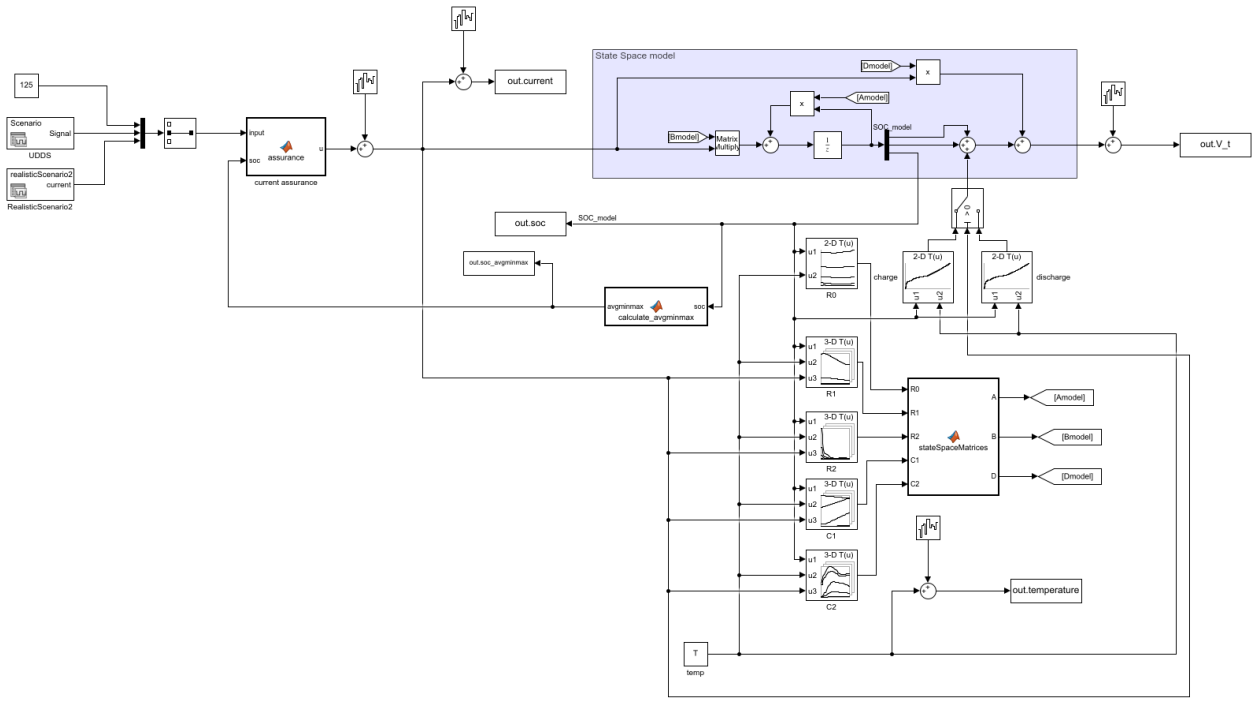
CHALMERS
UNIVERSITY OF TECHNOLOGY



Computationally Efficient State Of Charge Estimation Algorithms for Multi-Cell Systems

Evaluating the relation between computational complexity and performance for lithium-ion battery cells connected in series

Master's thesis in Systems, Control and Mechatronics

SOLEIL KYLANDER
STEFAN MOMENTE

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Computationally Efficient State Of Charge Estimation Algorithms for Multi-Cell Systems

Evaluating the relation between computational complexity and performance for lithium-ion battery cells connected in series

SOLEIL KYLANDER
STEFAN MOMENTE

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2023

Computationally Efficient State Of Charge Estimation Algorithms for Multi-Cell Systems
Evaluating the relation between computational complexity and performance for lithium-ion battery cells connected in series
SOLEIL KYLANDER
STEFAN MOMENTE

Supervisor: Yang Xu, Volvo Technology AB
Examiner: Torsten Wik, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: `Simulink` file of scalable number of cells connected in series.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Computationally Efficient State Of Charge Estimation Algorithms for Multi-Cell Systems
Evaluating the relation between computational complexity and performance for lithium-ion battery cells connected in series
SOLEIL KYLANDER
STEFAN MOMENTE
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The percentage of available electric charge in an electric vehicle's battery is referred to as the state of charge (SOC). The accuracy of the estimated minimum, maximum, and average SOC is of great importance to safely operate the vehicle. However, the SOC estimations are computationally expensive and therefore this thesis aims to reduce this cost while trying to maintain accuracy. Six different algorithms are proposed using sigma point- and extended Kalman filters. The algorithms' accuracy and complexity are tested in different operating conditions and evaluated against an algorithm estimating each cell individually. The results show that the optimal algorithm is almost as accurate and one thousand times faster in comparison to the algorithm estimating each cell individually. Additionally, the experiments show that the extended Kalman filter is less computationally demanding than the sigma point Kalman filter, with similar performance. The algorithms have a high dependency on the capacity and thus a joint SOC and capacity estimation is also tested. Estimations simulated without disturbances shows to be a promising method for future development. The findings in this thesis suggest how computational complexity can be reduced without the cost of losing a significant amount of accuracy.

Keywords: battery management system (BMS), extended Kalman filter (EKF), sigma point Kalman filter (SPKF), state of charge (SOC), state of capacity (SOQ), computational efficiency, electric vehicles (EVs), Li-ion batteries.

Acknowledgements

This Master's thesis have been conducted at Volvo GTT in Gothenburg, Sweden in conjunction with Chalmers University of Technology, Department of Electrical Engineering in Gothenburg, Sweden during the spring of 2023.

Our work would not have been possible without the tremendous help and assistance from the co-workers at the BMS-team, at Volvo GTT. During our time with the Volvo BMS-team we have received inputs, tips, guidance and maybe most importantly a very warm welcome. We would like to thank Olle Friberg, the manager of the BMS-team, for a terrific work environment and the equipment needed to complete our thesis work.

We would like to give a special thanks to our supervisor at Volvo GTT, Yang Xu, who has taken the extra time to help us even when traveling abroad. His patience, valuable insight, and ideas have been of great value and pointed us in the right direction throughout the master thesis work. A great thanks to Professor Torsten Wik, the examiner for this master thesis, for making this project possible in the first place.

Soleil Kylander & Stefan Momente, Gothenburg, October 2023

Contents

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BMS	Battery management system
CC	Coulomb counting
CDF	Cumulative distribution function
EC	Equivalent circuit
EKF	Extended Kalman filter
EVs	Electric vehicles
KF	Kalman filter
OCV	Open circuit voltage
PDF	Probability density function
RC	Resistor-capacitor
RLS	Recursive least squares
SOC	State of charge
SOH	State of health
SOQ	State of capacity
SPKF	Sigma point Kalman filter
UDDS	Urban dynamometer driving schedule

Nomenclature

Below is the nomenclature of indices, parameters, and variables that have been used throughout this thesis.

Indices

k	Denotes the discrete parameter or variable at time instance k
l	Denotes an arbitrary integer between 1 and n

Parameters

n	Number of battery cells connected in series
N	Number of states
p	Filter order
Q	Cell capacity
T	Cell temperature
W	Weight for sigma points
δ	Initial value for filter weights
Δt	Sampling time
λ	Forgetting factor

Variables

A, B, C, D	State space matrices
c	An estimation of the integrated current used when estimating SOC
$C_{1,2}$	RC circuit capacity
d	An estimation of the integrated current used when estimating SOQ
f	Nonlinear motion model

F	Linearized motion model
h	Nonlinear measurement model
H	Linearized measurement model
i	Cell current
K	Kalman gain
P	State covariance matrix
Q	Process noise covariance matrix
R	Measurement noise covariance matrix
R_0	Ohmic resistance
$R_{1,2}$	RC circuit resistance
S	Innovation covariance
t	Time instance
$V_{0,1,2}$	Voltage
V_{OC}	Open circuit voltage
V_t	Terminal voltage
u	Control input
v, w	White noise
x	State vector
y	Measured output
z	State of charge
Δz	Difference in state of charge between two iterations
χ	Sigma point

List of Figures

List of Tables

1
Introduction

Electric vehicles (EVs) have gained much popularity in recent years, and the market is growing fast [1]. EVs have contributed to creating a more sustainable future, taking a step towards the reduction of greenhouse gases, a subject that is gaining more attention and is reflected in many national and international regulations. The European Commission has set up a goal of achieving a 55% greenhouse gas emission reduction by 2030, a mission which is shared by many car and heavy-duty vehicle companies [2]. Volvo Cars have declared that they are planning to become a fully electric car company by 2030 [3]. Ford, Volkswagen, Volvo Group, Renault, and Scania, to name a few, have similar missions to reach environmental, health, and societal objectives [1].

According to the International Energy Agency, the sales of EVs have doubled in 2021 from the previous year, and are expected to continue growing [4]. The rapid rise of EV sales has increased the demand and pressure on the battery market. Together with the war in Ukraine additional pressure and challenges have appeared [5]. Since lithium-ion batteries are one of the most widely used batteries in the field of EVs [6], the price of lithium, but also other metals used in the production of EV batteries, have increased [5]. It is estimated that the supply of such materials would have to be increased by one-third to meet the demands of EVs by 2030 [5]. However, since the materials are finite, recycling and maximizing the value of the materials will be critical after 2030 [4]. Since a battery is typically considered expired when its state of health (SOH) is below 80%, it is of great importance that the vehicle producers, customers, and the society as a whole strive for the design and usage of batteries to be optimized with respect to performance and battery lifetime [7].

Since the battery lifetime depends on the accuracy of the estimations it is of great importance to reduce the estimation errors as much as possible. Having high accuracy and a low margin of error can be achieved by good estimations algorithms and battery models, which result in a larger operating range of the battery. This would increase the value of the battery, preserve energy and optimize the usage of finite resources [8].

1.1 Background

In EVs it is the battery management system (BMS) that monitors and controls the battery system to optimize the batteries' performance and lifetime [8], [9]. The BMS is crucial in any rechargeable electrical product but it is especially important in EV

applications since it is working in real-time harsh charge and discharge conditions and environments [10]. The BMS can be classified into two main functions, monitoring and control [11]. The monitoring function consists of physical measurements, estimation of different states, parameters, and fault detection. The control function ensures that the charge and discharge of the battery are kept within safe margins. It balances the state of charge (SOC) between different cells and it governs thermal management [11].

SOC is one of the key states used by the BMS. SOC represents the available charge stored in the battery. It is used to optimize energy management but also to avoid deep discharges or overcharges and other hazardous situations [9]. SOC cannot be measured directly, instead it needs an estimation algorithm based on voltage, current, and temperature measurements [7].

A multi-cell system is a set of cells connected in series, parallel, or a combination of both. Calculating the SOC of a multi-cell system can be done by calculating the SOC for each cell individually. However, this process has a high computational complexity [9]. To reduce the computational demand, cells or groups of cells can be grouped together and estimated jointly. Reducing the computational demand also means reducing costs. The BMS is limited to its hardware's capacity, thus by reducing computational complexity the now freed memory can be used for other tasks or alternatively the hardware can be replaced with a less expensive one [12].

1.1.1 Related work

The advanced estimation and control algorithms in BMS have become a hot research topic in recent years. There have been several studies in the estimation of SOC especially. In some of the most recent studies extended Kalman filter (EKF) and sigma point Kalman filter (SPKF) are used in the estimation algorithms for SOC [9], [13]. According to Plett [14], SPKF has a higher accuracy with the same complexity while a different study [13] concludes that in most cases the two filters behave similarly in their accuracy, rate of convergence, and robustness.

Most studies estimating SOC only consider one cell [13] [15] [16]. Computing the estimations for each individual cell in a multi-cell system will become computationally complex, and might not get useful in real-time applications. In the study [17] bar-delta filters were used to get an estimate of the average SOC and SOH at battery-pack level. Bar-delta filters have two filters, a bar and a delta filter. The bar filter computes the average cell SOC based on average cell voltage, and the delta filter calculates the individual differences in SOC and the average SOC based on individual cell voltage [9]. In the de Souza Aranha and Giesbrecht's study [9], an EKF one-step ahead prediction is used as a bar filter with the average cell voltage as input, and the delta filter is then applied on the batteries state space model. The downside to bar-delta filters is that it only estimates an average SOC value and does not calculate the maximum and minimum SOC value. When only an average SOC value is used in the BMS, larger margins are needed to avoid deep discharges and overcharges, thus not reaching the battery's full potential. Studies regarding computational complexity in SOC estimations are hard to find, and there seems to be a

gap in the research on the correlation between computational complexity, accuracy, and robustness. The recent increase in EVs and their popularity also raises the need for further research regarding this topic.

1.2 Research objectives

The aim of this thesis is to reduce computational demand when estimating the minimum, maximum, and average SOC for a multi-cell system in real-time. The intended outcome is to produce an estimation algorithm that is scalable and adaptive for different numbers of cells and cell imbalances. This estimation algorithm should have a lower computational time and space complexity than an estimation algorithm that estimates each cell individually. However, it is of great importance that the accuracy of the estimations is sufficient since there is a trade-off between computational complexity and accuracy.

1.3 Limitations

Since the main focus of this thesis is to compare the performance between the implemented estimation algorithms and an algorithm estimating each cell individually this is thought to be most easily done using simulations in `Matlab` and `Simulink`. Thus no physical testing is conducted. Hardware data is provided by Volvo GTT. The measurement and input noise are presumed to be Gaussian. State of temperature estimations are outside the scope of this study and are therefore set to a constant value for each of the cells. For the same reason, state of capacity (SOQ) is also assumed to be known for the SOC estimation algorithms. Multi-cell systems where all cells are connected in series are investigated, i.e., systems with cells connected in parallel or in a combination with series-connected are not studied.

2
Theory

In this chapter, we introduce the equations, cell model, state space representation, and filters that are used in the thesis. Furthermore, an introduction to the concept of computational complexity is given.

2.1 SOC

SOC is a measurement of the electric charge available at a certain instance given as a percentage of the maximum possible charge that can be stored in a battery. Since the SOC cannot be measured, an estimation algorithm is needed. SOC depends not only on the cell degradation but also on the current and temperature [18]. For a multi-cell system with cells connected in series, the cell with the lowest SOC limits the battery since current can only flow if all cells are charged. The cell with the highest SOC will limit the battery during charging since overcharging can damage the battery. The average SOC of all cells is also of interest since this value can be used as a general indicator. Thus the minimum, maximum, and average values are of importance.

Many different techniques are used for SOC estimation, such as discharge test, Coulomb counting (CC), measurement of the electrolyte's physical properties, open circuit voltage (OCV), impedance spectroscopy, and internal resistance [19]. CC is based on the supplied or withdrawn current. This method outputs the difference in SOC over a certain time period given the capacity of the cell and the current over the time period, according to

$$z(t) = z(t_0) + \frac{1}{Q}\int_{t_0}^{t} i(t)dt \tag{2.1}$$

where $z(t)$ is the SOC at time t, $i(t)$ is the current and Q is the capacity.

2.2 SOQ

Moseley and Garche defined capacity as 'The battery capacity corresponds to the quantity of the electric charge which can be accumulated during the charge, stored during the open circuit stay, and released during the discharge in a reversible manner' [20, p. 413]. The nominal capacity of a battery is always specified, but the capacity will decrease over time [21], [22]. Since capacity cannot be directly measured, an estimation algorithm is needed. The capacity, or the SOQ, is directly correlated to

the SOC (see Equation 2.1). If the CC equation is rearranged, it can be used to compute the capacity, i.e.,

$$Q = \frac{\int_{t_0}^{t} i(t)dt}{z(t) - z(t_0)} \tag{2.2}$$

2.3 Cell model

There are several different approaches to model a battery. The most common approaches are electrochemical models and electrical equivalent circuit (EC) networks, or combinations of the different approaches [6]. The most commonly used battery modeling approach for real-time applications in the EV field is the electrical EC model [6]. EC models can be divided into two groups: the integral-order models and fractional-order models [23]. The integral-order EC models use a voltage source connected with resistors and capacitors in different ways. The few model parameters provide simplicity in computation while maintaining robustness. Fractional-order models use constant phase elements instead of the capacitors in the integral-order models [23]. This allows the battery characteristics to be simulated over the whole frequency range, but at the cost of slow and complex simulations. For this reason, integral-order models have been used in this project.

Two commonly used integral-order EC models are the first- and second-order resistor-capacitor (RC) circuits [24]. These circuits consist of a resistor connected in series with one or more RC-pairs, which is a resistor connected in parallel with a capacitor. The resistor models the ohmic resistance while the RC-pairs models the polarization of resistance and capacitance. The difference between the first- and second-order circuits is that the latter has one additional RC-pair connected in series. For automotive applications, the second-order RC model is the preferred choice according to [24] and is hence used here.

Figure 2.1 shows the schematic of the second-order RC circuit. The current i and the terminal voltage V_t are measurable. The current will be positive while charging and negative when discharging.

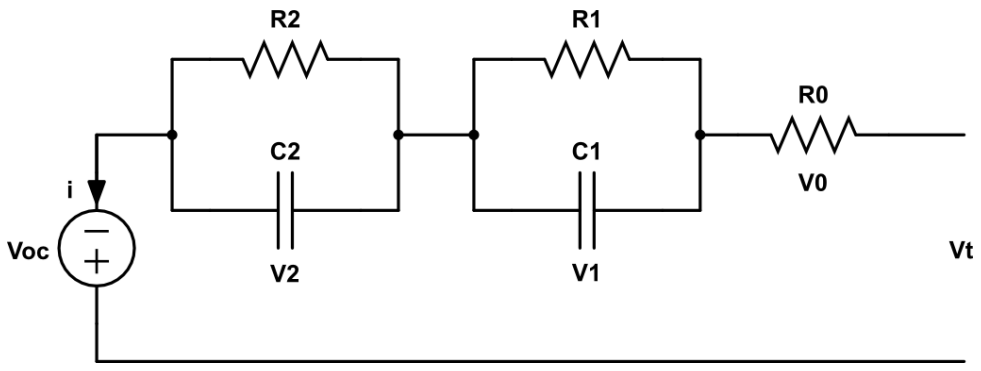


Figure 2.1: Second-order RC circuit model representing the dynamics of one cell.

2.3.1 State space model

The EC in Figure 2.1 can be described by the following equations and will later be rewritten to a state space model. Kirchhoff's second law, also known as the voltage law or loop law gives an expression of the voltages in the circuit [25]. Naming the voltage over the first RC-pair by V_1 and for the second RC-pair V_2, as well as naming the voltage over the ohmic resistance, R_0, by V_0 we get

$$V_t(t) = V_{oc}(T, z) + V_2(t) + V_1(t) + V_0(t) \tag{2.3}$$

The OCV, V_{oc}, depends on the temperature and SOC and the voltages across the capacitors can be found by using Kirchoff's first law, better known as Kirchoff's current law [25]. V_0 can be expressed using Ohm's law:

$$V_0(t) = R_0 i(t) \tag{2.4}$$

$$\dot{V}_1(t) = -\frac{1}{R_1 C_1} V_1(t) + \frac{1}{C_1} i(t) \tag{2.5}$$

$$\dot{V}_2(t) = -\frac{1}{R_2 C_2} V_2(t) + \frac{1}{C_2} i(t) \tag{2.6}$$

The terminal voltage is measurable and hence it is modeled as the measured output, y. The current is modeled as the control input to the cell which leaves V_1, V_2, and z as the states. Using Equation 2.3 - Equation 2.6 and Equation 2.1 a continuous time state space model can be formulated:

$$\underbrace{\begin{bmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{z} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} -\frac{1}{R_1 C_1} & 0 & 0 \\ 0 & -\frac{1}{R_2 C_2} & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} V_1 \\ V_2 \\ z \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} \frac{1}{C_1} \\ \frac{1}{C_2} \\ \frac{1}{Q} \end{bmatrix}}_{B} i \tag{2.7}$$

$$\underbrace{V_t}_{y} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}}_{C} x + \underbrace{R_0}_{D} i + V_{oc}(T, z) \tag{2.8}$$

2.4 Recursive least squares algorithm

The recursive least squares (RLS) algorithm is a recursive algorithm widely used for parameter estimation in online applications [26]. Suppose a signal $u(k)$ is transmitted over a channel so that it is received as

$$y(k) = u(k)\theta + e(k) \tag{2.9}$$

where k is the discrete time instance and e is noise. Collecting the p most recent samples of the system, the received signal can be written as

$$Y_p = U_p \theta + E_p \tag{2.10}$$

where

$$Y_k = \begin{bmatrix} y(k) & y(k-1) & \dots & y(k-p+1) \end{bmatrix}^T \tag{2.11}$$

$$U_k = \begin{bmatrix} u(k) & u(k-1) & \dots & u(k-p+1) \end{bmatrix}^T \tag{2.12}$$

$$E_k = \begin{bmatrix} e(k) & e(k-1) & \dots & e(k-p+1) \end{bmatrix}^T \tag{2.13}$$

The RLS filter then models the received signal

$$\hat{y}(k) = u(k)\hat{\theta}(k) \tag{2.14}$$

to estimate the original signal by minimizing the following cost function

$$\sum_{j=k-p}^{k} \lambda^{k-j}(y(j) - \hat{y}(j))^2 \tag{2.15}$$

The RLS's objective is to minimize the cost function, by updating the estimated channel $\hat{\theta}$. The cost function has a decaying factor called the forgetting factor, $\lambda \in (0, 1]$, which weights the samples in decaying order. The choice of λ depends on the importance of older versus more recent data and often has a big effect on the RLS's performance [27]. A forgetting factor set to one is called a growing window RLS algorithm and leads to all data points being weighted equally. Usually, λ is set to a value very close to one [28], [29].

Besides λ, there are two other tuning parameters: p and δ. The last mentioned parameter is a value used to initialize P, $P(0) = I\delta$, where I is the identity matrix. Usually, δ is set to a small value if the initial estimation of the filter is considered accurate, a bigger δ-value can lead to a faster convergence at the cost of overshoot. The filter order, p, determines the length of remembered samples which determines the length of Y_k. The RLS algorithm can be summarized as seen in the following equations [28]:

$$\alpha(k) = U_k - Y_k^T\hat{\theta}(k-1) \tag{2.16}$$

$$g(k) = \frac{P(k-1)Y_k}{\lambda + Y_k^T P(k-1)Y_k} \tag{2.17}$$

$$P(k) = \lambda^{-1}P(k-1) - g(k)Y_k^T\lambda^{-1}P(k-1) \tag{2.18}$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \alpha(k)g(k) \tag{2.19}$$

where $\alpha(k)$ and $g(k)$ are internal variables in the filter.

2.5 Kalman filter

The Kalman Filter (KF) is the optimal filter under certain conditions, i.e., the equations describing the system are linear in the states, and the noises are white and normally distributed with zero mean [30]:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + w_k \quad , \quad w_k \sim \mathcal{N}(0, Q_k) \\ y_k &= C_k x_k + D_k u_k + v_k \quad , \quad v_k \sim \mathcal{N}(0, R_k) \end{aligned} \tag{2.20}$$

The filter consists of a prediction step (Equation 2.21) followed by an update step (Equation 2.22). The filter outputs an estimated state covariance matrix $\hat{P}_{k|k}$ and an estimated state vector $\hat{x}_{k|k}$.

$$\begin{aligned} \hat{x}_{k|k-1} &= A_k \hat{x}_{k-1|k-1} + B_k u_k \\ P_{k|k-1} &= A_k P_{k-1|k-1} A_k^T + Q_k \end{aligned} \tag{2.21}$$

$$\begin{aligned} K_k &= P_{k|k-1} C_k (C_k P_{k|k-1} C_k^T + R_k)^{-1} \\ P_{k|k} &= (I - K_k C_k) P_{k|k-1} \\ \upsilon_k &= y_k - C_k \hat{x}_{k|k-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k \upsilon_k \end{aligned} \tag{2.22}$$

The two types of KF used in the thesis are EKF and SPKF. As mentioned in Subsection 1.1.1 the two methods are widely used in this field, although there seems to be some contradicting information regarding their complexity and accuracy. For this reason, it is interesting to compare them in this application.

2.5.1 Extended Kalman filter

The EKF is an extension of the KF in such a way that it can handle nonlinear state space models [31]. The nonlinear model

$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k) + w_k \quad , \quad w_k \sim \mathcal{N}(0, Q_k) \\ y_k &= h_k(x_k, u_k) + v_k \quad , \quad v_k \sim \mathcal{N}(0, R_k) \end{aligned} \tag{2.23}$$

is linearized in each iteration of the filter

$$\begin{aligned} F_k(x_k, u_k) &= \left. \frac{\partial f_k(x, u)}{\partial x_k} \right|_{x_k, u_k} \\ H_k(x_k, u_k) &= \left. \frac{\partial h_k(x, u)}{\partial x_k} \right|_{x_k, u_k} \end{aligned} \tag{2.24}$$

The prediction and update steps of the filter are carried out similarly to the KF, according to

$$\begin{aligned} \hat{x}_{k|k-1} &= f_k(\hat{x}_{k-1|k-1}, u_k) \\ P_{k|k-1} &= F_k(x_{k-1|k-1}, u_k) P_{k-1|k-1} F_k(x_{k-1|k-1}, u_k)^T + Q_k \end{aligned} \tag{2.25}$$

$$\begin{aligned} S_k &= H_k(x_{k|k-1}, u_k) P_{k|k-1} H_k(x_{k|k-1}, u_k)^T + R_k \\ K_k &= P_{k|k-1} H_k(x_{k|k-1}, u_k)^T S_k^{-1} \\ P_{k|k} &= P_{k|k-1} - K_k S_k K_k^T \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k (y_k - h_k(x_{k|k-1}, u_k)) \end{aligned} \tag{2.26}$$

2.5.2 Sigma point Kalman filter

The SPKF is a filter that handles a nonlinear state space model without linearization [32]. The idea behind the filter is to use deterministically chosen weighted sample points, called sigma points, to capture the complete true mean and covariance [33]. The sigma points are then propagated through the true nonlinear system to capture the posterior mean and covariance. The following equations shows how the sigma points are created:

$$\begin{aligned} P &= P^{1/2}\left(P^{1/2}\right)^T \\ \chi^{(j)} &= \hat{x} + \sqrt{N}P_j^{1/2}, \quad j = \{1, 2, \ldots, N\} \\ \chi^{(j+N)} &= \hat{x} - \sqrt{N}P_j^{1/2}, \quad j = \{1, 2, \ldots, N\} \\ W_j &= \frac{1}{2N} \end{aligned} \tag{2.27}$$

where χ are the sigma points and W_j are their corresponding weights [31]. The number of sigma points created before each prediction and update step is twice the number of states, N, which means that the total number of sigma points created in each iteration is four times the number of states. Note that $\hat{x}$ is the expected state vector and that P_j is the j:th column of P, which is the state covariance matrix.

Once the sigma points are created, they are propagated through the dynamic model to get the predicted state mean vector and the predicted state covariance matrix, earlier referred to as the posterior mean and covariance, according to

$$\begin{aligned} \hat{x}_{k|k-1} &\approx \sum_{j=1}^{2n} f_k(\chi_{k-1}^{(j)}, u_k)W_j \\ P_{k|k-1} &\approx Q_{k-1} + \sum_{j=1}^{2n}(f_k(\chi_{k-1}^{(j)}, u_k) - \hat{x}_{k|k-1})(f_k(\chi_{k-1}^{(j)}, u_k) - \hat{x}_{k|k-1})^T W_j \end{aligned} \tag{2.28}$$

The covariance of the process noise is included as Q_{k-1}. Once the prediction step is completed, new sigma points are created using the newly computed state mean vector and state covariance matrix. These new sigma points can then be used for the update step of the filter, according to

$$\begin{aligned} \hat{y}_{k|k-1} &\approx \sum_{j=1}^{2n} h_k(\chi_k^{(j)}, u_k)W_j \\ P_{xy} &\approx \sum_{j=1}^{2n}(\chi_k^{(j)} - \hat{x}_{k|k-1})(h_k(\chi_k^{(j)}, u_k) - \hat{y}_{k|k-1})^T W_j \\ S_k &\approx R_k + \sum_{j=1}^{2n}(h_k(\chi_k^{(j)}, u_k) - \hat{y}_{k|k-1})(h_k(\chi_k^{(j)}, u_k) - \hat{y}_{k|k-1})^T W_j \\ K_k &= P_{xy}S_k^{-1} \\ P_{k|k} &= P_{k|k-1} - K_k S_k K_k^T = P_{k|k-1} - P_{xy}K_k^T \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_{k|k-1}) \end{aligned} \tag{2.29}$$

Finally, the filter returns the filtered state vector $\hat{x}_{k|k}$ and the filtered state covariance matrix $P_{k|k}$.

