Comparison of Nonlinear Filtering Methods for Battery State of Charge Estimation

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Masters of Science
in
Electrical Engineering

by

Klaus Zhang

B.S. Pennsylvania State University, 2011

May 2014

Abstract

In battery management systems, the main figure of merit is the battery's State of Charge, typically obtained from voltage and current measurements. Present estimation methods use simplified battery models that do not fully capture the electrical characteristics of the battery, which are useful for system design. This thesis studied State of Charge estimation for a lithium-ion battery using a nonlinear, electrical-circuit battery model that better describes the electrical characteristics of the battery. The accuracy and speed of the estimation was investigated for various nonlinear filters through numerical simulation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Batteries, particularly rechargeable ones, are used extensively in daily life. They provide the energy for such electrical systems as communication, automotive, and renewable power systems. In order to design for and operate these systems, an accurate battery model and a means of simulating the model efficiently is needed. For example, modern battery charge and health management schemes use high-fidelity battery models to track the state of charge (SOC) and state of health (SOH); this information is then used to predict and optimize the runtime of the battery. However, widely-used chemical batteries have nonlinear capacitive effects, which require the use of a nonlinear filter for accurate prediction of its states in the presence of noise. This thesis explores one possible solutions to this problem by choosing an appropriate battery model and testing the accuracy and speed of various nonlinear filters in determining the SOC through simulation. Note that only filters using point-based numerical approximation methods were studied, as opposed to those using density-based methods. See [1] for more information about the differences between various numerical approximation methods in relationship to Bayesian filtering.

1.1 Electrical Characteristics of Rechargeable Batteries

A high-fidelity battery model has to accurately reproduce the various characteristics of a battery. Most models keep track of the total capacity and SOC in order to predict remaining runtime. More accurate models include nonlinear effects, such as the rate-capacity effect and the recovery effect, along with self-discharge and the effects of ambient temperature. The dynamic electrical attributes, such as the current-voltage (i-v) characteristics and transient responses, can also be modeled. The remainder of this section defines these characteristics.

The capacity of a battery is the amount of electric charge it can store, measured in the SI unit Ampere-hours (Ah). Commonly, for rechargeable battery specifications, the subunit milliampere-hour (mAh) is used. Related is the available capacity, which is the amount of charge that the battery can currently deliver. Due to the electrochemical nature of batteries, a battery's available capacity decreases as the rate of discharge increases, known as the rate-capacity effect. Therefore, the capacity for a battery is typically stated for a given discharge rate. Related to this is the recovery effect, so called because when a battery is allowed to rest during an idle period, the battery "recovers" available capacity previously lost during discharge due to the rate-capacity effect. Thus, a battery that is discharged at a high rate until its available capacity reaches zero, when allowed to rest, regains a portion of its lost capacity.

Both the rate-capacity effect and the recovery effect can be explained by the electrochemical nature of the battery. During discharge, the concentration of the active material around the electrode is depleted, and the active materials in the depletion region move towards the electrode to reduce the concentration gradient [2].

Because the speed at which the concentration gradient is equalized is limited, the faster the rate of discharge, the less the active material is replenished, resulting in a decrease in the available capacity. Likewise, when the battery is allowed to rest, the active material gradient has additional time to equalize, and the available capacity is increased.

Closely related to the capacity is the SOC. This thesis defines it as the ratio between the remaining capacity and the maximum capacity, with both capacities measured using the amount of active material within the battery. This definition then denotes the proportion of remaining chemical energy rather the available energy and is unaffected by the rate-capacity and recovery effects. Note that a fully charged battery has an SOC of unity and a fully discharged battery has an SOC of zero, regardless of the available capacity. Additionally, there exists a nonlinear relationship between the SOC of the battery and its open-circuit voltage V_{OC}, which is useful for simulation of the i-v characteristics and transient responses. V_{OC} is the limit of the measured battery voltage after recovery.

Other more minor effects that are usually incorporated into models are self-discharge, the effect of ambient temperature, and aging. Self-discharge refers to an idle battery decreasing its SOC over time due to internal chemical reactions. It is dependent on the type of battery, SOC, ambient temperatures, and other factors. The ambient temperature has effects on the internal resistance of the battery and the self-discharge rate. Commonly, the battery is designed to operate with a narrow range of temperatures. Below the operating temperature range, the internal resistance increases, decreasing the capacity. Above the operating range, the internal resistance decreases, not only increasing the capacity but also the self-discharge rate; thus, the actual capacity is lowered due to the increased self-discharge. Aging refers to the decrease in battery performance measures, such as capacity, self-discharge, and internal resistance, over time due to unwanted chemical reactions. In practice, aging is indicated by the SOH, defined as the ratio between the current maximum capacity and that of a new battery. The SOH threshold at which the battery performance is considered too degraded varies by application.

This thesis is mainly concerned with estimating the SOC from noisy measurements. The SOH is much easier to estimate as it changes slowly over charge cycles. Additionally, no simplified expression exists for the SOH, and it is usually determined empirically. Thus, only the estimation of the SOC was studied by this paper.

1.2 Battery Models

This thesis studied the estimation of the SOC of a battery given knowledge of the resistive load on the battery as well as noisy measurements of the voltage across its terminals. A known resistive load profile, rather than the current, was used because in a real-life usage, it is difficult to exactly control the current drawn by a load. In order to estimate the SOC for a general load profile, incorporation of the rate-capacity and recovery effects as well as the transient i-v characteristics is desirable. Furthermore it is useful to have a model easily tunable for different battery types. To find a battery model that meets these goals, the major types of battery models are reviewed and their characteristics are compared. Jongerden and Haverkort determined four main categories for battery models, namely electrochemical, analytical, stochastic, and electrical-circuit [3]. Additionally, battery models that use computational intelligence exist, e.g. [4–8]. The remainder of this section reviews these five types and determines the most suitable battery model for this study.

1.2.1 Electrochemical

Electrochemical models are describe the chemical processes that place in the battery in great detail. These are generally the most accurate, but they require in-depth knowledge of the chemical processes to create and impose large computational costs [9]. One of the most widely known electrochemical models was developed by Doyle, Fuller, and Newman for lithium and lithium-ion batteries using noninvasive voltage-current cycling experiments [10–12]. It consists of six coupled, nonlinear differential equations that capture lithium diffusion dynamics and charge transfer kinetics. The model is able to predict i-v response and provides a design guide for thermodynamics, kinetics, and transport across electrodes. A implementation of their model in Fortran, called Dualfoil, is available for free online.[1] The program needs more than 60 parameters along with the load profile in order to compute the battery properties. Setting the parameters requires detailed knowledge of the battery, but the result of the program is highly accurate. Other battery models are often compared to it rather than to experimental results.

1.2.2 Computational Intelligence

Computational intelligence is a branch of computer science interested in problems that require the intelligence of humans and animals to solve. One of the earliest definitions by Bezdek states that computational intelligent systems use pattern recognition on low-level, numerical data and do not use knowledge as with artificial intelligence [14, 15]. Methods such as neural networks, fuzzy systems, and evolutionary computation are commonly classified as computational intelligence. Battery models using such methods as neural networks [5, 6], support vector machines [7],

[1] J. Newman, *Fortran programs for the simulation of electrochemical systems*, `http://www.cchem.berkeley.edu/jsngrp/fortran.html`, 1998.

and hybrid neural-fuzzy models [8] have been studied. These models learn the non-linear relationships between battery properties, such as SOC, current, voltage, and temperature, through a computationally costly training process. However, once trained, they incur a much lower cost and can achieve comparable accuracy to electrochemical models.

1.2.3 Analytical

Analytical models are simplified electrochemical models that trade off accuracy for simplicity. One of the simplest such models is Peukert's law for lead-acid batteries, which states that for a one-ampere discharge rate [16]

$$C_p = I^k t, \tag{1.1}$$

where C_p is the capacity at a one-ampere discharge rate in Ah, I is the discharge current in A, t is the time to discharge the battery in hours, and $k \geq 1$ is the dimensionless Peukert constant, typically between 1.1 and 1.3 for a lead-acid battery. The constant k only equals unity for an ideal accumulator, so for real batteries, k is always greater than unity. Thus, for a given increase in the discharge current, the discharge time decreases by a proportionally greater amount. Therefore, the effective, or available, capacity $C \times t$ is reduced. Peukert's law can be extended to some other battery chemistries, such as lithium-ion. Note that Peukert's law only models the rate-capacity effect and not the recovery effect. More complicated models, such as the kinetic battery model and the diffusion model, are able to describe both effects.

The kinetic battery model (KiBaM), initially created for large lead-acid batteries, describes the battery as a kinetic process, using two charge wells for the bound and available charges connected by a valve whose flow rate is proportional to the height difference between the wells [17]. The change of charge in the wells is given by

$$\begin{cases} \dfrac{dy_1}{dt} = -I + k\left(h_2 - h_1\right) \\ \dfrac{dy_2}{dt} = -k\left(h_2 - h_1\right), \end{cases} \tag{1.2}$$

where y_1, y_2 are the charges, h_1, h_2 are the heights of the wells, the parameter k controls the rate of charge flow between the wells, and I is the applied load. The flow rate of the valve should be lower than the typical discharge rate of the battery. During discharge from the available-charge well, the bound charges flow through the valve to equalize the heights of the two wells. It can be seen that for slower discharge rates, more charge flows through the valve and the effective capacity increases. Likewise, during idle periods for the battery, the available charge increases.

Related to the KiBaM is the diffusion model, which describes the movement of the ions in the electrolyte of a lithium-ion battery [18]. Like in the kinetic battery model, the difference in the concentration of adjacent ions along the length of the battery determines the diffusion rate of the ions. The available charges are those ions directly touching the electrode of the battery. It can be seen that the KiBaM is a first-order approximation of the diffusion model [9], since the individual ions in the diffusion model are replaced by two charge wells in the KiBaM.

