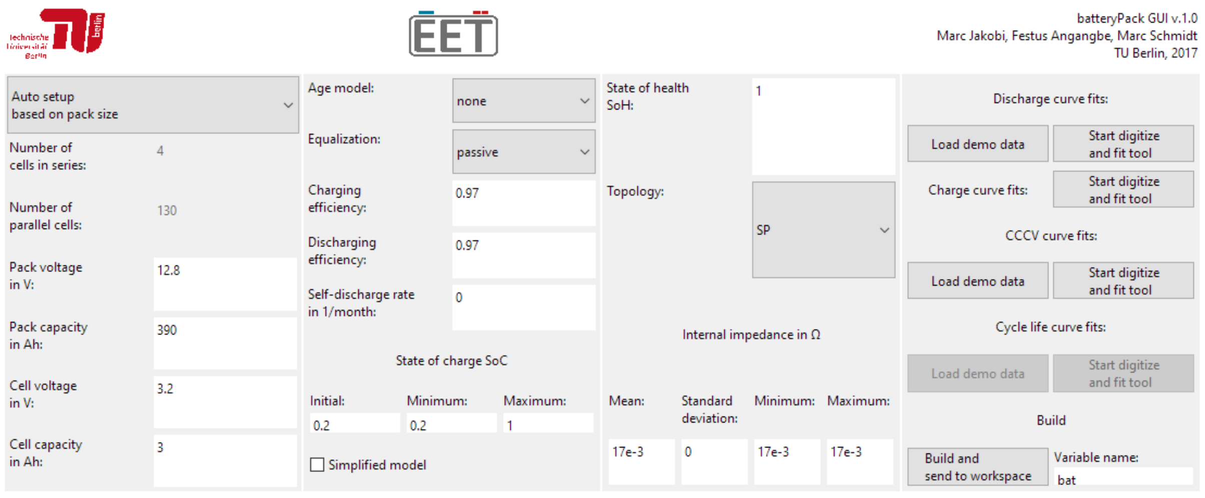

batteryPack GUI v.1.0
Marc Jakobi, Festus Angangbe, Marc Schmidt
TU Berlin, 2017

Auto setup based on pack size

Number of cells in series:	4
Number of parallel cells:	130
Pack voltage in V:	12.8
Pack capacity in Ah:	390
Cell voltage in V:	3.2
Cell capacity in Ah:	3
Age model:	none
Equalization:	passive
Charging efficiency:	0.97
Discharging efficiency:	0.97
Self-discharge rate in 1/month:	0
State of health SoH:	1
Topology:	SP

State of charge SoC

Initial:	Minimum:	Maximum:
0.2	0.2	1

☐ Simplified model

Internal impedance in Ω

Mean:	Standard deviation:	Minimum:	Maximum:
17e-3	0	17e-3	17e-3

Discharge curve fits:
Load demo data
Start digitize and fit tool
Charge curve fits:
Start digitize and fit tool
CCCV curve fits:
Load demo data
Start digitize and fit tool
Cycle life curve fits:
Load demo data
Start digitize and fit tool
Build
Build and send to workspace

Variable name: bat

Cell Resolved Matlab® OOP Model of a Lithium Iron Phosphate Battery Pack

Marc Jakobi, Festus Anyangbe, Marc Schmitdt
February 26, 2017
HTW Berlin

Supervision:
M.Sc. Steven Neupert
TU Berlin

Contents

List of Figures

List of Tables

Abbreviations

OO	object oriented
OOP	object oriented programming

List of Symbols

Symbol	Unit	Description
C_{dis}	Ah	discharge capacity
F	As/mol	Faraday constant
I	A	Current
SoC	%	state of charge
R	J/(mol · K)	universal gas constant
$rmse$	-	root mean squared error
T	K or °C	temperature
V	V	voltage
z_{Li}	-	ionic charge number of lithium

1 Interfaces

In OOP languages such as JAVA™and C++, the term "interface" commonly refers to an abstract set of methods that specifies the behaviour that objects must implement.

2 Discharge curves

Many battery data sheets provide measured discharge curves, on which the charging and discharging behaviour of this model is based. Rather than determining the curves according to the internal impedance, a common approach [1], this model determins the curves directly by means of digitizing the images and creating a curve fit. The classes used for fitting and modelling the discharge curves are described in the following subsections.

2.1 Single discharge curve

For modelling a single discharge curve, the class `dischargeFit` is used, which implements the interface `curveFitInterface`. The curve is fitted according to [2], using a function that is loosely based on the Nernst equation with two exonential functions superimposed as a correction for the voltage drops at the beginning and end of the curve.

