CHALMERS
UNIVERSITY OF TECHNOLOGY

Adaptive estimation of battery state of charge

Master's thesis in System Control and Mechatronics programme

PEIXI GONG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017

Adaptive estimation of battery state of charge

PEIXI GONG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of System Control and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2017

Adaptive estimation of battery state of charge
PEIXI GONG

Supervisor: Yongwei Gao, China Euro Vehicle Technology AB
Examiner: Nikolce Murgovski, Electrical engineering

Master's Thesis 2017: EX088/2017
Department of Electrical Engineering
Division of System Control and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 70 025 2932

Typeset in LaTeX
Gothenburg, Sweden 2017

Adaptive estimation of battery state of charge
PEIXI GONG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Batteries are one of the main energy sources for Hybrid Electric Vehicles (*HEV*). Battery control unit (*BCU*) has been applied broadly to monitor and control the performance of batteries for modern cars. One main functionality of *BCU* is to provide information of state of charge (*SOC*) for the driver. The conventional method to estimate *SOC* is coulomb counting, which has several drawbacks. This thesis presents an adaptive *SOC* estimation method that is based on real time parameter identification and close loop estimation. The main theories behind this algorithm are recursive least squares (*RLS*) and Kalam filter. The validation concludes a result of maximum *SOC* estimation residual of 1.6%. With this estimation method be implemented in the future, drivers can have better understanding of the performance of the vehicle battery, and the engineers can design more trustable battery control strategies. The programming of the algorithm was in *Matlab/Simulink*, the tests were done in *Simulink* and *CANoe*.

Keywords: Hybrid electric vehicle, battery, battery ageing, adaptive, state of charge, system identification, recursive least squares, Kalman filter.

Acknowledgements

This Master's thesis is made as a corporation with China Euro Vehicle Technology (*CEVT*) in Göteborg. Here, I wish to express my sincere gratitude to everyone who directly or indirectly help me to accomplish this thesis.

First of all, I would like to appreciate all staff working at computer aided engineering (*CAE*) energy department. I want to thank Sofia Ore, the manager of the computer aided engineering *CAE* energy department, who made everything happen in the first place and arranged the best work environment and equipments. Yongwei Gao, the *CAE* engineer and the supervisor of this thesis, has been supporting my work for the whole project period. A great thank to Michael Palander, the *CAE* engineer and a truly concerned friend, has arranged training courses of Maplesoft's *MapleSim* and Vector's *CANoe* which are both very valuable for my future career. Thanks to Yuwei Zhou, my informal partner who is truly an expert in cars, with his help I avoided so much unnecessary labours and also learned so much knowledge about cars. A special thank to Jörgen Claesson who is the field application engineer from Vector Software, Jörgen was the mentor of the *CANoe* courses. From the beginning as a freshman in *CANoe* till now, I have gain a lot of knowledge in control area network (*CAN*) and hardware in loop (*HIL*) thanks to Jörgen. I also want to thank, Anders Werner, Jens Larsen, Hanjie Xu, Meng Yao, Jonatan Rydberg, Patrik Nilsson, as well as Edo Drenth, for participating in my presentation and offering many valuable feedbacks.

I want to thank the other Master students who have been doing their thesis at *CEVT*. Pedram Hajigholi, Wang Han, Changxin Yang, Mohammed Arafath, we have spent months working together and established good friendships.

To be the most important person in my appreciations, professor Nikolce Murgovski, from Electrical Engineering, has been helping me with this degree project. As the examiner of this thesis, professor Nikolce has always been patiently listening to my regular presentations and giving professional advices. Also I would like to appreciate professor Jonas Sjöberg from Electrical Engineering who is the teacher of System Identification. By taking this course I learned the basic theories that have been used in this project. Thanks to Björn Fridholm who is the author of the research paper that I mostly referring to, during the several short meetings with Björn I have received many useful directions.

In the end, I would like to thank my family and friends whose names may not be listed. Studying in a country that is far away from home is a challenge, you were always together with me when I was very down and lost motivation. Thank you all for the great support.

Peixi Gong, Gothenburg, August 2017

Contents

Nomenclature

Physics value

Q_{nom}	Nominal capacity
i_{cell}	Cell current
u_{cell}	Cell voltage/terminal voltage
η	Coulomb efficiency
u_c	RC circuit voltage
R	RC circuit resistance
C	RC circuit capacity
R_{ohm}	Impedance resistance
Δt	Sampling rate

Abbreviations

SOC	State of charge
OCV	Open circuit voltage
$NEDC$	New european driving cycle
NMC	Nickel Manganese Cobalt
RLS	Recursive least squares
AKF	Adaptive Kalman filter
EKF	Extended Kalman filter
CAN	Controller area networks
ECU	Electronic control unit
TCU	Transmission control unit
BCU	Battery control unit
FMI	Functional mock-up interface
FMU	Functional mock-up unit
DOF	Degrees of freedom
ZOH	Zero order hold
MSE	Mean squares error

List of Figures

1
Introduction

1.1 Background

More than 1.5 million hybrid electric vehicles (*HEV*) have been sold worldwide in 2012. The main markets of *HEV* are US, Japan and Europe. Over 400 000 *HEV*s have been sold in the US alone, 800 000 in Japan and 150 000 in Europe. From 2008 to 2010 in Japan, *HEV* sales had increased from 3% to 10% of the total car market. The 2011 earthquake and tsunami drastically impacted the car and battery production in Japan, but the sales bounced back in 2012 [1]. After that, the global sale for electric vehicles (*EV*) has more than doubled since 2014, following a 72% increase in 2015. 2016 saw an increase of 41% to reach nearly 8000 000 *EV* [2].

For *HEV* and *EV* that consume electricity as power resources, the availability for drivers to view the remaining energy of the battery has the same importance as the gasoline gauge. State of charge (*SOC*) is such an index that indicates this property. In addition, *SOC* is important data for analyzing the performance of battery. The knowledge of *SOC* can be analyzed in order to design rational battery control strategies to protect the battery, prevent overcharge/discharge, prolong the battery life as well as save energy [3]. However, the battery is a complicated electrochemical plant where no existing method is available to directly measure its chemical energy, which makes the estimation of *SOC* difficult [4].

There are various types of methods to estimate *SOC*. The most commonly used method is coulomb counting. This method counts the flow of the coulombs then integrates them over time. An arbitrary state of *SOC* is calculated by adding the coulomb integral to the known initial *SOC* [5]. Coulomb counting has two main drawbacks. The first is the estimation error increases at every cycle, and the second is the inflexibility to deal with battery ageing.

Many factors contributes to battery ageing. Some characterised specific factors are charging rate, discharging rate, temperature, pauses and *SOC* range [7]. The battery has two types of ageing, calender ageing and cycle ageing. Calender ageing corresponds to the irreversible proportion of lost capacity during storage, which leads to a decrease of the capacity. Cycle ageing is associated with the number of times the battery is charged or discharged. The battery's chemical and physical components deteriorate during its life time, which brings undesirable effects, i.e. the loss of capacity, faster temperature rise during operation, less charge acceptance,

higher internal resistance (R_{ohm}), lower voltage, and more frequent self-discharge [8].

Cumulative error occurs because of the impact of capacity loss. Coulomb counting requires recalibration of SOC at each time when the battery is fully charged. Nonetheless, the capacity loss makes the fully charged state difficult to correctly detect. After a long time in operation, the battery capacity will have a big discrepancy to its initial value, which causes the calculated SOC to be inaccurate [6]. Another problem is inflexibility: cycle ageing causes real battery parameters to vary over time. If the parameters in the mathematical model of battery stay fixed, the SOC calculation will gradually grow biased from the real value.

1.2 Scope and methods

The aim of the thesis is to design a SOC estimation method that works better than coulomb counting in mathematical accuracy aspect and adapts to parameters variation. In order to keep the simplicity of analysis and validation, the thesis focuses on two parameters which are the inner resistance and the time constant. For nominal capacity, its accurate value is usually found by look-up table.

To solve the problem, the thesis proposes a real time method to estimate SOC based on parameter identification. This method has the advantage of adaptability, which guarantees that the estimation of parameters captures the real parameters' dynamics. An equivalent circuit is used to model the complicated chemical battery ideally. The algorithm of parameters identification has two main parts. Part one has an open loop algorithm that implements recursive least squares (RLS) method to identify the impedance resistance, while the regressor data use cell current and cell voltage. Part two has a closed loop algorithm that implements adaptive Kalman filter (AKF) with extended Kalman filter (EKF) in parallel to identify the time constant, while the regressor data use cell current and a new defined signal.

The validation uses New European Driving Cycle ($NEDC$) data which is collected from IPG's *CarMaker*. The validation process has two phases. The first phase tests the algorithm only in *Simulink* in order to prove its basic functionality. The second phase tests the algorithm in a synchronised mode between *CarMaker* and *CANoe*, in order to test the algorithm's functionality in control area network (CAN) environment.

1.3 Limitations

The development of the algorithm uses only a very simple battery model which does not have high degrees of freedom (DOF) in parameters, which means it cannot reflect all the dynamic behaviours of a real battery. The data of open circuit voltage (OCV) and SOC relation is not detailed enough to accurately hold the discrepancy between charge and discharge. Several cancellation assumptions are made in the

algorithm, which makes the estimation accuracy highly related to the sampling rate and the well selection of the error model.

2

Cell model and theories

2.1 Battery cell equivalent circuit

Equivalent circuit is the circuit diagram usually used in electrical analysis, to describe the characteristics of a complicated physical circuit plant in a simple way. Since battery cell is an electrochemical plant, its main characteristics can be modeled by an equivalent circuit.

For the equivalent circuit of a battery cell, three important features need to be included:

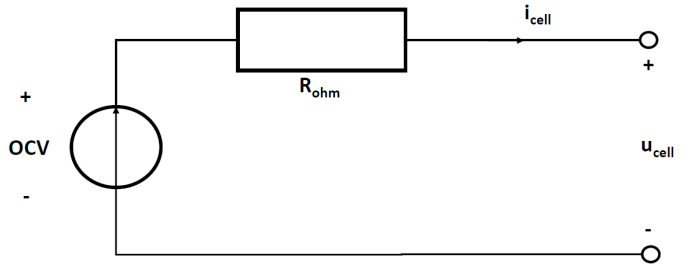
1. An ideal voltage source that represents the open circuit voltage (OCV),
2. The internal resistance,
3. The transient dynamic response during charge and discharge [9].

No current flow exists in the circuit when an ideal battery is unloaded, thus the cell has neither any type of energy consumption nor voltage drop. The battery in this condition is completely at rest, and its terminal voltage equals to OCV. See Figure 2.1a.

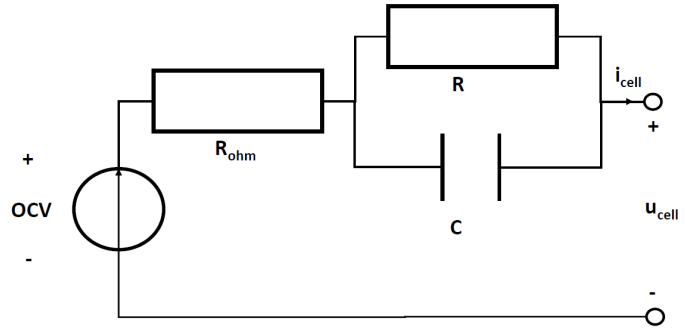
The internal resistance converts part of the electric energy to heat, which causes a voltage drop that results in an inequality between cell voltage and OCV. This effect is modeled by adding a resistance R_{ohm} to the circuit in series with the ideal voltage source. See Figure 2.1b.

During charge and discharge, the effect of cell voltage on cell current is not immediate, but delayed. The discussion of the reason for this delay in the chemical aspect is not part of the thesis. In a nutshell, the effect can be summarized by the following: during charge and discharge, the chemical components in the battery cell respond to slow changes because of the chemical reaction. When the load is removed instantaneously, it takes longer time for the cell voltage to restore to OCV. See Figure 2.1c.

(a) Battery circuit modeled as open circuit voltage.



(b) Battery circuit modeled with an OCV and an internal resistance connected in series.



(c) Battery circuit modeled with an OCV, inner resistance and an RC-circuit.

Figure 2.1: Three types of equivalent battery circuit models. Within this thesis, the model in (c) is used.