1.2.4 Stochastic

Stochastic models describe the discharging and the recovery effect as stochastic processes. The first models were developed by Chiasserini and Rao and based on discrete-time Markov chains [19]. They studied two models of a battery in a communication device that transmitted packets. The simpler model described the battery as a discrete-time Markov chain with $N + 1$ states, numbered from 0 to N and corresponding to the number of charge units available in the battery. Transmitting one packet requires one charge unit of energy. Thus, in continuous transmission, N packets can be sent. At every time step, a charge unit is either consumed with probability $a_1 = q$ or recovered with probability $a_0 = 1 - q$. The battery is considered empty when the 0 state is reached or when a theoretical maximum of T charge units have been consumed. The second model is an extension of the first, allowing for more than one charge unit to be consumed in a time step, modeling more bursty usage. Additionally, the battery has a non-zero probability of staying in the same charge state, indicating no consumption or recovery during a time step. Chiasserini and Rao extended their model further in following papers by adding state and phase dependence [2, 20, 21]. The state number is the number of charge units, and the phase number is the number of consumed charge units. Having fewer charge units decreases the probability of recovery, while having more consumed charge units increases the probability of recover. Using these models, one can model different loads by setting the transitions probabilities. However, the order of the transitions is uncontrollable, so it is impossible to model fixed load patterns and compute their impact on battery life.

Chiasserini and Rao mainly investigated the gain G in transmitted packets using a pulsed discharge relative to using a constant discharge, defined as $G = m/N$, where m is the mean number of transmitted packets. The gain increases when the load decreases, due to an increase in the recovery probability. Additionally, the gain increases for lower discharge demand rates and higher current densities. These load profiles result in discharge currents close to the specified limits of the battery, causing the available capacity to decrease overly quickly. Therefore, the recovery effect is especially strong for these cases during pulsed discharge, greatly increasing the gain. Chiasserini and Rao compared the computation of the gain parameter for different current densities and demand rates using the stochastic model to that of the electrochemical model of Doyle et al. They found an average deviation of 1% and a maximum deviation of 4%. This shows that the stochastic model accurately describes battery behavior during pulsed discharge. However, this model is only able to compute relative lifetimes.

In 2005, Rao et al. [22] proposed a stochastic battery model based on the Kinetic Battery Model (KiBaM) of Manwell and McGowan. This stochastic KiBaM was for a nickel-metal hydride (NiMH) battery. The differential equations governing the original KiBaM were modified to include an extra factor h_2 governing the flow of charge between the wells. This changes Equation (1.2) into

$$\begin{cases} \dfrac{dy_1}{dt} = -I + k_s h_2 \left(h_2 - h_1\right) \\ \dfrac{dy_2}{dt} = -k_s h_2 \left(h_2 - h_1\right), \end{cases} \tag{1.3}$$

This change causes the recovery effect to weaken as the remaining charge decreases. The stochastic model was also modified to allow the possibility of no recovery during idle periods. The stochastic KiBaM describes the battery using a discrete-time, transient Markov process. The states are labeled with the parameters (i, j, t), with i and j representing the discrete charge levels of the available and bound charge wells and t representing the length of the current idle period. Like the stochastic model of Chiasserini and Rao, it is impossible to fully model a real-life discharge pattern using the stochastic KiBaM. Rao et al. compared the results of their model with experimental results using an AAA NiMH battery. Two sets of experiments were conducted, the first with varying frequency of the load and a 50% duty cycle and the second with varying off-time and a constant on-time. Their model accurately predicted the lifetime and delivered charge from the battery, with a maximum error of 2.65%.

1.2.5 Electrical-Circuit

Electrical-circuit models for batteries developed from the discovery of capacitative effects at the electrode-electrolyte interface. Helmholtz first proposed the existence of a double layer of charge at the interface in 1879. In 1899, Warburg proposed a series resistance and capacitance circuit model with an infinitely low current density. The Warburg capacitance C_W named after him varies inversely with the square root of the frequency [23]. In 1947, Randles proposed a model consisting of a double-layer polarization capacitance C_p in parallel with the series combination of a resistor R and a capacitance C [24]. In 1994, Kovacs improved Randles circuit with the addition of Warburg impedance Z_W replacing the capacitance C and the solution resistance R_s in series with the original Randles circuit [25]. In addition, he renamed C_p to the double layer capacitance C_{dl} and R to the charge-transfer resistance R_{ct}. These proposals came from a desire to represent impedance spectra created using electrochemical impedance spectroscopy (EIS). The various elements in the models represent the different processes within a battery, which have different time constants. While these attempts model the impedance and, thus, account for the nonlinear rate-capacity and recovery effects, they do not consider the capacity and self-discharge of the battery.

In 1993, Hageman created simplified electrical-circuit models using PSpice for nickel-cadmium (NiCd), lead-acid, and alkaline batteries [26]. The circuits shared the common elements of *i*) a capacitor that represents the battery capacity, *ii*) a discharge rate normalizer that determines the additional capacity loss at high discharge rates, *iii*) a circuit that discharges the battery, *iv*) a lookup table of battery voltage versus SOC, and *v*) a resistor that represents the battery's internal resistance [26, 27]. In addition, battery models for NiCd batteries simulated the thermal effects under high discharge rates. The main lookup table is formed by discharging a battery at a low rate at a constant current (20 to 200 hours). At high discharge rates, the discharge rate normalizer reduces the battery voltage below the value from looking up the SOC in the table. This normalizer is implemented using additional lookup tables. These circuit models are much simpler than electrochemical models, but they are also less accurate with an approximate error of 10%. Furthermore, creation of the lookup tables requires considerable data. These circuit-based models are mainly concerned with modeling the remaining discharge time and are referred to as runtime-based models.

In 2006, Chen and Rincón-Mora proposed a combination of a runtime-based model and a impedance-based model consisting of a series resistor and two parallel resistor-capacitor networks [28]. A schematic for their model is shown in Figure 2.1. The elements of the impedance part of the model had parameters that depended on the SOC. Additionally, the runtime model included a resistance that modeled the self-discharge rate. Their proposed model has the advantage of accurate prediction of the SOC using the runtime-based portion while also modeling nonlinear transient effects, such as the rate-capacity and recover effects, with the impedance-based portion. Furthermore, the battery data can be collected using EIS measurements, which requires neither detailed knowledge of the battery chemistry nor lengthy, low-rate discharge experiments.

1.2.6 Evaluation

Of the model types, only some are fit for use with filtering algorithms. The computational-intelligence and stochastic models do not adequately describe the dynamics of the battery system for use in the filters covered by this study. On the other hand, electrochemical, analytical, and electrical-circuit models do describe the system dynamics in a compatible manner. Furthermore, they model the nonlinear rate-capacity and recovery effects. Of these, only the electrical-circuit model has the advantage of modeling the internal impedance of the battery, which is useful in the design of battery systems. The relevant characteristics of the model types are summarized in Table 1.1. It can be seen that electrical-circuit models are the most suitable for this study. Among them, the proposal by Chen and Rincón-Mora is most appropriate for the purposes of this thesis, because it is the only one discussed by this paper that describes both the capacity and the transient effects. Therefore, their proposed model is used for the simulation of the battery and comparing the performance of different filters.

Table 1.1: Summary of relevant characteristics of various battery model types.

Model Type	Dynamics	Nonlinear Effects	Transient Effects	I-V Characteristics	Design Difficulty
Electrochemical	Y	Y	Y	N	High
Computational Intelligence	N	Y	Y	N	High
Analytical	Y	Y	N	N	Low
Stochastic	N	Y	N	N	Low
Electrical-Circuit	Y	Y	Y	Y	Medium

1.3 Nonlinear Filtering Methods

Filtering refers to the methodology for estimating the state of a time-varying system that is indirectly observed through noisy measurements. Specifically, the state at the current time is estimated using the measurements from the current and previous times. The state of a system is a group of dynamic variables that evolve through time, and its evolution through time is governed by a dynamic system, perturbed by process noise. The measurements are functions of the state and the measurement noise.

Systems are classified as either linear or nonlinear. The state dynamics and measurements of a linear system are linear functions of the state, inputs, and noises. Particularly, the superposition principles of additivity and homogeneity are satisfied by a linear system. Nonlinear systems do not satisfy the principle of superposition because the functions defining the systems are not all linear, i.e. some are nonlinear. A battery can be modeled as a nonlinear, time-varying system, with state variables that describe such states as the SOC and the SOH. The measurements are typically the voltage and the current. Note that the SOH was not considered by this thesis for reasons described in Section 1.1. Additionally, only the voltage was measured since a known resistive load was used as an input to the system in place of the current measurement. This replacement was done because a piecewise constant discharge profile is convenient for simulation purposes and it is more realistic to have a constant discharge load than a constant discharge current. A state-space representation of the battery system proposed by Chen and Rincón-Mora in [28] is described in more detail in the following chapter.

For linear systems, the optimal filtering solution with respect to the minimum mean squared error (MMSE) is given by the least squares solution, meaning the least squares solution equals the posterior mean. A closed form solution to the discrete-time linear filtering problem is given by the Kalman filter [29], which is a linear MMSSE (LMMSE) filter. Consider a linear system in discrete time with n states and m measurements defined by

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + B_k \mathbf{u}_k + L \mathbf{w}_k \tag{1.4}$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + M \mathbf{v}_k, \tag{1.5}$$

where $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{u} \in \mathcal{R}^{N_u}$, and $\mathbf{z} \in \mathcal{R}^m$ are vectors of the state variables, known inputs, and measurements, respectively; $\mathbf{w} \sim \mathcal{N}(0, Q_k)$, $Q_k \in \mathcal{R}^{N_w \times N_w}$, and $\mathbf{v} \sim \mathcal{N}(0, R_k)$, $R_k \in \mathcal{R}^{N_v \times N_v}$ are normally distributed noise variables; $F \in \mathcal{R}^{n \times n}$, $B \in \mathcal{R}^{n \times N_u}$, $H \in \mathcal{R}^{m \times n}$, $L \in \mathcal{R}^{n \times N_w}$, and $M \in \mathcal{R}^{m \times N_v}$ are matrices; and a subscript k on a variable indicates the value of that variable at time t_k. First, the Kalman filter propagates the estimates of the state variables $\hat{\mathbf{x}}$ and the estimation covariances $P \in \mathcal{R}^{n \times n}$ according to

$$\hat{\mathbf{x}}_{k|k-1} = F_k \hat{\mathbf{x}}_{k-1|k-1} + B_k \mathbf{u}_k \tag{1.6}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + L Q_k L^\top. \tag{1.7}$$

Then, the estimates are updated using the measurements according to

$$\tilde{\mathbf{z}}_k = \mathbf{z}_k - H_k \hat{\mathbf{x}}_{k|k-1} \tag{1.8}$$

$$S_k = H_k P_{k|k-1} H_k^\top + M R_k M^\top \tag{1.9}$$

$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \tag{1.10}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \tilde{\mathbf{z}}_k \tag{1.11}$$

$$P_{k|k} = \left(I - K_k H_k\right) P_{k|k-1}. \tag{1.12}$$

The Kalman filter operates under the assumption that the process and measurement noises are Gaussian, so the posterior distribution is also Gaussian and numerical approximations of the posterior distribution are unnecessary. Under this assumption, the Kalman filter produces the optimal solution in the maximum likelihood (ML) and the maximum a posterior (MAP) senses, in addition to in the MMSE sense. However, the Gaussian assumption is unnecessary for the Kalman filter to produce the optimal MMSE estimate for a general linear system.