Section describing interface, etc.

$$\begin{aligned} V(SoC) = x_1 - \frac{R \cdot T}{z_{\text{Li}} \cdot F} \cdot ln\left(\frac{SoC}{1 - SoC}\right) + x_2 \cdot SoC + x_3 \\ +(x_4 + (x_5 + x_4 \cdot x_6) \cdot SoC) \cdot exp(-x_6 \cdot SoC) \\ +x_7 \cdot exp(-x_8 \cdot SoC) \end{aligned} \quad (1)$$

where $x_1,\ ..,\ x_8$ are the fit parameters, $R = 8.3144598$ J/(mol · K) is the universal gas constant, $z_{\text{Li}} = 1$ is the ionic charge number of lithium, $F = 96485.3328959$ As/mol is the Faraday constant, SoC is the state of charge, V is the voltage in V and T is the temperature in K at which the curve was recorded. The curves are fitted using the levenberg-marquardt algorithm and either the `lsqcurvefit` method, the `fminsearch` method or a combination of both, depending on the user's preference.

2.1.1 Object creation

A `dischargeFit` object is created with the digitized raw data - the voltage V in V, the discharge capacity C_{dis} in Ah, the current I in A at which the curve was recorded and the temperature T in K at which the curve was recorded.

```
d = dischargeFit(V, C_dis, I, T);
```

`V` and `C_dis` are vectors containing the digitized raw data from the data sheet. Further options, such as initial values for the fit parameters $x_1,\ ..,\ x_8$ and the fit method can be passed to the constructor using Matlab's name-value pair syntax:

```
d = dischargeFit(V, C_dis, I, T, 'OptionName', OptionValue);
```

By default, the initial fit parameters are set to zero and the curve is fit by first using `lsqcurvefit`, followed by `fminsearch`. The initial fit parameters are stored in a vector `x0` of `length` 8, which can be passed via the option name `'x0'`, for example using the following syntax:

```
x0 = ones(8, 1);
d = dischargeFit(V, C_dis, I, T, 'x0', x0);
```

The method used for the curve fitting can be passed to the constructor using the option name `'mode'`. The corresponding value must be one of the following three strings:

- `'lsq'` for `lsqcurvefit`
- `'fmin'` for `fminsearch`
- `'both'` for `lsqcurvefit` followed by another fit using `fminsearch`

e.g.

```
d = dischargeFit(V, C_dis, I, T, 'mode', 'fmin');
d.plotResults
```

Depending on the curve and on the technology, one of the methods may return a better result.

2.1.2 Visual validation

A visual validation can be performed by calling the class's `plotResults` method (see above). In Figure 1, the results of two `dischargeFit` objects using the same raw data are compared.

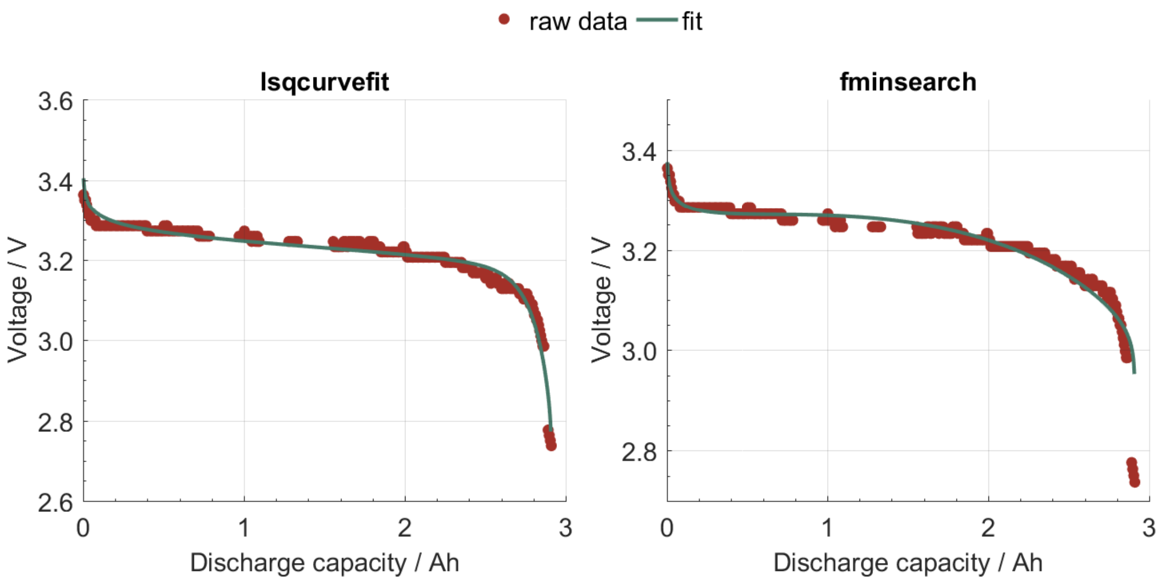

Figure 1: *Fit results of the* `dischargeFit` *class using the fit methods* `lsqcurvefit` *and* `fminsearch`*, respectively. The raw data was extracted from [3].*

