Projekt-Labor: Automatisierungstechnik

Development and validation of a SoC algorithm for solarbatterychargecontroller Mppt-1210-hus by Libre.solar with lead acid WS2122

handed in

Emile Schons 903606

EDV.Nr.:123 456

of the faculty VII – Elektrotechnik –
of Berlin University of Applied Sciences and Technology handed in Projektarbeit

to finalize the Projekt-Labor
Master of Engineering (M.Eng.)
im Studiengang
Energie und Automatisierungstechnik

Tag der Abgabe November 1, 2021

Gutachter
Gutachter1   Berlin University of Applied Sciences and Technology

Contents

List of Figures

Introduction

The scope of the project is to produce an algorithm for the solar charge controller (mppt-1210-hus) [1] produced by libre.solar, which outputs the state of charge (SoC) of the lead acid battery attached to it.
A key principle followed throughout the project is to use, wherever possible, Free Open-source Hardware (FOSH) and Free Open-source Software (FOSS). Consequently, the whole project has been released as source code with a license adhering to these principles [Schons 2021] is available at [2]. As the SoC can only be measured indirectly, it has to be deduced from available measurements. The charge controller exposes battery current and voltage measurements once a second for this purpose.
There are three main approaches for estimating the SoC: model-based, data-driven, coulomb-counting w/ Open circuit voltage (OCV) [Espedal et al. 2021]. Best results seems to be achieved with combining the approaches to profit from their different strengths.

Current integration method - Coulomb Counting /w OCV

Seems the most simple way to define SoC:

$$z(t) = z(0) - \frac{1}{Q_C} \int_0^{\Delta t} i(t)\, dt \tag{1.1}$$

where:

z = the SoC
i = battery current
Q_C = battery capacity
Δt = the time interval between two current measurements

The initial SoC $z(0)$ is determined by consulting a OCV-SoC lookup table. Starting from there the charge in relation to total capacity entering and leaving the battery is subtracted from SoC. This method has several disadvantages:
-The initial SoC incorporates already an error, which won't be corrected till recalibration
-Error in counting charges and measurement errors accumulate quickly
-The algorithms only recalibrate reliably on full charge, which especially in solar systems does not occur regularly, producing awful SoC till recalibration.

The coulomb counting method is most often combined with ongoing voltage measurements, while battery is at rest, known as OCV and a resulting OCV-SoC-lookup as shown in figure 1.1. This mitigates the recalibration problem in case the battery gets some rests, but this enhancement still does not take hysteresis into account, which make OCV-SoC-lookup even with voltages at rest unreliable [Tjandra et al. 2014] [Khan und Choi 2016].

[1] `https://github.com/LibreSolar/mppt-1210-hus`

[2] `https://github.com/mulles/Doc_Projekt-Labor_SOC`

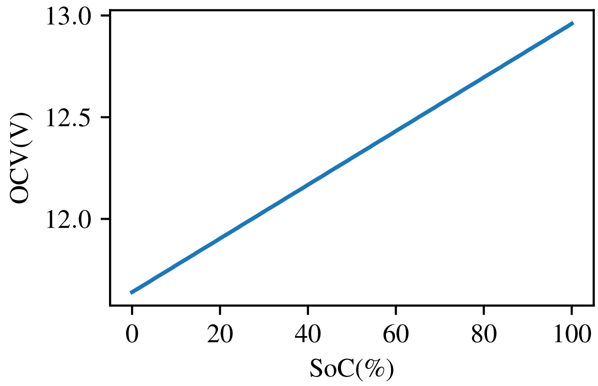


Figure 1.1: OCV-SoC lead acid battery

The existing SoC function $Charger :: update\_soc$ [3] embebbed in mppt-1210-hus uses OCV-SoC-lookup function based on linear interpolation of ocv_empty and ocv_full:

```
void Charger::update_soc(BatConf *bat_conf)
{
    static int soc_filtered = 0;       // SOC / 100 for better filtering

    if (fabs(port->current) < 0.2) {
        int soc_new = (int)((port->bus->voltage - bat_conf->ocv_empty) /
                   (bat_conf->ocv_full - bat_conf->ocv_empty) * 10000.0);

        if (soc_new > 500 && soc_filtered == 0) {
            // bypass filter during initialization
            soc_filtered = soc_new;
        }
        else {
            // filtering to adjust SOC very slowly
            soc_filtered += (soc_new - soc_filtered) / 100;
        }

        if (soc_filtered > 10000) {
            soc_filtered = 10000;
        }
        else if (soc_filtered < 0) {
            soc_filtered = 0;
        }
        soc = soc_filtered / 100;
    }

    discharged_Ah += -port->current / 3600.0F;   // charged current is positiv
}
```

TODO Remove 1.2 and produce one that fits Libre.solar Algo maybe cite https://www.sciencedirect.com/topics/engineering/coulomb-counting but it does not seems to be a good reference.

