Supercapacitor testing procedure and data analysis

Implemented testing procedure makes possible supercapacitor testing and analysis according to the norm *Electric double-layer capacitors for use in hybrid electric vehicles – Test methods for electrical characteristics (IEC 62576:2009; EN 62576:2010)*. Number of adjustable parameters makes possible other methods of analysis, e.g. better equivalent serial resistance determination in highly nonlinear discharge curve by using 3rd order polynomial approximation.

Procedures are implemented both in Matlab and Python. There are minor differences in syntax and resultant images:

	Matlab	**Python**
Multiple data groups	array: […] cell array: { … }	tuple: (…)
Resultant graphs	Stored as Matlab figure (.fig) and Enhanced Windows Metafile (.emf)	Stored as high resolution PNG image (.png)
Data input function	Defined in separate M-file, specified as "@function" in parameter list	Defined in "readFunctions.py", specified as string argument containing function name
Toolboxes / libraries	/	numpy, matplotlib

Testing requirements

Supercapacitor should be charged to nominal voltage, preferably by constant current. It is advisable to apply constant nominal voltage until charging current decrease and reach steady value. Discharge procedure should be as follows:

1. Rest for 5s (set testing device to open circuit condition, i.e. without producing any voltage or current)
2. Apply discharge current and keep it until desired discharge voltage is reached (usually 50% of nominal voltage)

Record time and voltage with at least ten samples per second.

For the analysis procedure data should be stored in ASCII (text) file with two header rows and data in two tab or space delimited columns, first for time and second for voltage. Decimal separator should be dot ('.'). There is no file naming restrictions, except that care should be taken to ensure that sorting order corresponds to the cycle's order, as described in function parameter description.

Parameter *dataFunct* provides data input function, making possible accommodating procedure to differently formatted input data.

Function call

```
SCanalysis(I, Ur, m, S, 'text1', 'text2', 'text3', 'text4', 'text5', files, dataFunct,  [Ur1 Ur2], [Uc1 Uc2], pf1,pf2);
```

Parameters:

Ur[V]	Nominal voltage (to which SC is charged)
I [mA]	Discharge current a) Scalar (e.g. 21.5) Analysis for single current is performed b) Vector (e.g. [10 17.8 31.6 56.2 100]) Performs multi cycle – multi current analysis.
m[mg]	Mass of active material. Should be zero if not used.
S[cm^2]	Area of one electrode. Should be zero if not used.
text1..text5	Five comment lines that will be printed at plots
files	File name filter. Files that match the filter are sorted according to the file name. Each file should contain data for one discharge cycle. Care should be taken to ensure that sorting order corresponds to the cycles order, e.g. name files as '17_04_25_Data_001.txt' … '17_04_25_Data_100.txt'. In this case 'fnames' field should be set to *Data*.txt, where '*' matches any number of characters. If vector of discharge currents is defined, *files* field should be defined as cell array, having same number of elements as the *I* vector (and optionally one additional element), where each element defines file name filter for corresponding current measurements, e.g.: {'I_15mA*.txt' 'I_22mA*.txt'} If additional file is defined, it should contain repeated measurement for the first current. Data from that file is shown with red cross at plots.
dataFunct	Handle of function for reading data from file. Function should accept two parameters, file name and discharge current. If data contains measured current, discharge current could be discarded. Function should return N-by-3 array, where first column contains *time* in seconds, second *voltage* in volts and third *current* in amperes.
[Ur1 Ur2]	Resistance calculation start and end voltage. Voltage is specified as a percentage of nominal voltage Ur, e.g. [0.9 0.7]. Straight-line approximation is applied to the discharge curve between Ur1 and Ur2. Equivalent serial resistance is calculated from the voltage

	drop at the discharge start time, which is the determined from the value of the straight line at the discharge start time. Special cases: [1 *t*] – Ur1 is voltage drop at first sample after the discharge current was applied; Ur2 is voltage *t* seconds later. 3rd order polynomial approximation is used to determine the voltage drop at the discharge start time.
[Uc1 Uc2]	Capacitance calculation start and end voltage. Voltage is specified as a percentage of nominal voltage Ur, e.g. [0.9 0.7]. Capacitance is calculated by energy conversion method: $C = \frac{2W}{{U_{C1}}^2 - {U_{C2}}^2}, \qquad W = I \int_{t1}^{t2} u(t)dt$ *t1* and *t2* are times where voltage is *Uc1* and *Uc2* respectively. Special cases: [1 Uc2] – Uc1 is voltage drop at first sample after the discharge current was applied. [1 0] – Uc1 is voltage drop at first sample after the discharge current was applied, Uc2 is voltage of last recorded discharge curve sample.
pf1	Plot Frequency. Discharge curve plot will be generated each *pF1* cycles.
pf2	Plot Frequency. Cumulative discharge curve plot will be generated and discharge curve will be plotted each pF2 cycles.

