



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 October 22, 2021

Gutachter

Gutachter1 Berlin University of Applied Sciences and Technology

Contents

1	Introduction	3
1.1	why Kalman, why EKF?	3
2	Implementation	5
2.1	Hardware Setup for validation	5
2.2	EKF Implementation	5
2.2.1	Definition Mathematics Equations	7
2.2.2	Kalman-SoC	9
3	Validation	11
3.1	Graphs with Matplotlib or plotly	11
4	Outlook	13
A	Appendix	15
	Bibliography	17

List of Figures

2.1	Setup to verify SoC Algorithm	6
3.1	A plot created with PythonTeX and Matplotlib	11
3.2	A plot created with PythonTeX and Plotly	12

Chapter 1

Introduction

TODO convert document to english. for examples Kapitel 1 needs to be Chapter one, so does Inhaltverzeichnis and so
(4d) (mostly offline)

1.1 why Kalman, why EKF?

[Espedal et al. 2021]

The kalman filter is a great tool to base a SoC algorithms on, as it is clearly defined by mathematical theories more than by a arbitrary algorithm designed by a programmer. One needs to use a extend kalman filter as the problem to solve is non linear, TODO explain why it is non linear.

Other methods are based on auto regressive exogenous (ARX) model or a combination of it and the Kalman Filter. TODO reference to resources.

Chapter 2

Implementation

2.1 Hardware Setup for validation

TODO Migrate from the other document.

TODO What resolution is necessary.

2.2 EKF Implementation

date: (4d)

A good step by step guide to implement the extended kalman filter is [Rzepka et al. 2021].

To design an extended kalman filter algorithm three steps are necessary: 1.The control-input model and the observation model need to be well designed. 2.The constants inside the models expressed as matrices need to be defined, which is as well known as parametrization of the model. 3.The model needs to be implemented as code running on embedded systems.

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.

Particularly challenging is the parametrization, which is typically done offline ¹ and requires quite advanced equipment. 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

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'