For nonlinear systems, optimal filtering solutions are generally intractable, so various numerical approximation methods have been developed. Chen describes seven categories of such methods, namely Gaussian/Laplace approximation, iterative quadrature, multigrid method and point-mass approximation, moment approximation, Gaussian sum approximation, deterministic sampling approximation, and Monte Carlo sampling approximation [1]. Note that only filters using point-based numerical approximation methods were studied, as opposed to those using density-based methods. This was done because typical battery management systems do not have the computational power to employ costly density-based methods, and point-based methods use the simple LMMSE update of the Kalman filter. Point-based and density-based methods are also known as local and global approaches, respectively. Additionally, only filters using methods from the two most popular categories of Gaussian approximation and deterministic sampling approximation [30] were used to further limit the scope of this study.

Gaussian approximation operates by assuming the posterior distribution is Gaussian. Then, the Taylor-series-based extended Kalman filter (EKF) [31] or the Gaussian-describing-function-based statistically linearized filter (SLF) [32] can be used. Li and Jilkov state that the EKF approximates the nonlinear dynamic and measurement functions, while the SLF simplifies the nonlinear stochastic system to a linear system so that linear filtering results are applicable [30]. Deterministic sampling methods are special numerical methods that estimate the mean and covariance. This category includes the unscented Kalman filter (UKF) [33] and the cubature Kalman filter (CKF) [34]. The main advantage of the deterministic sampling methods is they are derivative-free. The remainder of this section details the general implementation of these filters for a discrete-time system of the form

$$\mathbf{x}_k = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_k) + L(\mathbf{x}_{k-1}, \mathbf{u}_k)\mathbf{w}_k \tag{1.13}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + M(\mathbf{x}_k, \mathbf{u}_k)\mathbf{v}_k, \tag{1.14}$$

where $\mathbf{f} \in \mathcal{R}^n$ and $\mathbf{h} \in \mathcal{R}^m$ are nonlinear vector functions, $L \in \mathcal{R}^{n\times N_w}$ and $M \in \mathcal{R}^{m\times N_v}$ are nonlinear matrix functions, and the input $\mathbf{u}$ is assumed to be piecewise constant, meaning $\mathbf{u}(t) = \mathbf{u}(t_k) = \mathbf{u}_k$ for $t_{k-1} < t \leq t_k$. An implicit, first-order Taylor-Heun numerical integration method was used to discretize the continuous-time dynamics $\mathbf{f}$ of the chosen battery model. The specifics of the discretization are discussed in Section 2.2.

1.3.1 Extended Kalman Filter

One of the most popular nonlinear filters is the extended Kalman filter (EKF), which approximates the nonlinear state and measurement functions using Taylor series expansion. This study uses the first-order expansion for the EKF. The prediction step follows the discretization approach proposed by Mazzoni [35] to numerically approximate the continuous-time dynamics in Equation (2.37) and the continuous-time covariance differential equation

$$\dot{P} = F(\mathbf{x}, \mathbf{u})P + PF^\top(\mathbf{x}, \mathbf{u}) + L(\mathbf{x}, \mathbf{u})QL^\top(\mathbf{x}, \mathbf{u}), \tag{1.15}$$

with the Jacobian $F = \partial \mathbf{f}/\partial \mathbf{x}$. The discretization $\mathbf{f}_d$ of $\mathbf{f}$ is given in Section 2.2, and the discretion of the covariance matrix is as follows. For a time step of $\delta = t_k - t_{k-1}$, where m is a positive integer, the prediction step is given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_d(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \tag{1.16}$$

$$P_{k|k-1} = P_{k-1|k-1} + G_\tau \Big\{ F(\hat{\mathbf{x}}_\tau, \mathbf{u}_k) P_k + P_k F^\top(\hat{\mathbf{x}}_\tau, \mathbf{u}_k) + L(\hat{\mathbf{x}}_\tau, \mathbf{u}_k) Q_\tau L^\top(\hat{\mathbf{x}}_\tau, \mathbf{u}_k) \Big\} G_\tau^\top \delta, \tag{1.17}$$

where $t_\tau = t_{k-1} + \delta/2$,

$$G_\tau = \left(I - F(\hat{\mathbf{x}}_\tau)\frac{\delta}{2} \right)^{-1}, \tag{1.18}$$

and

$$\hat{\mathbf{x}}_\tau = \frac{1}{2}\left(\hat{\mathbf{x}}_k + \hat{\mathbf{x}}_{k+1} - F(\hat{\mathbf{x}}_k) f(\hat{\mathbf{x}}_k) \frac{\delta^2}{4} \right). \tag{1.19}$$

It can be seen that the differential equation for the covariance matrix was approximated using a modified Gauss-Legendre formula with an implicit increment rule, following Mazzoni. The numerical approximations for the state and covariance are both A-stable, which is necessary for the chosen battery model, and consistent to the first-order. The update equations for the EKF come from the LMMSE filter and are

$$K_k = P_{k|k-1} H_k^\top \left(H_k P_{k|k-1} H_k^\top + M(\hat{\mathbf{x}}_k) R_k M^\top(\hat{\mathbf{x}}_k) \right)^{-1} \tag{1.20}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \big(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \big) \tag{1.21}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}, \tag{1.22}$$

with the Jacobian $H = \partial \mathbf{h}/\partial \mathbf{x}$.

1.3.2 Unscented Kalman Filter

The unscented Kalman filter (UKF) is an efficient, generally derivative-free filtering algorithm that relies on the unscented transformation (UT). The UT is useful for forming the Gaussian approximation to the joint distribution of random variables x and y for $x \sim \mathcal{N}(m, P)$ and $y = g(x)$, where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a nonlinear function. Then, the first and second moments corresponding to the mean and covariance can be easily found. Specifically, suppose the Gaussian approximation of the joint probability density of x and y has the form

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} m \\ \mu_U \end{bmatrix}, \begin{bmatrix} P & C_U \\ C_U^\top & S_U \end{bmatrix} \right). \tag{1.23}$$

Then, the UT picks $2n+1$ sample points $\{x_i\}$, commonly known as sigma points, along with the same number of weights $\{w_i\}$, as follows [36]. First, the sigma points are chosen from the columns of the matrix $\sqrt{(n+\lambda)P}$, giving

$$x^{(0)} = m_x \tag{1.24}$$

$$x^{(i)} = m_x + \left[\sqrt{(n+\lambda)P}\right]_i, \quad i = 1, \ldots, n \tag{1.25}$$

$$x^{(i)} = m_x - \left[\sqrt{(n+\lambda)P}\right]_{i-n}, \quad i = n+1, \ldots, 2n \tag{1.26}$$

with the weights

$$W_0^{(m)} = \frac{\lambda}{n+\lambda} \tag{1.27}$$

$$W_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{1.28}$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(n+\lambda)}, \quad i = 1, \ldots, 2n. \tag{1.29}$$

The parameter λ is defined as

$$\lambda = \alpha^2(n+\kappa) - n, \tag{1.30}$$

and the constants α, β, and κ are parameters of the method. For the UKF, α is a small positive number, e.g. 10^{-3}, $\beta = 2$ is ideal for a Gaussian distribution, and κ is typically 0. Each sigma point is transformed by

$$y^{(i)} = g(x^{(i)}), \quad i = 0, \ldots, 2n. \tag{1.31}$$

Then, moments are approximated by

$$\mu_U = \sum_{i=0}^{2n} W_i^{(m)} y^{(i)} \tag{1.32}$$

$$S_U = \sum_{i=0}^{2n} W_i^{(c)} (y^{(i)} - \mu_U)(y^{(i)} - \mu_U)^\top \tag{1.33}$$

$$C_U = \sum_{i=0}^{2n} W_i^{(c)} (x^{(i)} - m)(y^{(i)} - \mu_U)^\top. \tag{1.34}$$

The square root of the positive definite matrix P is defined as a matrix A such that $P = AA^\top$. Note that A is not unique. For performance reasons, the Cholesky factorization is typically used. Let the UT algorithm be denoted by

$$[\mu_U, S_U, C_U] = \mathrm{UT}(g, m, P). \tag{1.35}$$

Then, for the discretized system in Equations (1.13) and (1.14), the prediction and update steps for the UKF can be written as

$$[\hat{\mathbf{x}}_{k|k-1}, \tilde{P}_{k|k-1}] = \mathrm{UT}(\mathbf{f}_d, \hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}) \tag{1.36}$$

$$P_{k|k-1} = \tilde{P}_{k|k-1} + L(\hat{\mathbf{x}}_{k|k-1}) Q_k L^\top(\hat{\mathbf{x}}_{k|k-1}) \tag{1.37}$$

$$[\mu_k, \tilde{S}_k, C_k] = \mathrm{UT}(\mathbf{h}, \hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}) \tag{1.38}$$

$$S_k = \tilde{S}_k + M(\hat{\mathbf{x}}_{k|k-1}) R_k M^\top(\hat{\mathbf{x}}_{k|k-1}) \tag{1.39}$$

$$K_k = C_k S_k^{-1} \tag{1.40}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(z_k - \mu_k) \tag{1.41}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^\top. \tag{1.42}$$

Note that the mean and covariances were estimated using the UT, but the update is equivalent to the LMSSE update used in the Kalman filter.