[3] https://github.com/LibreSolar/charge-controller-firmware/blob/5196694e42c38eee18ab25e65bd51ec578eba101/src/bat_charger.cpp#L325

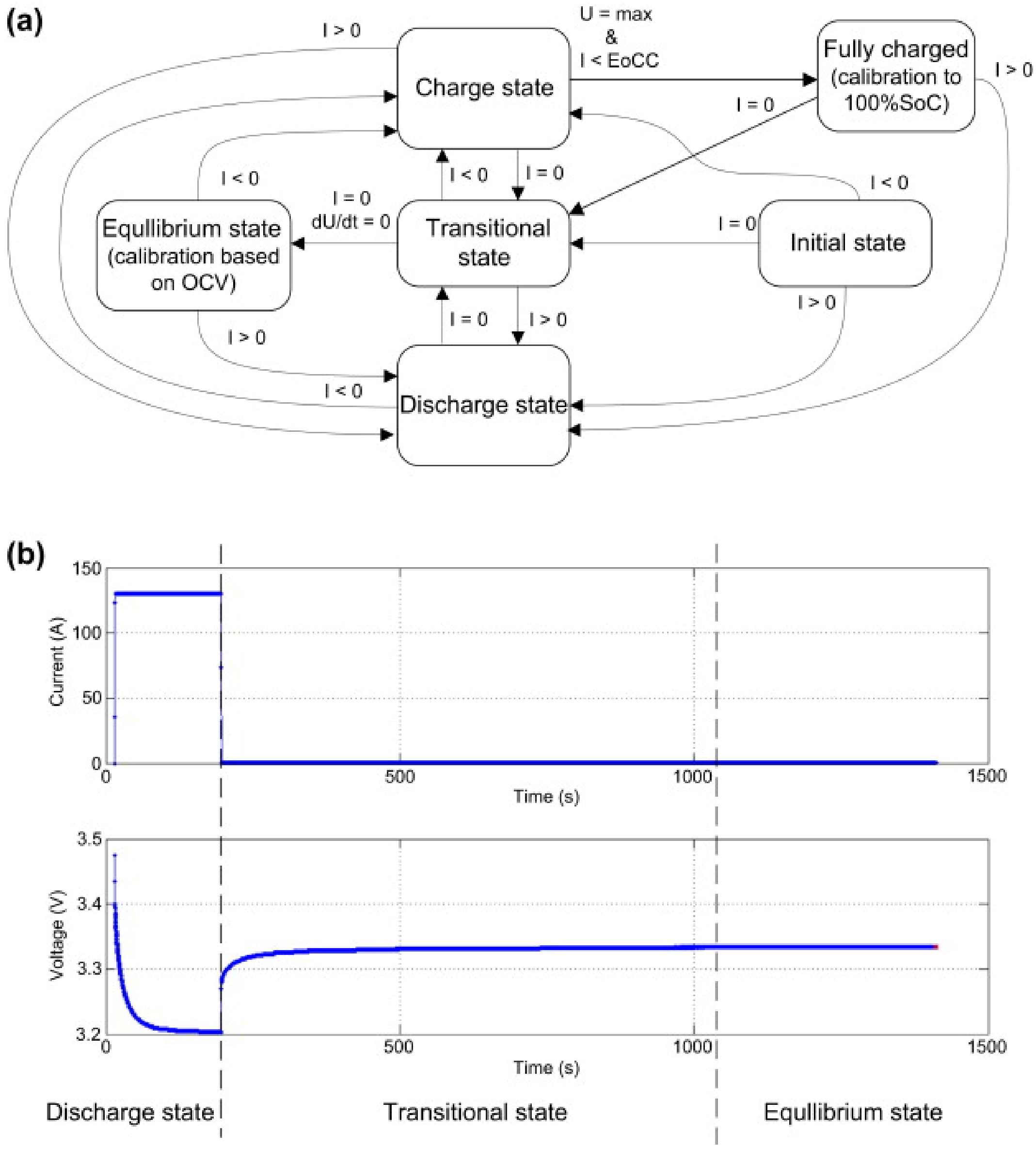


Figure 1.2: State maschine libre.solar

Equivalent circuit based models

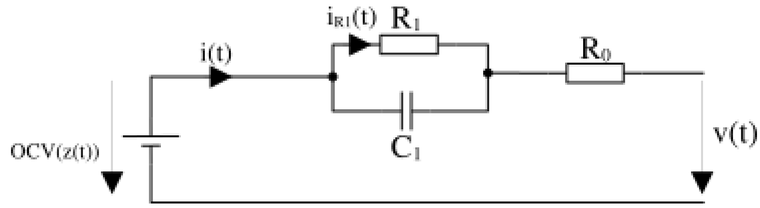


Figure 1.3: Equivalent circuit model

The figure 1.3 shows an empirical model known as Thévenin model or equivalent circuit model (ECM), adhering to the equation:

$$v(t) = OCV(z(t)) - R_1 i_{R_1}(t) - R_0 i(t) \tag{1.2}$$

where:

v = voltage measured on battery terminals under load
z = the SoC
OCV = a function based on a SoC-OCV-lookup table
R_1, C_1, R_0 = don't exist as physical components inside the battery, their values are chosen to fit battery test data for example that of a hybrid pulse power characterization (HPPC) battery test

Other empirical models are the auto regressive exogenous (ARX) model and dual polarisation (DP) model

Physics based models

As Gregory Plettv puts it in his lectures notes of course ECE5720 [4] : "Battery management and controls using physics-based models is a major focus of our present research efforts and (I hope) of a future course" in 2015. The first papers on the topic are available now [Miguel et al. 2021], but physics based models (PBM) still are hard to implement and not much material or software code is available. Thus, the ECM model is preferred for now. Moreover, the PBM based models have a higher computational complexity than empirical models, making it non suitable for the mppt-1210-hus micro-controller unit (MCU) being a 32bit ARM STM32L072.