Stored values (Matlab implementation only):

All stored values are matrices having 'number of cycles' rows and 'number of file filters' columns. They are stored into the 'Results.mat' file:

C	Capacitance
Cs_m	Specific capacitance per mass
Cs_a	Specific capacitance per area
R	Equivalent serial resistance
Rd	Discharge resistance, calculated from self-discharge curve during initial 5s rest period: $R_d = -\frac{t}{C \ln(U_r/U_t)}$
nl	Discharge curve nonlinearity, calculated as mean square deviation from the ideal (linear) discharge curve.
Pd	Specific power density, per mass.
E	Specific energy, per mass.

Also number of plots are stored, both in extended windows metafile format ('emf') and matlab figure format ('fig').

Examples

Single data set analysis

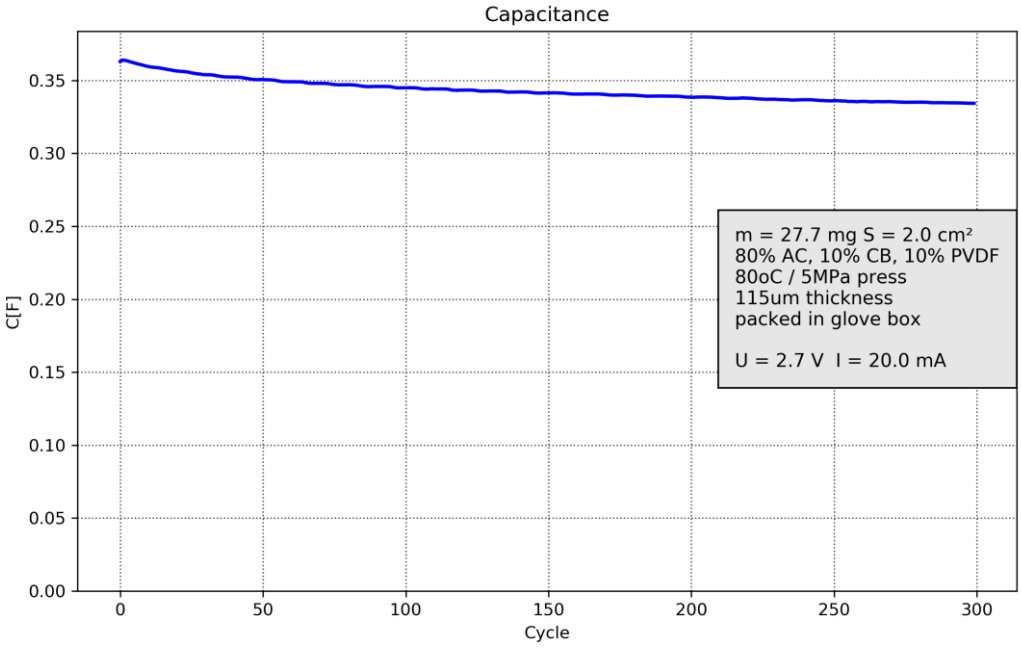
```
SCanalysis(2.7,20,27.7,2,'80% AC, 10% CB, 10% PVDF', '80oC / 5MPa press',
      '115um thickness', 'packed in glove box', '', '*KRONO2*.oxw',
      @readAL, [0.9,0.7], [0.9,0.7], 10, 50);
```