2.5.2.1 Cholesky decomposition

To calculate $P^{1/2}$ in Equation 2.27 the Cholesky decomposition algorithm can be used [34]. It computes the lower triangular matrix

$$P = P^{1/2}\left(P^{1/2}\right)^T = \begin{bmatrix} p_{11} & 0 & 0 \\ p_{21} & p_{22} & 0 \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} p_{11} & p_{21} & p_{23} \\ 0 & p_{22} & p_{32} \\ 0 & 0 & p_{33} \end{bmatrix} \tag{2.30}$$

for the general formula of a three-by-three matrix. In order for the Cholesky decomposition to be possible the matrix P needs to be a positive definite symmetric matrix [34].

When a matrix only has positive eigenvalues it is positive definite [35]. The eigenvalues of a diagonal matrix are its diagonal elements, which means that a diagonal matrix with all diagonal elements positive is both symmetric and positive definite. This will always be the case initially, given that the initial state estimates are uncorrelated, which is a required property in the SPKF [32].

2.5.3 Measurement disturbances

To replicate the real-life system, measurement disturbances are added to the model. Even though the current is modeled as the control input, it is actually measured and is corrupted by noise [36]. Thus the input disturbance, $\tilde{w}_k$, will be added to the control input as shown by

$$x_{k+1} = Ax_k + B(u_k + \tilde{w}_k) = Ax_k + Bu_k + \underbrace{B\tilde{w}_k}_{w_k} \tag{2.31}$$

The measurement disturbance, $\tilde{v_k}$, is added to represent the uncertainty of the sensor measuring the terminal voltage. Both the input and measurement disturbances will affect the measurement model, y_k, for the estimation algorithms according to

$$y_k = Cx_k + D(u_k + \tilde{w}_k) + \tilde{v}_k = Cx_k + Du_k + \underbrace{D\tilde{w}_k + \tilde{v}_k}_{v_k} \tag{2.32}$$

The following equations show how the input and output disturbances can be translated into covariances used in the KFs, given that the disturbances $\hat{v}_k$ and $\hat{w}_k$ are uncorrelated:

$$Q_k = \mathbb{E}\{w_k w_k^T\} = \mathbb{E}\{B\tilde{w}_k\tilde{w}_k^T B^T\} = B\mathbb{E}\{\tilde{w}_k\tilde{w}_k^T\}B^T \tag{2.33}$$

$$\begin{aligned} R_k &= \mathbb{E}\{v_k v_k^T\} = \mathbb{E}\{(D\tilde{w}_k + \tilde{v}_k)(D\tilde{w}_k + \tilde{v}_k)^T\} \\ &= \mathbb{E}\{D\tilde{w}_k\tilde{w}_k^T D^T + \tilde{v}_k\tilde{v}_k^T\} = D\mathbb{E}\{\tilde{w}_k\tilde{w}_k^T\}D^T + \mathbb{E}\{\tilde{v}_k\tilde{v}_k^T\} \end{aligned} \tag{2.34}$$

2.6 Computational complexity

Computational complexity measures the computing resources that an algorithm requires when running [37]. It often refers to the number of required operations and memory storage [38]. Big $\mathcal{O}$ notation is a mathematical expression often used to describe a function's complexity, especially the function's growth rate. However, these expressions can be hard to determine. Another way of measuring computational complexity is using the computer's computational time or execution time, which is the real-time it takes for an algorithm to run. This type of measuring heavily depends on the computer's performance. For example, the number of processing units, the number of tasks running in the background, if the power cord is plugged in, size of memory, along with others. Since the computer's performance can change quickly over time, it can lead to random spikes in execution time. Calling a function multiple times allows us to distinguish any outliers. There is no convenient way in `MATLAB` to measure the memory usage in real-time, therefore this will not be further investigated [39].

The Big $\mathcal{O}$ notation of KFs is fundamentally dependent on the implementation and which algorithm is used when performing operations, such as matrix multiplication. However, according to [40], the EKFs have a complexity of $\mathcal{O}(n^2)$ where n is the number of states, whereas SPKFs have a complexity of $\mathcal{O}(n^3)$ [41]. If the SPKFs are using Cholesky decomposition, which has a complexity of $\mathcal{O}(n^3)$, the SPKFs can never have a smaller complexity than $\mathcal{O}(n^3)$ [42]. Khalid et al. [43] supports this theory, and with their implementation EKFs are less computationally complex compared to SPKFs. Despite this, Gregory L. Plett has stated that SPKFs are of the same order as EKFs but with higher accuracy [44].

The complexity of an RLS filter also depends on its implementation, but mainly on the filter length, which is a tuning parameter. According to [45], the RLS filter's arithmetic complexity is proportional to the filter length. Arithmetic complexity is the number of arithmetic operations needed to perform a computation [46]. Consequently, increasing the filter length increases the arithmetic complexity as well as the memorized samples, which can lead to more accurate estimates. Selecting the perfect length is a matter of desired performance relative to computational demand and can ultimately only be determined by empirical testing.

3
Method

The main focus of this report is to create an estimation algorithm for a multi-cell system with lower computational complexity than the estimation algorithm that estimates each cell's SOC individually. The computational demand should be reduced while maintaining accuracy and robustness. This was done by first creating a single-cell model, which was then expanded to create the multi-cell system model. From the model, measurements and SOC values were extracted. The measurements were used as input for the estimation algorithms, and the SOC values were used to get an absolute error of the estimated SOC values. Lastly, some improvements regarding computational complexity in the multi-cell model were made, which is the model used for the simulations. The first estimation algorithm created was the one estimating each cell's SOC individually. Subsequently, a variety of estimation algorithms were created. The evaluation process consisted of an accuracy test and a robustness test for several different scenarios, comparing the computational time and the absolute error of the algorithms.

The SOC estimation algorithms created depend heavily on the capacity. It was therefore interesting to continue the research by creating an SOQ estimation algorithm. To keep to the subject of computational complexity the SOQ estimation algorithms were compared with similar evaluation objectives as for the SOC. Lastly, to improve the SOQ algorithm so that it can be implemented in the SOC estimation we propose yet another algorithm, estimating SOC and SOQ jointly. However, this algorithm is a rather theoretical concept and cannot handle noise.

3.1 Cell model implementation

A discretized model is necessary for the filters and for the processor, hence the cell model was discretized with sampling time Δt and zero-order hold. The discretized model is given by

$$A_d = e^{A\Delta t} \quad , \quad B_d = \int_0^{\Delta t} e^{A(\Delta t - \tau)} d\tau B \tag{3.1}$$

where A and B are the system and input matrices in Equation 2.7. Since $C_d = C$ and $D_d = D$ the discretized measurement model parameters are the same, resulting in the following state space model:

$$\begin{cases} x_{k+1} = \underbrace{\begin{bmatrix} e^{-\frac{\Delta t}{R_1 C_1}} & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{R_2 C_2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{A_d} x_k + \underbrace{\begin{bmatrix} (1 - e^{-\frac{\Delta t}{R_1 C_1}}) R_1 \\ (1 - e^{-\frac{\Delta t}{R_2 C_2}}) R_2 \\ \frac{\Delta t}{Q} \end{bmatrix}}_{B_d} i_k \\ y_k = \underbrace{\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}}_{C} x_k + \underbrace{R_0}_{D} i_k + V_{oc}(T, z) \end{cases} \tag{3.2}$$

Here we have suppressed the time index k in A_d and B_d to simplify.

3.1.1 Single cell model

A single cell was modeled using the state space model in Equation 3.2. The parameters R_0, R_1, R_2, C_1, C_2, V_{OC} are all dependent on both temperature and SOC, and some of them also on current [47], [48]. Since the SOC, temperature, and current change each iteration, the parameters are updated online which means that the state space matrices will change as well. This was done by using pre-defined look-up tables provided by Volvo GTT. Given the temperature, SOC, and current, the look-up tables can output the parameters mentioned above. Since the look-up tables are based on experimental data, only a fixed number of points are available for each input. To get the best possible estimation of each parameter, linear interpolation is used when the inputs differ from the experimental data points.

A function was created to ensure the battery does not overcharge or go into deep discharge. This would in an EV be handled by the BMS. The function monitors the SOC and changes the sign of the input signal when SOC is close to 0% or 100%. The active safety margin was given by an arbitrary tolerance. Due to the BMS's electrochemical features, the control input in the cells is not identical to the control input set by the BMS. This is modeled by an added white Gaussian noise with a small magnitude compared to the input signal set by the BMS.

3.1.2 Multi-cell model

The single-cell model was extended to simulate any positive integer, n, of cells connected in series. Figure 3.1 depicts how the EC model is extended.

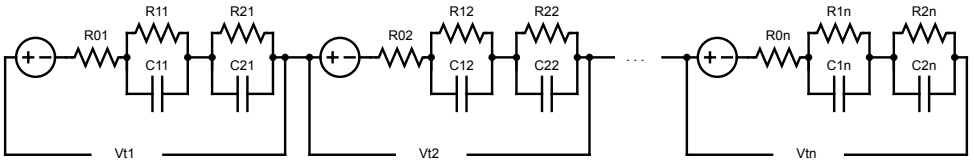


Figure 3.1: Second-order RC circuit model representing the dynamics of a multi-cell model, where the cells are connected in series.

Since the cells are connected in series, all of the cells have the same current i while the terminal voltage of each cell is unique. Equation 3.3 - Equation 3.11 show how

the state space model was extended, $j = \{1, 2\}$ denotes the RC-pair in Figure 2.1 and $l = \{1, 2, \ldots, n\}$ indicates which cell for the parameters $C_{j,l}$, $R_{j,l}$, and $V_{j,l}$. The state vector was arranged so that the first state of each cell is placed as the n first states in the combined state vector. The second and the third states follow the same pattern, resulting in a total of $3n$ states.

$$A_{d1} = \text{diag}(e^{-\frac{\Delta t}{R_{11}C_{11}}}, e^{-\frac{\Delta t}{R_{12}C_{12}}}, \ldots, e^{-\frac{\Delta t}{R_{1n}C_{1n}}}) \tag{3.3}$$

$$A_{d2} = \text{diag}(e^{-\frac{\Delta t}{R_{21}C_{21}}}, e^{-\frac{\Delta t}{R_{22}C_{22}}}, \ldots, e^{-\frac{\Delta t}{R_{2n}C_{2n}}}) \tag{3.4}$$

$$B_{d1} = \left[(1 - e^{-\frac{\Delta t}{R_{11}C_{11}}})R_{11}, (1 - e^{-\frac{\Delta t}{R_{12}C_{12}}})R_{12}, \ldots, (1 - e^{-\frac{\Delta t}{R_{1n}C_{1n}}})R_{1n}\right]^T \tag{3.5}$$

$$B_{d2} = \left[(1 - e^{-\frac{\Delta t}{R_{21}C_{21}}})R_{21}, (1 - e^{-\frac{\Delta t}{R_{22}C_{22}}})R_{22}, \ldots, (1 - e^{-\frac{\Delta t}{R_{2n}C_{2n}}})R_{2n}\right]^T \tag{3.6}$$

$$B_{d3} = \left[\frac{\Delta t}{Q_1}, \frac{\Delta t}{Q_2}, \ldots, \frac{\Delta t}{Q_n}\right]^T \tag{3.7}$$

$$x_k = [V_{11}, V_{12}, \ldots, V_{1n}, V_{21}, V_{22}, \ldots, V_{2n}, z_1, z_2, \ldots, z_n]^T \tag{3.8}$$

$$y_k = [V_{T1}, V_{T2}, \ldots, V_{Tn}]^T \tag{3.9}$$

$$x_{k+1} = \underbrace{\begin{bmatrix} A_{d1} & 0 & 0 \\ 0 & A_{d2} & 0 \\ 0 & 0 & I_n \end{bmatrix}}_{3n\text{x}3n} x_k + \underbrace{\begin{bmatrix} B_{d1} \\ B_{d2} \\ B_{d3} \end{bmatrix}}_{3n\text{x}1} i_k \tag{3.10}$$

$$y_k = \underbrace{\begin{bmatrix} I_n & I_n & 0_n \end{bmatrix}}_{n\text{x}3n} x_k + \underbrace{\begin{bmatrix} R_{01} \\ R_{02} \\ \vdots \\ R_{0n} \end{bmatrix}}_{n\text{x}1} i_k + \underbrace{\begin{bmatrix} V_{oc}(T, z_1) \\ V_{oc}(T, z_2) \\ \vdots \\ V_{oc}(T, z_n) \end{bmatrix}}_{n\text{x}1} \tag{3.11}$$

3.1.3 Cell imbalances

Imbalances between multiple cells connected in series are inevitable [49]. These imbalances are deviations in individual cell voltage from manufacturing, operation, and thermal conditions. The imbalances were included in the simulation by altering the parameters between different cells. Each cell has its parameters multiplied with random factors taken from a uniform distribution:

$$\begin{aligned} j = \{1, 2\}, \qquad & l = \{1, 2, \ldots, n\} \\ R_{0,l}(z_l, T_l) = R_{0,nom}(z_l, T_l) \cdot \mathcal{R}_{0,l}, \quad & \mathcal{R}_{0,l} \sim \mathcal{U}(1, 1.1) \quad \forall l \\ R_{j,l}(i, z_l, T_l) = R_{j,nom}(i, z_l, T_l) \cdot \mathcal{R}_{j,l}, \quad & \mathcal{R}_{j,l} \sim \mathcal{U}(1, 1.1) \quad \forall j, l \\ C_{j,l}(i, z_l, T_l) = C_{j,nom}(i, z_l, T_l) \cdot \mathcal{C}_{j,l}, \quad & \mathcal{C}_{j,l} \sim \mathcal{U}(0.9, 1) \quad \forall j, l \\ Q_l = Q_{nom} \cdot \mathcal{Q}_l, \quad & \mathcal{Q}_l \sim \mathcal{U}(0.9, 1) \quad \forall l \end{aligned} \tag{3.12}$$

We call these random factors $(\mathcal{R}_{0,l}, \mathcal{R}_{j,l}, \mathcal{C}_{j,l}, \mathcal{Q}l)$ "the cell imbalance factors". This means that the parameters are still updated online each iteration but scaled to

represent imbalances between cells. The temperature of each cell depends on its location in the battery. The cells are assumed to be arranged in a linear fashion, thus cell 1 and cell n are considered to be located at the ends of the battery and have a temperature of $20°C$. The cell in the middle has a temperature of $25°C$ and the rest linearly decreases to $20°C$ towards the ends, i.e.,

$$T_l = 25 - \frac{5}{n/2}\,\mathrm{abs}\,(l - n/2) \tag{3.13}$$

3.1.4 Computationally efficient state space model

The state space model was rewritten according to

$$\begin{aligned} A_{e1} &= \begin{bmatrix} e^{-\frac{\Delta t}{R_{11}C_{11}}} & e^{-\frac{\Delta t}{R_{12}C_{12}}} & \dots & e^{-\frac{\Delta t}{R_{1n}C_{1n}}} \end{bmatrix}^T \\ A_{e2} &= \begin{bmatrix} e^{-\frac{\Delta t}{R_{21}C_{21}}} & e^{-\frac{\Delta t}{R_{22}C_{22}}} & \dots & e^{-\frac{\Delta t}{R_{2n}C_{2n}}} \end{bmatrix}^T \\ A_{e3} &= \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \end{aligned} \tag{3.14}$$

$$x_{k+1} = \begin{bmatrix} A_{e1} \\ A_{e2} \\ A_{e3} \end{bmatrix} \odot x_k + \begin{bmatrix} B_{d1} \\ B_{d2} \\ B_{d3} \end{bmatrix} i_k \tag{3.15}$$

$$y_k = \begin{bmatrix} V_{11} \\ V_{12} \\ \vdots \\ V_{1n} \end{bmatrix} + \begin{bmatrix} V_{21} \\ V_{22} \\ \vdots \\ V_{2n} \end{bmatrix} + \begin{bmatrix} R_{01} \\ R_{02} \\ \vdots \\ R_{0n} \end{bmatrix} i_k + \begin{bmatrix} V_{oc}(T, z_1) \\ V_{oc}(T, z_2) \\ \vdots \\ V_{oc}(T, z_n) \end{bmatrix} \tag{3.16}$$

By doing this, computations resulting in sums of zeros were avoided. Note that A in Equation 3.14 is a vector instead of a matrix and hence it uses the Hadamard product $\odot$, which takes each element and multiplies it with the element in the corresponding row in the state vector. This improvement resulted in the number of multiplications being reduced from $9n^2$ to $3n$. In Equation 3.16 no multiplications with the state vector are required, instead, the corresponding states are extracted from the state vector. In total, this state space model has $9n$ multiplications. Equation 3.14 - Equation 3.16 represent the multi-cell model used in the upcoming experiments.

3.2 SOC estimation algorithms

The measurement model, Equation 2.8, is nonlinear and was linearized so it could be applied in the EKF. A Taylor expansion of the nonlinear term $V_{oc}(T, z)$, using only the first order derivative was used to write the measurement model as a linear function of the state, i.e,

$$y = \underbrace{\begin{bmatrix} 1 & 1 & \frac{\partial V_{OC}}{\partial z} \end{bmatrix}}_{C_{linearized}} x + \underbrace{R_0}_{D_{linearized}} i \tag{3.17}$$

This resulted in the linearized discrete state space model:

$$\begin{cases} x_{k+1} = \underbrace{\begin{bmatrix} e^{-\frac{\Delta t}{R_1 C_1}} & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{R_2 C_2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{A_d} x_k + \underbrace{\begin{bmatrix} (1 - e^{-\frac{\Delta t}{R_1 C_1}}) R_1 \\ (1 - e^{-\frac{\Delta t}{R_2 C_2}}) R_2 \\ \frac{\Delta t}{Q} \end{bmatrix}}_{B_d} i_k \\ y_k = \underbrace{\begin{bmatrix} 1 & 1 & \frac{\partial V_{OC}}{\partial z} \end{bmatrix}}_{C_{linearized}} x_k + \underbrace{R_0}_{D_{linearized}} i_k \end{cases} \tag{3.18}$$

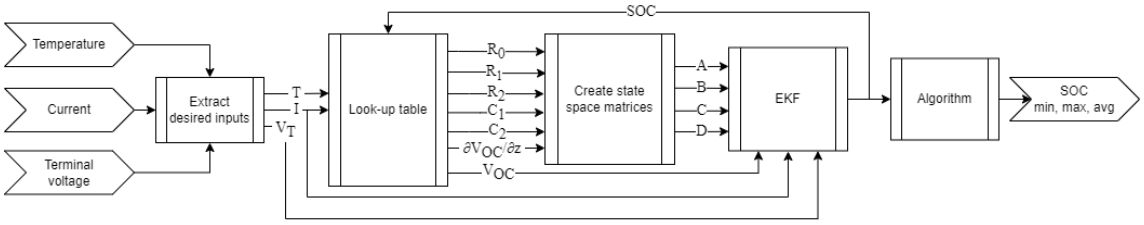
For n cells connected in series, it presents as the following:

$$C_{d3} = \text{diag}\left(\frac{\partial V_{OC1}}{\partial z_1}, \frac{\partial V_{OC2}}{\partial z_2}, \ldots, \frac{\partial V_{OCn}}{\partial z_n} \right) \tag{3.19}$$

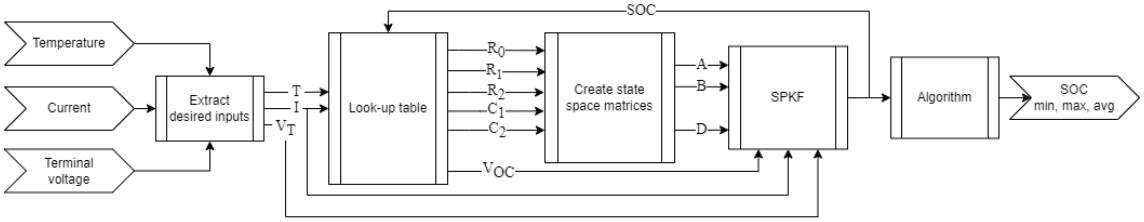
$$\begin{cases} x_{k+1} = \begin{bmatrix} A_{d1} & 0 & 0 \\ 0 & A_{d2} & 0 \\ 0 & 0 & I_n \end{bmatrix} x_k + \begin{bmatrix} B_{d1} \\ B_{d2} \\ B_{d3} \end{bmatrix} i_k \\ y_k = \begin{bmatrix} I_n & I_n & C_{d3} \end{bmatrix} x_k + \begin{bmatrix} R_{01} \\ R_{02} \\ \vdots \\ R_{0n} \end{bmatrix} i_k \end{cases} \tag{3.20}$$

The algorithms' inputs are the measured terminal voltages, the temperatures, and the measured current. The mentioned values are extracted from the multi-cell model, a separate `Simulink` file, and as previously mentioned the terminal voltage and current are corrupted by additive white Gaussian noise. Additionally, the algorithms receive the capacity, cell imbalance factors, and the initial state vector. When estimating SOC for several cells connected in series, it is the maximum, minimum, and average SOC that are of interest since they are used in the BMS. The maximum, minimum, and average SOC is the output of each algorithm.

As mentioned earlier, in Section 2.5, each estimation method has two versions. One using EKF and the other using SPKF. All of the algorithms can be divided into equal subparts, EKF and SPKF will differ slightly, see Figure 3.2. First, the input signals, temperatures, current and terminal voltages are extracted and reshaped if needed. Then the extracted inputs together with the estimated SOC are inserted in the lookup table where the parameters are retrieved and later converted to state space matrices. The state space matrices are then used in the chosen KF. The function converting the parameters to state space matrices also includes the cell imbalances, as explained in Subsection 3.1.3 with some modification depending on the algorithm. Lastly, the estimated states are converted to the final output, the minimum, maximum and average SOC. In Figure 3.2b it is easy to spot that the SPKF does not require $\partial V_{OC}/\partial z$ nor the C matrix since it does not linearize the state space model.



(a) Visual representation of the estimation algorithms using EKF.



(b) Visual representation of the estimation algorithms using SPKF.

Figure 3.2: Visual representation of the SOC estimation algorithms.

3.2.1 Extended Kalman filter implementation

The implementation of the EKF is the same for all algorithms, as seen in Subsection 2.5.1. In Equation 2.23, f_k is equal to Equation 3.10, h_k is equal to Equation 3.11, F_k is equal to the A-matrix in Equation 3.10 and H_k is equal to the C-matrix in Equation 3.20. The noise covariance matrices, R_k and Q_k are generated with the following code:

```
% Generate Q_k and R_k
% cov_meas_noise and cov_input_noise are scalars
n = number_of_cells;
R = diag(D)*cov_meas_noise*diag(D)'+ eye(n)*cov_input_noise;
Q = diag(diag(B*cov_input_noise*B'));
```

3.2.2 Sigma point Kalman filter implementation

The implementation of the SPKF is also the same for all algorithms, as seen in Subsection 2.5.2. In Equation 2.28 and Equation 2.29, f_k is given by Equation 3.10 and h_k is given by Equation 3.16. The state covariance matrix, P, is used to calculate $P^{1/2}$ through the Choleskey decomposition function `chol(P,'lower')` in `Matlab`. The noise covariance matrices, R_k and Q_k, are generated using the same code as for the EKF.

3.2.3 The SOC algorithms

We have given each algorithm a name depending on the number of KFs used and the type of KF. XEKF has as many EKFs as the number of cells, likewise, XSPKF has as many SPKFs as the number of cells. 3EKF and 3SPKF follow the same naming convention using 3 EKFs or SPKFs, respectively. 1EKF, 1EKF$_{log}$, 1SPKF,

and 1SPKF$_{log}$ uses only one EKF or SPKF, respectively. Below are explanations of their implementation.

3.2.3.1 XKF

This method uses one KF for each cell, generating an accurate estimation of each cell's SOC. All of the input signals are directly used as input to the lookup tables, giving the parameters R_0, R_1, R_2, C_1, C_2, V_{OC} and $\frac{\partial V_{OC}}{\partial z}$ which are then inserted to the EKFs or SPKFs. The state space matrices are computed in the KFs. This is because we used the `MATLAB` function `sparse` and `speye` for the A and C matrices. `Sparse` and `speye` operate similarly by creating a matrix that only stores nonzero values, thus reducing the required memory and the number of operations. `Sparse` matrices outside a `MATLAB function block` is not allowed, but by computing the matrices inside the KF function we surpassed this problem. When computing the state space matrices, cell imbalances are included as described in Subsection 3.1.3. The average SOC value is then calculated by taking the mean of all SOC values, while the minimum and maximum value is computed by extracting the highest and lowest SOC value. This is the algorithm that we used for comparison with the other algorithms concerning computational time, accuracy, and robustness.

3.2.3.2 3KF

3KF has three KFs, one for each cell with the maximum, minimum, and average SOC. Experimenting with different cell imbalance factors showed that it is the difference in the Q-factors that produces the largest change in the SOC. This can also be deducted by observing Equation 2.1. Thus by selecting the cell with the highest capacity, we select the cell with the smallest SOC value, and vice versa. The average SOC value is calculated by using the average terminal voltage and the average parameter factors.

Firstly the input signals of the cells with the smallest and largest Q factors have to be extracted as well as computing an average terminal voltage and temperature. As Figure 3.2 depicts, the reshaped input signals are then used for the lookup tables, following the function converting the parameters to state space matrices. Here the parameters are multiplied by their corresponding cell imbalance factor, $\mathcal{R}$, $\mathcal{C}$, $\mathcal{Q}$. Regarding the average SOC, mean cell imbalance factors were used. The estimated states are then the estimated minimum, maximum, and average SOC values.

3.2.3.3 1KF

1KF has only one KF, estimating a random cell's SOC for each driving cycle. This method utilizes the idea that the current is equal for each cell when they are connected in series. Equation 2.1 states that the integral of the current is the same for each cell and can be viewed as a shared variable, c_k, resulting in

$$z_{k,l} = z_{k-1,l} + \frac{1}{Q_l}\int_{t_{k-1}}^{t_k} i(\tau)d\tau \approx z_{k-1,l} + \frac{\Delta t i_{k-1}}{Q_l} \tag{3.21}$$

$$\Delta z_{k,l} = \frac{c_k}{Q_l} \tag{3.22}$$

We have made two algorithms using only one KF. The first approach is called 1KF while the other is called $1KF_{log}$. The two implementations are identical up until the point where they calculate the minimum, average and maximum SOC from the estimated states. First, a random index is given, representing a cell. The input signals with that index are then extracted and inserted into the lookup tables. The parameters are then multiplied with the cell imbalance factors (see Subsection 3.1.3) of the selected indices before the state space matrices are created and inserted into the KF.

3.2.3.3.1 Approach 1, 1KF The estimated SOC of the random cell is used to calculate the variable c, using Equation 3.22. Using the same equation and the estimated c, SOC can be calculated for all cells. The SOC values are then used to simply pick the minimum and maximum SOC, while the average SOC is computed.

