CHALMERS
UNIVERSITY OF TECHNOLOGY

Adaptive Dual Estimation of Battery State of Charge and State of Health

Master's thesis in Systems, Control & Mechatronics

Leon Glass

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Adaptive Dual Estimation of Battery State of Charge and State of Health

LEON GLASS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2019

Adaptive Dual Estimation of Battery State of Charge and State of Health
LEON GLASS

Supervisor: Faisal Altaf, VOLVO Group Trucks Technology
Examiner: Torsten Wik, Department of Electrical Engineering

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

Typeset in LaTeX
Gothenburg, Sweden 2019

Adaptive Dual Estimation of Battery State of Charge and State of Health
LEON GLASS
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The growing focus on electric vehicles has led to a high demand for advanced battery management system (BMS) technologies for optimal utilization of batteries in terms of energy efficiency, safety, and robustness. In particular, battery estimation functions like state of charge (SoC) and state of health (SoH) are a crucial part of BMS. Recently, dual state and parametric estimation using Kalman filtering has emerged as a prominent solution to the SoC estimation problem. Many augmentations and improvements on dual state and parametric estimation have been individually proposed in the battery estimation community. In this thesis a comprehensive SoC and SoH estimation scheme is developed, which incorporates many recent advances of battery estimation. This includes adaptive tuning of the state filter in SoC estimation, a sensitivity analysis feature to enhance the parameter filter in SoC estimation and changes made to the estimation algorithms to accommodate these enhancements. The SoC estimator is developed in two variations, where one version uses extended Kalman filtering (EKF) and the other version uses unscented Kalman filtering (UKF). The most significant contribution made by this thesis is the approach of cascading the dual SoC estimator with a moving horizon estimator (MHE) for SoH estimation. This improves SoC estimation by incorporating SoH estimates and vice versa.
The proposed SoC and SoH estimation scheme is validated both in simulations and in experiment. The results show that it gives higher SoC estimation accuracy even when compared to estimators using high fidelity battery model data not available to it. The SoC and SoH estimator developed in this thesis distinguishes itself by the superior quality of its estimates in the presence of effects like battery aging, low measurement accuracy and trajectories exhibiting low sensitivity with respect to the system's parameters. Additional research is needed in order to employ adaptive tuning of the SoC estimator in the presence of large sensor bias.

Keywords: state of charge, state of health, battery management system, state and parametric estimation, Kalman filter, moving horizon estimation, adaptive tuning, sensitivity analysis

Acknowledgements

I would like to thank my supervisor Dr. Faisal Altaf at the Volvo Group for all the guidance, patient instruction and encouragement he gave me throughout my work on this thesis.
To my fellow thesis students at the Volvo Group I am thankful for the many interesting discussions we shared and the relaxed and productive atmosphere I enjoyed and profited from very much.
I am grateful to the Volvo Group for providing the means and financial support that made this thesis possible.
I would like to thank Prof. Torsten Wik at Chalmers for his invaluable contributions to this thesis.
I extend my gratitude to Philipp Köhler and Prof. Frank Allgöwer at the University of Stuttgart who have given me the opportunity to study in the double master's program Technische Kybernetik/Systems, Control & Mechatronics.
Finally, I am eternally grateful to my family for supporting me from the earliest days of childhood and encouraging my talents at every turn of the way.

Leon Glass, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

The automotive industry is on the move towards environmentally sustainable modes of transportation. We continue to see a strong push away from cars powered by internal combustion engines and towards electric vehicles (EVs) and plug-in hybrid electric vehicles (PHEVs). Relying on EVs as a major means of transport brings as many challenges as it brings opportunities. Many technologies have to be developed further in order to make the new form of mobility as sustainable, efficient and reliable as possible. One challenge of prime importance is the development of modern, highly accurate battery management systems (BMS).
At the heart of every EV lies its source of power: the battery. The battery as a mode of energy storage, while highly valued for its environmental sustainability, comes with many challenges. Some of these challenges arise from the sheer complexity of the system: the battery is a complex electrochemical system featuring a multitude of chemical, thermal and electric phenomena. Correct function of a battery results from the interplay of all these and requires specific conditions such as proper operation and temperature of the battery. Ensuring this is the purpose of a BMS.

1.1 Motivation

The battery management system has the main task of ensuring that the battery is operated inside its safe operating area. This plays a major role in protecting the battery from damage. There are many good reasons why this is highly desirable:
First and foremost, user safety critically depends on the battery's integrity. Batteries contain highly reactive, corrosive and toxic materials. Unsafe operation of batteries can potentially lead to catastrophic damage, such as battery explosion. Second, the battery is the main source of power of EVs. This means that the immediate functionality and mobility of the vehicle depends on the battery's ability to supply power to the vehicle's powertrain. This ability can only be guaranteed, if the battery is operated in its safe operating area. Third, battery damage will result in a reduced battery lifespan (i.e. short timespan where the battery's quality is adequate for usage). The immediate outcome of this is that the battery has to be replaced more often. On the one hand this leads to a rising cost of the vehicle throughout its period of usage. On the other hand this also undermines the sustainability of the EV, as frequent battery changes also mean higher demand on critical resources needed for modern, high capacity batteries. Such materials include Lithium, Cobalt, Nickel, Manganese and other metallic oxides. These materials not only have to be mined when new demand arises, they also have to be recycled or disposed of properly.

Serious damage to the environment can result, if excessive quantities of battery waste material are generated or discarded improperly. The extraction of rare metals needed for EV batteries is often connected to serious harm to the environment, as well as inhumane conditions for local workers, since these materials are often mined in developing countries [25].

One features that lies at the core of a BMS is battery state of charge (SoC) estimation. A useful metaphor to give readers new to the area of research an intuitive idea of the concept is the following: The battery's SoC is the closest equivalent an EV has to the fuel gauge of a combustion engine driven car.

The importance of having accurate information about the SoC cannot be overstated. SoC monitoring is used to predict the vehicles remaining range. SoC is also a critical factor in delimiting the battery's safe operating area. Here, the concept of overcharging and overdischarging is key. This means continuing to charge or discharge the battery once its SoC has reached 0% or 100% respectively. This leads to adverse chemical reactions. During overcharging the electrolyte decomposes and reacts with the material of the cathode. During overdischarging both the anode's and the cathode's structures corrode [16]. Both of the above lead to a permanent deterioration in the battery's quality. This comes in the form of increased internal resistance or decreased total capacity. Conversely, having an accurate estimate of SoC allows optimal use of the battery within the design limits. This enables robust and safe utilization of available energy resource leading to higher efficiency of the vehicle powertrain.

In contrast to the level in a fuel tank, SoC cannot reasonably be measured in automotive applications. Rather. SoC depends on the concentration of Lithium ions in the electrolyte. To use the language of control theory, SoC is an internal state of the system, rather than a measurable quantity. Principally, the two quantities readily measurable are the current supplied to the system and the system's final (terminal) voltage. This is why state estimation algorithms from the field of control theory are applied widely to find a given battery's SoC. In order to apply the methods of model based control, we first need a model.

Knowledge about the battery's state of health (SoH) is another important factor of a BMS. Battery SoH as a quantity describes the battery's health, that is how much the battery's energy and/or power delivery ability has deteriorated since its fabrication. Such deterioration takes place even when not triggered by improper operation of the battery. A wide array of aging mechanisms are responsible for this behavior. For a more detailed review, see [2, 3, 30]. As such, SoH is a useful indicator of whether a battery has outlived its usefulness for a certain application. Once battery SoH has dropped below a certain threshold, a BMS can use this information to trigger a replacement of said battery. It is pertinent to underline that a battery need not be discarded when its SoH disqualifies it from serving its previous application. Different applications place different demands on batteries. A battery that can no longer be used for one application, can still be used for a less challenging application.

1.2 Battery Modeling

Model based control relies heavily on the fact that the controlled system is accurately reflected by the model used for the design of any applied control methods. However, we also have to make application of the developed control solutions feasible in the automotive sector. Therefore, when planning to design an estimator, one should make very conscious choice of design model. The tradeoff between model fidelity and model complexity can have wide-reaching repercussions in estimator performance. This is even more important, when advanced and computationally intensive methods of state observation are applied. Examples of this are Unscented Kalman Filtering (UKF) and Moving Horizon Estimation (MHE). Batteries exhibit thermal as well as electric dynamics. As they are quite complex systems, a wide array of modeling approaches has been explored in the literature.

1.2.1 Electrochemical Modeling

Electrochemical models are derived from first principles of physics. They set out to capture all characteristics of the complex battery system. To achieve such high fidelity, the battery has to be modeled as having distributed dynamics. This leads to a model consisting of a set of partial differential equations (PDEs). This model is then highly useful for understanding the electrochemical processes that take place inside the battery. For all the benefits of this model's high fidelity, there are a couple of drawbacks, which render this approach all but useless for the present application.

First, models using PDE's require significant computational and memory resources, rendering it infeasible for on-board systems. Second, electrochemical models involve a large number of unknown parameters, which have to be determined (usually experimentally) in order to be able to run the model. Third, using a model consisting of PDEs complicates observer design, considerably.

For the above drawbacks, electrochemical models are seldom used in automotive applications of SoC estimation, despite generating a lot of academic interest. They are not investigated further in this thesis.

1.2.2 Black Box Modeling

Black box models try to model the system's behavior using no knowledge of the underlying system dynamics. This is mainly done using some available amount of experimental training data. One way to achieve this sort of model is to use neural networks. To achieve suitable accuracy, large amounts of data and parameters are needed. Model accuracy is highly dependent on correct excitation of the plant and may be restricted to those areas where the system was explored experimentally. Due to limited availability of data, and because high accuracy is very much needed, this thesis does not pursue this type of model further.

1.2.3 Equivalent Circuit Models

Equivalent Circuit Models (ECMs) are widely used in SoC estimation, as they combine favorable features like low complexity and easy observer design with adequate model accuracy. The model seeks to capture the key effects of electrochemical phenomena happening inside a battery cell using an ordinary differential equation (ODE) based model. ECMs fall in the category of lumped models, since they describe phenomena normally modeled by PDEs using ODEs only. The battery system is represented by a combination of electric elements such as voltage sources, resistors and capacitors [7].
Populating ECMs with suitable parameters is challenging. Due to these parameter's complicated nonlinear dependence on SoC, temperature and battery current, finding an explicit model for them is not practical. Therefore, using this type of model for SoC estimation requires simultaneous parametric estimation to ensure satisfactory fidelity of the ECM.
There are abundant variations on the ECM. Within the family of ECMs one once again faces the tradeoff between model fidelity and simplicity. Variations on ECM include different model orders, as well as varying degrees of hysteresis modeling. For an overview of different ECMs see [7]. A second order model with two resistance capacitor (RC) branches is found to perform very well with many different battery chemistries and under various conditions. The paper states clearly, however, that temperature, SoC and current dependencies are not captured and need to be captured through varying model parameters.

1.3 Estimation Algorithms

The task of SoC estimation of a battery cell is best described as a problem of internal state estimation. We are given a dynamical system, where some system inputs and outputs are measureable. The task is to reconstruct the system's internal state.
The Kalman filter (KF) is among the most widely used linear state observers. Given some linear process and measurement equations the KF can give accurate estimation of a linear, observable dynamical system. The KF has been widely and successfully applied to a great variety of engineering problems such as global positioning (GPS). The Kalman filter is also immensely popular in control applications, where a reconstruction of internal state is needed for the purpose of state feedback control. However, due to its linear nature the KF is decidedly limited in the range of problems it is applicable to. Many relevant technical systems require nonlinear process equations to be modeled accurately. EV and PHEV battery systems are no exception here: The nonlinear dependence of battery systems' output voltage on SoC makes it infeasible to model them using a set of linear differential equations. For applications like this, the KF is not applicable.
There exist several extensions to the KF to enable it to observe nonlinear processes. The most common of these extensions is the Extended Kalman Filter (EKF). It applies the principles of the KF to nonlinear processes. This can be done by calculating the first order Taylor approximation of the process equations, and thus a set of linear equations, at every computational step. This is equivalent to supplying the

KF with a linear time varying system as opposed to a linear time invariant system, as in the case of regular KF. Note that Taylor approximation requires differentiation of the process equations and is always performed at the current estimated state. Access to accurate estimations of the previous states is therefore paramount to ensure convergence of the filter. Approximation of the process equations away from the true state may lead to linear equations that no longer sufficiently model the process. The EKF has developed into the quasi-standard option for state observation of nonlinear processes.

Despite the EKF's popularity there exist other extensions of the KF to make it applicable to nonlinear processes. The Unscented Kalman Filter (UKF) is another modified Version of the KF, which also allows for observation of nonlinear systems. Instead of relying on linear approximations of the system, the UKF executes a number of function evaluations at every step to approximate the covariance matrix. This comes with a couple of advantages. Notably, derivatives of the system equations are not required, nor does the system even need to be differentiable. The UKF also has the potential to perform better than the EKF, whenever the system is highly nonlinear and linearization doesn't constitute a sensible approximation.

Gregory L. Plett applies both the EKF and the UKF to the problem of SoC estimation in his highly influential series of papers [20, 21, 22, 23, 24]. Here, EKF and UKF are used to perform simultaneous state and parametric estimation. Experimental data from the Urban Dynamometer Driving Schedule (UDDS) is used to validate the filters. This drive cycle assumes constant temperatures and relatively gentle changes in SoC. UKF is found to slightly outperform EKF, while requiring the same computational resources.

Since then, using dual state and parametric estimation techniques for SoC estimation has been widely studied, with researchers trying to improve many different aspects of SoC estimation.

One field of research in recent times has been the introduction of adaptive tuning mechanisms [5, 28, 6]. This aims to improve robustness and performance of the dual estimation scheme under challenging conditions such as extreme SoC or temperature values.

Some researchers try to optimize the computational efficiency of Kalman filtering setups [11].

Capacity estimation is another highly important field. Battery capacity estimation is both crucial to ensuring SoC estimate accuracy throughout a battery's lifespan and highly useful as a SoH indicator. Therefore, capacity estimation has been widely studied in recent years [4, 27, 9].

Other parameters used in battery modeling can vary over the lifetime of a battery. Recently, there has been research on adapting the OCV-SoC curve throughout a battery's lifetime in order to improve estimation quality [10].

Further practical challenges in SoC estimation include handling the sensor bias of automotive grade sensors, which can be highly detrimental to estimation quality. This issue is tackled in [18, 14].

The problem of mitigating observability issues in parametric observation has been researched, recently. Sensitivity based data selection schemes have been proposed by [12, 13].

However, much work focuses on improving estimation techniques in one isolated area, neglecting numerous other issues of battery estimation. There is still a lot of work to be done in researching the relationship between many of the above augmentations. The compatibility of these feature and their combined improvements to SoC estimation need to be investigated.

1.4 Objectives

This thesis aims to take a more holistic approach to battery estimation. We incorporate many of the above improvements into a standard state and parametric dual estimation scheme. In particular, we set the following objectives for this thesis:

1. We implement a dual estimation algorithm to realize simultaneous state and parameter estimation of the battery system. Herein the technique of dual estimation serves to decouple the nonlinear dependencies of the parameters of the battery's electrical dynamics on a multitude of variables, such as state of charge, input current and temperature. Thus, we employ two Kalman filters, where one filter is used to estimate the battery system's state in terms of the electrical component of the battery and the other filter is used to estimate the parameters of the model to account for the parameters' dependency on state, temperature and input. This is achieved using only the commonly available measurements of the battery system's current and voltage. The above is implemented using the two filtering approaches of Extended Kalman filtering and Unscented Kalman filtering.
2. We augment each of the constituent filters of the dual estimator by a feature designed to improve performance under difficult conditions: In order to improve estimation performance under extreme circumstances such as extreme SoC values or temperatures, we augment the state estimator by an adaptivity feature. This feature allows the filter to self-tune its design parameters during run-time. Specifically, we adjust the covariance matrices supplied to the Kalman filters.
3. We improve the estimator's performance by augmenting the parameter filter by a data sensitivity feature. This feature has the purpose of checking the data received from the battery for sensitivity with respect to the battery parameters. The parameter filter only updates its estimate when sufficient sensitivity to the parameters has been established. This serves to avoid amplification of measurement noise by operating the filter with data insensitive to the parameters it estimates.
4. We improve the above dual estimation scheme by introducing a MHE, which takes as an input past values of the battery system's voltage and current to estimate the battery's actual capacity to account for the process of battery aging. The estimate of actual capacity is used to adapt the battery model upon which the dual estimator is based. This helps to decrease model plant mismatch and thereby improve estimation quality.

At the end of this thesis we review our success in accomplishing these objectives.

1.5 Scope

The thesis is limited to deliberations on the battery cell level. This means that the estimation scheme described therein is limited to observing the state of a single battery cell. Thus, no battery uniformity and equalization issues are treated. The terms "battery" and "cell" are used interchangeably in this thesis.
This thesis does not explicitly take the thermal dynamics of the battery into account. Instead, the influence of this part of the cell's dynamics is simply be taken into consideration as noise inside the estimation scheme.

1.6 Organization

This thesis is structured in the following way:
Chapter 2 starts by giving a brief review of the key issues and variables used in battery management. It proceeds by presenting the theory needed to effectively employ ECM for battery estimation. The Chapter then presents a thorough review of the Kalman filtering methods used in this thesis, including their background. This is followed by
Chapter 3 will detail how the theory presented in Chapter 2 is implemented to suit the peculiarities of the present task.
In chapter 4 we validate our resulting estimation setup. This is done both in simulation using a high fidelity validation model and using experimental data gathered from battery laboratories.
In chapter 5 we give some closing remarks and recommendations for future work, where further research is still needed.

2
Theory

The following chapter introduces theory required to work with battery estimation. We start by introducing our readers to the key concepts of battery management and introducing some useful terminology. We proceed to derive an ECM to serve as the basis of all our following efforts to design an estimator. A preliminary system analysis follows. We bear in mind, that the ECM only partly encapsulates our system's dynamics, which limits our ability to analyze the system rigorously using this model.
We continue to provide a thorough theoretical introduction to the family of state estimation methods collectively known as Kalman filtering. On that basis we acquaint our readers with the algorithms of Unscented Kalman filtering (UKF) and Extended Kalman filtering (EKF) for state estimation of nonlinear systems. The general approach to explore Kalman filtering in this thesis is to motivate all variations of Kalman filtering hereinafter presented as special cases of the more general sequential probabilistic inference approach to state estimation of dynamical systems.

2.1 Review of Key Issues of Battery Management

A battery is an energy storage device. More specifically, a battery is a device capable of using electrical energy to cause a reversible chemical reaction. This process is known as charging the battery. A charged battery can revert this process by using another chemical reaction to generate electrical energy, which is known as discharging the battery. The ability to charge or discharge a battery is limited by the availability of the required chemicals within the battery.
We pick the example of a Lithium-Ion battery to further illustrate the workings of a battery. Internally, a battery cell is constructed as follows: two electrodes are soaked in electrolyte and divided by a separator (i.e. a semi-permeable membrane) to stop electric short circuit. The discharging process consists of the following: Lithium active particles diffuse to the surface of the negative electrode. Here, they react to produce positively charged Lithium ions and electrons. The Lithium ions pass through the separator, while the electrons, which cannot pass through the separator, pass through the outer circuit. This results in a current through the outer circuit. The capacity of a battery in terms of the amount of energy it can supply depends on the amount of Lithium available in the cell. The inverse reactions take place during charging of the battery [34]. Let this description suffice to gain an intuitive understanding of the chemical inner workings of a battery. More detailed descriptions of these reactions are very complex and add nothing to the readers

understanding of this thesis.
To understand the Equivalent Circuit Model that is used in the course of this thesis, we need some concepts regarding the electric part of the model. Most importantly, we review the concepts of SoC and SoH and get a grasp of the time scales of the battery as a dynamical system.

2.1.1 State of Charge

The battery SoC is the EVs figurative stand-in for the fuel gauge in conventional vehicles. This means that SoC is an indicator of how much energy is stored in the battery at the current time. As we have seen, the amount of energy stored in a battery is directly linked to its pool of Lithium ions and, thus, to its resources in terms of number of electrons. This is quantified in the form of capacities
The total capacity Q is defined as the number of Coulombs drawn from a battery in order to transition it from its fully charged state to its fully discharged state. While we must expect this value to be subject to change due to battery aging and fluctuating temperature, we expect it to remain constant over short to medium time spans.
The residual capacity Q_r is the number of Coulombs drawn from the battery as to transition it from its present state to its fully discharged state. We readily expect this quantity to vary even over small time spans, as we charge or discharge the battery during operation. We pose the limit of $Q_\mathrm{r} \in [0, Q]$ by definition, where $Q_\mathrm{r} = Q$ means that the battery is fully charged and $Q_\mathrm{r} = 0$ means that the battery is fully discharged. Charging a battery beyond its fully charged state or discharging it below its fully discharged state is known as overcharging and overdischarging respectively. Overcharging and overdischarging are to be avoided at all costs: they provide no additional charge, while decomposing the battery's electrodes, decomposing its electrolyte and accelerating capacity fade in irreversible chemical reactions. In addition to this, they pose a threat to the functionality and integrity of the battery by causing internal short circuits and exothermic reactions [16]. This, aside from damaging the battery may cause safety hazards as serious as leakage of corrosive substances and explosion.
The SoC is the relationship between the battery's current capacity Q_r and its total capacity Q. We describe SoC by

$$z = \frac{Q_\mathrm{r}}{Q}, \tag{2.1}$$

where $z \in [0, 1]$ is the SoC [32]. Defining battery SoC becomes more of a challenge when considering battery systems consisting of more than one cell. In this case the definition of SoC depends on the presence of absence of a cell balancing mechanism [16]. The problem of cell balancing is very important when dealing with multiple cell batteries. Specifically, having a battery with well balanced cells is beneficial, since it allows one to fully utilize the battery's charge without overcharging or overdischarging individual cells. Since this thesis concerns itself with single cell estimation only, equalization approaches will not be discussed in detail. See [25] for a general overview of balancing techniques and further references.
Since SoC is very much an internal state of the battery, in that it is the direct

result of the chemical configuration inside the battery, we see that SoC cannot be conveniently measured. We also observe the utility of SoC and terminal voltage in preventing overcharging and overdischarging of the battery: to prevent further charging at $z = 1$ or at the end-of-charge-voltage (EoCV) is to prevent overcharging. To prevent further discharging at $z = 0$ or at the end-of-discharge-voltage (EoDV) is to prevent overdischarging. Therefore, we recognize that having accurate information on battery SoC and terminal voltage is paramount to ensuring safe operation of the battery.

2.1.2 State of Health

As we have seen the SoC of a battery seeks to encapsulate information about the amount of charge or, by extension, energy currently stored in the battery. SoH on the other hand is used to describe how much the battery, as a device, has deteriorated in its functionality since fabrication. While the definition presented for the SoC in (2.1) is, at least for the single cell case, universally accepted, the definition of SoH remains subject to debate.

Battery deterioration manifests itself in two forms: capacity fade and increase in internal resistance. As such, SoH describes irreversible change in the battery, whereas SoC describes reversible change. Capacity fade is mainly a side effect of the battery undergoing cycling, i.e. repeatedly charging and discharging the battery. Capacity fade and impedance growth is also due to the calendar aging of the battery. From these causes we have an intuitive grasp of the necessary difference in time scale between SoC and SoH. While changes in SoH only accrue over many days of battery usage, changes to SoC can happen in minutes or even seconds. We will realize as we develop a model for our estimation scheme that even changes in the time scale of milliseconds are relevant for SoC estimation. It is therefore prudent to also organize our estimation schemes for SoC and SoH so as to reflect this difference in time scale. Despite having solid phenomenological insight in the effects that cause battery aging the information is still not easy to encapsulate in a single variable [25].

Since battery health deterioration is best represented by capacity fade and internal resistance growth, the parameters best suited to carry this information are the battery's total capacity Q and the battery's internal resistance, which we will call R_0. Although both characterize SoH, there is one central difference that sets them apart.

Internal resistance R_0 , in addition to being caused by declining battery health, depends on the systems thermal state and to a lesser degree on current supplied to the battery and SoC. Thus, R_0 can potentially undergo fluctuations much faster than the usual time scale of SoH. Internal resistance is commonly included in dual SoC estimation [20, 24, 1, 12, 13]. This will also be the case in this thesis.

In terms of the total number of active particles in a battery, total capacity Q can reasonably assumed to be constant with regards to changes in temperature within the safe operation limits of the battery. The amount of charge we can actually draw from the battery can vary due to change in the end-of-charge and end-of-discharge voltages. Despite these effects, total capacity is usually taken to be entirely constant in SoC estimation [20, 24, 1, 12, 13]. Defining SoH using the total capacity facilitates

the use of more advanced estimation methods for SoH estimation, as this enables us to fully capitalize on the slower time scale of SoH.
The more common approach to define SoH is by examining the battery's capacity fade. However, since we will observe the battery's internal resistance in the course of parametric estimation, we can get a more complete image of battery health by analyzing this information, too.
We compare the battery's current total capacity Q with its rated total capacity Q_{rated} as supplied by the battery's manufacturer at fabrication. Note that Q_{rated} describes capacity at $23°C$ and a C-rate of 1. This leads to

$$\Psi = \frac{Q}{Q_{\text{rated}}}, \tag{2.2}$$

where Ψ is battery SoH. The definition in (2.2) makes estimating SoH tantamount to estimating total capacity Q. Consequently, we can use our estimate of SoH to further improve SoC estimation as total capacity Q is critical for model accuracy of the ECM. In our final estimation setup, the SoH estimator will directly feed the SoC estimator.
From the definition in (2.2) we expect SoH to decline monotonically with total capacity Q throughout the battery's life span.
Whereas accurate estimates of total capacity Q are highly useful for improving SoC estimation accuracy, proper usage of information about SoH is unclear. Rather than being self-explanatory, SoH estimates pose new questions: how far will we allow SoH to sink before discarding the battery from its current application? What applications can the battery feasibly be used for after this?
Finding well-founded answers to these questions is crucial to making efficient and sustainable use of existing battery resources. This is where SoH exerts its full utility. We see that information on SoH can help to replace batteries at the proper time and reuse them in the proper application when interpreted correctly. The interpretation of SoH data in this sense falls outside the scope of this thesis. We use SoH estimation to improve SoC estimation and observe that it has the aforementioned inherent uses.

2.1.3 Open Circuit Voltage

Open-circuit voltage (OCV) is the voltage measured across the battery's terminals when no load is applied to the battery and it has reached an equilibrium. This means that influences from past dynamic loads have been allowed to decay and we measure the intrinsic electromotive force of the battery cell. Usually, we find an injective function

$$V_{\text{OC}} : [0, 1] \to \mathbb{R}, \quad z \mapsto V_{\text{OC}}(z) \tag{2.3}$$

which maps SoC to OCV. That is, we find a function that maps SoC uniquely to OCV. Figure 2.1 shows a plot of the function V_{OC} for the battery used for validation in this thesis. The function V_{OC} is notably influenced by hysteresis, meaning that it will differ depending on whether the cell was lately charged or discharged. V_{OC} also varies slightly under changing temperatures. Both of these influences are, however, widely taken to be negligible for NMC batteries in estimation problems [20, 23, 12].

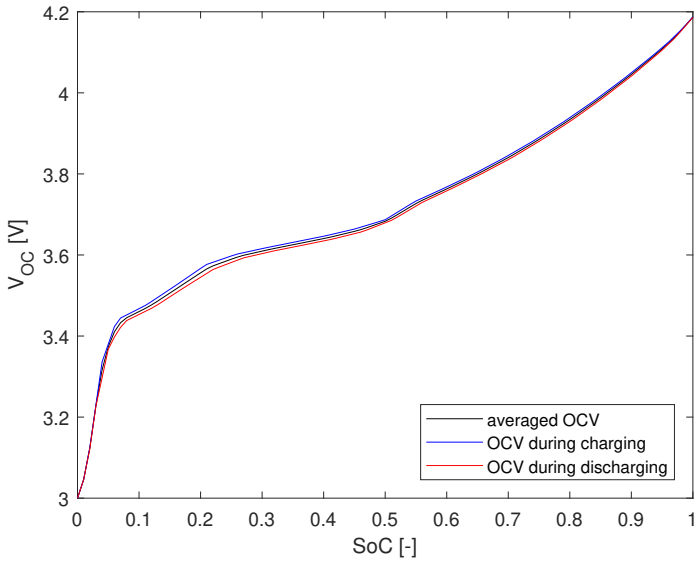

Figure 2.1: This Figure shows the plot of an example of the function $V_{\mathrm{OC}}(z)$, which maps SoC to OCV. The function shown is the OCV curve of the battery used for validation of the estimation setup used in this thesis.

In theory measuring OCV could be used to determine SoC using V_{OC}. In practice, this idea is difficult to apply and often highly inaccurate.
In automotive applications, measuring OCV is simply infeasible due to almost continuous battery operation over a wide region of SoC under heavy dynamic load. This precludes any measurement of OCV, as OCV can only be accurately measured under no load. Additionally, batteries commonly used in automotive applications, such as $LiFePO_4$ cells feature flat portions. This means that there are regions of SoC, where a very small change in OCV leads to a big change in the corresponding SoC value. Thus, for such batteries, even small measurement noise can lead to heavily distorted inferred SoC values.
We conclude that measuring OCV does not constitute a satisfactory method of estimating SoC.

2.1.4 Terminal Voltage

Terminal voltage is the voltage we measure across the battery's terminals during operation. We assign the variable V_{t} to mean terminal voltage. Whenever the battery is under load, or has recently been under load, terminal voltage V_{t} differs from OCV V_{OC}. As the load is removed, terminal voltage converges back to OCV. The difference between terminal voltage and OCV is due to a variety of effects.

2.1.4.1 Static polarization

Static polarization is a term that captures static resistances in the battery. This includes the resistance of the electrolyte itself and resistance at the battery connection ports. We may model static polarization simply using a resistor.

2.1.4.2 Dynamic Polarization

Dynamic polarization describes a family of effects caused by the battery chemicals' inertia. To change the internal state of the battery, physical movement by its constituent elements, such as its electrolyte is required. This movement, in the form of diffusion, cannot happen instantaneously. This dynamic behaviour of the battery is readily modeled by one or more resistor-capacitor (RC) sub-circuits, where each sub-circuit models the kinetics of a different chemical process. For a more in-depth look at these processes, see [8]. A small number of RC sub-circuits usually suffices to characterize all present diffusion effect satisfactorily [29]. Modeling dynamic polarization by two RC sub-circuits provides a good trade off between model accuracy and keeping model order reasonably low [7].

2.2 Equivalent Circuit Model

In order to apply the methods of model-based control to any system, one first needs to derive a set of modeling equations, which describe the system with sufficient fidelity. One widespread and extremely versatile way of doing this is using a state-space model. In a state-space model, the internal state of the system is represented by a set of variables called the system's state. The state variables and the inputs to the system are linked by a system of first-order differential equations. The measured outputs of the system depend on the state and input algebraically. Mathematically, we represent this as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{u}), \end{aligned} \tag{2.4}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the input vector, $\mathbf{y} \in \mathbb{R}^p$ is the output vector, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the state transition function, $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ is the output function and $\mathbf{x}_0 \in \mathbb{R}^n$ is the initial state.
We will strive to find a battery model of satisfactory fidelity, which preserves the form given in (2.4).To this end we will explore in detail the family of Equivalent Circuit Models (ECMs) in the following section . We will start by outlining the general modeling approach of this class of models, while describing the advantages and disadvantages thereof. We will then proceed to derive a specific ECM for application in battery estimation in the present thesis.

2.2.1 Modeling approach

Deriving a battery model from first principles leads to a system of partial differential equations (PDEs). Such a model grants precise insights into the chemical and thermal dynamics of a battery and is an effective tool in understanding its inner workings at the most fundamental level. To complement their imposing fidelity, electrochemical models require considerable computational and memory resources to simulate [34]. In order to be able to simulate an electrochemical battery model, however, first, one has to populate them with vast numbers of unknown parameters. Due to their large number of parameters electrochemical models are typically prone to over-fitting problems, when identifying said parameters [7]. In addition to

these challenges, models using PDEs can be quite problematic to use as a basis for designing an estimator with the methods of control theory, especially in the case of nonlinear PDEs as we encounter in battery modeling.
We deem electrochemical battery models unfit as a basis for our SoC and SoH estimation scheme. The main reason for this is because we cannot hope to allot the computational and memory resources needed to power an electrochemical model on the embedded system found in EVs. Not using an electrochemical model also saves us the chore of identifying its myriad parameters and designing a PDE-based estimator.
To circumvent the significant downsides to models consisting of systems of PDEs we turn to lumped models. Lumped models seek to describe dynamical systems originally modeled by PDEs with a finite number of ODEs, while reasonably preserving model fidelity. The most notable class of lumped battery models is called Equivalent Circuit Models (ECMs), which are used to represent the electrical dynamics of batteries. It is important to note, that ECMs do not directly model the dynamic characteristics of battery impedance. This is, instead, captured by state-dependent variations of the electrical circuit parameters [7].
All ingredients necessary for constructing an ECM have been presented in section 2.1, where we specifically introduced lumped electrical components to model complex electrochemical processes inside the battery.

2.2.2 Model Derivation

In order to derive an ECM we first have to choose a model complexity. This means, that we have to choose a number of RC sub-circuits sufficient to model the battery's dynamic behavior. As hinted before, we choose to use two RC sub-circuits as a good trade off between fidelity and complexity of the model. In doing so we follow the example of [15, 1, 33, 12]. Other sensible choices within the class of ECMs are possible. For an excellent overview of sensible model choices, see [7].
Figure 2.2 depicts an equivalent circuit in the style of ECM, which represents the electrical sub-component of the battery. As reasoned above, the model employs 2 RC sub-circuits to model the kinetics of the battery's internal chemical components, such as diffusion of the electrolyte. These effects were explored under the term of dynamic polarization in section 2.1.4.2. The electric parameters of these RC sub-circuits are named R_1, C_1 and R_2, C_2 respectively. The voltage drop over these branches is termed V_1 and V_2, respectively.
We model the battery's internal resistance, also called static polarization, using a resistor with resistance R_0. We call the voltage drop at this resistor V_0.
We model the battery's supply of OCV as a voltage source. This voltage source supplies the SoC- dependent voltage $V_{\mathrm{OC}}(z)$. The function $V_{\mathrm{OC}}(\cdot)$ is experimentally determined in lab tests.
The system's terminal voltage is represented by the variable V_t. This is the voltage measured across the battery's terminals, which is used to power external loads.
The current I supplied to the battery is defined to be positive during charging and negative during discharging.
We emphasize that this equivalent circuit does not in itself model the battery

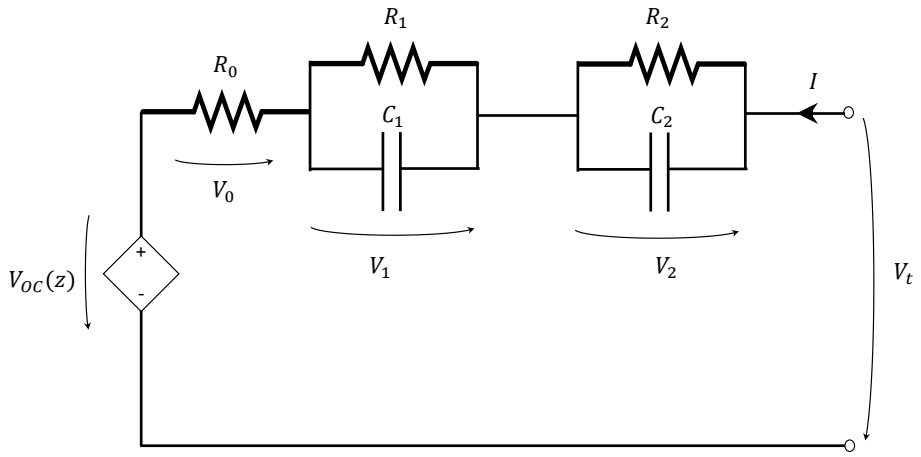

Figure 2.2: This Figure shows the equivalent circuit used by the ECM to model the battery. The ECM uses 2 RC sub-circuits to model chemical mechanisms inside the battery. Arrows pointing from positive to negative potential are used to indicate voltages.

impedance dynamics. We aim to capture the impedance dynamics in the form of time varying parameters R_0, R_1, C_1, R_2, C_2. We do not, however, derive an explicit model for the change of said parameters. We instead assume the impedance dynamics of the system to evolve more slowly than its electric dynamics and perform parametric estimation to identify the parameters. This assumption of different time scales of the electric states and impedance parameters of the system is widely made in the literature [20, 23, 12].
We are now in a position to derive state space equations to describe the model in Figure 2.2.
Using Kirchhoff's circuit laws we can capture the behavior at the RC sub-circuits with the equations

$$\begin{aligned} \dot{V}_1(t) &= -\frac{1}{R_1(t)C_1(t)}V_1(t) + \frac{1}{C_1(t)}I(t), \quad V_1(0) = V_{1,0} \\ \dot{V}_2(t) &= -\frac{1}{R_2(t)C_2(t)}V_2(t) + \frac{1}{C_2(t)}I(t), \quad V_2(0) = V_{2,0}. \end{aligned} \tag{2.5}$$

Further, we find the equation

$$V_t(t) = R_0(t)I(t) + V_1(t) + V_2(t) + V_{\mathrm{OC}}(z(t)) \tag{2.6}$$

to describe the systems measured terminal voltage. As motivated above, we model the parameters R_0, R_1, C_1, R_2, C_2, which vary with temperature, SoC and battery current, as time varying in (2.5) and (2.6).