For numerical stability reasons, this study employed a change to the above UKF procedure as suggested by Julier et al. [37]. In Equation (1.36), the covariance is estimated by

$$\tilde{P}_{k|k-1} = \sum_{i=0}^{2n} W_i^{(c)} (\hat{\mathbf{x}}_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1}^{(0)})(\hat{\mathbf{x}}_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1}^{(0)})^\top, \tag{1.43}$$

where the covariance is evaluated about the projected mean rather than the weighted mean. This change ensures the positive definiteness of the covariance matrix, as required by the definition of covariance. Another change, discovered by the author, that results in better numerical stability and lower MSE is the estimation of the cross-covariance in Equation (1.38) by

$$C_k = \sum_{i=0}^{2n} W_i^{(c)} ({}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(i)} - {}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(0)})({}_{P_{k|k-1}}\hat{\mathbf{z}}_k^{(i)} - \mu_k)^\top, \tag{1.44}$$

where the prescripts indicate the estimate covariance matrix that was used to calculate the sigma points, so ${}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(i)}$ comes from the transformation of ${}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k-1|k-1}^{(i)}$ through $\mathbf{f}_d$ and ${}_{P_{k|k-1}}\hat{\mathbf{z}}_k^{(i)}$ comes from the transformation of ${}_{P_{k|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(i)}$ through $\mathbf{h}$.

Note that similar to Equation (1.43), the cross-covariance is evaluated about the projected mean $\hat{\mathbf{x}}_{k|k-1}^{(0)}$ and the weighted mean μ_k. The increase in stability and accuracy with the change in Equation (1.44) was discovered through experimentation. The reason for the improvement is unknown, but the change was used to produce the simulation results.

1.3.3 Cubature Kalman Filter

The cubature Kalman filter (CKF) is similar to the UKF except that it uses the spherical-radial cubature rule rather than the UT to approximate the Gaussian integrals. Indeed, the prediction and update steps of the CKF follow Equations (1.36) to (1.42) except that the UT algorithms in Equations (1.36) and (1.38) are replaced by the corresponding cubature algorithm. This thesis explores the third-order and fifth-order CKFs, whose implementations are discussed in the following two sections.

1.3.3.1 Third-Order CKF

The third-order spherical-radial CKF of Arasaratnam et al. [34, 38] is a special case of the UKF with $\alpha = 1$, $\beta = 0$, and $\kappa = 0$. The third-order cubature rule chooses $2n$ cubature points, giving [34]

$$x^{(i)} = m_x + \left[\sqrt{nP}\right]_i, \quad i = 1, \ldots, n \tag{1.45}$$

$$x^{(i)} = m_x - \left[\sqrt{nP}\right]_{i-n}, \quad i = n+1, \ldots, 2n, \tag{1.46}$$

where the matrix square root is computed using Cholesky factorization, as in the UT. Then, the moments are approximated by

$$\mu_U = \frac{1}{2n}\sum_{i=1}^{2n} y^{(i)} \tag{1.47}$$

$$S_U = \frac{1}{2n}\sum_{i=1}^{2n}(y^{(i)} - \mu_U)(y^{(i)} - \mu_U)^\top \tag{1.48}$$

$$C_U = \frac{1}{2n}\sum_{i=1}^{2n}(x^{(i)} - m)(y^{(i)} - \mu_U)^\top. \tag{1.49}$$

The prediction and update steps follow Equations (1.36) to (1.42) with the UTs in Equations (1.36) and (1.38) replaced by the third-order cubature rule given by Equations (1.45) to (1.49). The resulting third-order CKF is exact for polynomials of order three. Compared to the UKF, the third-order CKF is numerically more stable due to its positive weights. While the UKF has some desirable theoretical properties, its weights can be negative, causing numerical problems in some cases [39].

To increase numerical stability and accuracy, a change similar to that in Equation (1.44) was used, giving

$$C_k = \frac{1}{2n}\sum_{i=0}^{2n}({}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})({}_{P_{k|k-1}}\hat{\mathbf{z}}_k^{(i)} - \mu_k)^\top. \tag{1.50}$$

This change was verified through experimentation to produce better results.

1.3.3.2 Fifth-Order CKF

The fifth-order spherical-radial CKF is a higher-order extension of the third-order CKF that is exact for polynomials of order five. Its cubature rule chooses $2n^2 + 1$

cubature points, giving [40, 41]

$$x^{(0)} = m_x \tag{1.51}$$

$$x^{(i)} = m_x + \left[\sqrt{n+2}e_i\right], \qquad i = 1, \dots, n \tag{1.52}$$

$$x^{(i)} = m_x - \left[\sqrt{n+2}e_{i-n}\right], \qquad i = n+1, \dots, 2n \tag{1.53}$$

$$x^{(i)} = m_x + \left[\sqrt{n+2}s^{+}_{i-2n}\right], \qquad i = 2n+1, \dots, 2n + \frac{n(n-1)}{2} \tag{1.54}$$

$$x^{(i)} = m_x - \left[\sqrt{n+2}s^{+}_{i-2n-n(n-1)/2}\right], \quad i = 2n + \frac{n(n-1)}{2} + 1, \dots, 2n + n(n-1) \tag{1.55}$$

$$x^{(i)} = m_x + \left[\sqrt{n+2}s^{-}_{i-2n-n(n-1)}\right], \qquad i = 2n + n(n-1) + 1, \dots, 2n + \frac{3n(n-1)}{2} \tag{1.56}$$

$$x^{(i)} = m_x - \left[\sqrt{n+2}s^{-}_{i-2n-3n(n-1)/2}\right], \quad i = 2n + \frac{3n(n-1)}{2} + 1, \dots, 2n^2, \tag{1.57}$$

where e_i are the columns of the Cholesky factorization $\sqrt{P}$ and

$$s_i^{\pm} = \left\{ \frac{1}{\sqrt{2}}(e_j \pm e_k) : j < k; j, k = 1, 2, \dots, n \right\} \tag{1.58}$$

are scaled linear combinations of the columns e_i. The weights on the points are

$$W_0 = \frac{2}{n+2} \tag{1.59}$$

$$W_i = \frac{4-n}{2(n+2)^2}, \quad i = 1, \dots, 2n \tag{1.60}$$

$$W_i = \frac{1}{(n+2)^2}, \quad i = 2n+1, \dots, 2n^2. \tag{1.61}$$

Then, moments are approximated by

$$\mu_U = \sum_{i=0}^{2n^2} W_i y^{(i)} \tag{1.62}$$

$$S_U = \sum_{i=0}^{2n^2} W_i (y^{(i)} - \mu_U)(y^{(i)} - \mu_U)^\top \tag{1.63}$$

$$C_U = \sum_{i=0}^{2n^2} W_i (x^{(i)} - m)(y^{(i)} - \mu_U)^\top. \tag{1.64}$$

As in the third-order CKF, the prediction and update steps follow Equations (1.36) to (1.42) with the UTs in Equations (1.36) and (1.38) replaced by the fifth-order cubature rule given by Equations (1.52) to (1.64). Note that unlike the third-order CKF and like the UKF, the weights of the fifth-order CKF can be negative.

As in the third-order CKF, to increase numerical stability and accuracy, a change similar to that in Equation (1.44) was used, giving

$$C_k = \sum_{i=0}^{2n^2} W_i ({}_{P_{k-1|k-1}}\hat{\mathbf{x}}_{k|k-1}^{(i)} - \hat{\mathbf{x}}_{k|k-1})({}_{P_{k|k-1}}\hat{\mathbf{z}}_k^{(i)} - \mu_k)^\top. \tag{1.65}$$

This change was verified through experimentation to produce better results.

1.3.4 Statistically Linearized Filter

In the statistically linearized filter (SLF), the nonlinear state and measurement functions are statistically linearized to minimize the MSE. Then, the resulting linear system can be filtered using the linear Kalman filter. Specifically, given $\mathbf{x} \sim \mathcal{N}(m, P)$, the nonlinear function $\mathbf{f}(\mathbf{x})$ is linearized as [30, 32, 42]

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{b} + A(\mathbf{x} - m), \tag{1.66}$$

where the parameters $\mathbf{b}$ and A are chosen to minimize the error

$$\mathrm{MSE}(\mathbf{b}, A) = E\left[\|\mathbf{f}(\mathbf{x}) - \mathbf{b} - A(\mathbf{x} - m)\|^2\right]. \tag{1.67}$$

Differentiating the MSE expression and setting the derivatives to zero, produces the optimal values

$$\mathbf{b} = E[\mathbf{f}(\mathbf{x})] \tag{1.68}$$

$$A = E[\mathbf{f}(\mathbf{x})(\mathbf{x} - m)^\top]P^{-1}. \tag{1.69}$$

These values reproduce the mean exactly but the covariance is an approximation. The expectations can be calculated analytically or numerically. Due to the difficulty of finding the analytical forms of the expectations, this study chooses to approximated them numerically using the third-order spherical-radial cubature rule described in Section 1.3.3.1, which has the advantages of numerical stability and low computational complexity compared to the UT and fifth-order cubature rule, respectively. The cubature approximation results in

$$\mathbf{b}_\mathbf{x} = E[\mathbf{f}_d(\mathbf{x})] \approx \frac{1}{2n}\sum_{i=1}^{2n} \mathbf{f}_d(\mathbf{x}^{(i)}) \tag{1.70}$$

$$A_\mathbf{x} = E[\mathbf{f}(\mathbf{x})(\mathbf{x} - m)^\top]E[(\mathbf{x} - m)(\mathbf{x} - m)^\top]^{-1} = E[F(\mathbf{x})] \approx \frac{1}{2n}\sum_{i=1}^{2n} F(\mathbf{x}^{(i)}) \tag{1.71}$$

for the expectations of the state mean and covariance and

$$\mathbf{b}_\mathbf{z} = E[\mathbf{h}(\mathbf{x})] \approx \frac{1}{2n}\sum_{i=1}^{2n} \mathbf{h}(\mathbf{x}^{(i)}) \tag{1.72}$$

$$A_\mathbf{z} = E[\mathbf{h}(\mathbf{x})(\mathbf{x} - m)^\top]E[(\mathbf{x} - m)(\mathbf{x} - m)^\top]^{-1} = E[H(\mathbf{x})] \approx \frac{1}{2n}\sum_{i=1}^{2n} H(\mathbf{x}^{(i)}) \tag{1.73}$$

for the expectations of the measurement state and covariance, where the cubature points come from the columns of $\sqrt{nP}$. With the given statistically optimal linearization, the resulting linear system can be filtered using a procedure similar to the linear Kalman filter. The prediction phase consists of