¹not 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

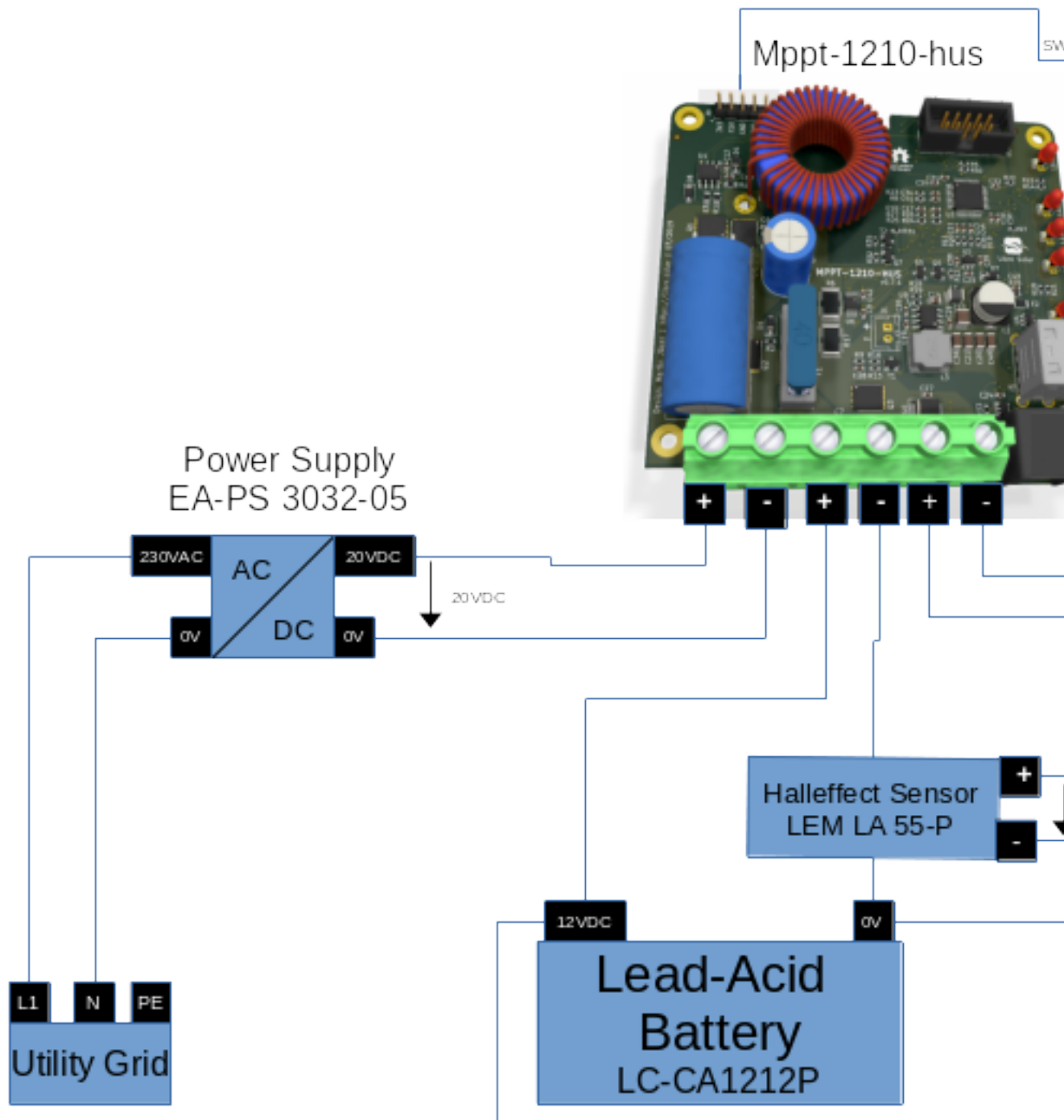


Figure 2.1: Setup to verify SoC Algorithm

'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!.

As the OCV curve is non-linear the kalman filter can not be used.

state transition and observation models

2.2.1 Definition Mathematics Equations

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

Most general form of state observer equations:

$$x_{k+1} = Ax_k + Bu_k \text{ (state observer equation)}$$

$$y_k = Cx_k + dDu_k \text{ (output equation)}$$

$u_k \equiv i_k$ 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 .

$$A = I \text{ identity matrix f.i } I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} -\frac{\Delta t}{Q_C} & 0 \\ 0 & 1 \end{bmatrix} \text{ is the control-input model, deduced from } f(x_k, i_k, \Delta t) = x_k - \frac{1}{Q_C} \int_0^{\Delta t} i_k dk$$

$C = [0, -R_1, -R_2, \dots, M] = H(x_k)$ and $H(x_k)$ is the **observation model**, which maps the state space into the observed space and TODO understand the relationship to $h(x_k)$

$$D = R_0$$

Most general form of Kalman Filter equations:

$$x_{k+1} = f(x_k, u_k) + w_k \text{ (State Space equation)}$$

$$z_k = h(x_k) + v_k \text{ (Output equation)}$$

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

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.

System's dynamic model

The $f()$ function is defined as the $f(x_k, i_k, \Delta t) = x_k - \frac{1}{Q_C} \int_0^{\Delta t} i_k dk$ 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 $h(x_k)$ 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: $v_k = D \text{OCV}(z_k) + Cx_k + Di_k$ with $C = [0, -R_1, -R_2, \dots, M]$ 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 $\hat{\mathbf{x}}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$. Moreover an estimate of the covariance P_{k+1} 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) $\mathbf{P}_{k+1} = \mathbf{B}_k \mathbf{P}_k \mathbf{B}_k^T + \mathbf{Q}_k$

In the **update** step

$\mathbf{K}_k = \mathbf{P}_{k+1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k+1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$ 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

$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$

$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$

Update

$\tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})$

$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k$

$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}$

$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$

$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$

2.2.2 Kalman-SoC

A Fork of Okra-Solar Algorithm.

Features:

- Works without the input of a Equivalent Circuit Model (ECM) specific to the physical battery, which would need to be parameterized doing advanced measurements during charging and discharging of the battery.

- Inputs: Current and Voltage Measurements and OCV Lookup Table

- Outputs: SoC in %Wh

TODO Discuss the Code Snippets? But Some Code in Appendix or reference Github Repo only?

Chapter 3

Validation

3.1 Graphs with Matplotlib or plotly

Python says “Hello we got it running finally!! YEahjjjjjo ”

```
>>> x = 400  
>>> x = x*2  
>>> y = 10
```