For easy representation, we concatenate the parameters of the ECM into the vector

$$\theta(t) = \begin{bmatrix} R_0(t) \\ R_1(t) \\ \tau_1(t) \\ R_2(t) \\ \tau_2(t) \end{bmatrix}, \tag{2.7}$$

where $\tau_i(t) = R_i(t)C_i(t)$ are the time constants of the RC-branches.
SoC is best modeled as an integrator, where the battery's input current is integrated. This leads to

$$z(t) = \frac{\eta}{Q} I(t), \quad z(0) = z_0, \tag{2.8}$$

where η is the battery's coulombic efficiency. It is important to note, that this equation describes the dynamics of the SoC only within the interval of $z \in [0, 1]$

We use $\mathbf{x}(t) = \begin{bmatrix} V_1(t) \\ V_2(t) \\ z(t) \end{bmatrix}$ as the state, $u(t) = I(t)$ as the input and $y(t) = V_t(t)$ as the output. For brevity of notation we omit the time argument of state, input, output and parameters, hereinafter. We find a state space representation of the form

$$\begin{aligned} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u; \theta) &= \underbrace{\begin{bmatrix} -\frac{1}{\theta_3} & 0 & 0 \\ 0 & -\frac{1}{\theta_5} & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{:=\mathbf{A}_c} \mathbf{x} + \underbrace{\begin{bmatrix} \frac{\theta_2}{\theta_3} \\ \frac{\theta_4}{\theta_5} \\ \frac{\eta}{Q} \end{bmatrix}}_{:=\mathbf{B}_c} u, \quad \mathbf{x}(0) = \begin{bmatrix} V_{1,0} \\ V_{2,0} \\ z_0 \end{bmatrix} \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, u; \theta) &= \theta_1 u + x_1 + x_2 + V_{\mathrm{OC}}(x_3). \end{aligned} \tag{2.9}$$

2.2.2.1 Model Discretization

In the context of data processing and filtering it is convenient to have a discrete representation of one's model. Especially Kalman filtering, the mode of state estimation we will avail ourselves of in this thesis, depends on having a discrete state space model. Therefore, we discretize the model given in (2.9). We use the method of zero-order-hold discretization to discretize the model (2.9).
We choose some samll sample time h and assume constant input during this sampling time. We plug this into the state space's solution from time $t = kh$ to $t = (k + 1)h$,

with $k \in \mathbb{N}$. Let $x(kh) := x_k$ and $u(kh) = u_k = \text{const.}$. We obtain

$$x_{k+1} = e^{\mathbf{A}_c h} x_k + \int_{kh}^{(k+1)h} e^{\mathbf{A}_c((k+1)h-\tau)} \mathrm{d}\tau u_k \mathbf{B}_c. \tag{2.10}$$

We plug in $\mathbf{A}_c$ and $\mathbf{B}_c$ from (2.9) to get a process equation of the form

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k, \quad \mathbf{x}_0 = \begin{bmatrix} V_{1,0} \\ V_{2,0} \\ z_0 \end{bmatrix}, \tag{2.11}$$

where

$$A = \begin{bmatrix} \exp\left(-\frac{h}{\theta_3}\right) & 0 & 0 \\ 0 & \exp\left(-\frac{h}{\theta_5}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

$$B = \begin{bmatrix} \theta_2(1 - \exp\left(-\frac{h}{\theta_3}\right)) \\ \theta_4(1 - \exp\left(-\frac{h}{\theta_5}\right)) \\ \frac{\eta h}{Q} \end{bmatrix}. \tag{2.13}$$

We leave our output equation unchanged to reach the discrete state space representation

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u, \quad \mathbf{x}_0 = \begin{bmatrix} V_{1,0} \\ V_{2,0} \\ z_0 \end{bmatrix} \tag{2.14}$$

$$y_k = \theta_1 u + x_{k,1} + x_{k,2} + V_{\mathrm{OC}}(x_{k,3}),$$

where y_k is the output, that is the terminal voltage, at time $t = kh$. With this we have reached a form of model with which we can work within the framework of Kalman filtering. Before starting any kind of observer design, however, we analyze our derived model.

2.2.3 Observability Analysis

As we have motivated in section 2.1, its is our stated goal to estimate the internal state of the battery as represented in our state space model from (2.14). The field

of observability analysis within control theory is concerned with whether a system's internal states can be inferred from its external inputs and outputs. This property is called observability of a system. Since the bulk of literature on observability describes continuous systems, we analyze the system given in (2.9) and assume sufficient quality of discretization so that the results of our observability analysis apply even when we deal with the system in the time-discrete characterization we have derived.

Consider nonlinear systems of the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{g}_i(\mathbf{x}) u_i, \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}), \end{aligned} \tag{2.15}$$

where $\mathbf{x} \in \mathcal{X}$ is the state, $u_i \in \mathbb{R}$ is the input, $\mathbf{y} \in \mathbb{R}^p$ is the output, $\mathbf{f} : \mathcal{X} \to \mathbb{R}^n$, $\mathbf{g_i} : \mathcal{X} \to \mathbb{R}^n$ and $\mathbf{h} : \mathcal{X} \to \mathbb{R}^p$ are all smooth functions.

We first define local observability.

Definition 1 *Consider system* (2.15) *and two states* $\mathbf{x_1}$ *and* $\mathbf{x_2}$*. Denote the system output at time* t *with initial state* $\mathbf{x_i}$ *and input* $\mathbf{u}$ *as* $\mathbf{y}(\mathbf{x_i}, \mathbf{u}, t)$*, where* $i = 1, 2$*.* $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *are said to be distinguishable e if there exists an input function* $\mathbf{u}$ *such that* $\mathbf{y}(\mathbf{x_1}, \mathbf{u}, t) \neq \mathbf{y}(\mathbf{x_2}, \mathbf{u}, t)$ *for a finite* t*. System 2.15 is locally observable at* $\mathbf{x}_1$*, if there exists a neighborhood* $\mathcal{N}$ *of* $\mathbf{x}_1$*, such that the only state in* $\mathcal{N}$ *that is not distinguishable from* $\mathbf{x}_1$ *is* $\mathbf{x}_1$ *[31].*

We now have the language to understand the following theorem on local observability:

Theorem 1 *Consider the system described by* (2.15)*, and suppose* $\mathbf{x}_0 \in \mathbb{X}$ *is given. Consider the forms*

$$(\mathrm{d}L_{z_s} L_{z_{s-1}} \cdots L_{z_1} h_j)(\mathbf{x}_0), s \geq 0, z_i \in \{\mathbf{f}, g_1, \cdots, g_m\}, \tag{2.16}$$

evaluated at $\mathbf{x}_0$*. Suppose there are* n *linearly independent row vectors in this set. Then the system is locally observable around* $\mathbf{x}_0$ *[31].*

To evaluate the condition given in (2.16) we compute the needed gradients of the Lie derivatives. We arrive at

$$\begin{aligned} \mathrm{d}h &= \begin{bmatrix} 1 & 1 & \dfrac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z} \end{bmatrix} \\ \mathrm{d}L_f^k h &= \begin{bmatrix} \dfrac{1}{(-\tau_1)^k} & \dfrac{1}{(-\tau_2)^k} & 0 \end{bmatrix} \\ \mathrm{d}L_g^k h &= \begin{bmatrix} 0 & 0 & \dfrac{1}{Q^k} \dfrac{\mathrm{d}^{k+1} V_{\mathrm{OC}}}{\mathrm{d}^{k+1} z} \end{bmatrix}, \end{aligned} \tag{2.17}$$

where $k \in \mathbb{Z}^+$. Lie derivatives along both f and g are constants and can be ignored since they give gradients of rank 0 (i.e. $L_g L_f h = 0$). In accordance with theorem 1

we examine the matrix

$$\mathcal{O} = \begin{bmatrix} \mathrm{d}h \\ \mathrm{d}L_f h \\ \mathrm{d}L_g h \\ \mathrm{d}L_f^2 h \\ \mathrm{d}L_g^2 h \\ \vdots \end{bmatrix} \tag{2.18}$$

for column rank. Plugging in (2.17), we see that $\mathcal{O}$ has full column rank, iff

$$\frac{\mathrm{d}^k V_{\mathrm{OC}}}{\mathrm{d}z}(\mathbf{x}_0) \neq 0 \tag{2.19}$$

holds for at least one $k \in \mathbb{Z}^+$. We remark that we also require $\tau_1 \neq \tau_2$. This is considered as given, since τ_1 and τ_2 model different chemical processes. In case $\tau_1 = \tau_2$ the first two states both model the same chemical process, which makes them indistinguishable.
We conclude that the battery model is locally observable if all derivatives of $V_{\mathrm{OC}}(z)$ are not zero at the same point in the state space. For a deeper exploration of the topic of observability of nonlinear systems, see [31]. For a study in the observability of ECM-like battery models, albeit deviating slightly from the modeling conventions this thesis uses, see [35].
We note that the standard criterion of testing the first-order Taylor approximation of the system for observability leads to the observability matrix

$$P = \begin{bmatrix} 1 & 1 & \frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z} \\ -\frac{1}{\tau_1} & -\frac{1}{\tau_2} & 0 \\ \frac{1}{\tau_1^2} & -\frac{1}{\tau_2^2} & 0 \end{bmatrix}. \tag{2.20}$$

This matrix only has full rank when $\frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z} \neq 0$.
Figure 2.3 shows a plot of $\frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z}$ at different temperatures. We observe that $\frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z}$ never becomes zero, regardless of temperature or SoC. With this we regard the observability of the ECM as established for all observer types. We note, however, that (2.19) is less restrictive as an observability criterion. Therefore, higher order observers like the unscented Kalman filter have an advantage in observability over first order observers like the extended Kalman filter.
The prerequisites for state observation of this model are given. Of course, in order to be able to run this model, we need to populate it with appropriate parameters.

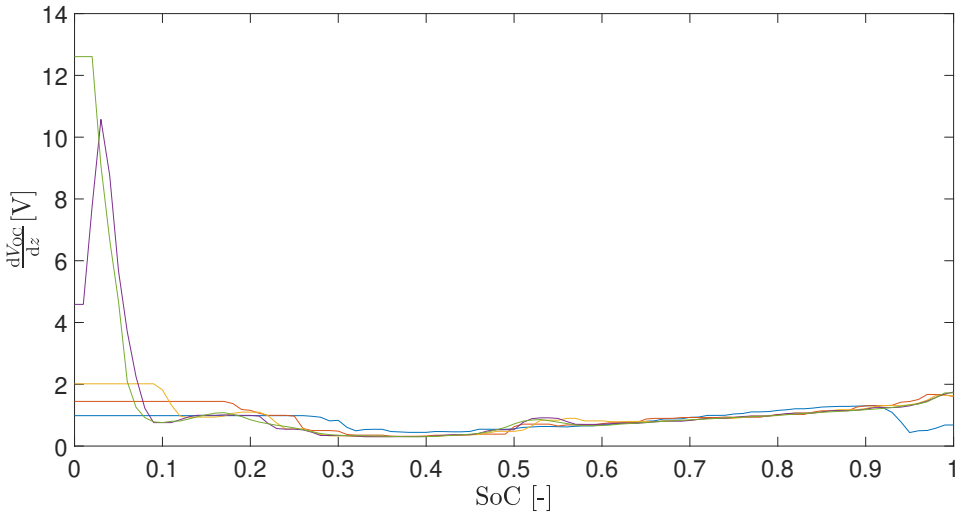

Figure 2.3: This Figure shows a plot of $\frac{\mathrm{d}V_{\mathrm{OC}}}{\mathrm{d}z}$ for the battery used for experimental validation in this thesis at different temperatures.

2.2.4 Sensitivity Analysis

Sensitivity analysis concerns itself with how well parameters, as estimated by a parameter filter, can be identified from a given set of data. Using the methods of classical observability analysis in this context does not present itself as practical [33]. The parameters dynamics themselves are not modeled in the approach chosen in this thesis, making observability analysis of the parameters infeasible.
Sensitivity analysis, in turn, has developed as a more practical approach to improving dual estimation schemes. Given a continuous stream of measurement data, sensitivity analysis tries to identify information-rich segments of this data for use in parameter estimation. Poor data in terms of identifiabilty of the parameters is identified and discarded.
One approach recently charted in battery estimation is deriving filter equations, which allow for the online computation of the partial derivatives of the states with respect to the parameters $\frac{\partial V}{\partial \theta}$ [13]. This quantity is called data sensitivity. Use of these partial derivatives can be motivated using a more general measure of estimation accuracy called the Cramer-Rao bound. For further information, see [13, 12].
To compute analytic expressions for the sensitivities we take the discrete version of the state equations

$$V_k = \exp(-\frac{h}{\tau})V_{k-1} + R(1 - \exp(-\frac{h}{\tau}))u_{k-1}, \tag{2.21}$$

governing the voltages at the RC sub-circuits. We compute the Z-transform of this equation, which leads to

$$V(z) = \frac{R(1 - \exp(-\frac{h}{\tau}))z^{-1}}{1 - z^{-1}\exp(-\frac{h}{\tau})}U(z). \tag{2.22}$$

Finally, by computing the partial derivatives of this transfer function, we find filter equations to compute the required sensitivities in real time. The equations are

$$\begin{aligned} S_{R_i}(z) &= \frac{\partial V}{\partial R}(z) = \frac{(1-\exp(-\frac{h}{\tau}))z^{-1}}{1-z^{-1}\exp(-\frac{h}{\tau})}U(z) \\ S_{\tau_i}(z) &= \frac{\partial V}{\partial \tau}(z) = R\frac{h}{\tau^2}\exp(-\frac{h}{\tau})\frac{z^{-2}-z^{-1}}{(1-z^{-1}\exp(-\frac{h}{\tau}))^2}U(z). \end{aligned} \tag{2.23}$$

Thus, we have derived filtering equations to describe the model's sensitivity to the parameters characterizing the diffusion processes taking place inside the battery. However, we will also estimate the static resistance R_0, which does not appear in the process equations. Therefore, the above approach is not applicable to estimating the sensitivity of R_0.
Thankfully, a much simpler but effective way to estimate sensitivity with respect to R_0 exists. We point to the fact that R_0 is the only parameter to feature in the output equation. Interpreting it, for argument's sake, as an input, we see that it has relative degree 0, whereas the "inputs" R_1, R_2, τ_1 and τ_2 all have relative degree 1. Another way of saying this is that R_0 is the only parameter with direct feedthrough. This motivates the following tactic: Whenever the system input changes rapidly, before the other parameters "have time" to influence the system output, we can estimate R_0 well from the output. Thus, sensitivity towards R_0 is high, whenever

$$S_{R_0,k} = u_k - u_{k-1} \tag{2.24}$$

is high.
With this, we have arrived at a set of equations which take the system input and compute sensitivity of the states to parameter changes based on this data. We use this information to improve parameter estimation.
We now have a firm grasp of the properties of the model we use as a basis to implement our state and parameter estimation scheme. In the next section, we proceed to introduce algorithms suited to carry out the imposing task of dual state and parameter estimation.

2.3 Estimation Algorithms

Any causal system can be represented by a function that computes the system's output from its past and present inputs. For a large class of systems we can define a variable to sum up all past inputs into one array. This variable is called the system's state. Thus, system output can be computed solely based on the present input and state. Since the state comprises a history of all past states it is generally not measurable at the present time. For many applications we desire knowledge of the state. The field of estimation theory provides methods of estimating hidden states, given uncertain knowledge of the observed process and noisy measurements thereof.

For example, this thesis has the goal of estimating the internal state of a battery, given a model that approximately describes the processes inside the battery and imperfect measurements. The following derivation of Kalman filtering algorithms from a more general probabilistic inference framework has been adapted from [23].

2.3.1 Background of Kalman filtering

To explore the methods of estimation theory we look at a general system in discrete-time state-space form

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}, k-1) \tag{2.25}$$
$$y_k = h(x_k, u_k, v_k, k), \tag{2.26}$$

where k is the time index, $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state at time k, $u_k \in \mathbb{R}^p$ is the input at time k, $y_k \in \mathbb{R}^q$ is the output at time k. (2.25) is called the process equation. (2.26) is called the measurement equation. w_k and v_k are stochastic variables called the process noise and measurement noise, respectively. w_k models deviation of the process equation from the actual real world process. v_k represents errors in the sensor used to observe the system.

We conclude from this system representation that the relationship between the states, inputs and outputs at different times cannot be expressed in deterministic way. Instead, we use the language of conditional probability. For example, $p(x_k|x_{k-1})$ is the probability of x_k after x_{k-1}. This probability explicitly depends on the stochastic variable w_k.

Our goal now is to use the available information of all past and present measurements $\mathbb{Y}_k = \{y_0, y_1, \cdots, y_k\}$ to give the best possible estimate of the present state $\hat{x}_k$. This problem is known as probabilistic inference.

We take the best estimate of the state to be the expected value of the state, given all past measurements. We may compute this by the conditional mean

$$\hat{x}_k = \mathbb{E}[x_k|\mathbb{Y}_{k-1}] = \int_{\mathcal{X}} x_k p(x_k|\mathbb{Y}_k)\mathrm{d}x_k, \tag{2.27}$$

where $\mathbb{E}[\cdot]$ is the statistical expectation operator. In Bayesian inference we recursively compute the posterior probability density $p(x_k|\mathbb{Y}_k)$ in two steps:

Step 1: Prediction

During the prediction step we use the past posterior probability density function and our knowledge of the process to predict the present probability density

$$p(x_k|\mathbb{Y}_k) = \int_{\mathcal{X}} p(x_k|x_{k-1})p(x_{k-1}|\mathbb{Y}_{k-1})\mathrm{d}x_{k-1}, \tag{2.28}$$

given only past measurements.

Step 2: Correction

We then correct our prediction using Bayes' rule by

$$p(x_k|\mathbb{Y}_k) = \frac{p(y_k|x_k)p(x_k|\mathbb{Y}_{k-1})}{p(y_k|\mathbb{Y}_{k-1})}, \tag{2.29}$$

which assumes a conditionally independent series of measurements y_i given x_k. This is satisfied by any system where any state is exclusively dependent on the state that immediately preceded it, also known as the Markov property. Computing the constituent elements of equation (2.29) is only a question of utilizing one's knowledge of the process and measurement equations, like so:

$$p(y_k|\mathbb{Y}_{k-1}) = \int_{\mathcal{X}} p(y_k|x_k)p(x_k|\mathbb{Y}_{k-1})\mathrm{d}x_k \tag{2.30}$$

$$p(x_k|x_{k-1}) = \sum_{w:x_k=f(x_{k-1},u_{k-1},w_{k-1},k-1)} p(w) \tag{2.31}$$

$$p(y_k|x_k) = \sum_{v:y_k=h(x_k,u_k,v_k,k)} p(v). \tag{2.32}$$

Finding a closed-form solution to this is outside the realm of feasibility for nearly all real world applications. This fact renders the algorithm in its present form and generality unusable for our purposes.
The algorithms we will use in this thesis can all be derived from this general framework by making some simplifying assumptions, in order to make implementing them in real world applications more tractable. One very natural such assumption is the assumption of Gaussian probability density functions. In fact, it is this assumption that lies at the heart of the family of algorithms called Kalman filtering. A Gaussian probability density function can be fully characterized by only its mean value and covariance. Under this assumption we no longer have to propagate the probability density functions using integrals and instead end up with a much simplified general Kalman filtering framework, that revolutionizes the way sequential probabilistic inference can be implemented. The Gaussian recursion equations read

$$\hat{x}_k^+ = \hat{x}_k^- + L_k(y_k - \hat{y}_k) = \hat{x}_k^- + L_k(\tilde{y}_k) \tag{2.33}$$

$$\Sigma_{\tilde{x},k}^+ = \Sigma_{\tilde{x},k}^- + L_k\Sigma_{\tilde{y},k}L_k^T, \tag{2.34}$$

where

$$\hat{x}_k^+ = \mathbb{E}[x_k|\mathbb{Y}_k] \tag{2.35}$$

$$\hat{x}_k^- = \mathbb{E}[x_k|\mathbb{Y}_{k-1}] \tag{2.36}$$

$$\hat{y}_k = \mathbb{E}[y_k|\mathbb{Y}_{k-1}] \tag{2.37}$$

$$\Sigma_{\tilde{x},k}^- = \mathbb{E}[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] = \mathbb{E}[(\tilde{x}_k^-)(\tilde{x}_k^-)^T] \tag{2.38}$$

$$\Sigma_{\tilde{x},k}^+ = \mathbb{E}[(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T] = \mathbb{E}[(\tilde{x}_k^+)(\tilde{x}_k^+)^T] \tag{2.39}$$

$$\Sigma_{\tilde{y},k} = \mathbb{E}[(y_k - \hat{y}_k)(y_k - \hat{y}_k)^T] = \mathbb{E}[(\tilde{y}_k)(\tilde{y}_k)^T] \tag{2.40}$$

$$L_k = \mathbb{E}[(x_k - \hat{x}_k^-)(y_k - \hat{y}_k)^T]\Sigma_{\tilde{y},k}^{-1} = \Sigma_{\tilde{x},\tilde{y}}^-\Sigma_{\tilde{y},k}^{-1}. \tag{2.41}$$

Here, a circumflex indicates an estimated quantity, the superscripts "+" and "-" represent a posteriori and a priori estimates respectively and a tilde represents the deviation of a quantity. Using these equations we can compile a set of 6 general steps to implement Kalman filtering:

Step 1: State prediction

We compute an estimation of the present state $\hat{x}_k^-$ by using our knowledge of

the process equation (2.25), and computing the state's expected value using (2.36). This is done using all previous measurements $\mathbb{Y}_{k-1}$, hence the terms prediction and a priori estimate are applied to this step. Thus, we obtain the form

$$\hat{x}_k^- = \mathbb{E}[x_k|\mathbb{Y}_{k-1}] = \mathbb{E}[f(x_{k-1}, u_{k-1}, w_{k-1}, k-1)|\mathbb{Y}_{k-1}] \tag{2.42}$$

Step 2: State covariance prediction
Based on our predicted state, we compute an a priori estimate of the error covariance matrix. This is done using estimation (2.38) and using our knowledge of the process equations (2.25).
Step 3: Output prediction
We can now predict an estimate of the system output by plugging the above a priori information into (2.26) and (2.37), hence obtaining

$$\hat{y}_k = \mathbb{E}[y_k|\mathbb{Y}_{k-1}] = \mathbb{E}[h(x_k, u_k, v_k, k)|\mathbb{Y}_{k-1}]. \tag{2.43}$$

Step 4: Compute estimator gain L_k
We can now compute the estimator gain or Kalman gain L_k using equation (2.41).
Step 5: State correction
In this step we use the estimator gain L_k and the output prediction error $\tilde{y}_k = y_k - \hat{y}_k$ to correct out a priori state estimate with the help of (2.33).
Step 6: State covariance correction
Finally, using the a posterori state estimate $\hat{x}_k^+$ and equation (2.34), we calculate a corrected estimate of the error covariance matrix $\hat{\Sigma}_{\tilde{x},k}^+$. With this we have computed a posteriori estimates for the state and error covariance of the system. This, under the assumption of Gaussian noise is equivalent to propagating the probability density function using integrals, where no Gaussian noise can be assumed. We now wait for the next sampling interval, update k and repeat the process from step 1.

In the following sections, more specific forms of this general Kalman filtering algorithm is derived by plugging in further assumptions on the process and the way it is impacted by stochastic disturbances. In doing this, we establish a continuous line of reasoning between probabilistic inference and the Kalman filtering algorithms used in this thesis. These approaches to Kalman filtering mainly differ in their ways of calculating the statistical variables given in (2.35) - (2.41), that is in their way of computing expected values and error covariance matrices.

2.3.2 Unscented Kalman filtering

The unscented Kalman filter (UKF) is a specialization of Kalman filtering, which approximates the mean values and covariance matrices in (2.35)-(2.41) by using repeated function evaluations. This approach of repeated function evaluation is more generally utilized by a class of filters called particle filters. In particle filters, a large number of stochastically distributed, possible states (particles) around the previous best estimate is propagated using the process equations. The set of propagated particles is then used to approximate the statistical quantities in (2.35)-(2.41), with

a larger number of particles leading to a better approximation of the mean values and covariance matrices. The big advantage of this approach is that no direct assumptions are made on the structure of the probability density functions that appear in the problem. For this reason, particle filtering can even be applied to processes affected by non-Gaussian noise. Because of their need for vast numbers of particles, however, particle filters tend to be very computationally intensive and thus infeasible for the bulk of practical applications.
The UKF makes use of the assumption of Gaussian noise to vastly reduce the number of particles needed in order to obtain good estimates of the state. This is done by using the information about the probability density function contained in the covariance matrix to deterministically choose a small number of highly representative particles called the sigma points. The UKF thus belongs to the class of Kalman filters known as sigma-point Kalman filters, with the UKF distinguishing itself as using the so-called unscented transform, in order to choose the sigma points. We will explore two key concepts of UKF before we are ready to apply
The most basic step of UKF is choosing the array of particles or sigma points. Taking some mean value $\bar{x} \in \mathbb{R}^n$ and covariance $\Sigma_{\tilde{x}}$ we take the set

$$\mathcal{X} = \left\{ \bar{x}, \bar{x} + \gamma\sqrt{\Sigma_{\tilde{x}}}, \bar{x} - \gamma\sqrt{\Sigma_{\tilde{x}}} \right\}, \tag{2.44}$$

of $p + 1 = 2L + 1$ sigma points indexed from 0 to p, where γ is a weighing factor. The matrix square root can be efficiently computed numerically as a lower triangular matrix using Cholesky factorization [23]. This constitutes one of the most vulnerable parts of the entire algorithm, as calculating the matrix square root of covariance matrices with potentially very small entries is numerically susceptible.
Conversely, we can calculate the mean

$$\bar{x} = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{X}_i \tag{2.45}$$

and covariance

$$\Sigma_{\hat{x}} = \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_i - \bar{x})(\mathcal{X}_i - \bar{x})^T, \tag{2.46}$$

where $\alpha_i^{(m)}$ and $\alpha_i^{(c)}$ are scalar weighing parameters. The values for the design parameters γ, $\alpha_i^{(m)}$ and $\alpha_i^{(c)}$ can be taken from Table 2.1. It can be easily verified that using these parameters, the backtransformations in (2.45) and (2.46) hold.
Applying UKF to an estimation problem we consider an augmented state vector x^a and an augmented covariance matrix Σ_x^a. This serves to consider information about the process and measurement noise explicitly, as well as information about the state. These augmented variables are defined by

$$x_k^a = \begin{bmatrix} x_k \\ \bar{w} \\ \bar{v} \end{bmatrix}, \tag{2.47}$$

	γ	$\alpha_0^{(m)}$	$\alpha_i^{(m)}$	$\alpha_0^{(c)}$	$\alpha_i^{(c)}$
UKF	$\sqrt{L+\lambda}$	$\frac{\lambda}{L+\lambda}$	$\frac{1}{2(L+\lambda)}$	$\frac{\lambda}{L+\lambda}+(3-a^2)$	$\frac{1}{2(L+\lambda)}$

Table 2.1: This table shows the design parameters used to create a UKF. Here,

$$\lambda = a^2(L+\kappa) - L$$

is a scaling parameter, where

$$10^{-2} \leq a \leq 1$$

and κ is chosen as 0 or $3-L$. We note

$$\sum_{i=0}^{p} \alpha_i^{(m)} = 1$$

but

$$\sum_{i=0}^{p} \alpha_i^{(c)} \neq 1.$$

We conclude that the main design parameter in this procedure is a.

where $\bar{w}$ and $\bar{v}$ are the mean values of the process noise and measurement noise, respectively and

$$\Sigma_{\tilde{x},k}^{a} = \text{diag}(\Sigma_{\tilde{x},k}, \Sigma_w, \Sigma_v). \tag{2.48}$$

Armed with the knowledge about the unscented transform and augmented state and covariance, we now set about adapting the general Kalman filtering algorithm as presented in 2.3.1. The steps in applying UKF look as follows:

Step 1: State prediction

We first calculate the set of $p+1$ sigma points to propagate, according to (2.44). This leads to

$$\mathcal{X}_{k-1}^{a,+} = \left\{ \hat{x}_{k-1}^{a,+}, \hat{x}_{k-1}^{a,+} + \gamma\sqrt{\hat{\Sigma}_{\tilde{x},k-1}^{a,+}}, \hat{x}_{k-1}^{a,+} - \gamma\sqrt{\hat{\Sigma}_{\tilde{x},k-1}^{a,+}} \right\}. \tag{2.49}$$

This is a critical step in UKF, as computing the matrix square root of the augmented covariance matrix $\sqrt{\hat{\Sigma}_{\tilde{x},k-1}^{a,+}}$ is computationally intensive and vulnerable for badly conditioned matrices, even when done with the effective method of Cholesky factorization. From this set we extract the portions describing the state and term them $\mathcal{X}_{k-1,i}^{x,+}$, and the portions describing the noise and term them $\mathcal{X}_{k-1,i}^{w,+}$ and $\mathcal{X}_{k-1,i}^{v,+}$. Here, the subscript i refers to the index of the original sigma point. We now propagate these $p+1$ sigma points using the process equations (2.25) to obtain the set of a priori sigma points

$$\mathcal{X}_{k,i}^{x,-} = f(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, \mathcal{X}_{k-1,i}^{w,+}, k-1). \tag{2.50}$$

Subsequently, we can approximate (2.36) using the weighted mean

$$\hat{x}_k^- = \mathbb{E}[f(x_{k-1}, u_{k-1}, w_{k-1}, k-1)|\mathbb{Y}_{k-1}] \tag{2.51}$$

$$\approx \sum_{i=0}^{p} \alpha_i^{(m)} f(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, w_{k-1}, \mathcal{X}_{k-1,i}^{w,+}, k-1) = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{X}_{k,i}^{x,-}. \tag{2.52}$$

This constitutes an approximation, since the weighted mean does not necessarily coincide with the true expected value of the state in the sense of (2.27). With this, we obtain our a priori state estimate.

Step 2: State covariance prediction

We use the set of a priori sigma points to calculate an a priori estimate of the error covariance. Thus, we calculate the predicted error covariance by

$$\hat{\Sigma}_{\tilde{x},k}^- = \Sigma_w + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)^T. \tag{2.53}$$

Step 3: Output prediction

We predict the system output by evaluating the measurement equation (2.26) for all predicted sigma points $\mathcal{X}_{k,i}^{x,-}$ to obtain the set of predicted outputs

$$\mathcal{Y}_{k,i} = h(\mathcal{X}_{k,i}^{x,-}, u_k, \mathcal{X}_{k-1,i}^{v,+}, k). \tag{2.54}$$

By plugging these points in a weighted sum, we reach an overall prediction

$$\hat{y}_k \approx \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{Y}_{k,i}. \tag{2.55}$$

of the systems output. This is again an approximation due to substituting a weighted sum of sigma points for a true expected value.

Step 4: Compute estimator gain L_k

We first use the predicted sigma points and outputs to compute the covariance matrices

$$\hat{\Sigma}_{\tilde{y},k} = \Sigma_v + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{Y}_{k,i} - \hat{y}_k)(\mathcal{Y}_{k,i} - \hat{y}_k)^T \tag{2.56}$$

and

$$\hat{\Sigma}_{\tilde{x},\tilde{y},k}^- = \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{Y}_{k,i} - \hat{y}_k)^T \tag{2.57}$$

required in (2.41). Plugging this into (2.41), we can compute the Kalman gain L_k.

Step 5: State correction

This step remains unchanged. We simply correct the predicted state using

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \tilde{y}_k \tag{2.58}$$

exactly as in (2.33)

Step 6: State covariance correction

The last step remains equally unchanged. We insert our results up until now into (2.34) to get our corrected estimate of the error covariance

$$\hat{\Sigma}_{\tilde{x},k}^+ = \hat{\Sigma}_{\tilde{x},k}^- - L_k \hat{\Sigma}_{\tilde{y},k} L_k^T. \tag{2.59}$$

With this we have derived a highly efficient version of the general Kalman filtering algorithm presented in 2.3.1 for use on nonlinear systems. The accuracy of this filter is largely only limited by how representative the particular sigma points are of the true probability density functions, as this is the only place we have conceded approximations in the general Kalman filtering algorithm.

2.3.3 Extended Kalman filtering

Extended Kalman filtering is the most widely used variety of the Kalman filtering. In many fields, it is employed as a quasi-standard for internal state observation of nonlinear processes [20]. EKF finds a comparatively easy formulation of the Kalman filtering framework by making some assumptions and approximations in addition to the ones we have already made.
In particular, the EKF assumes both the process equations and the measurement equations to be "close to linear", in the sense that it should be possible to approximate these functions well using a first order Taylor expansion at any given point in the state-space. Another way of saying this is that the system has no "hard" nonlinearities. If this holds true, the system can be reasonably approximated using a time-varying linear system. If this is a good approximation, notably, we obtain

$$\mathbb{E}[f(\cdot)] = f(\mathbb{E}[\cdot]) \tag{2.60}$$

and

$$\mathbb{E}[h(\cdot)] = h(\mathbb{E}[\cdot]), \tag{2.61}$$

which tremendously simplifies handling (2.35)-(2.41).
We see the specific instances of these approximations in the adapted steps 1 and 3 of the EKF framework, where we make approximations in the style of (2.60) and (2.61) and in steps 2 and 4 where approximate nonlinear equations by a first order Taylor approximation around the current operating point.
Using the EKF algorithm reduces the Kalman filtering framework to the following:

Step 1: State prediction
We produce an a priori state estimate by approximating (2.36) with

$$\hat{x}_k^- = \mathbb{E}[f(x_{k-1}, u_{k-1}, w_{k-1}, k-1)|\mathbb{Y}_{k-1}] \approx f(\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}, k-1), \tag{2.62}$$

where $\bar{w}_{k-1} = \mathbb{E}[w_{k-1}] = 0$ in the case of white noise. This is an approximation as presented in (2.60). In comparison to UKF, this is the same as to only propagate the central sigma point and neglect all others, giving a worse a priori state estimate.