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{b}_{\mathbf{x}} \tag{1.74}$$

$$P_{k|k-1} = A_{\mathbf{x}} P_{k-1|k-1} A_{\mathbf{x}}^{\top} + L(\mathbf{x}, \mathbf{u}) Q_k L^{\top}(\mathbf{x}, \mathbf{u}). \tag{1.75}$$

Note that the form is very similar to the Kalman filter prediction steps given by Equations (1.6) and (1.7), where $E[\mathbf{x} - m] = 0$ has been used to simplify the calculation for $\hat{\mathbf{x}}_{k|k-1}$. The update phase consists of

$$S_k = A_{\mathbf{z}} P_{k|k-1} A_{\mathbf{z}}^{\top} + M(\mathbf{x}, \mathbf{u}) R_k M^{\top}(\mathbf{x}, \mathbf{u}) \tag{1.76}$$

$$K_k = P_{k|k-1} A_{\mathbf{z}}^{\top} S_k^{-1} \tag{1.77}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k (\mathbf{z}_k - \mathbf{b}_{\mathbf{z}}) \tag{1.78}$$

$$P_{k|k} = (I - K_k A_{\mathbf{z}}) P_{k|k-1}. \tag{1.79}$$

Again, this is very similar to the Kalman filter update steps given by Equations (1.8) to (1.12). The SLF is similar to the EKF in the sense that its equations have a similar form to the Kalman filter equations. In fact, ignoring the numerical approximation of the expectations, the SLF uses first-order Fourier-Hermite series expansion to approximate the nonlinear functions whereas the EKF uses Taylor series expansion. Furthermore, the SLF implementation of this study uses the same mean estimation method as the third-order CKF. The covariance estimation differs because the SLF uses information about the first derivatives of the state and measurement functions.

Chapter 2

Problem Setup

2.1 Battery Model

As discussed in the previous section, this thesis considers the electrical-circuit battery model proposed by Chen and Rincón-Mora [28] and shown in Figure 2.1. The left portion of the circuit models the capacity, SOC, and runtime, while the right portion models the transient i-v characteristics. For convenience, the model is designed so that the SOC of the battery equals the voltage V_{SOC}, in volts. The parameters C_{cap} and R_{sd} are assumed constant for a given battery and determine the capacity and self-discharge rate of the battery. The other parameters are all nonlinear functions of V_{SOC} and determine the transient i-v response as well as the open-circuit voltage V_{OC}. From a typical TCL PL-383562 polymer lithium-ion battery, Chen and Rincón-Mora extracted these parameters and fit them to curves, obtaining

$$R_s(V_{\text{SOC}}) = 0.1562e^{-24.37V_{\text{SOC}}} + 0.07446 \tag{2.1}$$

$$R_{ts}(V_{\text{SOC}}) = 0.3208e^{-29.14V_{\text{SOC}}} + 0.04669 \tag{2.2}$$

$$C_{ts}(V_{\text{SOC}}) = -752.9e^{-13.51V_{\text{SOC}}} + 703.6 \tag{2.3}$$

$$R_{tl}(V_{\text{SOC}}) = 6.603e^{-155.2V_{\text{SOC}}} + 0.04984 \tag{2.4}$$

$$C_{tl}(V_{\text{SOC}}) = -6056e^{-27.12V_{\text{SOC}}} + 4475 \tag{2.5}$$

$$V_{\text{OC}}(V_{\text{SOC}}) = -1.031e^{-35V_{\text{SOC}}} + 3.685 + 0.2156V_{\text{SOC}} - 0.1178V_{\text{SOC}}^2 + 0.3201V_{\text{SOC}}^3 \tag{2.6}$$

The resistance and capacitance parameters shown above are approximately constant for SOC > 0.2 and change exponentially for SOC < 0.2. The open-circuit voltage also changes exponentially for SOC < 0.2 but is approximately linear for SOC > 0.2. Note that the capacitances C_{ts} and C_{tl} are negative for SOC values close to zero, which is both unrealistic according to the experimental data collected by Chen and Rinón-Mora and problematic mathematically. To solve this, a lower bound was placed on the V_{SOC} input to the capacitance functions. Thus, for inputs below some threshold value v_T, the capacitances are adjusted to their value at that threshold, producing

$$\hat{C}_{ts}(V_{\text{SOC}}) = \begin{cases} C_{ts}(V_{\text{SOC}}), & V_{\text{SOC}} \geq v_T \\ C_{ts}(v_T), & V_{\text{SOC}} < v_T \end{cases} \tag{2.7}$$

$$\hat{C}_{tl}(V_{\text{SOC}}) = \begin{cases} C_{tl}(V_{\text{SOC}}), & V_{\text{SOC}} \geq v_T \\ C_{tl}(v_T), & V_{\text{SOC}} < v_T \end{cases} \tag{2.8}$$

The threshold v_T was chosen based on the experimental data of Chen and Rinón-Mora, specifically so that the threshold capacitance values are approximately equal to the lowest such values measured by them. A threshold of $v_T = 0.015$ V accomplishes this goal.

This study used the nonlinear parameters given by Chen and Rincón-Mora for

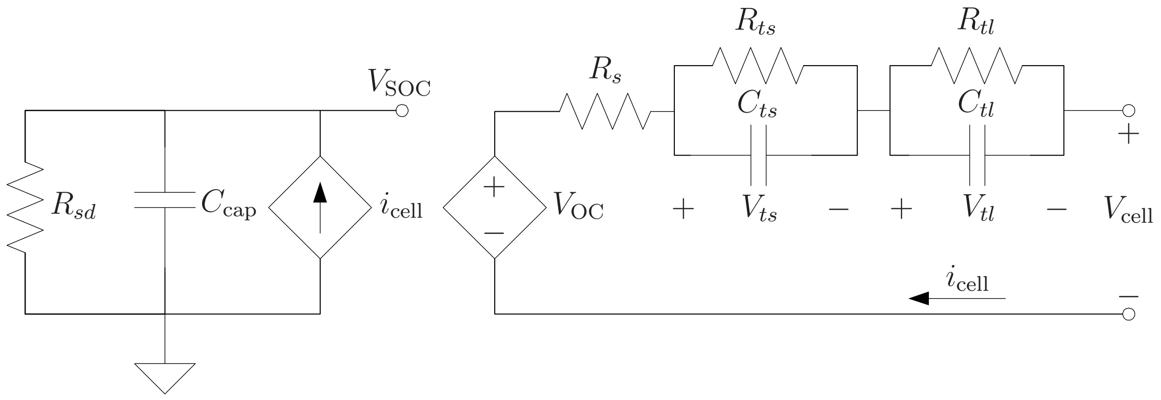

Figure 2.1: Electrical-circuit battery model.

the implementation of a battery using their battery model in Matlab. In addition, the thresholding defined in Equations (2.7) and (2.8) was used with $v_T = 0.015$ V. The other, constant parameters were chosen to produce a capacity of 1 Ah and a self-discharge rate of 4% per month. To do so, the capacitance C_{cap} is calculated to hold the desired capacity when $V_{\text{SOC}} = 1$ V, and then the resistance R_{sd} is set to produce the desired self-discharge rate. For a given capacity of $C^\dagger$ in Ah, C_{cap} needs to be

$$C_{\text{cap}} = \frac{Q}{V_{\text{SOC}}} = \frac{C^\dagger}{1\text{ V}} = 3600C^\dagger\,[\text{F}]. \tag{2.9}$$

Then, the resistance R_{sd} is chosen so that the time constant $\tau = RC$ results in the desired drop of $\xi = 0.04$ over $T = 1$ month as follows

$$V(t) = V_0 e^{-T/\tau} = V_0(1-\xi) \tag{2.10}$$

$$\tau = -T/\ln(1-\xi) = -2592000/\ln 0.96\,[\text{s}]. \tag{2.11}$$

Then, $R_{sd} = \tau/C_{\text{cap}}$. Thus, the parameters are $C_{\text{cap}} = 3600$ F and $R_{sd} = 17.6376$ kΩ.

In order to simulate the use of the modeled battery, discharging and charging loads were implemented, as shown in Figures 2.2. For discharging, a resistive load

R_L is placed across the battery terminals, creating a discharge rate of $i_{\text{cell}} = V_{\text{cell}}/R_L$. For charging, a negative resistance $-R_L$, where $R_L > 0$, is used, creating a charging current of $-i_{\text{cell}} = V_{\text{cell}}/R_L$. Thus, any arbitrary charging or discharging current can be set by choosing the appropriate resistance R_L. Furthermore, an open circuit can be simulated by choosing R_L sufficiently large so that $i_{\text{cell}} \approx 0$. Additional consideration has to be taken to produce constant current and constant voltage charging conditions for standard charging procedure. Typically, the specific battery modeled by the given parameters is charged at a rate of $C_5/5$ until a terminal voltage of 4.2 V is reached, where $C_5/5$ is the discharge rate at which a full battery is completely discharged in 5 hours [43]. Then, the battery is charged at a constant voltage of 4.2 V until the charging current is below $C_5/20$. The constant current condition can be met by varying R_L so that V_{cell}/R_L stays constant, while the constant voltage condition is met by varying R_L so that $i_{\text{cell}}R_L$ stays constant.

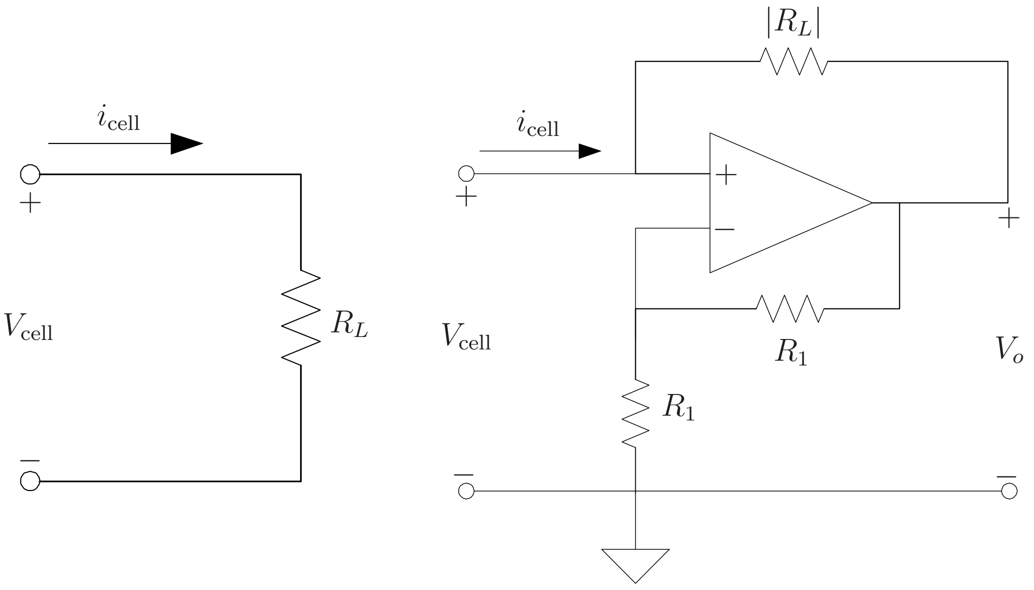

Figure 2.2: Loads to (a) discharge and (b) charge the battery.