The equivalent circuit in Figure 2.1c has modeled the main characteristics of a battery cell, which has three parameters R_{ohm}, R and C. Since R and C appear together as a multiplication, it will later be replaced by the time constant τ. The parallel network of R and C is called RC-circuit, which models the transient behavior of the battery cell. The RC-circuit also indicates DOF of the model. For battery models with more complicated dynamic behaviors, more RC-circuits should be added to the basic circuit model. The extension of RC-circuits increases the DOF in parameters. Figure 2.2 gives an example equivalent circuit of two RC-circuits with five parameters.

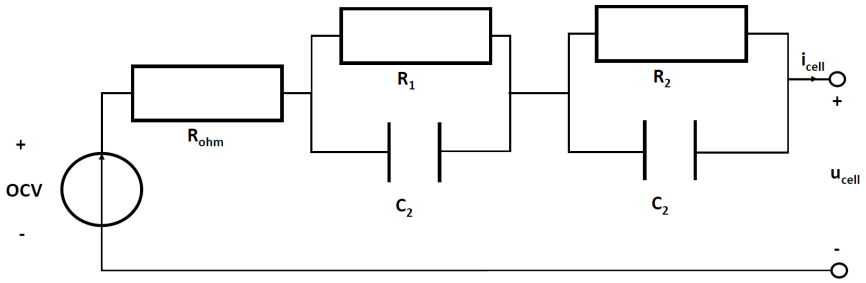


Figure 2.2: Battery circuit model that has two RC networks and an inner resistance connected in series, which has 5 parameters.

2.2 State of charge

State of charge is one of the most important information in battery, but its definition presents many different issues [10]. A generally used definition is the ratio of the battery's instantaneous capacity to the nominal capacity,

$$SOC(t) = \frac{Q(t)}{Q_{nom}}. \tag{2.1}$$

The nominal capacity determines the maximum amount of energy that can be stored in the battery, where the unit is Ah or mAh. This parameter is given by the battery manufacturer. Practically, Q_{nom} is a varying parameter due to operating condition and ageing, but the identification of which is not included in the thesis. Commonly Q_{nom} is updated by checking a look-up table in industrial applications. SOC indicates the remaining electric energy stored in the battery, where the definition shows that the fully charged cell has 100% SOC while the fully discharged cell has 0% SOC. Because the instantaneous capacity $Q(t)$ is an integral of cell current over time, obviously SOC's rate of change is

$$\dot{SOC} = \frac{i_{cell}(t)}{Q_{nom}}. \tag{2.2}$$

Cell current $i_{cell}(t)$ can either be positive or negative, which refers to as charge or discharge respectively. The value of SOC at time instant t with a known initial value at time instant t_0 is calculated by

$$SOC(t) = SOC(t_0) + \frac{1}{Q_{nom}} \int_{t_0}^{t} i_{cell}(\tau) d\tau. \tag{2.3}$$

In reality, the performance of charge and discharge is not perfect but with an energy loss which names Faraday loss. To describe Faraday loss, coulomb efficiency coefficient η is multiplied to the integral term, such that the update of SOC turns to be

$$SOC(t) = SOC(t_0) + \frac{\eta}{Q_{nom}} \int_{t_0}^{t} i_{cell}(\tau).d\tau. \tag{2.4}$$

Coulomb efficiency is a value close to unit one. In the remainder of this thesis, η is set as 1 in order to keep the simplicity of the model.

2.3 *OCV-SOC* data

A straightforward idea of finding *SOC* is to check the battery cell's instantaneous *OCV* and compare it with the maximum *OCV*. This is called voltage estimation method. However, to measure *OCV* is not convenient. Due to the transient behavior of battery cell, the terminal voltage becomes close to *OCV* only when battery is unmounted for a while and reaches its equilibrium state [11]. This phenomena makes it unrealistic to implement voltage estimation method during real driving.

By offline experiments, the data of *OCV* and *SOC* relation can be collected. Figure 2.3 demonstrates 41 data points that are collected from a *NMC* battery. *OCV* at fully charged state is about 4.2 volts. The second data point shows that *OCV* is about 3.4 volts when battery is almost empty. The first data point represents that *OCV* is 2.8 volts when *SOC* is completely zero, which has a great jump to the second point. However, the first data point is not important, because the information of *SOC* is not interesting when it is extremely low. Usually the driver will not use the battery to such a low level, otherwise the battery will soon be empty and the driver would be aware that the battery needs recharging. In practice, charge and discharge have similar distribution shape for *OCV-SOC* data but with a gap in between. This could be problematic in battery *SOC* estimations. So in this thesis, only the data for discharge is used.

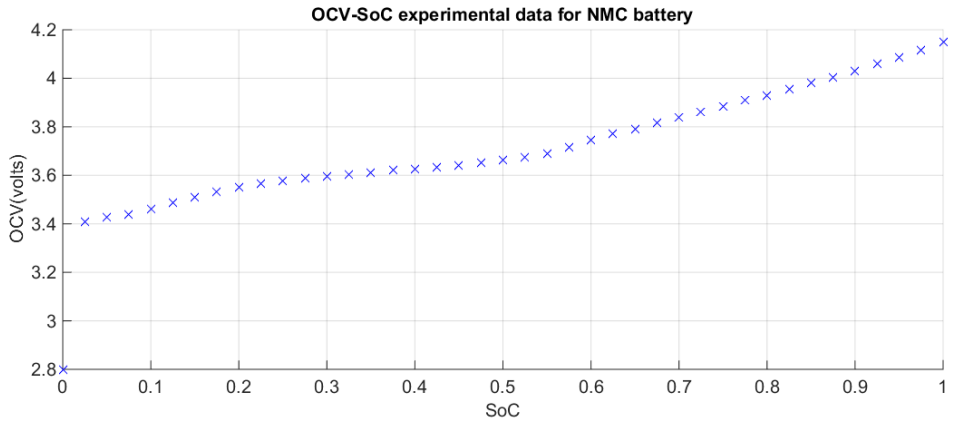


Figure 2.3: The collected *OCV-SOC* data from a *NMC* battery.

It can be concluded by observing Figure 2.3 that, a function $OCV = f(SOC)$ is exist to describe the *OCV* and *SOC* relation data. The function is affected by several factors where the main one is temperature. This is usually solved by referring to a look-up table of different operating conditions. The *NMC* battery's *OCV-SOC* data used here relates to the temperature between 22°C to 30°C. The data has very small variance in this range of temperature so the function can be considered independent from temperature.

2.4 State space model

The equivalent circuit in Figure 2.1c is frequently implemented to model the battery cell. Input of the model is cell current $i_{cell}(t)$. Output of the model is cell voltage $u_{cell}(t)$ which also names terminal voltage.

Naming the current that flows through R as $i_R(t)$, and the current that flows through C as $i_c(t)$. Since R and C are connected in parallel, the summation of their currents equals to cell current $i_{cell}(t)$,

$$i_R(t) + i_c(t) = i(t). \tag{2.5}$$

Further, $i_c(t) = C \cdot \dot{u}_c(t)$, which gives

$$i_R(t) + C \cdot \dot{u}_c(t) = i(t). \tag{2.6}$$

Rewrite the above equation to

$$\dot{u}_c(t) = -\frac{1}{C} i_R(t) + \frac{1}{C} i(t). \tag{2.7}$$

Because $u_c(t) = u_R(t)$, and $i_R(t) = u_R(t)/R$, then

$$\dot{u}_c(t) = -\frac{1}{R \cdot C} u_c(t) + \frac{1}{C} i(t). \tag{2.8}$$

If choose $SOC(t)$ and $u_c(t)$ as the states, then (2.8) and (2.2) together define the update of states. Write them together

$$\begin{aligned} \dot{SOC}(t) &= \frac{1}{Q_{nom}} i(t), \\ \dot{u}_c(t) &= \frac{1}{R \cdot C}(t) + \frac{1}{C} i(t). \end{aligned} \tag{2.9}$$

Thus the matrices representation is

$$\begin{bmatrix} \dot{u}_c(t) \\ \dot{SOC}(t) \end{bmatrix} = \begin{bmatrix} -1/(R \cdot C) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_c(t) \\ SOC(t) \end{bmatrix} + \begin{bmatrix} 1/C \\ 1/Q_{nom} \end{bmatrix} i(t) \tag{2.10}$$

System output is

$$u_{cell}(t) = u_c(t) + f(SOC(t)) + R_{ohm} i_{cell}(t). \tag{2.11}$$

2.5 Recursive least squares

Recursive least squares method is widely used for real time parameter identification. *RLS* is a least squares error method when implemented in real time estimation conditions. The parameters estimation is driven by minimizing the estimation residual criterion while a decaying window coefficient controls the usage of data. The estimation residual criterion is

$$E = \sum_{i=0}^{n} \lambda^{n-i}(y(i) - \hat{y}(i))^2, \tag{2.12}$$

where λ is the forgetting factor that defines the window of utilizing past data, n is the total number of samples, i is the sample instants index. The model output is $y(i)$, the estimation output is $\hat{y}(i)$. The estimation output $\hat{y}(k)$ is usually calculated from regression model.

In discrete time and assuming a noise free environment, many physical processes which are linear in parameters can be expressed by regression equation of

$$y(k) = a_1 y(k-1) + \cdots + a_{na} y(k-na) + b_{nk} u(k-nk) + \cdots + b_{nk+nb} u(k-nk-nb), \tag{2.13}$$

where nk is the delay between input and output, na is the number of past output, nb is the number of past input. Because the equation is linear in parameters, it can be instead written as a coefficient vector

$$\theta = \begin{bmatrix} a_1 \\ \vdots \\ a_{na} \\ b_{nk} \\ \vdots \\ b_{nk+nb} \end{bmatrix} \tag{2.14}$$

that multiplies with a regressor vector

$$\varphi(k-1) = \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-na) \\ u(k-nk) \\ \vdots \\ u(k-nk-nb) \end{bmatrix}, \tag{2.15}$$

such that

$$y(k) = \theta^T \varphi(k-1). \tag{2.16}$$

If the system is multi-input-multi-output, the vectors should be replaced by matrices. The regressor vector contains past information about both input and output.

The parenthesis of φ means that it has information until time instant $(k-1)$.

This model can be implemented to estimate the output, by replacing θ with $\hat{\theta}(k)$. The reason for $\hat{\theta}$ to be time variant is that parameters update at each step. The estimation of output is

$$\begin{aligned}\hat{y}(k) &= \begin{bmatrix}\hat{a}_1(k) & \cdots & \hat{a}_{na}(k) & \hat{b}_{nk}(k) & \cdots & \hat{b}_{nk+nb}(k)\end{bmatrix} \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-na) \\ u(k-nk) \\ \vdots \\ u(k-nk-nb)\end{bmatrix} \\ &= \hat{\theta}^T(k)\varphi(k-1).\end{aligned} \tag{2.17}$$

The forgetting factor λ defines a decaying window that has different weights on past data samples. In this case, λ is a value defined between (0,1] but usually set to very close to 1. When λ is not 1, the data that is closer to current time instant will have more weights in the residual criterion, while the far past data's weights decrease exponentially. This decreasing makes the old data much less influencing the estimation. If λ is set to 1, the weights on all data are the same. In principle, a large λ has a comparatively moderate window such that the estimation captures long term properties, so the dynamic of estimated parameters will be smooth. On the contrary, a small λ has a rapid decaying window that makes the estimation focus on short term data, which makes the estimated parameters capture real time variations of the system that might change in a rapid way.