Step 2: State covariance prediction
We calculate

$$\hat{\Sigma}_{\tilde{x},k}^- = \mathbb{E}[(\tilde{x}_k^-)(\tilde{x}_k^-)^T] \tag{2.63}$$

by finding an estimate of $\tilde{x}_k^-$. To that end we start from the definition

$$\tilde{x}_k^- = x_k - \hat{x}_k^- = f(x_{k-1}, u_{k-1}, w_{k-1}, k-1) - f(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}, k-1) \tag{2.64}$$

and perform a first-order Taylor expansion of the second term around the current operating point $\{x_{k-1}, u_{k-1}, w_{k-1}, k-1\}$ to get

$$\begin{aligned}\hat{x}_k^- =& f(x_{k-1}, u_{k-1}, w_{k-1}, k-1) \\ &+ \underbrace{\left[\frac{\partial f}{\partial x_{k-1}}\right]_{(x_{k-1}=\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}=\bar{w}_{k-1}, k-1)}}_{:=\hat{A}_{k-1}} (\hat{x}_{k-1}^+ - x_{k-1}) \\ &+ \underbrace{\left[\frac{\partial f}{\partial w_{k-1}}\right]_{(x_{k-1}=\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}=\bar{w}_{k-1}, k-1)}}_{:=\hat{B}_{k-1}} (\bar{w}_{k-1} - w_{k-1}).\end{aligned} \tag{2.65}$$

Inserting this into (2.64) we obtain

$$\tilde{x}_k^- \approx \hat{A}_{k-1}\tilde{x}_{k-1} + \hat{B}_{k-1}\tilde{w}_{k-1}. \tag{2.66}$$

This, we can substitute into (2.38), which leads to

$$\hat{\Sigma}_{\tilde{x},k}^- \approx \hat{A}_{k-1}\hat{\Sigma}_{\tilde{x},k-1}^+\hat{A}_{k-1}^T + \hat{B}_{k-1}\Sigma_w\hat{B}_{k-1}. \tag{2.67}$$

Very often in practical applications, we assume additive noise due to not having an explicit model for how process noise affects the system. This is to say we assume

$$f(x_k, u_k, w_k, k) = f(x_k, u_k, k) + w_k, \tag{2.68}$$

whereby the above simplifies to

$$\hat{\Sigma}_{\tilde{x},k}^- \approx \hat{A}_{k-1}\hat{\Sigma}_{\tilde{x},k-1}^+\hat{A}_{k-1}^T + \Sigma_w, \tag{2.69}$$

since

$$\hat{B}_{k-1} = I. \tag{2.70}$$

Step 3: Output prediction
We make approximation (2.61) to get an estimate of the predicted output

$$\hat{y}_k = \mathbb{E}[h(x_k, u_k, v_k, k)|\mathbb{Y}_{k-1}] \approx h(\hat{x}_k^-, u_k, \bar{v}_k, k), \tag{2.71}$$

where in some applications we may assume white noise. Again, in comparison to UKF this corresponds to only evaluating the output function at the central sigma point.

Step 4: Compute estimator gain L_k
In order to calculate the estimator gain matrix, we first need to find the output prediction error

$$\tilde{y}_k = y_k - \hat{y}_k = h(x_k, u_k, v_k, k) - h(\hat{x}_k^-, u_k, \bar{v}_k, k), \tag{2.72}$$

which we can do analogously to step 2 with a first-order Taylor expansion of the second term. We find

$$\begin{aligned}\hat{y}_k =& h(x_k, u_k, v_k, k) \\ &+ \underbrace{\left[\frac{\partial h}{\partial x_k}\right]_{(x_k=\hat{x}_k^-, u_k, v_k=\bar{v}_k, k)}}_{:=\hat{C}_k} (\hat{x}_k^- - x_k) \\ &+ \underbrace{\left[\frac{\partial f}{\partial v_k}\right]_{(x_k=\hat{x}_k^-, u_k, v_k=\bar{v}_k, k)}}_{:=\hat{D}_k} (\bar{v}_k - v_k).\end{aligned} \tag{2.73}$$

from there we can compute the necessary covariance matrices as

$$\hat{\Sigma}_{\tilde{y},k} \approx \hat{C}_k \hat{\Sigma}_{\tilde{x},k}^- \hat{C}_k^T + \hat{D}_k \Sigma_v \hat{D}_k^T, \tag{2.74}$$

or in the case of white noise

$$\hat{\Sigma}_{\tilde{y},k} \approx \hat{C}_k \hat{\Sigma}_{\tilde{x},k}^- \hat{C}_k^T + \Sigma_v, \tag{2.75}$$

and

$$\hat{\Sigma}_{\tilde{x},\tilde{y},k}^- \approx \mathbb{E}[(\tilde{x}_k^-)(\hat{C}_k \tilde{x}_k^- + \hat{D}_k \tilde{v}_k)^T] = \hat{\Sigma}_{\tilde{x},k}^- \hat{C}_k^T. \tag{2.76}$$

The Kalman gain is now calculated as

$$L_k = \hat{\Sigma}_{\tilde{x},\tilde{y},k}^- \hat{\Sigma}_{\tilde{y},k}^{-1} = \hat{\Sigma}_{\tilde{x},k}^- \hat{C}_k^T [\hat{C}_k \hat{\Sigma}_{\tilde{x},k}^- \hat{C}_k^T + \hat{D}_k \Sigma_v \hat{D}_k^T]^{-1} \tag{2.77}$$

Step 5: State correction
The state correction step remains unchanged from the general Kalman filtering framework:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \tilde{y}_k. \tag{2.78}$$

Step 6: State covariance correction
We can plug the above into (2.39) to obtain

$$\begin{aligned}\hat{\Sigma}_{\tilde{x},k}^+ &= \hat{\Sigma}_{\tilde{x},k}^- - L_k \hat{\Sigma}_{\tilde{y},k} L_k^T = \hat{\Sigma}_{\tilde{x},k}^- - L_k \hat{\Sigma}_{\tilde{y},k} (\hat{\Sigma}_{\tilde{y},k})^{-T} (\hat{\Sigma}_{\tilde{x},\tilde{y},k}^-)^T \\ &= \hat{\Sigma}_{\tilde{x},k}^- - L_k \hat{C}_k \hat{\Sigma}_{\tilde{x},k}^- = (I - L_k \hat{C}_k) \hat{\Sigma}_{\tilde{x},k}^-.\end{aligned} \tag{2.79}$$

We have thus derived a very simple Kalman filtering algorithm based in the general framework given in 2.3.1. The validity of this simple, efficient variation of Kalman filtering rests upon the questionable assumptions we made at the beginning of this section. They are the assumption that the process equations (2.25) and the measurement equations (2.26) can be well approximated using linear functions, as well as the assumption that the past corrected state estimate $\hat{x}_{k-1}^+$ lies at the true expected value of the state's probability density function in the sense of (2.27).
Whenever these assumptions do not hold, filter performance of the EKF will suffer, falling behind the performance of filters that do not make these problematic approximations, such as the UKF. On the other hand, whenever these assumptions hold well, EKF performance will rival that of UKF despite being based on a much simpler variation of Kalman filtering.
One notable difficulty in implementing the EKF algorithm is the need to compute the partial derivatives of the process and measurement equations, as presented in steps 2 and 4 of the EKF algorithm.

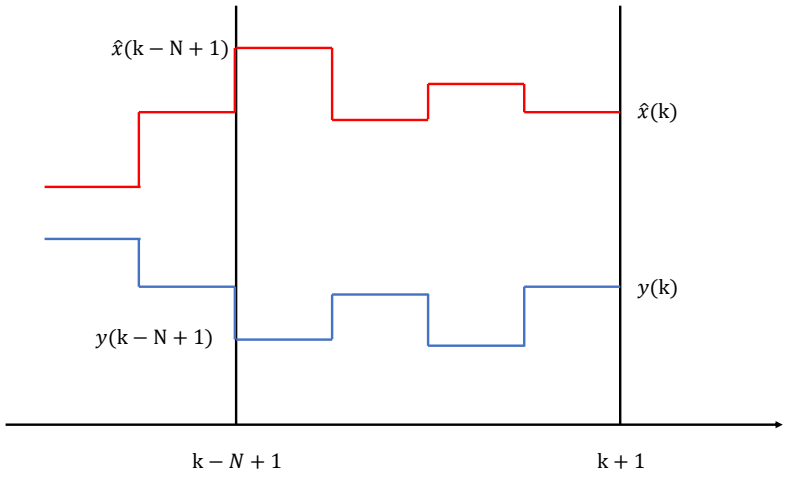

Figure 2.4: This Figure illustrates the principle of MHE. Here, t is the current time and our horizon extends N samples into the past, where all measurements between t and $t - N + 1$ are considered. The state is equally observed in this period.

2.3.4 Moving Horizon Estimation

In Kalman filtering in general, we sum up the entirety of the systems history in the past best estimate $\hat{x}_{k-1}^{+}$ and covariance $\hat{\Sigma}_{\tilde{x},k-1}^{+}$. All information about the system's past trajectory is then surmised from these two variables. Moving horizon estimation (MHE) takes a different approach to this. Instead of considering only one time step of the dynamical system to be observed, we examine the system's trajectory, that is, its explicit history of past inputs and outputs to get an estimate of the present state.

Ideally, we would like to use all past measurements, an approach known as full information estimation. This, however, becomes computationally intractable in real-world applications. We therefore content ourselves with using just a finite number, say N, past measurements. This window of N samples is moved to include the present measurement at every new sampling period, hence the name moving horizon estimation.

Considering this horizon of N samples, we would like to compute an estimate of the present state using the methods of numerical optimization. In the process, we compute a past trajectory of the states to explain the observed measurements. One big advantage of using numerical optimization is that it enables us to systematically include constraints on the variables we estimate. This turns out to be of great utility, as we use MHE to estimate battery parameters, which must lie within a certain valid range.

To be able to solve this problem using optimization, we first need a problem formu-

lation. We assume a nonlinear dynamical system of the form

$$x_k = f(x_{k-1}, w_{k-1}) \tag{2.80}$$
$$y_k = h(x_k) + v_k, \tag{2.81}$$

where k is the time index, $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state at time k, $y_k \in \mathbb{R}^q$ is the output at time k. w_k and v_k are bounded, stochastic noise variables. Using this, we formulate a standard MHE optimization problem

$$\begin{aligned} \min_{x(\cdot|k), w(\cdot|k), v(\cdot|k)} \quad & \sum_{i=k-N-1}^{k} \mathcal{L}(w(i|k), v(i|k)) + \Gamma(x(k-N|k)) \\ \text{s.t.} \quad & x(i+1|k) = f(x(i|k), w(i|k)) \\ & y(i) = h(x(i|t)) + v(i|t) \\ & x(i|k) \in \mathcal{X}, \quad i \in \{k-N, \ldots, k-1\}, \end{aligned} \tag{2.82}$$

where $x(i|k)$ is the state at time step i, predicted at step k, $\mathcal{X}$ is its domain, $\mathcal{L}(\cdot,\cdot)$ is some general cost function into which we plug the noise terms and $\Gamma(\cdot)$ is a prior weighting function to penalize deviation from the first estimate $x(t-N|k)$. The trajectories solving this optimization problem are termed $x^*(i|k)$, $w^*(i|k)$ and $v^*(i|k)$. The cost function $\mathcal{L}(\cdot,\cdot)$ penalizes the model disturbance w and the measurement error v. A typical choice of cost would be

$$\mathcal{L}(w, v) = qw^2 + rv^2, \tag{2.83}$$

where q and r are positive scalars. This function is positive definite, which comes with substantial benefits regarding numerical solution of the problem (2.82). A similar typical choice for the prior weighting function would be

$$\Gamma(x((t-N)|k)) = (x((t-N)|k) - \hat{x}(t-N))^T P (x((t-N)|k) - \hat{x}(t-N)), \tag{2.84}$$

where P is a positive definite weighing matrix. The basic MHE scheme can now be performed as follows:

1. Obtain new measurements
2. Solve the optimization problem (2.82)
3. Extract the last element in the sequence of best state estimates $x^*(k|k)$ as a final estimate for the current state.
4. back to step 1

Depending on the system properties as well as the noise terms, theoretical results on MHE are available. However, such theoretical guarantees, as in [17], require complex mathematical frameworks, which go beyond the scope of this thesis. For a more complete introduction to MHE, see [26].

3
Implementation

In the following chapter, we apply the theoretical methods introduced in chapter 2 to battery state estimation.
We use the methods of Kalman filtering we have presented, namely EKF and UKF to empower us to perform SoC estimation. We choose Kalman filtering for SoC estimation, because this is a time and safety critical task in a nonlinear system, which does not allow for the computational complexity that comes with methods based in numerical optimization such as MHE. We alluded in section 2.2.2 that the model we have chosen as a design model for our state observers does not model all relevant dynamics of the system. Instead, the ECM relies on parameter variations of its electrical constants in order to reflect the entirety of the battery cell system accurately. For this reason, we have to employ Kalman filtering not only to estimate the battery systems states, in terms of the ECM, but also to identify the systems parameters. The process of estimating model parameters is called parametric estimation. In order to successfully perform SoC estimation of the present battery system we need a framework of simultaneous state and parameter estimation.
We apply MHE to SoH estimation, since SoH estimation is not a time critical process, evolving on a much slower time scale than SoC and other dynamic battery behavior. From this application of MHE we hope to gain improved estimation accuracy over the estimates we would get from using a simpler estimation setup, such as Kalman filtering.
The SoC and SoH estimators, do not, however, work separately from each other. Instead, information exchange between the two is vital in improving estimates. Figure 3.1 shows the information flow of our overall scheme. The SoH estimator's capacity estimates are fed to the SoC estimator, in order to improve SoC estimates. Conversely, the SoC estimator's state and parameter estimates are used by the SoH estimator to improve its capacity estimates. This results in a feedback loop between the SoH estimator and the SoC estimator. The SoC estimtator in turn contains the feedback loop of the state and parameter filters. We populate the block diagram in Figure 3.1 with algorithms in the following chapter.

3.1 State of Charge Estimation

In section 2.3 we derived an array of state estimation algorithms to find the internal state of a dynamical system of the form (2.25) and (2.26). As we have remarked before, we cannot hope to use the ECM model in estimation without populating it with a set of time-varying parameters in real time. This can be done by observing

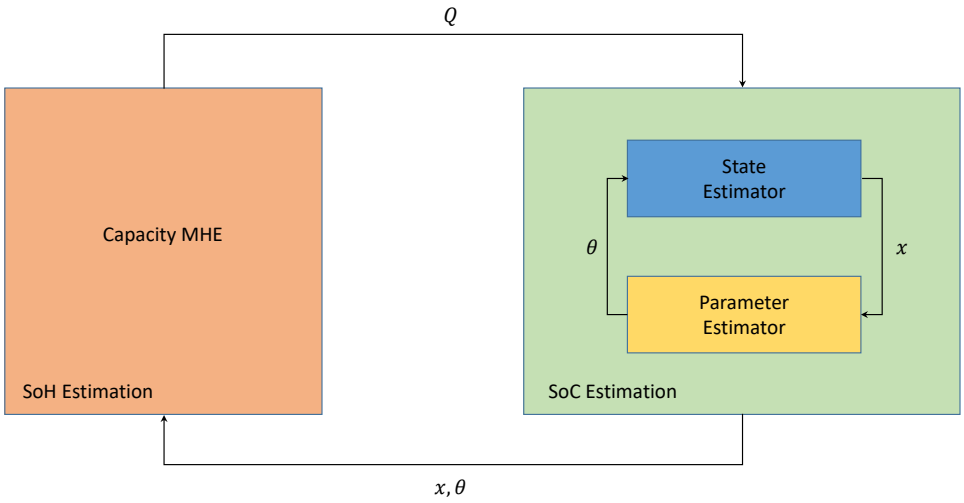

Figure 3.1: This Figure shows the information exchange between the SoC and SoH estimators.

the parameters using an estimator, much in the same way that we use an estimator to find a system's state.
One very useful approximation we can make is to assume different time scales of the dynamics of the ECM's states and its parameters, with the parameters evolving on a much slower time scale than the states. Such an assumption means that, while we model the dynamic behavior of the states using a standard state-space formulation

$$\begin{aligned} x_{k+1} &= f(x_k, u_k; \theta_k) + w_k, \quad x_0 = x_{init} \\ y_k &= h(x_k, u_k; \theta_k) + v_k, \end{aligned} \tag{3.1}$$

we use a random walk model

$$\begin{aligned} \theta_{k+1} &= \theta_k + r_k, \quad \theta_0 = \theta_{\text{init}} \\ d_k &= h(x_k, u_k; \theta_k) + v_k, \end{aligned} \tag{3.2}$$

in order to model the slower change of the parameters. x_k, u_k and y_k are defined as in (2.25) and (2.26). The definitions of f and h are slightly changed; we model the battery system using additive, white process and measurement noise w_k and v_k. The process equations have no explicit time dependency. In (3.1) the parameters are not assumed to be true arguments of f and g, but by virtue of our assumption of different time scales of the dynamics are assumed to be constant. This is often referred to in the literature as quasi-static [24]. (3.2) assumes that the slow parameter change can be modeled using a random walk with the white noise term r_k and the initial value θ_{init}. Although it seems counter intuitive to model parameter change as a random walk, this can be done within the framework of Kalman filtering, since our estimator then uses measurement data to adapt the estimated parameters to the "true" parameters of the system. Finding an explicit model for the parameters θ is

difficult, since they are not exclusively affected by temperature, but also by system current, cell aging and other effects.
Since we now have state-space representations of both the states and the parameters we wish to estimate, we are in a position to adapt the Kalman filtering algorithms presented in section 2.3 to simultaneous state and parametric estimation. The most natural approach to do this would be to simply concatenate the state and parameter dynamics to obtain

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ \\ \theta_{k+1} \end{bmatrix} &= \begin{bmatrix} f(x_k, u_k; \theta_k) + w_k \\ \\ \theta_k + r_k \end{bmatrix}, \quad \begin{bmatrix} x_{init} \\ \\ \theta_{init} \end{bmatrix} \\ y_k &= h(x_k, u_k; \theta_k) + v_k, \end{aligned} \tag{3.3}$$

which can be used as an estimation model in the Kalman filtering algorithms from 2.3. This technique is called joint estimation. The simplicity of this approach, however, comes at some disadvantages.
Firstly, we incur the disadvantage of having to perform large matrix operations, since concatenating the states and parameters inflates the dimensionality of the estimation problem.
Secondly, we face an implementation with a propensity for poor numeric conditioning due to the vastly different time scales within the augmented state space model [24].
For these reasons, we choose to employ two separate Kalman filters to estimate the state and parameters separately. Since the Kalman filter used for state estimation depends on information about the parameters, and vice versa, the two Kalman filters form a feedback loop. Thus, both filters use the estimate of the other filter, respectively, as an input. This approach is called dual estimation. Dual estimation, although avoiding the problems of joint estimation suffers from one draw back: by decoupling the state and parameter dynamics we disregard any potential cross-correlations between the two [24].
In the following sections we apply the previously derived algorithms of EKF and UKF to the problem of dual state and parametric system using an ECM battery model. Throughout the entire filtering algorithms we thus replace the general state space in equations (2.25) and (2.26) with the ECM, such that

$$f(x_k, u_k, w_k; \theta_k) = A(\theta_k)x_k + B(\theta_k)u_k + w_k, \tag{3.4}$$

where $A(\theta_k)$ and $B(\theta_k)$ are the matrices from the ECM model as given in (2.14) with the parameters θ_k inserted and

$$h(x_k, u_k, v_k; \theta_k) = V_{\mathrm{OC}}(x_{k,3}) + x_{k,1} + x_{k,2} + \theta_1 u_k + v_k. \tag{3.5}$$

For brevity of notation and to help understanding, we refrain from plugging these definitions into the algorithms below, wherever omitting them doesn't detract from the clarity of our description.

3.1.1 Dual Unscented Kalman Filtering

In this section we present our implementation of the dual unscented Kalman filtering algorithm (DUKF), which works to realize dual state and parametric estimation of

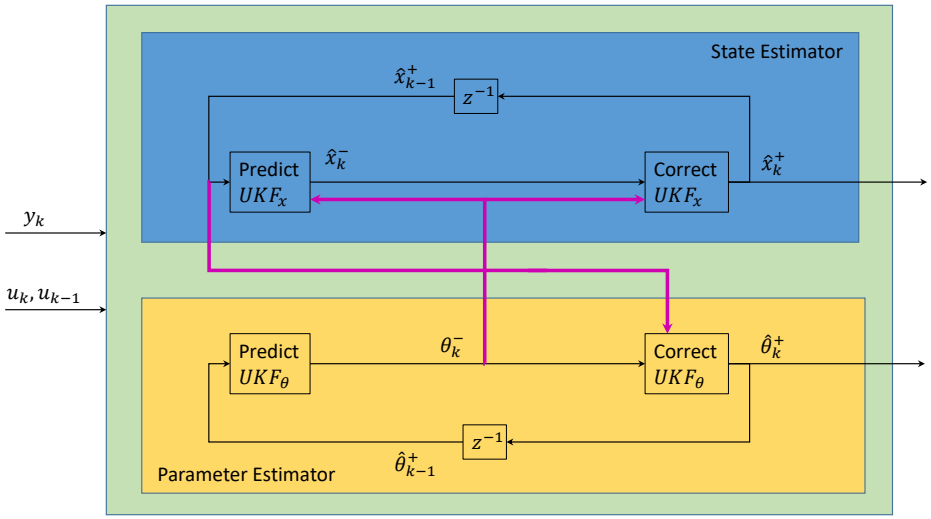

Figure 3.2: This Figure shows the information flow in DUKF. The state estimator is shown in blue; the parametric filter is shown in yellow. Regular information flow is represented by black arrows; information exchange between the filters is represented by thick, pink arrows. Information about the measurement and input data is assumed to be available at every part of the DUKF.

a battery. We begin by modifying the UKF algorithm for use in dual state and parametric estimation of battery systems. The adapted algorithm is presented in the following section.

3.1.1.1 Algorithm

In DUKF the state estimator relies on the parameter filter's estimates and vice versa. Because of the different time scales of these processes, we choose to supply the state estimator with the predicted estimates of the parameter filter and the parameter estimator with the past corrected estimates of the state filter. A diagram of the information flow in DUKF is depicted in Figure 3.2.
We change the output equation used in the parameter estimator to

$$d_k = h(f(x_{k-1}, u_{k-1}; \theta_{k-1}), u_k; \theta_{k-1}) \tag{3.6}$$

explicitly include the effect of the process equation.
Using this information we can integrate two instances of the UKF algorithm into the following dual estimation scheme. We label each step in the DUKF algorithm to indicate which of the individual filters we mean and which step in the general UKF algorithm we perform. For example, step $\theta.1$ means that we are performing the first step in the UKF algorithm for the parameter filter. We present the full algorithm below:

Step $\theta.1$: parameter prediction
We compute an a priori estimate of the parameters. Since we model the change

in parameters as a random walk, we simply assume them to stay constant, which leads to the prediction

$$\hat{\theta}_k^- = \mathbb{E}[\theta_k|\mathbb{Y}_{k-1}] = \theta_{k-1}^+. \tag{3.7}$$

Note that this is a stark simplification from the original UKF algorithm.

Step $\theta.2$: parameter covariance prediction

By virtue of our assumption of additive noise we can predict parameter covariance as

$$\hat{\Sigma}_{\tilde{\theta},k}^- = \hat{\Sigma}_{\tilde{\theta},k-1}^+ + \Sigma_r. \tag{3.8}$$

Step $x.1$: state prediction

Based on this prediction of parameters we can perform the time update of the state filter. We start by computing the sigma points

$$\mathcal{X}_{k-1}^{a,+} = \left\{ \hat{x}_{k-1}^{a,+}, \hat{x}_{k-1}^{a,+} + \gamma\sqrt{\hat{\Sigma}_{\tilde{x},k-1}^{a,+}}, \hat{x}_{k-1}^{a,+} - \gamma\sqrt{\hat{\Sigma}_{\tilde{x},k-1}^{a,+}} \right\}. \tag{3.9}$$

This necessitates a Cholesky factorization of $\hat{\Sigma}_{\tilde{x},k-1}^{a,+}$. Using the process equation we propagate these to the a priori sigma points

$$\mathcal{X}_{k,i}^{x,-} = f(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}; \hat{\theta}_k^-) + \mathcal{X}_{k-1,i}^{w,+}, \tag{3.10}$$

which can, via a weighted average, be converted to our final state prediction

$$\hat{x}_k^- = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{X}_{k,i}^{x,-}. \tag{3.11}$$

Step $x.2$: state covariance prediction

Using the set of predicted sigma points, we compute an a priori estimate of the state covariance matrix. This, as presented in the general UKF algorithm, is done using the weighted average

$$\hat{\Sigma}_{\tilde{x},k}^- = \Sigma_w + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)^T. \tag{3.12}$$

Step $\theta.3$: parameter filter output prediction

We give an a priori estimate of the parameter filter output function $\hat{d}_k$. In order to do this, we first compute a set

$$\mathcal{W}_k = \left\{ \hat{\theta}_k^-, \hat{\theta}_k^- + \gamma_\theta\sqrt{\hat{\Sigma}_{\tilde{\theta},k}^-}, \hat{\theta}_k^- + \gamma_\theta\sqrt{\hat{\Sigma}_{\tilde{\theta},k}^-} \right\} \tag{3.13}$$

of sigma points for the parameters. These points we propagate through the output function to get the set

$$\mathcal{D}_{k,i} = h(f(\hat{x}_{k-1}^+, u_{k-1}, \mathcal{W}_{k,i})u_k, \mathcal{W}_{k,i}), \tag{3.14}$$

which we can average to our predicted output of the parameter filter

$$\hat{d}_k = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{D}_{k,i}. \tag{3.15}$$

Step $x.3$: state filter output prediction
This step remains relatively unchanged from the general UKF algorithm, except that we use the predicted parameters again. We calculate a set of points by passing the a priori state sigma points through the output function, using our predicted parameters. We get

$$\mathcal{Y}_{k,i} = h(\mathcal{X}_{k,i}^{x,-}, u_k, \hat{\theta}_k^-) + \mathcal{X}_{k-1,i}^{v,+}. \tag{3.16}$$

The predicted output of the state estimator is calculated by averaging these points to

$$\hat{y}_k = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{Y}_{k,i} \tag{3.17}$$

Step $x.4$: state filter estimator gain
Analogously to the general UKF formulation, we calculate the required covariance terms

$$\hat{\Sigma}_{\tilde{y},k} = \Sigma_v + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{Y}_{k,i} - \hat{y}_k)(\mathcal{Y}_{k,i} - \hat{y}_k)^T \tag{3.18}$$

and

$$\hat{\Sigma}_{\tilde{x},\tilde{y},k}^{-} = \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{Y}_{k,i} - \hat{y}_k)^T. \tag{3.19}$$

These can be plugged into the estimator gain's definition

$$L_k^x = \hat{\Sigma}_{\tilde{x},\tilde{y},k}^{-} \hat{\Sigma}_{\tilde{y},k}^{-1} \tag{3.20}$$

Step $\theta.4$: parameter filter estimator gain
We compute the estimator gain matrix for the parameter filter using the sigma points generated in the parameter filter. This leads to

$$\Sigma_{\tilde{d},k} = \Sigma_v + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{D}_{k,i} - \hat{d}_k)(\mathcal{D}_{k,i} - \hat{d}_k)^T \tag{3.21}$$

and

$$\Sigma_{\hat{\theta}\hat{d},k}^{-} = \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{W}_{k,i} - \hat{\theta}_k^-)(\mathcal{D}_{k,i} - \hat{d}_k)^T. \tag{3.22}$$

We insert this into the definition of the estimator gain to obtain

$$L_k^\theta = \Sigma_{\hat{\theta}\hat{d},k}^{-} \Sigma_{\tilde{d},k}^{-1}. \tag{3.23}$$

Step $x.5$: state correction
We calculate an a posteriori state estimate

$$\hat{x}_k^+ = \hat{x}_k^- + L_k^x \tilde{y}_k. \tag{3.24}$$

Step $x.6$: state covariance correction
Using previous results, we correct our estimate of the state covariance matrix. The final covariance estimate is

$$\hat{\Sigma}_{\tilde{x},k}^{+} = \hat{\Sigma}_{\tilde{x},k}^{-} - L_k^x \hat{\Sigma}_{\tilde{y},k} (L_k^x)^T. \tag{3.25}$$

Step $\theta.5$: parameter correction
We use the parameter estimator gain to get the a posteriori estimate

$$\hat{\theta}_k^+ = \hat{\theta}_k^- + L_k^\theta(y_k - \hat{d}_k) \tag{3.26}$$

of the parameters. Note that we use a different measurement residual to correct the parameters, than we use to correct the state.
Step $\theta.6$: parameter covariance correction
We calculate the corrected parameter covariance by

$$\hat{\Sigma}_{\tilde{\theta},k}^+ = \hat{\Sigma}_{\tilde{\theta},k}^- - L_k^\theta \Sigma_{\tilde{d},k}(L_k^\theta)^T. \tag{3.27}$$

With this, we have integrated the workings of two distinct UKF to create a functioning DUKF.

3.1.2 Dual Extended Kalman Filtering

In the following section we demonstrate our adaptation of the EKF algorithm presented in section 2.3 to the problem of dual state and parametric estimation of our battery system and thus its transformation into the dual extended Kalman filter (DEKF). We choose the same approach as in DUKF in that we modify one filter to estimate the system parameters. We then carefully integrate it with our state estimator such that the two filters works in synergy to perform state and parametric estimation. The following section presents the algorithm.

3.1.2.1 Algorithm

As in DUKF the individual EKFs must be placed in a feedback loop in order to function and give accurate estimates of the states and parameters. As such, we choose the same order of arranging the individual filter steps of the two filters in DEKF we chose in DUKF. The state filter uses the predictions of the parameter filter to calculate its state estimates. The parameter filter uses the state estimator's past corrected estimates to compute its parameter estimates. The information flow in DEKF is depicted in Figure 3.3. We remark that there is more information exchange between the individual EKFs in DEKF than there is between the individual UKFs in DUKF. Unlike in DUKF, in DEKF we pass the state filter's past estimator gain to the parameter filter. To explore why this is necessary we set about computing the Jacobians required in the EKF algorithm as presented in 2.3. Because we use the state space models in (3.1) and (3.2) to model the two processes of change in the system's state and parameters respectively, we need to compute the Jacobians $\frac{\partial f}{\partial x}(\hat{x}_k^-, u_k; \theta_k)$, $\frac{\partial h}{\partial x}(\hat{x}_k^-, u_k; \theta_k)$ and $\frac{\partial h}{\partial \theta}(\hat{x}_k^-, u_k; \theta_k)$. The first two terms are computed easily enough by taking the partial derivatives of the ECM in (2.14). This leads to

$$\frac{\partial f}{\partial x}(\hat{x}_k^-, u_k; \theta_k) = A(\theta_k) = \text{diag}(\exp\left(-\frac{h}{\theta_3}\right), \exp\left(-\frac{h}{\theta_5}\right), 1) \tag{3.28}$$

and

$$\frac{\partial h}{\partial x}(\hat{x}_k^-, u_k; \hat{\theta}_k^-) = \underbrace{\begin{bmatrix} 1 & 1 & \frac{\partial V_{\text{OC}}}{\partial x_3}(\hat{x}_{k,3}^-) \end{bmatrix}}_{:=C_k^x}. \tag{3.29}$$

We determine the unknown function $\frac{\partial V_{\mathrm{OC}}}{\partial x_3}(\cdot)$ using numeric derivation of the function $V_{\mathrm{OC}}(\cdot)$, which we identified experimentally. We find the third Jacobian to be

$$\frac{\partial h}{\partial \theta}(\hat{x}_k^-, u_k; \theta_k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.30}$$

We substitute this by the corresponding total derivative in hopes of capturing the effect of the states' dependency on the parameters. This was not necessary in (3.28) and (3.29), as the process and measurement equations already explicitly depend on the states. The measurement equation, however, only implicitly depends on the parameters, with the exception of R_0, on which it depends explicitly. Thus we get

$$\frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \hat{\theta}_k^-) = \frac{\partial h}{\partial \theta}(\hat{x}_k^-, u_k; \hat{\theta}_k^-) + \frac{\partial h}{\partial x}(\hat{x}_k^-, u_k; \hat{\theta}_k^-)\frac{\mathrm{d}\hat{x}_k^-}{\mathrm{d}\theta} := C_k^\theta, \tag{3.31}$$

the last term of which can be expanded to

$$\begin{aligned} \frac{\mathrm{d}\hat{x}_k^-}{\mathrm{d}\theta} &= \frac{\partial f}{\partial \theta}(\hat{x}_{k-1}^+, u_k; \hat{\theta}_k^-) + \frac{\partial f}{\partial x}(\hat{x}_{k-1}^+, u_{k-1}; \hat{\theta}_k^-)\frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta}, \\ \frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta} &= \frac{\mathrm{d}\hat{x}_{k-1}^-}{\mathrm{d}\theta} - L_{k-1}^x \frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_{k-1}^-, u_{k-1}; \hat{\theta}_k^-), \end{aligned} \tag{3.32}$$

where

$$\frac{\partial f}{\partial \theta}(x_k, u_k; \hat{\theta}_k^-) = \begin{bmatrix} 0 & [1-\exp(-\frac{h}{\hat{\theta}_{k,3}^-})]u_k & \frac{h}{(\hat{\theta}_{k,3}^-)^2}\exp(-\frac{h}{\hat{\theta}_{k,3}^-})[x_{1,k}-\hat{\theta}_{k,1}^- u_k] & 0 & 0 \\ 0 & 0 & 0 & [1-\exp(-\frac{h}{\hat{\theta}_{k,5}^-})]u_k & \frac{h}{(\hat{\theta}_{k,5}^-)^2}\exp(-\frac{h}{\hat{\theta}_{k,5}^-})[x_{2,k}-\hat{\theta}_{k,4}^- u_k] \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.33}$$

With this, we have a recursive way of computing the total derivative $\frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \theta_k)$. The computation of this total derivative requires the past estimator gain of the state filter L_{k-1}^x. This explains the additional information exchange between the state and parameter filter, as depicted in Figure 3.3. We also note that this recursive way of computing $\frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \theta_k)$ necessitates initial values for the variables $\frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \theta_k)$ and $\frac{\mathrm{d}\hat{x}_{k-1}^-}{\mathrm{d}\theta}$, the choice of which are discussed later.

We now have all necessary derivatives for the DEKF to work. Thus, we integrate two EKF filters to perform the dual estimation task. Generally this happens by plugging in our results up until now into the general EKF algorithm from 2.3. We label each step in the following algorithm to indicate which of the individual filters we mean and which step in the general EKF algorithm we perform. For example, step $\theta.1$ means that we are performing the first step in the EKF algorithm for the parameter filter. The algorithm reads:

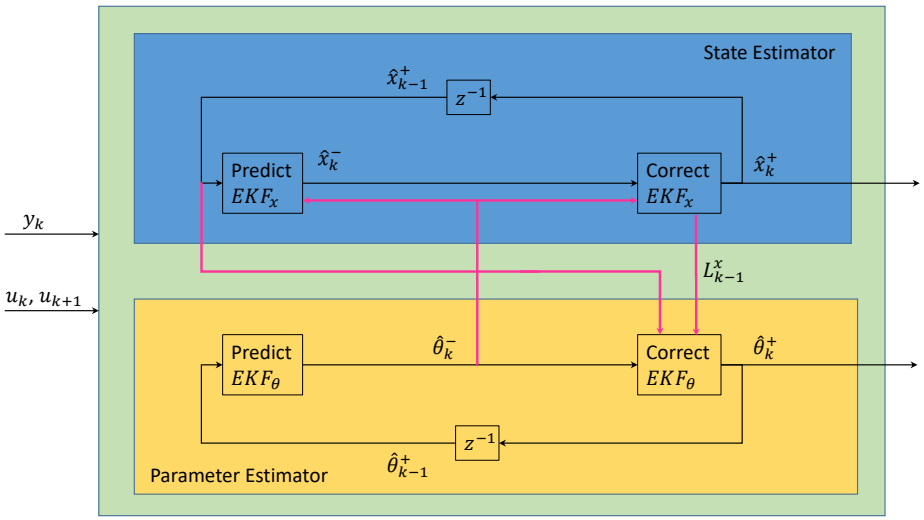

Figure 3.3: This Figure shows the information flow in DEKF. The state estimator is shown in blue; the parametric filter is shown in yellow. Regular information flow is represented by black arrows; information exchange between the filters is represented by thick, pink arrows. Information about the measurement and input data is assumed to be available at every part of the DEKF. We note that there is more information exchange between the two individual filters than in DUKF. This comes in the form of the past estimator gain of the state filter being supplied to the parameter filter.