Data-driven

These approaches are based on artificial
intelligence (AI) and need good training data and the right training parameters. Combined with PBM's it is a very promising research field [Miguel et al. 2021], which first successes to reduce the computational complexity so online estimation of SoC on MCU is possible.

[4] http://mocha-java.uccs.edu/ECE5720/ECE5720-Notes02.pdf

Implementation

It is popular to combine any of the approaches introduced in 1 with a kalman filter (KF) in order to reduce the influence of white (uncorrelated) noise in measurements and to combine voltage and current measurements known as sensor fusion. The second principle makes the algorithm resilient to inaccurately chosen ECM parameters. In this project the combination of coulomb counting 1 and ECM 1 has been chosen.

As the ECM's equation 1.2 is non-linear an extended kalman filter (EKF) needs to be used. The EKF linearizes the non-linear system by calculating the Jacobian matrix, which is the Taylor series expansions of first order.

2.1 EFK implementation fork

A good step by step guide to implement the extended kalman filter is [Rzepka et al. 2021]. The algorithm refined in this project is based on a fork of the algorithm named $kalman-soc$[1] designed by Matthew Johnston for the company okra-solar.
Some refinements consists of improvements or adaptions to libre.solar use case and code base.

Changelog:
-adoption of libre solar code style guide
-change of energy counting to coulomb counting
-use of float instead of integer in order to guarantee correct calculations, with no overflow.
note exact changes are traceable with the commits to the fork.

2.1.1 Equations defining the EKF

The EKF consist of two equations describing the system the SoC is to be predicted for:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{i}_k) + \boldsymbol{w}_k \ (state\ space\ equationakaobservationmodelakastatetransition?) \tag{2.1}$$

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \qquad (control\ output\ equationakacontrol - inputmodel) \tag{2.2}$$

where:

$f(\boldsymbol{x}_k, \boldsymbol{i}_k, \Delta t)$ = $x_k - \frac{\Delta t}{Q_C} i_k$ (discrete coulomb counting equation 1.1)
$h(\boldsymbol{x}_k)$ = $OCV(z_k)$
y_k = battery terminal voltage, v_k is common
i_k = battery current, u_k is common
x_k = the discrete SoC, in literature z_k is common

[1] https://github.com/mulles/kalman-soc

w_k = noise
v_k = noise

note that in some literature's $z_k \equiv x_k$ and $i_k \equiv u_k$

$$v_k = D\,OCV(z_k) + Cx_k + Di_k \tag{2.3}$$

where:

C = $C = [0, -R_1, -R_2, ..., M]$
D = R_0

2.1.2 Parametrization of the ECM model and other parameters inside the EFK

The parametrization is particularly challenging. It is typically done offline [2] and requires quite advanced equipment.

2.1.3 EKF inner functioning

TODO Make a diagram where you can see the input output and steps

Most general form of state observer equations:
$x_{k+1} = Ax_k + Bu_k$ (state observer equation)
$y_k = Cx_k + dDu_k$ (output equation)

$u_k \equiv i_k$ a control vector (input), defined as the measurement of current through the battery at time k.
$y_k \equiv v_k$ an observation (or measurement), defined as a voltage measurement of the battery at time k.
$A = I$ identity matrix f.i $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$B = \begin{bmatrix} -\frac{\Delta t}{Q_C} & 0 \\ 0 & 1 \end{bmatrix}$ is **the control-input model**, deduced from $f(\boldsymbol{x}_k, \boldsymbol{i}_k, \Delta t) = x_k - \frac{1}{Q_C}\int_0^{\Delta t} i_k \, dk$
$C = [0, -R_1, -R_2, ..., M] = H(x_k)$ and $H(x_k)$is the **observation model** , which maps the state space into the observed space and TODO understand the relationship to $h(x_k)$
$D = R_0$

$\hat{x}_{k+1} = (A - BK)\,\hat{x}_k + L\,(y_k - \hat{y}_k)$ (here u_k seems missing)

Predicted variables $\hat{y}_k$ and $\hat{x}_k$ are commonly denoted by a "hat" to distinguish them from y_k and $x(k)$ of the physical system. As the state of charge (SoC) denoted as $\hat{x}_k = SoC_k$ cannot be measured directly it is always a predicted variable, opposed to $y(k)$, which is the measured circuit voltage. Consequently $\hat{y}_k = v_k$ is the predicted circuit voltage also known as measurable output:
$\hat{y}_k = (C - DK)\,\hat{x}_k$
Input to the extend kalman filter (EKF) is current and voltage measurement and the period of time between these measurements. The initialization of the EKF outputs the initial estimate state of charge $x_{k|k=0}$ another input to the EKF.