The recursive version of *RLS* algorithm that is used in real time estimation has one frame update of the following formulas,

$$\begin{aligned}\varepsilon(k) &= y(k) - \hat{\theta}^T(k-1)\varphi(k), \\ K(k) &= \frac{\lambda^{-1}P(k-1)\varphi(k)}{1+\lambda^{-1}\varphi^T(k)P(k-1)\varphi(k)}, \\ P(k) &= \lambda^{-1}P(k-1) - \lambda^{-1}K(k)\varphi^T(k)P(k-1), \\ \hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\varepsilon(k),\end{aligned} \tag{2.18}$$

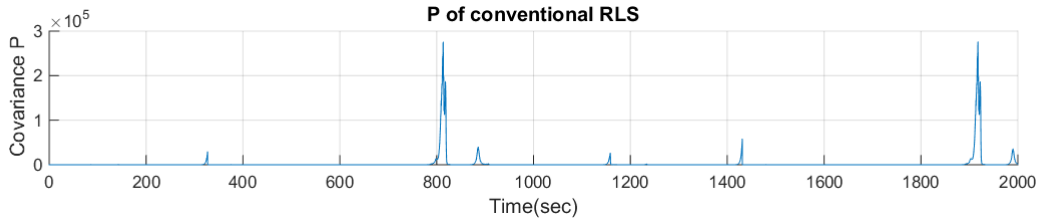
where K is the parameter update gain, P is the parameter covariance. To trigger the algorithm, initial guess of $\hat{\theta}(0)$ and $P(0)$ are required. The rule of choosing value is, if the guess of $\hat{\theta}(0)$ has good confidence then $P(0)$ is set to a small value to avoid transient overshoot, otherwise $P(0)$ is set to a large value to make the convergence faster.

2.6 *RLS* with bound

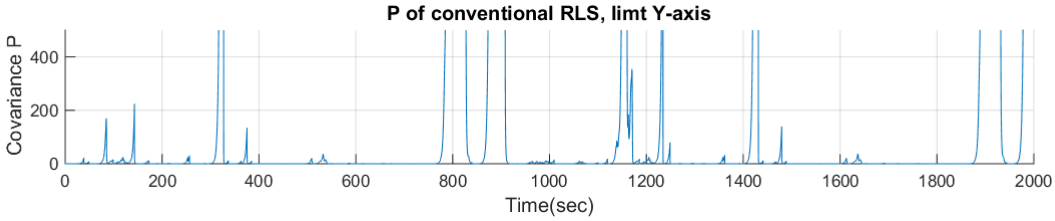
The identification of system parameters highly depends on the quality of input signal. The property that indicates the ability of a sequence of data to accurately identify parameters is called the signal's excitation [13]. In some cases the input signal contains very little energy such that $K(k)$ in (2.18) is a value very close to zero. This results in that the update of $P(k)$ almost appears to be

$$P(k) = \lambda^{-1} P(k-1), \tag{2.19}$$

which makes $P(k)$ change exponentially if the signal is continuously bad excited for a while. See Figure 2.4.



(a) Covariance P grows largely when the signal is continuously bad excited for a while.



(b) A closer scaled look of covariance P, it can be seen that despite the extreme large spikes, there are small spikes.

Figure 2.4: RLS with bound

When once again the signal turns to be well excited, $P(k)$ becomes already very large such that it leads to a sudden jump of $\hat{\theta}(k)$ towards the correct value. This is shown in Figure 2.5.

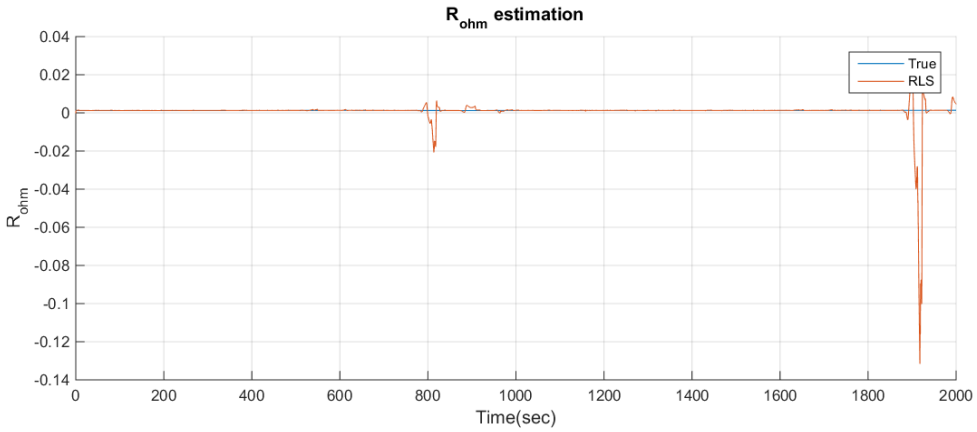


Figure 2.5: The sudden change of the estimation of impedance resistance.

A modification is added to the conventional *RLS* algorithm in order to avoid the suddenly rapid change of estimation, which is to set a bound value for $P(k)$. If $P(k)$ is less than the bound value, it is kept unchanged. Otherwise $W(k)$ is used to replace it. The modified version of *RLS* is

$$\begin{aligned}
\varphi(k) &= y(k) - \hat{\theta}^T(k-1)\varphi(k), \\
K(k) &= \frac{\lambda^{-1}P(k-1)\varphi(k)}{1+\lambda^{-1}\varphi^T(k)P(k-1)\varphi(k)}, \\
W(k) &= P(k-1) - K(k)\varphi^T(k)P(k-1), \\
P(k) &= \begin{cases} W(k)/\lambda(k), \text{if } W(k)/\lambda(k) \leq \text{bound}, \\ W(k), \text{if otherwise.} \end{cases}
\end{aligned} \tag{2.20}$$

2.7 Kalman filter

Kalman filter utilizes system input and output signals to estimate the inner states of the model. Original Kalman filter is only available for linear models. Extended Kalman filter is the version for nonlinear models which must be linearized first before implementing the algorithm.

The system states update is modeled by equation

$$x(k+1) = Fx(k) + Bu(k) + w(k), \tag{2.21}$$

where F is the state transition matrix, B is the input-to-state matrix, w is the process noise. The process noise w is assumed to subject to Gaussian normal distribution where $w \sim N(0, R_w)$. R_w is the covariance matrix.

The output is modeled by equation

$$z(k) = Hx(k) + Du(k) + v(k), \tag{2.22}$$

where H is the observation matrix, D is the input-to-output matrix, and v is the observation noise. The observation noise v is also assumed as Gaussian normal distributed where $v \sim N(0, R_v)$. R_v is the covariance matrix.

The concept of Kalman filter is to take one step ahead estimation on the states to generate *a priori* guess. Afterwards, based on the output estimation residual together with the Kalman gain to construct *a posteriori* states estimation.

The recursive algorithm of Kalman filter in real time has one frame updates of the following formulas,

$$\begin{aligned}
\varepsilon(k) &= z(k) - (H\hat{x}(k|k-1) + Du(k)),\\
\hat{x}(k|k-1) &= F\hat{x}(k-1|k-1) + Bu(k),\\
P(k|k-1) &= FP(k-1|k-1)F^T + R_w,\\
K(k) &= P(k|k-1)H^T(HP(k|k-1)H^T + R_v)^{-1},\\
\hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)\varepsilon(k),\\
P(k|k) &= (I - K(k)H)P(k|k-1).
\end{aligned} \tag{2.23}$$

Kalman filter can also be utilized to estimate system parameters. In such case, the parameters variation model replaces the states process model by

$$\theta(k+1) = \theta(k) + w(k), \tag{2.24}$$

and the output model is a regression term plus noise term

$$y(k) = \varphi^T(k)\theta(k) + v(k), \tag{2.25}$$

so the recursive parameter estimation algorithm is

$$\begin{aligned}
\varepsilon(k) &= y(k) - \varphi^T(k)\hat{\theta}(k),\\
K(k) &= P(k-1)\varphi(k)(R_v(k) + \varphi^T(k)P(k))^{-1},\\
P(k) &= P(k-1) - K(k)\varphi^T(k)P(k-1) + R_w(k),\\
\hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\varepsilon(k).
\end{aligned} \tag{2.26}$$

2.8 Adaptive Kalman filter

When implementing Kalman filter for parameters estimation, the covariance growth problem which is similar to *RLS* algorithm occurs when the input signal has bad excitation. In Kalman filter, the growth of $P(k)$ is not as severe as *RLS* since as $K(k)$ is close to zero the update of covariance $P(k)$ is

$$P(k) = P(k-1) + R_w(k). \tag{2.27}$$

But still, a sudden jump of estimated parameters are expected to occur if the bad excited signal continues for long time before once again turning back to well excitation.

The technique to avoid sudden change is to define R_w as a variable rather than a fixed value, such that it drives $P(k)$ to a user defined covariance P_d [14]. The definition of R_w in AKF is

$$R_w(k) = \frac{P_d\varphi(k)\varphi^T(k)P_d}{R_v(k) + \varphi^T(k)P_d\varphi(k)}. \tag{2.28}$$

3

Parameters and SOC estimation

3.1 Discretization of the model

For the necessity of computerized calculation, discrete time model must be derived. Zero-order hold (ZOH) with sampling interval Δt is used as the discretization method [12].

First to discretize $SOC(t)$. The integral of cell current in sample interval Δt is approximately $\Delta t \cdot i_{cell}(k)/Q_{nom}$, where k is the sample index. Thus the update of SOC for each sampling step is

$$SOC(k+1) = SOC(k) + \frac{\Delta t}{Q_{nom}} i_{cell}(k). \tag{3.1}$$

Second to discretize $u_c(t)$. The consequence is

$$u_c(k+1) = e^{-\frac{\Delta t}{R \cdot C}} u_c(k) + R(1 - e^{-\frac{\Delta t}{R \cdot C}}) i_{cell}(k). \tag{3.2}$$

Writing the states update and system output together,

$$\begin{aligned} u_c(k+1) &= e^{-\frac{\Delta t}{R \cdot C}} u_c(k) + R(1 - e^{-\frac{\Delta t}{R \cdot C}}) i_{cell}(k), \\ SOC(k+1) &= SOC(k) + \frac{\Delta t}{Q_{nom}} i_{cell}(k), \\ u_{cell}(k) &= u_c(k) + f(SOC(k)) + R_{ohm} i_{cell}(k). \end{aligned} \tag{3.3}$$

In these equations, $R \cdot C$ is called the time constant, which is related to the system transient response, and u_c is the voltage of RC-circuit. The matrices representation of the model is

$$\begin{bmatrix} u_c(k+1) \\ SOC(k+1) \end{bmatrix} = \begin{bmatrix} e^{-\frac{\Delta t}{R \cdot C}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_c(k) \\ SOC(k) \end{bmatrix} + \begin{bmatrix} R(1 - e^{-\frac{\Delta t}{R \cdot C}}) \\ \frac{\Delta t}{Q_{nom}} \end{bmatrix} i_{cell}(k), \tag{3.4}$$

$$u_{cell}(k) = u_c(k) + f(SOC(k)) + R_{ohm} i_{cell}(k). \tag{3.5}$$

The output cannot be straightforwardly written to as multiplication of matricies because the OCV term is a nonlinear function of SOC.

For the model that has two RC-circuits, the expression is

$$\begin{bmatrix} u_{c1}(k+1) \\ u_{c2}(k+1) \\ SOC(k+1) \end{bmatrix} = \begin{bmatrix} e^{-\frac{\Delta t}{R_1 \cdot C_1}} & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{R_2 \cdot C_2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{c1}(k) \\ u_{c2}(k) \\ SOC(k) \end{bmatrix} + \begin{bmatrix} R_1(1 - e^{-\frac{\Delta t}{R_1 \cdot C_1}}) \\ R_2(1 - e^{-\frac{\Delta t}{R_2 \cdot C_2}}) \\ \frac{\Delta t}{Q_{nom}} \end{bmatrix} i_{cell}(k), \tag{3.6}$$

$$u_{cell}(k) = u_{c1}(k) + u_{c2}(k) + f(SOC(k)) + R_{ohm} i_{cell}(k). \tag{3.7}$$

Models with more RC-circuits are more complex and have more DOF in parameters that improves the parameters estimation. The rule of selecting a proper model order is to ensure the estimation accuracy while preferring a model that is as simple as possible.

For single RC-circuit model, there are two unknown parameters R_{ohm} and τ. Nominal capacity Q_{nom} comes from the battery manufacturer. Sampling rate Δt is determined by the user.

3.2 Impedance resistance estimation

In (3.4), symbol τ is used to represent the time constant and assume the discrete time system has enough fast sampling rate. Thus Δt is a value close to zero that makes $e^{-\Delta t/\tau}$ close to 1 and $R(1 - e^{-\Delta t/\tau})$ close to 0, which leads to a fairly slow update of u_c. Similarly, SOC varies slowly under fast sampling rate, additionally because Q_{nom} is a much larger value in magnitude compared to i_{cell}.