Step $\theta.1$: parameter prediction
Since we model the parameters as performing a random walk, our prediction is for them to stay constant, which means

$$\hat{\theta}_k^- = \mathbb{E}[\theta_k|\mathbb{Y}_{k-1}] = \hat{\theta}_{k-1}^+. \tag{3.34}$$

Step $\theta.2$: parameter covariance prediction
We predict the additive noise r_k to increase the parameter error covariance to

$$\hat{\Sigma}_{\tilde{\theta},k}^- = \hat{\Sigma}_{\tilde{\theta},k-1}^+ + \Sigma_r. \tag{3.35}$$

Step $x.1$ state prediction
We predict the current state by propagating the past best estimate of the state through the process equation. We obtain

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_{k-1}; \hat{\theta}_k^-) = A(\hat{\theta}_k^-)\hat{x}_{k-1}^+ + B(\hat{\theta}_k^-)u_k. \tag{3.36}$$

Step $x.2$ state covariance prediction
Since we have assumed additive noise, we can simply use (2.69) to compute an a priori estimate of the state error covariance

$$\hat{\Sigma}_{\tilde{x},k}^- = A(\hat{\theta}_k^-)\hat{\Sigma}_{\tilde{x},k-1}^+ A(\hat{\theta}_k^-)^T + \Sigma_w. \tag{3.37}$$

Step $\theta.3$ and $x.3$: parameter filter output prediction
We use the same output prediction for both filers, leading to one common a priori estimate of the system output

$$\hat{y}_k = g(\hat{x}_k^-, u_k; \hat{\theta}_k^-) = V_{\mathrm{OC}}(\hat{x}_{k,3}^-) + \hat{x}_{k,1}^- + \hat{x}_{k,2}^- + \hat{\theta}_{k,1}^- u_k. \tag{3.38}$$

Step $x.4$: parameter filter estimator gain
We get the estimator gain for our state filter by simply evaluating (2.77) using the variables of the state filter. We get

$$L_k^x = \hat{\Sigma}_{\tilde{x},k}^- (C_k^x)^T [C_k^x \hat{\Sigma}_{\tilde{x},k}^- (C_k^x)^T + \Sigma_v]^{-1} \tag{3.39}$$

Step $\theta.4$: parameter filter estimator gain
In order to compute the estimator gain for the parameter filter, we need $C_k^\theta = \dfrac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \theta_k)$. This we compute by updating the recursive equations in (3.31) and (3.32). Having done this, we compute the parameter filter gain by evaluating (2.77), which leads to

$$L_k^\theta = \hat{\Sigma}_{\tilde{\theta},k}^- (C_k^\theta)^T [C_k^\theta \hat{\Sigma}_{\tilde{\theta},k}^- (C_k^\theta)^T + \Sigma_v]^{-1}. \tag{3.40}$$

Step $x.5$: state correction
We correct our state estimate

$$\hat{x}_k^+ = \hat{x}_k^- + L_k^x \tilde{y}_k \tag{3.41}$$

using the state estimator gain and measurement residual.

Step $x.6$: state covariance correction
We compute an a posteriori state covariance estimate

$$\hat{\Sigma}^{+}_{\tilde{x},k} = \hat{\Sigma}^{-}_{\tilde{x},k} - L^{x}_{k}\hat{\Sigma}_{\tilde{y},k}(L^{x}_{k})^{T} \tag{3.42}$$

by (2.79).
Step $\theta.5$: parameter correction
The corrected parameter estimate can be computed by

$$\hat{\theta}^{+}_{k} = \hat{\theta}^{-}_{k} + L^{\theta}_{k}\tilde{y}_{k}. \tag{3.43}$$

Step $\theta.5$: parameter correction
Our final estimate of the parameter error covariance matrix is

$$\hat{\Sigma}^{+}_{\tilde{\theta},k} = \hat{\Sigma}^{-}_{\tilde{\theta},k} - L^{\theta}_{k}\hat{\Sigma}_{\tilde{y},k}(L^{\theta}_{k})^{T} \tag{3.44}$$

computed in analogy to (2.79).
Thus we have combined two EKFs as detailed in the general algorithm in 2.3 to solve the present task of dual state and parametric estimation in our DEKF.

3.1.3 Filter Weight Adaptation

We wish now to augment our dual estimation schemes by an uncertainty adaptivity feature. Specifically, we add to our dual estimators by adding a program that adapts the values chosen for the process noise covariance Σ_w and sensor noise covariance Σ_v. This corresponds to tuning the state filters online, in accordance with the trajectory the system is following. We pursue two main goals with this approach.
The adaptivity mechanism should improve overall filter performance. In particular it helps the estimator to adapt to varying conditions where the effect of all disturbances modeled by the process and sensor noise w_k and v_k changes throughout operations. An example of such conditions is operation in extreme temperatures or at the limits of the SoC range, where the simplified modeling approach we chose in this thesis may be brought to the limits of its validity.
Adding this adaptivity feature greatly reduces the burden placed on the user of our estimation algorithms, as tuning the covariance matrices of the state UKF becomes unnecessary. This may sound like a trifle initially, but ends up making a huge difference in man-hours required to set up a working example of our estimation scheme.
We adapt the covariance matrices $\Sigma_{w,k}$ and $\Sigma_{v,k}$ at every time step k using an algorithm called covariance matching. In order to do this, we choose a suitable adaptivity horizon L_{ad} and compile an approximation

$$F_k = \frac{1}{L_{ad}} \sum_{n=k-L_{ad}+1}^{k} \mu_n \mu_n^T \tag{3.45}$$

of the measurement residual covariance, using the measurement residuals μ_n of the past L_{ad} time steps. Covariance matching is based on this quantity and all further computations derive from it.

Using the state estimator gain L_k^x and our estimate of the measurement residual covariance F_k, we obtain an approximation of the process noise covariance

$$\Sigma_{w,k} = L_k^x F_k (L_k^x)^T. \tag{3.46}$$

An intuitive motivation of this equation is that L_k^x maps measurement residuals to state corrections in the state correction step of Kalman filtering (2.33). Here it is used to map from measurement residual covariance to state error covariance.
Due to the different natures of the DUKF and DEKF algorithms, we have different quantities at our disposal with in the two estimation schemes. Because of this, we have to calculate the sensor noise covariance by two slightly varying equations.
In DUKF, we add the output covariance $\hat{\Sigma}_{\tilde{y},k}$ generated by the sigma points' variation from step $x.4$ to F_k, to get an estimate

$$\Sigma_{v,k} = F_k + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{Y}_{k,i} - \hat{y}_k)(\mathcal{Y}_{k,i} - \hat{y}_k)^T \tag{3.47}$$

of the sensor noise covariance at the current time.
In DEKF, we compute an estimate of the sensor noise covariance using

$$\Sigma_{v,k} = F_k + C_k^x \hat{\Sigma}_{\tilde{x},k}^+ (C_k^x)^T. \tag{3.48}$$

The matrices $\Sigma_{w,k}$ and $\Sigma_{v,k}$ are then used as process and sensor covariance matrices in the UKF and EKF algorithms. Since they require variables from the correction step, such as $\hat{\Sigma}_{\tilde{x},k}^+$, we delay them by one sample. This means that $\Sigma_{w,k-1}$ and $\Sigma_{v,k-1}$ are used at time k.
In the DEKF algorithm, this changes (3.37) in step $x.2$ to

$$\hat{\Sigma}_{\tilde{x},k}^- = A(\hat{\theta}_k^-) \hat{\Sigma}_{\tilde{x},k}^- A(\hat{\theta}_k^-)^T + \Sigma_{w,k-1} \tag{3.49}$$

and (3.39) in step $x.4$ to

$$L_k^x = \hat{\Sigma}_{\tilde{x},k}^- (C_k^x)^T [C_k^x \hat{\Sigma}_{\tilde{x},k}^- (C_k^x)^T + \Sigma_{v,k-1}]^{-1}. \tag{3.50}$$

For the DUKF scheme, this changes equation (3.12) in step $x.2$ to

$$\hat{\Sigma}_{\tilde{x},k}^- = \Sigma_{w,k-1} + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)^T \tag{3.51}$$

and equation (3.18) in step $x.4$ to

$$\hat{\Sigma}_{\tilde{y},k} = \Sigma_{v,k-1} + \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{Y}_{k,i} - \hat{y}_k)(\mathcal{Y}_{k,i} - \hat{y}_k)^T. \tag{3.52}$$

Using this approach to adapt the DUKF in the present application, we face a problem that seriously affects the integrity of the entire filter. Since we only have one system output, namely the battery's terminal voltage V_t, F_k is a scalar in our case. We recognize that this reduces the definition of $\Sigma_{w,k}$ in (3.46) to an outer product, such that $\Sigma_{w,k}$ is never of rank greater than 1. Since the augmented state covariance

matrix $\hat{\Sigma}^{a,+}_{\tilde{x},k-1}$ is a block diagonal matrix incorporating $\Sigma_{w,k}$, this means that $\hat{\Sigma}^{a,+}_{\tilde{x},k-1}$ also doesn't have full rank and is thus not positive definite. Because of this, Cholesky decomposition of $\hat{\Sigma}^{a,+}_{\tilde{x},k-1}$ fails and the DUKF algorithm does not work.
To remedy this we use a reduced formulation of the DUKF where we use the sigma points to explore the state space around our last estimate only, choosing not to explore the noise space anymore. This is equivalent to refraining from ever using the augmented state and covariance matrix and setting all noise terms calculated in the state UKF to zero. $\hat{\Sigma}^{a,+}_{\tilde{x},k-1}$ now no longer has $\Sigma_{w,k}$ as a block diagonal entry, and is positive definite once more. However, making this modification also slightly changes the structure of the state UKF. In particular, this reduces the equations for finding the a priori state estimate in step $x.1$ to

$$\mathcal{X}^{+}_{k-1} = \left\{ \hat{x}^{+}_{k-1}, \hat{x}^{+}_{k-1} + \gamma\sqrt{\hat{\Sigma}^{+}_{\tilde{x},k-1}}, \hat{x}^{+}_{k-1} - \gamma\sqrt{\hat{\Sigma}^{+}_{\tilde{x},k-1}} \right\} \tag{3.53}$$

and

$$\mathcal{X}^{x,-}_{k,i} = f(\mathcal{X}^{+}_{k-1}, u_{k-1}, , \hat{\theta}^{-}_{k}). \tag{3.54}$$

In step $x.3$ the propagation of sigma points through the output function reduces to

$$\mathcal{Y}_{k,i} = h(\mathcal{X}^{x,-}_{k,i}, u_k, \hat{\theta}^{-}_{k}). \tag{3.55}$$

This results in a slight loss in complexity and estimation quality of the filter. We see in validation, that this loss is outweighed by the benefits of the adaptivity feature.
Since we compute the covariance matrices used in the next step, we need initial values for them. The choice of these initial values are discussed later.
We remark that because of its derivation from the measurement residual this adaptivity feature is very susceptible to sensor bias. It can be easily verified that large sensor bias leads the mechanism to tune the filter so as to prefer its predictions vastly and neglect the measurements.
Because of the adaptive tuning mechanism's reliance on measurement residual to compute process and sensor noise covariances it is susceptible to incorrect initialization of the estimator. Transient large measurement residuals due to inaccurate initial values may lead to bad tuning by the filter weight adaptation. To avoid this, we choose to only activate the adaptive tuning mechanism after we have allowed the estimator to converge. We allot a fixed time t_{sa} for this, the choice of which will be discussed together with all other design variables.
This mechanism is now usable for online choice of process and sensor covariance in both filtering strategies. We implement an adaptive version of our DUKF and DEKF, incorporating this covariance matching feature. The above adaptivity mechanism was adapted from [28, 19]. See there for further references and theoretical background.

3.1.4 Sensitivity Analysis

In section 2.2.4 we introduced some filter equations in order to determine data sensitivity. Data sensitivity is a quantity meant to help us identify input trajectories to the system which permit estimation of the system's parameters. We proposed

to improve the quality of our estimates of the battery system's parameters by only choosing to update our parameter estimate whenever our sensitivity filters indicate an information-rich trajectory.
Applying this approach in practice one soon runs into a problem: What to do with the parameter error covariance, when the sensitivity check decides to freeze the parameter estimate?
Suppose the sensitivity filter, at some time step, returns the information, that the current input trajectory is poor in information about one particular parameter. We can simply decide, instead of computing a correction $\hat{\theta}^+_{k,i}$ of that parameter based on low-quality data, to freeze that parameter, that is to propagate its old value $\hat{\theta}^+_{k-1,i}$ in time. It follows from this, that we also don't want to update the entries in the parameter error covariance matrix $\hat{\Sigma}^+_{\tilde{\theta},k-1}$ corresponding to this particular parameter in accordance with the poor data. However, simply freezing the corresponding values in the error covariance matrix does not solve the problem, as editing the parameter error covariance matrix in such an arbitrary way compromises its positive definiteness. Opting to perform only the prediction step

$$\hat{\Sigma}^-_{\tilde{\theta},k} = \hat{\Sigma}^+_{\tilde{\theta},k-1} + \Sigma_r \tag{3.56}$$

is an equally bad idea, as this leads to covariance build-up in case of long input trajectories without sensitive data. Covariance build up can destabilize the filter.
We combat this problem by dividing the parameter UKF into three different sub-filters, where one filter estimates the static resistance R_0, and the other two filters each estimate the parameters of one of the RC-branches. This way, if one of the filters receives the signal, that sensitivity with respect to one of its parameters is low, it can simply freeze all estimates without compromising the functionality of the entire parameter filter. Subdividing the parameter filter in this way doesn't change the inner workings of parameter UKF as such. The algorithms presented in 3.1.1 and 3.1.2 can still be applied in their unchanged form, the only distinction being that there is now more than one parameter filter, but each of them estimates less parameters on its own.
With this, we now only need to define suitable sensitivity thresholds for our data sensitivity feature to improve estimation performance.
It has to be remarked, however, that there is a feedback loop between the parameter filter and the sensitivity check feature, in the sense that each uses the other's past output to compute its own current output. Since we use logical operators in this design concept, there can be no trivial stability analysis of this feedback system. Thus, performing such a stability analysis falls out of the scope of this thesis, but should be kept in mind for future research.

3.1.5 Tuning and Initialization

The performance of any Kalman filter is critically dependent on tuning and initialization. By initialization we mean the choice of initial values for the states, parameters and their respective covariance matrices in this context. Generally, the filter should be robust against initialization with wrong values, as knowing the system's state precisely at initialization is almost impossible in practice. By tuning

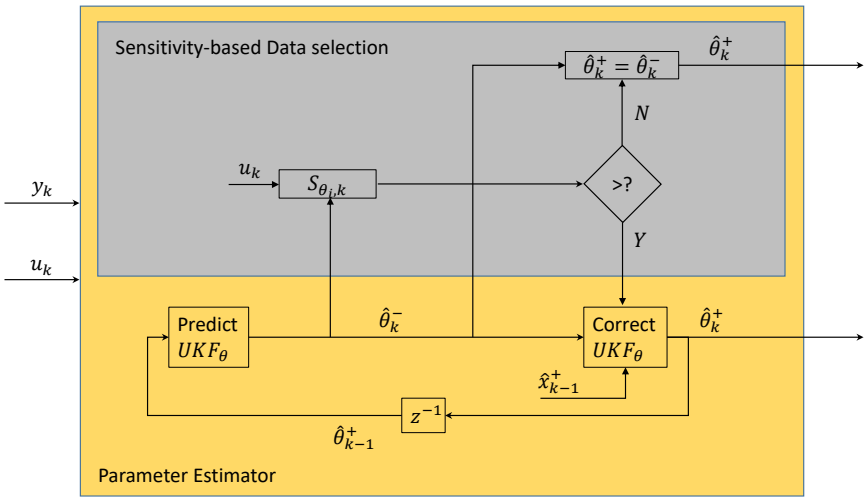

Figure 3.4: This Figure shows information flow in the sensitivity analysis feature. The grey part inside the parameter filter depicted in yellow is the sensitivity check. It evaluates the sensitivity equations and freezes the measurement update in case of low sensitivity. In case of high sensitivity, the parameter filter works as usual.

we mean choosing suitable estimates for the covariance matrices of the process and measurement noise for both the state and parameter filters. It is pertinent to stress here that all Kalman filters are prone to divergence and instability when tuned badly. Manual tuning of Kalman filters often turns out to devour many hours of work, as it is not an exact science. Instead tuning is an iterative process of choosing some tuning parameters and validating them in simulation. There are, however, some guidelines to help make sensible choices of tuning parameters.

3.1.5.1 Initialization

To make a well-reasoned choice of initial values for our dual estimation setup, we once again consider the physical characteristics of the battery system. We recall that the first two states in our state estimator represent the voltages resulting from dynamic polarization of the battery cell. These voltages, which we modeled as dropping over an RC sub-circuit decay quickly in the absence of an input current. If we assume some time to pass between disengaging the estimation setup and reactivating it, we can reasonably assume these states to start at 0 on startup.
Choosing a good initial value for the SoC can be a more tricky affair. There are certainly many cases, where using the last estimated SoC value upon shutdown of the estimator as an initial value at startup is a good choice. Due to self-discharging of batteries, however, this may not always be the case, especially in the case of very long resting times of the battery. Another way to find initial SoC values would be to measure battery terminal voltage and take the SoC that corresponds to this voltage in the OCV-SoC function. The accuracy of this method depends on whether the

battery has reached equilibrium and on the shape of the OCV-SoC curve. Since there is no guaranteed way to find accurate initial SoC values in all operating scenarios, we initialize our estimators at $z = 0.5$ in all our tests. This method requires no previous knowledge, whatsoever, and is at least sure to keep the initial deviation of the SoC below 0.5.
By these deliberations, we obtain an initial state vector

$$x_{\text{init}} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}, \tag{3.57}$$

which we can use for initializing our estimation scheme independently of the simulation scenario of experiment.
Choosing the initial error covariance matrix for the state estimator $\hat{\Sigma}_{\tilde{x},0}$ boils down to the question of how certain we are that our initial guesses for the states are accurate. We choose $\hat{\Sigma}_{\tilde{x},0}$ as a diagonal matrix, where the intuition for choosing each of the diagonal entries is the following: the more certain we are of the initial value we chose for the corresponding state, the smaller the value for that specific diagonal entry of $\hat{\Sigma}_{\tilde{x},0}$. However, scaling of the individual variables and algorithm choice play a role in choosing a magnitude for the entries in $\hat{\Sigma}_{\tilde{x},0}$, making the process of finding $\hat{\Sigma}_{\tilde{x},0}$ less straightforward, than may seem from this guideline. Choosing the right values is an iterative process, in any sort of estimator tuning.
We argued that the initial guesses for the first two states can be regarded as accurate in most scenarios. As an initial value for the SoC, we did not find a good candidate for all cases. Considering this, our first guess would be for the first two diagonal entries of $\hat{\Sigma}_{\tilde{x},0}$ to be small in relation to their respective state's magnitude and the last diagonal entry to be large in relation to its respective state's magnitude. After a process of tuning, we end up with the covariance matrix

$$\hat{\Sigma}_{\tilde{x},0}^{DUKF} = \text{diag}(10^{-8}, 10^{-8}, 10^{-1}) \tag{3.58}$$

for use with the DUKF algorithm and with the covariance matrix

$$\hat{\Sigma}_{\tilde{x},0}^{DEKF} = \text{diag}(10^{-4}, 10^{-4}, 1) \tag{3.59}$$

for use with the DEKF algorithm.
Finding initial values for the parameter set is not a straightforward task. None of the parameters can be measured directly, of course, so we have to use some experimental system identification methods to get an idea of what good starting values would look like. In this thesis we have at our disposition experimentally determined data under a couple of different temperatures for the specific battery cell we use for validation. These measurements must be considered as approximations and do not reflect the true parameters as the true parameters depend on SoC, current and other factors. We use this data in order to obtain initial guesses for the parameter vector θ_{init}, solely

based on the ambient temperature measured for each validation experiment. This information we can assume as readily available in all battery estimation applications. To arrive at an initial guess for the error covariance matrix of the parameter filter we base our first guess mainly on the variables' magnitude. The resistances used in the ECM model lie in the order of magnitude of 1mΩ, the time constant of the first branch lies in the order of magnitude of 1s and the time constant of the second branch lies in the order of magnitude of 10s. Using this information and performing a process of tuning, we get the initial covariance matrices

$$\hat{\Sigma}^{DUKF}_{\tilde{\theta},0} = \hat{\Sigma}^{DEKF}_{\tilde{\theta},0} = \mathrm{diag}(10^{-8}, 10^{-10}, 10^{-4}, 10^{-10}, 10^{-5}). \tag{3.60}$$

In the case of DEKF, we also need some initial values for $\frac{\mathrm{d}h}{\mathrm{d}\theta}(\hat{x}_k^-, u_k; \hat{\theta}_k^-)$ and $\frac{\mathrm{d}\hat{x}_{k-1}^-}{\mathrm{d}\theta}$. Since no obvious way of computing sensible initial values presents itself, we choose 0 for both.
We also need initial values for the covariance matrices in the adaptivity feature. However, these initial values hardly affect the filter performance at all, since they only affect the first sample, where covariance is mainly determined by our choice of initial covariance anyway. We chose this initial value equal to the covariance matrix we employ when we don't use the adaptivity feature.

3.1.5.2 Tuning

By tuning we mean choosing a certain set of design parameters for a filter. Tuning can make or break an estimator in terms of performance and even destabilize it. The main design parameters in our estimation setup are the state process noise covariance Σ_w and the measurement noise covariance Σ_v and the parameter process noise Σ_r. The DUKF needs an additional design variable for each component filter a_x and a_θ to control the spread of the sigma points around the past best estimate of both its sub-filters. In case we use the adaptivity feature described in 3.1.3, we need no estimate of Σ_w, since our adaptivity algorithm chooses it autonomously.
Very generally, the covariance matrices tell the estimation algorithm, how trustworthy a certain source of information is. If we choose large entries in a covariance matrix, we tell our algorithm, that the source of information is noisy and thus the information is not trustworthy. Small entries mean that we assume a noise-free source of information, thus giving trustworthy information.
We tune the estimators in an iterative process of trial and error, validating the tuning with simulations and experimental data at every step. Following this method, after many hours of tuning, we arrive at the values

$$\Sigma_w^{UKF} = \mathrm{diag}(10^{-3}, 10^{-5}, 10^{-8}), \tag{3.61}$$

$$\Sigma_v^{UKF} = 5 \cdot 10^{-3}, \tag{3.62}$$

$$\Sigma_r^{UKF} = \mathrm{diag}(10^{-12}, 10^{-12}, 10^{-10}, 10^{-11}, 10^{-4}), \tag{3.63}$$

$$a_x = 0.5 \tag{3.64}$$

and

$$a_\theta = 0.12 \tag{3.65}$$

for the DUKF. We get

$$\Sigma_w^{EKF} = \mathrm{diag}(10^{-5}, 10^{-7}, 10^{-9}), \tag{3.66}$$

$$\Sigma_v^{EKF} = 5 \cdot 10^{-1} \tag{3.67}$$

and

$$\Sigma_r^{EKF} = \mathrm{diag}(10^{-11}, 10^{-15}, 10^{-5}, 10^{-6}, 10^{-3}) \tag{3.68}$$

for the DEKF.
We also need to choose design parameters for our sensitivity analysis and adaptivity features. Again from an iterative tuning process, we choose the sensitivity thresholds

$$s_{R_0} = 5 \tag{3.69}$$

$$s_{R_1} = 10 \tag{3.70}$$

$$s_{\tau_1} = 10^{-3} \tag{3.71}$$

$$s_{R_2} = 5 \tag{3.72}$$

$$s_{\tau_2} = 5 \cdot 10^{-5}. \tag{3.73}$$

For the adaptivity feature, an adaptivity horizon of

$$L_{\text{adapt}}^{EKF} = 10^3 \tag{3.74}$$

samples works well for the DEKF, whereas a longer horizon of

$$L_{\text{adapt}}^{UKF} = 7 \cdot 10^3 \tag{3.75}$$

is preferable for the DUKF. We choose to activate the adaptivity feature at

$$t_{sa} = 100\text{s}, \tag{3.76}$$

when the filter has had ample time to converge.

3.2 State of Health Estimation

In the following section, we apply the method of MHE as presented in section 2.3 to estimate battery state of health. As we explored in 2.1, SoH can be estimated indirectly by estimating the battery's total capacity Q.

3.2.1 Moving Horizon Estimation

The main task in setting up a MHE is to find a suitable optimization problem formulation in the form of (2.82). A natural place to start is to formulate the equality constraints resulting from the dynamics of the system. We start using the state space equations from the first two states in their unchanged form, since they don't contain the total capacity Q. We thus have the constraints

$$x(i+1|k) = \underbrace{\begin{bmatrix} \exp(-\frac{h}{\tau_{1,k}}) & 0 \\ 0 & \exp(-\frac{h}{\tau_{1,k}}) \end{bmatrix}}_{:=A_k} x(i|k) + \underbrace{\begin{bmatrix} R_{1,k}(1-\exp(-\frac{h}{\tau_{1,k}})) \\ R_{2,k}(1-\exp(-\frac{h}{\tau_{2,k}})) \end{bmatrix}}_{:=B_k} u_i. \tag{3.77}$$

This means that the MHE depends on the parameters estimated by the SoC estimator in order to evaluate the equality constraints that capture the system dynamics. Next, we tentatively formulate the SoC dynamics as a constraint and get

$$z(i+1|k) = z(i|k) + \frac{h\eta}{Q}u_i. \tag{3.78}$$

We recognize, that this is a nonlinear constraint, since Q is our main decision variable. To prevent this, we introduce the transformation

$$\tilde{Q} = \frac{1}{Q}, \tag{3.79}$$

where $\tilde{Q}$ is our new decision variable. Substituting this in (3.78), we get a linear constraint

$$z(i+1|k) = z(i|k) + h\eta\tilde{Q}u_i. \tag{3.80}$$

With this, we have made constraints encapsulating the process equations. Next, we use the system's output equation as a constraint to obtain

$$y_i = h(x(i|k), z(i|k), u_i, e_i) = V_{\mathrm{OC}}(z(i|k)) + x_1(i|k) + x_2(i|k) + R_0u_i + e(i|k), \tag{3.81}$$

where $e(i|k)$ is a decision variable with the purpose of simulating measurement noise. We take due notice of the fact that this is a nonlinear constraint. It is therefore not be possible to find a formulation as a quadratic program, nor is the problem convex. This also necessitates the use of a nonlinear solver. We choose the cost function

$$\mathcal{L}(e) = e^T De, \tag{3.82}$$

where $e = [e(i|k), \ldots, e(i|k-N+1)]^T$ with k as the current step and N as the horizon length and $D = \mathrm{diag}(d_1, \ldots, d_N)$. This cost function motivates to explain the measured outputs as well as possible by adapting the ECM. We choose the prior weighting function

$$\Gamma(\hat{x}(t-N)) = (x(t-N|k) - \hat{x}(t-N))^T P(x(t-N|k) - \hat{x}(t-N)), \tag{3.83}$$

where $P = \mathrm{diag}(p_1, p_2, p_3)$. This prior weighting enables us to vary the initial condition $x(t-N|k)$ around that estimated by our SoC estimator, to give the MHE the flexibility to correct possible estimation errors by the SoC estimator. Choosing D and P is a matter of tuning, which we do after finding a suitable problem formulation.
Finally, we can constrain all decision variables to stay within specific domains. This is one of the unique benefits of MHE's numerical approach to estimation. Capacity estimation can benefit from this by posing the constraint

$$\tilde{Q} \in \mathcal{Q} = \left[\frac{1}{Q_{\max}}, \frac{1}{Q_{\min}}\right], \tag{3.84}$$

where $Q_{\max} = 42\mathrm{Ah}$ and $Q_{\min} = 30\mathrm{Ah}$ are natural limits within which our capacity estimate should lie. Similarly, we restrict our state estimates to the domain

$$x(i|k) \in \mathcal{X} = \mathbb{R} \times \mathbb{R} \times [0, 1], \tag{3.85}$$

meaning that we require the SoC to be between 0 and 1 at all times. With this we arrive at the problem formulation

$$\begin{aligned}
\min_{x(\cdot|k),\tilde{Q},e(\cdot|k)} \quad & e^T De + (x(k-N|k) - \hat{x}_{k-N})^T P(x(t-N|k) - \hat{x}_{k-N}) \\
\text{s.t.} \quad & x(i+1|k) = A_k x(i|k) + B_k u_i \\
& z(i+1|k) = z(i|k) + h\eta\tilde{Q}u_i \\
& y_i = h(x(i|k), z(i|k), u_i, e(i|k)) \\
& z(i|k) \in [0,1] \\
& \tilde{Q} \in \mathcal{Q}, \quad i \in \{k-N, \dots, k-1\}\,.
\end{aligned} \tag{3.86}$$

We can implement this formulation. However, there is one additional problem we face in making this setup work: Our ability to accurately estimate Q critically depends on the trajectories reflected in the MHE prediction horizon. We want trajectories that traverse the entirety of the state space in terms of SoC. With the step size h we use in the SoC estimator and using such trajectories, the number of decision variables skyrockets. We have $4(N+1)$ decision variables. Even when we charge and discharge the battery very rapidly, one cycle has about $N = 10^4$ steps in SoC estimation. Having as many as $4 \cdot 10^4$ decision variables is not feasible in an automotive application, especially considering that the problem is nonlinear. We tackle this challenge by choosing a larger step size for discretization in MHE, an approach known as downsampling. Downsampling can be dangerous by exacerbating unwanted discretization effects. This is a necessary evil here, since there is no other sensible way to reduce the problem size.
We choose a step size for MHE in the following manner: first, we choose a horizon N. Second, we let our battery system follow a suitably long trajectory, and divide that trajectory into N steps of step size h_{MHE}. In case we experience significant errors due to discretization effects, we can adapt N during tuning.
To implement this problem numerically, we first gather our decision variables in one vector

$$\xi = \begin{bmatrix} \tilde{Q} & x_{k-N}^T & z_{k-N} & x_{k-N+1}^T & z_{k-N+1} & e_{k-N+1} & \dots & x_k^T & z_k & e_k \end{bmatrix}^T. \tag{3.87}$$

Using this vector, we reformulate the optimization problem to

$$\begin{aligned}
\min_{\xi} \quad & \xi^T H \xi + f(\hat{x}_{k-N})\xi \\
\text{s.t.} \quad & A_{\text{eq}}(\hat{\theta})\xi = b_{\text{eq}} \\
& A_{\text{ineq}}(\hat{\theta})\xi = b_{\text{ineq}} \\
& \Upsilon(\xi;\hat{\theta}) = 0,
\end{aligned} \tag{3.88}$$

the matrices $H, f, A_{\text{eq}}, b_{\text{eq}}$ are presented in Appendix A. $\Upsilon(\xi;\hat{\theta})$ evaluates the nonlinear measurement constraint at N points given ξ and the trajectory $\hat{\theta}$ of parameters estimated by the SoC estimator.
We elaborate on the data exchange between the SoC and SoH estimators: the SoH estimator obtains the full trajectory of the parameters θ estimated by the SoC

estimator. The reasoning behind this is that we wish to capture the uncertainty of our capacity estimate only in the noise terms used in MHE. We do not wish to capture disturbances caused by uncertainty in the system's electrical parameters. For the states, we only take one initial estimate $\hat{x}_{k-N}$ from the SoC estimator, since we want a consistent fit of the capacity to our model. Beyond this, there is no information exchange between the SoC and SoH estimators.

3.2.2 Tuning

The tuning parameters of the MHE as presented in (3.86) are the weighing factors D and P in its cost function and the number of samples N. Here, large values in D encourage fitting our model predictions closely to the measured output trajectory. Large values in P favor small initial deviations from the estimate $\hat{x}_{k-N}$. Big values for N leads to decreased discretization effects, which is why we aim to choose N as large as numerically possible.
To get a good estimate of capacity, it is vital to fit our predicted trajectories especially well at the fringes of the SoC range (i.e. the low and high regions of the OCV-curve). To this end, we prepare to assign higher values to the d_i corresponding to the points with the highest and lowest values of terminal voltage y_i. After an iterative tuning process, we choose the values

$$d_{\text{top}} = 70 \tag{3.89}$$
$$d_{\text{bottom}} = 10^3 \tag{3.90}$$
$$d_{\text{else}} = 0.1 \tag{3.91}$$

for D, where d_{top} is assigned to points with a maximum in y_i, d_{bottom} is assigned to points with a minimum in y_i, and d_{else} to all other points. The larger choice of d_{bottom} is motivated by larger model and measurement errors that frequently occur in this range. This means that strong reliance on the modeling equations, as would be signified by small values in d_i, yields bad estimates.
We choose

$$P = \text{diag}(10^{-4}, 10^{-4}, 10^{-1}) \tag{3.92}$$

to allow for some variation in initial SoC $\hat{z}_{k-N}$ to curb the effects of potentially slightly incorrect SoC estimates at time $k-N$. For the number of samples considered, we find $N = 500$ to be a good trade off between accuracy and efficiency. This, however, strongly depends on the system we implement our dual estimator on and the computational restrictions that come with that system.
The quality of the SoH estimates is heavily dependent on the trajectories we let the algorithm explore. The importance of traversing the SoC range completely cannot be overstated. To illustrate this, consider the example of a trajectory with no change in SoC. The capacity Q is impossible to identify, since it does not change the estimated trajectories. Conversely, consider a trajectory where the SoC traverses its entire domain from 0 to 1. Throughout this the capacity will influence the derivative of the SoC. Thus, wrong values for the capacity will generate large measurement residuals by making the SoC diverge from its true value. This means that the capacity can be identified from this type of trajectory. For this reason, we only

estimate capacity for trajectories where the system has passed through the upper and lower 15% of the SoC range.
With this, our SoH estimator in the form of a capacity filter is set up and ready for validation.

4

Results

In the following chapter, we validate the dual SoC and SoH estimators presented in this thesis. We start by presenting our setup for validating the estimators using simulations and experimental data. We proceed by introducing some performance indices to help describe the estimators' performance. We then show the effectiveness of our estimators by applying them to the simulation and experimental setup, measuring their performance using the performance indices. The reference values needed to evaluate the performance indices and shown in plots are measured using high precision measurements under lab conditions. For simulated trajectories, they are taken directly from the simulation model.

4.1 Validation Setup

We begin by describing the setup used for validation. We introduce the following nomenclature for our different filtering implementations: A filter using the data sensitivity feature is prefixed DS-. A filter using the adaptive tuning mechanism is prefixed A-. For example, a dual extended Kalman filter using both augmentations is thereby termed ADSDEKF. We validate our estimators using two different plants and three performance indices.

4.1.1 Validation Plants

A preliminary validation of our implementation can be done by using a high-fidelity battery model. This serves as a preliminary validation for two reasons. First, since even this high-fidelity model makes some necessary abstractions from a real battery system and therefore produces less meaningful data than a real battery. Second, because we use a proprietary model for this purpose, which can't be discussed in terms of its structure.

We obtain a thorough validation of our estimators by using a lithium nickel manganese cobalt oxide ($LiNiMnCoO_2$ or NMC) battery for evaluation of the state estimators' performance. This is a type of lithium-ion cell. The battery we use has a nominal capacity of $Q_{nom} = 37Ah$, with a cell voltage ranging from $3 - 4.2V$. The battery has a nominal impedance of approximately $1m\Omega$ at a frequency of $1kHz$. The same type of cell is used in short term testing cycles for verifying SoC estimation and long term testing cycles for investigating SoH estimation.

4.1.2 Performance Indices

In this section, we introduce a variety of performance indices. These help us in quantifying, how well SoC is estimated by our estimators.

4.1.2.1 Root Mean Square Error

The first performance index we use is the root mean square error (RMSE). We compute the RMSE of our SoC estimate. Thus, we get the index

$$\tilde{z}_{RMS} = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(z_k - \hat{z}_k)^2} \tag{4.1}$$

Here, T denotes the total number of time steps in the experiment. Smaller values of the RMSE quantities designate good estimates. RMSE encapsulates the estimation quality over a whole trajectory in a single scalar value.

4.1.2.2 Infinity Norm of the Estimation Error

Next, we are interested in the maximum deviation of our SoC estimate from the true value throughout the entire trajectory. This is known as the infinity norm of the estimation error and designated by

$$|\tilde{z}|_\infty = \underset{k\in I_p}{max}\sqrt{(z_k - \hat{z}_k)^2}, \tag{4.2}$$

where $I_p = \{250, \ldots, T\}$. We omit the first 250 samples because we initialize our SoC estimator with significant error in SoC. We don't, however, want to capture this initial error in our performance index. 250 data points are equivalent to 20 seconds with a step size of 0.08 seconds.

4.1.2.3 Mean Error

We average the SoC error over the entire trajectory. To prevent corrupting the performance index, we again exclude the first 250 samples. This leads to a mean SoC error of

$$\mu_{\tilde{z}} = \frac{1}{T - 250}\sum_{I_p}(z_k - \hat{z}_k). \tag{4.3}$$

With this, we have introduced our readers to our validation setup and presented the tools to interpreting the results gained from it.