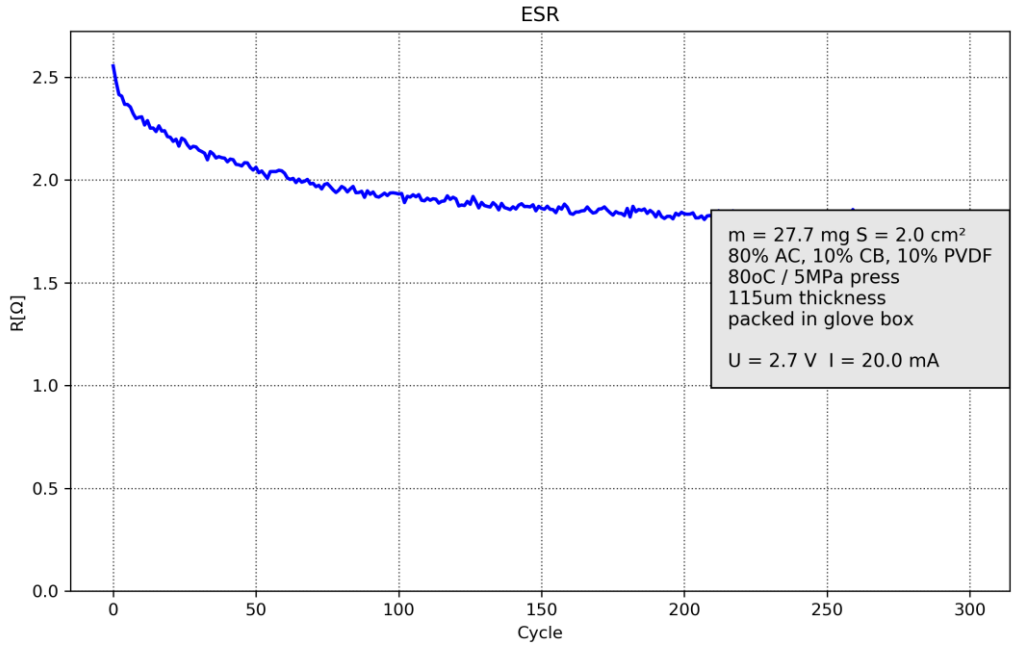
- SC is discharged from 2.7 V, current is 20 mA
- mass of active material is 27.7 mg
- area is 2 cm^2
- discharge cycles are stored in files '1012KRONO2_0001.oxw' to '1012KRONO2_0300.oxw'
- function 'readAL' is used to read data from above mentioned files
- ESR is calculated from segment between 90% and 70% of initial voltage, using linear approximation of the discharge curve
- C is calculated from segment between 90% and 70% of initial voltage
- discharge curve is plotted for each ten cycles
- cumulative discharge plot is generated each 50 cycles

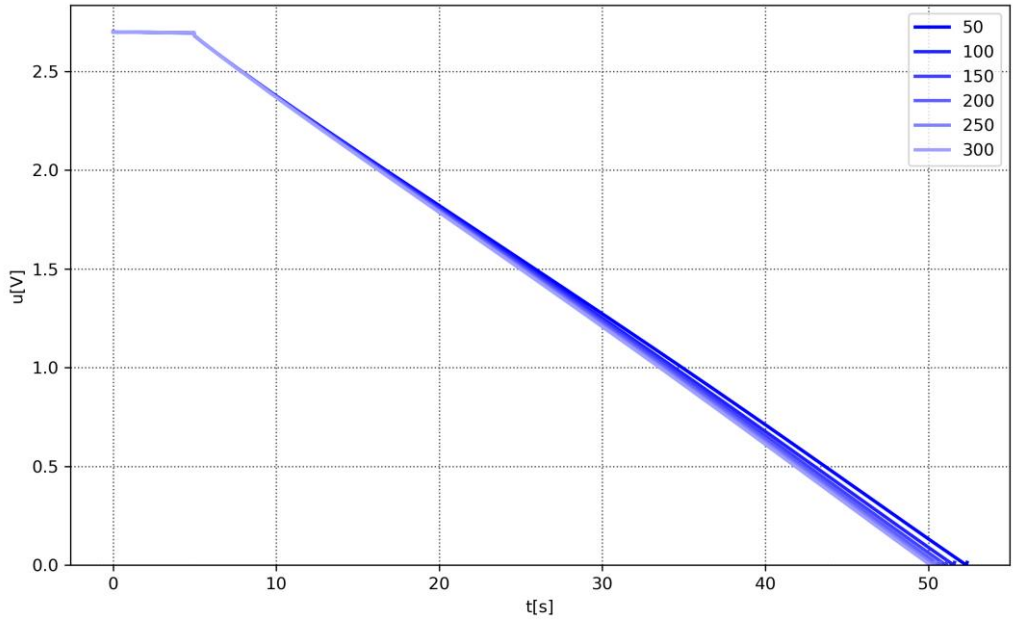
Python call:

```
SCanalysis(2.7,20,27.7,2,"80% AC, 10% CB, 10% PVDF", "80oC / 5MPa press",\
      "115um thickness", "packed in glove box", "", "*KRONO2*.oxw", \
      "readFunctions.readAL", (0.9,0.7), (0.9,0.7), 10, 50)
```