With the above assumptions, u_c and OCV are nearly constant at two adjacent time instants. A new formula is constructed by subtracting (3.5) for two adjacent instants,

$$\begin{aligned} u_{cell}(k) - u_{cell}(k-1) &= R_{ohm}(i_{cell}(k) - i_{cell}(k-1)), \\ \Delta u_{cell}(k) &= R_{ohm} \Delta i_{cell}(k), \\ \Delta u_{cell}(k) &= \theta_{ohm} \varphi_{ohm}(k), \end{aligned} \tag{3.8}$$

where

$$\begin{aligned} \theta_{ohm} &= \left[R_{ohm} \right], \\ \varphi_{ohm}(k) &= \left[\Delta i_{cell}(k) \right]. \end{aligned} \tag{3.9}$$

Therefore $\varphi_{ohm}(k)$ contains all available data at time instant k, and RLS can be implemented to estimate θ_{ohm}. Due to the assumption that the OCV term and u_c term have been cancelled in (3.8), the estimation of R_{ohm} is always biased where the accuracy depends on sampling rate. The faster the sampling rate is, the lesser the influence on accuracy from the cancellation assumption is.

A special modification that smooths the estimation is to introduce a dead zone. Note that the regressor data used for estimation is $\Delta i_{cell}(k)$ which is the cell current at time k subtracted by the cell current at time $k-1$. If the signal to noise ratio is small then the noise may make the estimation drift away. A solution is to introduce a dead zone that allows parameters adaptation in RLS algorithm only when the change of current is larger than the dead zone. To some extent, the dead zone prevents the the noise's impacts on estimation. The usage of dead zone is of practical meaning, which filters away part of the effects of input noises and makes the whole estimation smoother, but this method requires some pre-knowledge of the noise.

3.3 Time constant estimation

3.3.1 Estimation model

Replacing $R \cdot C$ in (3.4) with symbol τ, and define

$$\begin{aligned} \alpha &= e^{-\frac{\Delta t}{\tau}}, \\ \beta &= R(1 - e^{-\frac{\Delta t}{\tau}}). \end{aligned} \tag{3.10}$$

In (3.5), move $u_c(k)$ and R_{ohm}ţ$oti_{cell}(k)$ to the left side of the equation,

$$u_{cell}(k) - \hat{R}_{ohm}(k) i_{cell}(k) - f(\hat{SOC}(k)) = u_c(k), \tag{3.11}$$

where $u_{cell}(k)$ and $i_{cell}(k)$ are measured signals, $\hat{R}_{ohm}(k)$ is the estimation of R_{ohm} at instant k, $\hat{SOC}(k)$ is the estimation of SOC at instant k. Define the left side of (3.11) as a new variable $y(k)$, yielding

$$y(k) = u_{cell}(k) - \hat{R}_{ohm}(k) i_{cell}(k) - f(\hat{SOC}(k)). \tag{3.12}$$

This newly defined signal contains all the known information at time instant k except $\hat{SOC}(k)$. $\hat{R}_{ohm}(k)$ comes from the RLS, but $\hat{SOC}(k)$ is unknown at instant k. For such a reason, $\hat{SOC}(k-1)$ is used instead. Rewrite the update of $u_c(k)$ of (3.4) in transfer function form use α and β,

$$\begin{aligned} u_c(k) &= \alpha u_c(k-1) + \beta i_{cell}(k-1), \\ u_c(k) &= \alpha q^{-1} u_c(k) + \beta q^{-1} i_{cell}(k), \\ u_c(k)(1 - \alpha q^{-1}) &= \beta q^{-1} i_{cell}(k), \\ u_c(k) &= \frac{\beta q^{-1}}{1 - \alpha q^{-1}} i_{cell}(k). \end{aligned} \tag{3.13}$$

Combine (3.11), (3.12) and (3.13), a new equation of $y(k)$ and $i_{cell}(k)$ is to be found.

$$\begin{aligned} y(k) &= \frac{\beta q^{-1}}{1 - \alpha q^{-1}} i_{cell}(k), \\ y(k) &= \alpha y(k-1) + \beta i_{cell}(k-1) = \theta_\tau^T \varphi_\tau(k-1), \end{aligned} \tag{3.14}$$

where

$$\theta_\tau = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$
$$\varphi_\tau(k-1) = \begin{bmatrix} y(k-1) \\ i_{cell}(k-1) \end{bmatrix} \tag{3.15}$$

The parameters vector contains the parameters that need to be estimated. The regressor vector contains information until time $k-1$.

3.3.2 Error model

The data that is utilized has several error sources that have great impacts on τ estimation, which are

1. Input output measurement error,
2. The inaccuracy of $\hat{R}_{ohm}(k)$,
3. The error caused by that $\hat{SOC}(k-1)$ is used to calculate OCV.

Expanding the plant model with an error model could improve the estimation accuracy of the plant model parameters, which is a topic in the area of system identification. Here is an explanation with unprofessional words. The data that is utilized to estimate τ is not good data because it contains uncertain terms of $\hat{R}_{ohm}(k)$ and $f(\hat{SOC}(k-1))$. If this data is directly implemented to the estimation algorithm, it must make the plant model parameters estimation be biased. In the other way, an error model adds some new parameters to the equation, which increases the DOF of the whole model. Assume that the error model is chosen properly, then the uncertainties of the data could be transferred to the error model parameters during the estimation, while the rest of the data is used to estimate the plant model parameters. For details of how the error model works, read this reference [15].

Summarizing all the above error sources in one term $e(k)$ and adding it to the model, gives

$$y(k) = \frac{\beta q^{-1}}{1-\alpha q^{-1}} i_{cell}(k) + e(k), \tag{3.16}$$

so the regression form is

$$y(k) = \alpha y(k-1) + \beta i_{cell}(k-1) + e(k) - \alpha e(k-1). \tag{3.17}$$

Note that (3.16) is a simple output-error model which has no parameters about the error term. The accuracy of time constant estimation will have big problem if the error term is not properly modeled, because $e(k)$ includes not only white noise but also colored components which come from $\hat{R}_{ohm}(k)$ and $f(\hat{SOC}(k-1))$. Applying a proper error model will improve the estimation accuracy by assigning more DOF to the error term.

According to the listed three error components, one conclusion is derived that the integrated error may be an almost constant offset with a rapid measurement superimposed. A good error model for such case is a first order system driven by white noise [16], which is

$$e(k) = ce(k-1) + v(k). \tag{3.18}$$

This error cannot be calculated because of the unknown noise $v(k)$. However, it is allowed to be estimated by one step ahead prediction

$$\hat{e}(k) = ce(k-1). \tag{3.19}$$

Consequently, the optimal estimation model that contains the plant model plus one step ahead prediction error model is

$$\begin{aligned} y(k) &= \alpha y(k-1) + \beta i_{cell}(k-1) + ce(k-1) - \alpha e(k-1) \\ &= \alpha y(k-1) + \beta i_{cell}(k-1) + (c-\alpha)e(k-1) \\ &= \alpha y(k-1) + \beta i_{cell}(k-1) + \gamma e(k-1) \\ &= \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} y(k-1) \\ i_{cell}(k-1) \\ e(k-1) \end{bmatrix} \\ &= \theta_{\tau 2}^T \varphi_{\tau 2}(k-1), \end{aligned} \tag{3.20}$$

where $e(k)$ is also the estimation residual, whose approximation is

$$e(k) = y(k) - \theta_{\tau 2}^T(k-1)\varphi_{\tau 2}(k-1). \tag{3.21}$$

AKF algorithm is implemented in the model of (3.20). The error components that originate from using inaccurate $\hat{R}_{ohm}(k)$ and $f(\hat{SOC}(k-1))$ will be transferred to the error model, which makes the time constant estimation more precise.

3.3.3 Signal scaling

Signal $y(k)$ with voltage as the element and signal $i_{cell}(k)$ with current as the element are usually not at the same magnitude level. It is therefore better to scale them before using AKF. Let k_y and k_i be the scaling coefficient, which changes (3.20) to

$$\begin{aligned} k_y \cdot y(k) &= \alpha \cdot k_y \cdot y(k-1) + \beta \frac{k_y}{k_i} \cdot k_i \cdot i_{cell}(k-1) + k_y \gamma \cdot e(k-1) \\ &= \begin{bmatrix} \alpha & \beta \frac{k_y}{k_i} & k_y \gamma \end{bmatrix} \begin{bmatrix} k_y \cdot y(k-1) \\ k_i \cdot i_{cell}(k-1) \\ e(k-1) \end{bmatrix} \\ &= \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix} \begin{bmatrix} k_y \cdot y(k-1) \\ k_i \cdot i_{cell}(k-1) \\ e(k-1) \end{bmatrix}. \end{aligned} \tag{3.22}$$

As $y(k-1)$ and $i_{cell}(k-1)$ are measured, $e(k-1)$ is estimated and the parameters can be identified with AKF. Resistance R and capacity C can be reversely solved as

$$\begin{aligned} R &= \frac{k_i}{k_y} \cdot \frac{\theta_2}{1-\theta_1}, \\ C &= \frac{k_y}{k_i} \cdot \frac{\Delta t(\theta_1 - 1)}{\theta_2 log(\theta_1)}. \end{aligned} \tag{3.23}$$

3.3.4 Closed loop estimation

In (3.12), OCV term is calculated by taking in the estimated SOC of previous step. In actual case, the SOC estimation consequences come from EKF, which makes the time constant estimation and SOC estimation form a closed loop. Figure 3.1 illustrates the logic.

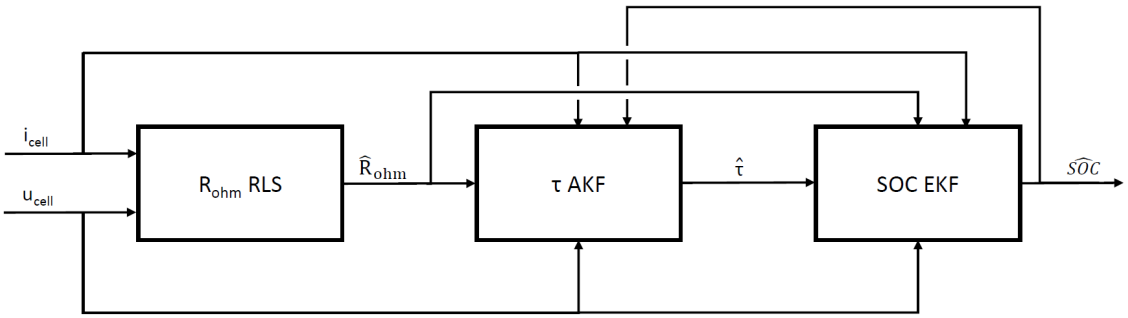


Figure 3.1: The logic of the closed loop estimation. Time constant and SOC estimation forms a closed loop.

The first block executes RLS algorithm with information of i_{cell} and u_{cell} provided, and outputs estimated $\hat{R}_{ohm}$. The second block takes $\hat{R}_{ohm}$ as parameter then takes in information of i_{cell}, u_{cell} and $\hat{SOC}$ that comes from the third block to execute AKF algorithm, and outputs estimated $\hat{\tau}$. The third block takes $\hat{R}_{ohm}$ and $\hat{\tau}$ as parameters, uses information of i_{cell} and u_{cell} to execute EKF algorithm, and outputs estimation $\hat{SOC}$ then feeds it back to the second block. Apparently, the second and third blocks work in a closed loop.

3.4 *SOC* estimation with extended Kalman filter

With all the unknown parameters identified at each sample instant, three of the four matrices that EKF needed are found, which are

$$\begin{aligned} F(k) &= \begin{bmatrix} \hat{\alpha}(k) & 0 \\ 0 & 1 \end{bmatrix}, \\ B(k) &= \begin{bmatrix} \hat{\beta}(k) \\ \Delta t / Q_{nom} \end{bmatrix}, \\ D(k) &= \begin{bmatrix} \hat{R}_{ohm}(k) \end{bmatrix}. \end{aligned} \tag{3.24}$$

The last needed matrix H is the observation matrix, but the observation formula contains nonlinear term $f(SOC(k))$ which must be linearized first in order to get the matrix.

Based on the data that is given in Figure 2.3. The function of OCV-SOC is approximated by curve fitting tool in *Matlab*. After trying several models and orders, a 4th order polynomial model is found to have the smallest mean square error (MSE) to fit the data without taking into account the first data point, see Figure 3.2.