3.2.3.3.2 Approach 2, $1KF_{log}$ This method also estimates the variable c from the random cell's SOC, but instead of calculating SOC for all cells, the distribution of the SOQ is used to estimate the mean SOC. The minimum and maximum SOC estimations are calculated by using the maximum and minimum capacities and the estimated c, according to

$$z(t) > z(t_0) \implies \begin{cases} z_{max} = z_{t_0} + \frac{1}{Q_{min}}\int_{t_0}^{t} i(\tau)d\tau \\ z_{min} = z_{t_0} + \frac{1}{Q_{max}}\int_{t_0}^{t} i(\tau)d\tau \end{cases} \tag{3.23}$$

since $\int_{t_0}^{t} i(\tau)d\tau$ must be positive. Similarly,

$$z(t) < z(t_0) \implies \begin{cases} z_{max} = z_{t_0} + \frac{1}{Q_{max}}\int_{t_0}^{t} i(\tau)d\tau \\ z_{min} = z_{t_0} + \frac{1}{Q_{min}}\int_{t_0}^{t} i(\tau)d\tau \end{cases} \tag{3.24}$$

As can be observed in the equations the calculated maximum and minimum SOC depend on whether the SOC has increased or decreased during the active cycle. If the distribution of the different cell capacities Q_i is known, the mean SOC can be estimated with a different approach while still only utilizing one KF. This approach mathematically derives the distribution of the SOC in order to estimate the average SOC. Consider the definition:

$$f_{\Delta Z}(\Delta z) = \frac{\partial}{\partial z} F_{\Delta Z}(\Delta z) \tag{3.25}$$

where

$$
\begin{aligned}
F_{\Delta Z}(\Delta z) = \Pr(\Delta Z \leq \Delta z) &= \left\{ \Delta z \geq \Delta Z = \frac{c}{Q_{rv}} \right\} = \Pr\left(Q_{rv} \geq \frac{c}{\Delta z} \right) \\
&= 1 - \Pr\left(Q_{rv} < \frac{c}{\Delta z} \right) = 1 - F_{Q_{rv}}\left(\frac{c}{\Delta z} \right)
\end{aligned} \tag{3.26}
$$

where Q_{rv} is a random variable for the capacity. Then,

$$
f_{\Delta Z}(\Delta z) = \frac{\partial}{\partial \Delta z} F_{\Delta Z}(\Delta z) = - \underbrace{f_{Q_{rv}}\left(\frac{c}{\Delta z} \right)}_{f_{Q_{rv}}(q)} \cdot \left(\frac{-c}{\Delta z^2} \right) \tag{3.27}
$$

and

$$
\mathbb{E}\{\Delta z\} = \int_{-\infty}^{\infty} \Delta z f_{\Delta Z}(\Delta z) d\Delta z = \int_{z_{min}}^{z_{max}} \frac{c}{\Delta z} f_{Q_{rv}}\left(\frac{c}{\Delta z} \right) d\Delta z \tag{3.28}
$$

Equation 3.25 demonstrates the relation between the probability density function (PDF) on the left-hand side and the cumulative distribution function (CDF) on the right-hand side. Equation 3.26 - Equation 3.28 derives the formula for calculating the average SOC with Q given by any distribution with PDF $f_{Q_{rv}}$. If the distribution of the cell capacities is uniform the PDF is defined as Equation 3.29 and the mean SOC can then be derived as

$$
Q_{rv} \sim \mathcal{U}(Q_{min}, Q_{max}) \quad \Longrightarrow \quad f_{Q_{rv}}\left(\frac{c}{\Delta z} \right) = \frac{1}{Q_{max} - Q_{min}} \tag{3.29}
$$

$$
f_{\Delta Z}(\Delta z) = \frac{c}{\Delta z^2 (Q_{max} - Q_{min})} \tag{3.30}
$$

$$
\begin{aligned}
\mathbb{E}\{\Delta z\} &= \int_{\frac{c}{Q_{max}}}^{\frac{c}{Q_{min}}} \frac{c}{\Delta z (Q_{max} - Q_{min})} d\Delta z \\
&= \frac{c}{Q_{max} - Q_{min}} \int_{\frac{c}{Q_{max}}}^{\frac{c}{Q_{min}}} \frac{1}{\Delta z} d\Delta z = \frac{c \cdot (\ln(Q_{max}) - \ln(Q_{min}))}{Q_{max} - Q_{min}}
\end{aligned} \tag{3.31}
$$

3.2.4 Computational complexity

As stated previously in Section 2.6, the complexity of the KF depends on its implementation. The main difference between the filters is the computations of the sigma points using the Cholesky decomposition which is computationally demanding. Additionally the functions `sparse` and `speye` was used to decrease the computational complexity of XKF. These functions were applied on the matrices with a large number of zeros, which were the A and the C matrices for XKF. The two functions drastically lower the computation time for XKF, but if implemented in the other algorithms, the computational time stays the same or even increases. This is due to that the A and C matrices are relatively small, thus requiring less space and time to create them than it would using the `sparse` and `speye` functions.

Since MATLAB is not open source, we do not have access to the implementations of certain operations. For example, matrix multiplication and hence complexity is unknown. Therefore, we use the computational time to measure the algorithm's complexity, which is also recommended by MATLAB [39].

3.3 SOC experiments

The experiments consist of several simulations where the control input, initial SOC, and the number of cells are altered. The initial values of V_1 and V_2 are set to 0 for all cells and experiments. The covariance matrix P has the same initial matrix across the experiments, a null matrix for the EKF and an identity matrix for the SPKF. There are three different control input scenarios, as well as initial SOC values and the number of cells. This results in a total of 27 different simulation scenarios listed in Table 3.1. We created two current input scenarios, input indices 1 and 3, that should capture realistic charging and discharging events. Input index 2 is a real driving cycle including mostly discharging and is called the Urban Dynamometer Driving Schedule (UDDS). The inital SOC, z_0, is altered between scenarios, which allows us to see if it affects the error but also gives us more data to draw conclusions from.

Table 3.1: The 27 different SOC simulation scenarios where z_0 denotes the initial SOC and input index denotes different current inputs seen in Figure 3.4.

z_0	0.2	0.2	0.2	0.5	0.5	0.5	0.8	0.8	0.8
input index	1	2	3	1	2	3	1	2	3
number of cells	100	100	100	100	100	100	100	100	100
scenario number	1	2	3	4	5	6	7	8	9
z_0	0.2	0.2	0.2	0.5	0.5	0.5	0.8	0.8	0.8
input index	1	2	3	1	2	3	1	2	3
number of cells	200	200	200	200	200	200	200	200	200
scenario number	10	11	12	13	14	15	16	17	18
z_0	0.2	0.2	0.2	0.5	0.5	0.5	0.8	0.8	0.8
input index	1	2	3	1	2	3	1	2	3
number of cells	300	300	300	300	300	300	300	300	300
scenario number	19	20	21	22	23	24	25	26	27

Each simulation scenario was evaluated by taking the absolute error between the cell model's and the estimation algorithms' minimum, maximum, and average SOC. This evaluates the estimation algorithm's accuracy. The computational complexity of the estimation algorithms was evaluated based on the execution time of the scenarios.

In addition to the accuracy tests, the same simulations were carried out in the same framework with the only difference being an added temperature noise to give some uncertainty in the state space matrices. These tests aim to check the robustness of the estimation algorithms. Figure 3.3 depicts the cells with the highest temper-

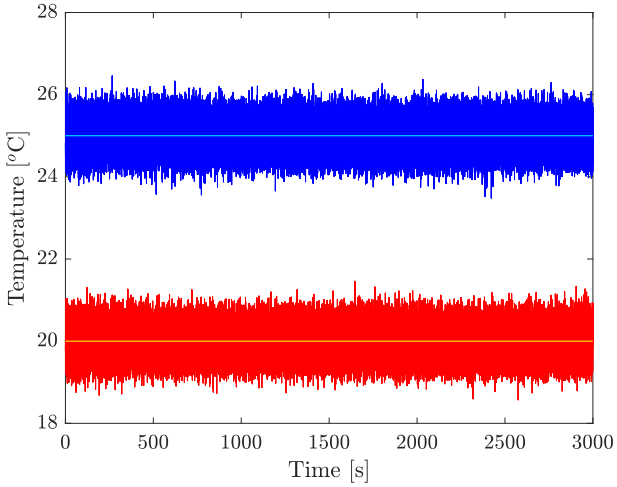


Figure 3.3: The highest and lowest temperature of 100 cells with and without added white Gaussian noise.

ature and lowest temperature and their corresponding noise, which is used in the robustness tests.

Figure 3.4a, Figure 3.4b and Figure 3.4c show the different control inputs for the initial SOC value 0.5 with 100 cells. Input current one and three are different when altering the initial value. However, they remain consistent with the number of cells. All of the scenarios (simulated with 100 cells) can be found in Appendix A. Due to the natural imbalance between cells, see Subsection 3.1.3, the terminal voltages are different for each cell. Only the cells with the largest and smallest terminal voltages are shown in the figures. Inputs one and three have a maximum difference between the largest and smallest terminal voltage of around 0.8 V.

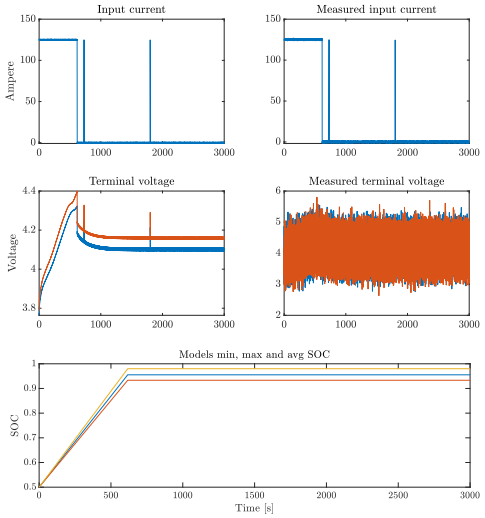
3.4 SOQ estimation

The SOC estimation algorithms heavily depend on the SOQ, which is mentioned in Section 2.1. Therefore, it was of interest to continue the research by creating a computationally efficient and accurate online SOQ estimation algorithm. When estimating SOC we assume that the capacity is constant and known, when in reality it decreases over time and is not directly measurable, as mentioned in Section 2.2. Since the capacity of each cell is assumed to be known when estimating SOC, the SOQ estimation needs to estimate every cell's capacity.

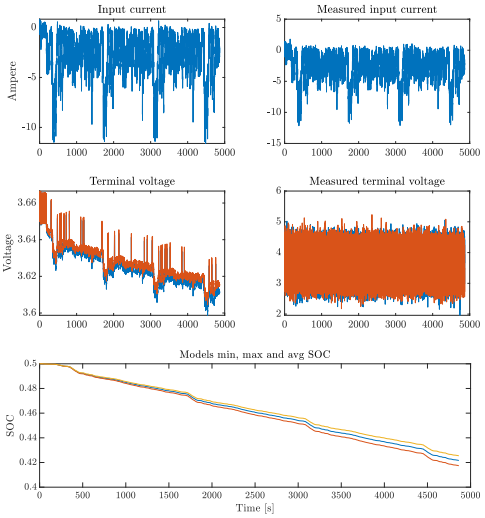
3.4.1 The SOQ algorithms

An RLS filter can be used to filter out inaccurate measurements (see Section 2.4). The RLS filter models the system as

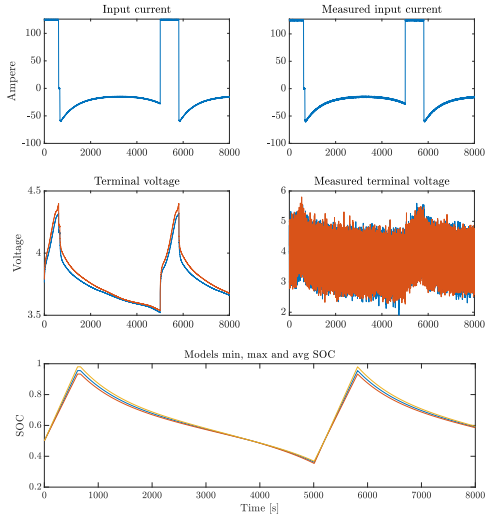
$$y(k) = u(k)\theta + e(k) \tag{3.32}$$



(a) The first row shows input current 1, a constant charge until SOC reaches 100% with some tolerance, the input current is then set to zero. The noises in the plots are numerical artefacts.



(b) The first row shows input current 2, which is the UDDS [50].



(c) The first row shows input current 3, constant charging until SOC reaches 100% with some tolerance followed by a discharge.

Figure 3.4: The first row shows the corresponding input current. The second row shows the terminal voltage for the cell with the smallest and largest terminal voltage. The right-hand side is the left-hand side measured. The last plot shows the resulting min, max, and average SOC. This scenario is with a z_0 of 0.5 and 100 cells.

into

$$\hat{y}(k) = u(k)\hat{\theta}(k) \tag{3.33}$$

where its implementation uses the discretized version of Equation 2.2:

$$Q = \frac{\Delta t \cdot i_k}{\Delta z_k} \tag{3.34}$$

The implementation can be seen in the following equations

$$\hat{y}(k) = Q\Delta z_k \tag{3.35}$$
$$y(k) = \Delta t \cdot i_k \tag{3.36}$$

where t is the sampling time, i_k is the current, $u(k)$ is difference in SOC, Δz_k, and $\hat{\theta}$ is the estimated capacity, Q.

When the current or the Δz approaches zero CC cannot be applied. According to Equation 2.2 the capacity is equal to zero when the current is zero and infinity when the Δz is zero. To avoid faulty capacity estimation a safety margin was introduced so that when either the current or Δz is below the safety margin, no update is performed. Instead, the old capacity estimate is kept. This may lead to faulty capacity estimations to some extent but will avoid unreasonable values.

Two methods estimating SOQ are proposed, XRLS and 1RLS, both using RLS filters, as can be seen in Figure 3.5 and Figure 3.6. Both filters retrieve the SOC from XEKF since it has a faster execution time than XSPKF and provides an accurate SOC for every cell. The current is extracted from the cell model.

3.4.1.1 XRLS

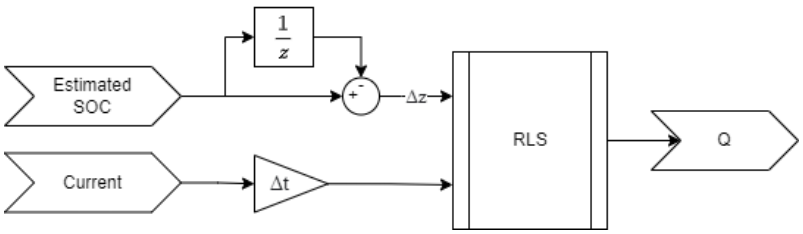


Figure 3.5: Diagram that depicts the process of the SOQ estimation algorithm XRLS.

XRLS has one RLS filter for each cell, computing each cell's SOQ. Figure 3.5 portrays a simple representation of how the algorithm works. Firstly the SOC, extracted from XEKF, is time delayed one sample in order to retrieve Δz. The current is multiplied by the sampling time, and together with Δz they are inserted into the RLS filter, as seen in Equation 3.35 and Equation 3.36. Δz is an array, thus the implementation of the RLS filter will differ from what is explained in Section 2.4. A

for-loop is used to iterate through Equation 2.16 - Equation 2.19. There will be as many iterations as the number of cells, implying that there is one RLS filter for each cell.

3.4.1.2 1RLS

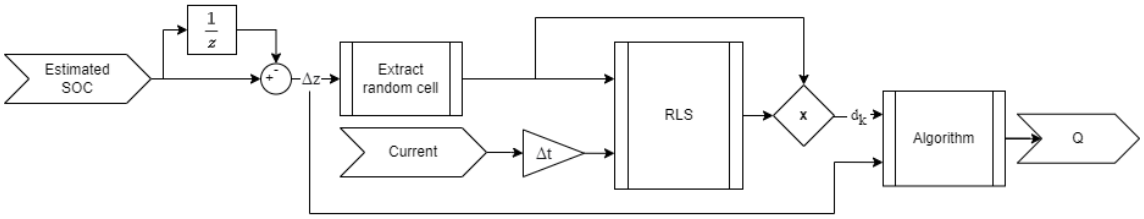


Figure 3.6: Diagram that depicts the process of the SOQ estimation algorithm 1RLS.

This method is based on the same reasoning as 1KF. Namely that the integrated current is consistent for every cell and thus only one filter is required regardless of the number of cells. Figure 3.6 portrays a simple representation of how the algorithm works. Firstly the SOC, extracted from XEKF, is time delayed one sample in order to retrieve Δz. 1RLS then randomly selects one cell's difference in SOC, $\Delta z_{j,k}$, and it is inserted into the RLS filter together with the current multiplied by the sampling time. The RLS algorithm returns the estimated capacity, $Q_{j,k}$, for the randomly picked cell j which is then multiplied with $\Delta z_{j,k}$ resulting in the variable d_k.

$$d_k = \Delta z_{j,k} \cdot Q_{j,k}, \qquad j \in \{1, 2, \ldots, n\} \tag{3.37}$$

Similar to the variable, c_k, calculated in Subsubsection 3.2.3.3, d_k should be equal for all cells. The calculated variable is together with Δz_k inserted into the function estimating the capacity of all other cells as shown in Equation 3.38.

$$Q_{k,l} = \frac{d_k}{\Delta z_{k,l}}, \qquad l = \{1, 2, \ldots, n\} \tag{3.38}$$

3.5 SOQ experiments

The performance of the algorithms, the absolute percentage error in estimated capacity compared to true capacity and their execution time determines which of the algorithms is superior. The conclusion was drawn through experiments. The experiments use the UDDS driving cycle as control input with an initial SOC of 0.5 (see Figure 3.4b). These parameters are not the main focus of this experiment and therefore do not need to be changed. Instead, there are three different scenarios where the model's capacity changes for 100, 200, and 300 cells, resulting in a total of 9 simulations, listed in Table 3.2. There are three tuning parameters in the RLS filer, the filter length p, the forgetting factor λ, and the initial value for the filter weight

δ. These parameters will be the same for the two estimation algorithms and will stay consistent during the experiments. The tuning parameters were determined through empirical testing, as $\lambda = 0.999$ and $\delta = 1$. The minimal filter length of $p = 2$ was chosen since this resulted in minimal computational cost, as explained in Section 2.6, while still performing satisfactorily.

The following scenarios are extreme, with the purpose of highlighting and magnifying the comparison between the two algorithms, but also since we do not have the resources to simulate over several driving cycles. In reality, it would take over 100 full cycles for the capacity to drop 5 percent and in some scenarios even more cycles depending on the battery and the conditions during the experiments [51], [52]. Since the rate of change is magnified so will the absolute errors, this should be taken into account when observing the results.

Table 3.2: The 9 different simulation scenarios for the SOQ experiments.

scenario number	1	2	3	4	5	6	7	8	9
z_0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
input index	2	2	2	2	2	2	2	2	2
number of cells	100	200	300	100	200	300	100	200	300
Q-scenario	1	1	1	2	2	2	3	3	3

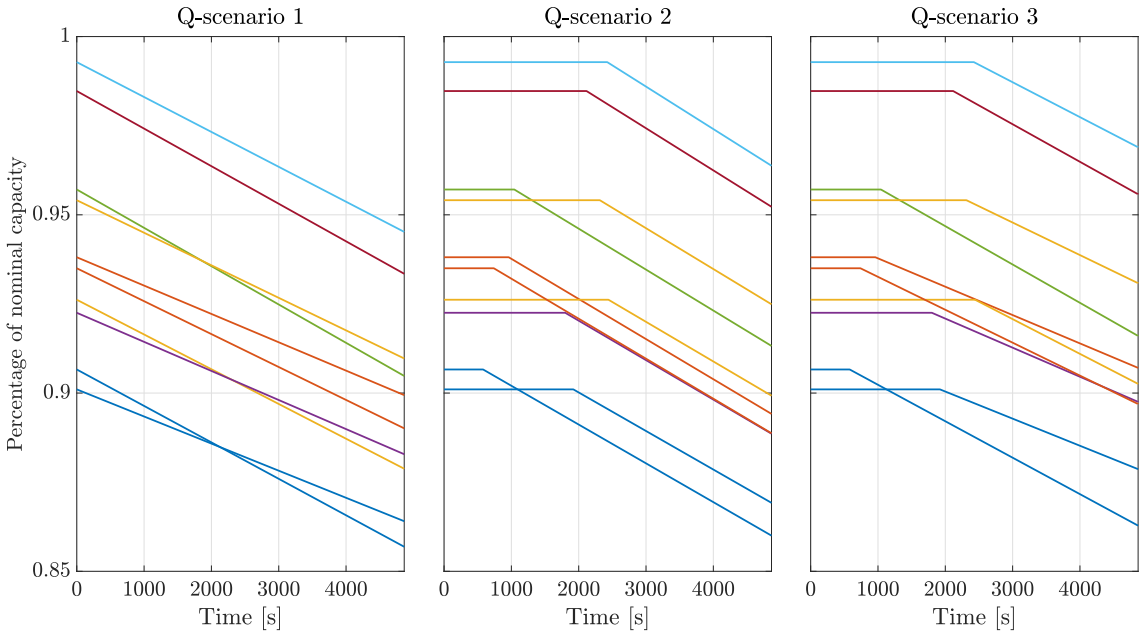


Figure 3.7: Three scenarios where the capacity decays over time for 10 cells. The y-axis represents the cell capacity normalized over the nominal capacity, a value of 1 implies the capacity is equal to the nominal capacity. Scenario 1, seen to the left, demonstrates each cell's different rates of change. Scenario 2, seen in the middle, demonstrates all cell's equal rates of change decrementing at different points in time. Scenario 3, seen to the right, demonstrates a combination of the two previous scenarios.

Figure 3.7 depicts the scenarios where the model's capacity changes for 10 cells. The first scenario aims to test the algorithm's ability to handle different rates of change while the second aims to test if it can handle varieties at different points in time. The last scenario tests the algorithm's ability to handle a combination of the previous two. The following equations

$$\begin{aligned} \text{Slope}_l &= 10^{-5} \cdot (0.3084 \cdot \mathcal{S}_l - 1.028), \quad \mathcal{S} \sim \mathcal{U}(0,1) \\ \text{Time}_l &= 2000 \cdot \mathcal{T}_l + 500 \text{ [s]}, \quad \mathcal{T} \sim \mathcal{U}(0,1) \end{aligned} \tag{3.39}$$

shows how the random rate of change for each cell is generated and how the random point in time is selected. The randomly generated slope ensures that the capacity decreases by 3.5 to 5 percent during the simulation and the point in time will be somewhere between 500 and 2500 seconds in the simulation. From these equations, the three Q-scenarios can be scaled to any number of cells.

3.6 SOC and SOQ joint estimation algorithm

The previous SOQ estimation algorithms, described in Section 3.4, retrieve the Δz from XEKF which estimates SOC given the true capacity. In fact, the capacity should be retrieved from the SOQ estimation and is thus not known beforehand. If the estimated SOQ was inserted in the XEKF, a mathematical loop would occur. Since both algorithms utilize CC to estimate their respective parameters, the estimated SOQ is used to estimate the SOC which is then used to estimate the SOQ. To avoid this loop the two estimation algorithms cannot both utilize CC. In this section, we propose an SOQ estimation algorithm that estimates the capacity which can be implemented in the SOC estimations. The proposed algorithm is a theoretical concept, it does not include any type of filter and can thus not handle disturbances. If the algorithm performs satisfactorily we could argue that it is possible to use this concept in future work.

There are offline estimation methods that can estimate SOQ. For example, when the battery is at rest and the dynamic voltages are equal to zero the OCV can be directly measured, which is then used to compute the capacity [53]. Although this is possible, the scope of the thesis is to increase computational efficiency in online estimations.

The proposed SOQ estimation algorithm estimates SOC without applying CC and thus the SOC can be used to estimate SOQ with CC. The SOC is obtained by first estimating the OCV and then converting it to SOC with a look-up table provided by Volvo GTT. The OCV was calculated according to

$$V_{OC}(z,T) = V_t(t) - V_2(t) - V_1(t) - R_0(z,T)i(t) \tag{3.40}$$

where the current and terminal voltage is measured and the voltages over the RC-pairs are estimated.

Figure 3.8 illustrates how the estimation algorithm operates. Temperature, terminal voltage, current, and the estimated capacity is inserted into the SOC_{OCV} block which computes an estimated SOC, z. The difference in SOC, Δz, is then calculated and inserted together with the current multiplied with the sampling time in the algorithm block, which computes the SOQ, as stated in Equation 3.34. The safety margin mentioned in Subsection 3.4.1 is implemented in the algorithm to avoid the capacity from reaching zero or infinity.

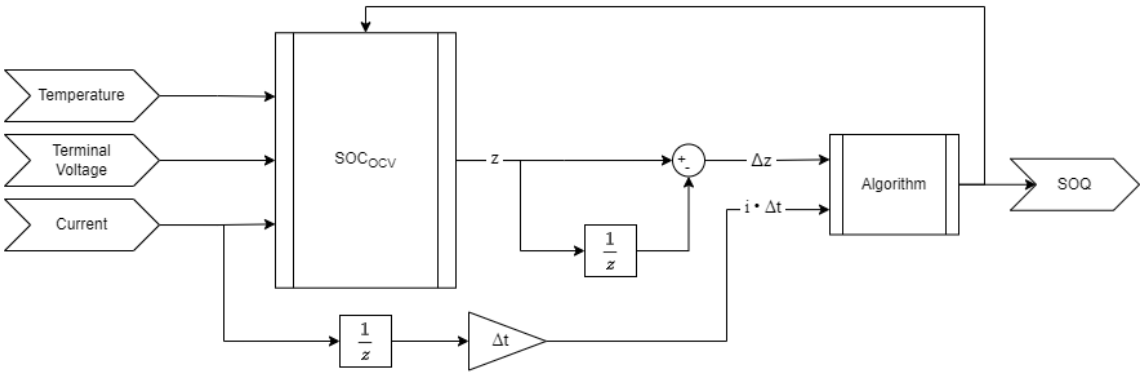


Figure 3.8: Diagram of the SOC and SOQ joint estimation algorithm.

Figure 3.9 illustrates how the SOC_{OCV} block operates. The SOQ together with the current and this time step's SOC, z_k, is inserted into the prediction block. The prediction block utilizes CC to predict the next time step's SOC, z_{k+1} (see Equation 3.21). The predicted SOC, the current, and the temperature is then used in the look-up table to retrieve the parameters R_0, R_1, R_2, C_1, and C_2. The parameters are then inserted into the V_{OC} estimation block where they are scaled according to Subsection 3.1.3. The scaled parameters, current, and the measured terminal voltage, y_k, are used to calculate the OCV, according to Equation 3.10 and Equation 3.11. The latter mentioned equations are implemented with the help of the `MATLAB` function `Sparse`, allowing a faster execution time. Finally, the estimated OCV is converted to SOC through a second lookup-table block.

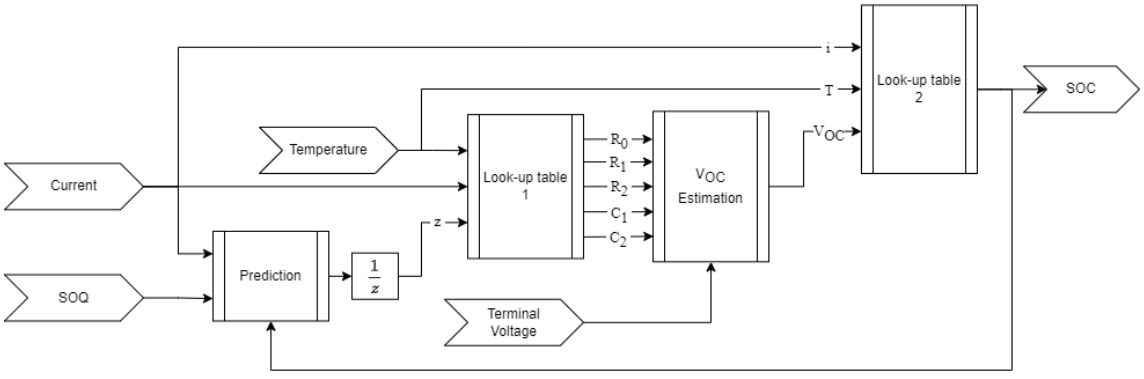


Figure 3.9: Diagram of the SOC_{OCV} algorithm.

3.7 SOQ and OCV joint experiments

The implemented algorithm is a theoretical concept and has no type of filter and is thus not able to handle any type of disturbance. Accordingly, in order to establish if the algorithm performs satisfactorily experiments are carried out without additive noise. There are a total of 9 simulation scenarios listed in Table 3.2 where the Q-scenarios can be seen in Figure 3.7. The algorithm is evaluated on the absolute percentage error of the estimated capacities in the experiments. If the errors are considered satisfactorily we could argue that the theoretical concept of the algorithm can be used in future work.