This use of the load R_L to control the current i_{cell} suggests that it is the input to the system. Moreover, the outputs of the system are V_{cell} and i_{cell}. However, since knowledge of one of them along with R_L allows for the calculation of the other, the two outputs have a known relationship between them. Therefore, only one of the outputs is necessary to fully define the input-output relationship of the system. In this study, the voltage V_{cell} was chosen as the output.

For ease of numerical simulation, it is useful to find the state-space system for the circuit. The state-space representation is derived using the physical variable definition, in which the state variables are chosen to represent the voltages across the capacitors. Choosing $x_1 = V_{\text{SOC}}$, $x_2 = V_{ts}$, and $x_3 = V_{tl}$ achieves this goal and results in the state-space representation

$$\dot{x}_1 = -\frac{x_1}{R_{sd}C_{\text{cap}}} - \frac{V_{\text{OC}}(x_1) - x_2 - x_3}{(R_s(x_1) + R_L)C_{\text{cap}}} + f_{w,1}(\mathbf{x}, R_L, \mathbf{w}) \tag{2.12}$$

$$\dot{x}_2 = -\frac{x_2}{R_{ts}(x_1)C_{ts}(x_1)} + \frac{V_{\text{OC}}(x_1) - x_2 - x_3}{(R_s(x_1) + R_L)C_{ts}(x_1)} + f_{w,2}(\mathbf{x}, R_L, \mathbf{w}) \tag{2.13}$$

$$\dot{x}_3 = -\frac{x_3}{R_{tl}(x_1)C_{tl}(x_1)} + \frac{V_{\text{OC}}(x_1) - x_2 - x_3}{(R_s(x_1) + R_L)C_{tl}(x_1)} + f_{w,3}(\mathbf{x}, R_L, \mathbf{w}) \tag{2.14}$$

$$V_{\text{cell}} = \frac{V_{\text{OC}}(x_1) - x_2 - x_3}{1 + R_s(x_1)/R_L} + f_v(\mathbf{x}, R_L, \mathbf{v}), \tag{2.15}$$

where R_L is the input to the system, V_{cell} is the output, f_w is the process noise function, f_v is the measurement noise function, and the nonlinear parameters depending on x_1 are given by Equations (2.1) to (2.6) along with the thresholding defined in Equations (2.7) and (2.8). It is obvious from this formulation that the system is nonlinear to both the input and the states. In order to establish the noise expressions, the types of noise present in the battery have to first be determined.

This thesis assumes that the process and measurement noises in this system are due to thermal noise in the resistances for the internal impedance of the battery R_s, R_{ts}, and R_{tl}, and for the load R_L. This is motivated by measurements of the voltage noise in batteries conducted by Boggs et al. that showed the measured noise is mainly

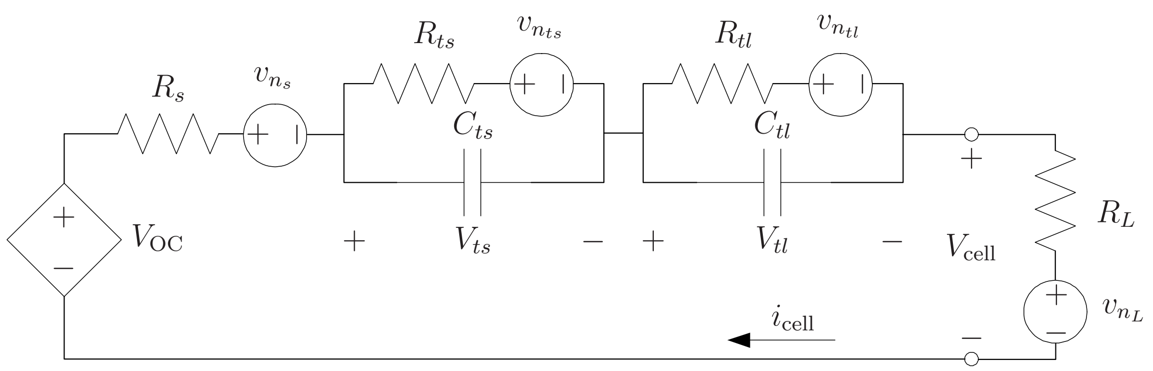

Figure 2.3: Modeling of thermal noise in resistances as voltage sources in series with the resistances.

due to thermal noise; the correlation between the battery terminals suppresses shot noise [44]. This thermal noise is assumed to be Gaussian white noise with a power spectral density (PSD) of [45]

$$S_n(\omega) \cong 2kT \text{ watts per Hz} \qquad \text{for} \qquad |\omega| \ll 2\pi kT/h, \tag{2.16}$$

where T is the temperature of the conducting medium in Kelvin, k is the Boltzmann's constant, and h is the Planck's constant. Figure 2.3 shows that the thermal noise due to the resistances is modeled as voltage sources in series with the resistances, with PSDs of $S_v(\omega) = 2kTR$ for a corresponding resistance R. Using this definition, the noise functions are given by

$$f_{w,1} = \frac{v_{n_s} + v_{n_L}}{(R_s(x_1) + R_L)C_{\mathrm{cap}}} \tag{2.17}$$

$$f_{w,2} = \frac{v_{n_{ts}}}{R_{ts}(x_1)C_{ts}(x_1)} - \frac{v_{n_s} + v_{n_L}}{(R_s(x_1) + R_L)C_{ts}(x_1)} \tag{2.18}$$

$$f_{w,3} = \frac{v_{n_{tl}}}{R_{tl}(x_1)C_{tl}(x_1)} - \frac{v_{n_s} + v_{n_L}}{(R_s(x_1) + R_L)C_{tl}(x_1)} \tag{2.19}$$

$$f_v = -\frac{v_{n_s} + v_{n_L}}{1 + R_s(x_1)/R_L}. \tag{2.20}$$

It can be seen that the resistances change over time, which causes the covariance of the sources v_n to also change. For the purposes of modeling, it is useful to define noise variables that have constant covariance. Using the square root of the power supplied by the noise sources as the noise variables accomplishes this goal and produces the variables $w_1 = v_{n_s}/\sqrt{R_s}$, $w_2 = v_{n_{ts}}/\sqrt{R_{ts}}$, $w_3 = v_{n_{tl}}/\sqrt{R_{tl}}$, and $w_4 = v_{n_L}/\sqrt{|R_L|}$ along with $v_1 = v_{n_s}/\sqrt{R_s}$ and $v_2 = v_{n_L}/\sqrt{|R_L|}$, which all have constant covariances of $2kT$. Note the use of the absolute value of R_L in the definition of w_4 and v_2, since R_L can become negative. In the case of $R_L < 0$, their covariances remain at $2kT$ while the sign of v_{n_L} is negated, which is implemented in the system using the signum function, defined as

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{2.21}$$

Then, the state space representation of the system becomes

$$\dot{x}_1 = -\frac{x_1}{R_{sd}C_{\text{cap}}} - \frac{V_{\text{OC}}(x_1) - x_2 - x_3 - \sqrt{R_s(x_1)}w_1 - \operatorname{sgn}(R_L)\sqrt{|R_L|}w_4}{(R_s(x_1) + R_L)C_{\text{cap}}} \tag{2.22}$$

$$\dot{x}_2 = -\frac{x_2 - \sqrt{R_{ts}(x_1)}w_2}{R_{ts}(x_1)C_{ts}(x_1)} + \frac{V_{\text{OC}}(x_1) - x_2 - x_3 - \sqrt{R_s(x_1)}w_1 - \operatorname{sgn}(R_L)\sqrt{R_L}w_4}{(R_s(x_1) + R_L)C_{ts}(x_1)} \tag{2.23}$$

$$\dot{x}_3 = -\frac{x_3 - \sqrt{R_{tl}(x_1)}w_3}{R_{tl}(x_1)C_{tl}(x_1)} + \frac{V_{\text{OC}}(x_1) - x_2 - x_3 - \sqrt{R_s(x_1)}w_1 - \operatorname{sgn}(R_L)\sqrt{R_L}w_4}{(R_s(x_1) + R_L)C_{tl}(x_1)} \tag{2.24}$$

$$V_{\text{cell}} = \frac{V_{\text{OC}}(x_1) - x_2 - x_3 - \sqrt{R_s(x_1)}v_1 - \operatorname{sgn}(R_L)\sqrt{R_L}v_2}{1 + R_s(x_1)/R_L}. \tag{2.25}$$