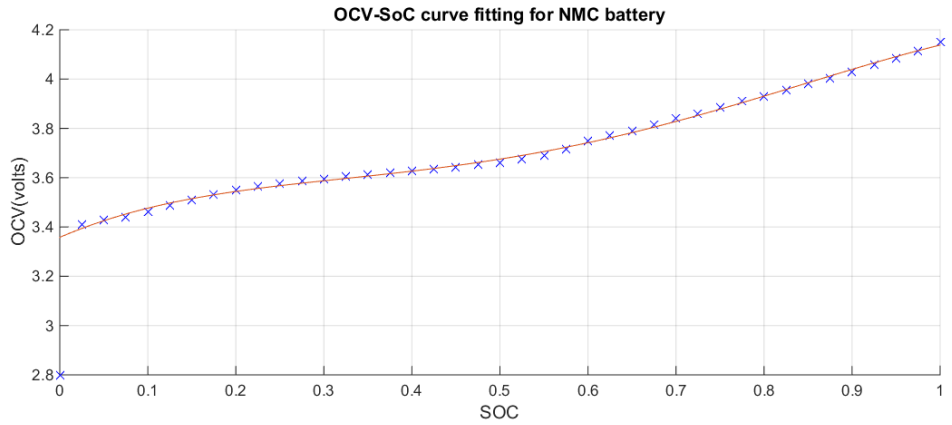


Figure 3.2: A 4th order polynomial function is found to best fit the data of OCV-SOC. The first data point is neglected.

The polynomial is

$$OCV = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix} \begin{bmatrix} SOC^4 \\ SOC^3 \\ SOC^2 \\ SOC \\ 1 \end{bmatrix}, \tag{3.25}$$

where the coefficients are

$$\begin{cases} p_1 = -2.498, \\ p_2 = 5.843, \\ p_3 = -4.102, \\ p_4 = 1.537, \\ p_5 = 3.358. \end{cases} \tag{3.26}$$

Slope-intercept equation linearly describes the OCV value at each step, as

$$OCV = m \cdot SOC + b, \tag{3.27}$$

where m is the instant slope while b is the instant intercept.

At a certain step, m is calculated as the derivative of OCV with regard to SOC, which is

$$\begin{aligned} m &= \frac{d(OCV)}{d(SOC)} \\ &= -9.992\tilde{SOC}^3 + 17.529\tilde{SOC}^2 - 8.204\tilde{SOC} + 1.537. \end{aligned} \tag{3.28}$$

The tilde symbol means that the SOC in (3.28) is a particular value at a certain step rather than a general SOC in (3.27). $SOC(k)$ is unknown at instant k, but $m(k)$ can be approximately calculated thanks again to the fast sampling rate. Use one step former data of SOC, to obtain

$$m(k) = -9.992SOC^3(k-1)+17.529^2SOC(k-1)-8.204SOC(k-1)+1.537. \tag{3.29}$$

The approximation of b is also based on the information at instant $k-1$, which is

$$b(k) = OCV(k-1) - m(k) \cdot SOC(k-1). \tag{3.30}$$

Replacing the OCV term in (3.5) by (3.27) and plug in the calculated $m(k)$ and $b(k)$. The new equation is linearized as

$$u_{cell}(k) = u_c(k) + m(k) \cdot SOC(k) + b(k) + R_{ohm}i_{cell}(k). \tag{3.31}$$

For the reason that $b(k)$ is not allowed to reside in the multiplication matrices, it is moved to the left side to be subtracted by $u_{cell}(k)$ such that a new observation variable $z(k)$ is generated. Therefore, the new observation is

$$z(k) = u_{cell}(k) - b(k) = \begin{bmatrix} 1 & m(k) \end{bmatrix} \begin{bmatrix} u_c(k) \\ SOC(k) \end{bmatrix} + [R_{ohm}]i_{cell}(k), \tag{3.32}$$

where the observation matrix is

$$H(k) = \begin{bmatrix} 1 & m(k) \end{bmatrix}. \tag{3.33}$$

Until now the last matrix is found. The original Kalman filter algorithm that is implemented on this model can estimate SOC, and its mapping on OCV feeds back to be used in the time constant estimation.

The acquisition of $m(k)$ and $b(k)$ requires current step's information about OCV and SOC, which is impossible. Again, thanks to the slow change of SOC and fast sampling rate, estimation of SOC at time $k-1$ is applied to approximate $m(k)$ and $b(k)$.

4

Verification in *Simulink*

4.1 Data set

The data set that was utilized to verify the algorithm was *NEDC* cycle data collected from *CarMaker*. Figure 4.1 shows the cell current signal. Positive value means that the vehicle consumes battery energy.

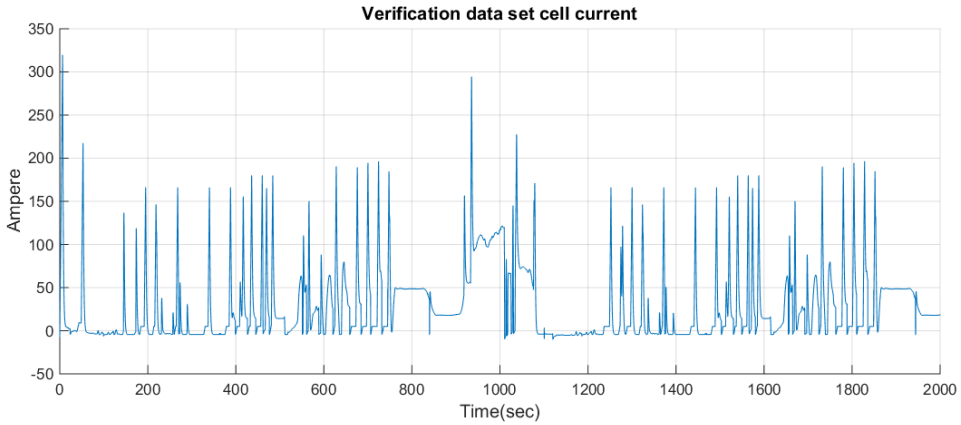


Figure 4.1: Cell current signal of the validation data set.

4.2 Verification cases

Because of the complexity of analyzing close loop estimation and in order to hold the verification in a clear and progressive way, the verification was tested in four cases:

1. Impedance resistance (R_{ohm}) verification,
2. Open loop τ verification,
3. Open loop SOC verification,
4. Comprehensive verification.

4.3 Verification logic of R_{ohm}

The estimation of R_{ohm} is not related to any other parts. The verification has a logic of Figure 4.2.

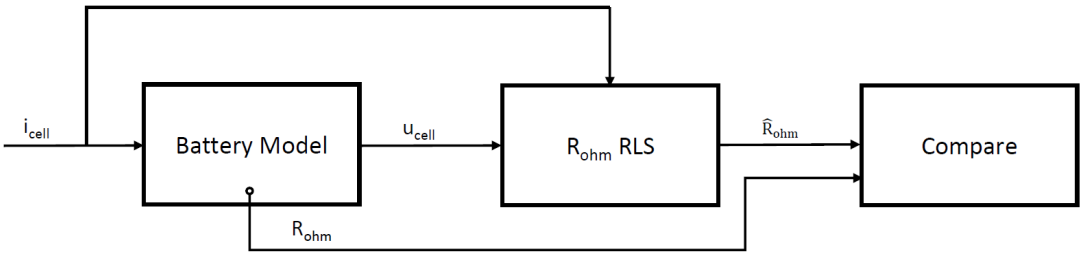


Figure 4.2: The verification logic of R_{ohm}. In the estimation of this parameter, it is neither about open loop nor closed loop.

The battery model takes i_{cell} as input signal to generate required information which is u_{cell} for RLS algorithm, comparing the reference R_{ohm} with the estimated one.

4.4 Open loop verification logic of τ

The estimation of time constant is related to SOC estimation, which makes it difficult to verify. Thus the verification implements the logic of Figure 4.3 to avoid the close loop.

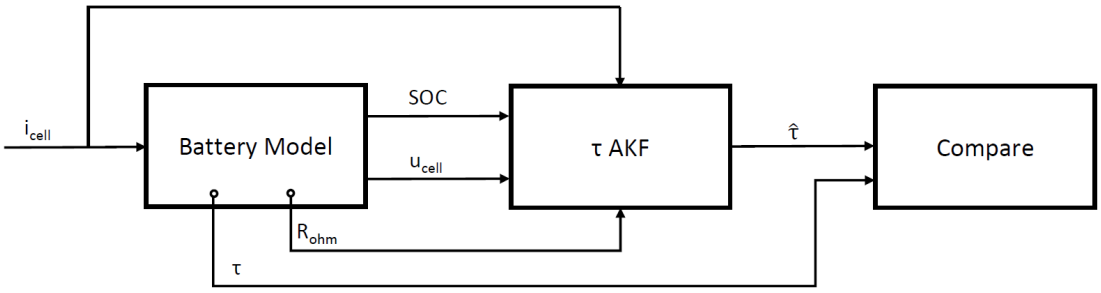


Figure 4.3: The verification logic of time constant. In this estimation case, an open loop is designed to avoid the influence of SOC on $\hat{\tau}$.

The battery model generates all needed information for AKF algorithm which are SOC and u_{cell}, as well as the parameter R_{ohm}, comparing the reference time constant with the estimated one.

4.5 Open loop verification logic of *SOC*

Estimation of SOC requires all parameters to be identified, where the estimation of τ is in closed loop. To avoid the closed loop, the logic in Figure 4.4 is applied.

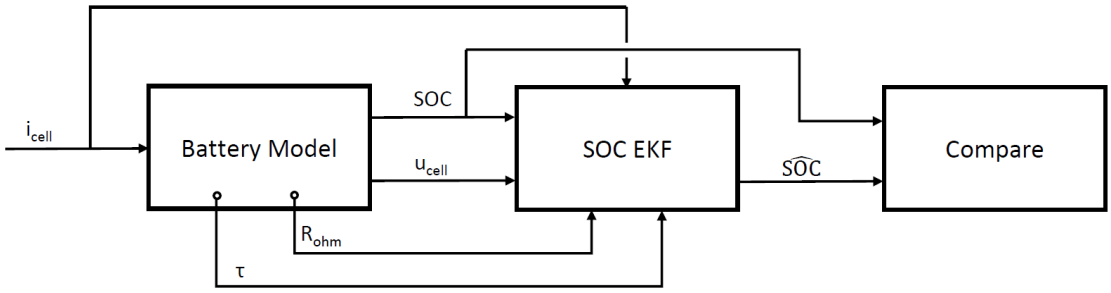


Figure 4.4: The verification logic of SOC. In this case, an open loop is designed to avoid the influence of τ on $\hat{SOC}$.

This verification case inputs the i_{cell} signal to battery model and it generates all needed information. Those information include SOC and u_{cell} as well as parameters R_{ohm} and τ, comparing the reference SOC with the estimated one.

4.6 Comprehensive verification logic

In this verification case, all the estimation algorithms are combined together, where the close loop estimation is included. Figure 4.5 explains the logic.

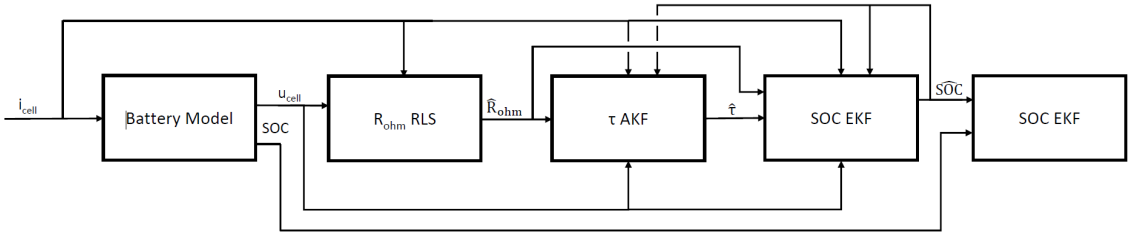


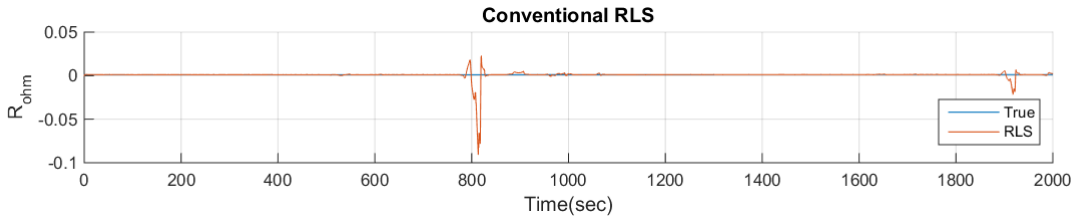
Figure 4.5: The verification logic of combined algorithm. No avoidance is made in this case.