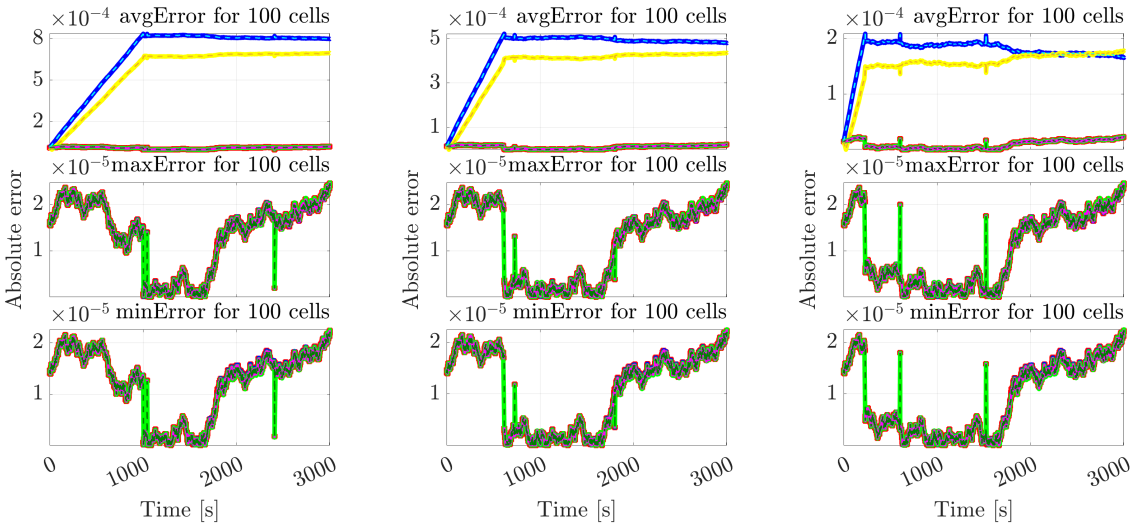
4
Results and discussion

The algorithms created are evaluated through experiments, explained in Section 3.3, Section 3.5, and Section 3.7. This section aims to present and discuss the results of the experiments and is divided into three subsections. One for the SOC estimation algorithms, one for the SOQ estimation algorithms, and one for the joint estimation algorithm.

4.1 SOC

The SOC estimation algorithms' performances are based on their accuracy and robustness test results and their execution time, see Section 3.3. The accuracy and robustness results from all the experiments can be found in Appendix B and Appendix C respectively. These results are then condensed in the summary section, see Subsection 4.1.4.

4.1.1 Accuracy



(a) $z_0 = 0.2$. **(b)** $z_0 = 0.5$. **(c)** $z_0 = 0.8$.

Figure 4.1: Absolute error of the average, maximum, and minimum SOC evaluated by simulation scenarios 1, 4, and 7 listed in Table 3.1, i.e., input index 1, 100 cells, and $z_0 = \{0.2, 0.5, 0.8\}$. The colors correspond to the different algorithms as seen in Figure 4.2.

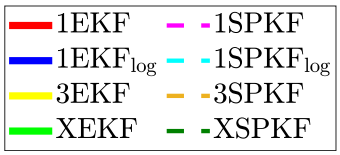
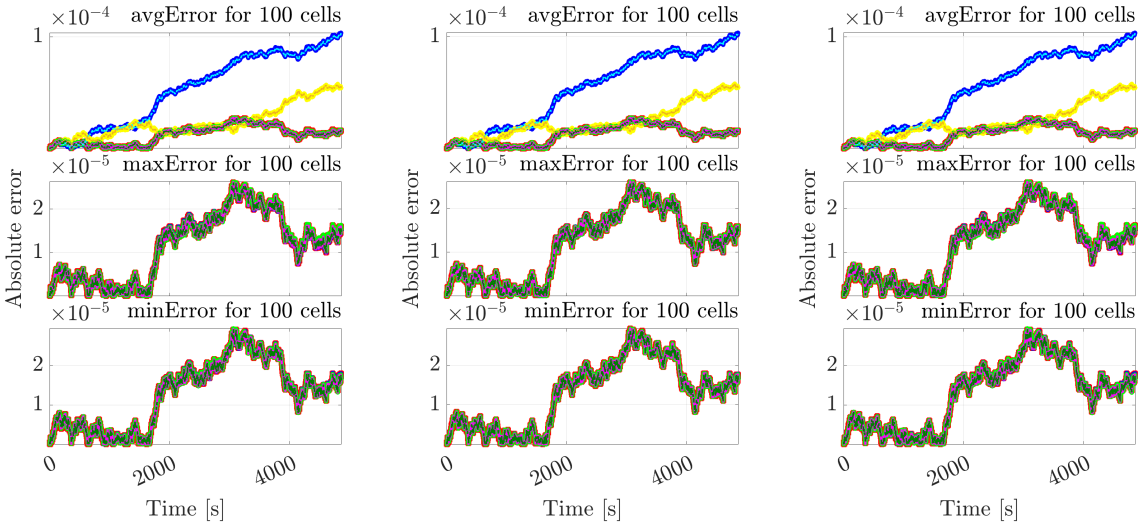


Figure 4.2: Labels used in the accuracy and robustness tests.

By quickly observing the results in Figure 4.1 the initial SOC seems to have quite a big impact on the absolute error. However, this is not the case, it is rather how the initial SOC affects the control input signal. For the scenario with input index 1, seen in Figure 3.4a, the input current will stay constant until SOC reaches 100%, thus if the initial SOC is bigger or smaller the SOC will reach 100% at different points in time.

The same observations can not be made from Figure 4.3, since the initial SOC does not affect the input current signal UDDS, also referred to as input index 2 which can be seen in Figure 3.4b. Thus it can be concluded that the initial SOC does not affect the absolute error of the estimations.



(a) $z_0 = 0.2$. **(b)** $z_0 = 0.5$. **(c)** $z_0 = 0.8$.

Figure 4.3: Absolute error of estimated average, maximum, and minimum SOC evaluated by simulation scenarios 2, 5, and 8 listed in Table 3.1, i.e., input index 2, 100 cells, and $z_0 = \{0.2, 0.5, 0.8\}$. The colors correspond to the different algorithms as seen in Figure 4.2.

Observing the results in Appendix B there is a clear trend of how 1KF$_{log}$'s absolute error of the average SOC is decreasing with the number of cells. Experiment using input index 1 shows this very clearly, as seen in Figure 4.4. From the same figure it is evident that the mentioned trend cannot be found for the absolute error of the minimum and maximum SOC value. This is a result of the distribution of the capacities on which the average SOC estimation algorithm is based. When the number of cells increases, the capacities will be more equally distributed according to the assumed distribution and will hence give a better estimate. The minimum

and maximum value does not use a distribution in their algorithm and thus, do not lead to the same improvement.

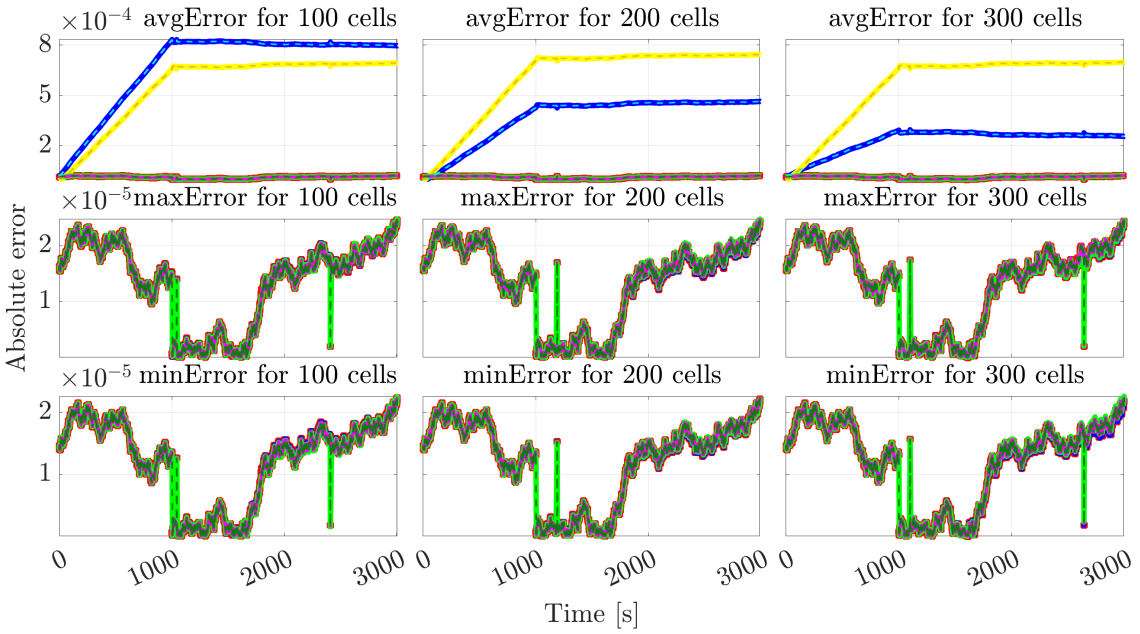


Figure 4.4: Absolute error of the estimated average, maximum, and minimum SOC evaluated by simulation scenarios 1, 10, and 19 listed in Table 3.1, i.e., input index 1 with $z_0 = 0.2$ for 100, 200, and 300 cells. The colors correspond to the different algorithms as seen in Figure 4.2.

The algorithms' absolute error of the minimum SOC is very similar for each scenario, the same can be said for the maximum estimated SOC. 3KF and XKF should in theory have exactly the same absolute error of the minimum and maximum estimates since they are selecting the same cells. 3KF selects the cell with the smallest and largest capacity, which corresponds to the cell with the largest and smallest SOC, respectively. XKF estimates all cells and selects the smallest and largest SOC which is the cells 3KF selected. 1KF and 1KF_{log} should also have the same size in absolute error for the same reason. 1KF calculates all SOC with the variable c and capacity, as seen in Equation 3.21 and Equation 3.22, then selects the smallest and largest SOC which should be equal to calculating the SOC with the variable c, smallest and largest capacity, which is what 1KF_{log} is executing.

An interesting observation is that the absolute error of the minimum SOC seems to have an identical shape as the absolute error of the maximum SOC for all estimation algorithms. By observing the figures closer it can be noticed that the only thing that differs between them is the magnitude of the errors. This is because of the difference in the capacity between the cells. The larger the capacity the less the current noise will affect the estimation, which can be motivated by

$$\Delta z_k = \frac{\Delta t(i_k + \tilde{w}_k)}{Q} = \Delta t \frac{i_k}{Q} + \Delta t \frac{\tilde{w}_k}{Q} \tag{4.1}$$

The SOC estimations are affected by the measurement noise applied to the control input and the terminal voltage. Since all cells have the same current the added noise affects each cell equally. However, the cells have different noises on their terminal voltage. One theory is that the terminal voltage noises are not large enough to affect the SOC estimation. Instead, the estimation error could be a result of the control input noise and thus the absolute error has a similar shape.

The accuracy of the average SOC differs a lot between the different algorithms, unlike the error for the minimum and maximum values. XKF and 1KF have very similar absolute error and has the highest accuracy for input index 1 and 3, found in Appendix A. For the same input indices with 100 cells, 1EKF$_{log}$ has the worst performance followed by 3KF. For 200 and 300 cells 3KF is the least accurate. For input index 2, the UDDS, the algorithms are performing quite similarly with a much smaller absolute error compared to the other input scenarios which can be seen in Figure 4.5. Especially for 200 and 300 cells, XKF, 1KF and 1EKF$_{log}$ perform very similarly, with 1EKF$_{log}$ having a higher accuracy than XKF at times.

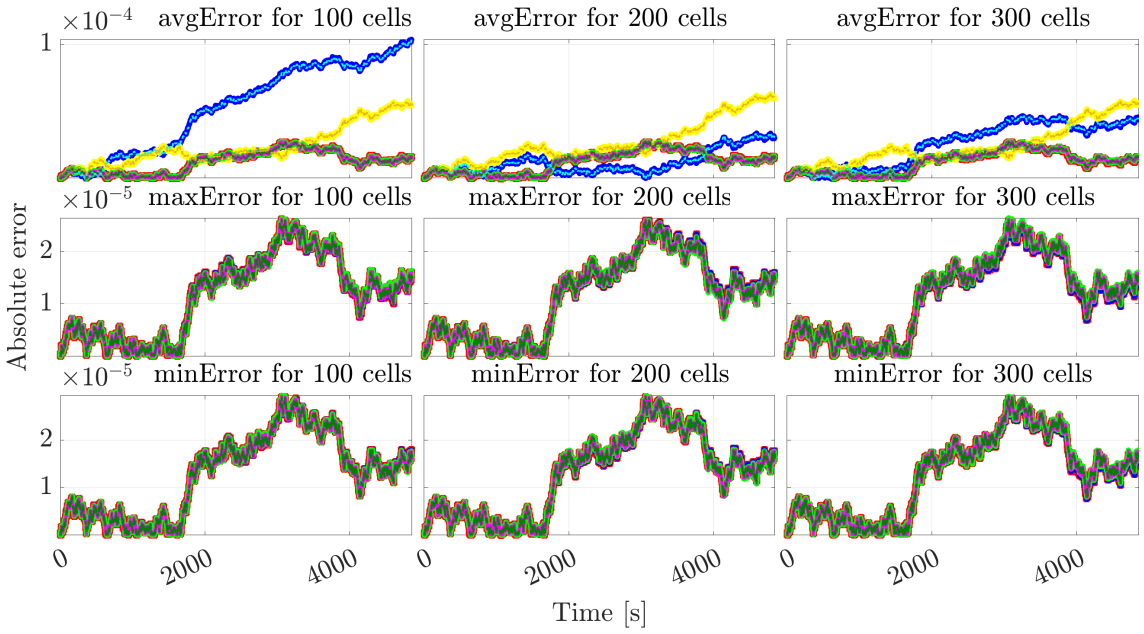


Figure 4.5: Absolute error for average, maximum, and minimum SOC evaluated by simulation scenarios 2, 11, and 20 listed in Table 3.1. Namely input index 2 with $z_0 = 0.2$ for 100, 200, and 300 cells. The colors correspond to the different algorithms as seen in Figure 4.2.

There seems to be no noticeable difference between SPKF's and EKF's accuracy. The main difference between the implementation of the filters is that the EKF linearizes the measurement model while SPKF can utilize the nonlinear model. However, this does not seem to affect the accuracy of the SOC estimation.

4.1.2 Robustness

The robustness tests were performed to catch differences in the performances of SPKF and EKF, the results can be seen in Appendix C. However, when comparing the results from the robustness tests and the accuracy tests there is no noticeable difference, thus concluding that the noise on the temperature does not contribute enough to see a difference in performance between the two KFs. Figure 4.6 shows how the percentual difference between 3EKF and 3SPKF increases when the constant additive temperature disturbance increases. In the robustness tests the temperature has a maximum error of approximately $1.5°C$ as can be seen in Figure 3.3, which according to Figure 4.6 leads to less than a 1% difference between the algorithms. However, as expected the SPKF has a higher robustness compared to EKF but it is only noticeable when having a very large temperature disturbance. An error larger than $5°C$ is considered very large and is not a reasonable error.

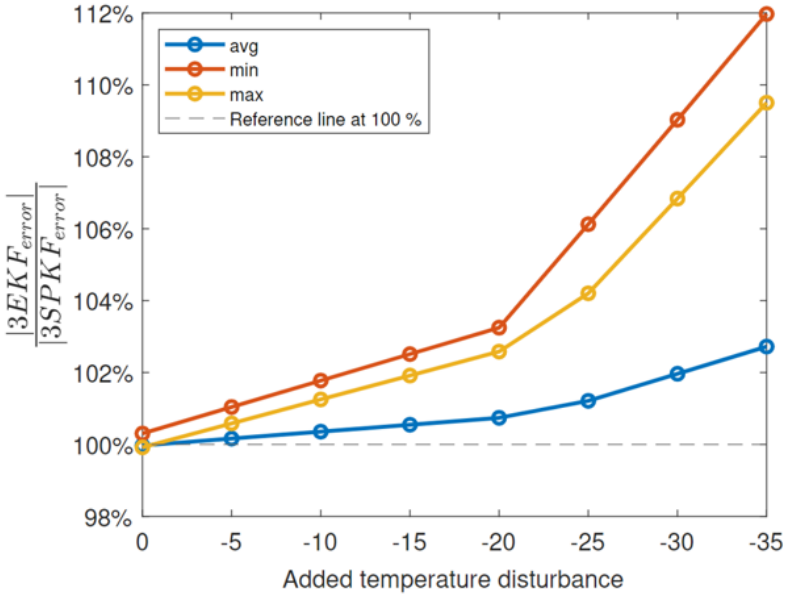


Figure 4.6: Percentage difference between the absolute error of 3EKF and 3SPKF for different additive temperature disturbances. Control input index 2, $z_0 = 0.5$, 100 cells.

4.1.3 Computational complexity

In this section, we present the execution time of the simulations in Table 3.1 and compare computational complexity and its relation to the number of cells. Since the execution time of the simulations varies depending on for example the computer's power and memory it is difficult to follow a trend in computational complexity. This is clearly depicted in Figure 4.7, where 3SPKF has one simulation with a much higher execution time than the rest. The execution time of the robustness tests and accuracy tests should be similar since the added temperature noise does not increase the complexity, this can also be seen in Figure 4.7.

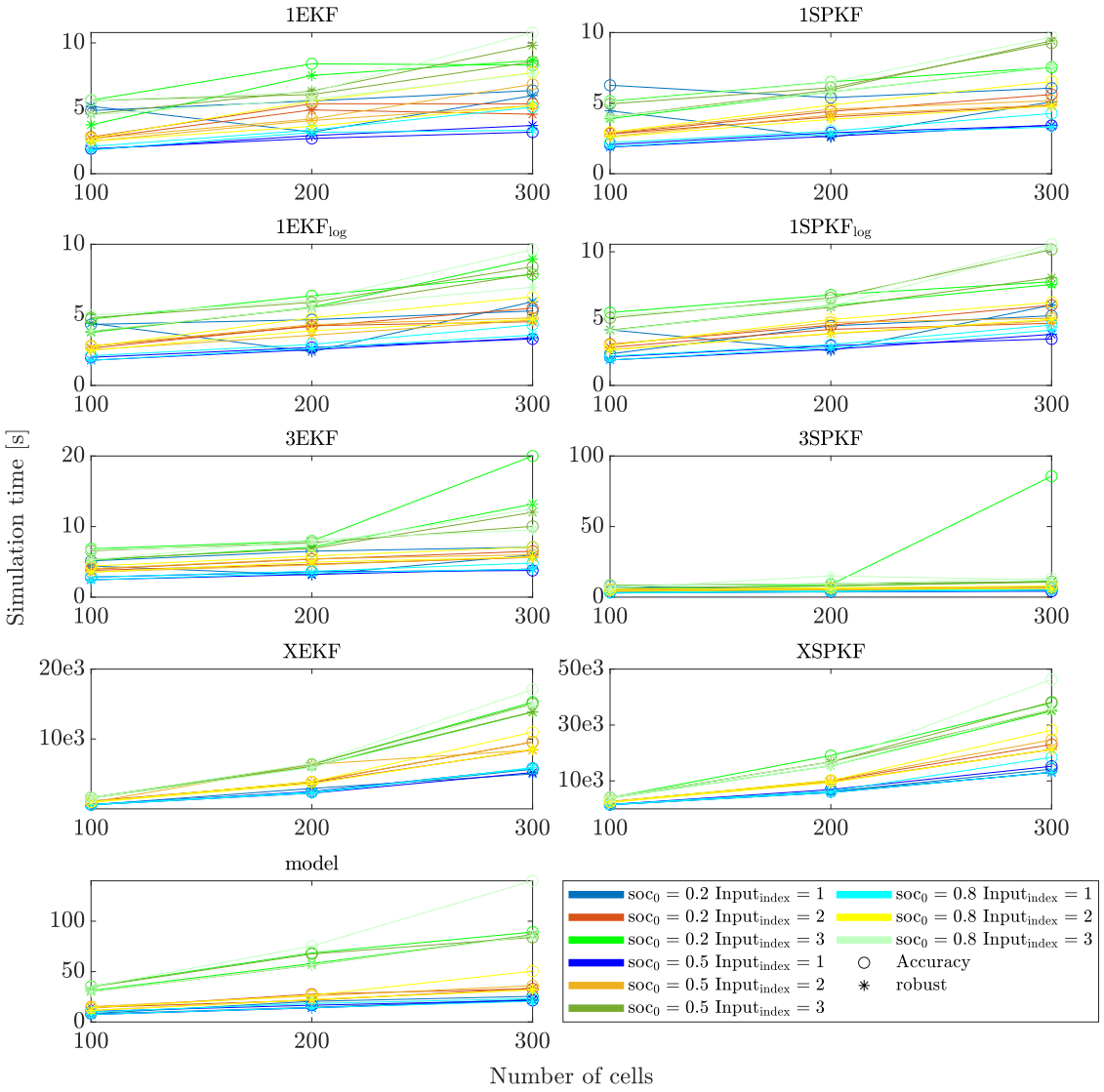
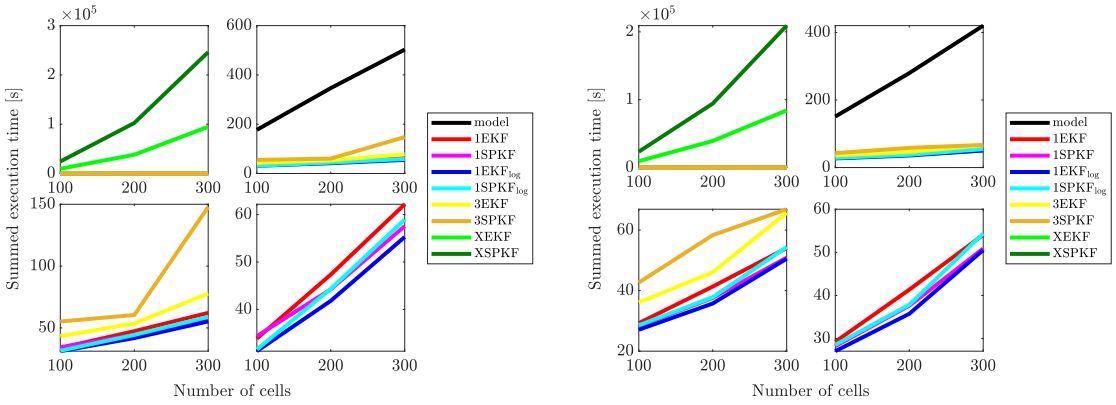


Figure 4.7: The execution time it took to simulate all scenarios for 100, 200, and 300 cells for the model and estimation algorithms. A circle indicates the accuracy test data and a star indicates the robustness test data. The data is color-coordinated depending on the input index. Input indexes 1, 2, and 3 have blue, red, and green colors respectively.

The difference in initial SOC does not affect the execution time since it does not affect the simulated time. The simulated time is different for each input index. Input index 1 has a simulated time of 3000 seconds, input index 2 has around 5000 seconds, and input index 3 has around 8000 seconds. However, what can be concluded is that the execution time increases with the number of KF and the number of cells. To be able to compare the computational time between the algorithms we have generated a figure with the summation of all the execution times for the different numbers of cells, see Figure 4.8.



(a) Accuracy test. **(b)** Robustness test.

Figure 4.8: The summed execution times of simulations of all scenarios for 100, 200, and 300 cells for the model and estimation algorithms. Each figure in (a) and (b) depict the same data, some magnified more than others to distinguish between the algorithms.

In Figure 4.8 it is clear that XEKF and XSPKF have a high growth rate compared to the other algorithms which appear more linear. There is also a trend of SPKFs having longer execution times than EKFs. Here it is also clear that an increase in KFs equals longer execution times. 1KF and 1KF$_{log}$ have the same number of KFs but it seems from Figure 4.8 that 1KF$_{log}$ have a shorter execution time. This is due to that 1KF computes all SOC values while 1KF$_{log}$ only computes a distribution, thus 1KF has more computations. This is small but results in a noticeable difference between the execution times.

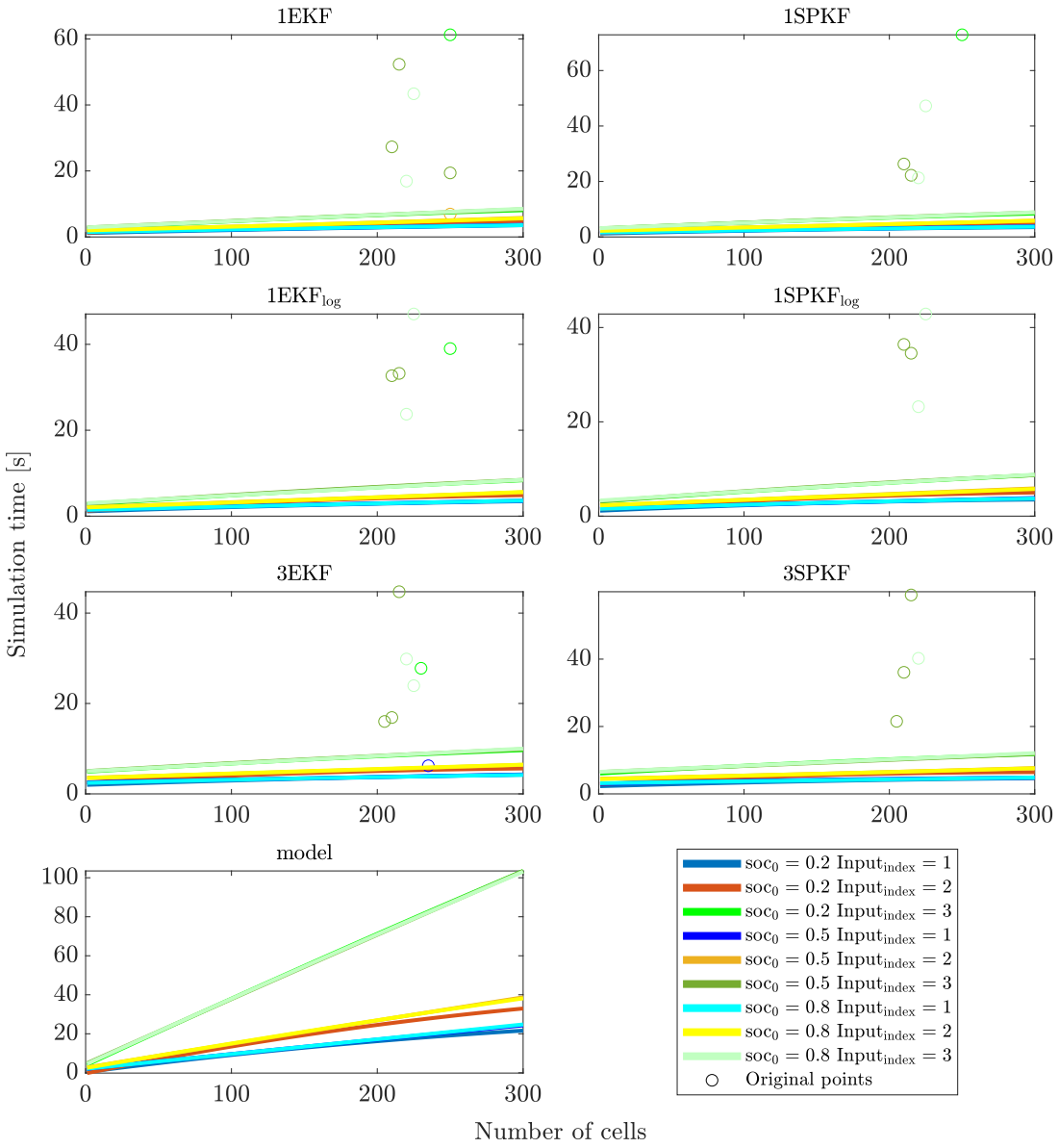


Figure 4.9: The execution times fitted to a second-degree polynomial of all scenarios for cells going from 1 to 300 in steps of 5. A circle indicates an outlier point. The data is color-coordinated depending on the input index. Input indexes 1, 2, and 3 have blue, red and green colors accordingly.

Instead of comparing the execution times from the accuracy and robustness tests, we have executed the experiments where the number of cells ranges from 1 to 300 in steps of 5. This results in a larger data set, which makes our conclusions more reliable. The execution times fitted to a second-degree polynomial can be seen in Figure 4.9. By doing this the data can either appear exponential, linear, or constant, and thus we can draw a conclusion on their complexity. As can be seen in Figure 4.9 are many outliers, Figure 4.10 depicts the results without these outliers.