It can be seen that the system can be written in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) \tag{2.26}$$

$$y = h(\mathbf{x}, \mathbf{u}, \mathbf{v}). \tag{2.27}$$

It is useful to find the derivatives of $\mathbf{f}$ and h with respect to $\mathbf{x}$, $\mathbf{w}$, and $\mathbf{v}$ for use with the filters. This thesis defines the Jacobians $F = (\partial \mathbf{f}/\partial \mathbf{x})$, $H = (\partial h/\partial \mathbf{x})$, $L = (\partial \mathbf{f}/\partial \mathbf{w})$, and $M = (\partial h/\partial \mathbf{v})$. Their equations are

$$F = \left[\begin{array}{ccc}
\dfrac{-1}{R_{sd}C_{\text{cap}}} + \dfrac{\begin{array}{r}(Voc - x_2 - x_3)R_s' \\ -\,(R_s + R_L)V_{\text{OC}}'\end{array}}{(R_s + R_L)^2 C_{\text{cap}}} & & \\
\dfrac{(R_{ts}C_{ts})' x_2}{R_{ts}C_{ts}} + \dfrac{\begin{array}{r}(R_s + R_L)C_{ts}V_{\text{OC}}' \\ -\,(Voc - x_2 - x_3)\big[(R_s + R_L)C_{ts}\big]'\end{array}}{\big[(R_s + R_L)C_{ts}\big]^2} & & \\
\dfrac{(R_{tl}C_{tl})' x_3}{R_{tl}C_{tl}} + \dfrac{\begin{array}{r}(R_s + R_L)C_{tl}V_{\text{OC}}' \\ -\,(Voc - x_2 - x_3)\big[(R_s + R_L)C_{tl}\big]'\end{array}}{\big[(R_s + R_L)C_{tl}\big]^2} & & \\
\cdots & \dfrac{1}{(R_s + R_L)C_{\text{cap}}} & \dfrac{1}{(R_s + R_L)C_{\text{cap}}} \\
\cdots & \dfrac{-1}{R_{ts}C_{ts}} + \dfrac{-1}{(R_s + R_L)C_{ts}} & \dfrac{-1}{(R_s + R_L)C_{ts}} \\
\cdots & \dfrac{-1}{(R_s + R_L)C_{tl}} & \dfrac{-1}{R_{tl}C_{tl}} + \dfrac{-1}{(R_s + R_L)C_{tl}}
\end{array}\right] \tag{2.28}$$

$$H = \left[\begin{array}{ccc}
\dfrac{\begin{array}{r}(1 + R_s/R_L)V_{\text{OC}}' \\ -\,(V_{\text{OC}} - x_2 - x_3)R_s'/R_L\end{array}}{(1 + R_s/R_L)^2} & \dfrac{-1}{1 + R_s/R_L} & \dfrac{-1}{1 + R_s/R_L}
\end{array}\right] \tag{2.29}$$

$$L = \begin{bmatrix} \dfrac{\sqrt{R_s}}{(R_s + R_L)C_{\text{cap}}} & 0 & 0 & \dfrac{\operatorname{sgn} R_L \sqrt{|R_L|}}{(R_s + R_L)C_{\text{cap}}} \\[2ex] \dfrac{-\sqrt{R_s}}{(R_s + R_L)C_{\text{cap}}} & \dfrac{\sqrt{R_{ts}}}{R_{ts}C_{ts}} & 0 & \dfrac{-\operatorname{sgn} R_L \sqrt{|R_L|}}{(R_s + R_L)C_{\text{cap}}} \\[2ex] \dfrac{-\sqrt{R_s}}{(R_s + R_L)C_{\text{cap}}} & 0 & \dfrac{\sqrt{R_{tl}}}{R_{tl}C_{tl}} & \dfrac{-\operatorname{sgn} R_L \sqrt{|R_L|}}{(R_s + R_L)C_{\text{cap}}} \end{bmatrix} \tag{2.30}$$

$$M = \begin{bmatrix} \dfrac{-\sqrt{R_s}}{1 + R_s/R_L} & \dfrac{-\operatorname{sgn} R_L \sqrt{|R_L|}}{1 + R_s/R_L} \end{bmatrix}, \tag{2.31}$$

where $()'$ indicates derivation with respect to x_1 and the dependence on x_1 has been omitted due to space constraints. Furthermore, it is useful to find the Hessian of $\mathbf{f}$ with respect to $\mathbf{x}$. Due to symmetry and $\partial^2 f_k/\partial x_i \partial x_j = 0$ for $i, j = 2, 3$, only the first column of each tensor component of the Hessian is given. The resultant Hessian is

$$\frac{\partial^2 f_1}{\partial x_i \partial x_1} = \begin{bmatrix} \dfrac{\begin{aligned}(R_s + R_L)\big[(V_{\text{OC}} - x_2 - x_3)R_s'' - (R_s + R_L)V_{\text{OC}}''\big] \\ - 2R_s'\big[(V_{\text{OC}} - x_2 - x_3)R_s' - (R_s + R_L)V_{\text{OC}}'\big]\end{aligned}}{(R_s + R_L)^3 C_{\text{cap}}} \\[3ex] \dfrac{-R_s'}{(R_s + R_L)^2 C_{\text{cap}}} \\[3ex] \dfrac{-R_s'}{(R_s + R_L)^2 C_{\text{cap}}} \end{bmatrix} \tag{2.32}$$

$$\frac{\partial^2 f_2}{\partial x_i \partial x_1} = \begin{bmatrix} \dfrac{\begin{aligned}\big\{R_{ts}C_{ts}(R_{ts}C_{ts})'' \\ - \big[(R_{ts}C_{ts})'\big]^2\big\}x_2\end{aligned}}{(R_{ts}C_{ts})^2} + \dfrac{\begin{aligned}(R_s + R_L)C_{ts}\big\{(R_s + R_L)C_{ts}V_{\text{OC}}'' - (V_{\text{OC}} - x_2 - x_3)\big[(R_s + R_L)C_{ts}\big]''\big\} \\ + 2\big[(R_s + R_L)C_{ts}\big]'\big\{(R_s + R_L)C_{ts}V_{\text{OC}}' - (Voc - x_2 - x_3)\big[(R_s + R_L)C_{ts}\big]'\big\}\end{aligned}}{\big[(R_s + R_L)C_{ts}\big]^3} \\[3ex] \dfrac{(R_{ts}C_{ts})'}{R_{ts}C_{ts}} + \dfrac{\big[(R_s + R_L)C_{ts}\big]'}{\big[(R_s + R_L)C_{ts}\big]^2} \\[3ex] \dfrac{\big[(R_s + R_L)C_{ts}\big]'}{\big[(R_s + R_L)C_{ts}\big]^2} \end{bmatrix} \tag{2.33}$$

$$\frac{\partial^2 f_3}{\partial x_i \partial x_1} = \begin{bmatrix} \dfrac{\begin{array}{l}\big\{R_{tl}C_{tl}(R_{tl}C_{tl})'' \\ \quad - \big[(R_{tl}C_{tl})'\big]^2\big\}x_3\end{array}}{(R_{tl}C_{tl})^2} + \dfrac{\begin{array}{l}(R_s+R_L)C_{tl}\big\{(R_s+R_L)C_{tl}V''_{\mathrm{OC}} - (V_{\mathrm{OC}} - x_2 - x_3)\big[(R_s+R_L)C_{tl}\big]''\big\} \\ \quad + 2\big[(R_s+R_L)C_{tl}\big]'\big\{(R_s+R_L)C_{tl}V'_{\mathrm{OC}} - (Voc - x_2 - x_3)\big[(R_s+R_L)C_{tl}\big]'\big\}\end{array}}{\big[(R_s+R_L)C_{tl}\big]^3} \\ \dfrac{\big[(R_s+R_L)C_{tl}\big]'}{\big[(R_s+R_L)C_{tl}\big]^2} \\ \dfrac{(R_{tl}C_{tl})'}{R_{tl}C_{tl}} + \dfrac{\big[(R_s+R_L)C_{tl}\big]'}{\big[(R_s+R_L)C_{tl}\big]^2} \end{bmatrix} \tag{2.34}$$

Additionally, the covariances of the error variables are

$$Q = 2kTI_4 \tag{2.35}$$

$$R = 2kTI_2, \tag{2.36}$$

where I_n is a $n \times n$ identity matrix.

2.2 Discretization of System Dynamics

Note that the choice of discretization method tends to depend on the filter, especially for highly nonlinear systems. Therefore, the simulations used a continuous-discrete (mixed) time system, with the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)) \tag{2.37}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}(t_k), \mathbf{u}(t_k), \mathbf{v}(t_k)), \tag{2.38}$$

where $\mathbf{x}$, $\mathbf{u}$, $\mathbf{z}$, $\mathbf{w}$, and $\mathbf{v}$ are the vectors of the states, the inputs, the measurements, the process noises, and the measurement noises, respectively. For simplicity of simulation, the input $\mathbf{u}(t)$ is assumed to be constant over each time step so that $\mathbf{u}(t) = \mathbf{u}(t_k)$ for $t_k \leq t < t_{k+1}$. For the prediction phase, the continuous-time dynamics were solved numerically, with each filter using different approximation techniques. The state error covariances are also approximated numerically. Then, the updates were performed based on the update of the linear Kalman filter.

Furthermore, note that the system is extremely stiff for some inputs $\mathbf{u}$, which can be seen from the stiffness ratio based on the ratio of the largest eigenvalue of F to its smallest eigenvalue, where F is the Jacobian of the dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$; when the stiffness ratio is much greater than unity, the system is stiff [46, 47]. From EIS studies of batteries, the major chemical processes have widely differing time constants; low frequency mass transport effects like diffusion are on the order of 10^{-6} to 10^{0} Hz, middle frequency effects caused by charge transfer and the electrochemical double layer are on the order of 10^{0} to 10^{3} Hz, and the high frequency conductance and skin effects are on the order of 10^{3} to 10^{4} Hz [48]. Therefore, the approximate stiffness ratio is $10^{10} \gg 1$, and the system is stiff. As a result of the stiffness, any numerical integration method needs to be A-stable, i.e. the method converges for all systems whose eigenvalues have negative real parts. For example, simulation results show that the fourth-order Runge-Kutta method diverges even at step sizes $< 10^{-2}$ seconds.

Särkkä and Solin state that a linearized discretization approach, in which the continuous-time system is first discretized and then approximated as Gaussian, tends to work better than a discretized linearization approach, in which the system is first approximated as a Gaussian process and then discretized [39]. This thesis follows this guideline and performs the prediction using linearized approximations of a discretization of the continuous-time dynamics. To increase the accuracy of the discretized integration in the prediction phase, the sampling period is divided into M steps of equal length and the integration is performed in M steps. The specifics of the linearized discretization approach for each filter along with their general implementation are discussed in the remainder of this chapter.