[2]not generated by the micro controller unit (MCU) the algorithm is deployed to, thus not on the system running the final algorithm. Offline means on more advanced hardware, which is not available in the final product. See `https://en.wikipedia.org/wiki/Online_model`

System's dynamic model
The $f()$ function is defined as the $f(\boldsymbol{x}_k, \boldsymbol{i}_k, \Delta t) = x_k - \frac{1}{Q_C}\int_0^{\Delta t} i_k \, dk$ or more discrete $f(x_k, i_k, \Delta t) = x_k - \frac{\Delta t}{Q_C} i_k$, thus current measurement i_k The $h(\boldsymbol{x}_k)$ function is defined as the OCV lookup table. A general OCV lookup table for the battery chemistry can be used or for better results a specific OCV lookup established by offline measurements for the given battery should be used. To further improve the OCV prediction a correction of it can be performed by a equivalent circuit model (ECM) of the battery feeded by the current measurement used to predict the SoC: $v_k = D \, OCV(z_k) + Cx_k + Di_k$ with $C = [0, -R_1, -R_2, ..., M]$ and $D = R_0$
-Measurement equation, input (measured voltage, OCV lookup table, current if the correction with a Enhanced Self-Correcting (ESC) Cell Model /ECM) -> output SOC) equation should be use standard letters: filterpy, wikipedia, gregoryPlett, Step by Step Guide
After having defined the observation and control-input model as matrices and described their meaning in case of SoC estimation we proceed to the functioning of the EKF, which is typically divided into to steps, Predict and Update.
In the **predict** step a future state of charge estimate $\hat{x}_{k+1}$ is predicted based on the current state of charge estimate $\hat{x}_k$ and the current i_k during the period Δt (between k and $k+1$) by calculating $\mathbf{\hat{x}_{k+1}} = \mathbf{Ax_k} + \mathbf{Bu_k}$ Moreover an estimate of the covariance $P_{k+1}x$ is calculated based on noise covariances Q_k and current P_k (wiki: a measure of the estimated uncertainty of the prediction of the system's state) $\mathbf{P}_{k+1} = \mathbf{B}_k \mathbf{P}_k \mathbf{B}_k^\intercal + \mathbf{Q}_k$
In the **update** step
$\mathbf{K}_k = \mathbf{P}_{k+1} \mathbf{H}_k^\intercal (H_k \mathbf{P}_{k+1} \mathbf{H}_k^{\intercal + \mathbf{R}_k)^{-1}}$ is the kalman gain, which weights whether the SoC based on the measurement of the circuit voltage v_k is more trusted than the SoC prediction based on current i_k
$\hat{\mathbf{x}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)(\hat{\mathbf{x}}_{k+1}) + (\mathbf{K}_k)(\mathbf{H}_k \hat{\mathbf{x}}_k + \mathbf{v}_k)$ TODO update $\hat{x}_k$ because one can not now want one want to calculate

$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1} \mathbf{H}_{k+1}) \, \mathbf{P}_k$ update of the covariance TODO is H_k the same as OCV?

$\mathbf{R_k}$ the covariance of the observation noise

Most general equation of extended kalman filter EKF:

$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$ (state transition model)
$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$ (observation model)
f -> predicted state from the previous estimate
h -> compute the predicted measurement from the predicted state

Predict
$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k)$
$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k$

Update
$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})$
$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$
$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$
$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$
$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1}$

Validation

2 Arten von Datensätze:
Vergleich alter Algo neuer Algo, mit Daten von einem Device. X mit Zeit? Vllt schafft man dies im gleiche Schritt wie man die Zeiträume verbessert? Y Achse in 100
Titel SoC Libre.solar Algo vs Kalman-soc auf Datensatz von Device xy im Zeitraum xx. Kein Titel in der Graphik selber.
Mit Matplotlib da ist wsl mehr Hilfe verfügbar? Schnell mal mit Plotly versuchen.
-Prozent Abweichung dazwischen. Warum ist er besser? Scheint schwer zu diskutieren.
-Meine Daten?
-Neue Daten generieren? Der Prozess ist ja eigentlich recht automatisiert.
It is difficult to say that how much the kalman-soc algo is better, but sure is that both are different and that there are many cases where it is sure that libre.solar performs poorly, show examples timeframes.
Discuss results. (4d)