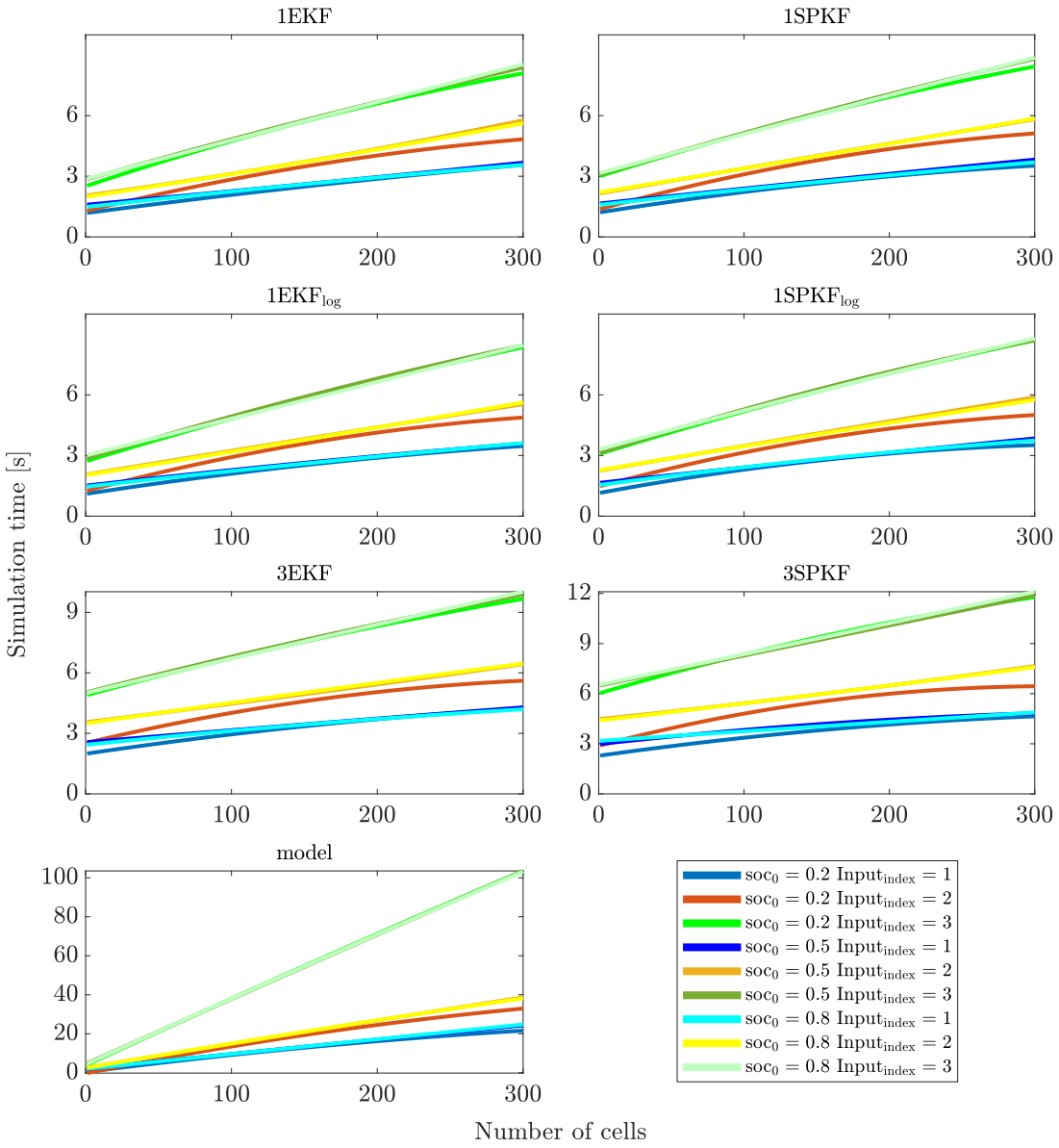


Figure 4.10: The execution times fitted to a second-degree polynomial of all scenarios for cells going from 1 to 300 in steps of 5. Input indices 1, 2, and 3 have blue, red, and green colors respectively.

The polynomial coefficients from Figure 4.10 can be seen in Figure 4.11. As can be seen in the Figure 4.11 the second-order coefficient is smaller when compared to the coefficients of other orders, so much so that all algorithms can be considered to have a linear relationship with the number of cells. It is worth mentioning that even if the second-order coefficient is very small it is at least negative, meaning it does not increase the execution time.

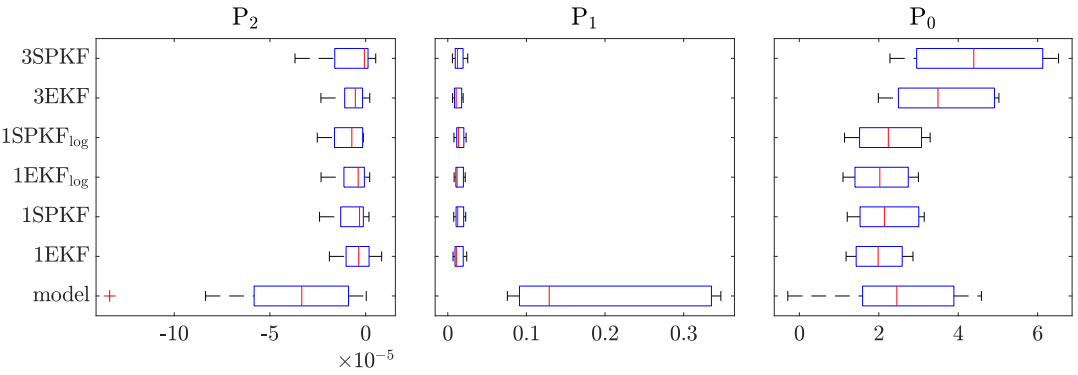
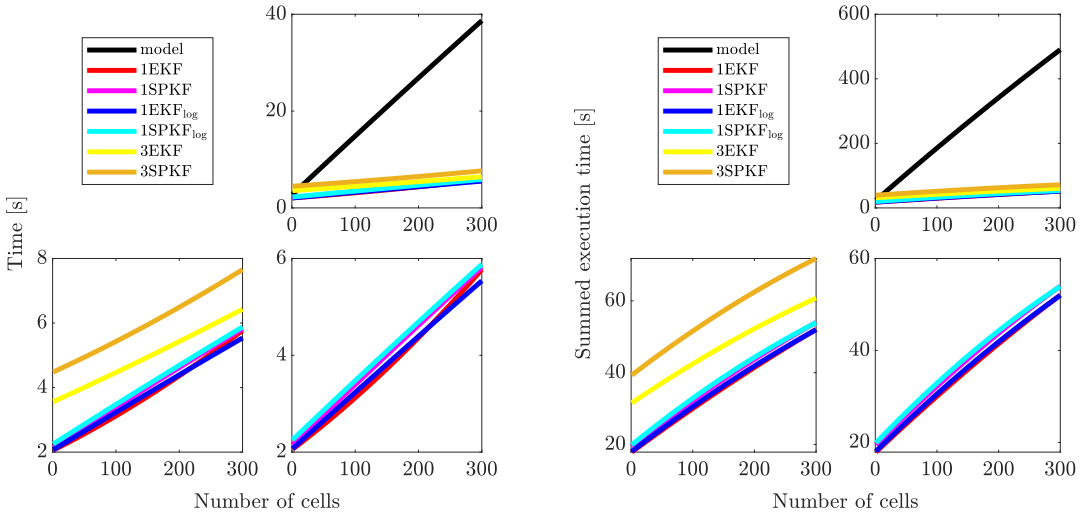


Figure 4.11: The polynomial coefficients for all scenarios of the fitted points in Figure 4.10 shown in a boxplot. The figure to the left shows the coefficient of the second-order, the middle figure shows the coefficients of the first order and the figure to the right shows the constant values.

To be able to compare the computational time between the algorithms, figures have been generated, displaying the summation of all the execution times for the different numbers of cells, see Figure 4.12. Figure 4.12 show similar results as Figure 4.8, being that all algorithms in the latter figure have a strong linear relationship to the number of cells and that the number of KFs is the main contributor to execution time.

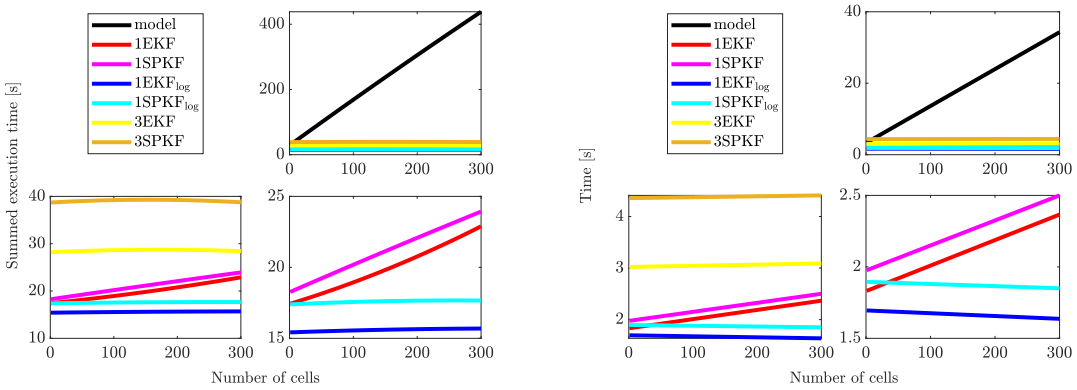


(a) Scenario with input index 2 and $z_0 = 0.5$ fitted to a second-order polynomial.

(b) A summation of all scenarios fitted to a second-order polynomial.

Figure 4.12: Data taken from Figure 4.10 is reshaped to give a better understanding of the algorithms' complexity. The three figures in (a) depict the same data magnified differently to better distinguish between the estimation algorithms. The three figures in (b) depict the summation of all scenarios magnified differently.

As stated in Subsection 3.1.4 the multi-cell model has a total of $9n$ multiplications, thus the number of multiplications in the multi-cell model has a linear relationship to the number of cells. This linearity can also be seen in the figures shown in this subsection. The linearity of the other algorithms cannot be as easily explained. For $1KF_{log}$ the number of multiplications and other operations are constant, unchanged by the number of cells. 1KF calculates the SOC of each cell using the estimated variable c and the capacities resulting in $2n$ multiplications, thus 1KF's execution time should linearly depend on the number of cells. Similarly, 3KF calculates the average terminal voltage and temperature thus there are $2n$ additions and 2 divisions. We had a theory it was the extraction of the input signals causing the unexplained linearity of $1KF_{log}$. This theory was tested by simulating all the scenarios for cells going from 1 to 300 with steps of 5, with the modification of moving the code that extracts the input signals to the multi-cell model's `Simulink` file thus excluding it from the algorithm's `Simulink` file. Also, the part of the 3KF algorithm that calculates the average terminal voltage, temperature, and cell imbalance factors was moved to the cell model's `Simulink` file. The results can be seen in Figure 4.13.



(a) All scenarios are fitted to a second-order polynomial and then summed together.

(b) Scenario with input index 2 and $z_0 = 0.5$ fitted to a first order polynomial.

Figure 4.13: Execution times for the modified model and modified algorithms except for XKF, see Section 3.3, for cells going from 1 to 300 at steps of 5. The modification is the exclusion of the input extraction in the algorithms and the calculation of the average V_t, T, $\mathcal{R}$, $\mathcal{Q}$, and $\mathcal{C}$ moving it instead to the model. All figures in (a) show the same summations magnified differently. The figures in (b) all show the same simulation scenario magnified differently.

Observing Figure 4.13a, where the execution times are fitted to a second-degree polynomial, the 3EKF, 3SPKF, $1SPKF_{log}$, and $1EKF_{log}$ show a clear constant complexity while 1EKF and 1SPKF have a linear complexity with the number of cells. Figure 4.13b is fitted to a first-order polynomial and demonstrates the statement above. This prove that if it had not been for the extraction of inputs, $1SPKF_{log}$ and $1EKF_{log}$ would have constant complexity. If not for the extraction and the

calculation of the average V_t, T, $\mathcal{R}$, $\mathcal{Q}$, and $\mathcal{C}$, 3SPKF and 3EKF would also have constant complexity.

4.1.4 Summary

Since we concluded that z_0 does neither affect the accuracy nor the robustness, and the difference in results of the minimum and maximum SOC are minimal, the average SOC's accuracy and execution time are the values used to distinguish the algorithms. Therefore, we have gathered the execution time and average SOC error in a joint bar chart in Figure 4.14. Even though $1KF_{log}$'s error is decreasing with the number of cells and has at some points higher accuracy than XKF and 1KF the summation across time shows that XKF and 1KF have the highest accuracy and robustness, as can be seen in Figure 4.14. 1KF's and $1KF_{log}$'s execution times are very similar and 3KF's execution time is a few seconds longer. XKF's execution time is not shown on the bar charts but is about 100 times longer than the rest.

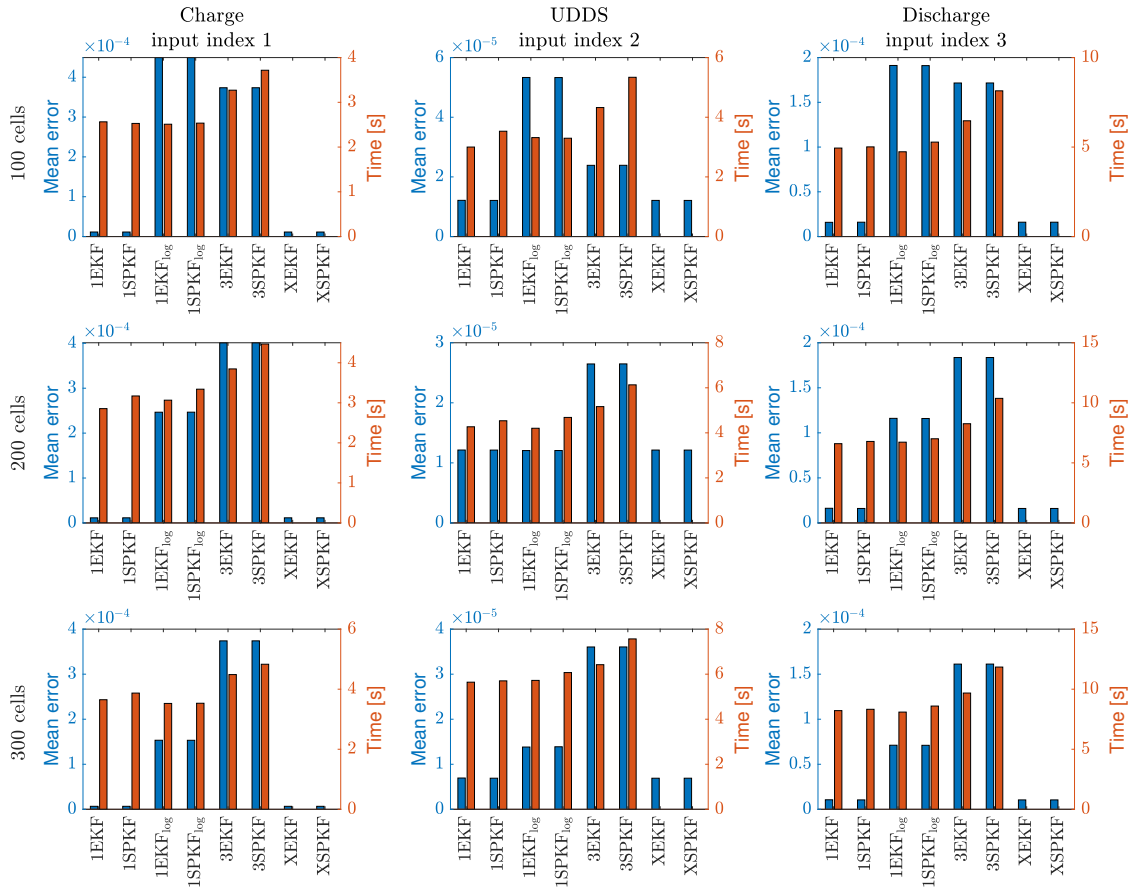


Figure 4.14: The average SOC's mean values of the absolute errors and their execution times across all inputs for $z_0 = 0.5$. XKF is not included because of its extreme execution time compared to the other algorithms. The columns determine the input index and the rows determine the number of cells.

In Figure 4.15 the mean of all errors in Figure 4.14 are shown. From the figure we can conclude that if the extractions of inputs and the average input calculations

were not part of the algorithms $1KF_{log}$ would have a lower execution time than 1KF. This modification, in general, lowers the execution time of all the algorithms, with the main upside being that the execution time of 3KF, $1KF_{log}$ have a constant relation to the number of cells. Without the modification that the execution time of 3KF, $1KF_{log}$ and 1KF have a linear behavior to the number of cells where 3KF has a y-interception point about twice as high as 1KF and $1KF_{log}$. As can be seen in Figure 4.15 the SPKF has a slightly longer execution time compared to the EKF and because of this, has a higher computational complexity.

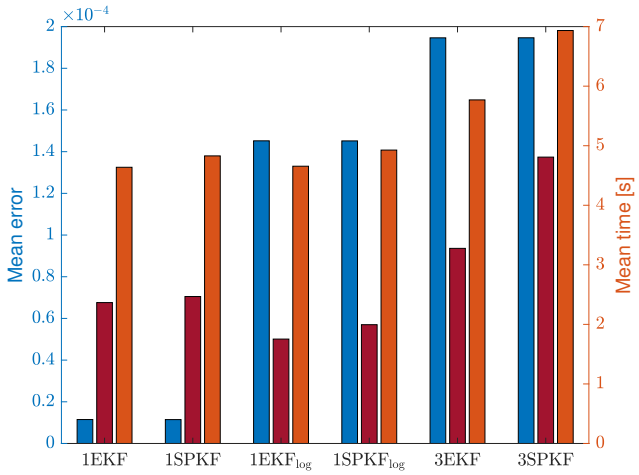


Figure 4.15: The blue bar depicts the calculated mean of all the errors in Figure 4.14. The orange bars depict the average execution time for the average SOC's absolute error when $z_0 = 0.5$, $n = \{100, 200, 300\}$, and $input_{index} = \{1, 2, 3\}$ seen in Figure 4.14. The red bars depict the average execution times calculated identically to the orange bars but for the case where the extraction of inputs and the calculations of the average V_t, T, $\mathcal{R}$, $\mathcal{Q}$, and $\mathcal{C}$ is not in the algorithms but in the model (see Subsection 4.1.3).

4.2 SOQ

The SOQ estimation algorithms, as explained in Section 3.4, are evaluated based on the absolute percentage error of the estimated capacity and execution time of the experiments defined in Section 3.5.

4.2.1 Accuracy

Figure 4.16 displays the average absolute error of the two algorithms scaled with the nominal capacity, while Figure 4.17 displays XRLS's results exclusively. XRLS is able to filter out the noise from all of the cells' SOC retrieved from XEKF while 1RLS is only able to filter out the noise of the randomly selected cell's SOC. This results in an extensive difference in accuracy between the algorithms. This can be seen in Figure 4.16 where it is visible that XRLS has higher accuracy than 1RLS, approximately one hundred times as accurate.

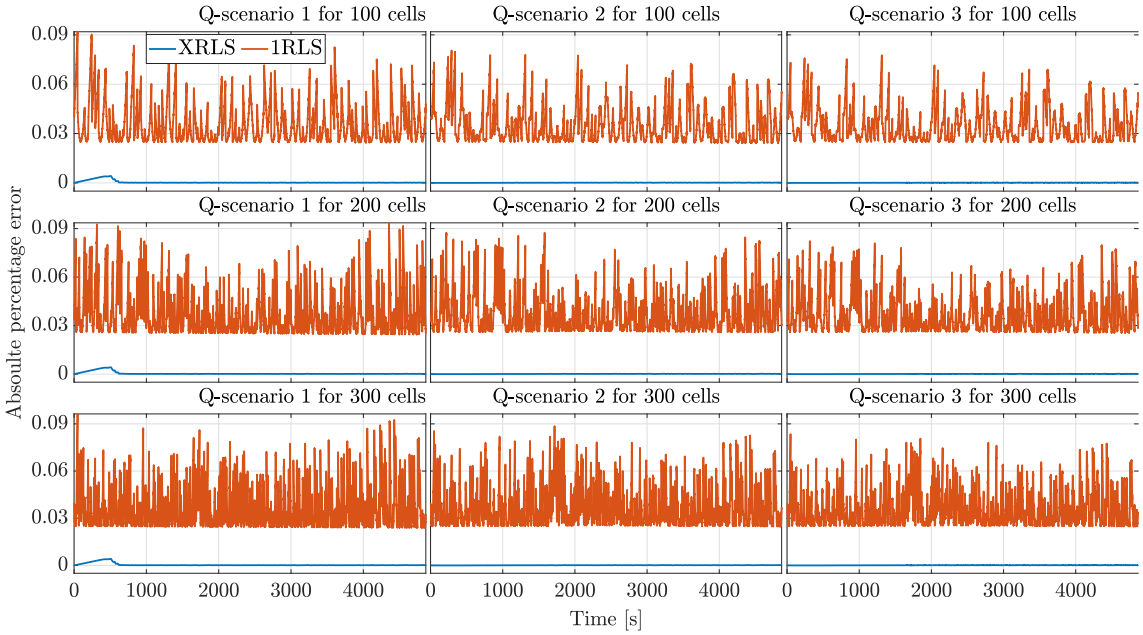


Figure 4.16: Average absolute error for XRLS and 1RLS, scaled with the nominal capacity. The 9 scenarios are listed and explained in Table 3.2.

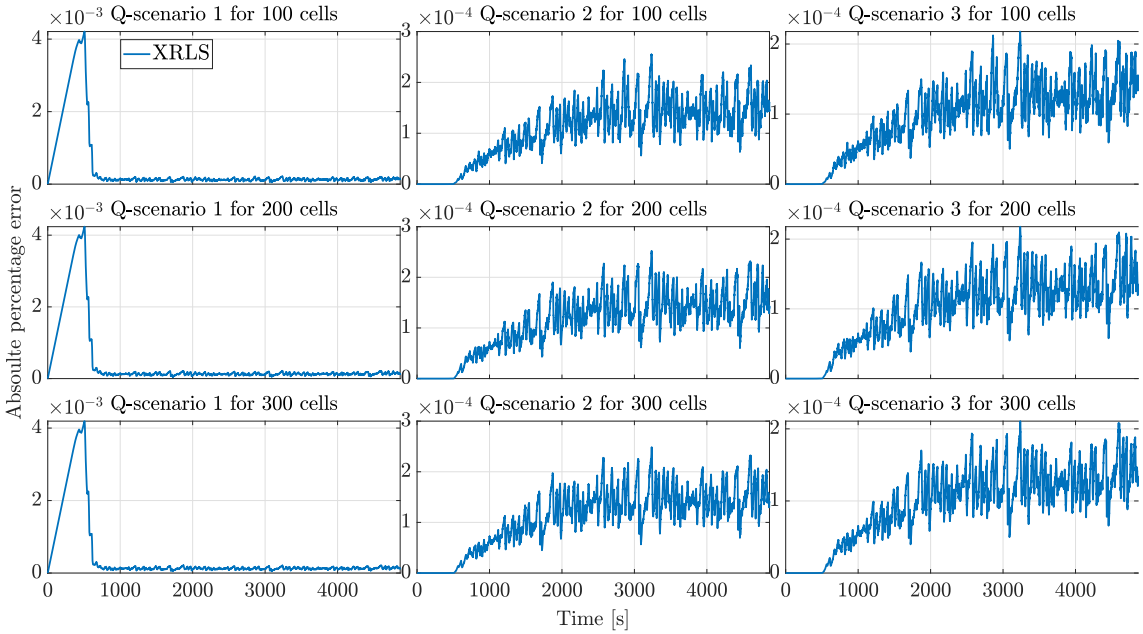


Figure 4.17: Average absolute error for XRLS scaled with the nominal capacity. The 9 scenarios are listed and explained in Table 3.2.

As explained in Section 3.5 the capacity in the experiments decreases abnormally quickly when in fact, a battery cell's capacity would not decrease as rapidly. It is therefore reasonable to assume that the absolute errors will be magnified. However, it is also possible that the algorithms would perform significantly differently with a more realistic capacity scenario. Thus, the results seen in Figure 4.16 are not

adequate to allow a conclusion on their performance. Instead, the results can be used to temporarily assume which algorithm is most accurate.

4.2.2 Computational complexity

As explained in Section 2.6 computational complexity is challenging to measure. Therefore we evaluate the algorithms according to their execution time. Figure 4.18 displays XRLS', 1RLS', and the model's execution time of the simulation scenarios listed in Table 3.2. XRLS', 1RLS', and the model's execution times are fitted to a second-order polynomial. By doing this the data can either appear exponential, linear, or constant, and thus we can draw a conclusion on their complexity. In Figure 4.18 XRLS appears to have a linear relationship to the number of cells while 1RLS appears to be more constant.

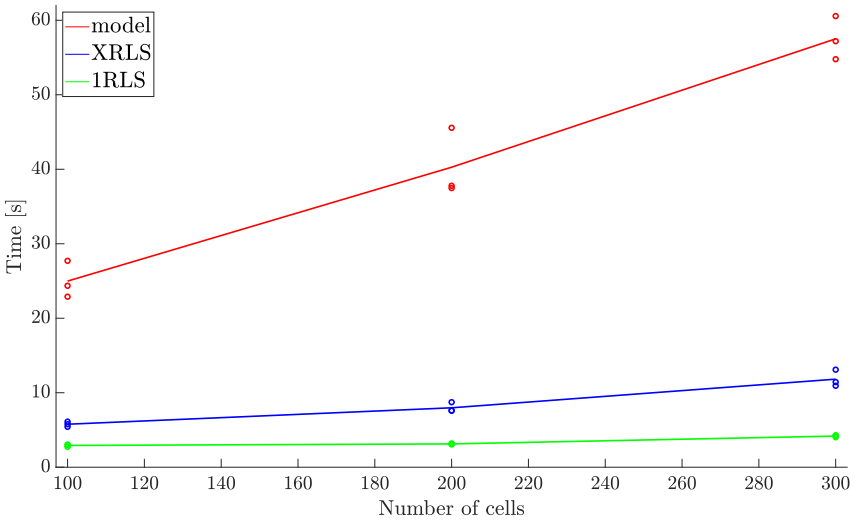


Figure 4.18: Execution time for the model, XRLS, and 1RLS of the experiments listed in Table 3.2. The circles are points for each respective scenario and the graphs are curves fitted to represent how the execution time varies depending on the number of cells.

The same conclusions can be drawn by observing their polynomial coefficients in

$$\begin{aligned} T_{model} &\approx 10^{-5} \cdot 9.7479n^2 + 0.1236n + 11.651 \\ T_{XRLS} &\approx 10^{-5} \cdot 8.1929n^2 - 0.0025n + 5.2035 \\ T_{1RLS} &\approx 10^{-5} \cdot 4.2243n^2 - 0.0106n + 3.5571 \end{aligned} \tag{4.2}$$

The algorithms' second-order polynomial coefficients are very small compared to the lower-order coefficients and can thus be seen as zero. This implies that 1RLS and XRLS can be assumed to have a linear computational complexity for n of interest.

In Equation 4.2 and in Figure 4.18 1RLS can be seen to have an overall lower execution time, not increasing as quickly with the number of cells as XRLS does.

Their overall execution time is similar to 3SPKF, where XRLS is slightly slower. With the same reasoning as 1KF, 1RLS should have a constant execution time since the number of filters does not increase. However, due to the computation of each cell's SOQ, seen in Equation 3.38, the execution time will be slightly linearly dependent on the number of cells. Since XRLS's number of RLS filters is linearly increasing with the number of cells, so should its execution time.

The filter length p is identical for both algorithms as mentioned earlier in Section 3.5. Increasing the filter length results in slower estimation algorithms since more computations are needed, as explained in Section 2.6. Since the 1RLS algorithm only has 1 filter, an increase in the filter length would not have as great of an impact as for XRLS, which has n number of filters. With this taken into account, the difference in computational complexity between the filters would rapidly increase as the filter length increases.

4.2.3 Summary

The experiments indicate that the XRLS algorithm is more accurate but requires a significantly longer time. The 1RLS algorithm executes at an almost constant rate. It is however unable to successfully filter out the input noise.

4.3 SOC and SOQ joint estimation

The absolute percentage error of the estimated capacities when executing the experiments described in Section 3.7 are presented in Figure 4.19.