4.2 SoC Estimation Validation

In this section, we validate the dual estimators we have implemented with respect to their ability to estimate SoC. We subject our plants to load profiles and conditions characteristic of the application in EVs. This means that we have battery cycles with medium C-rates ($\approx 2C$).

4.2.1 Simulation Results

We start by running some simulations using the high-fidelity model for validation. This should count as a preliminary validation, since the high-fidelity model makes some necessary simplifications from a real battery system. It is therefore easier to perform state estimation.

4.2.1.1 DUKF vs DEKF

Our first test involves simulating a battery in an EV-like trajectory which moves the battery's state throughout the SoC range of approximately 90-20 %. We simulate a standard temperature of $T = 25°C$. In this test, we pit the standard DUKF against the standard DEKF.
Figures 4.1 and 4.2 show the estimated and actual SoC trajectories.
Figures 4.3 and 4.4 depict the SoC errors. We observe that both of the estimators are highly successful in removing the initial SoC error of 40%. We already suspect from this plot that the DUKF does a better job of keeping the SoC error low. This is confirmed by our performance indices as given in Table 4.1: DUKF outperforms DEKF in all performance indices. The most noticeable decrease therein is achieved in the mean of the error, which is $\mu_{\tilde{z}}^{UKF} = 0.32237\%$, which is a 67% improvement on the value $\mu_{\tilde{z}}^{EKF} = 0.98098\%$ scored by the DEKF. The DUKF also achieves a markedly lower SoC RMSE than the DEKF, cutting SoC RMSE of the DEKF by 48%.
In Figures 4.5 and 4.6 we see the measurement residual trajectories. Here we observe a stark difference between DEKF and DUKF. While the DUKF stays within the limits of $\pm 20mV$, and indeed $\pm 10mV$ of measurement residual for most of the simulation. In contrast to this, the DEKF exceeds even the limits of $\pm 40mV$ at some instances.
Overall, we see that the theoretical advantages of UKF over EKF as shown in Chapter 2 are now verified in our simulation.

Norm	DEKF	DUKF	Improvement
$\tilde{z}_{RMS}[\%]$	1.9033	0.99078	47.9441%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	4.339	4.1664	3.9779%
$\mu_{\tilde{z}}[\%]$	0.98098	0.32237	67.1380%

Table 4.1: This table contains the values of the performance indices describing the trajectories of DEKF and DUKF for the simulated EV trajectory.

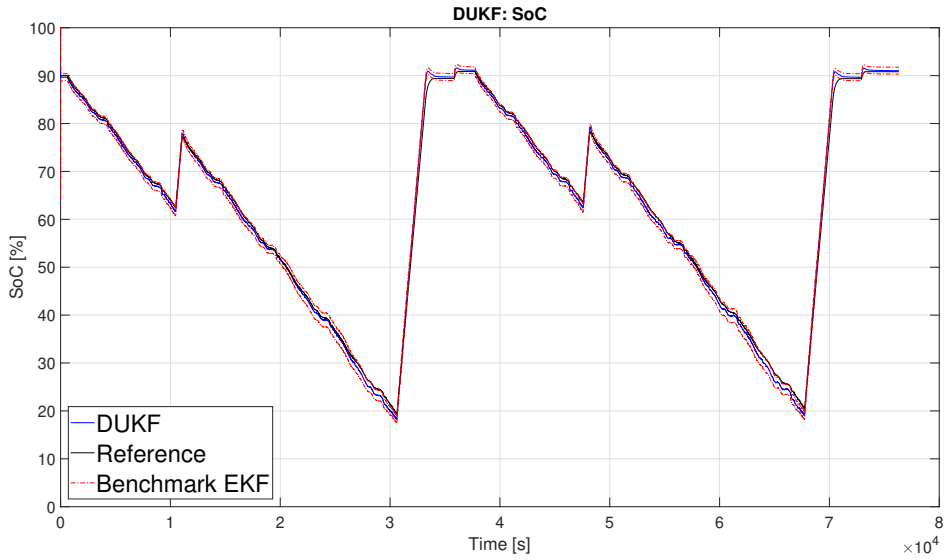

Figure 4.1: This figure shows SoC trajectories from the simulation with DUKF. The SoC of the simulation plant is shown in black, the SoC trajectory estimated by the DUKF in blue and the 3σ error bounds in red, dotted lines.

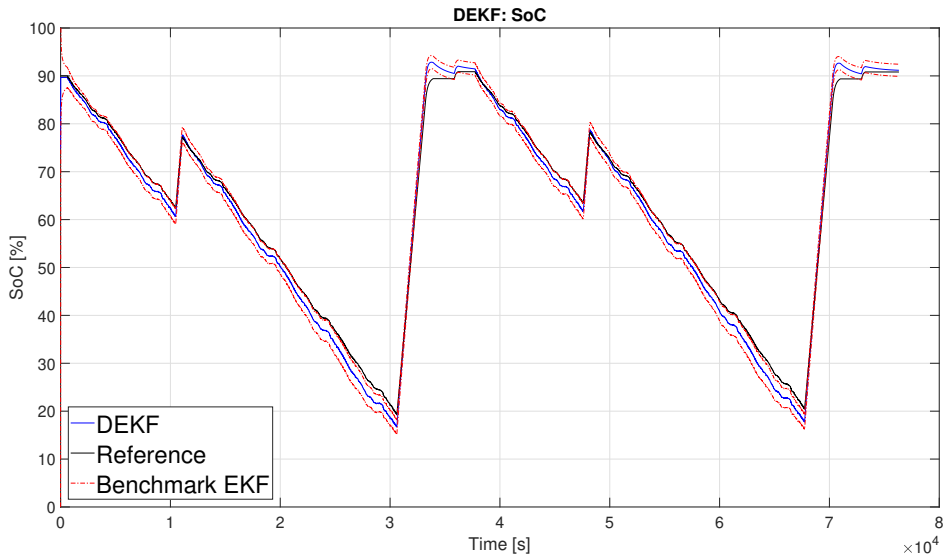

Figure 4.2: This figure shows SoC trajectories from the simulation with DEKF. The SoC of the simulation plant is shown in black, the SoC trajectory estimated by the DEKF in blue and the 3σ error bounds in red, dotted lines.

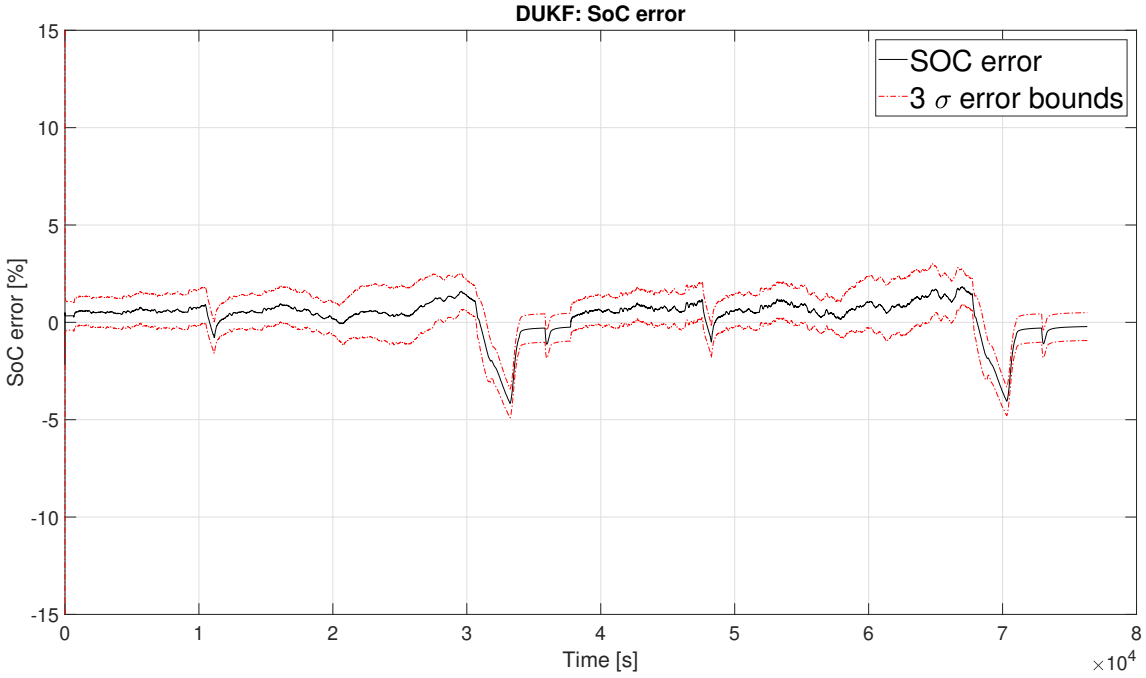

Figure 4.3: This figure shows the SoC error trajectory from the simulation with DUKF. The SoC error trajectory is shown in black and the 3σ error bounds in red, dotted lines.

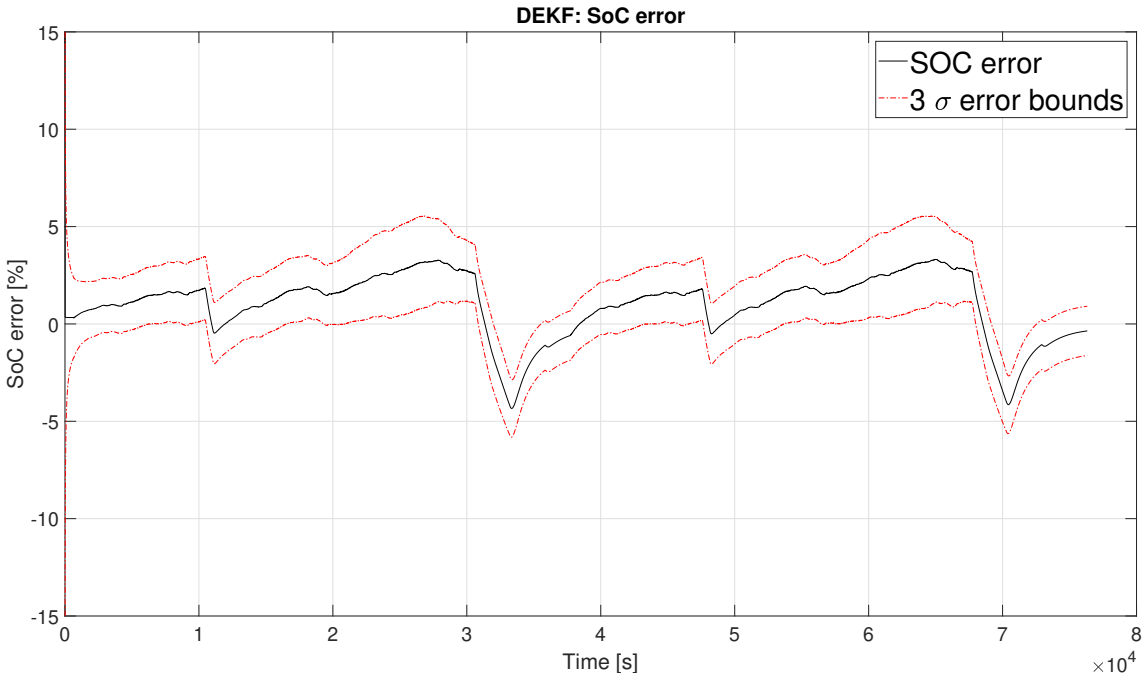

Figure 4.4: This figure shows the SoC error trajectory from the simulation with DEKF. The SoC error trajectory is shown in black and the 3σ error bounds in red, dotted lines.

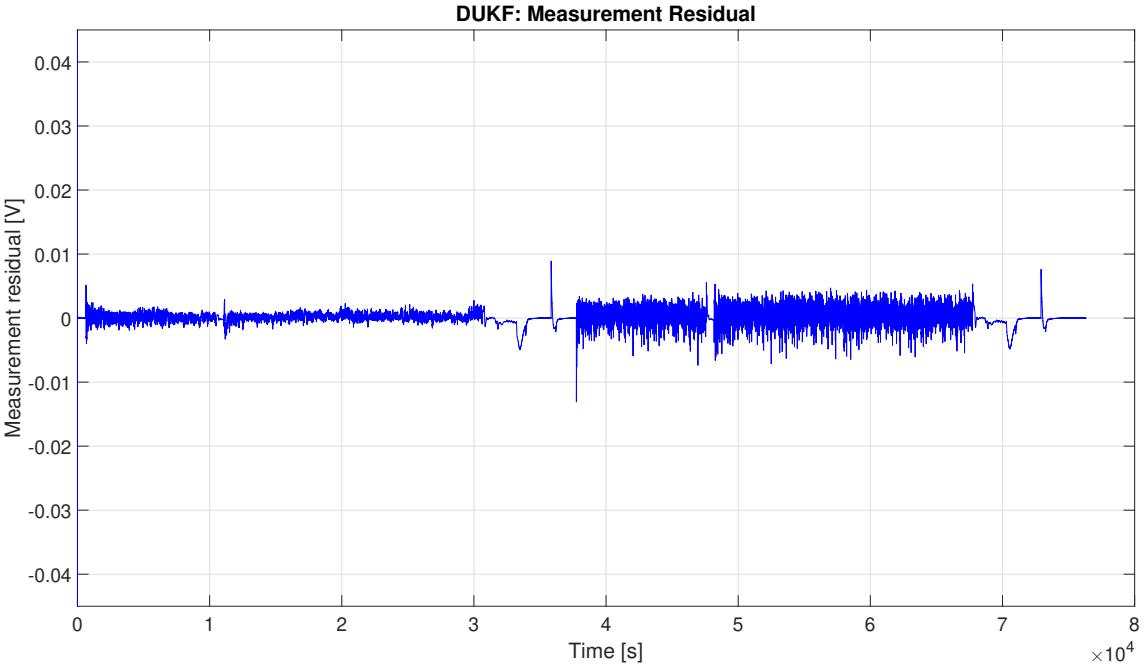

Figure 4.5: This figure shows the measurement residual trajectory from the simulation with DUKF.

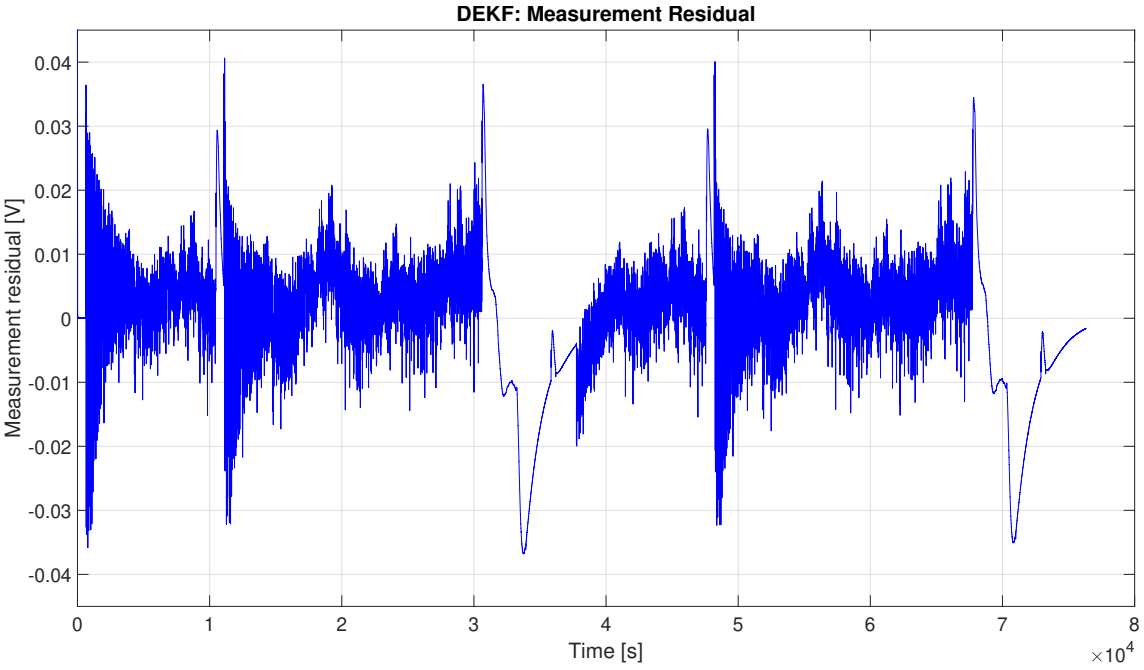

Figure 4.6: This figure shows the measurement residual trajectory from the simulation with DEKF.

4.2.1.2 DSDEKF vs DEKF

Next, we would like to validate the efficacy of our sensitivity analysis feature in use with the DEKF. To that end, we simulate the same EV load cycle we previously

used to compare DEKF and DUKF for two variations on the DEKF: We simulate once with our regular DEKF setup as used above, and once with the DSDEKF setup, that is making use of our data sensitivity check feature.
In Figures 4.7 and 4.8 we see the trajectories for the estimated resistances by both versions of the filter. We immediately note that they are strongly dissimilar.
R_0 is estimated as mildly decreasing in a smooth manner by the DSDEKF. The DEKF gives perfectly erratic estimates of R_0. In particular, its estimates of R_0 jump suddenly, whenever there is no system input. In addition, it estimates R_0 as negative for well over half the simulation time, which compromises the validity of our model.
R_1 is taken to decrease smoothly by the DSDEKF, while increasing in small jumps in the version without sensitivity feature. We note that these small jumps happen at the same time as the jumps in R_0. It seems probable that the bad estimate in R_0 has a corrupting effect on the estimate of R_1. To investigate this further, we look at the plots of the sensitivity of R_0 and R_1 gathered from the sensitivity analyzer. They are shown in Figures 4.11 and 4.12. This confirms our suspicion: shortly after time $t = 10^4$, the sensitivity analyzer prevents the parameter filter from updating its estimate for R_0, as sensitivity for that parameter is zero. At the same time, R_1 enjoys a period of high sensitivity. In Figure 4.8 we can see that at this exact time, both R_0 and R_1 change suddenly.
R_2 changes little in the estimation of both filters.
The trajectories for the estimated time constants can be seen in Figures 4.9 and 4.10.
We observe that the estimates of τ_1 remain nearly constant in both versions. τ_2 follows a relatively smooth downward trajectory in DSDEKF. In DEKF, τ_2 is changed quite abruptly and increases. We check the plots of the sensitivity of τ_2 in Figure 4.13 to corroborate our suspicion that this is due to episodes of low sensitivity, where the DSDEKF freezes its estimate of τ_2 and the DEKF simply gives a bad estimate.
Table 4.2 gives the values of our performance indices for this experiment. We see that the low quality of the parameter estimates of the DEKF lead to a deterioration of its SoC estimates, overall. The SoC RMSE increases markedly, we observe a slight increase in the SoC infinity norm and the mean SoC estimate error moves farther away from zero. In short, the DSDEKF scores better in all three performance indices.
We can conclude that the data sensitivity analysis feature verifiably improves the SoC estimates of the DEKF.

Norm	DEKF	DSDEKF	Improvement
$\tilde{z}_{RMS}[\%]$	1.9033	1.5563	18.2315%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	4.339	4.233	2.4430%
$\mu_{\tilde{z}}[\%]$	0.98098	0.66141	32.5766%

Table 4.2: This table contains the values of the performance indices describing the trajectories of DSDEKF and DEKF for the simulated EV trajectory.

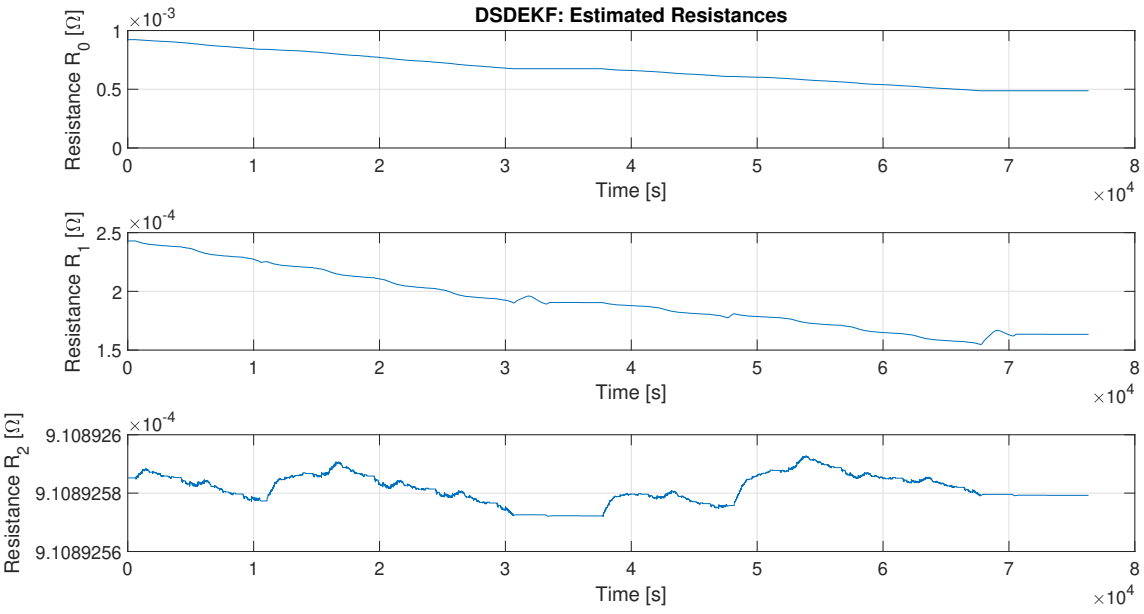

Figure 4.7: This figure depicts the resistances used in the ECM as estimated by the DSDEKF.

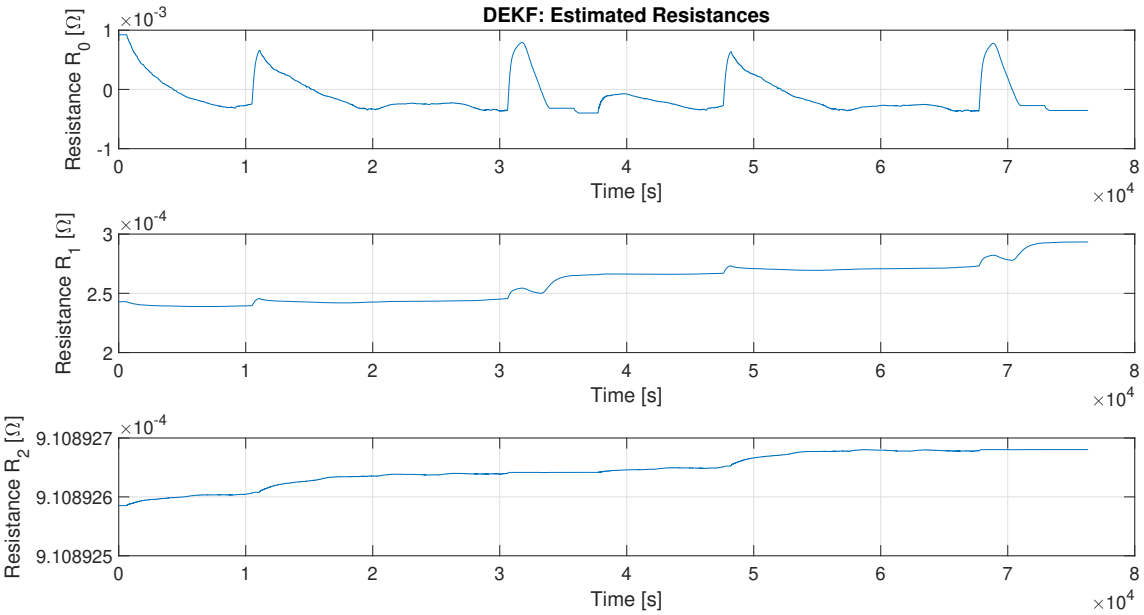

Figure 4.8: This figure depicts the resistances used in the ECM as estimated by the DEKF.

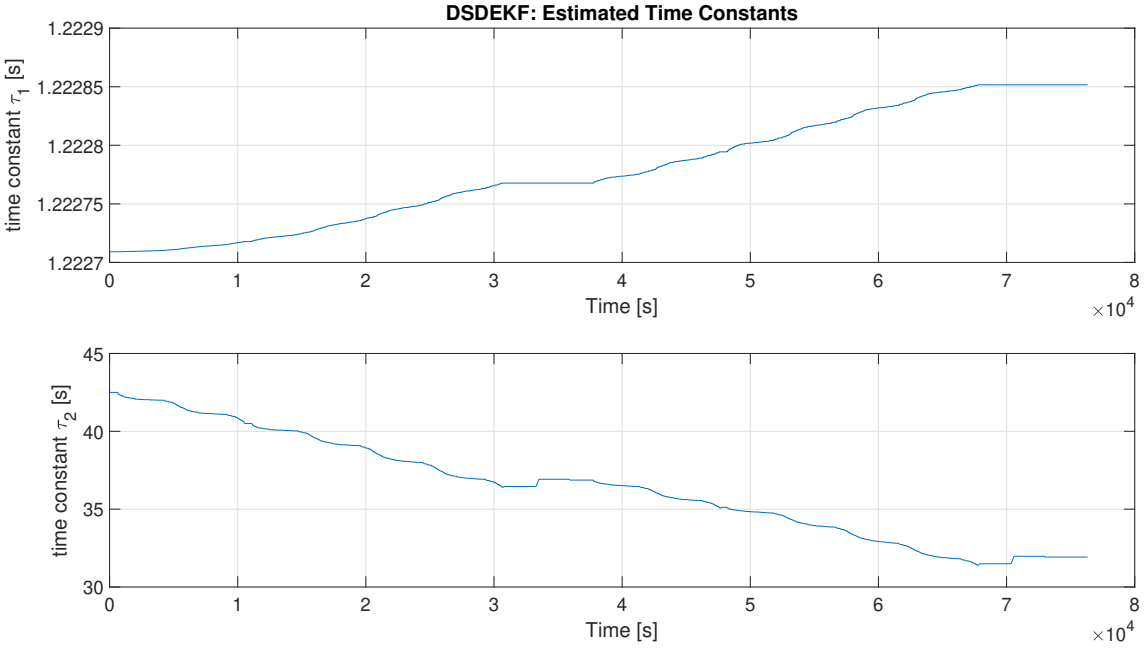

Figure 4.9: This figure depicts the time constants used in the ECM as estimated by the DSDEKF.

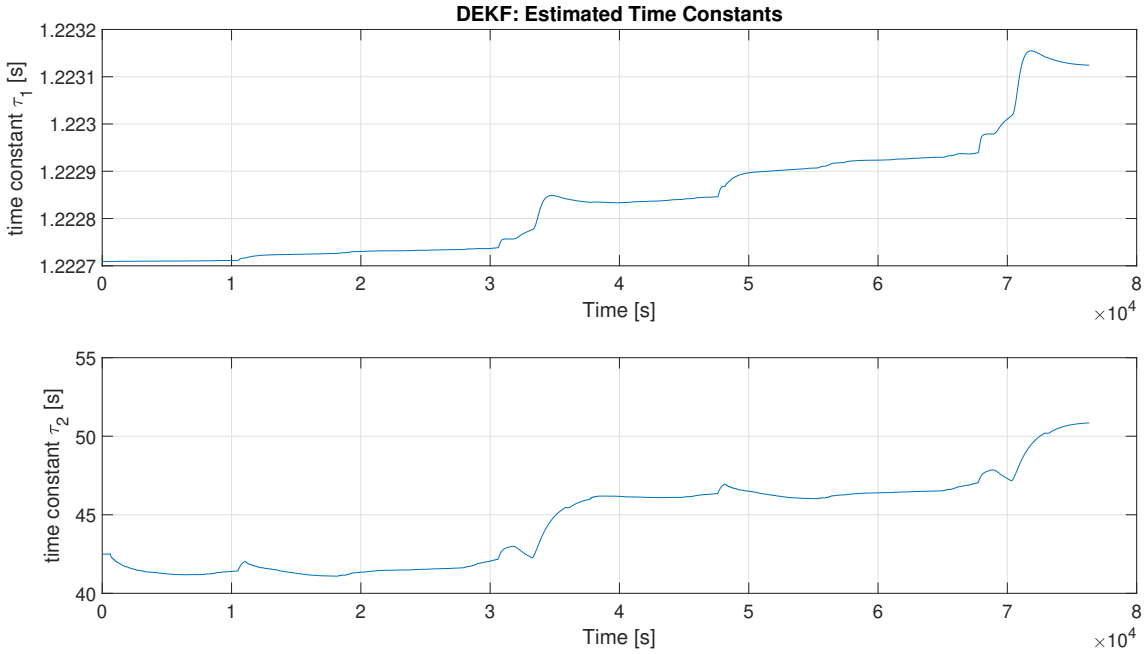

Figure 4.10: This figure depicts the time constants used in the ECM as estimated by the DEKF.

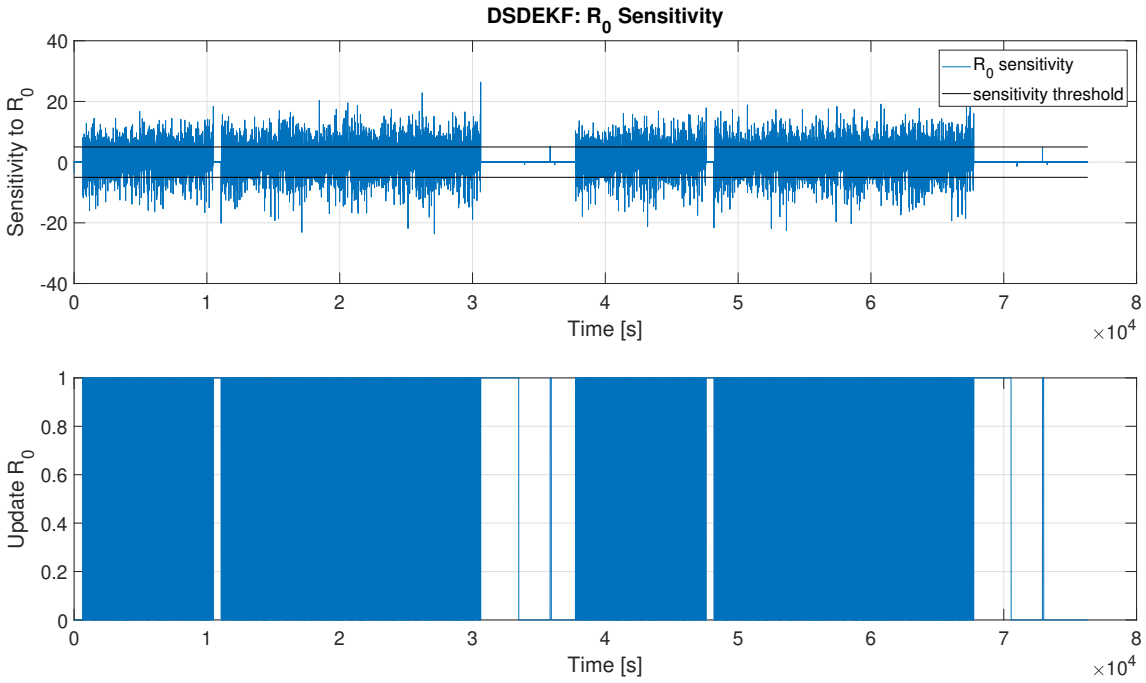

Figure 4.11: This figure shows the sensitivity with respect to R_0 in the simulation of DSDEKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

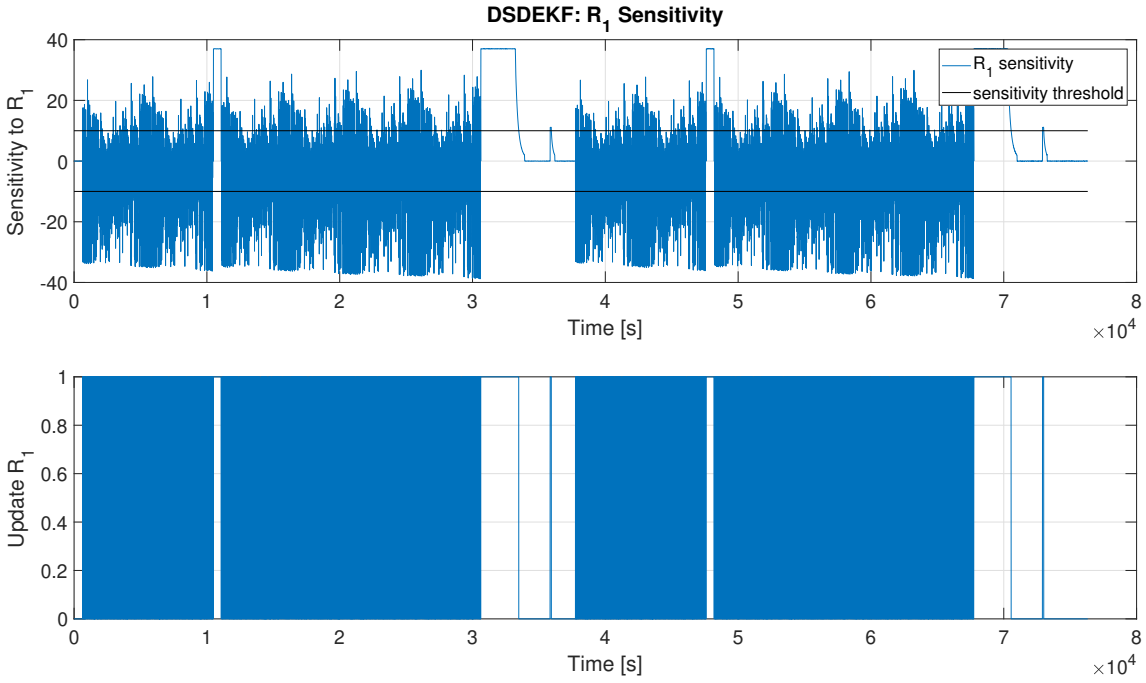

Figure 4.12: This figure shows the sensitivity with respect to R_1 in the simulation of DSDEKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

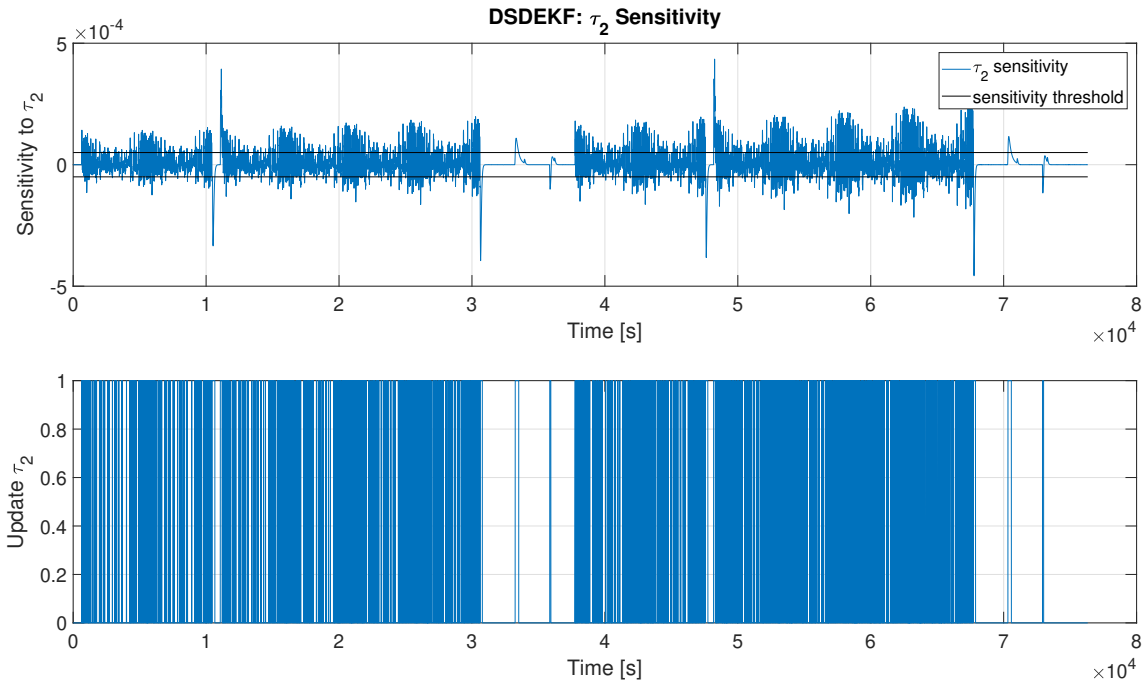

Figure 4.13: This figure shows the sensitivity with respect to τ_2 in the simulation of DSDEKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

4.2.1.3 DSDUKF vs DUKF

Our next step is to validate the data sensitivity analyzer in conjunction with the DUKF. We simulate the same trajectory as used above to validate the DSDEKF. We simulate both the DUKF and the DSDUKF, in order to verify that we get performance improvements from the data sensitivity feature.
In Figures 4.14 and 4.15 we see the trajectories for the estimated resistances by both versions of the filter.
R_0 is estimated as mildly decreasing in a smooth manner by the DSDUKF. The DUKF, much like the DEKF, gives erratic estimates of R_0. In particular, its estimates of R_0 jump suddenly, whenever there is no system input. This is prevented in the DSDUKF, which gives a much smoother estimate of R_0
The change in the estimates of R_1 is less clear. The DUKF estimates R_1 to be negative at one point where there is no input to the battery. It estimates R_1 to be zero at another. The DSDUKF fails to prevent this, instead predicting negative values at both points. This is despite the fact that the filters are tuned to change the parameters of the ECM slowly, when compared to the DEKF and DSDEKF.
The estimations of R_2 change little between the DUKF and DSDUKF. Figure 4.20 shows the sensitivity with respect to R_2 and when we update our estimate of R_2 in the DSDUKF. We confirm that the data sensitivity feature is active, even though this does not seem to manifest itself clearly in the trajectory of R_2.
The trajectories for the estimated time constants can be seen in Figures 4.16 and 4.17. They both change relatively little.
Table 4.3 gives the values of our performance indices for this experiment. Despite not improving the trajectories of the parameter estimates as much as the DSDEKF, the DSDUKF scores better than the DUKF in all performance indices. Improvements are slight, ranging from 4 to 6%.
We can conclude that the data sensitivity analysis feature improves the SoC estimates of the DUKF. It becomes clear from the parameter trajectories, that this technology requires further research in conjunction with DUKF, since the parameter estimates do not immediately suggest higher estimation quality.