3.1 Graphs with Matplotlib or plotly

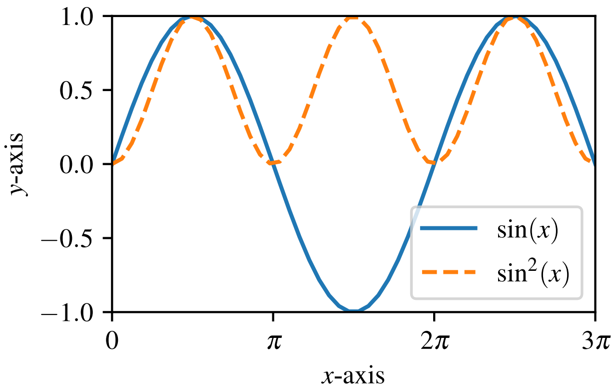


Figure 3.1: A plot created with PythonTeX and Matplotlib

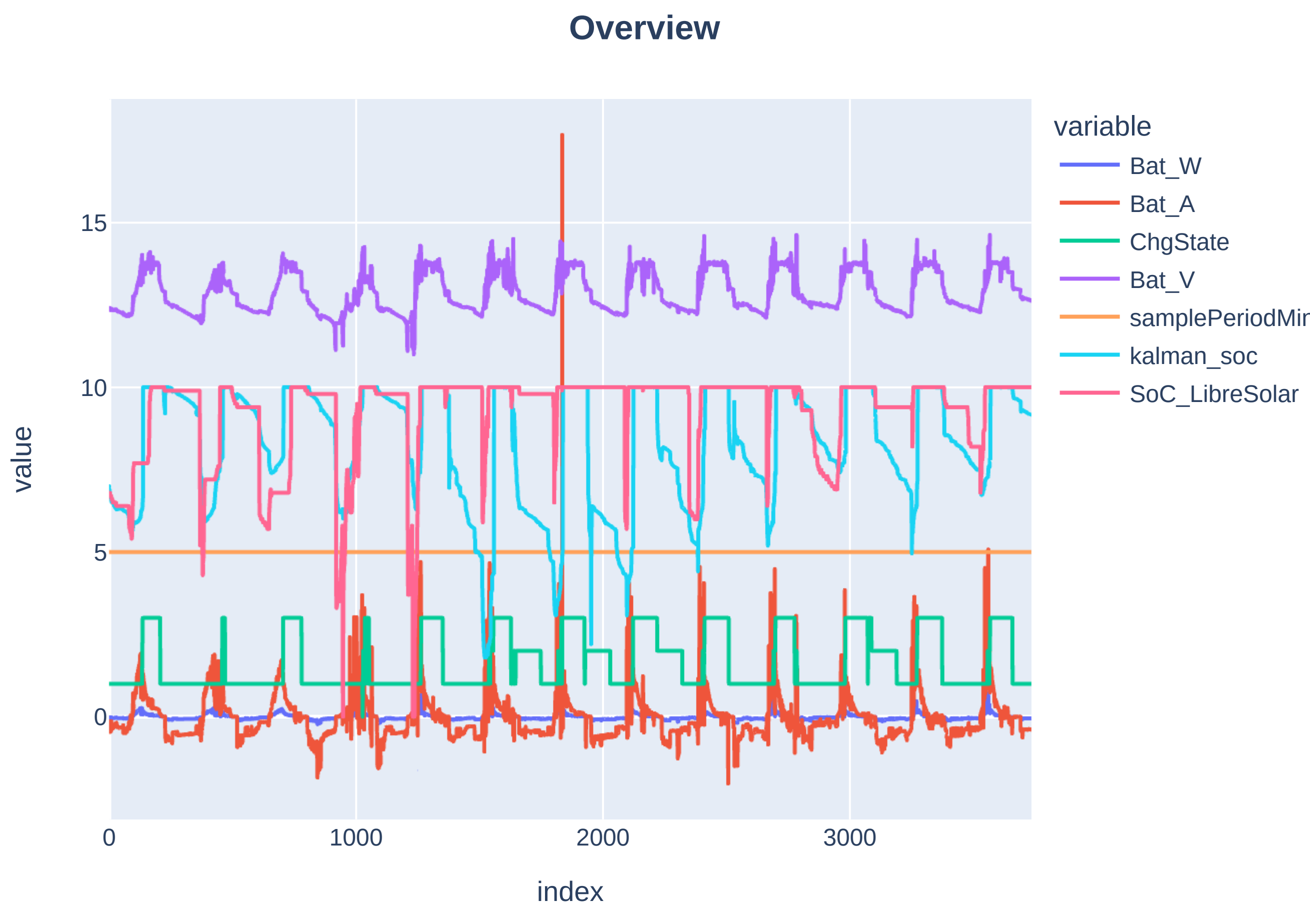


Figure 3.2: Overview of data

Figure 3.3: Overview of data

Conclusion

The key to success for a good SoC is a good definition of both models and a good parametrization. The implementation is independent of the use case and depends on the skills of the programmer, in his hands lies the optimization of the code. In case no FPU is available on the micro-controller unit (MCU) fix point arithmetic might be an option. The Adaptive extend Kalman Filter (AEKF) is an option, where the parameters are determined online, but still AEKF needs to be initialized with parameters not necessary fitting all batteries of a given chemistry, because of different capacities or tolerances and specific designs. TODO It needs to be researched if it only affects the convergence time or if convergence is not reached at all? A further advantage of an adaptive filter is that the state of health (SOH) can be determined along the side. When simple observation and control-input models are used as in the okra kalman-soc algorithm, the parametrization might be much easier? TODO to check if the initial parameters, in this case OCV in function of SoC data can be used generally or only on the batteries, the data has been generated with.

Citations concerning the on/off-line parametrization

"A sigma-point Kalman filter is further used to manage inaccuracies generated by the reduction process and experimental-related issues such as measurement error (noise) in the current and voltage sensors."
Offline parametrization:
'As a result, many experimental pretests investigating the effects of the internal and external conditions of a battery on its parameters are required, since the accuracy of state estimation depends on the quality of the information regarding battery parameter changes'
'Therefore, some tests data must be available in advance to find the parameters of these models.' [Hussein und Batarseh 2011]

Gregory Plett says:
'Two possible approaches:

First, an algorithm might somehow adapt the parameter values of the model during operation to match presently observed current–voltage behaviors; but, this **must be done very carefully to avoid making the model unstable or physically nonmeaningful**.