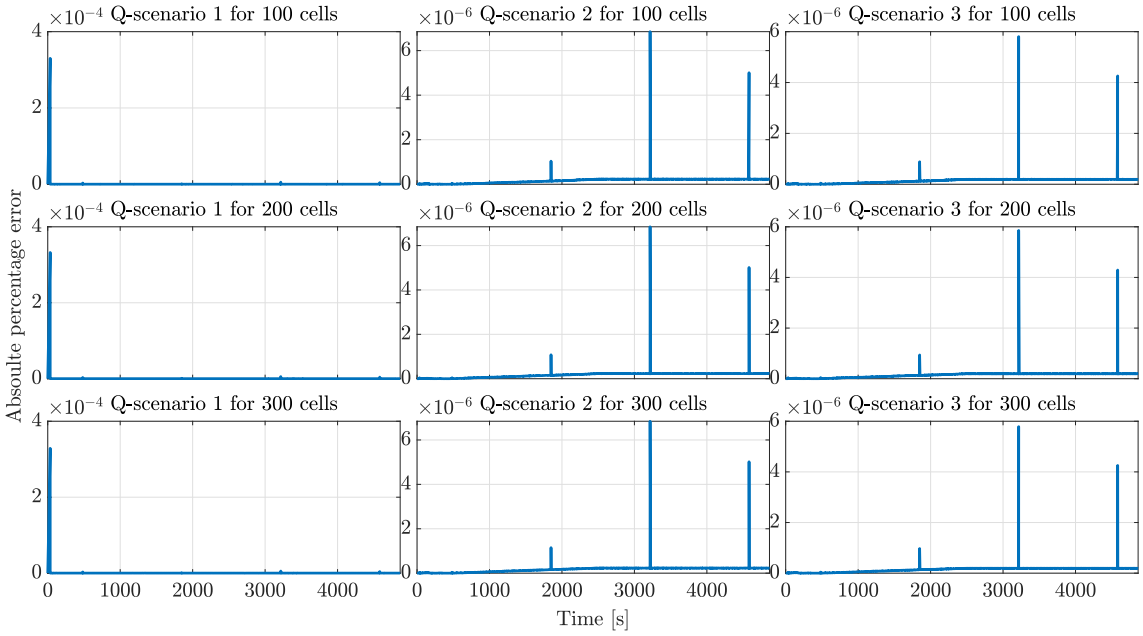


Figure 4.19: Absolute error between the estimated capacity and the model capacity averaged over the number of cells and scaled with the nominal capacity. Each graph represents a different experiment where the number of cells and the capacity differ.

Observing Figure 4.19 the absolute percentage error appears relatively small and constant except for sudden spikes. These spikes originates from the current or the difference in SOC approaching zero. As described Section 3.6 a safety margin is implemented to avoid faulty estimation values. If the current or difference in SOC is smaller than a set tolerance the safety function ensures that no update of the capacity is performed and instead outputs the latest capacity from memory. If no update is performed and the capacity is changing this will result in a spike in absolute error, as seen in Figure 4.19. If a more realistic decrease in the capacity was implemented the rate of change would be lowered, and thus the sudden spikes seen in Figure 4.19 would be reduced.

5
Conclusion and future work

5.1 Conclusion

The main aim of this thesis was to design a SOC estimation algorithm for a multi-cell system with lower computational complexity than a SOC algorithm estimating each cell individually. The algorithm's big $\mathcal{O}$ notation cannot be concluded due to `MATLAB` not being an open source. However, XKF has most likely a big $\mathcal{O}$ notation between n^2 and n^3. 1EKF had one of the lowest computational complexities and had as good accuracy and robustness as the XKF, which estimates each cell individually. 1EKF uses an EKF to estimate SOC for one cell, then using CC calculates the variable c which is used together with the capacity to determine the remaining SOC. The results of the experiments conducted show that this method is highly reliable and has an execution time that slowly increases linearly with the number of cells. If the number of cells connected in series exceeds far over 300, one could argue that the algorithm we call 1EKF_{log} would be more suitable. 1KF_{log} and 3KF have an execution time that slowly increases linearly with the number of cells. However, 1KF_{log} should have a constant execution time if not for the way `MATLAB` extracts the cells' voltage and temperature. An obvious finding emerging from the experiments was that SPKF has a higher execution time than EKF. This result will contribute to the sources finding that SPKF has a higher complexity than EKF.

The SOC algorithms had a high dependence on the capacities' accuracy, which resulted in a second aim of this thesis, namely investigating how SOQ could be estimated together with SOC. XRLS gives an accurate and robust SOQ estimation with an execution time increasing linearly with the number of cells. However, the estimated value cannot be inserted into the SOC algorithm because of the looping problem, explained in Section 3.4. Instead, another approach was used to avoid this problem. The proposed method works very well and shows promising results in the experiments. Despite the good results, the experiments were conducted with noise being absent. Thus no real conclusion about the method's performance can be drawn. Instead, we can only conclude it is a promising approach for solving the problem.

5.2 Suggestions for further research

Future studies should include testing on real batteries to verify the suggested cell model and the SOC estimating algorithms. This would require the implementation

to follow a certain toolchain such that it can be executed on the real system. Real-life testing could also be used to evaluate the robustness of the algorithms since the assumed parameters are prone to errors.

Since the SOC algorithms are highly dependent on the capacities' accuracy it would be interesting to include an additive noise on the capacity to further test the robustness of the SOC estimation algorithms.

The Q-scenarios used in the experiments that are explained in Section 3.5 and Section 3.7 have a very rapidly decreasing capacity that is not very realistic. The SOQ estimation algorithms need to be further evaluated on more realistic Q-scenarios, where the capacity decreases slowly over longer periods, representing several driving cycles.

To further advance with the SOQ estimation algorithm in Section 3.6 filters that can handle the disturbances on the measured terminal voltage and current need to be implemented. The estimation algorithm then needs to be evaluated with additive noise unlike the experiments explained in Section 3.7.

Bibliography

[1] I. E. Agency, "Global EV Outlook 2021." Available: https://iea.blob.core.windows.net/assets/ed5f4484-f556-4110-8c5c-4ede8bcba637/GlobalEVOutlook2021.pdf.

[2] Directorate-General for Climate Action, "Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions, stepping up europe's 2030 climate ambition investing in a climate-neutral future for the benefit of our people." COM/2020/562 final, Nov. 2020 [Online].

[3] V. Cars, "Volvo cars to be fully electric by 2030." Mar. 02, 2021. [Online] Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/277409/volvo-cars-to-be-fully-electric-by-2030.

[4] I. E. Agency, "Global EV Outlook 2022." Available: https://iea.blob.core.windows.net/assets/ad8fb04c-4f75-42fc-973a-6e54c8a4449a/GlobalElectricVehicleOutlook2022.pdf.

[5] I. E. Agency, "Global Supply Chains of EV Batteries." Available: https://iea.blob.core.windows.net/assets/4eb8c252-76b1-4710-8f5e-867e751c8dda/GlobalSupplyChainsofEVBatteries.pdf.

[6] A. Fotouhi, D. J. Auger, K. Propp, S. Longo, and M. Wild, "A review on electric vehicle battery modelling: From Lithium-ion toward Lithium–Sulphur," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 1008–1021, 2016.

[7] J. Van Mierlo, M. Berecibar, M. El Baghdadi, C. De Cauwer, M. Messagie, T. Coosemans, V. A. Jacobs, and O. Hegazy, "Beyond the State of the Art of Electric Vehicles: A Fact-Based Paper of the Current and Prospective Electric Vehicle Technologies," *World Electric Vehicle Journal*, vol. 12, no. 1, 2021.

[8] H. Rahimi-Eichi, U. Ojha, F. Baronti, and M.-Y. Chow, "Battery Management System: An Overview of Its Application in the Smart Grid and Electric Vehicles," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 4–16, 2013.

[9] J. C. M. de Souza Aranha and M. Giesbrecht, "Multi-cell SOC estimation for Li-Ion battery applied to an energy storage system," in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, pp. 1051–1056, 2020.

[10] Z. J. Zhang and P. Ramadass, *Batteries for Sustainability: Selected Entries from the Encyclopedia of Sustainability Science and Technology*, ch. Lithium-Ion Battery Systems and Technology, pp. 319–357. New York, NY: Springer New York, 2013.

[11] M. F. Samadi and M. Saif, "Integrated Battery Management System," *Intefrated Systems: Innovations and Applications*, pp. 173–194, 01 2015.

[12] M. Lelie, T. Braun, M. Knips, H. Nordmann, F. Ringbeck, H. Zappen, and D. U. Sauer, "Battery Management System Hardware Concepts: An Overview," *Applied Sciences*, vol. 8, no. 4, 2018.

[13] T. Xiao, X. Shi, B. Zhou, and X. Wang, "Comparative Study of EKF and UKF for SOC Estimation of Lithium-ion Batteries," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, pp. 1570–1575, 2019.

[14] G. Plett, *Battery Management Systems, Volume II: Equivalent-Circuit Methods*. Artech, 2015.

[15] P. Shen, M. Ouyang, L. Lu, J. Li, and X. Feng, "The Co-estimation of State of Charge, State of Health, and State of Function for Lithium-Ion Batteries in Electric Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 92–103, 2018.

[16] Y. Zou, X. Hu, H. Ma, and S. E. Li, "Combined State of Charge and State of Health estimation over lithium-ion battery cell cycle lifespan for electric vehicles," *Journal of Power Sources*, vol. 273, pp. 793–803, 2015.

[17] G. L. Plett, "Efficient battery pack state estimation using bar-delta filtering," International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium & Exhibition, (Stavanger, Norway), 2009.

[18] D. U. Sauer, G. Bopp, A. Jossen, J. Garche, M. Rothert, and M. Wollny, "State of charge - what do we really speak about ?," International Telecommunications Energy Conference, (Copenhagen, Denmark), 1999.

[19] S. Piller, M. Perrin, and A. Jossen, "Methods for state-of-charge determination and their applications," *Journal of Power Sources*, vol. 96, pp. 113–120, 06 2001.

[20] A. Kirchev, "Chapter 20 - Battery Management and Battery Diagnostics," in *Electrochemical Energy Storage for Renewable Sources and Grid Balancing* (P. T. Moseley and J. Garche, eds.), pp. 411–435, Amsterdam: Elsevier, 2015.

[21] M. S. Hosen, A. Pirooz, T. Kalogiannis, J. He, J. Van Mierlo, and M. Berecibar, "A Strategic Pathway from Cell to Pack-Level Battery Lifetime Model Development," *Applied Sciences*, vol. 12, no. 9, 2022.

[22] M. S. Hosen, D. Karimi, T. Kalogiannis, A. Pirooz, J. Jaguemont, M. Berecibar, and J. Van Mierlo, "Electro-aging model development of nickel-manganese-cobalt lithium-ion technology validated with light and heavy-duty real-life profiles," *Journal of Energy Storage*, vol. 28, p. 101265, 2020.

[23] Y. Wang, J. Tian, Z. Sun, L. Wang, R. Xu, M. Li, and Z. Chen, "A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems," *Renewable and Sustainable Energy Reviews*, vol. 131, p. 110015, 2020.

[24] L. Zhang, H. Peng, Z. Ning, Z. Mu, and C. Sun, "Comparative Research on RC Equivalent Circuit Models for Lithium-Ion Batteries of Electric Vehicles," *Applied Sciences*, vol. 7, no. 10, 2017.

[25] M. Urbano, *Introductory Electrical Engineering with Math Explained in Accessible Language*, ch. 19 - Kirchhoff's Laws, pp. 197–213. John Wiley & Sons, Ltd, 2019.

[26] M. Mahadi, T. Ballal, M. Moinuddin, and U. M. Al-Saggaf, "A Recursive Least-Squares with a Time-Varying Regularization Parameter," *Applied Sciences*, vol. 12, no. 4, 2022.

[27] A. L. Bruce, A. Goel, and D. S. Bernstein, "Convergence and consistency of recursive least squares with variable-rate forgetting," *Automatica*, vol. 119, p. 109052, 2020.

[28] S. Haykin, *Statistical Digital Signal Processing and Modeling*, ch. 9.4 - Recursive Least Squares. John Wiley And Sons Ltd, 1996.

[29] P. Gong, "Adaptive estimation of battery state of charge," Master's thesis, Chalmers University of Technology, Department of Electrical Engineering, 2017.

[30] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[31] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[32] I. Arasaratnam and S. Haykin, "Cubature Kalman Filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[33] R. Merwe, E. Wan, and S. Julier, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion–applications to integrated navigation," *Proceedings of the AIAA Guidance, Navigation & Control Conference*, vol. 3, Aug. 2004.

[34] G. Lindfield and J. Penny, "Chapter 2 - Linear Equations and Eigensystems," in *Numerical Methods (Fourth Edition)* (G. Lindfield and J. Penny, eds.), pp. 73–156, Academic Press, fourth edition ed., 2019.

[35] R. A. Horn and C. R. Johnson, *Matrix analysis*, ch. 7 - Positive Definite and Semidefinite Matrices. Cambridge University Press, 2017.

[36] Y. Li, J. Chen, and F. Lan, "Enhanced online model identification and state of charge estimation for lithium-ion battery under noise corrupted measurements by bias compensation recursive least squares," *Journal of Power Sources*, vol. 456, p. 227984, 2020.

[37] Britannica, The Editors of Encyclopedia, "Computational complexity." https://www.britannica.com/topic/computational-complexity. Accessed: Apr. 19, 2023 [Online]. Available: https://www.britannica.com/topic/computational-complexity.

[38] Wikipedia, "Computational Complexity." https://en.wikipedia.org/wiki/Computational_complexity. Accessed: Apr. 19, 2023 [Online]. Available: https://en.wikipedia.org/wiki/Computational_complexity.

[39] Measure the Performance of Your Code, *MathWorks*. Accessed: Apr. 20, 2023 [Online]. Available: https://se.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html.

[40] F. E. Daum, *Encyclopedia of Systems and Control*, ch. Extended Kalman Filters, pp. 411–413. London: Springer London, 2015.
[41] S. Lu, L. Cai, L. Ding, and J. Chen, "Two Efficient Implementation Forms of Unscented Kalman Filter," in *2007 IEEE International Conference on Control and Automation*, pp. 761–764, 2007.
[42] M. Raitoharju and R. Piché, "On Computational Complexity Reduction Methods for Kalman Filter Extensions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 10, pp. 2–19, 2019.
[43] A. Khalid, S. A. R. Kashif, N. Ain, M. Awais, M. Smieee, J. Carreño, J. Vasquez, J. Guerrero, and B. Khan, "Comparison of Kalman Filters for State Estimation Based on Computational Complexity of Li-Ion Cells," *Energies*, vol. 16, p. 2710, 03 2023.
[44] G. L. Plett, "Sigma-point Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 2: Simultaneous state and parameter estimation," *Journal of Power Sources*, vol. 161, no. 2, pp. 1369–1384, 2006.
[45] I.-D. Fîciu, C.-L. Stanciu, C. Anghel, and C. Elisei-Iliescu, "Low-Complexity Recursive Least-Squares Adaptive Algorithm Based on Tensorial Forms," *Applied Sciences*, vol. 11, no. 18, 2021.
[46] S. Winograd, *Arithmetic Complexity of Computations*. Society for Industrial and Applied Mathematics, 1980.
[47] X. Liu, X. Deng, Y. He, X. Zheng, and G. Zeng, "A Dynamic State-of-Charge Estimation Method for Electric Vehicle Lithium-Ion Batteries," *Energies*, vol. 13, no. 1, 2020.
[48] S. R. Hashemi, A. M. Mahajan, and S. Farhad, "Online estimation of battery model parameters and state of health in electric and hybrid aircraft application," *Energy*, vol. 229, p. 120699, 2021.
[49] A. Pirooz, Y. Firouz, J. Van Mierlo, and M. Berecibar, "Voltage Vector Redundancy Exploitation for Battery Balancing in Three-Phase CHB-Based Modular Energy Storage Systems," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 9, pp. 9364–9375, 2022.
[50] Y. Wang, C. Liu, R. Pan, and Z. Chen, "Experimental data of lithium-ion battery and ultracapacitor under DST and UDDS profiles at room temperature," *Data in Brief*, vol. 12, pp. 161–163, 2017.
[51] T. Baumhöfer, M. Brühl, S. Rothgang, and D. U. Sauer, "Production caused variation in capacity aging trend and correlation to initial cell performance," *Journal of Power Sources*, vol. 247, pp. 332–338, 2014.
[52] A. Carnovale and X. Li, "A modeling and experimental study of capacity fade for lithium-ion batteries," *Energy and AI*, vol. 2, p. 100032, 2020.
[53] W. Bao, H. Liu, Y. Sun, and Y. Zheng, "A Fast Prediction of Open-Circuit Voltage and a Capacity Estimation Method of a Lithium-Ion Battery Based on a BP Neural Network," *Batteries*, vol. 8, no. 12, 2022.

A

Input Scenarios

This appendix consists of 9 figures, each displaying the model's input current, measured input current, model's terminal voltage, measured terminal voltage, and the minimum, maximum, and average SOC retrieved from the battery model with 100 cells. Each figure is slightly different, each displaying the mentioned values for different combinations of the input index $\in \{1, 2, 3\}$ and the initial SOC $\in \{0.2, 0.5, 0.8\}$.

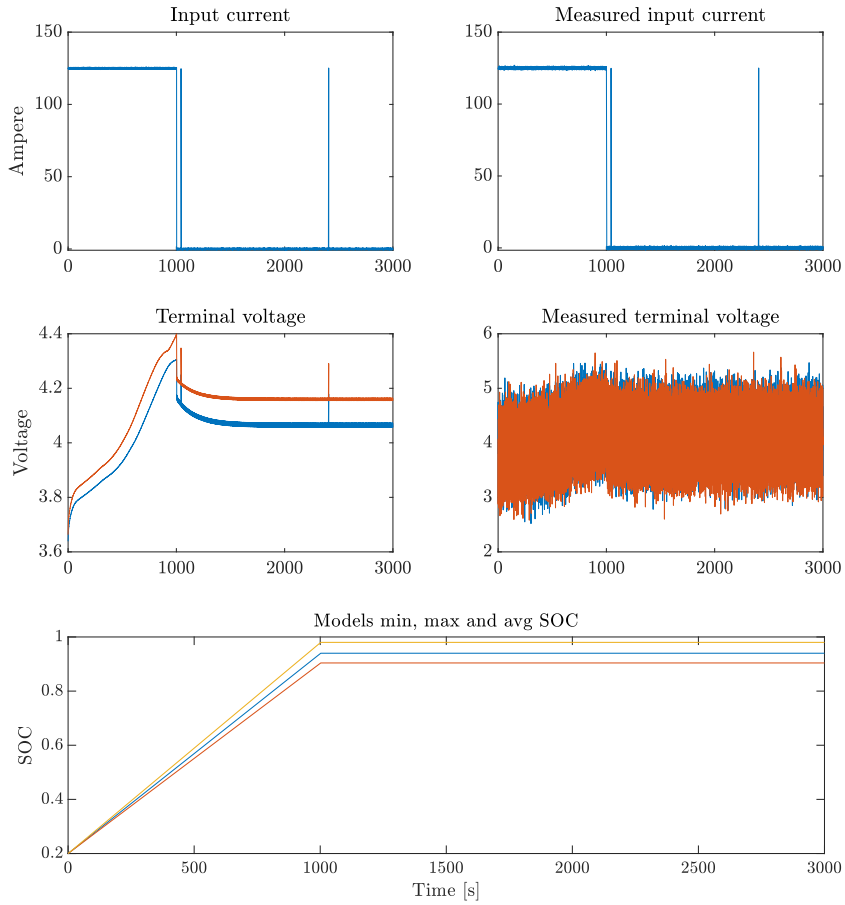


Figure A.1: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 1 listed in Table 3.1. Namely input index 1, $z_0 = 0.2$, 100 cells.

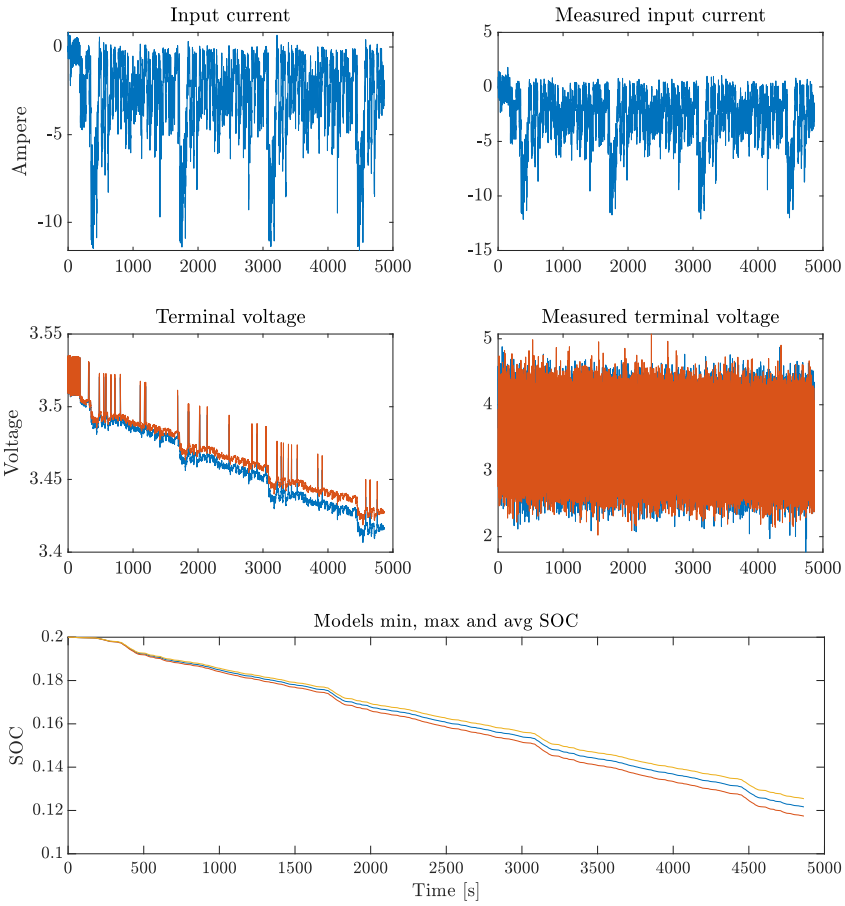


Figure A.2: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 2 listed in Table 3.1. Namely input index 2, $z_0 = 0.2$, 100 cells.

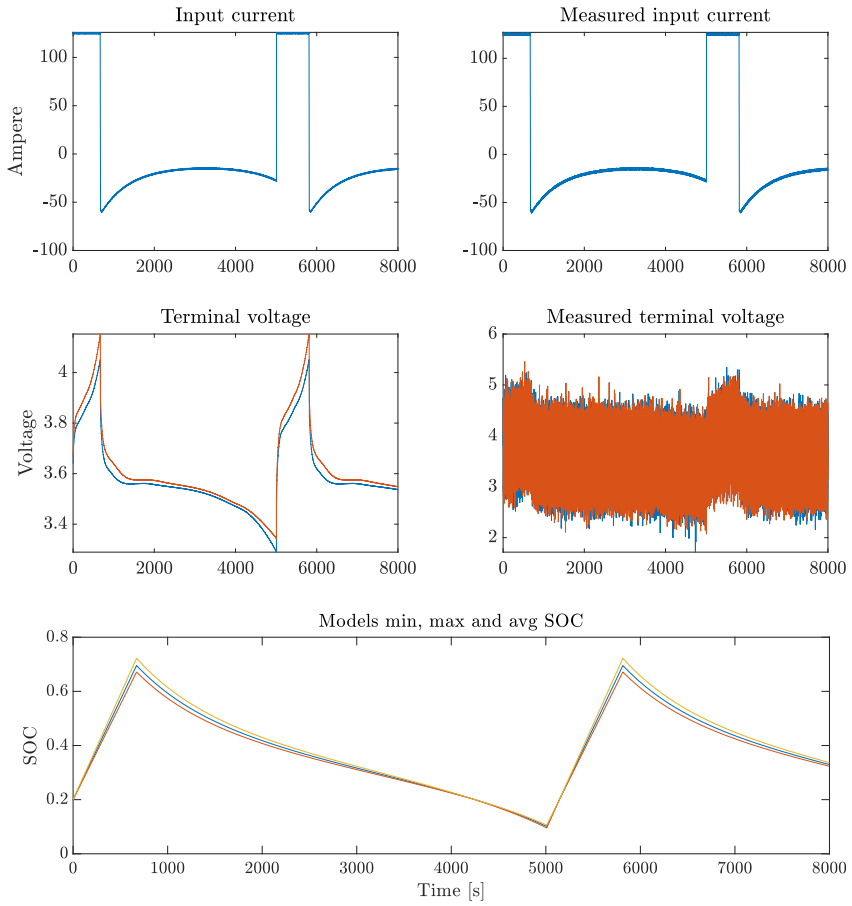


Figure A.3: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 3 listed in Table 3.1. Namely input index 3, $z_0 = 0.2$, 100 cells.

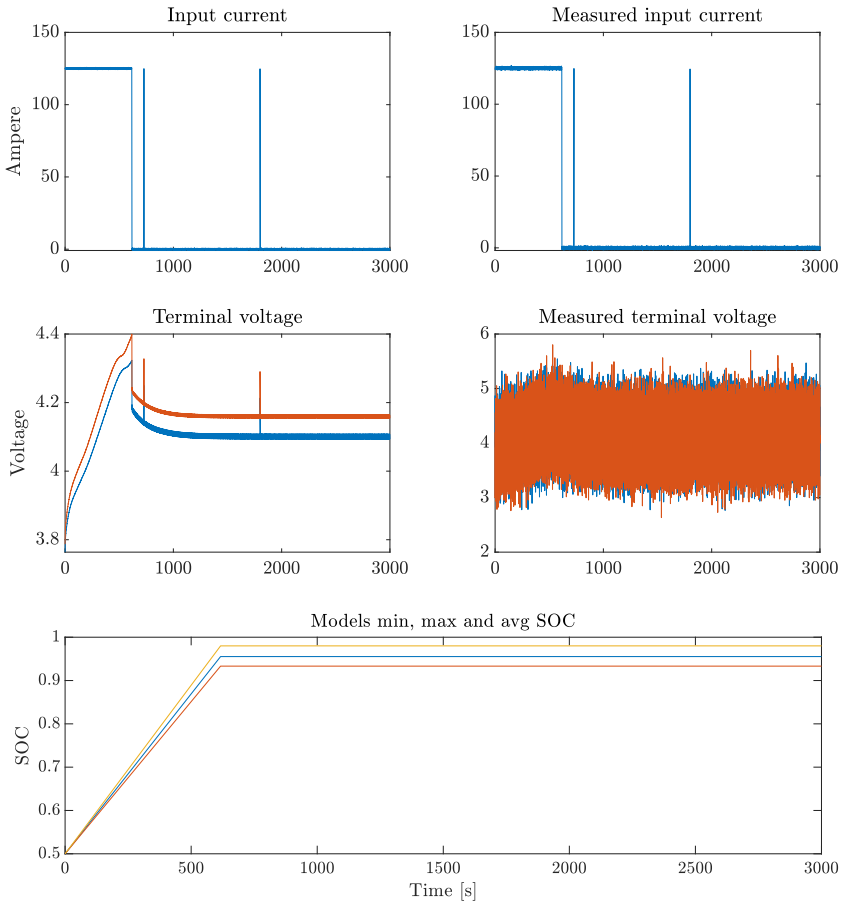


Figure A.4: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 4 listed in Table 3.1. Namely input index 1, $z_0 = 0.5$, 100 cells.

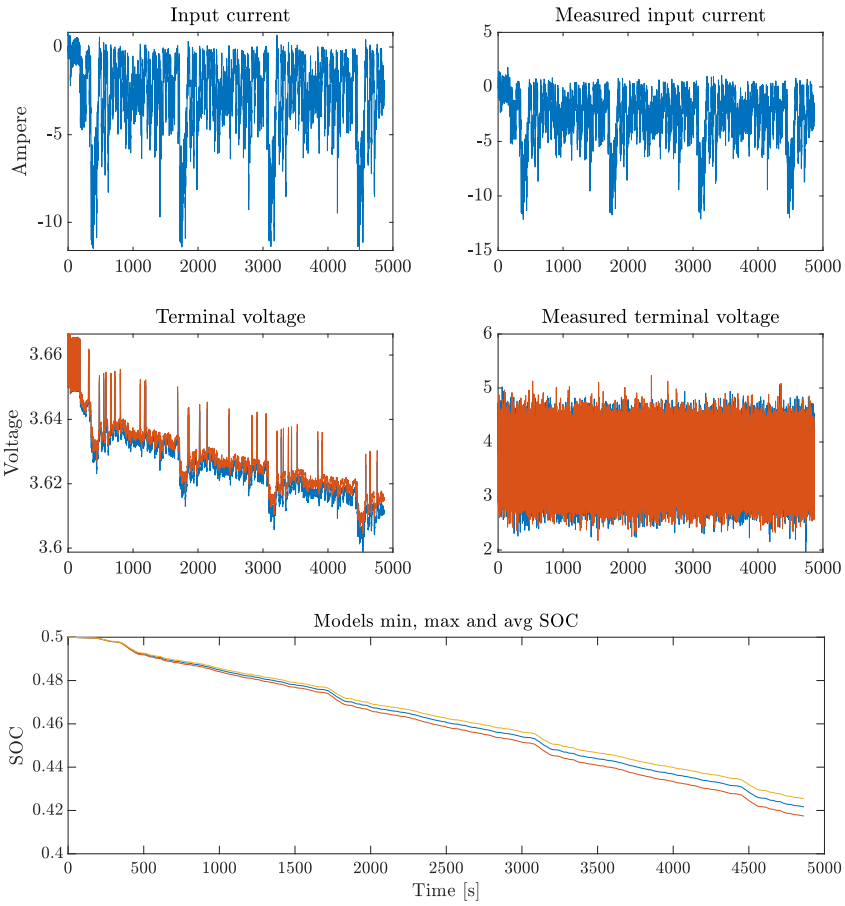


Figure A.5: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 5 listed in Table 3.1. Namely input index 2, $z_0 = 0.5$, 100 cells.