The difference to the other cases is that here the battery model only provides the signal of cell voltage while all parameters are estimated by certain blocks. The reference SOC is compared with the estimated one. This case is consistent with the real operating condition, where the sensors measure i_{cell} and u_{cell}.

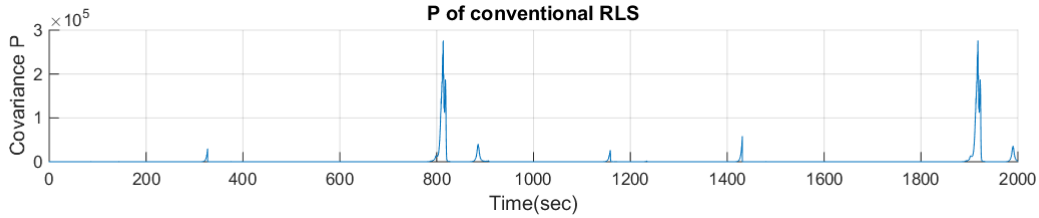
4.7 *Simulink* verification results

4.7.1 Verification result of R_{ohm}

Figure 4.6 shows the verification result of R_{ohm} in conventional RLS algorithm. Two large spikes occur at about 800 second and 1900 second. The reason is the bad excitation of the input signal, which refers to Figure 4.1. From the plot of covariance P, it can be seen that the covariance windup problem of the algorithm occurs at about 800 seconds and 1900 seconds, where the value goes to extremely large.



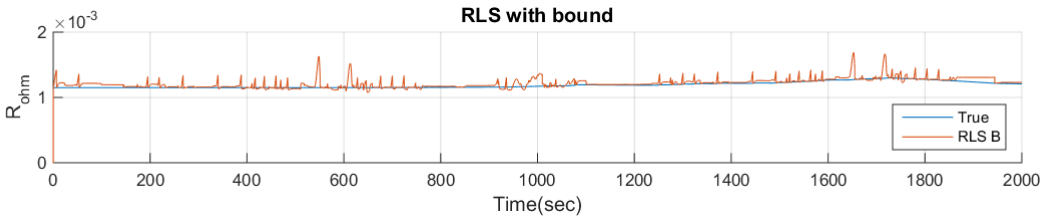
(a) The estimation of R_{ohm}. It has two large spikes.



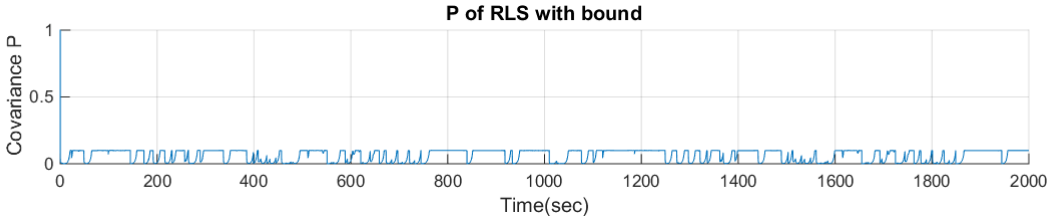
(b) The covariance P goes to extreme larges, which causes the sudden jump of estimation of R_{ohm}.

Figure 4.6: The performance of conventional *RLS*.

Figure 4.7 shows the verification result of *RLS* with a bound. It can be seen that the covariance value is limited with a bound about 0.1, while all the values above it are cut off. In this case, the estimation avoids large spikes.



(a) The estimation of R_{ohm}. With bound value be set, it can be seen that the large spikes disappear.



(b) The covariance P is limited. The values larger than 0.1 are cutted.

Figure 4.7: The performance of *RLS* with bound.

Figure 4.8 is the verification result of *RLS* with a bound plus dead zone. The covariance outlook is the same as *RLS* with a bound. The dead zone filters the small variations in input signal which could be caused by noises in realistic life. It can be seen that the estimation becomes much smoother. A drawback of dead zone is that it makes the convergence to true value slower. No adaptation is made at the first 150 seconds, because there is not enough information in input signal.

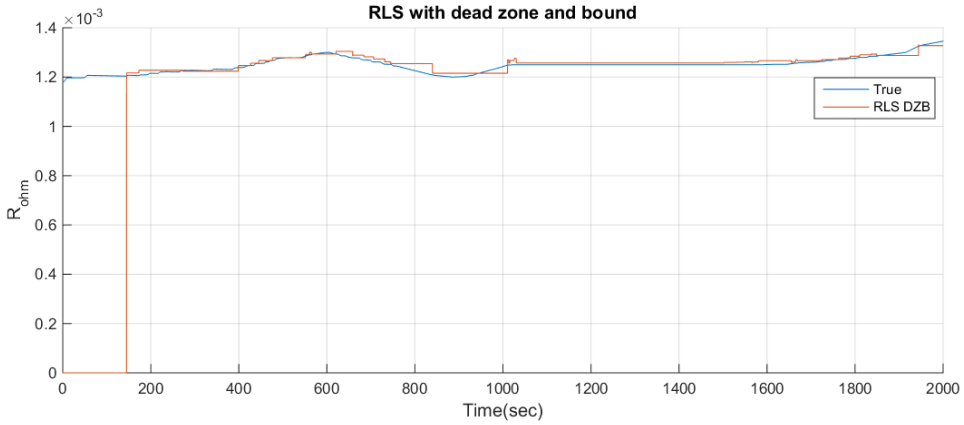


Figure 4.8: The estimation result of R_{ohm} in the case of *RLS* with deadzone and bound. No adaptation is made at the first 150 seconds.

Calculating the mean square error of the three types of *RLS* tests, has shown result of

$$MSE = \begin{cases} 5.312\text{e}{-5}, \text{conventional } RLS, \\ 3.1249\text{e}{-9}, RLS \text{ with bound}, \\ 9.3262\text{e}{-9}, RLS \text{ with bound and dead zone}. \end{cases} \tag{4.1}$$

Although the *MSE* of *RLS* with bound is smaller than *RLS* with deadzone and bound, in reality it is more important to keep $\hat{R}_{ohm}$ less fluctuated.

4.7.2 Open loop verification result of τ

The open loop *AKF* time constant verification is shown in Figure 4.9. The estimation captures the true value well.

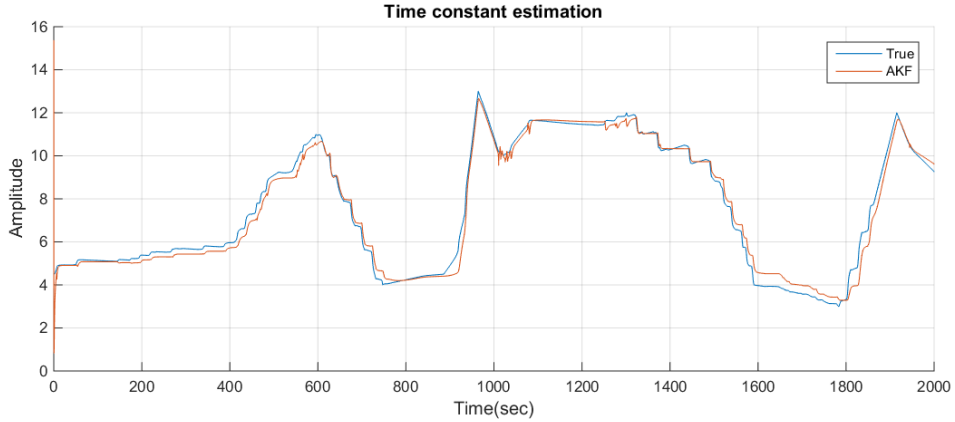


Figure 4.9: The *AKF* estimation of time constant in open loop case.

4.7.3 Open loop verification result of *SOC*

The open loop estimation of *SOC* is shown in Figure 4.10. Because all the parameters needed for *EKF* use exactly the same value of the battery model, the only error term comes from the linearization. Within expectation, the estimation residual should be very small. In fact the mean square error in this test is 1.2029e-18.

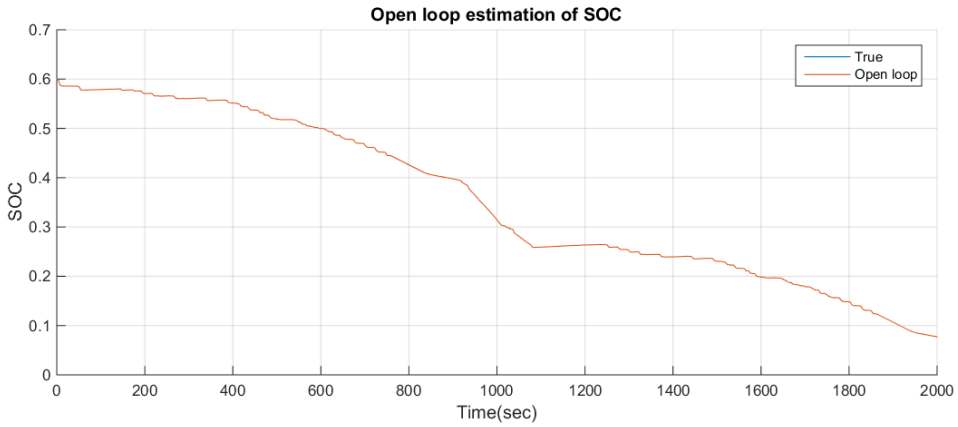
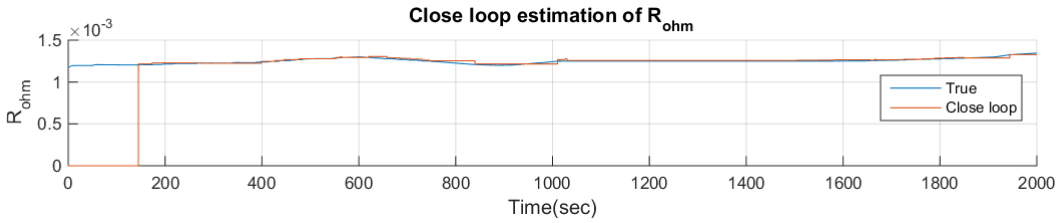


Figure 4.10: The *SOC* estimation result in open loop case.

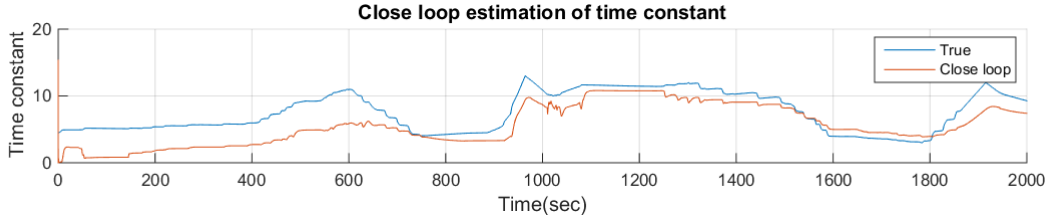
4.7.4 Comprehensive verification result

This verification case is the combination of all subparts estimations, including the closed loop.

The two plots in Figure 4.11 are closed loop parameters estimation results. Due to the complexity of close loop estimation and cancellation, the estimated time constant has an obvious bias to the reference value. However, since the time constant appears in the exponential coefficient, a small variation of the time constant towards the true value will not have severe impacts on $S\hat{O}C$.



(a) The estimation result of R_{ohm} in closed loop case



(b) The estimation result of time constant in closed loop case

Figure 4.11: Closed loop parameters estimation performance.

Figure 4.12 shows the closed loop *SOC* estimation result. From 0 sec to 700 sec and after 1800 sec the estimation are less accurate than the rest part, which is consistent with the phenomena in 4.11(b).