Norm	DUKF	DSDUKF	Improvement
$\tilde{z}_{RMS}[\%]$	0.99078	0.93313	5.8186%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	4.1664	3.9119	6.1084%
$\mu_{\tilde{z}}[\%]$	0.32237	0.30848	4.3087%

Table 4.3: This table contains the values of the performance indices describing the trajectories of DSDUKF and DUKF for the simulated EV trajectory.

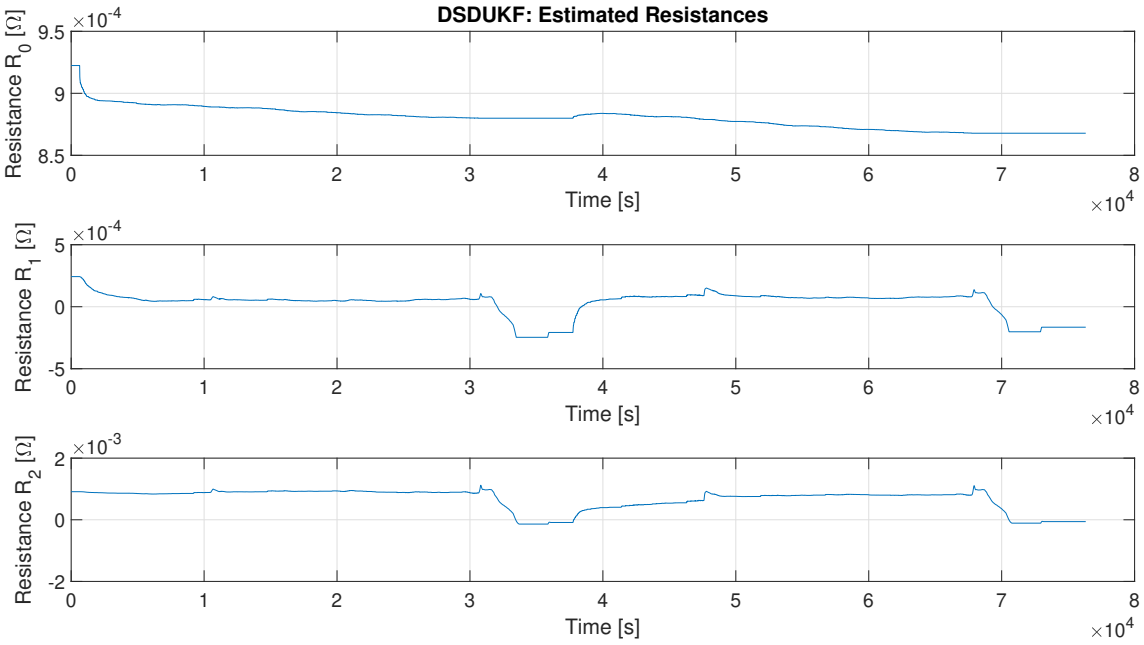

Figure 4.14: This figure depicts the resistances used in the ECM as estimated by the DSDUKF.

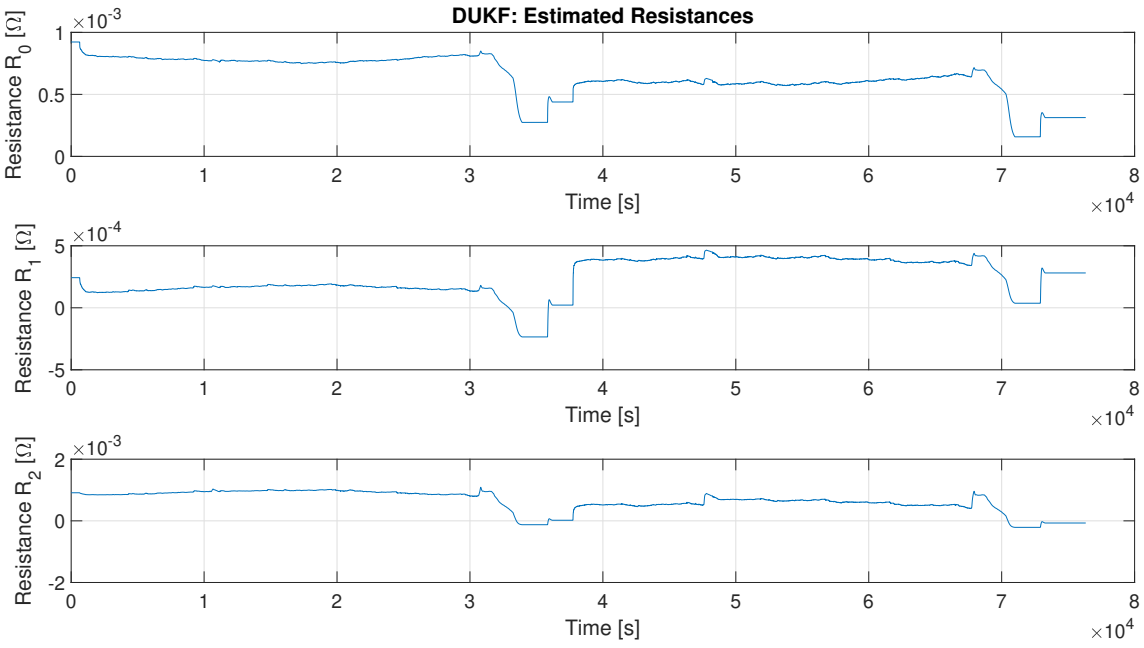

Figure 4.15: This figure depicts the resistances used in the ECM as estimated by the DUKF.

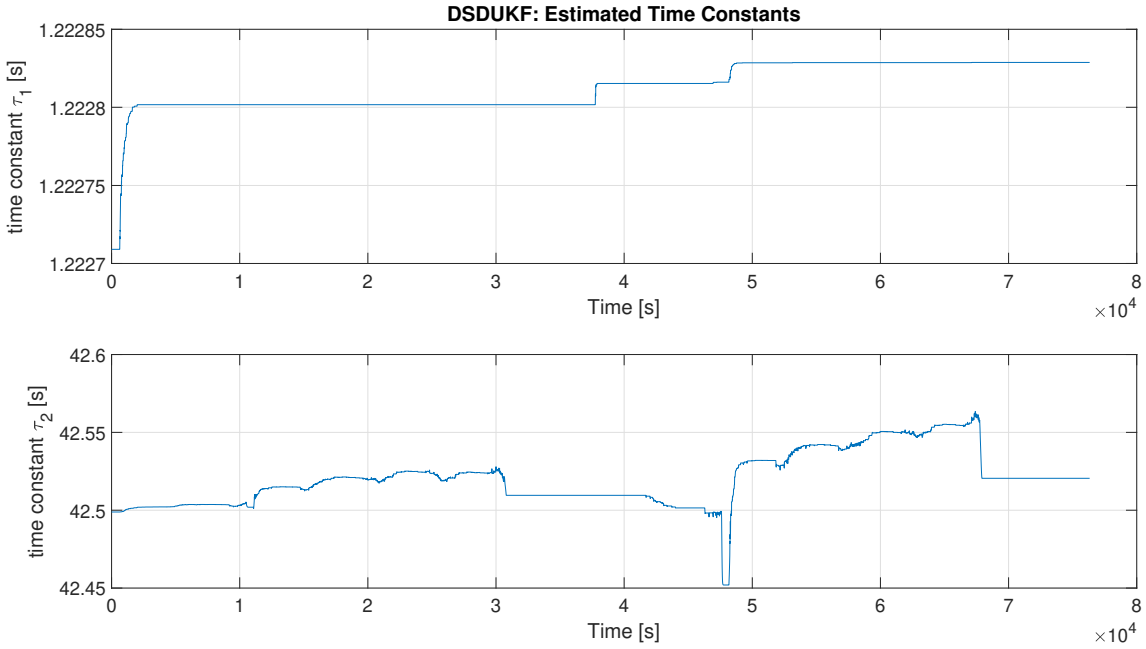

Figure 4.16: This figure depicts the time constants used in the ECM as estimated by the DSDUKF.

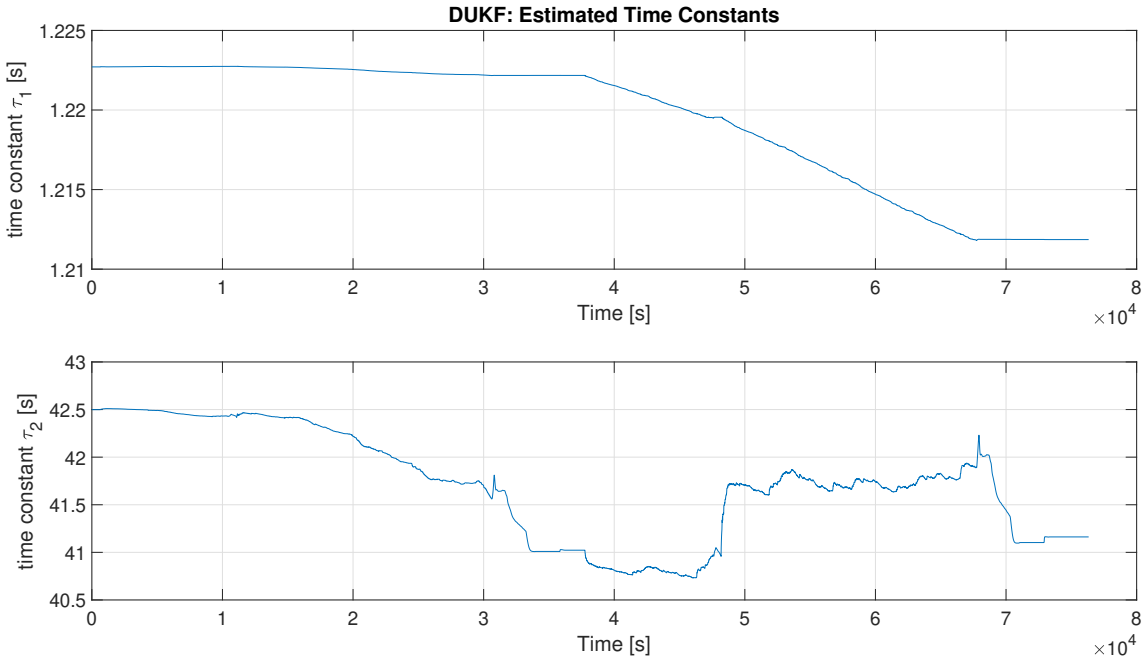

Figure 4.17: This figure depicts the time constants used in the ECM as estimated by the DUKF.

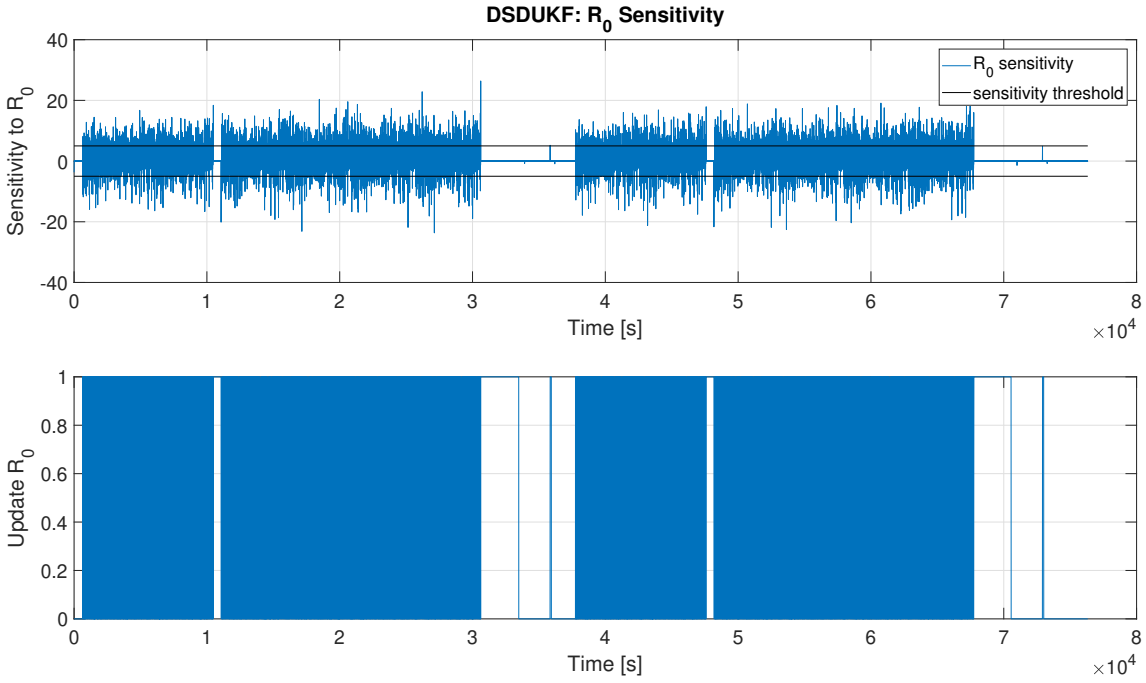

Figure 4.18: This figure shows the sensitivity with respect to R_0 in the simulation of DSDUKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

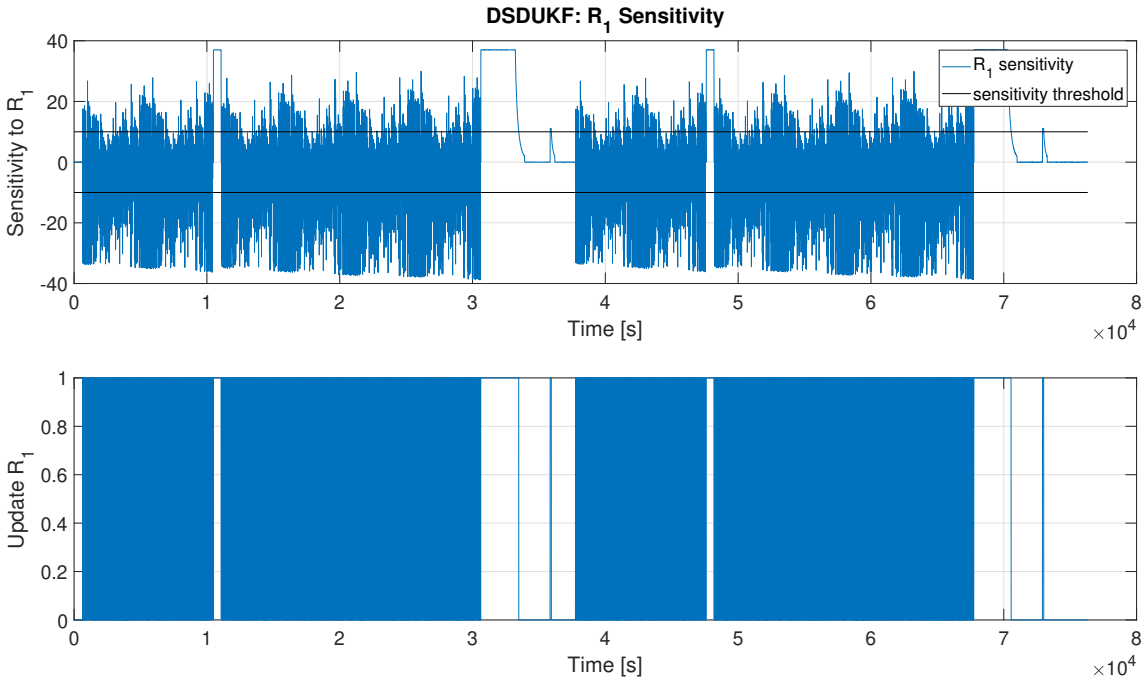

Figure 4.19: This figure shows the sensitivity with respect to R_1 in the simulation of DSDUKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

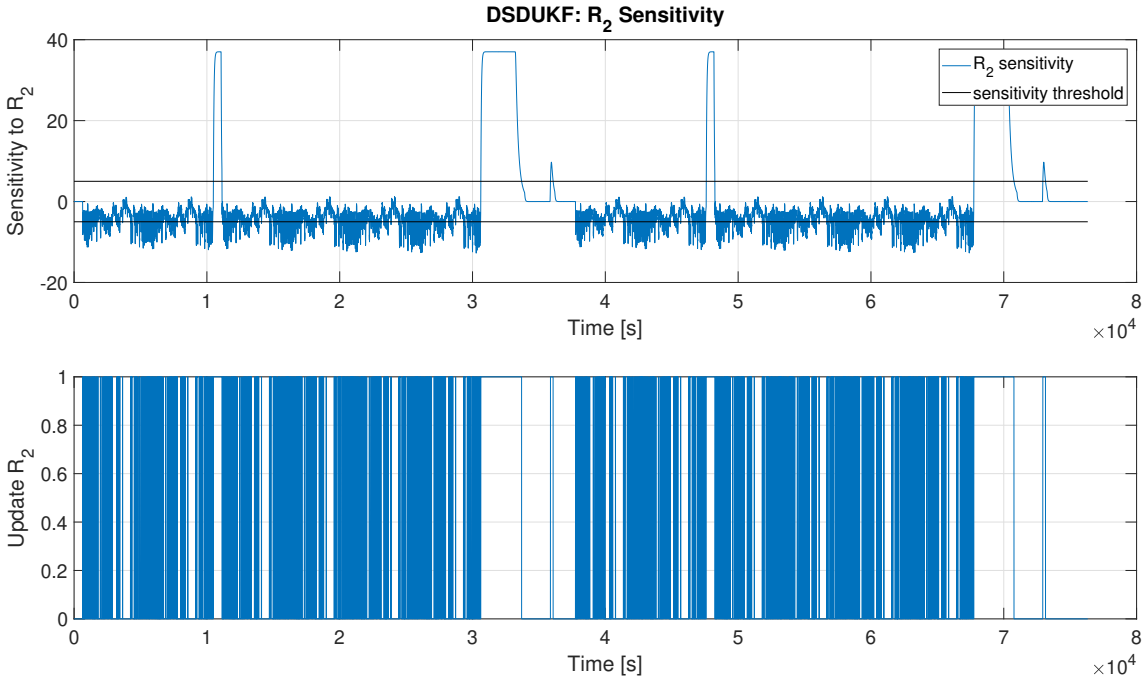

Figure 4.20: This figure shows the sensitivity with respect to R_1 in the simulation of DSDUKF. The upper plot shows the sensitivity as calculated in our sensitivity feature. The lower plot shows a value of 1, whenever the sensitivity feature decides to update the parameter estimate and 0, whenever it decides to freeze the estimate.

4.2.2 Experimental Results

Having performed an initial validation of our estimators using simulations, we want to verify their performance in the application on a real battery system. We use the NMC battery described in 4.1.1. In this section we also validate the adaptivity feature designed in this thesis. We have previously remarked on its susceptibility to sensor bias. Because of this, we use a high-resolution, bias-free sensor to operate the adaptive filters. To demonstrate their ability to deal with otherwise noisy measurements, we add Gaussian white noise to this precise measurement.
We benchmark all our filters against a state EKF, which looks up the ECM's parameters in lookup-tables generated in lab tests. These lab tests constitute an enormous effort, and should be seen as a big advantage over our dual filters in terms of information available to the filter. In terms of development, however, they are a disadvantage because of the additional cost and effort associated with them. We remind our readers that the dual filters developed in this thesis perform online parametric estimation and work without prior experimental data. We refer to the benchmark filter as bEKF for brevity of notation.

4.2.2.1 ADSDEKF vs DEKF vs benchmark EKF

In the first experiment, we validate the DEKF under real conditions. We also validate the version of adaptivity feature we developed for use with the DEKF, by pitting our regular DEKF against a DEKF enhanced by the adaptivity and data

sensitivity check features. For convenience, we refer to the DEKF with adaptivity and data sensitivity check feature as ADSDEKF. We use a similar load profile as previously, where the battery starts at a SoC of 91%, goes down to 13% and back to its initial state again. Lab tests were conducted at a temperature of $T = 37°C$.

Figures 4.21 and 4.22 show the battery current and voltage measurement values supplied as inputs to our filters. The current data given to the DEKF has a bias. The ADSDEKF receives bias-free, more noisy data. To give a fair comparison, we run the benchmark filter with both data sets and compare our filters to values given by the benchmark filter using the same inputs.

Figures 4.23 and 4.24 illustrate the SoC values estimated by the two filters. We see in Figure 4.23 that the DEKF's performance is very similar to the benchmark EKF and stays similarly close to the reference trajectory. Figure 4.26 shows us that the ADSDEKF gives a much improved SoC estimate compared to the benchmark filter.

Both the benchmark filter and the DEKF increase in their deviation from the reference SoC in the lower SoC ranges. This can be seen more clearly in Figure 4.25, where SoC error is depicted.The ADSDEKF avoids this almost altogether as can be seen in the corresponding SoC error plot in Figure 4.26. We conjecture that this is due to better handling of poor measurement information in the lower SoC ranges. To verify this, we look at the trajectories of the measurement residuals in Figures 4.27 and 4.28. We observe that the ADSDEKF's measurement residual develops a bias, whenever the SoC goes to its lower values. This behavior is not present in DEKF. The DEKF's measurement residual does not change qualitatively in this region. We conclude from this, that the accuracy of our measurement equations is poor in this region. To explain why the ADSDEKF performs better, we look at the tuning values used in the noise covariance matrices of the ADSDEKF. They are plotted in Figure 4.29. We observe that the adaptivity feature tunes the covariance matrix entries corresponding to the SoC to be very small at the lower end of the SoC spectrum. This estimate then increases by a factor of more than 100 as soon as the SoC rises again. This means that the adaptivity feature tunes the filter to take its own predictions into account more than the measurement at the lower edge of the SoC range, where measurement quality in this experiment is poor. Once measurement quality increases again, the filter is tuned to adapt more strongly towards the measurement information again. Sensor noise covariance estimates by the adaptivity feature undergo a mirrored development. They rise in the lower SoC regions, which leads to the measurement's influence on the state estimate diminishing. They plummet when the battery system exits the lower SoC regions to push the state estimate to reflect the measurement more closely. As a side effect of this we see the measurement residual of the ADSDEKF, as shown in Figure 4.28 increase when the SoC is at its lowest points and converge to zero again once the SoC rises. The measurement residual of the DEKF, as shown in Figure 4.27 displays uniform behavior throughout the experiment.

The performance indices in Table 4.4 show that the DEKF, despite working with less information, gives slightly even better SoC estimates than the benchmark filter. The DEKF's SoC RMSE is 0.7% lower than that of the benchmark filter. The maximum SoC error of the DEKF's estimates is a 43% improvement on that of the benchmark EKF. On average, however, the benchmark EKF stays decisively closer

to the true SoC. This can be seen from the smaller magnitude of its average SoC error.
The ADSDEKF gives drastically improved performance indices from the benchmark EKF. The SoC RMSE is improved by 51%. The maximum SoC error magnitude is 43% lower than that of the benchmark filter. On average the ADSDEKF's SoC estimate stays very close to the true SoC, improving the benchmark filter's mean error by 69%.

Sensor quality	low			high		
Norm	DEKF	bEKF	Impr.	ADSDEKF	bEKF	Impr.
$\tilde{z}_{RMS}[\%]$	1.5449	1.5559	0.7070%	0.49239	0.99909	50.7162%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	3.2688	5.7262	42.9150%	2.1754	4.0123	45.7817%
$\mu_{\tilde{z}}[\%]$	-1.3752	0.35357	-288.9470%	-0.17149	-0.56093	69.4276%

Table 4.4: This table contains the values of the performance indices describing the performances of DEKF and ADSDEKF. The data is generated from an experiment. The experiment consists of discharging the battery from 91% to 13% and charging it back to 91%.

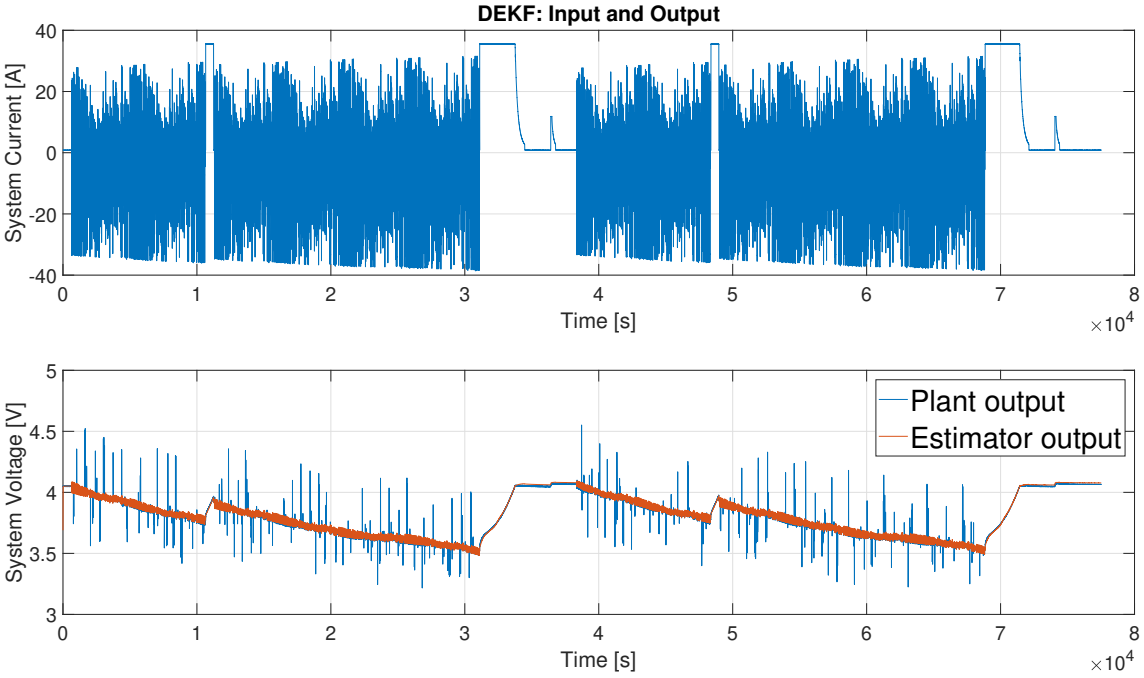

Figure 4.21: This figure shows the system inputs and outputs for the validation experiment as provided to the DEKF. Low-resolution sensors are used. The current measurement displays bias.

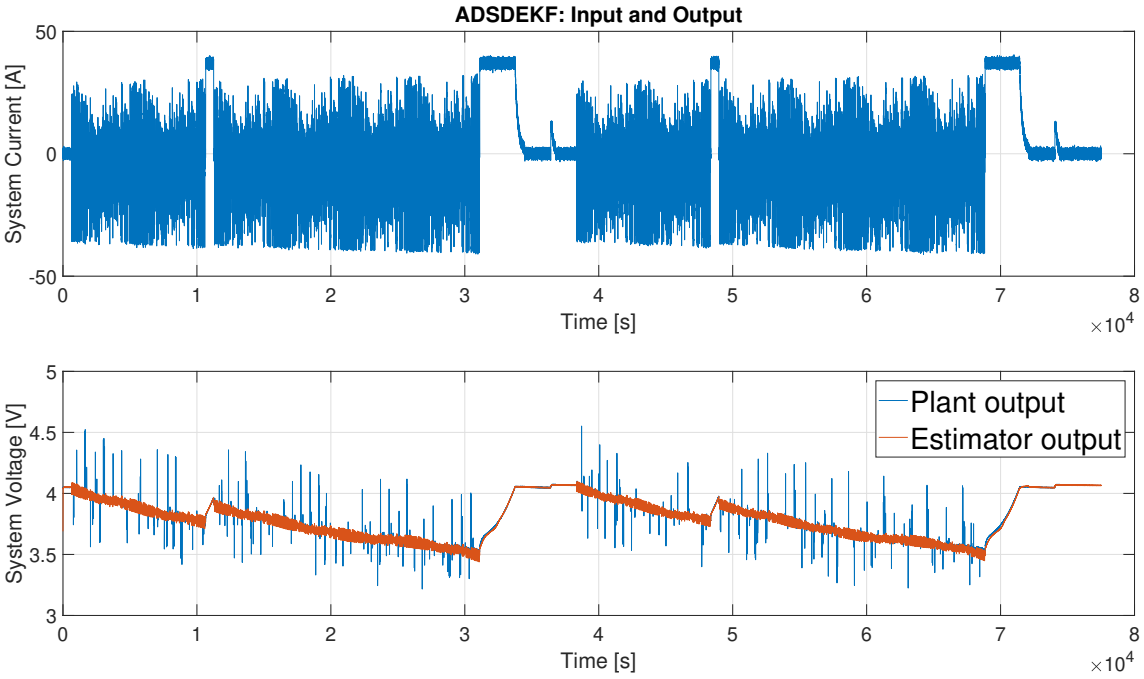

Figure 4.22: This figure shows the system inputs and outputs for the validation experiment as provided to the ADSDEKF. Although a high resolution current sensor is used to avoid bias, zero-mean Gaussian noise is artificially added to challenge the filter and see how it handles this kind of measurement uncertainty, at least.

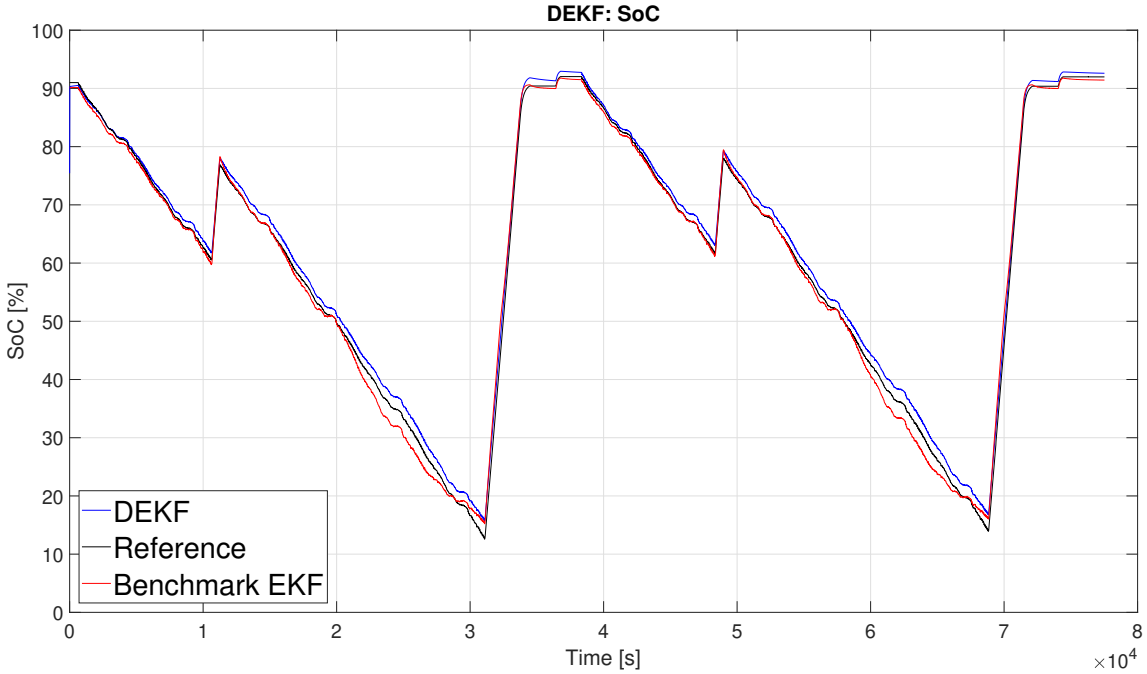

Figure 4.23: This figure shows the SoC trajectory as estimated by the DEKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF.

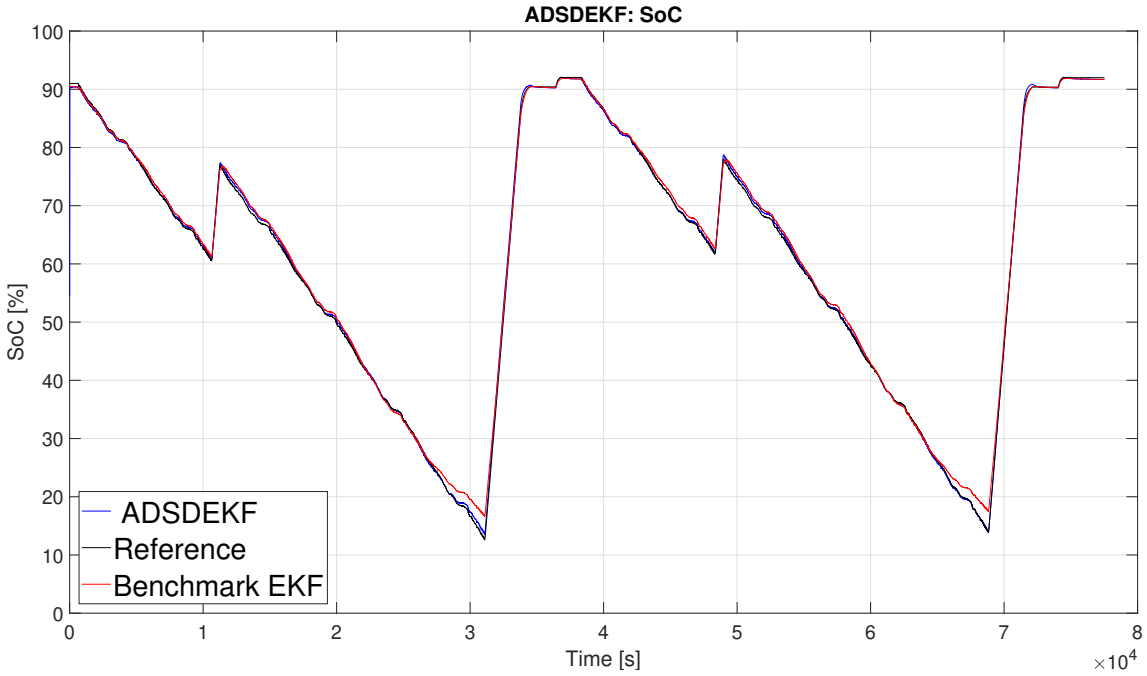

Figure 4.24: This figure shows the SoC trajectory as estimated by the ADSDEKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF.