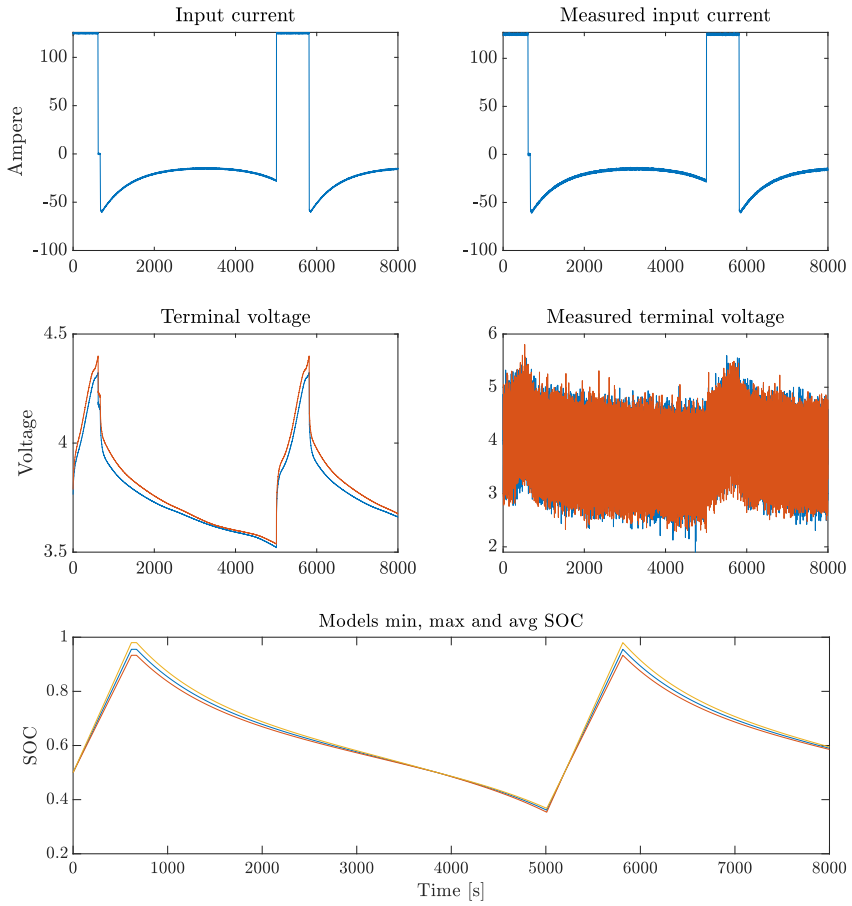


Figure A.6: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 6 listed in Table 3.1. Namely input index 3, $z_0 = 0.5$, 100 cells.

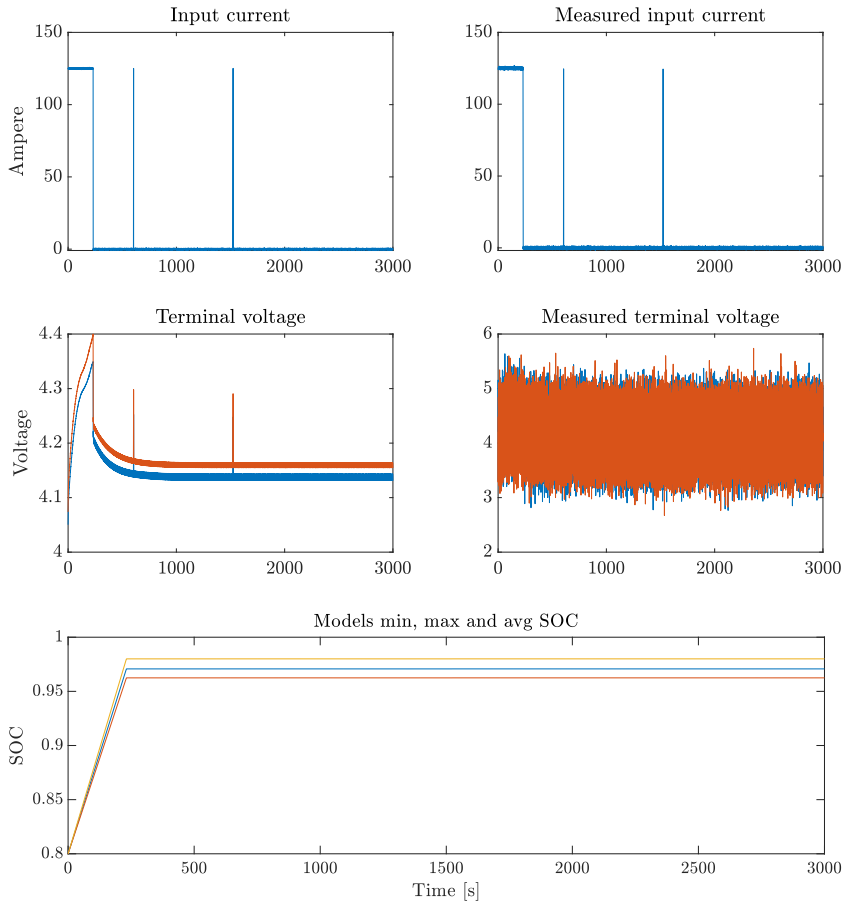


Figure A.7: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 7 listed in Table 3.1. Namely input index 1, $z_0 = 0.8$, 100 cells.

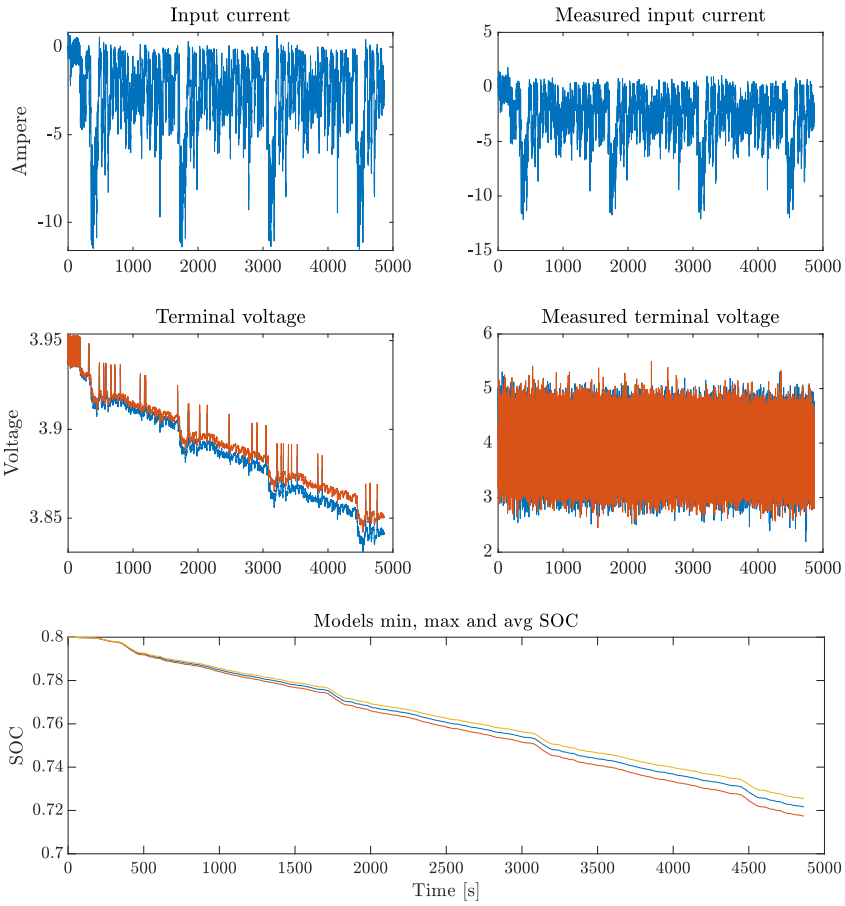


Figure A.8: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 8 listed in Table 3.1. Namely input index 2, $z_0 = 0.8$, 100 cells.

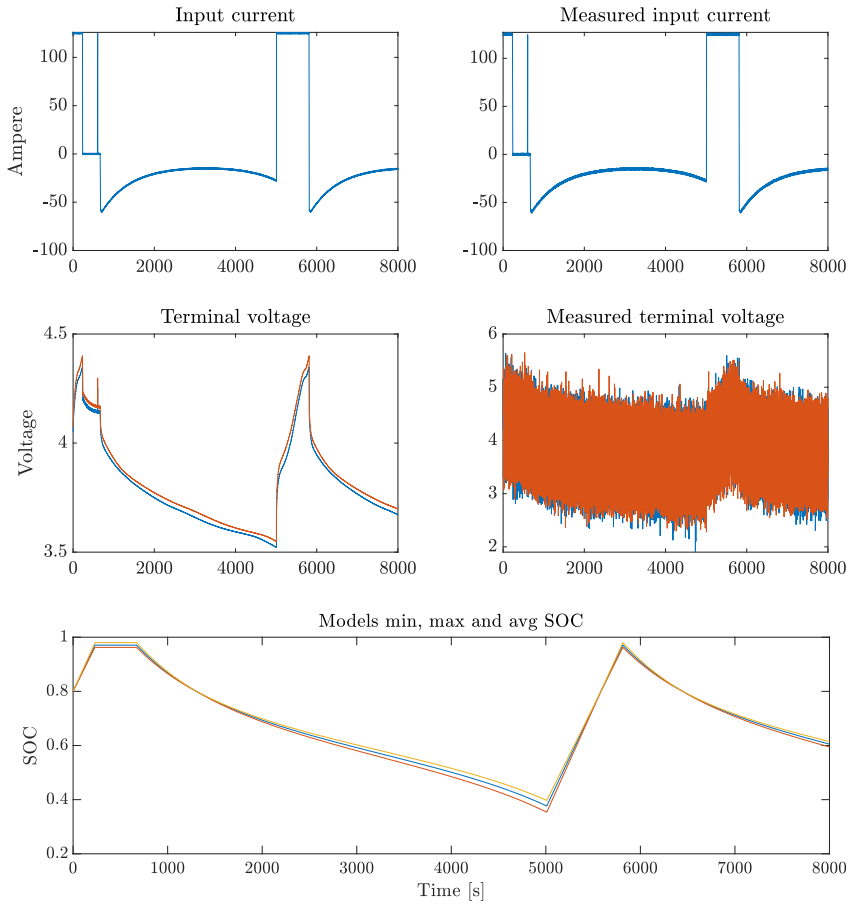


Figure A.9: Input current, measured input current, terminal voltage, measured terminal voltage, the models minimum, maximum, and average SOC for input scenario 9 listed in Table 3.1. Namely input index 3, $z_0 = 0.8$, 100 cells.

B

Accuracy test results

This appendix consists of 9 figures, each displaying the absolute error of the average, maximum, and minimum SOC estimated by the estimation algorithms XEKF, XSPKF, 3EKF, 3SPKF, 1EKF$_{log}$, 1SPKF$_{log}$, 1EKF, and 1SPKF for 100, 200, and 300 cells. Each figure is slightly different, each displaying the mentioned values for different combinations of the input index $\in \{1, 2, 3\}$ and the initial SOC $\in \{0.2, 0.5, 0.8\}$. The colors in the following figures corresponds to the different algorithms as seen in Figure 4.2.

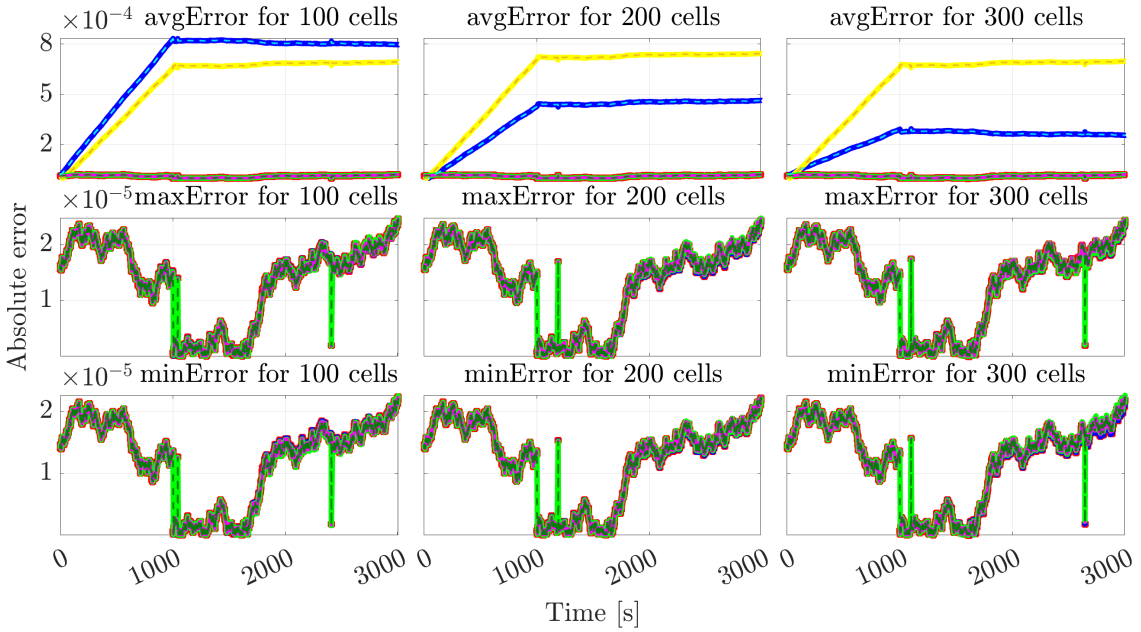


Figure B.1: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC evaluated by simulation scenarios 1, 10, and 19 listed in Table 3.1. Namely input index 1, $z_0 = 0.2$, 100, 200, 300 cells.

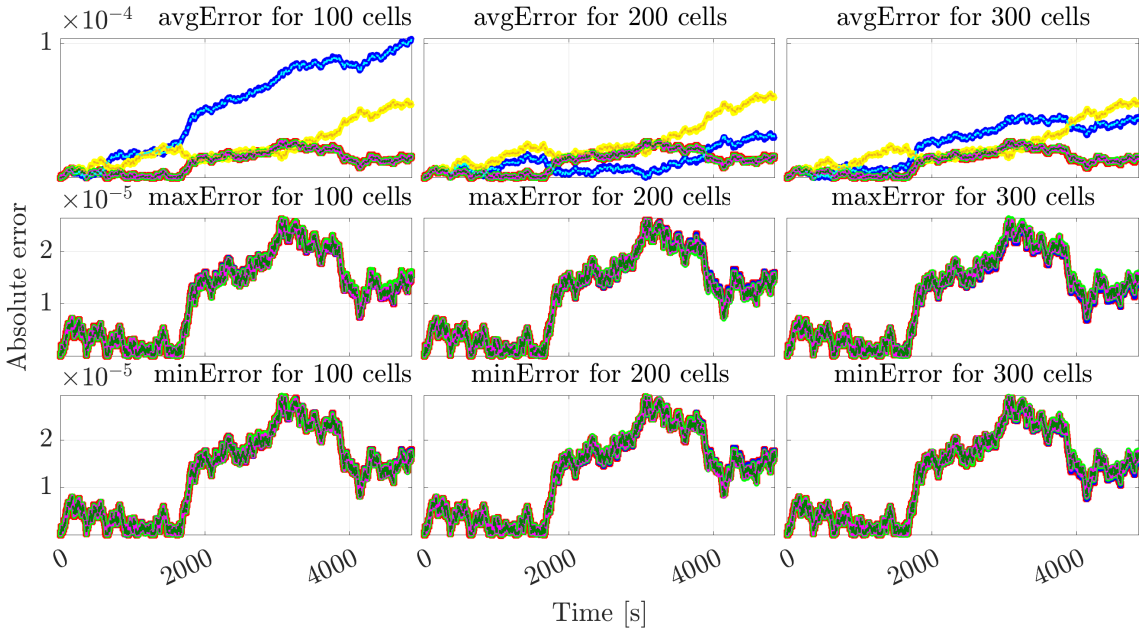


Figure B.2: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 2, 11, and 20 listed in Table 3.1. Namely input index 2, $z_0 = 0.2$, 100, 200, 300 cells.

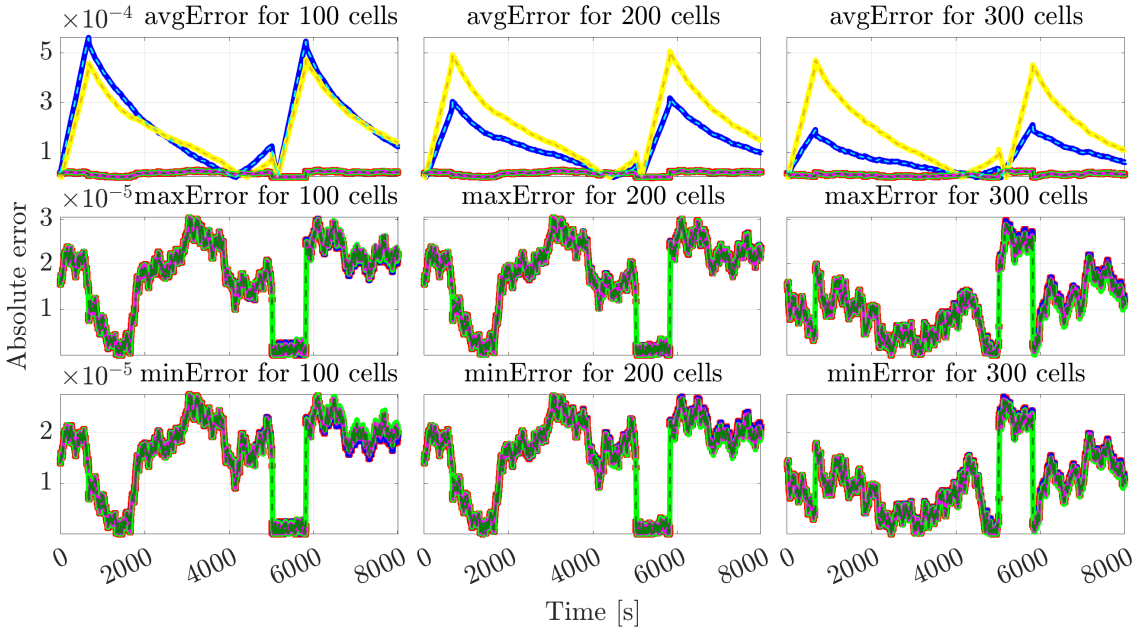


Figure B.3: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 3, 12, and 21 listed in Table 3.1. Namely input index 3, $z_0 = 0.2$, 100, 200, 300 cells.

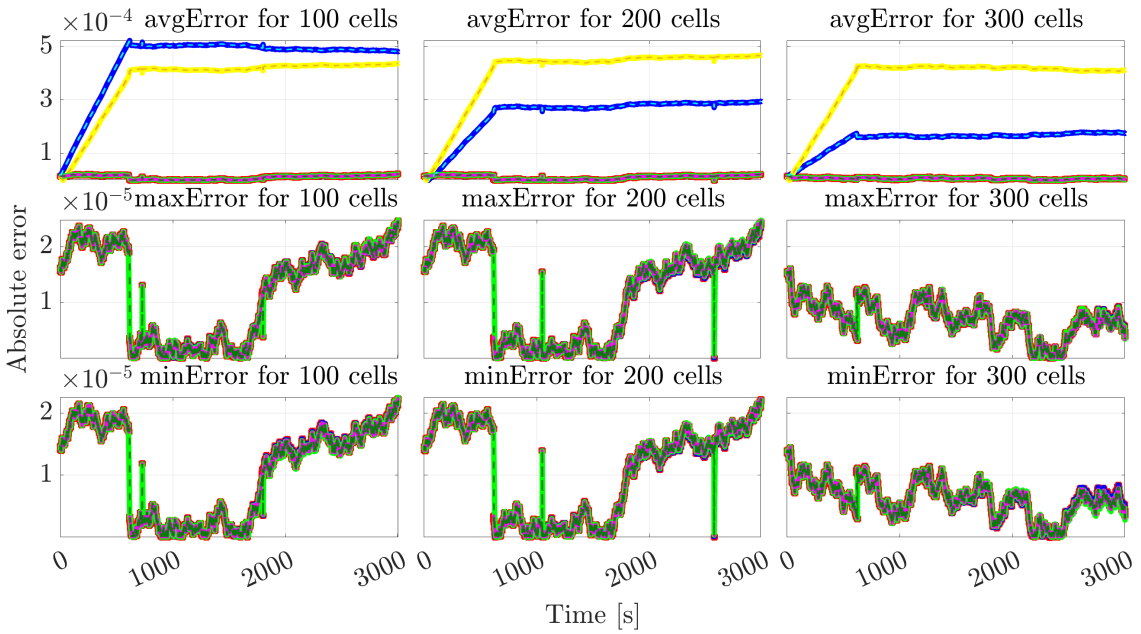


Figure B.4: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 4, 13, and 22 listed in Table 3.1. Namely input index 1, $z_0 = 0.5$, 100, 200, 300 cells.

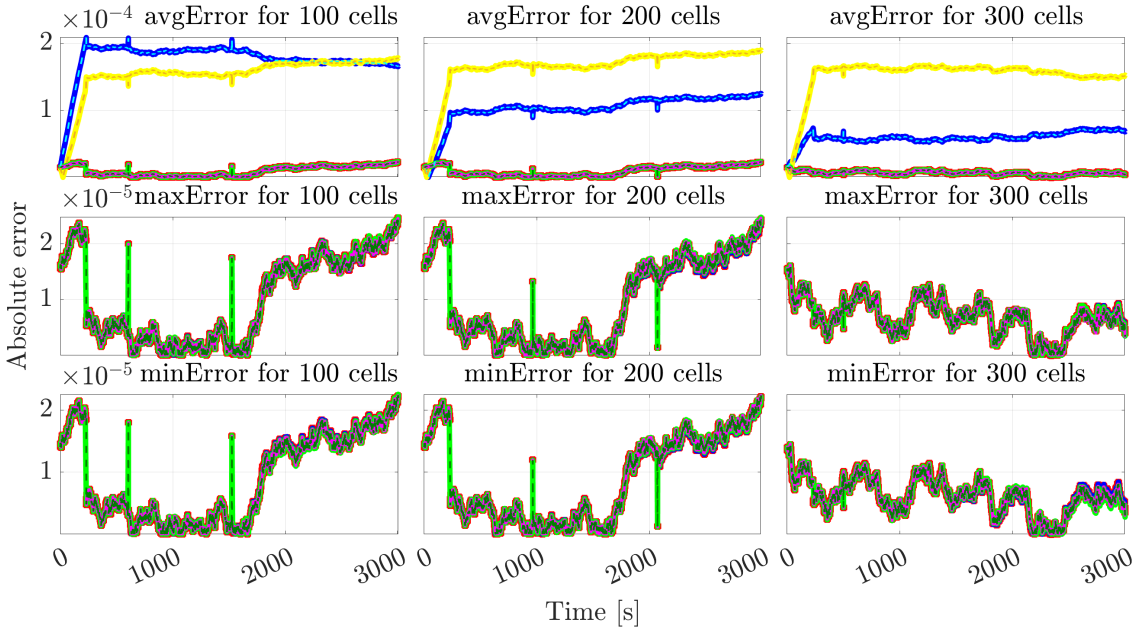


Figure B.5: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 7, 16, and 25 listed in Table 3.1. Namely input index 1, $z_0 = 0.8$, 100, 200, 300 cells.

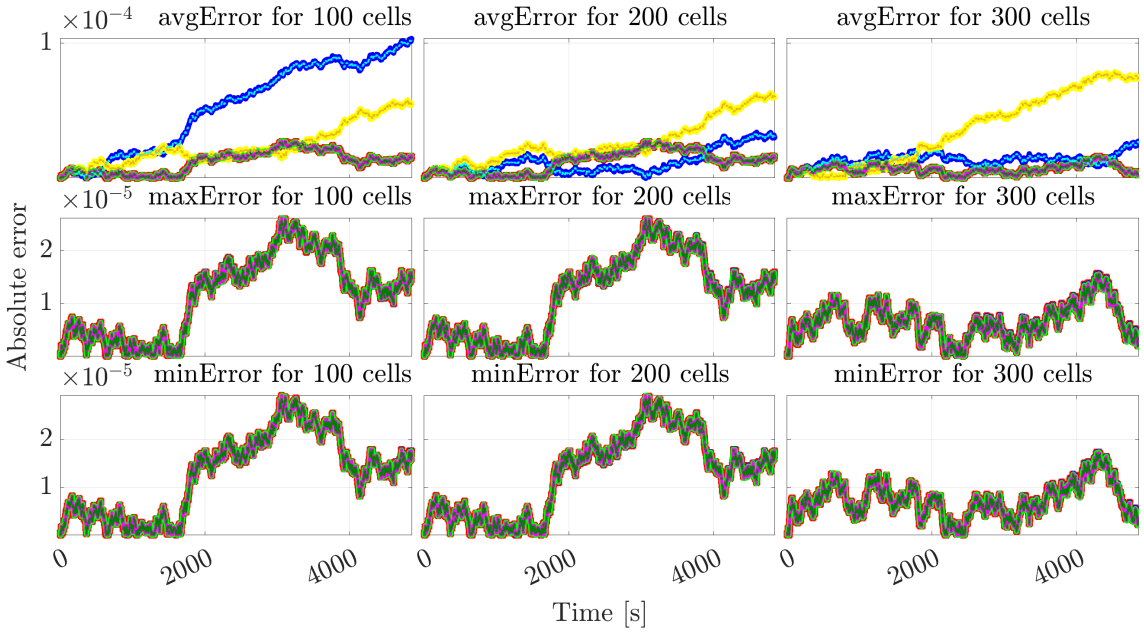


Figure B.6: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 5, 14, and 23 listed in Table 3.1. Namely input index 2, $z_0 = 0.5$, 100, 200, 300 cells.

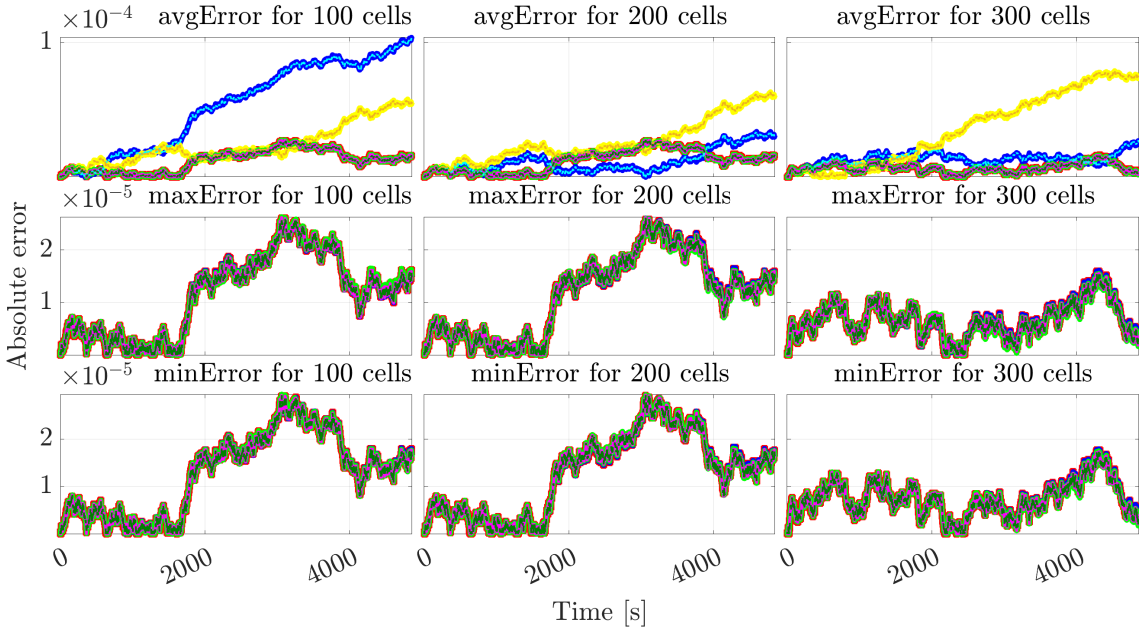


Figure B.7: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 8, 17, and 26 listed in Table 3.1. Namely input index 2, $z_0 = 0.8$, 100, 200, 300 cells.