The variable is $x = 800$. The Result of $y*2$ is $y = 10$

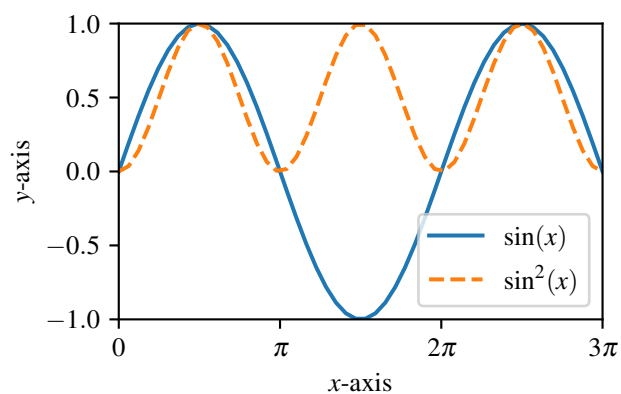
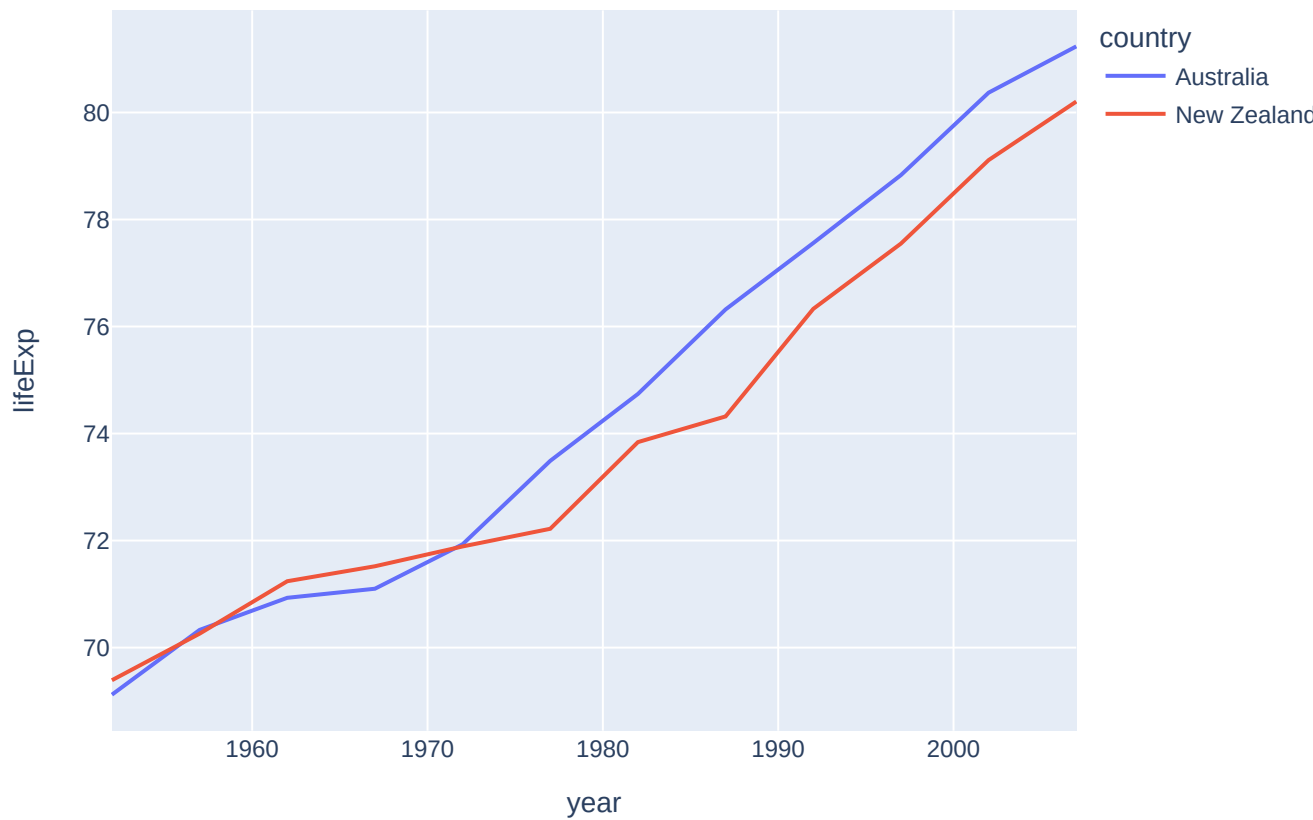


Figure 3.1: A plot created with PythonTeX and Matplotlib

Discuss results. (4d)



Loading [MathJax]/extensions/MathMenu.js

Figure 3.2: A plot created with PythonTeX and Plotly

Chapter 4

Outlook

Promising articles and/or methods for online parameter estimation:

online parameter estimation from the ARX model [?]

[?] [?]

Appendix A

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`

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 \bibTeX ausgewertet
 - `main.pdf` ist die Ausgabendatei mit der Druckvorlage
 - `main.tex` beinhaltet das Hauptdokument
 - `makefile` realisiert 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.
- [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.
-