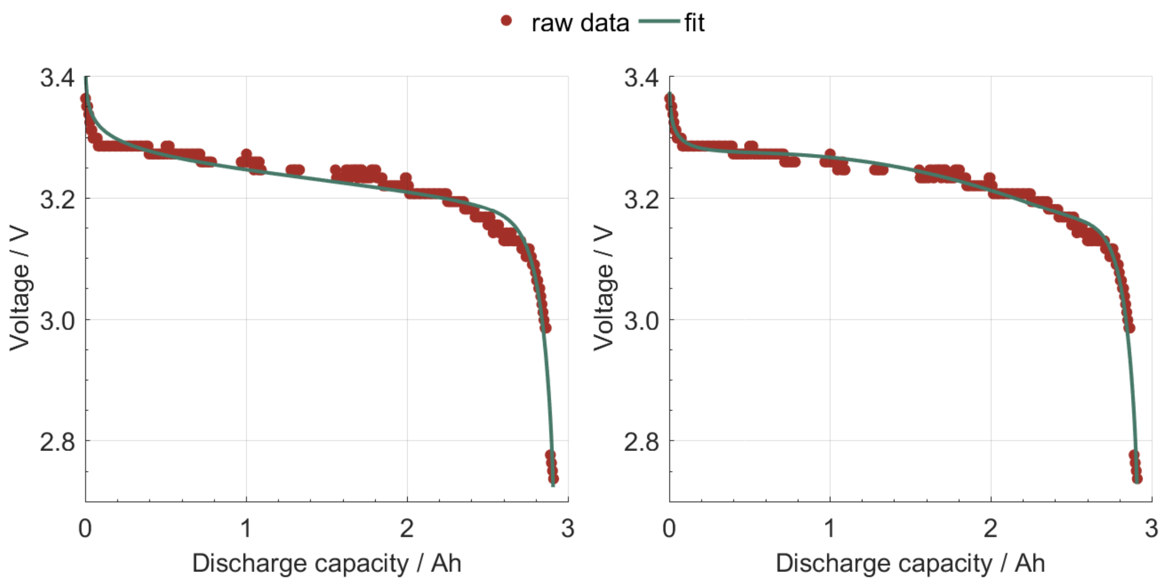

Figure 2: *Fit results of the* `dischargeFit` *class using the fit mode* `'both'` *with the default parameter initialization (left) and with a custom parameter initialization (right). The raw data was extracted from [3].*

In this example, `'lsq'` appears to return better results for the voltage drop at the end of the curve, while `'fmin'` results in a more precise fit for the voltage drop at the beginning of the curve. Further differences can be seen in the fits' curvatures. The `'lsq'` option results in a slightly flatter curve than the `'fmin'` mode. The results of a `dischargeFit` object using the `'both'` option are presented in Figure 2. Using the default fit parameter initialization of `zeros` (left) appears to improve the curvature and voltage drops slightly, compared to the other modes. Further improvements can be made by passing custom initial fit parameters to the constructor via the option `'x0'` (see Figure 2, right).

2.1.3 Object properties

Further fit quality analysis can be performed via the root mean squared error $rmse$, the mean difference in voltage between the raw data and the curve fit at the respective positions of the raw data $\overline{\Delta V}$ in V and the maximum difference between the raw data and the curve fit at the respective positions ΔV_{max} in V. The $rmse$ for a curve fit with the raw data y_{raw} and the fitted data y_{fit} at the same x coordinates is defined as

$$rmse = \sqrt{\sum_{i=1}^{n} (|y_{\text{raw},i} - y_{\text{fit},i}|)^2} \tag{2}$$

where i is the index of the measurement and n is the number of measurements.

References

[1] Lijun Gao, Shengyi Liu, and R. Dougal, "Dynamic lithium-ion battery model for system simulation," *IEEE Transactions on Components and Packaging Technologies*, vol. 25, no. 3, pp. 495–505, Sep. 2002, ISSN: 1521-3331. DOI: 10.1109/TCAPT.2002.803653.

[2] F. V. Werder, "Entwicklung eines batteriemodells auf lithium basis," PhD thesis, HTW Berlin, Berlin, Sep. 30, 2014.

[3] "Data sheet: BM26650etc1 - high power lithium iron phosphate cell," Batterien-Montage-Zentrum GmbH (BMZ), Karlstein, 2010. [Online]. Available: www.bmz-gmbh.de.