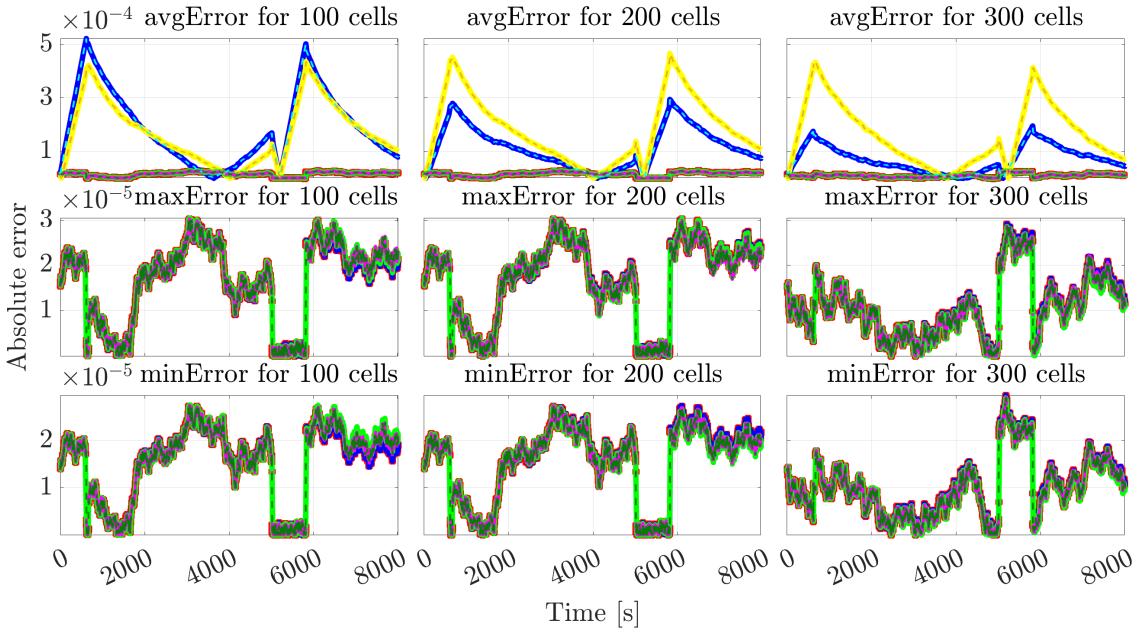


Figure B.8: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 6, 15, and 24 listed in Table 3.1. Namely input index 3, $z_0 = 0.5$, 100, 200, 300 cells.

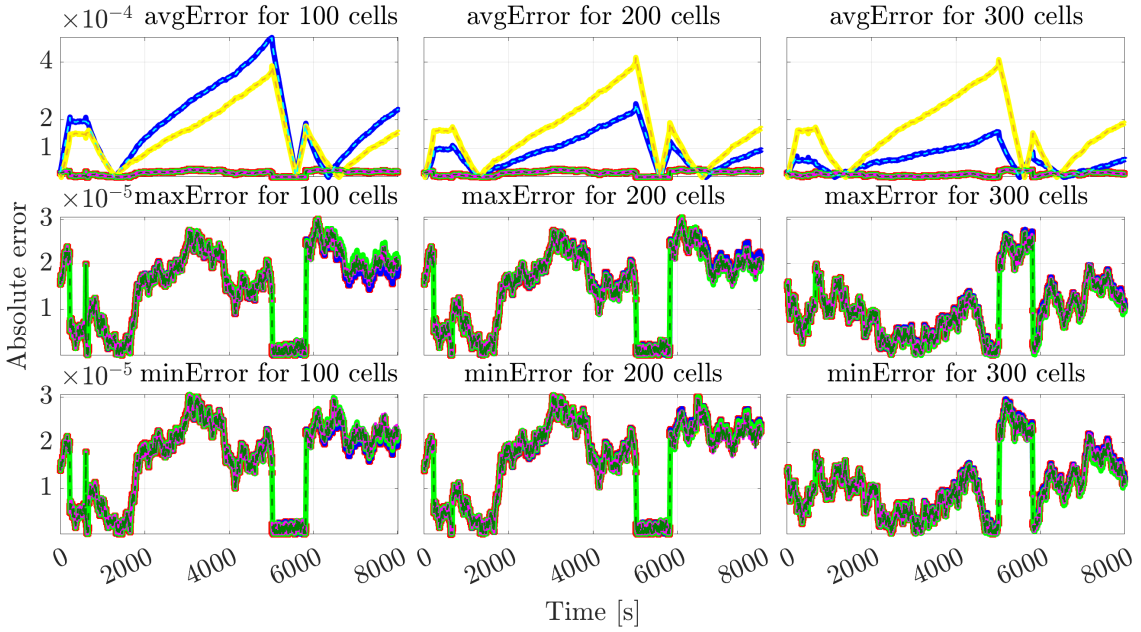


Figure B.9: Accuracy tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 9, 18, and 27 listed in Table 3.1. Namely input index 3, $z_0 = 0.8$, 100, 200, 300 cells.

C

Robustness test results

This appendix consists of 9 figures, each displaying the absolute error of the average, maximum, and minimum SOC estimated by the estimation algorithms XEKF, XSPKF, 3EKF, 3SPKF, 1EKF$_{log}$, 1SPKF$_{log}$, 1EKF, and 1SPKF for 100, 200, and 300 cells where there is an additive noise on the temperature input seen in Figure 3.3. Each figure is slightly different, each displaying the mentioned values for different combinations of the input index $\in \{1, 2, 3\}$ and the initial SOC $\in \{0.2, 0.5, 0.8\}$. The colors in the following figures corresponds to the different algorithms as seen in Figure 4.2.

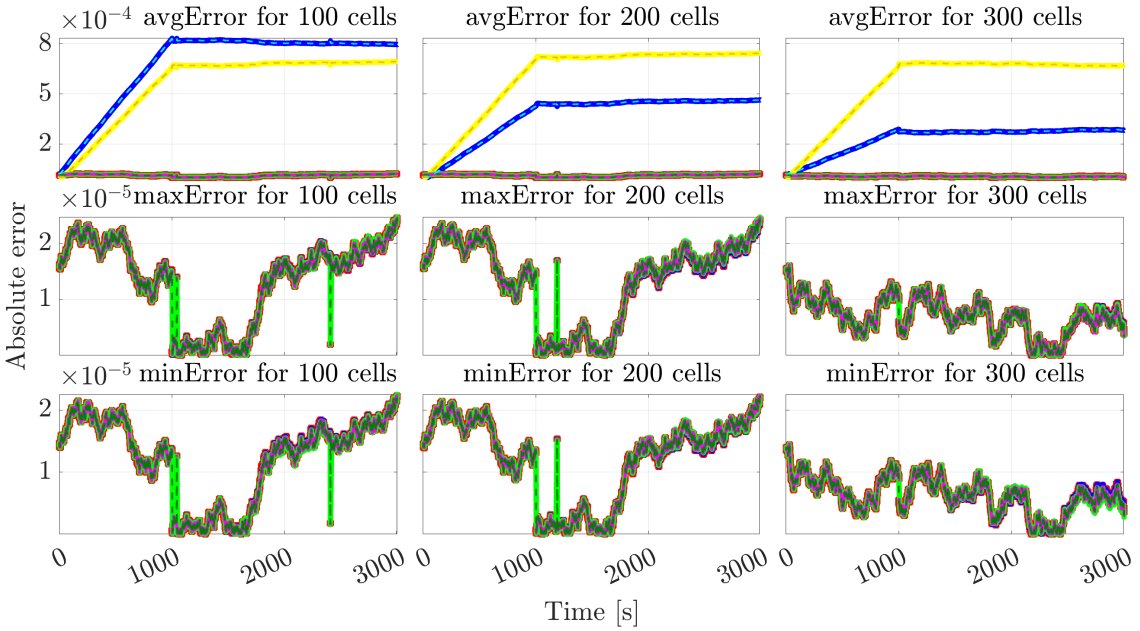


Figure C.1: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 1, 10, and 19 listed in Table 3.1. Namely input index 1, $z_0 = 0.2$, 100, 200, 300 cells.

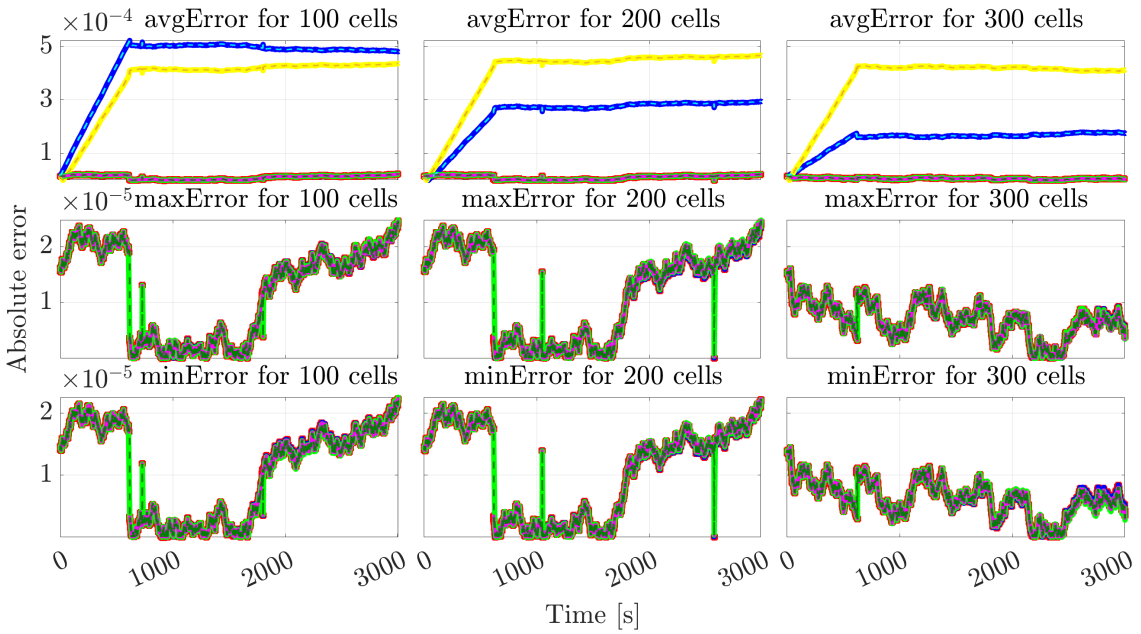


Figure C.2: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 4, 13, and 22 listed in Table 3.1. Namely input index 1, $z_0 = 0.5$, 100, 200, 300 cells.

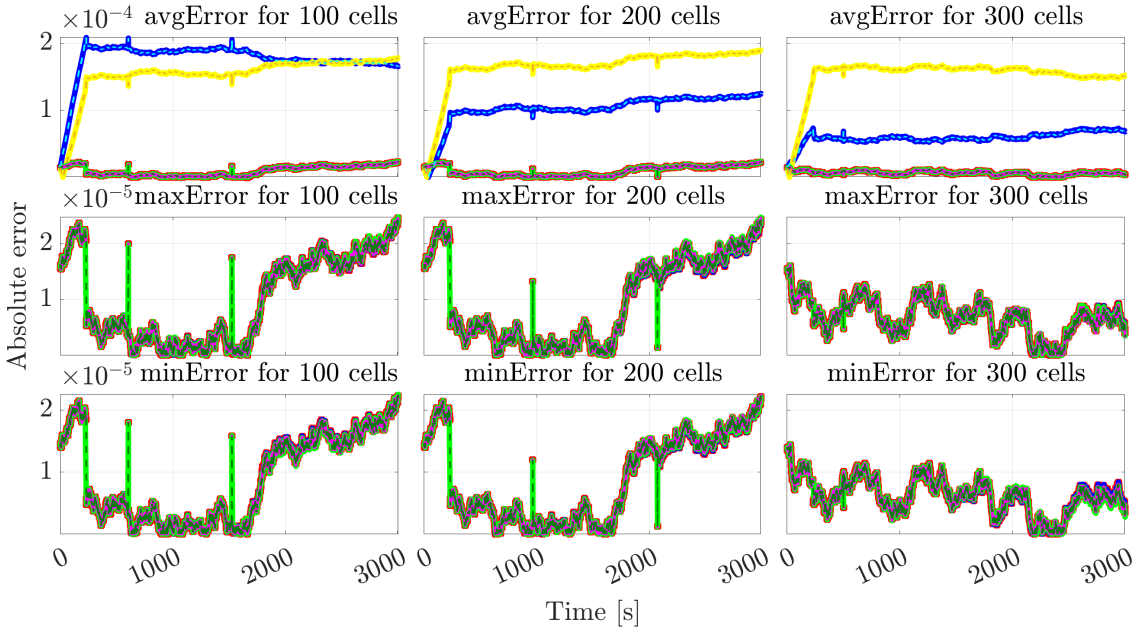


Figure C.3: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 7, 16, and 25 listed in Table 3.1. Namely input index 1, $z_0 = 0.8$, 100, 200, 300 cells.

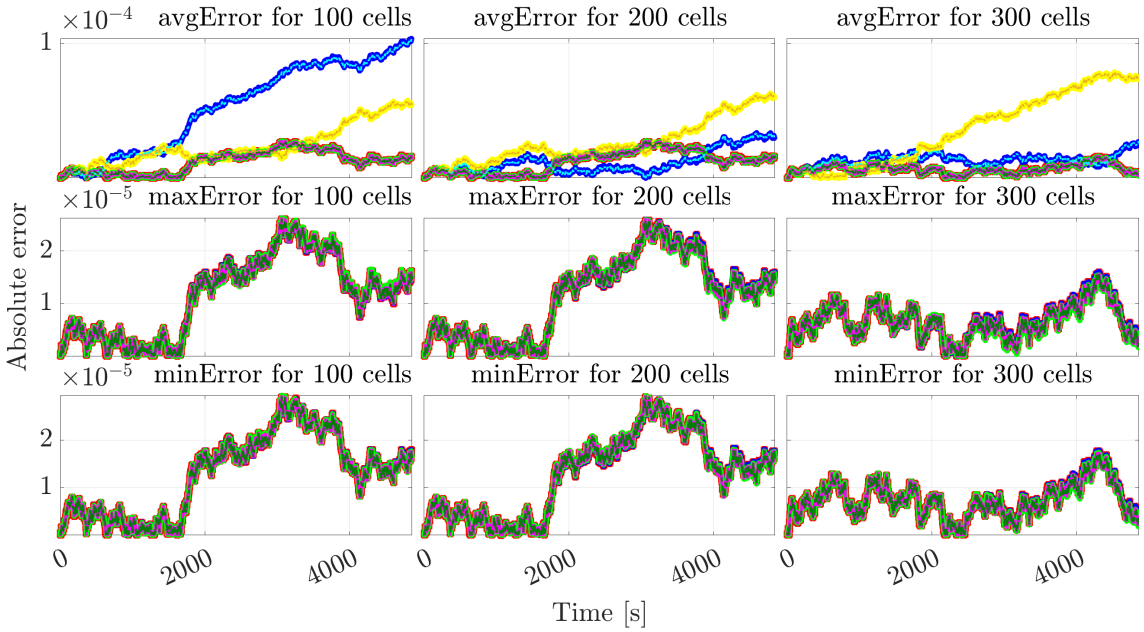


Figure C.4: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 2, 11, and 20 listed in Table 3.1. Namely input index 2, $z_0 = 0.2$, 100, 200, 300 cells.

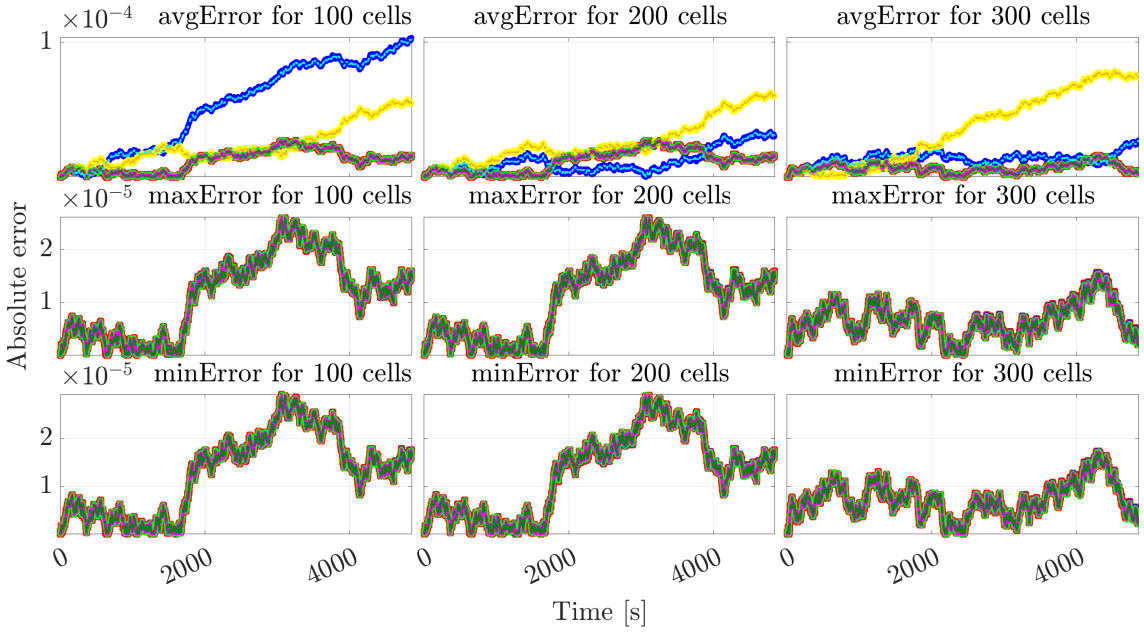


Figure C.5: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 5, 14, and 23 listed in Table 3.1. Namely input index 2, $z_0 = 0.5$, 100, 200, 300 cells.

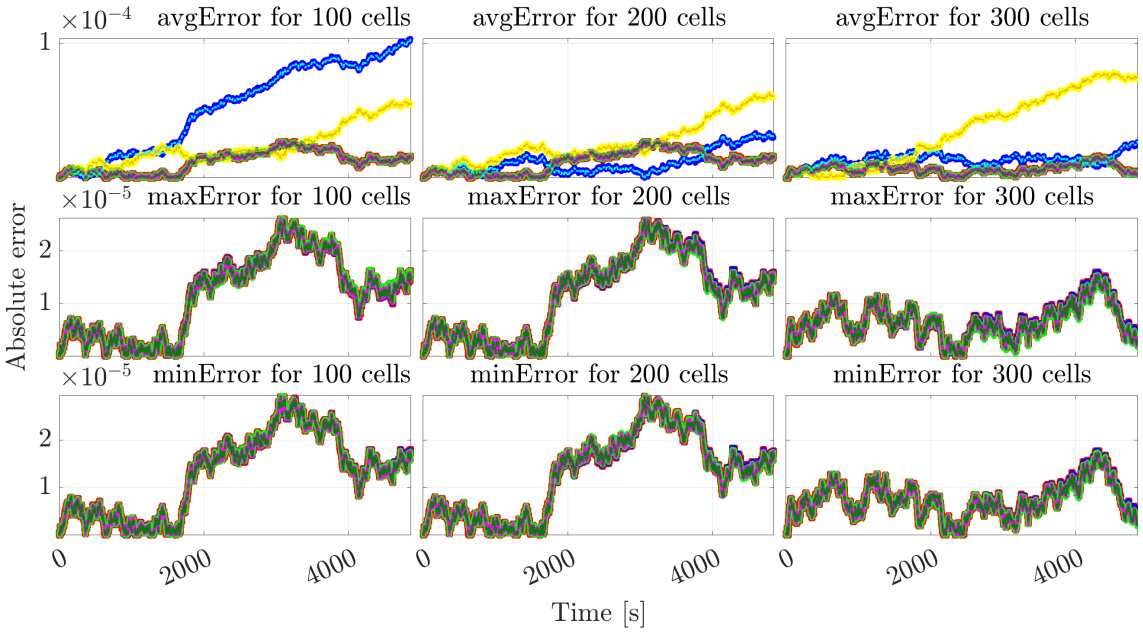


Figure C.6: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 8, 17, and 26 listed in Table 3.1. Namely input index 2, $z_0 = 0.8$, 100, 200, 300 cells.

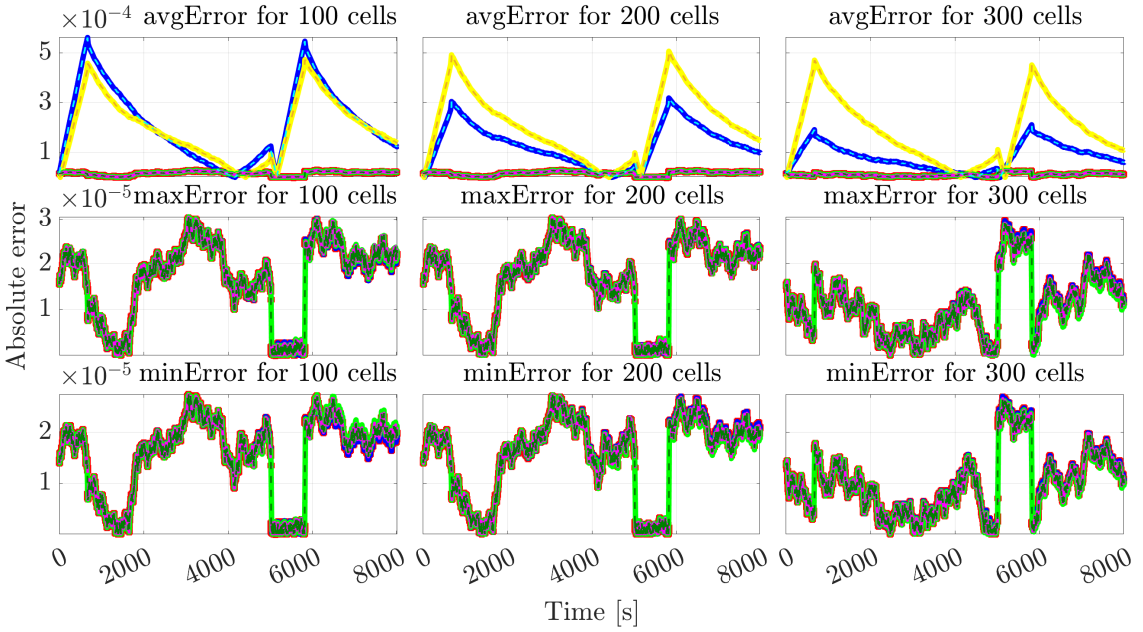


Figure C.7: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 3, 12, and 21 listed in Table 3.1. Namely input index 3, $z_0 = 0.2$, 100, 200, 300 cells.

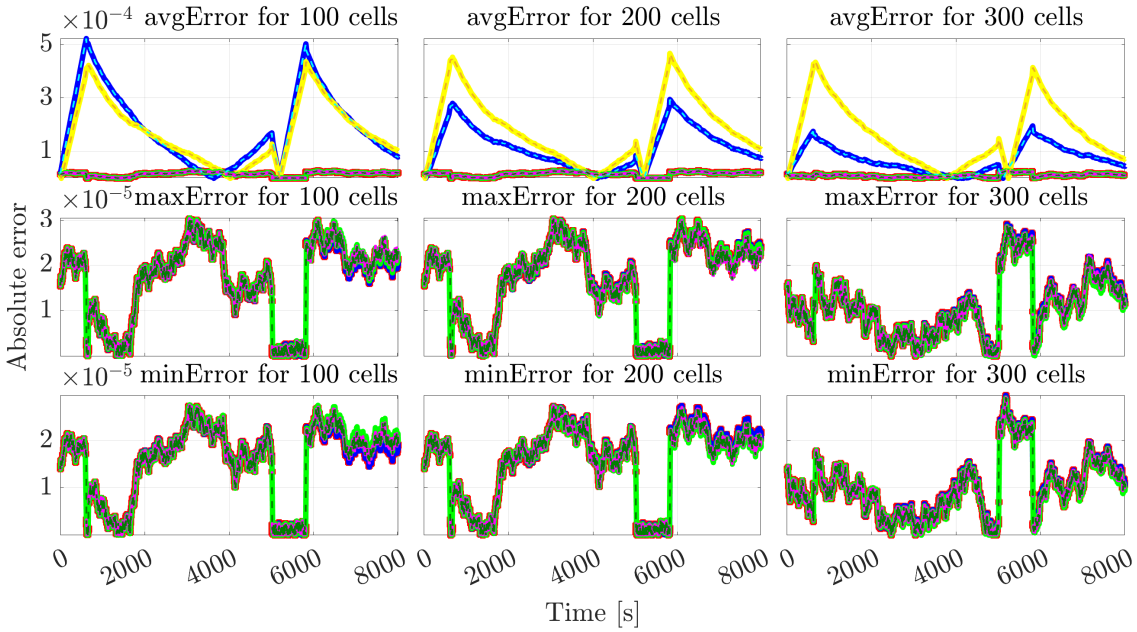


Figure C.8: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 6, 15, and 24 listed in Table 3.1. Namely input index 3, $z_0 = 0.5$, 100, 200, 300 cells.

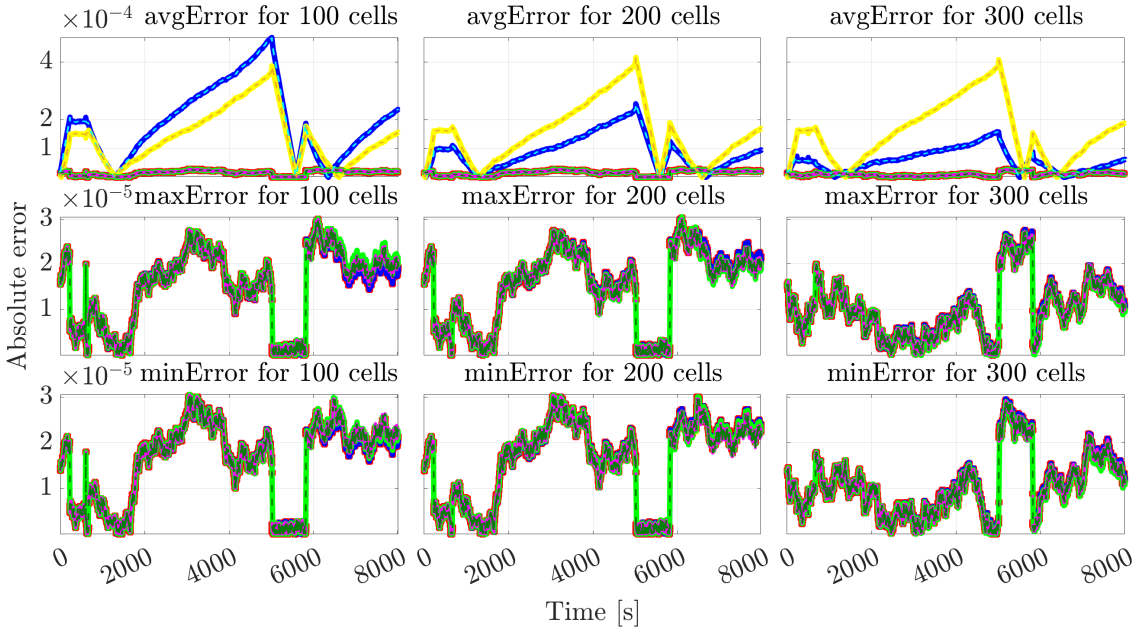


Figure C.9: Robustness tests absolute error of the estimated average, maximum, and minimum SOC estimated by simulation scenarios 9, 18, and 27 listed in Table 3.1. Namely input index 3, $z_0 = 0.8$, 100, 200, 300 cells.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY