be used for the enthalpy and the graphs.



$$SOC\ OCV\ reference = [\ OCV\ ref_0\ , \dots\ OCV\ ref_i\ \ , \dots\ OCV\ ref_{n-1}]$$

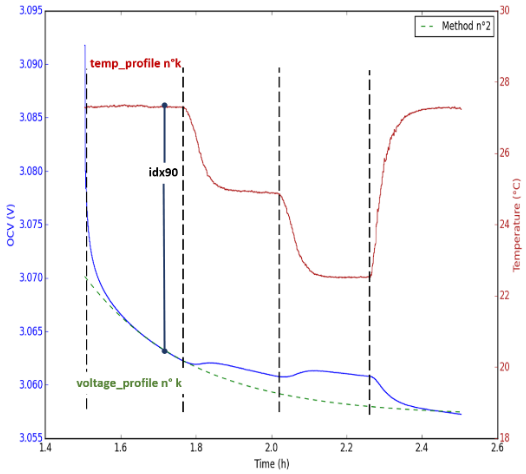
- **Block 5.2: Capacity reference**

Like for the OCV_reference, we keep a capacity reference that can be used to calculate the value of the SOC for the graphs. (same reference point like OCV)

$$SOC\ capacity\ reference = [capacity\ ref_0\ , \dots\ capacity\ ref_i\ \ , \dots\ capacity\ ref_{n-1}]$$

- **Block 5.3: Get the temperature levels, voltage levels_raw data, delta_temperature**

For the temperature levels n°k, a voltage and a temperature are extracted and saved as a reference for this temperature level. Those two points are the mean value of 6 points around the 90% of the profile n°k.



$$temperature\ levels_i = [\ T_{i,0}, \dots T_{i,k}\ \ , \dots\ T_{i,p-1}]$$

$$voltage\ levels\ rawdata\ _i = [\ E_{i,0}, \dots E_{i,k}\ \ , \dots\ E_{i,p-1}]$$

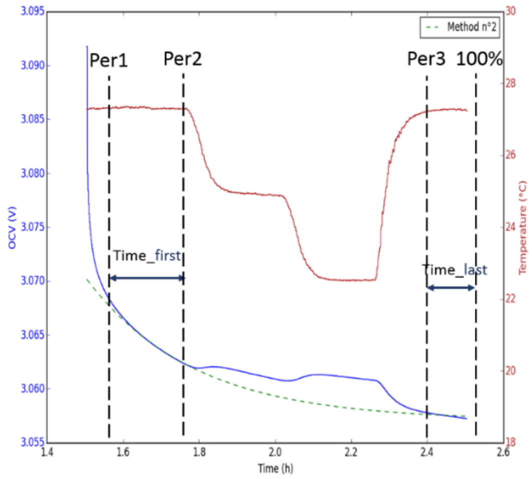
$$delta\ temperature_i = [\ \Delta T_{i,0}, \dots \Delta T_{i,k}\ \ , \dots\ \Delta T_{i,p-1}]\ \text{ with } \Delta T_{i,k} = T_{i,0} - T_{i,k}$$

- **Block 5.4: Data extraction for the fitting methods**

In this block, the voltage array and the time array used in the fitting methods are created. As explained before the fitting method are used to create the curve which represents the voltage if there was not any temperature variation during the SOC.

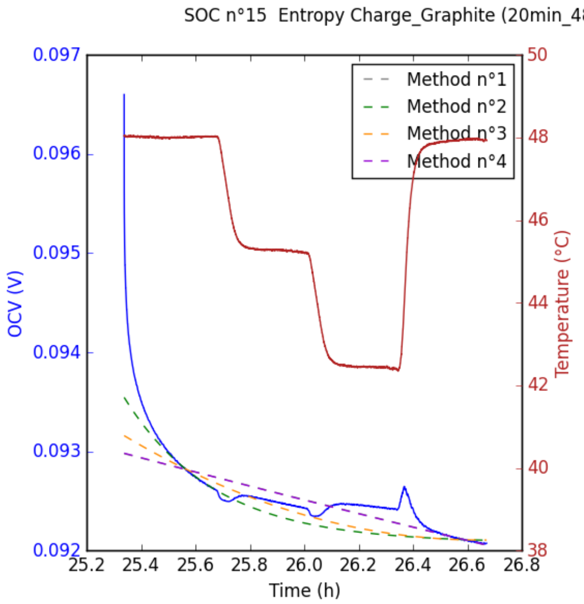
First, two portions of the OCV (raw data) and the time are extracted and then concatenated.

1) between per1% and per2% of the time and voltage of the first part of SOC_relax where temperature=temperature_reference
2) -between per3% and 100% of the time and voltage of the last part of SOC_relax where temperature=temperature_reference



- ❑ **Block 5.5: Voltage fitting and get ΔE for each fitting method**

In the actual code, 4 functions are used for the fitting, for each of them the best fitting coefficient are extracted and saved in *coef_fit_method_m*. Then those coefficient are used extrapolated the voltage curve as if there was not any temperature variation during the SOC.