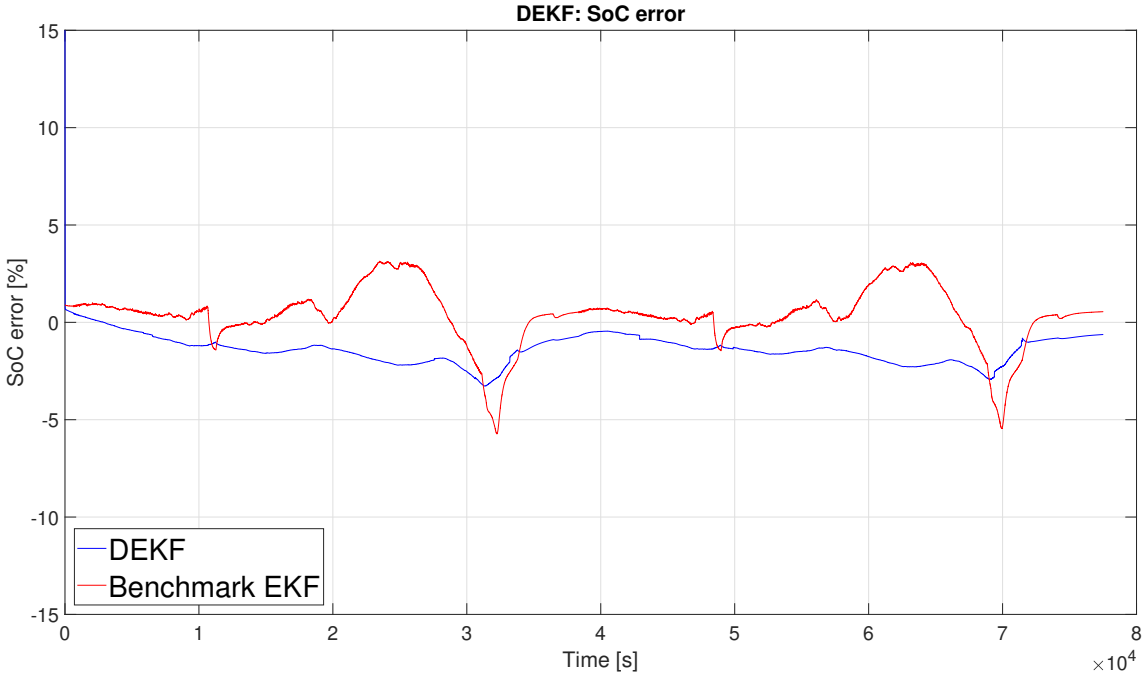

Figure 4.25: This figure shows the SoC error trajectory of the DEKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

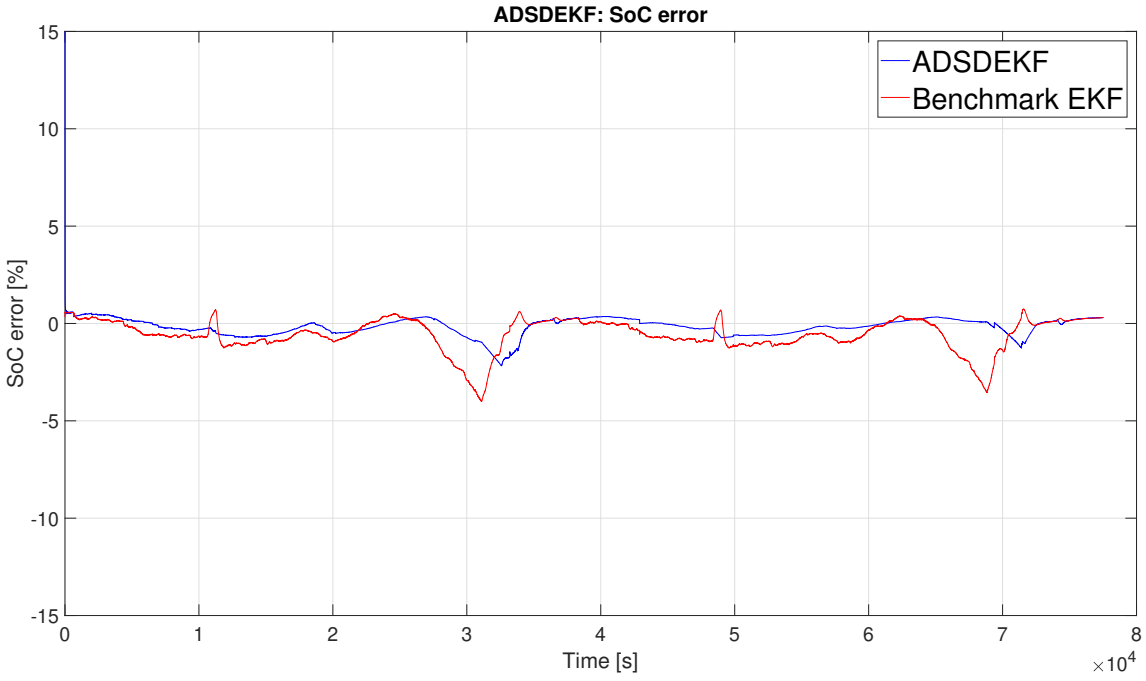

Figure 4.26: This figure shows the SoC error trajectory of the ADSDEKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

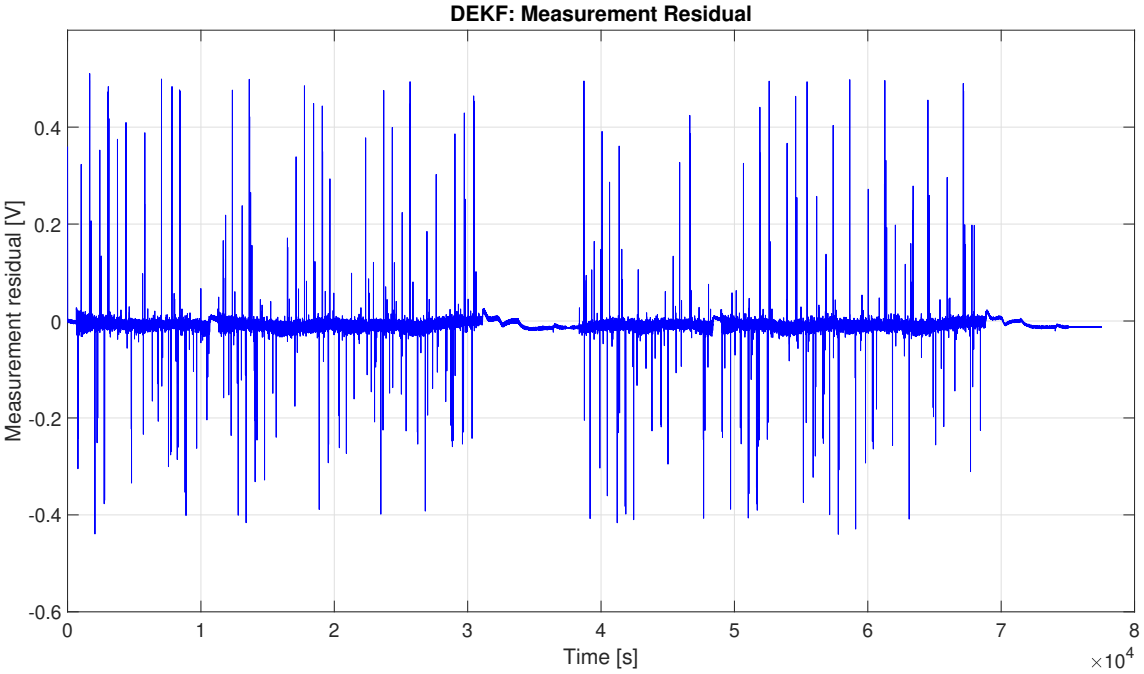

Figure 4.27: This figure shows the measurement residual of the DEKF using experimental data.

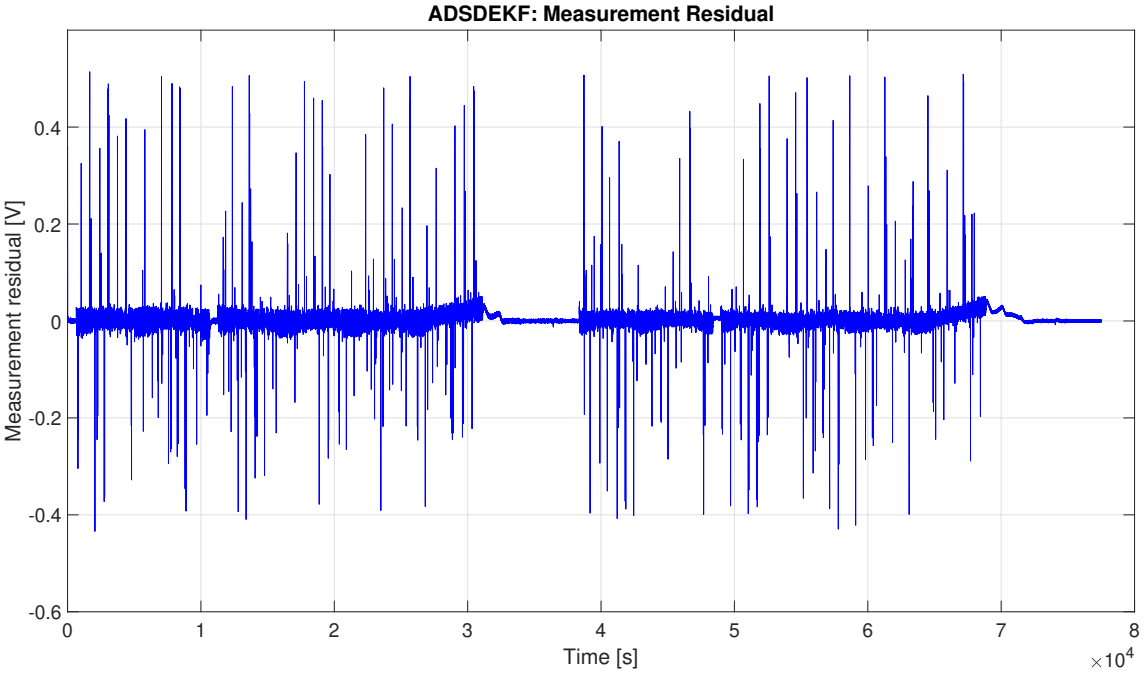

Figure 4.28: This figure shows the measurement residual of the ADSDEKF using experimental data.

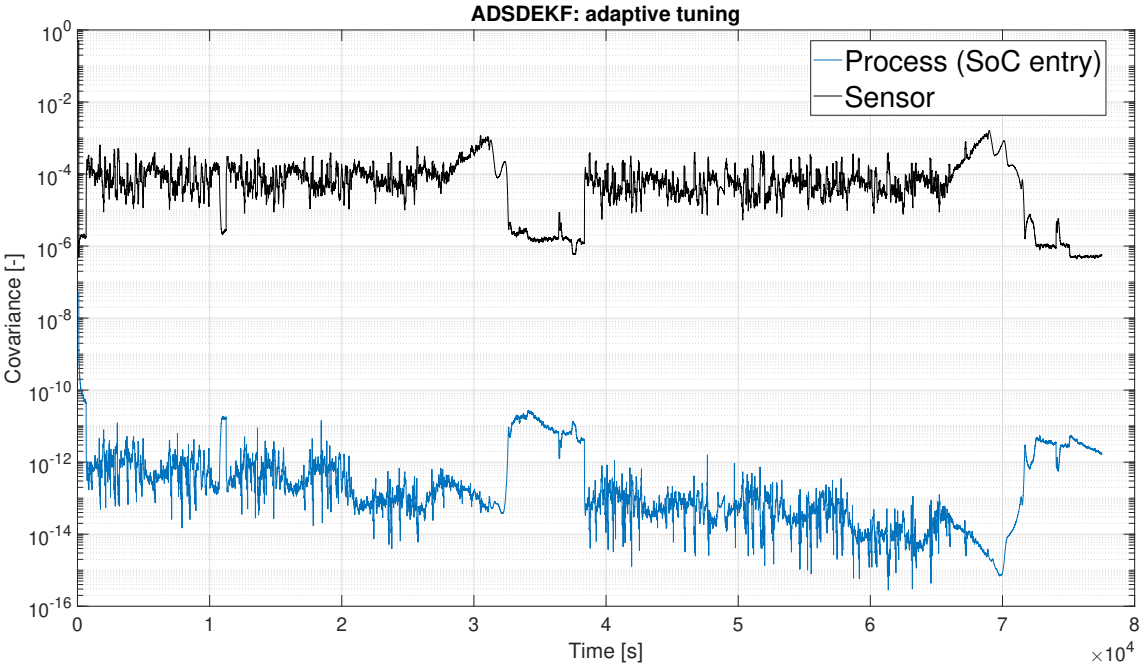

Figure 4.29: This figure shows the plots of the sensor noise covariance and the diagonal entry of the process noise covariance matrix corresponding to the SoC as approximated by the adaptivity feature. Sensor noise covariance is shown in black, process noise covariance is shown in blue. The vertical axis is logarithmic.

4.2.2.2 ADSDUKF vs DUKF vs benchmark EKF

In the second experiment, we validate the DUKF under real conditions. We again validate the adaptivity and sensitivity analysis feature developed for use with the DUKF by running the ADSDUKF and DUKF in parallel.
We use the same experimental data as in the previous experiment. This means that the benchmark filters performance in this experiment is identical to its performance in the previous experiment. The input data given to the DUKF is the same as shown in 4.21. The ADSDUKF receives the same, bias-free current data with additional noise as the ADSDEKF. This data is shown in Figure 4.22. Both the ADSDUKF and the DUKF are always compared with the benchmark EKF receiving the same inputs they receive.
The SoC trajectories are shown in Figures 4.30 and 4.31. The DUKF performs markedly better than the benchmark filter. Its SoC estimates remains closer to the true SoC especially in the region of $50\% - 20\%$ SoC. The higher quality of the DUKF's SoC estimates can be seen more clearly in Figure 4.32, where the SoC error of the DUKF and the benchmark filter is shown. The DUKF's SoC estimation error stays closer to zero at nearly all points. Notably, the DUKF also outperforms the DEKF in this experiment, which results from the theoretical advantages presented in chapter 2.
In Figure 4.31 we see that the ADSDUKF displays improved SoC estimates compared to the DUKF in the lower SoC regions. The trajectory looks similar to the ADSDEKF's SoC trajectory for the same experiment.
The plot of the SoC estimate error of the ADSDUKF is shown in Figure 4.33.

We observe that the ADSDUKF gives decisively lower errors than the benchmark filter. We see a tendency of the ADSDUKF to generate a bias in the measurement residual and tune the noise covariance matrices towards preferring model information over measurement information in the lower SoC ranges. This is analogous to the phenomenon described in more detail in section 4.2.2.1. A plot of the measurement residual can be seen in Figure 4.35. The relevant adaptive tuning matrix entries are plotted in Figure 4.36.
Table 4.5 contains the performance indices describing the filter performances in this experiment. The DUKF achieves a significantly lower values than the benchmark EKF in SoC RMSE. On average, its SoC estimate also stays 34% closer to the true SoC. In addition, the DUKF gives a maximum SoC error that is 35% lower than the benchmark EKF's maximum SoC error. As indicated by the SoC and SoC error trajectories, the performance indices show that the adaptive tuning mechanism improves the performance of the DUKF in every measure, with improvements over 50% on the benchmark filter in all categories. This pushes the ADSDUKF's performance beyond that of the ADSDEKF in all three of the performance indices, further corroborating the UKF algorithms superiority.

Sensor quality	low			high		
Norm	DUKF	bEKF	Impr.	ADSDUKF	bEKF	Impr.
$\tilde{z}_{RMS}[\%]$	1.0331	1.5559	33.6011%	0.43909	0.99909	56.0510%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	3.7283	5.7262	34.8905%	1.6489	4.0123	58.9039%
$\mu_{\tilde{z}}[\%]$	-0.23627	0.35357	33.1759%	-0.10503	-0.56093	81.2757%

Table 4.5: This table contains the values of the performance indices describing the performances of DUKF and ADSDUKF. The data is generated from an experiment. The experiment consists of discharging the battery from 91% to 13% and charging it back to 91%.

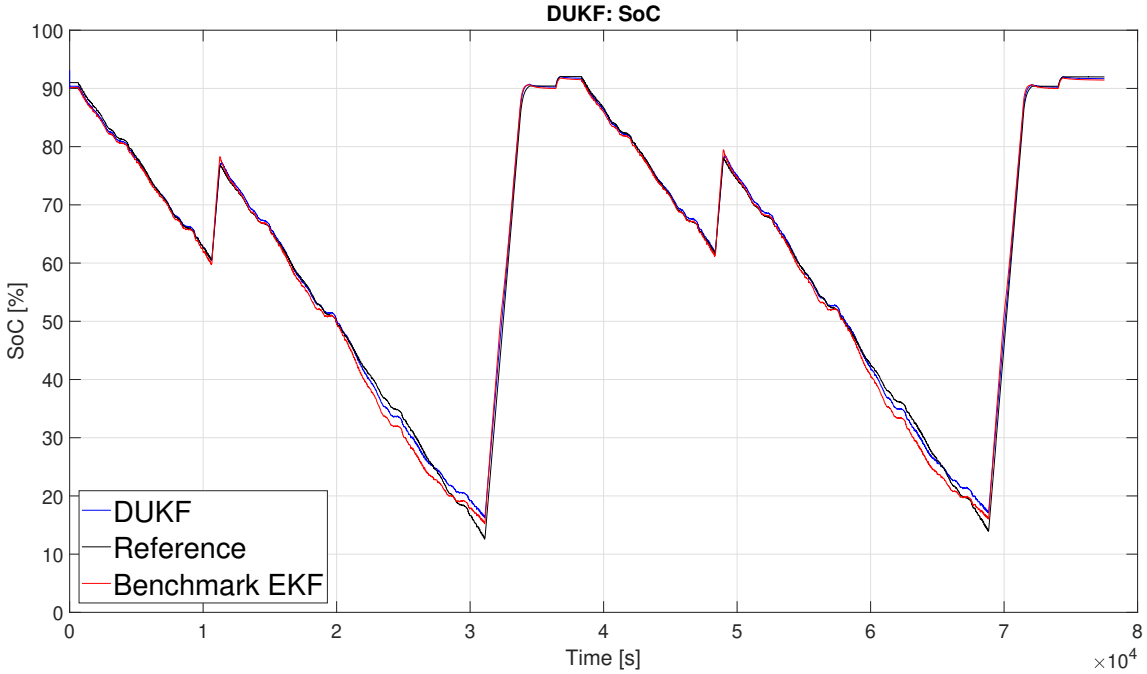

Figure 4.30: This figure shows the SoC trajectory as estimated by the DUKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF.

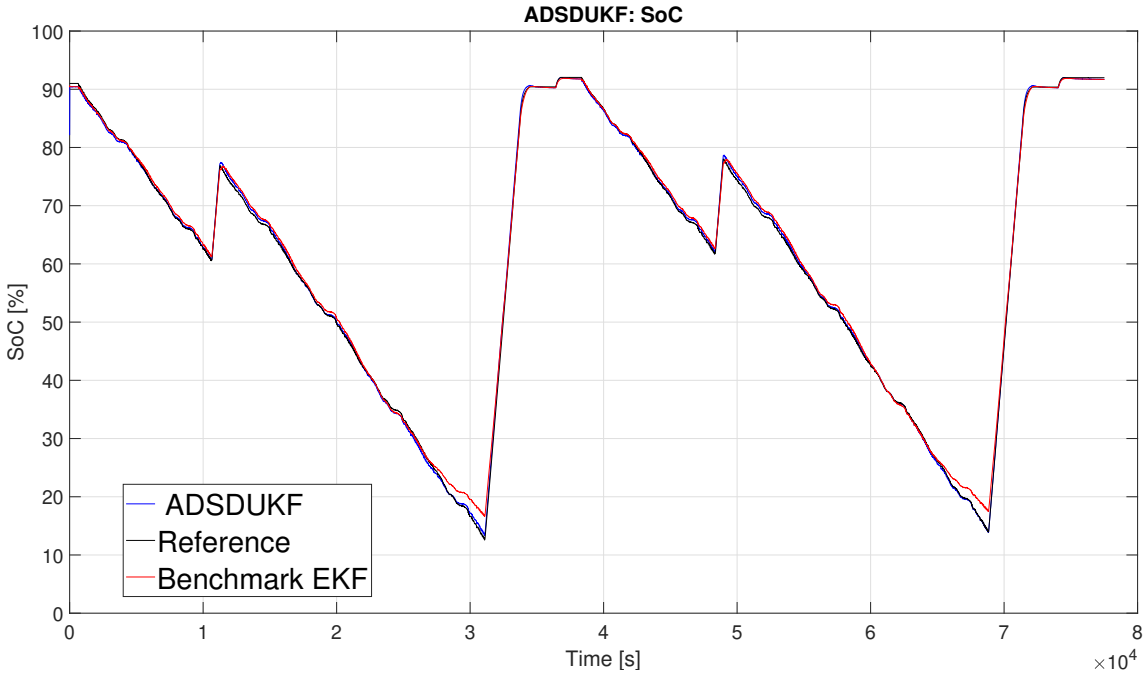

Figure 4.31: This figure shows the SoC trajectory as estimated by the ADSDUKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF.

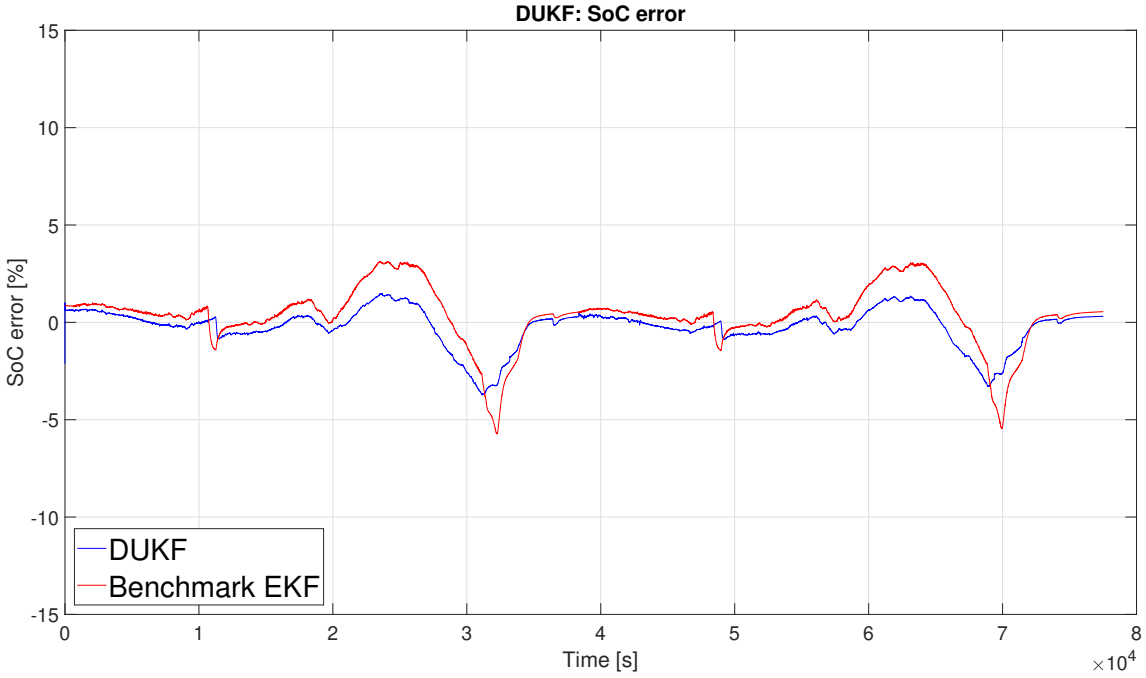

Figure 4.32: This figure shows the SoC error trajectory of the DUKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

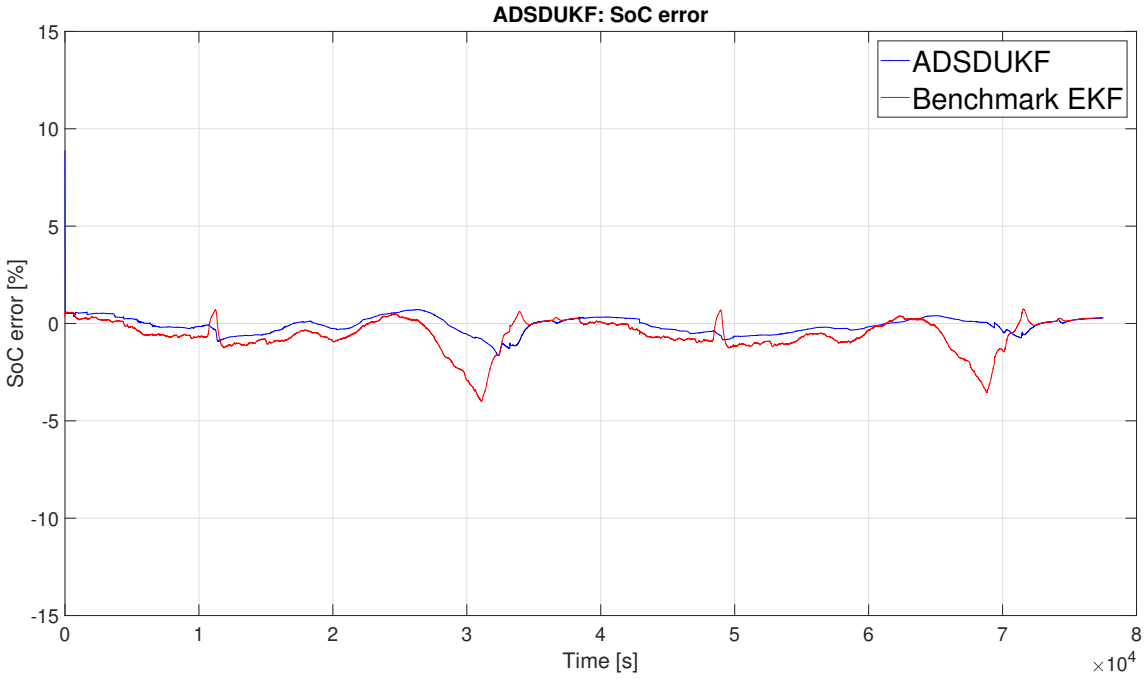

Figure 4.33: This figure shows the SoC error trajectory of the ADSDUKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

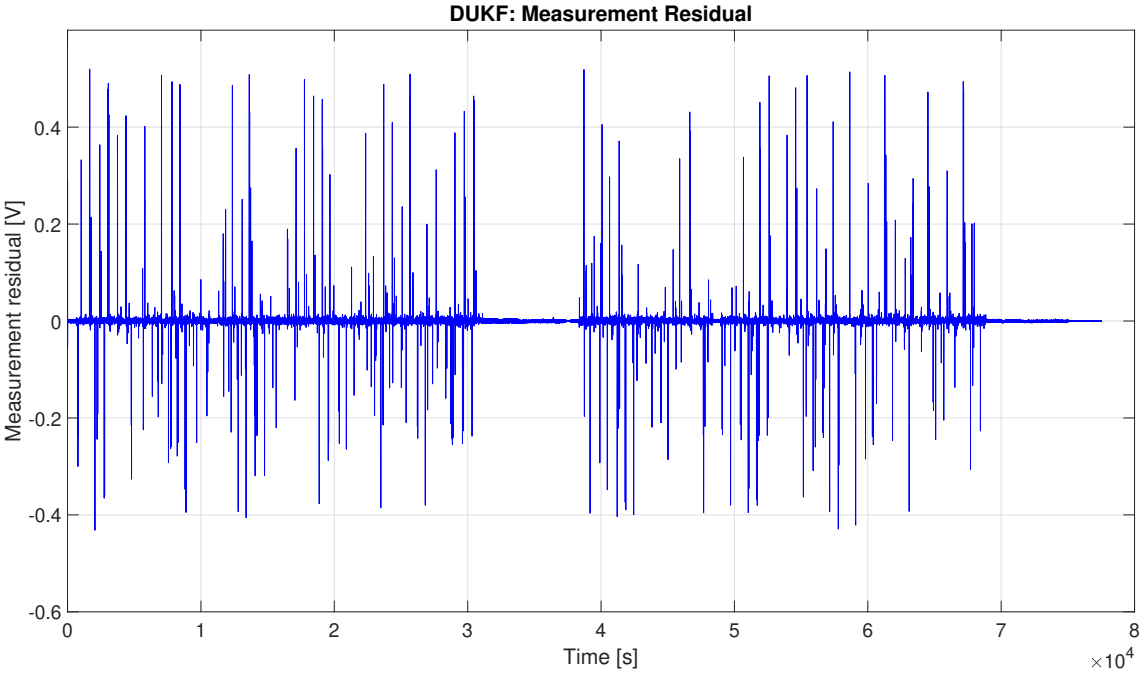

Figure 4.34: This figure shows the measurement residual of the DUKF using experimental data.

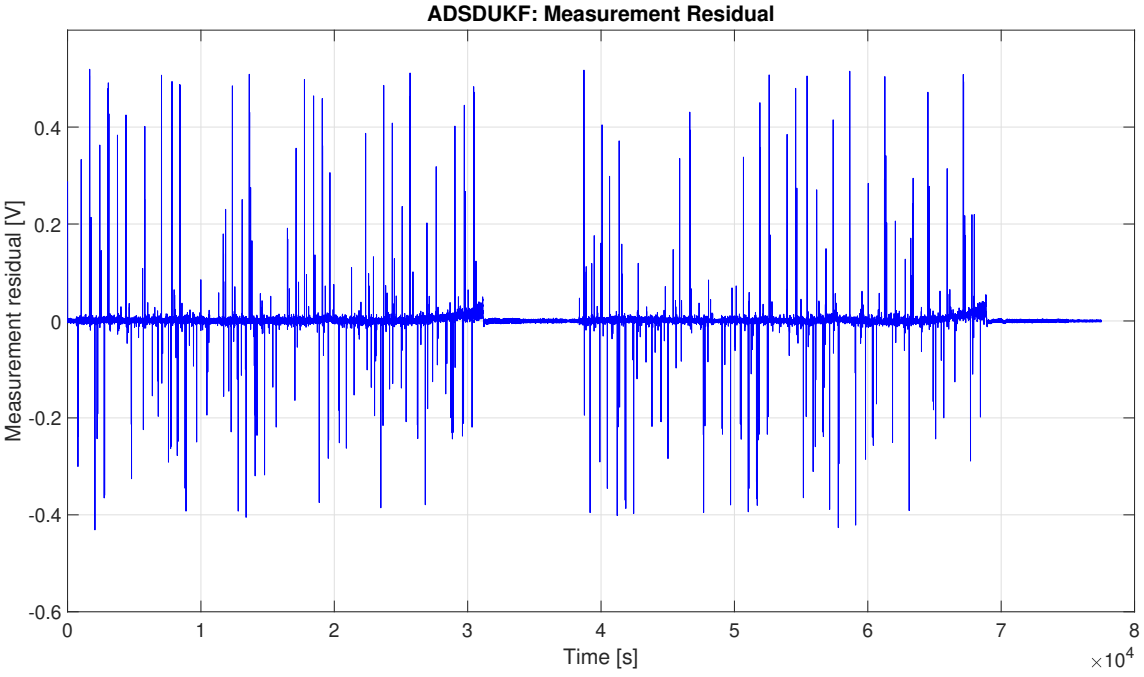

Figure 4.35: This figure shows the measurement residual of the ADSDUKF using experimental data.

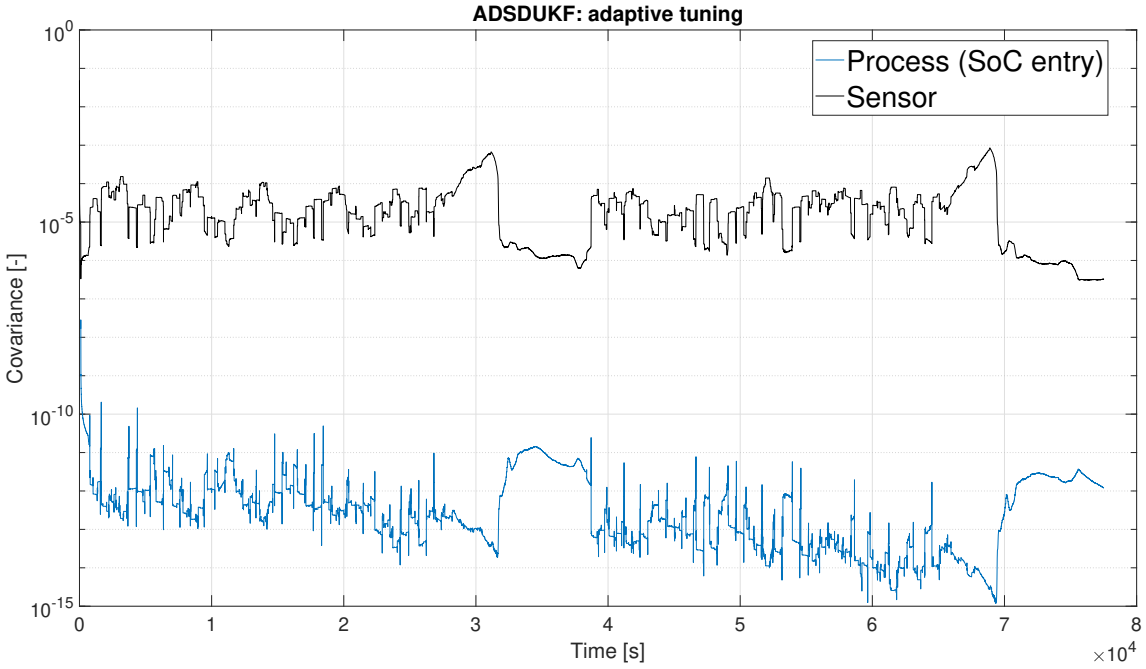

Figure 4.36: This figure shows the plots of the sensor noise covariance and the diagonal entry of the process noise covariance matrix corresponding to the SoC as approximated by the adaptivity feature. Sensor noise covariance is shown in black, process noise covariance is shown in blue. The vertical axis is logarithmic.

4.2.2.3 DSDUKF vs DSDEKF vs benchmark EKF

In this experiment, we validate the DSDEKF and the DSDUKF under challenging, realistic conditions, where even the benchmark EKF struggles to give good SoC estimates. Begin with a battery at 74% charge and discharge it completely to 0% charge. We then charge it back up to 74% charge. The experiment is conducted at $14°C$, which is below the optimal operating temperature. Aside from comparing DSDEKF and DSDUKF, this test also gives an indication as to whether the dual estimators developed in this thesis can compete with the preexisting benchmark EKF. All filters are operated using automotive-grade, low-resolution sensors.
Figure 4.38 shows the SoC trajectory estimated by the DUKF compared to the benchmark filter. The DUKF follows the trajectory of the benchmark filter relatively closely throughout most of the experiment. Towards the end of the test cycle, however, during the phase of rapid charging, the DUKF shows improved convergence to the true SoC compared to the benchmark filter. This is due to rapidly traversing the SoC ranges, where the battery's dynamics are poorly approximated by the linearizations of the EKF. Consequently, the DSDEKF, whose SoC estimates are plotted in Figure 4.39, does not show such improved convergence during charging.
The DSDUKF's SoC estimate error is depicted in Figure 4.40. The DSDUKF's improved convergence is especially visible around $t = 3 \cdot 10^4$, where the benchmark EKF reaches its maximum error values, while the DSDUKF's SoC estimation error converges back to zero. Figure 4.41 shows the DSDEKF's SoC estimation error. It stays relatively close to zero until around $t = 2.5 \cdot 10^4$, where it grows due to the hard nonlinearities in that region of the state space. During charging, the estimation

error grows rapidly to its maximum value of 7%.
Table 4.6 contains the performance indices for this experiment. We ascertain that both the DSDEKF and DSDUKF perform better than the benchmark filter in terms of SoC RMSE. The DUKF has the smallest value for SoC RMSE out of the three filters. The DSDUKF also achieves the smallest maximum SoC estimate error out of all three filters, improving the benchmark filter's score in this category by 26%. We can see from the average SoC error, however, that both the DSDUKF and the DSDEKF have a larger average bias than the benchmark filter. The DSDEKF's maximum SoC error is only 3% lower than that of the benchmark filter, however, its average SoC error's magnitude is lower than that of the DSDUKF.
We again see from this experiment the slight edge the DUKF gains over its EKF-based competitors by virtue of its theoretical advantages. In summary, we see that our dual estimators with data sensitivity checks are highly successful in giving accurate SoC estimates. They give more precise estimates than the benchmark filter, despite having access to less information.