Sample plots:







Additional plots:

- Discharge curve(s)
- Specific capacitance (by mass and area)
- Specific energy and power density
- Discharge curve nonlinearity

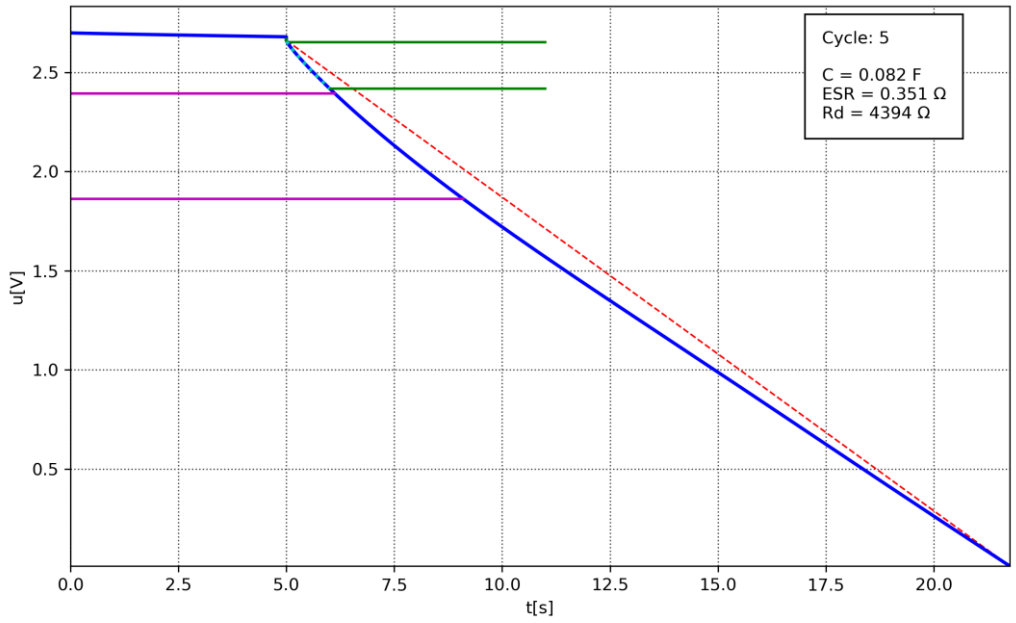
```
SCanalysis(2.7, [8 9.8 12 14.7 18.1 22.1 27.1 33.3 40.8 50], 10, 2,
      'label1', 'label2' , 'label3', 'label4','label5' , {'008_0mA_*.oxw'
      '009_8mA_*.oxw'   '012_0mA_*.oxw'   '014_7mA_*.oxw'   '018_1mA_*.oxw'
      '022_1mA_*.oxw'   '027_1mA_*.oxw'   '033_3mA_*.oxw'   '040_8mA_*.oxw'
      '050_0mA_*.oxw' 'nakon_*.oxw'}, @readAL, [1,1],[0.9,0.7],1,1);
```