Alternately, a set of models could be pre-computed at different feasible aging points and the model from this set that most closely predicts presently observed current–voltage dynamics could be selected from the set. This second approach guarantees stable and physically meaningful models since all models in the pre-computed set meet these criteria. We propose such an approach here.' —https://www.sciencedirect.com/science/article/abs/pii/S2352152X18301385?viaIn order to have online first algo, the parametrization needs to be done online!.

4.0.1 Outlook

Promising articles and/or methods for online parameter estimation:

online parameter estimation from the ARX model tran2017state
[Wang et al. 2021] xia2018online

Appendix

Stylefile

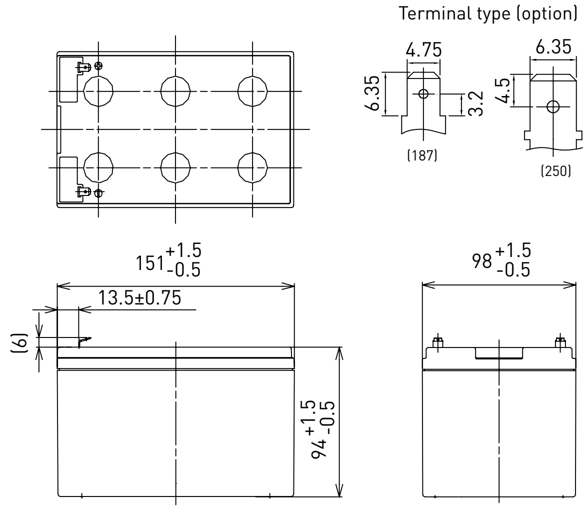
Die Styledatei für diese Abschlussarbeit ist `bhtThesis.sty`, die in der Archivdatei vorliegt. Diese muss von LaTeX auffindbar sein, muss also in einem LaTeX bekannten Ordner liegen:

- Ubuntu-Linux: `$HOME/texmf/tex/latex/bhtThesis/bhtThesis.sty`
- MikTeX: `c:\localtexmf\tex\latex\bhtThesis/bhtThesis.sty`

LC-CA1212P

FOR MAIN POWER SUPPLIES.
CYCLE LONG-LIFE TYPE

DIMENSIONS (MM)



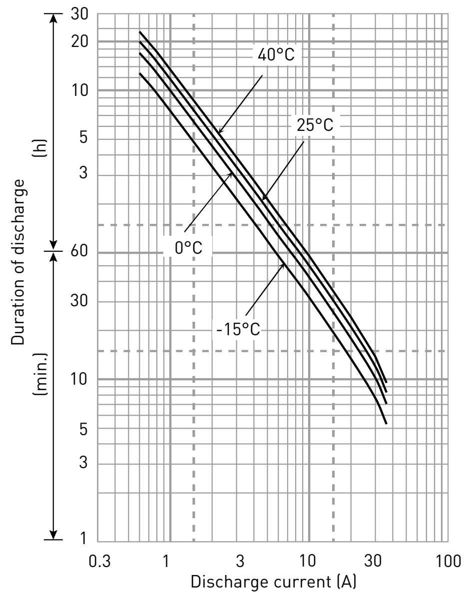
Contents indicated (including the recycle marking, etc.) are subject to change without notice.

Battery case resin: standard (UL94 HB)

SPECIFICATIONS

Name		LC-CA1212P/P1
Nominal voltage		12V
Nominal capacity (20 hour rate)		12Ah
Dimensions	Length	151mm
	Width	98mm
	Height	100mm
Approx. mass		3.80kg
Terminal		Faston 187/250
Capacity (25°C)	20 hour rate	12.0Ah
	10 hour rate	11.0Ah
	3 hour rate	9.3Ah
	1 hour rate	8.5Ah
Impedance	Fully charged battery (25°C)	15mΩ
Temperature dependency of capacity (20 hour rate)	40°C	102%
	25°C	100%
	0°C	85%
	-15°C	65%
Self-discharge (25°C)	After 3 month	91%
	After 6 month	82%
	After 12 month	64%

DURATION OF DISCHARGE VS. DISCHARGE CURRENT



WATT TABLE (25°C)

(Wattage/battery)

Cut-off	3min.	5min.	10min.	15min.	20min.	30min.	45min.	1h	1.5h	2h	3h	4h	5h	6h	10h	20h
9.6V	679	559	384	298	247	183	137	105	70.3	54.5	38.1	28.8	24.1	21.7	13.3	7.22
9.9V	649	537	373	288	241	177	135	104	69.9	54.2	37.8	28.8	24.1	21.7	13.3	7.22
10.2V	607	506	363	282	235	177	134	102	69.1	53.9	37.5	28.8	24.0	21.6	13.2	7.21
10.5V	556	475	343	271	231	172	133	100	68.5	53.3	36.9	28.7	24.0	21.6	13.2	7.20
10.8V	495	434	321	261	225	166	123	98	66.1	52.1	36.3	28.4	23.8	21.5	13.1	7.18