Norm	DSDUKF	DSDEKF	benchmark EKF	Improvements over benchmark	
				DSDUKF	DSDEKF
$\tilde{z}_{RMS}[\%]$	2.6743	2.7342	3.6154	26.0303%	24.3735%
$\lvert\tilde{z}\rvert_{\infty}[\%]$	5.1744	7.3556	7.5735	31.6776%	2.8771%
$\mu_{\tilde{z}}[\%]$	1.4637	-1.1759	1.1286	-29.6917%	-4.1910%

Table 4.6: This table contains the values of the performance indices describing the performances of DSDUKF and DSDEKF for the experimental EV trajectory.

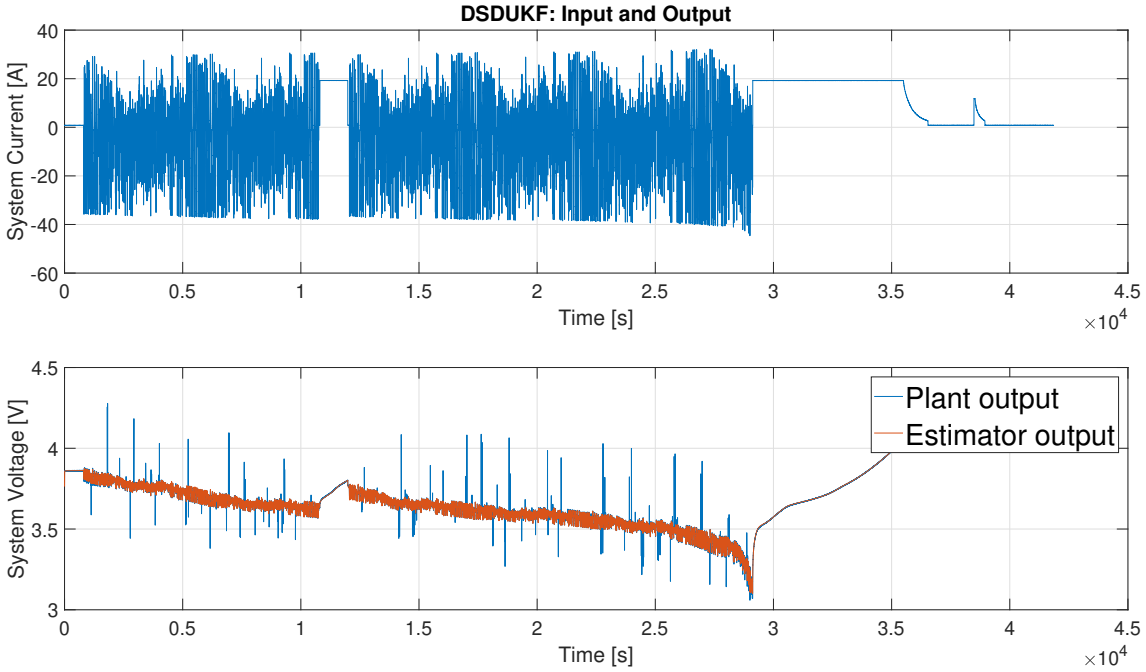

Figure 4.37: This figure shows the system inputs and outputs for the validation experiment as provided to both the DSDEKF and the DSDUKF. Low-resolution sensors are used. The current measurement displays bias.

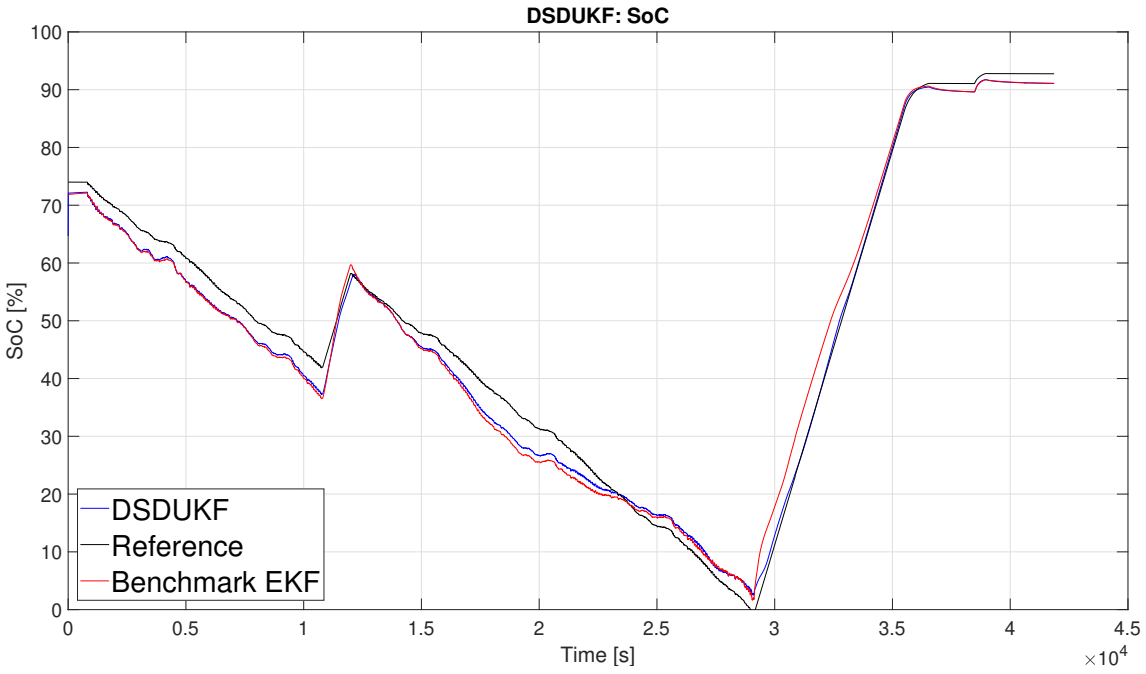

Figure 4.38: This figure shows the SoC trajectory as estimated by the DSDUKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF's estimate.

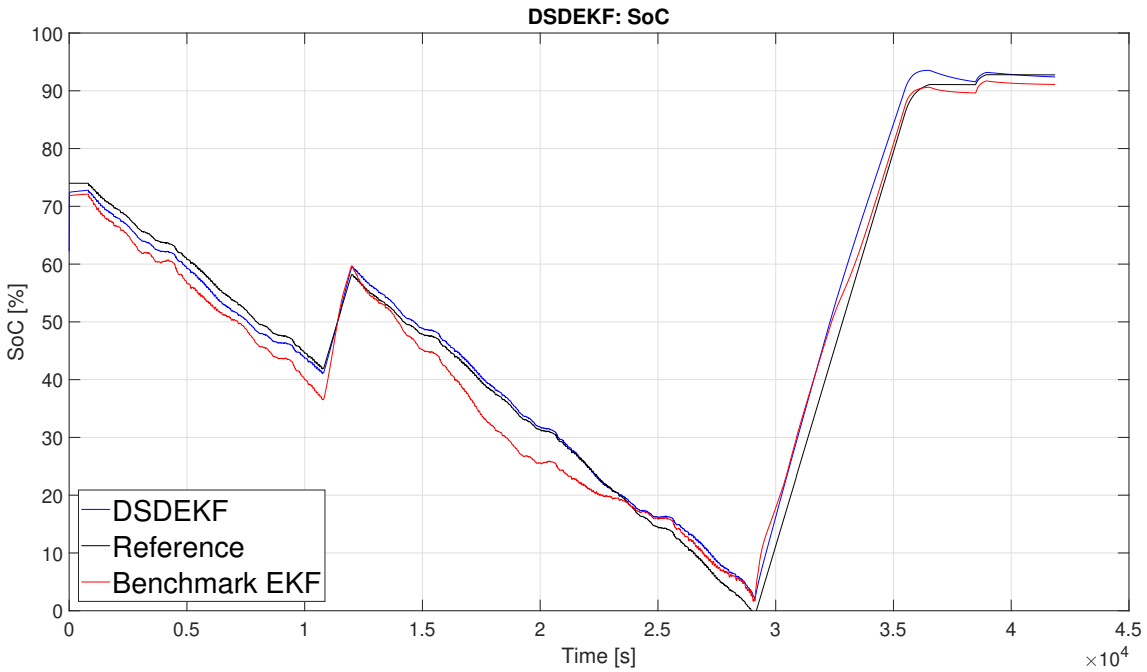

Figure 4.39: This figure shows the SoC trajectory as estimated by the DSDEKF from the experimental data in blue. True SoC measured by a high-precision sensor is shown in black. The red trajectory is the benchmark EKF's estimate.

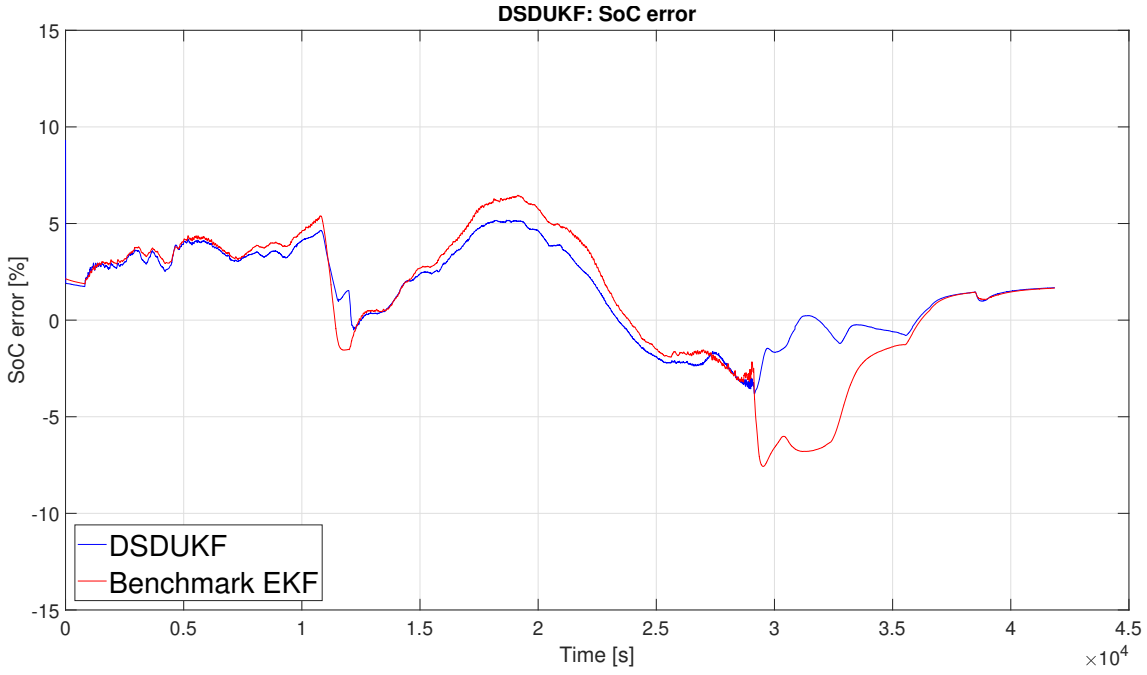

Figure 4.40: This figure shows the SoC error trajectory of the DSDUKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

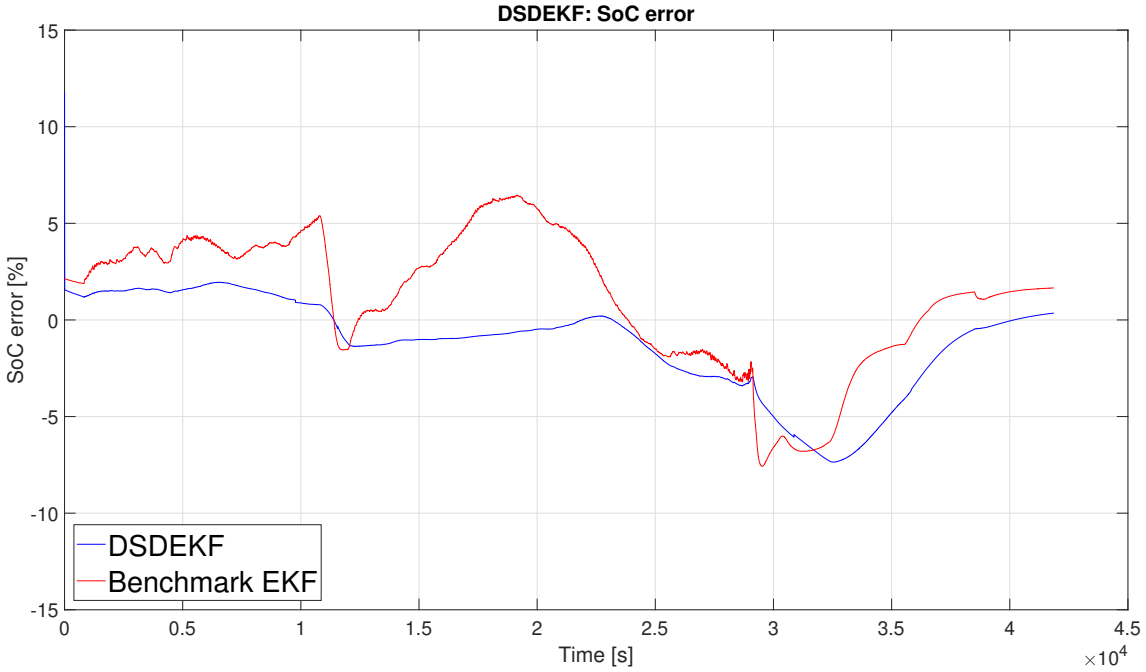

Figure 4.41: This figure shows the SoC error trajectory of the DSDEKF's estimate in blue. The red trajectory is the benchmark EKF's SoC error.

4.3 SoH Estimation Validation

In this section we will validate the SoH estimator developed in this thesis. To verify our estimator's ability to determine actual battery capacity, we perform the following experiment: We continually charge and discharge a battery over long periods of time, so as to cause capacity fade through aging of the battery. To be more precise, we cycle the battery between 95% and 5% SoC at $22°C$. At specific intervals, we do a so-called reference performance test (RPT) in laboratory using high-precision measurement equipment to characterize the battery's actual capacity at two different C-rates. The current and voltage measurement data from life-cycle testing (i.e. continuous charge/discharge testing, not RPT), is used by our dual SoC and SoH estimation scheme to give an estimate of actual capacity. This estimate should lie between the two measured battery capacities (1C capacity and 0.1C capacity).
The SoH estimator receives the estimates of the SoC estimator. In return the SoC estimtator receives the capacity estimates of the SoH estimator, in order to be able to continuously give accurate SoC estimates. This way, we not only validate our SoC and SoH estimators individually, but also validate correct interplay between the two estimators. Thus, we verify the functionality of our entire estimation scheme. We use the DSDEKF setup for SoC estimation in this experiment.
The results of the experiment are shown in Figure 4.42. It can be seen, that the SoH estimator generally gives reliable estimates of the battery's actual capacity. The second measurement point is the only instance where the SoH estimator gives an estimate that is more than 0.5 Ah away from the nearest measurement point. After this, the capacity estimate quickly reconverges to the accurate zone, despite the SoC estimator working with inaccurate capacity values until the next sample.

This shows that the dual estimation scheme is robust to some error in its belief about battery capacity.

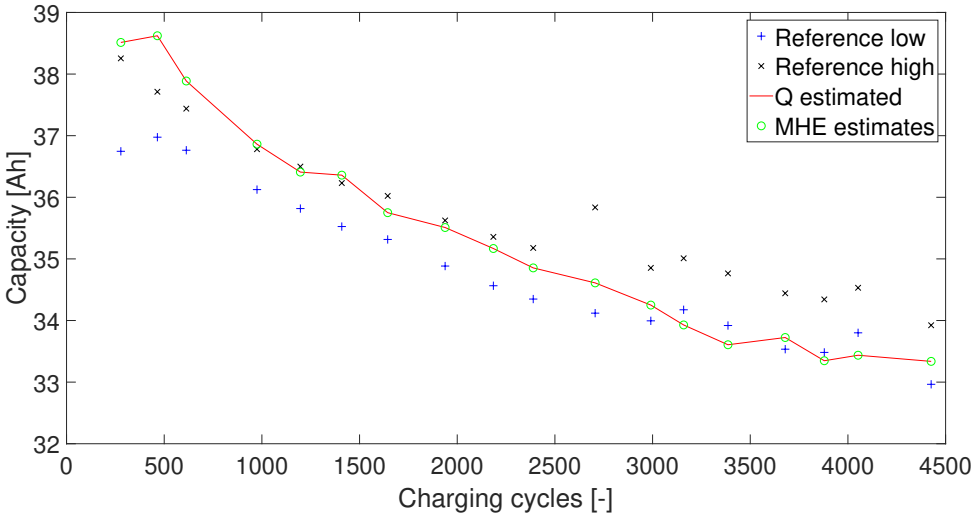

Figure 4.42: This figure shows the capacity estimates given by the SoC and SoH estimator. Green circles are the capacity estimates at different times. The red line is generated by linear interpolation between capacity estimates and only shown for better readability of the plot. High-precision capacity measurements at different C-rates are shown as black crosses and blue pluses. The unit of the x-axis is number of charging cycles.

It is helpful to look at some estimated SoC trajectories to understand the impact capacity estimates have on SoC estimation. Figure 4.43 shows two estimated SoC trajectories. They are SoC estimates made during the experiment we use for validation of the capacity estimator. One series of estimates is made by a DEKF without information about capacity fade. The other series of estimates is made by our combined SoC and SoH estimator. In the lower SoC ranges, the difference between the two estimates exceeds 10%. The combined SoC and SoH estimator continues decreasing to 5% SoC. The SoC estimator without information about the SoH vastly underestimates the depth of discharge, estimating more than 15% SoC, where the true SoC is at 5%. In a real use case scenario this can make the difference regarding driving range and reliable/robust utilization of a battery.
We see that using the combined SoC and SoH estimator developed in this thesis leads to accurate SoC estimates in spite of aging effects. This also shows the importance of said robustness against aging effects and the superiority of the setup presented in this thesis compared to simple dual SoC estimation.

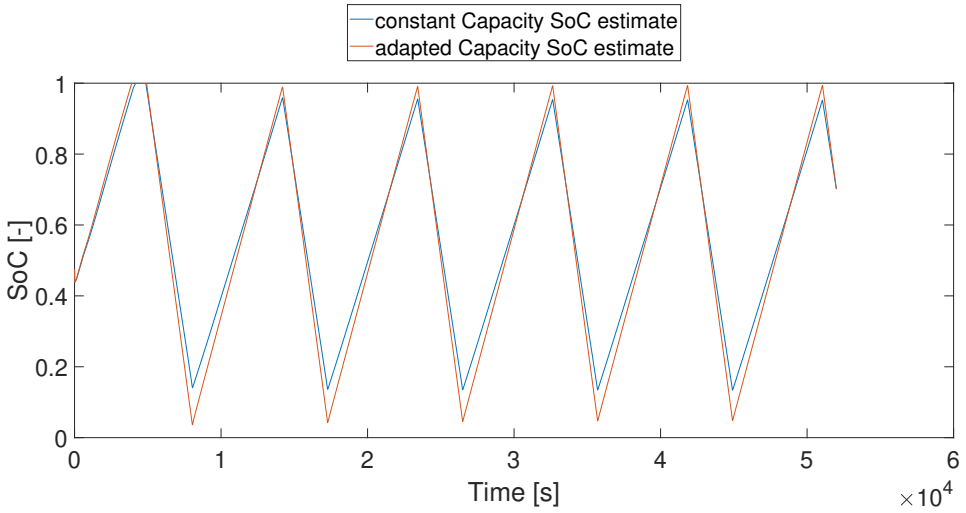

Figure 4.43: This figure SoC estimates of the DEKF during the final stages of experimental validation of the capacity estimator. In these final stages of cycle life testing the battery is at the end-of-life stage. The red trajectory is estimated by our dual SoC and SoH estimator. The blue trajectory is estimated by our SoC estimator without the capacity data from SoH estimation.

5
Summary and Conclusion

In the following chapter, we give a summary of the thesis and assess our success in accomplishing the objectives set in Chapter 1.4. We then give recommendations for future research.

5.1 Summary

We begin this thesis by motivating the importance of SoC and SoH estimation for the rapidly growing electric vehicle industry. We give an overview of the literature in battery modeling and estimation.
In chapter 2 we give a thorough introduction to the theoretical concepts employed in this thesis. We start by discussing key issues of battery management. Among these are state of charge, state of health, open-circuit voltage and terminal voltage. We then use this knowledge to derive an equivalent-circuit model as a design model for the estimators developed in this thesis. We note that the model must be populated with time varying parameters during run-time, necessitating dual state and parametric estimation. Thereupon, we analyze this model for observability. We proceed to present the estimation algorithms extended Kalman filtering, unscented Kalman filtering and moving horizon estimation. We give a detailed background to extended Kalman filtering and unscented Kalman filtering by deriving them from the more general probabilistic inference framework.
We adapt the theoretical concepts presented in chapter 2 to the present project in chapter 3. We implement dual state and parametric SoC estimators by adapting extended Kalman filtering and unscented Kalman filtering. We propose several ways of enhancing this dual estimation scheme: first, we develop an adaptive tuning mechanism to improve convergence of the state filter and adapt it for use in both Kalman filtering algorithms. Second, we design a data sensitivity check feature to improve the parameter filter by only letting it estimate the parameters from information-rich data sets. Third, we establish a synergy between our SoC and SoH estimators, with each filter profiting from the information provided by the other filter's estimates. We develop a SoH estimator using moving horizon estimation. Finally, we describe our tuning and initialization of all developed estimators.
In chapter 4 we validate the estimation schemes designed in chapter 3. We begin by performing a preliminary validation using a high-fidelity simulation model. We confirm the efficacy of our basic dual estimation schemes and the data sensitivity analyzer in simulation. We proceed to validate our SoC estimators' performance under realistic conditions in a series of lab tests. We compare them to a benchmark SoC

estimator. This benchmark filter is an extended Kalman filter, which estimates the battery system's state and takes the varying model parameters from lookup-tables generated in lab tests. We find that our adaptive tuning feature greatly improves estimation accuracy in both unscented Kalman filtering and extended Kalman filtering. It is, however, susceptible to sensor bias and not ready for application with automotive-grade sensors. We ascertain that our filters perform well, even under harsh, realistic conditions. We find that our dual estimators generally outperform the benchmark filter despite having access to less information than it. In a final experiment we show the effectiveness of our entire SoC and SoH estimation scheme. A battery is put through many charging cycles to cause capacity fade. Throughout this process, we employ our SoC and SoH estimation scheme. In so doing, we show that our setup gives greatly improved performance over a simple dual SoC estimator.

5.2 Main Achievements

We review our success with respect to the objectives set at the beginning of the thesis.
Objective 1 is achieved fully in this thesis. We present all necessary background on battery estimation and perform the modeling and system analysis that are prerequisite for any estimation. We proceed to give an introduction in some estimation techniques suitable for this problem. We succeed in implementing dual state and parametric SoC estimation schemes based on both extended Kalman filtering and unscented Kalman filtering. The two dual estimators' functionality, performance and robustness is validated. We do this by testing them both in simulation and using experimental data collected from automotive-grade sensors. We compare performance to a benchmark filter. Both filters perform better than the benchmark estimator used in validation.
We achieve Objective 2 partly in this thesis. We succeed in developing adaptive tuning mechanisms both for extended Kalman filtering and unscented Kalman filtering. Using these adaptivity features, we achieve much improved performance of our SoC estimators. In particular, the adaptive tuning mechanism is shown to improve SoC estimation quality radically under harsh conditions both in simulation and in experiments. However, the developed adaptivity mechanisms are found to be highly susceptible to the sensor bias present in automotive grade sensors. In the conducted experiments they are only shown to improve SoC estimation quality when used with high-grade, bias-free sensors. Therefore, at the state to which this thesis develops this technology, it is not yet ready to be used in automotive applications.
Objective 3 is accomplished in this thesis. We investigate the sensitivity of our process equations with respect to the parameters estimated by the SoC estimator. Using our knowledge of the system's dynamics, we develop sensitivity filters, which output a measure of how well the parameters can currently be estimated from the measurements. Using this sensitivity indicator, we develop a sensitivity analyzer as an augmentation to the dual SoC estimator. In particular, the sensitivity analysis feature improves the dual SoC estimator's parameter estimates by only allowing parameter estimate updates on the basis of information-rich data. This improves the overall quality of SoC estimates significantly, which we show in simulation.

We accomplish Objective 4 as set at the beginning of this thesis. We give a brief introduction to the technique of moving horizon estimation. We present a method to use the unique benefits of this technique to extract precise information about a battery's actual capacity from past trajectories of the state and parameter estimates of the dual SoC estimator. We confirm experimentally that this method works well to accurately estimate a battery's actual capacity throughout its aging process. We experimentally test this SoH estimation together with the previously developed SoC estimator. We demonstrate the improvement in SoC estimate quality caused by the synergy between SoH and SoC estimation.

5.3 Future Work

While this thesis has made some progress in harnessing many different ideas to improve battery estimation, there are areas, where further research is needed.
One important topic for future research is fortifying adaptive tuning mechanisms against sensor bias. Adaptive tuning of Kalman filters used for SoC estimation shows great promise in terms of improving estimation quality. However, we showed in this thesis, that further research is needed to make this technology work in the presence of sensor bias. Overcoming this challenge is key to using adaptive filtering in automotive applications. One possible way to do this would be to estimate sensor bias.
It has been shown in this thesis, how sensitivity analysis modifications can be used to improve parametric estimation. Additional research is needed to analyze the stability of sensitivity filtering equations like the ones presented in this thesis. The sensitivity analyzer developed in this thesis was used more successfully on DEKF than on DUKF. Consequently, the compatibility of such sensitivity analysis features with dual unscented Kalman filters should be investigated further.
This thesis achieves an estimation setup that works well despite battery aging effects. This robustness to battery aging could be increased further by developing methods of estimating the battery's SoC-OCV curve, as this changes over the course of the battery's life span. Developing methods for the estimation of the SoC-OCV curve could also decrease the reliance on experimental system identification, since the SoC-OCV curve is usually determined experimentally.
Having an estimate of SoH as given by the estimation scheme in this thesis is already highly useful for the systematic reuse and recycling of rechargeable batteries. The next step in powering large-scale efficient reuse of systems is giving health and remaining-useful-life prognoses. Such technologies could lead to great improvements in the sustainability of battery usage. Future researchers should also consider life long impedance tracking, which can give a more complete picture of a battery's SoH.
Capacity estimation using MHE is a very young field and the method presented in this thesis can be further refined. The downsampling technique used in this thesis can seriously harm estimation accuracy. Especially at the extremes of the SoC ranges, where battery voltage typically changes rapidly, small changes in sampling time can make a big difference in estimation accuracy. Minimizing such discretization errors can be key to improving MHE capacity estimation. One possibility to ameliorate the capacity MHE developed in this thesis would be to choose the sample

points more intelligently, thereby making sure to capture the system behavior well in the most important regions.

Bibliography

[1] Anup Barai, Kotub Uddin, W.D. Widanage, Andrew McGordon, and Paul Jennings. A study of the influence of measurement timescale on internal resistance characterisation methodologies for lithium-ion cells. *Scientific Reports*, 8, 01 2018.

[2] Anthony Barré, Benjamin Deguilhem, Sébastien Grolleau, Mathias Gerard, Frédéric Suard, and Delphine Riu. A review on lithium-ion battery ageing mechanisms and estimations for automotive applications. *Journal of Power Sources*, 241:680–689, 11 2013.

[3] M. Broussely, Ph. Biensan, F. Bonhomme, Ph. Blanchard, S. Herreyre, K. Nechev, and R.J. Staniewicz. Main aging mechanisms in li ion batteries. *Journal of Power Sources*, 146(1):90 – 96, 2005. Selected papers presented at the 12th International Meeting on Lithium Batteries.

[4] Alexander Farmann, Wladislaw Waag, Andrea Marongiu, and Dirk Sauer. Critical review of on-board capacity estimation techniques for lithium-ion batteries in electric and hybrid electric vehicles. *Journal of Power Sources*, 281, 05 2015.

[5] Björn Fridholm, Torsten Wik, and Magnus Nilsson. Robust recursive impedance estimation for automotive lithium-ion batteries. *Journal of Power Sources*, 304:33 – 41, 2016.

[6] H. He, R. Xiong, X. Zhang, F. Sun, and J. Fan. State-of-charge estimation of the lithium-ion battery using an adaptive extended kalman filter based on an improved thevenin model. *IEEE Transactions on Vehicular Technology*, 60(4):1461–1469, May 2011.

[7] Xiaosong Hu, Shengbo Li, and Huei Peng. A comparative study of equivalent circuit models for li-ion batteries. *Journal of Power Sources*, 198:359 – 367, 2012.

[8] Andreas Jossen. Fundamentals of battery dynamics. *Journal of Power Sources - J POWER SOURCES*, 154:530–538, 03 2006.

[9] I. Kim. A technique for estimating the state of health of lithium batteries through a dual-sliding-mode observer. *IEEE Transactions on Power Electronics*, 25(4):1013–1022, April 2010.

[10] Anton Klintberg, Changfu Zou, Björn Fridholm, and Torsten Wik. Kalman filter for adaptive learning of two-dimensional look-up tables applied to ocv-curves for aged battery cells. *Control Engineering Practice*, 84:230 – 237, 2019.

[11] Jacob Klintberg. On-line adaptive dual estimation of li-ion battery state and parameters in electric vehicles.

[12] X. Lin. Analytic analysis of the data-dependent estimation accuracy of battery equivalent circuit dynamics. *IEEE Control Systems Letters*, 1(2):304–309, Oct 2017.

[13] X. Lin. A data selection strategy for real-time estimation of battery parameters. In *2018 Annual American Control Conference (ACC)*, pages 2276–2281, June 2018.

[14] X. Lin. Theoretical analysis of battery soc estimation errors under sensor bias and variance. *IEEE Transactions on Industrial Electronics*, 65(9):7138–7148, Sep. 2018.

[15] Xinfan Lin, Hector E. Perez, Shankar Mohan, Jason B. Siegel, Anna G. Stefanopoulou, Yi Ding, and Matthew P. Castanier. A lumped-parameter electro-thermal model for cylindrical batteries. *Journal of Power Sources*, 257:1 – 11, 2014.

[16] Languang Lu, Xuebing Han, Jianqiu Li, Jianfeng Hua, and Minggao Ouyang. A review on the key issues for lithium-ion battery management in electric vehicles. *Journal of Power Sources*, 226:272 – 288, 2013.

[17] Matthias A. Müller. Nonlinear moving horizon estimation in the presence of bounded disturbances. *Automatica*, 79:306 – 314, 2017.

[18] Anand Narayan. State and parametric estimation of li-ion batteries in electrified vehicles.

[19] M. Partovibakhsh and G. Liu. An adaptive unscented kalman filtering approach for online estimation of model parameters and state-of-charge of lithium-ion batteries for autonomous mobile robots. *IEEE Transactions on Control Systems Technology*, 23(1):357–363, Jan 2015.

[20] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 1. background. *Journal of Power Sources*, 134(2):252 – 261, 2004.

[21] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 2. modeling and identification. *Journal of Power Sources*, 134(2):262 – 276, 2004.

[22] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 3. state and parameter estimation. *Journal of Power Sources*, 134(2):277 – 292, 2004.

[23] Gregory L. Plett. Sigma-point kalman filtering for battery management systems of lipb-based hev battery packs: Part 1: Introduction and state estimation. *Journal of Power Sources*, 161(2):1356 – 1368, 2006.

[24] Gregory 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*, 161(2):1369 – 1384, 2006.

[25] Habiballah Rahimi Eichi, Unnati Ojha, Federico Baronti, and Mo-Yuen Chow. Battery management system: An overview of its application in the smart grid and electric vehicles. *Industrial Electronics Magazine, IEEE*, 7:4–16, 06 2013.

[26] James B. Rawlings. *Moving Horizon Estimation*, pages 1–7. Springer London, London, 2013.

[27] M. A. Roscher, J. Assfalg, and O. S. Bohlen. Detection of utilizable capacity deterioration in battery systems. *IEEE Transactions on Vehicular Technology*, 60(1):98–103, Jan 2011.

[28] Fengchun Sun, Xiaosong Hu, Yuan Zou, and Siguang Li. Adaptive unscented kalman filtering for state of charge estimation of a lithium-ion battery for electric vehicles. *Fuel and Energy Abstracts*, 36:3531–3540, 05 2011.

[29] Resmi Suresh, Hemanth Kumar Tanneru, and Raghunathan Rengaswamy. Modeling of rechargeable batteries. *Current Opinion in Chemical Engineering*, 13:63 – 74, 2016. Energy and Environmental Engineering / Reaction engineering and catalysis.

[30] J. Vetter, P. Novák, M.R. Wagner, C. Veit, K.-C. Möller, J.O. Besenhard, M. Winter, M. Wohlfahrt-Mehrens, C. Vogler, and A. Hammouche. Ageing mechanisms in lithium-ion batteries. *Journal of Power Sources*, 147(1):269 – 281, 2005.

[31] M. Vidyasagar. *Nonlinear Systems Analysis: Second Edition*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2002.

[32] Wladislaw Waag, Christian Fleischer, and Dirk Uwe Sauer. Critical review of the methods for monitoring of lithium-ion batteries in electric and hybrid vehicles. *Journal of Power Sources*, 258:321 – 339, 2014.

[33] Nikolaos Wassiliadis, Jörn Adermann, Alexander Frericks, Mikhail Pak, Christoph Reiter, Boris Lohmann, and Markus Lienkamp. Revisiting the dual extended kalman filter for battery state-of-charge and state-of-health estimation: A use-case life cycle analysis. *Journal of Energy Storage*, 19:73 – 87, 2018.

[34] Cheng Zhang, Kang Li, Sex00E1n McLoone, and Zhile Yang. Battery modelling methods for electric vehicles - a review. *2014 European Control Conference (ECC)*, pages 2673–2678, 2014.

[35] Shi Zhao, Stephen R. Duncan, and David A. Howey. Observability analysis and state estimation of lithium-ion batteries in the presence of sensor biases. *CoRR*, abs/1510.06553, 2015.

A

Appendix 1

The matrices used in the numeric optimization problem formulation (3.88) are

$$H = \begin{bmatrix} 0 & & & & & & \\ & P & & & & & \\ & & 0_{3\times3} & & & & \\ & & & d_1 & & & \\ & & & & \ddots & & \\ & & & & & 0_{3\times3} & \\ & & & & & & d_N \end{bmatrix}, \tag{A.1}$$

$$f = \begin{bmatrix} 0 & -2(P\hat{x}_{k-N})^T 0_{1\times4N} \end{bmatrix}, \tag{A.2}$$

$$A_{\text{eq}} = \begin{bmatrix} 0_{2\times1} & -A_{k-N} & 0_{2\times1} & I_2 & 0_{2\times1} & & & \\ -h\eta u_{k-N} & 0_{1\times2} & -1 & 0_{1\times2} & 1 & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0_{2\times1} & & & \ddots & -A_{k-1} & 0_{2\times1} & I_2 & 0_{2\times1} \\ -h\eta u_{k-1} & & & & 0_{1\times2} & -1 & 0_{1\times2} & 1 \end{bmatrix} \tag{A.3}$$

$$b_{\text{eq}} = \begin{bmatrix} B_{k-N}u_{k-N} \\ 0 \\ \vdots \\ B_{k-1}u_{k-1} \\ 0 \end{bmatrix} \tag{A.4}$$

$$A_{\text{ineq}} = \begin{bmatrix} -1 & 0_{1\times 2} & 0 & & & & & & \\ 1 & 0_{1\times 2} & 0 & & & & & & \\ 0 & 0_{1\times 2} & -1 & & & & & & \\ 0 & 0_{1\times 2} & 1 & & & & & & \\ \vdots & \vdots & 0 & 0_{1\times 2} & -1 & & & & \\ & & 0 & 0_{1\times 2} & 1 & & & & \\ & & \vdots & \vdots & 0 & 0_{1\times 3} & -1 & & \\ & & & & 0 & 0_{1\times 3} & 1 & \ddots & \\ & & & & & \dots & \ddots & \ddots & \\ & & & & & & 0 & 0_{1\times 3} & -1 & 0 \\ 0 & 0_{1\times 2} & 0 & 0_{1\times 2} & & & 0 & 0_{1\times 3} & 1 & 0 \end{bmatrix} \tag{A.5}$$

$$b_{\text{ineq}} = \begin{bmatrix} -\dfrac{1}{Q_{\max}} \\ \dfrac{1}{Q_{\min}} \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \tag{A.6}$$