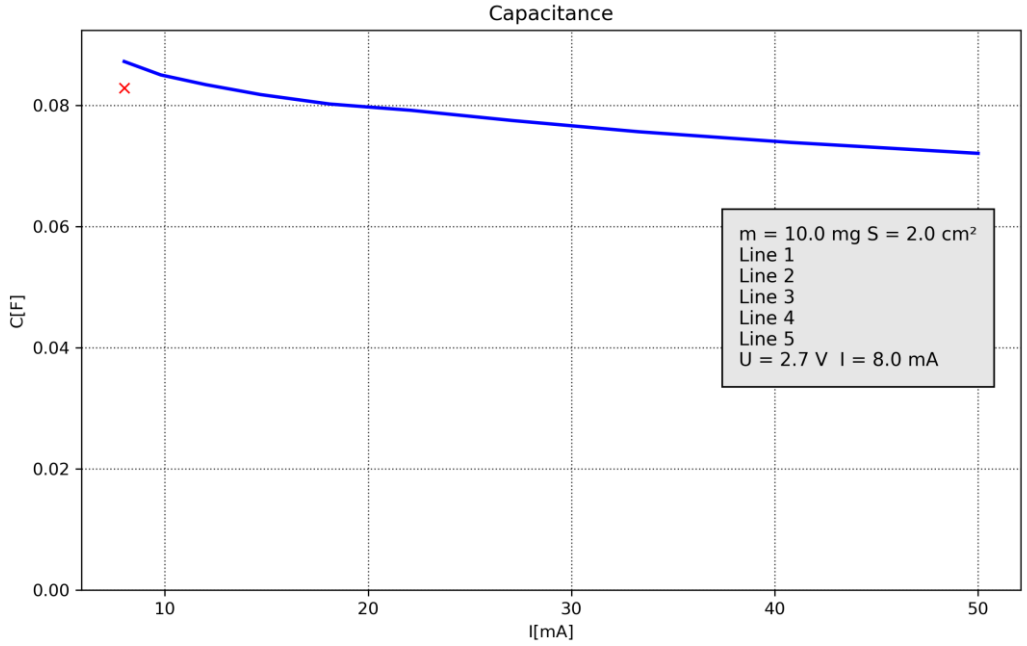
- SC is discharged from 2.7 V, with ten currents, from 8mA to 50mA
- mass is 10 mg
- area is 2 cm^2
- file name filter for each current is specified, as well as for additional set of files containing data from repeated measurement for starting current
- function 'readAL' is used to read data from above mentioned files
- ESR is calculated from segment beginning at the start of discharge to the point 1s later, using 3rd order polynomial approximation of discharge curve
- C is calculated from segment between 90% and 70% of initial voltage
- discharge curve is plotted for each cycle
- cumulative discharge plot is generated for each cycle

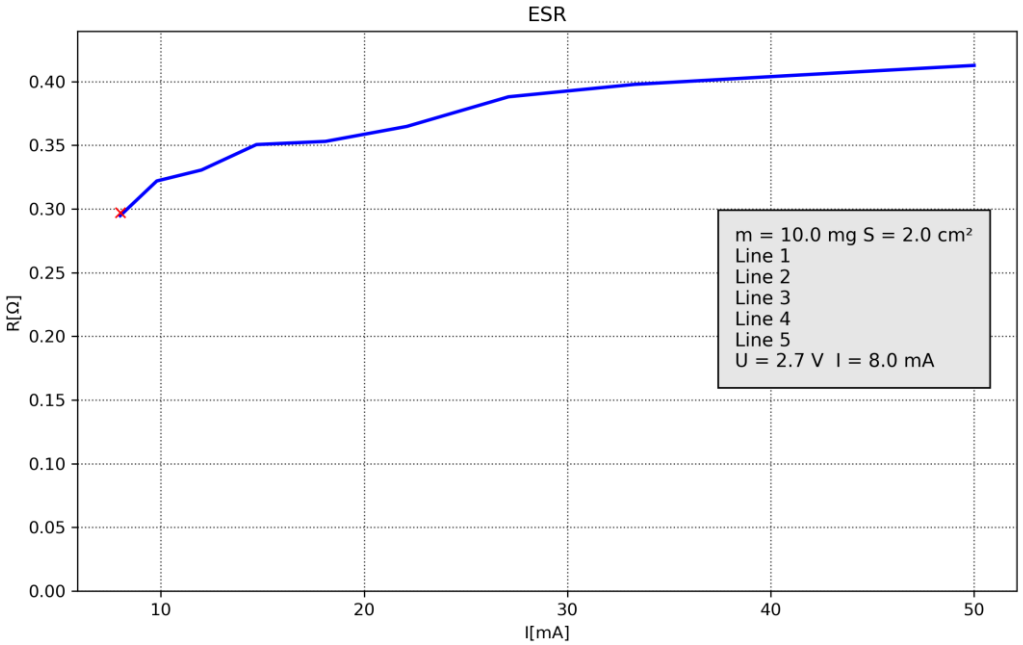
Python call:

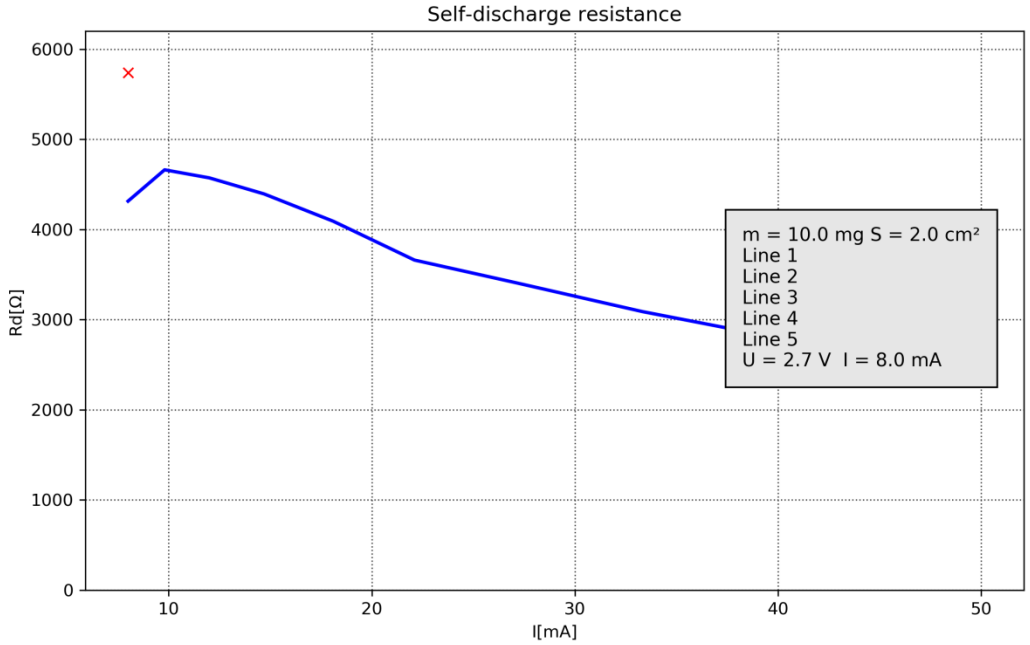
```
SCanalysis(2.7, (8, 9.8, 12, 14.7, 18.1, 22.1, 27.1, 33.3, 40.8, 50), 10,\
      2, "Line 1", "Line 2", "Line 3", "Line 4", "Line 5", \
      ("008_0mA_*.oxw", "009_8mA_*.oxw", "012_0mA_*.oxw", "014_7mA_*.oxw",\
      "018_1mA_*.oxw", "022_1mA_*.oxw", "027_1mA_*.oxw", "033_3mA_*.oxw",\
      "040_8mA_*.oxw", "050_0mA_*.oxw", "nakon_*.oxw"), \
      "readFunctions.readAL", (1,1), (0.9,0.7), 1, 1)
```

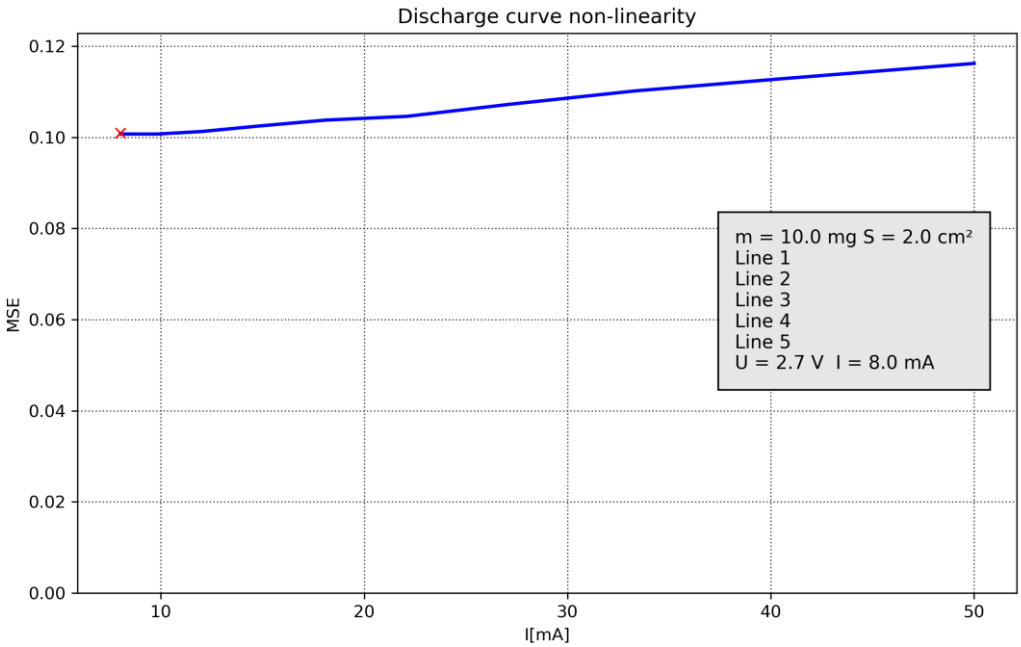
Sample plots:

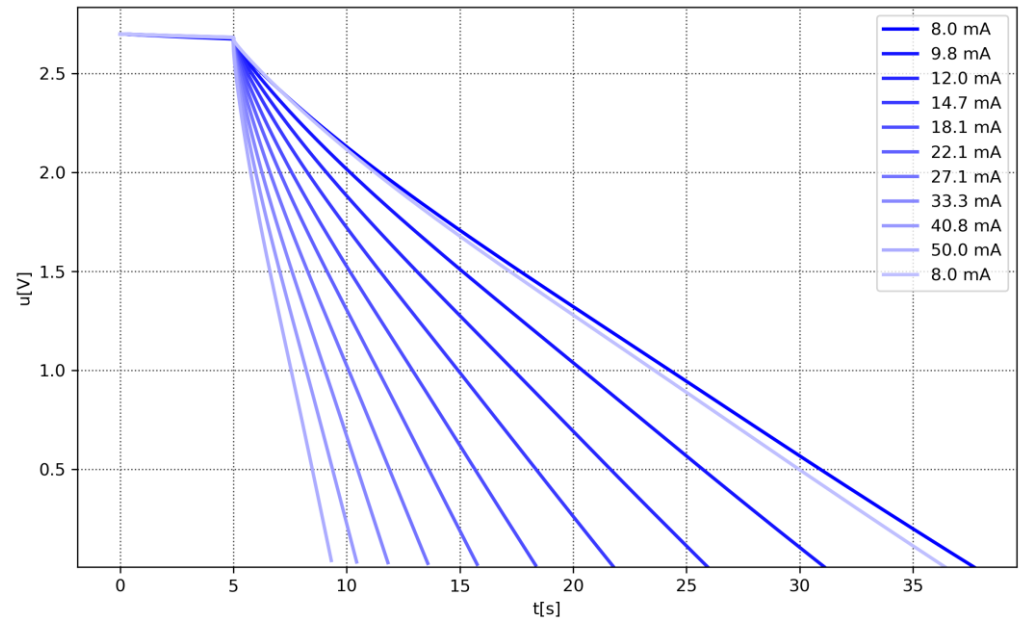

Cycle: 5
C = 0.082 F
ESR = 0.351 Ω
Rd = 4394 Ω
u[V]
t[s]
2.5
2.0
1.5
1.0
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Capacitance
m = 10.0 mg S = 2.0 cm²
Line 1
Line 2
Line 3
Line 4
Line 5
U = 2.7 V I = 8.0 mA
C[F]
I[mA]
0.08
0.06
0.04
0.02
0.00
10
20
30
40
50


ESR
R[Ω]
I[mA]
m = 10.0 mg S = 2.0 cm²
Line 1
Line 2
Line 3
Line 4
Line 5
U = 2.7 V I = 8.0 mA


Self-discharge resistance
Rd[Ω]
I[mA]
m = 10.0 mg S = 2.0 cm²
Line 1
Line 2
Line 3
Line 4
Line 5
U = 2.7 V I = 8.0 mA





Additional plots:

- Capacitance, specific capacitance, ESR, power and energy density and nonlinearity plots are generated for each current
- All the above plots are also generated related to the discharge current