AMPERE TABLE (25°C)

(Ampere/battery)

Cut-off	3min.	5min.	10min.	15min.	20min.	30min.	45min.	1h	1.5h	2h	3h	4h	5h	6h	10h	20h
9.6V	61.1	50.1	34.3	25.9	21.3	15.6	11.7	8.90	5.95	4.60	3.20	2.41	2.01	1.81	1.11	0.602
9.9V	58.4	48.2	33.3	25.0	20.8	15.1	11.5	8.80	5.92	4.58	3.18	2.41	2.01	1.81	1.11	0.602
10.2V	54.6	45.4	32.4	24.5	20.3	15.1	11.4	8.70	5.85	4.55	3.15	2.41	2.00	1.80	1.10	0.601
10.5V	50.0	42.6	30.6	23.6	19.9	14.7	11.3	8.50	5.80	4.50	3.10	2.40	2.00	1.80	1.10	0.600
10.8V	44.5	38.9	28.7	22.7	19.4	14.2	10.5	8.30	5.60	4.40	3.05	2.38	1.99	1.79	1.09	0.598

All mentioned values are average values

LC-CA1212P

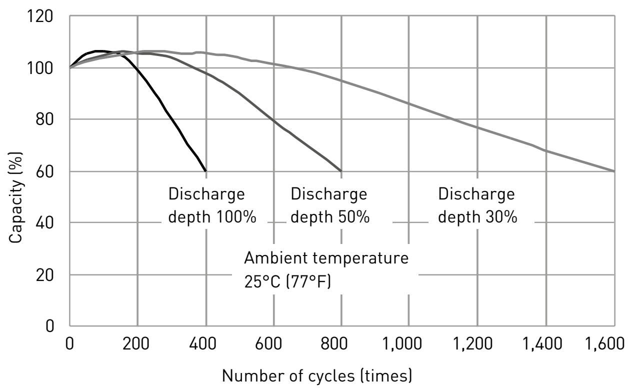
CHARGING METHOD (25°C)

Cycle use	Control voltage 14.5V - 14.9V Initial current 4.8A or smaller

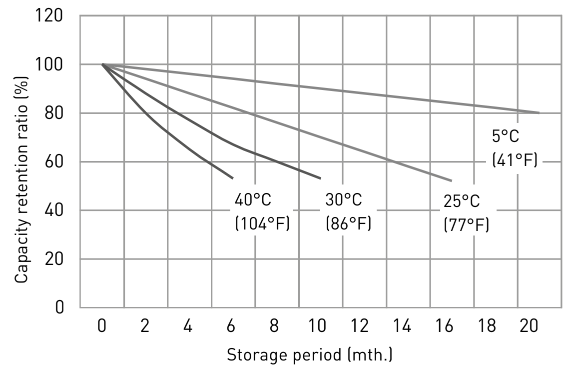
CUT-OFF VOLTAGE

Discharge current	0.6A - 2.4A	2.4A - 6A	6A - 12A	12A - 24A	24A - 36A
Cut-off voltage	10.5V	10.2V	9.9V	9.3V	8.7V

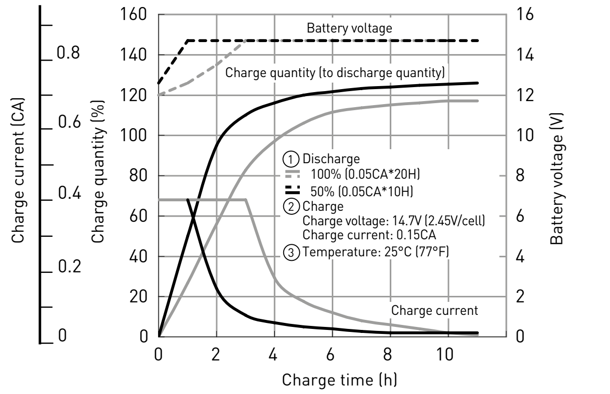
CYCLE LIFE VS. DEPTH OF DISCHARGE (ACC. IEC 61056)