Method n°1: $E_{est} = a\ln(t) + b$

Method n°2: $E_{est} = a\,e^{-bt} + c$

Method n°3: $E_{est} = a\ln(t)^2 + b\ln(t) + c$

Method n°4: $E_{est} = \frac{a+t}{b+t} + c$

- ❑ **Block 5.6: Get the MSE (Mean Square error) for each method and SOC**

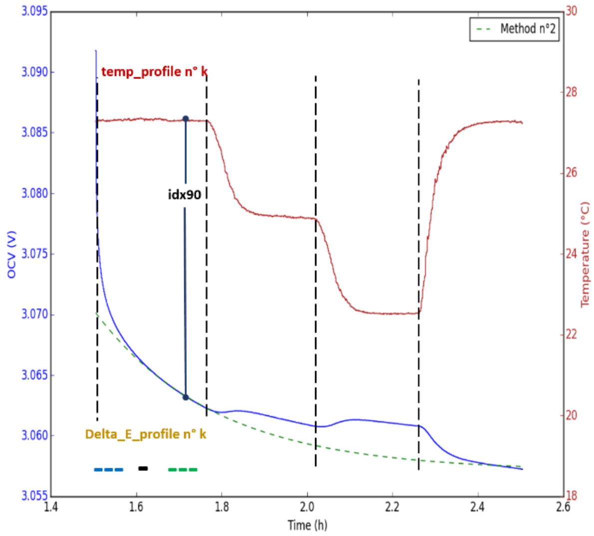
The MSE is the criteria to select the best fitting for one SOC, so it must calculate for each method. The residuals are calculate only over 3 parts of the time (between 40 % and 90% of each temperature levels k)

$$MSE = \frac{Total\ sum\ residual}{Number\ data\ points - number\ parameter\ fitting\ method}$$

$$Total\ sum\ residual = \sum_{k=0}^{p-1} \sum \Delta E^2$$

- **Block 5.7: Get ΔE for specific SOC and temperature levels for each method**

The same methodology is used for ΔE and ΔT:



$\Delta E\ levels_{\ i,m}\ = [\ \Delta E_{\ i,m,0}, \dots \Delta E_{i,m,k}\ \ , \dots\ \Delta E_{i,m,p-1}]$ with $\Delta E_{i,m,k} = E_{i,k} - E_{est,i,m,k}$

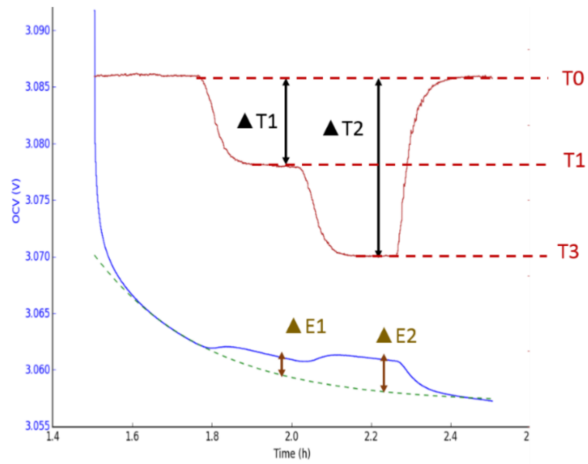
$\Delta E\ levels_{\ i}\ = [\ \Delta E\ levels_{\ i,1}, \dots \Delta E\ levels_{\ i,m}\ \ , \dots\ \Delta E\ levels_{\ i,4}]$

$delta\ temperature_{\ i}\ = [\ \Delta T_{i,0}, \dots \Delta T_{i,k}\ \ , \dots\ \Delta T_{i,p-1}]$ with $\Delta T_{i,k} = T_{i,0} - T_{i,k}$

- **Block 5.8: Get the entropy for specific soc, temperature level and method**

$\Delta S\ levels_{\ i,m}\ = [\ \Delta S_{\ i,m,1}, \dots \Delta S_{i,m,k}\ \ , \dots\ \Delta S_{i,m,p-1}]$ with $\Delta S_{\ i,m,k}\ = \ -n\ F\ .\frac{\Delta E_{i,m,k}}{\Delta T_{i,k}}$

$\Delta S\ levels_{\ i}\ = [\ \Delta S\ levels_{\ i,1}, \dots \Delta S\ levels_{\ i,m,k}\ \ , \dots\ \Delta S\ levels_{\ i,4}]$



- **Block 5.9: Get the mean entropy for each method and SOC**

$$\Delta S_{i,m} = \frac{1}{p-1}\sum_{k=1}^{p-1} \Delta S_{\ i,m,k} \quad = \quad \frac{1}{p-1}\sum_{j=1}^{p-1} - n\ F\ .\frac{\Delta E_{i,m,k}}{\Delta T_{i,k}}$$

The error for the entropy comes from the standard deviation on the average

- **Block 5.10: Select the best fit for each SOC**

The criteria of the best fit is the MSE, the method with the lowest MSE is chosen. Nevertheless, the method is not reliable since the estimated can be very closed to the OCV but not the best estimation of what the curve should be without temperature changes.

- **Block 6: CSV export**

You can find this code in Github