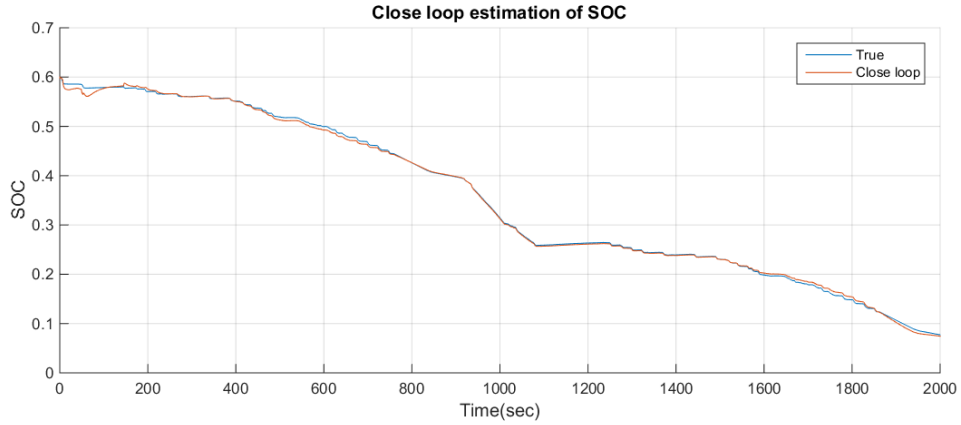


Figure 4.12: The estimation result of *SOC* in closed loop case

The absolute value of *SOC* estimation residual is shown in Figure 4.13. The largest residual occurs at the beginning, which is because of the initial guess and the slow adaptation of impedance resistance. The largest residual in percentage is about 1.6%. Most of time the residual in percentage is between 0 and 0.5%.

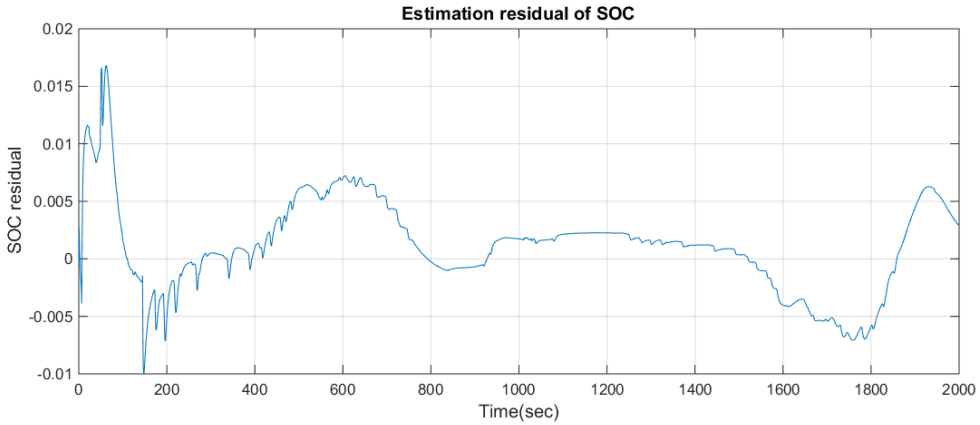


Figure 4.13: Estimation residual of *SOC*

5

Verification in *CAN* network

5.1 Softwares

CarMaker and *CANoe* were used in the project. *CarMaker* was the software utilized to simulate complete car model with certain scenarios. *CANoe* was applied to simulate *CAN* environment and conduct hardware in loop test.

In order to simulate the *ECU*s in *CAN* environment, transmission control unit (*TCU*) and battery control unit (*BCU*) were removed from *CarMaker* but are placed in *CANoe*. Note that *SOC* estimation is one functionality of *BCU*.

5.2 Network in the project

The functions that were only tested in *CarMaker* cannot ensure the well functioning in reality, because *CarMaker* did not simulate the data transmission in the network where the communication delay and signal priorities must be considered. An interesting and beneficial thing was to build a basic *CAN* network, such that this network became a skeleton for future works. For example, more nodes can be added to this network and more complicated data statistics may be collected.

The network was built by connecting two PCs, which was the minimum configuration. The layout is shown in Figure 5.1. PC1 ran *CANoe* while the models of *TCU* and *BCU* are implemented in it. PC2 ran both *CarMaker* and *CANoe* while the remaining models are implemented in *CarMaker*. The two PCs accessed the network through Vector VN1630a which is a portal device. The data exchange between *CarMaker* and *CANoe* was achieved through a special configuration of functional mock-up interface (*FMI*). The logistic orders during the data transmission are as follows. PC2 transmits *CarMaker* signals to the network. PC1 receives signals from the network and executes the calculations in the models of *TCU BCU*, then transmits the outputs back to network. PC2 receives the outputs then uses those in *CarMaker* simulation. The timing master of this network is *CANoe*.

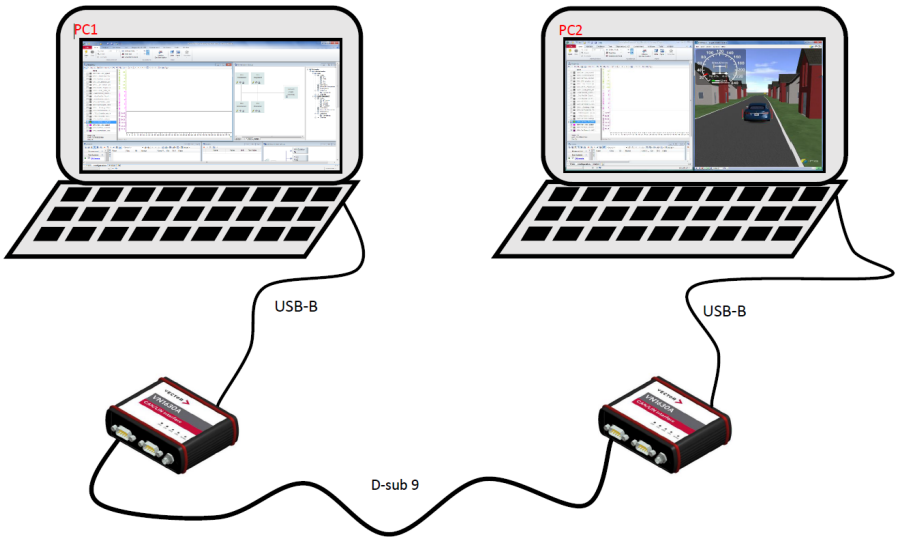


Figure 5.1: 2 PCs *CAN* configuration

5.3 *SOC* verification in *CAN*

The *SOC* estimation algorithm was integrated into *BCU* model. Since PC1 and PC2 simulated two nodes in the network, the network was used to verify the performance of the algorithm in *CAN* environment.

Figure 5.2 shows the *CANoe* configuration. The plot window inside is the outcome of *SOC* estimation.

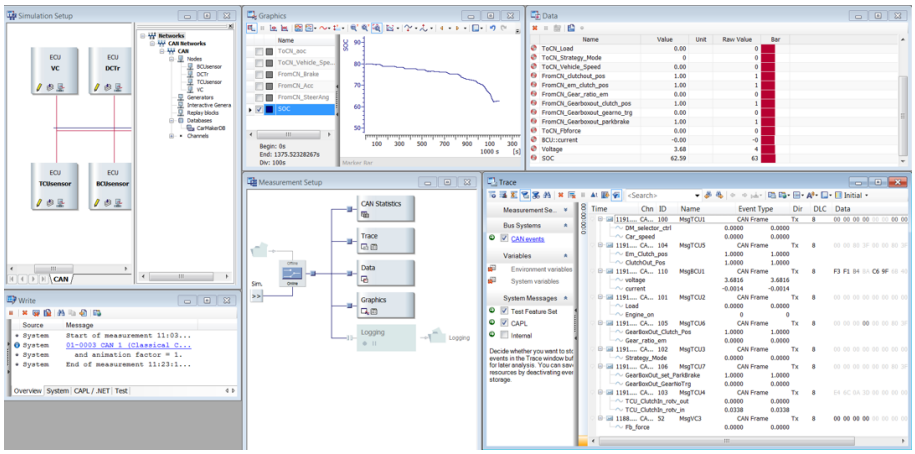


Figure 5.2: *SOC* estimation verification in CANoe

The estimation data was exported from *CANoe* while the reference data was exported from *CarMaker*. Figure 5.3 shows the comparison between them.

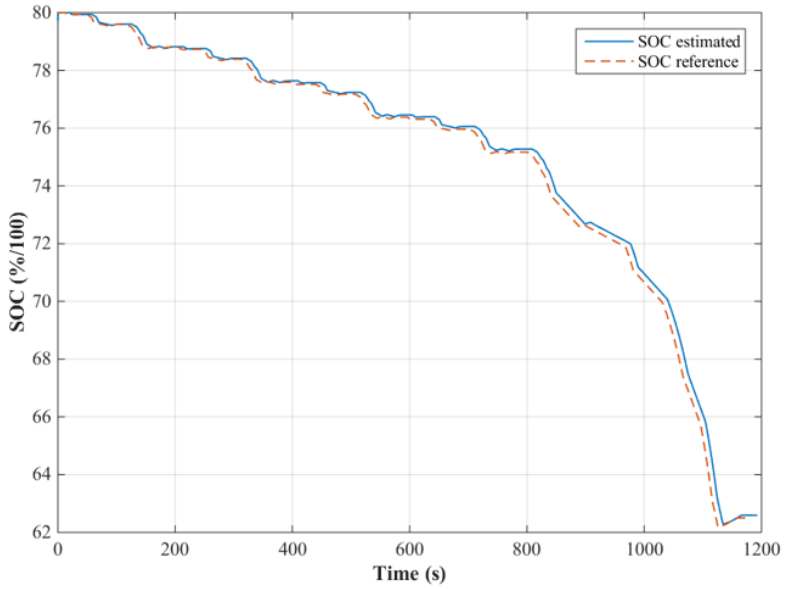


Figure 5.3: The comparison between *SOC* reference and *SOC* estimation

A gap between the two curves is inspected, which is because the measurement of *CANoe* did not start at the same time as *CarMaker* simulation. Beside that, the estimation has good accuracy.

6
Conclusion and future work

The verification tests presented in Chapter 4 prove that the algorithm has effect of adaptation on parameters R_{ohm} and τ. The drawback is that the implementation of the algorithm neglects some useful data during the estimation of both impedance resistance and time constant, which causes estimation bias. The solution is to assign the summarized error term with a proper model such that more *DOF* are given to the error model parameters, which increases the plant model parameters estimation accuracy. Another problematic part is the close loop, where the mutual dependencies between *AKF* and *EKF* are difficult to analyse. Eventually the concluded largest residual is 1.6%. The initial guess of *SOC* is a principal issue in *EKF*. Fortunately this value can be derived by measuring the battery terminal voltage and then map it back to *SOC* at the time when battery is at equilibrium state.

The equivalent model used in the project is the simplest case where only one *RC*-circuit is included. The estimation model that is based on this simple circuit cannot be used to estimate a real battery that has many *DOF*. To improve the algorithm, the model should be expanded to contain more *RC*-circuits so that the *DOF* is increased to match the estimation requirements. The problem that arises when using more *RC*-circuits is that more parameters need to be identified, which will cause the designer to spend more efforts on dealing with the close loop estimation.

The battery ageing effect only takes into account the impedance resistance and time constant. In practice, the ageing has correlations to many factors such as operating temperature, charging rate, *SOC* range. For a further research, the impacts of those factors should be included in the model to test the robustness of the algorithm when working in multiple parameters varying cases.

From system identification point of view, the validation data set used to estimate parameters must have coverage over broad spectrum to avoid accidentally correct estimation of parameters. For *NEDC*, the signal spectrum is not very rich in spectrum because the driver's behavior is constrained by the environment. A maturer validation process should try to implement larger and more detailed data sets.

Bibliography

[1] Pillot, Christophe, *Micro hybrid, HEV, P-HEV and EV market 2012-2025 impact on the battery business*. Barcelona, Spain , Electric Vehicle Symposium and Exhibition (EVS27), 2013 World. IEEE.

[2] Robert Rapier. *U.S. Electric Vehicle Sales Soared In 2016* [online]. `https://www.forbes.com/sites/rrapier/2017/02/05/u-s-electric-vehicle-sales-soared-in-2016/#6423bbae217f`

[3] Wen-Yeau Chang. *The State of Charge Estimating Methods for Battery: A Review*. ISRN Applied Mathematics Volume 2013 (2013), Article ID 953792. Department of Electrical Engineering, St. John's University, 499, Sec. 4, Tam King Road, Tamsui District, New Taipei City 25135, Taiwan.