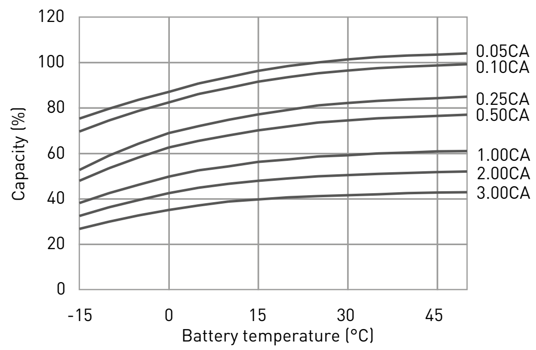
RESIDUAL CAPACITY TEST RESULT



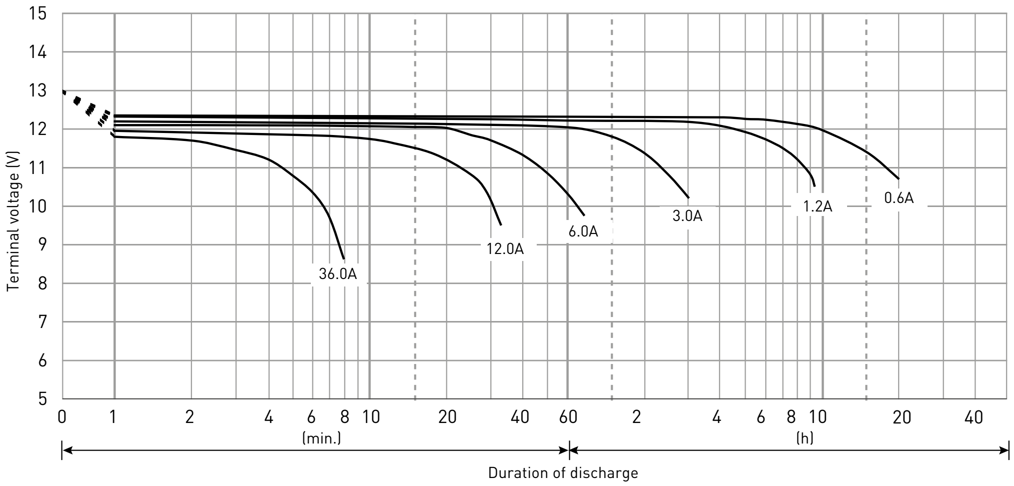
CONSTANT-VOLTAGE CONSTANT-CURRENT CHARGE CHARACTERISTICS FOR CYCLE USE



DISCHARGE CAPACITY BY TEMPERATURE AND BY DISCHARGE CURRENT



DISCHARGE CHARACTERISTICS



The data in this document are for descriptive purposes only and are not intended to make or imply any guarantee or warranty. Regarding handling and safety please consult our VRLA technical handbook chapter 1.

Beispieldokument

Dieses Dokument befindet sich im Unterordner `tryout` des zip-files. Sie können diese Dateien in einen Ordner kopieren, in dem Sie schliesslich arbeiten werden. Die Dateien sind die folgenden

- `abstract_de.tex` Kurzfassung in deutscher Sprache
- `abstract_en.tex` Kurzfassung in englischer Sprache
- `anhang.tex` der Anhang
- `bhtThesis.bib` beinhaltet die zu zitierenden Literaturstellen und wird von bibTEXausgewertet
- `main.pdf` ist die Ausgabendatei mit der Druckvorlage
- `main.tex` beinhaltet das Hauptdokument
- `makefile` realsiert das automatische mehrfache Übersetzen, hierfür muss `make` auf dem System installiert sein.
- `myapalike.bst` beinhaltet die Formatierung für das Literaturverzeichnis
- `personalMacros.tex` kann einzelne, persönliche Macros beinhalten, die das Schreiben erleichtern
- `titelseiten.tex` realisiert alle Seiten bis zum Beginn des ersten Abschnittes
- Ordner `pictures`
  - `BHT-Logo-Basis.eps`
  - `BHT-Logo-Basis.pdf`
- Ordner `kapitel1`
  - `ch1.tex` Quelltext des Kapitel 1
  - Ordner `pictures`
    - `schaltbild.pdf`
- Ordner `kapitel2`
  - `ch2.tex` Quelltext des Kapitel 2
  - Ordner `pictures`
    - leer

Bibliography

[Espedal et al. 2021] Espedal, I. B., Jinasena, A., Burheim, O. S., und Lamb, J. J., Current trends for state-of-charge (soc) estimation in lithium-ion battery electric vehicles. *Energies*, 14(11):3284.

[Hussein und Batarseh 2011] Hussein, A. A.-H. und Batarseh, I., An overview of generic battery models. In *2011 IEEE Power and Energy Society General Meeting*, Seiten 1–6. IEEE.

[Khan und Choi 2016] Khan, A. B. und Choi, W., Hysteresis modeling of the sealed flooded lead acid battery for soc estimation. In *Proceedings of the KIPE Conference*, Seiten 309–310. The Korean Institute of Power Electronics.

[Miguel et al. 2021] Miguel, E., Plett, G. L., Trimboli, M. S., Lopetegi, I., Oca, L., Iraola, U., und Bekaert, E., Electrochemical model and sigma point kalman filter based online oriented battery model. *IEEE Access*, 9:98072–98090.

[Rzepka et al. 2021] Rzepka, B., Bischof, S., und Blank, T., Implementing an extended kalman filter for soc estimation of a li-ion battery with hysteresis: A step-by-step guide. *Energies*, 14(13):3733.

[Schons 2021] Schons, E., Development and validation of a SoC algorithm for solarbatterychargecontroller Mppt-1210-hus by Libre.solar with lead acid.

[Tjandra et al. 2014] Tjandra, R., Thanagasundram, S., Tseng, K. J., und Jossen, A., Improved lithium-ion battery model with hysteresis effect. In *2014 IEEE Transportation Electrification Conference and Expo (ITEC)*, Seiten 1–8.

[Wang et al. 2021] Wang, Y., Huang, F., Pan, B., Li, Y., und Liu, B., Augmented system model-based online collaborative determination of lead–acid battery states for energy management of vehicles. *Measurement and Control*, 54(1-2):88–101.