[4] N. Watrin, B. Blunier, and A. Miraoui, *Review of adaptive systems for lithium batteries state-of-charge and state-of-health estimation*, in Proceedings of IEEE Transportation Electrification Conference and Expo, pp. 1–6, Dearborn, Mich, USA, June 2012.

[5] Steve Knoth. *Know a Battery's State-of-Charge By Counting Coulombs*. Linear Technology Corporation.

[6] *State of Charge(SOC) Determination*. Electropedia. `http://www.mpoweruk.com/soc.htm`

[7] Jens Groot. *State-of-health estimation of Li-ion batteries : ageing models*, pp. 10-11, ISBN 9789175971346, Chalmers University of Technology, 2014.

[8] Simone Barcellona, Morris Brenna, Federica Foiadelli, Michela Longo, Luigi Piegari. *Analysis of Ageing Effect on Li-Polymer Batteries*. ScientificWorldJournal. 2015; 2015: 979321. Published online 2015 Jul 5. doi: 10.1155/2015/979321.

[9] Ahmad Rahmoun, Helmuth Biechl. *Modelling of Li-ion batteries using equivalent circuit diagrams*. Przeglad Elektrotechniczny (Electrical Review), ISSN 0033-2097, R. 88 NR 7b/2012.

[10] N. Watrin, B. Blunier, and A. Miraoui. *Review of adaptive systems for lithium batteries state-of-charge and state-of-health estimation*, in Proceedings of IEEE

Transportation Electrification Conference and Expo, pp. 1–6, Dearborn, Mich, USA, June 2012.

[11] *How to Measure State-of-charge*. Battery University. BU-903. `http://batteryuniversity.com/learn/article/how_to_measure_state_of_charge`

[12] Björn Fridholm, Torsten Wik, Magnus Nilsson. *Robust adaptive Kalman filter for online impedance estimation in automotive batteries*, 2016. Section 3.1.

[13] Rolf Johansson. *System Modeling and Identification*. Section 8.5.

[14] Björn Fridholm, Torsten Wik, Magnus Nilsson. *Robust adaptive Kalman filter for online impedance estimation in automotive batteries*, 2016. Section 4.5.

[15] Rolf Johansson.*System modeling and identification*, 2013. Section 6.2.

[16] Björn Fridholm, Torsten Wik, Magnus Nilsson. *Robust adaptive Kalman filter for online impedance estimation in automotive batteries*, 2016. Section 4.4.

[17] Ebert, Christof; Jones, Capers (2009-04-01). *Embedded Software: Facts, Figures, and Future*. IEEE Computer Society Press. Retrieved 2010-09-15.

[18] Marco Guardigli, *Hacking Your Car* `http://marco.guardigli.it/2010/10/hacking-your-car.html`

[19] *History of CAN technology*. `https://www.can-cia.org/can-knowledge/can/can-history/`

[20] *What is CAN bus*. `http://canbuskits.com/what.php`

[21] Roland Priemer (1991).*Introductory Signal Processing*. World Scientific. p. 1. ISBN 9971509199.

[22] AT&T, Bell System Data Communications Technical Reference, *Data Set 103F Interface Specification*, May 1964

A

Appendix

A.1 Network

A.1.1 Overview

Modern cars are integration of mechanical components, actuators, and electronic hardwares. Electronic control units (*ECU*) are embedded microprocessors in charge of monitoring and processing the signals in car. Commonly, modern cars have up to 80 *ECU*s [17]. Such complicated embedded systems transmit and receive a lot amount of data, thus it requires an efficient and low lag solution of data transmission.

Several digital communication protocols were created to solve the above problem. The protocols are ranked to three classes [18],

1. Class A: up to 10Kbit/sec, multipurpose, asynchronous, used for non-realtime, smart sensors, wire reduction.
2. Class B: in the range 10Kbit/sec up to 125Kbit/sec, used for intermodule data transfer and non-realtime control.
3. Class C: critical, high speed, realtime communications between devices. *CAN* is one of those up to speed 1Mbit/sec. There are other alternatives, for example Flexray up to 10Mbit/sec.

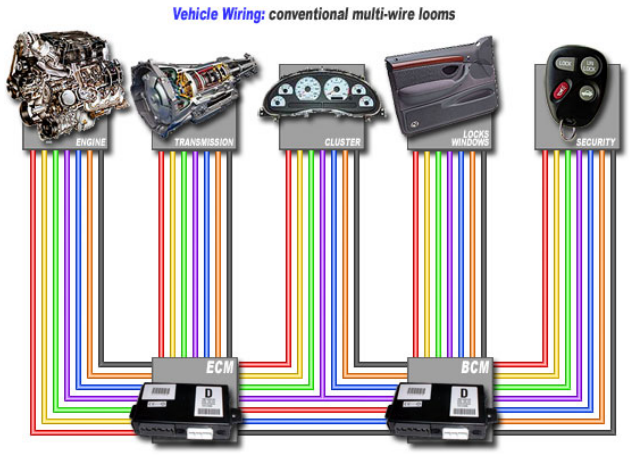
CAN is widely used in many cars for hardwares communication, but not the only type of network. FlexRay plays the role of backbone communication between several master domains, because it has a stronger capability to deal with big amount of data.

A.1.2 Controller area networks

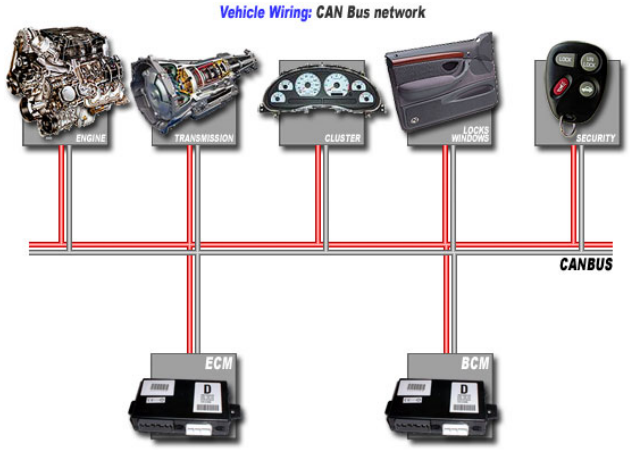
Controller area network which is also known as *CAN*, is a protocol developed by Robert Bosch GmbH at 1983. Since the release in 1986, *CAN* had become more and more widely used and eventually it becomes the standardized communication protocol in car.

The creation of *CAN* was to add new functionalities, but during its design it results in also reducing the wiring, which is a by product beyond the initial intention [19]. For the protocols before *CAN*, the nodes were connected to each other in the way of peer-to-peer, which brought a harassment of wring and error detection. Instead, *CAN* uses a bus as a mainstream, the messages are broadcasting in it such that all nodes connected to *CAN* have access to the information. Each messages contains

not only the data bits but also identifier bits which are used for the nodes to filter out irrelevant messages. Due to the special structure, *CAN* has a better handling of errors comparing to the conventional protocols. In addition, the structure ensures that disconnection of one node in *CAN* will not interfere the rest of the network.



(a) Conventional wiring



(b) *CAN* wiring

Figure A.1: Conventional wire and *CAN* bus [20]

Figure A.1a and A.1b compare the wiring of conventional bus and *CAN*. The complexity of the conventional network wiring makes it uneasy to conduct troubleshooting and it is also inflexible to implement system expansion and tests. For *CAN*, a central cable connects *ECU*s all together. New devices can be conveniently added to the network without interrupting the whole wiring. With all the available data flowing on the bus, devices can pick their own needed signals based on the identifier bits of specific message. *CAN* supports maximum bit rate of 1Mbit/sec and guarantees less than 120 microseconds latency within 40 meters.

There are several advantages and disadvantages listed below,

- Pros
  1. Reduces wiring complexity
  2. Low cost, light weight
  3. Standardized
  4. Data consistency
  5. High configuration flexibility
  6. Guaranteed bit rate
  7. Real time, small latency
- Cons
  1. High software expenditure
  2. Limit on central cable length
  3. Efficiency decreases as number of devices increases
  4. Low security

Despite the advantages, disadvantages still exist. The signals on *CAN* bus are event-driven, which means it requires well organized logics and priorities to transmit signals. Due to the physical properties of the cable, data on the bus has an upper limit which constraints the number of devices available in *CAN*. Further, data transmits freely on the bus and available for any devices, makes the information unsecured.

A.1.3 *CAN* Signal and message

A signal as referred to in communication systems, signal processing, and electrical engineering is a function that conveys information about the behavior or attributes of some phenomenon [21]. However, signal has an additional meaning in *CANoe*. In *CANoe*, signal is defined as a mapping of external data where it has types of unsigned, signed, float and double. The external data may come from other softwares. The signal must be defined with correct type as the external data in order to avoid overflow.

Message is the frame of information that the nodes send to netowrk. Message has two formats where one is *CAN* standard the other is *CAN* extended. Message is a sequence of binary numbers which are divided to several fields. Some of those fields are configuration fields that define the properties of the message and others are data fields that contain the useful data. The most often used fields are identifier field, data length field and data field.

A.1.3.1 Identifier field

The difference of *CAN* standard and *CAN* extended is the length of identifier field in the message. *CAN* standard has 11 bits length ID while *CAN* extended has 29 bits length ID.

The *CAN* specification specifies that the dominant bit is logical 0 and the recessive bit is logical 1. If two bits are sent to *CAN*, the dominant bit defeats the recessive bit such that the message with recessive bit shall be delayed until the message with

dominant bit finish its transmission.

ID number(hex)	ID field bits											Other fields
0x7	0	0	0	0	0	0	0	0	1	1	1	
0x8	0	0	0	0	0	0	0	1	0	0	0	

Table A.1: ID field

Table A.1 lists the identifier bits of two messages that have *ID* number 0x7 and 0x8 respectively. There will be a collision occurs at 8th bit When two nodes transmit those messages simultaneously. Node that transmit 0x7 will continue since logical 0 defeats logical 1, while node that transmit message 0x8 will pause until the other node finishes its current frame. Message with lower *ID* number has higher priority, this priority feature of messages makes *CAN* suitable for real time communication.

Identifier field also defines the uniqueness of messages, nodes only receive the messages that have the correct *ID*.

A.1.3.2 Data length code field

DLC field defines bit length of the data field. *DLC* of *CAN* standard can be set to a decimal number between 0 to 8. One unit value means one bytes data such that the data field length could vary from 0 bytes to 8 bytes, in another word 0 bits to 64 bits.

The *DLC* should be chosen properly to optimize the efficiency and utilization of the network, and the *ID* of the messages should be assigned according to the specification of the system.

A.1.3.3 Data field

Length of data field varies from 0 bits to 64 bits in relate to *DLC*. This field is the main field in messages where the information of data is saved. Data field has maximum 64 bits, which means that one message can carry one double typed data at most.

A.1.3.4 Baud rate

Digital data modem manufacturers commonly define the Baud rate as the modulation rate of data transmission and express it as bits per second [22]. The calculation of data transmission load uses formula:

$$B = \sum_{i=1}^{N} f_i \times l_i, \tag{A.1}$$

where N is the number of messages, f is the frequency of message, l is the length of the message frame. If two messages are sending with frequencies 15 and 20 frames

per second and with length of frame 50 bits and 70 bits, the total data transmission load is

$$B = (15 \times 50) + (20 \times 70) = 2150\text{bits/sec.} \tag{A.2}$$

Baud rate of the *CAN* network should be larger than this value to ensure the smooth of data transmission.

A.2 Functional Mock-up Interface

Functional Mock-up Interface was first released at 2010, it is a simulation model standard designed to improve compatibility for different suppliers and manufacturers. The model of *FMI* standard is called Functional Mock-up Unit. Model developed by Matlab/Simulink that is *FMI* supported can be compiled to *FMU* which is allowed to be read by other *FMI* supported softwares, e.g., *CANoe*. With *FMI*, the models designed by suppliers with different softwares can be integrated in a convenient and consistent way, which guarantees engineers more time to focus on the project itself rather than waste time on the compatibility and data exchange problems between different formats.

CarMaker and *CANoe* are both *FMI* supported softwares, the *TCU* and *BCU* models as well as other models in *CarMaker* are of *FMI* format.