One of the most popular nonlinear filters is the extended Kalman filter (EKF), which approximates the nonlinear state and measurement equations, typically using Taylor series expansion. The standard Kalman filter formulas are used for the update. For the discrete-time EKF, Taylor series expansion can be directly used on the system dynamics to linearize them. For the mixed-time system of this study, a linearized discretization approach proposed by Mazzoni [35] is used, in which the dynamics are first discretized and then approximated using Taylor series expansion. This approach has the advantage of A-stability. The discretization is performed using the trapezoidal approximation (Heun's method) of Equation (2.37). For convenience, denote the value of a quantity at time t_k using the subscript k and assume that $\delta = t_{k+1} - t_k$ is the time step. Additionally, a subscript of $k|k-1$ indicates the value at time t_k given the information at t_{k-1}. Then, the approximation produces

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + \frac{1}{2}\big(\mathbf{f}(\mathbf{x}_k) + \mathbf{f}(\mathbf{x}_{k+1})\big)\delta. \tag{2.39}$$

The vector field $\mathbf{f}$ at $\mathbf{x}_{k+1}$ is approximated by first-order Taylor expansion around $\mathbf{x}_k$, giving

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k)\delta + \frac{1}{2}F(\mathbf{x}_k)\left(\mathbf{x}_{k+1} - \mathbf{x}_k\right)\delta, \tag{2.40}$$

where $F(\mathbf{x}_k)$ is the Jacobian of $\mathbf{f}$ at $\mathbf{x}_k$. Solving for $\mathbf{x}_{k+1}$ yields

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + \left(I - F(\mathbf{x}_k)\frac{\delta}{2}\right)^{-1}\mathbf{f}(\mathbf{x}_k)\delta, \tag{2.41}$$

with the identity matrix I. This Taylor-Heun scheme uses linear Taylor expansion of f rather than then Euler prediction of the standard Heun scheme. This numerical integration scheme is convergent with order $\mathcal{O}(\delta^2)$ and A-stable [35]. For the state error covariance ODE given by

$$\dot{P} = F(\mathbf{x})P + PF^{\top}(\mathbf{x}) + L(\mathbf{x})Q(\mathbf{x})L^{\top}(\mathbf{x}), \tag{2.42}$$

where Q is the covariance of the error variables and L is the Jacobian of $\mathbf{f}$ with respect to the state error variables $\mathbf{w}$, the numerical integration method should have the key features of being consistent with the same order as the Taylor-Heun scheme, able to process nonautonomous differential equations, and A-stable. Mazzoni proposed using a modified Gauss-Legendre formula with an implicit increment rule, producing

$$P_{k+1} \approx P_k + M_\tau \left(F(\mathbf{x}_\tau)P_k + P_k F^{\top}(\mathbf{x}_\tau) + L(\mathbf{x}_\tau)Q(\mathbf{x}_\tau)L^{\top}(\mathbf{x}_\tau) \right) M_\tau^{\top} \delta, \tag{2.43}$$

with

$$M_\tau = \left(I - F(\mathbf{x}_\tau)\frac{\delta}{2} \right)^{-1} \quad \text{and} \quad \tau = t_k + \frac{\delta}{2}. \tag{2.44}$$

The value of $\mathbf{x}_\tau$ can be interpolated from $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ with a precision of $\mathcal{O}(\delta^3)$ using series expansion, producing

$$\mathbf{x}_\tau \approx \frac{1}{2} \left(\mathbf{x}_k + \mathbf{x}_{k+1} - F(\mathbf{x}_k)f(\mathbf{x}_k)\frac{\delta^2}{4} \right). \tag{2.45}$$

This modified Gauss-Legendre approximation is A-stable, consistent with order $\mathcal{O}(\delta^2)$, and ensures the positive definiteness of the resultant error covariance matrix [35]. This numerical integration can be repeated multiple times with a smaller step size should greater accuracy be desired. The update equations for the EKF come from the LMMSE filter and are

$$K_k = P_{k|k-1} H_k^\top \left(H_k P_{k|k-1} H_k^\top + M(\mathbf{x}_k) R(\mathbf{x}_k) M^\top(\mathbf{x}_k) \right)^{-1} \tag{2.46}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k \big(\mathbf{z}_k - \mathbf{h}(\mathbf{x}_{k|k-1}) \big) \tag{2.47}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}, \tag{2.48}$$

where R is the covariance of the measurement error variables, M is the Jacobian of $\mathbf{f}$ with respect to the measurement error variables $\mathbf{v}$, and H_k is the Jacobian of $\mathbf{f}$ at $\mathbf{x}_{k|k-1}$.

2.3 Simulation Setup

In order to simulate the stochastic system, the covariances of the noises as well as their generation and simulation methods have to be determined. This thesis assumed that a standard temperature of $T = 290$ Kelvin. Therefore, the noise variables have covariances of $\sigma^2 = 2kT = 8.0078 \times 10^{-21}$ W/Hz. Furthermore, to better differentiate the performance of the filters, additional measurement noise was introduced assuming an oscilloscope was used to perform the measurement. Specifically, consider the Tektronix TBS1022 oscilloscope with a DC gain accuracy of ±3% of the full range. Based on numerical simulations, an approximate range of 4 V peak-to-peak is necessary to fully capture the range of possible V_{cell} values, resulting in a measurement inaccuracy of ±0.12 V. This noise is assumed to be white Gaussian noise, and its range lies within three standard deviations. Thus, an additional measurement noise variable v_3 is introduced with a variance of

$$\sigma_{v_3}^2 = \left(\frac{0.12 \text{ V}}{3} \right)^2 = 0.0016 \text{ V}^2. \tag{2.49}$$

Then, the new measurement covariance matrix is

$$R = \operatorname{diag}(2kT, 2kT, 0.0016), \tag{2.50}$$

and the corresponding Jacobian $M = (\partial h/\partial \mathbf{v})$ is

$$M = \begin{bmatrix} \frac{-\sqrt{R_s}}{(R_s + R_L)C_{\text{cap}}} & \frac{-\operatorname{sgn} R_L \sqrt{|R_L|}}{(R_s + R_L)C_{\text{cap}}} & 1 \end{bmatrix}. \tag{2.51}$$

In order to numerically simulate the effect of white noise on a system, the simulation time step must be sufficiently smaller than the time constant of the fastest battery process. This is approximated as the product of the constant terms of the functions for R_{ts} and C_{ts}, halved to satisfy Nyquist conditions. Then, the simulation time step is taken to be 1/100 of the calculated maximum time step, as suggested by Matlab documentation. Thus, the simulations used a time step of

$$\delta_{\text{sim}} = \frac{R_{ts,\text{const.}} C_{ts,\text{const.}}}{200} = \frac{0.04669 \times 703.6}{200} = 0.164255 \text{ s.} \tag{2.52}$$

Simulation showed that the resulting time step is sufficiently small to capture the effects of the white noise, i.e. further reducing the step size had negligible effect. With the chosen step size, the noise is simulated as band-limited white Gaussian noise with a correlation time equal to the step size. At each time step, the noise values for the process noise sources are generated using random number generators producing normally-distributed numbers with means of zero and variances equal to the diagonals of the covariance matrix divided by the correlation time. The scaling of the variance by the correlation time ensures the response of the system to the approximate white noise has the same covariance as it would have to actual white noise. Note that the measurement noise is bandlimited not by the system but by the measurement device, in this case an oscilloscope. Thus, the calculated variance already takes into account the measurement bandwidth of the oscilloscope and scaling is unnecessary. For reproducibility, the random number generators were seeded with predictable numbers. This was done in Matlab by first seeding the main random number generator with a seed of 0. Then, for each Monte Carlo trial, five positive integers were generated, for the five noise sources, with the integers uniformly distributed between 1 and $2^{32} - 1$. These integers were used to seed the random number generators in a Simulink model. The use of the Simulink model allows for the random number generators to be easily seeds with different integers and does not affect the predictable sequence in Matlab.

Then, the battery was simulated using a Simulink model with the fourth-order Runge-Kutta method and an initial condition of $x_0 = [1, 0, 0]^\top$. A total of 100 Monte Carlo runs of the battery were performed with seed values for the noise sources generated using the method mentioned in the previously. Figure 2.4 shows the input load on the battery system, where the values off the graph are idle periods, simulated using a very large input of $R_L = 10^{10}\ \Omega$ so that the battery current is approximately zero. It can be seen that the input is piecewise constant. This input was chosen to to test the performance of the filters by gradually increasing the strength of the nonlinear rate-capacity and recovery effects. Note that care was taken to ensure the SOC remained within the range of zero to one, so discharge and charge times could not always be equal. Intially, the battery is idle for 10 minutes to allow the estimated covariances of the filters to converge. Then, the battery is discharged and charged at $R_L = 20\ \Omega$ for 290 and 270 minutes, respectively. The resulting low current from this load is approximately the testing current $C_5/5$ used to determine battery capacity, as discussed in Section 2.1. In order to increase the strength of the nonlinear effects, the absolute value of the current was increased by decreasing the input to $R_L = 10\ \Omega$ and discharging and charging for 150 minutes, each. Then, the input was further reduced to $R_L = 5\ \Omega$ and discharged and charged for 80 minutes and 70 minutes, respectively. Next, the battery was discharged and charged at $R_L = 4\ \Omega$ for 60 minutes, each. Following were discharge and charge periods at $R_L = 2\ \Omega$ for 35, 25, 25, 20, 25, 25, 20, and 15 minutes. The high current results in very strong rate-capacity effects. Finally, the battery was rested for 70 minutes, discharged at $R_L = 2\ \Omega$ for 25 minutes, rested for 75 minutes, and charged at $R_L = 10\ \Omega$ for 135 minutes. The two resting periods should show the strongest recovery effect. The total input time is 1635 minutes. The SOC and the noisy measurement of V_{cell} resulting from the given input is shown one Monte Carlo trial in Figures 2.5 and 2.6, respectively.

The Monte Carlo trials used varying sampling periods, calculated as some multiple K of the simulation step size. For example, for a desired sampling period of 300 seconds, the actual sampling period is

$$T_s = K\delta_{\text{sim}} = 1826 \times 0.164255 \text{ s} = 299.93 \text{ s}, \tag{2.53}$$

where the factor K is chosen to minimize the difference between the desired and actual periods. For convenience, the sampling period will refer to the actual sampling period calculated in this manner.

2.4 Filtering Setup

The filters used in this study were implemented in Fortran for speed reasons. The programs used LAPACK 3.5.0 and were compiled using gfortran 4.8 for a 32-bit Cygwin environment on a 64-bit Windows 7 computer with optimization flags of

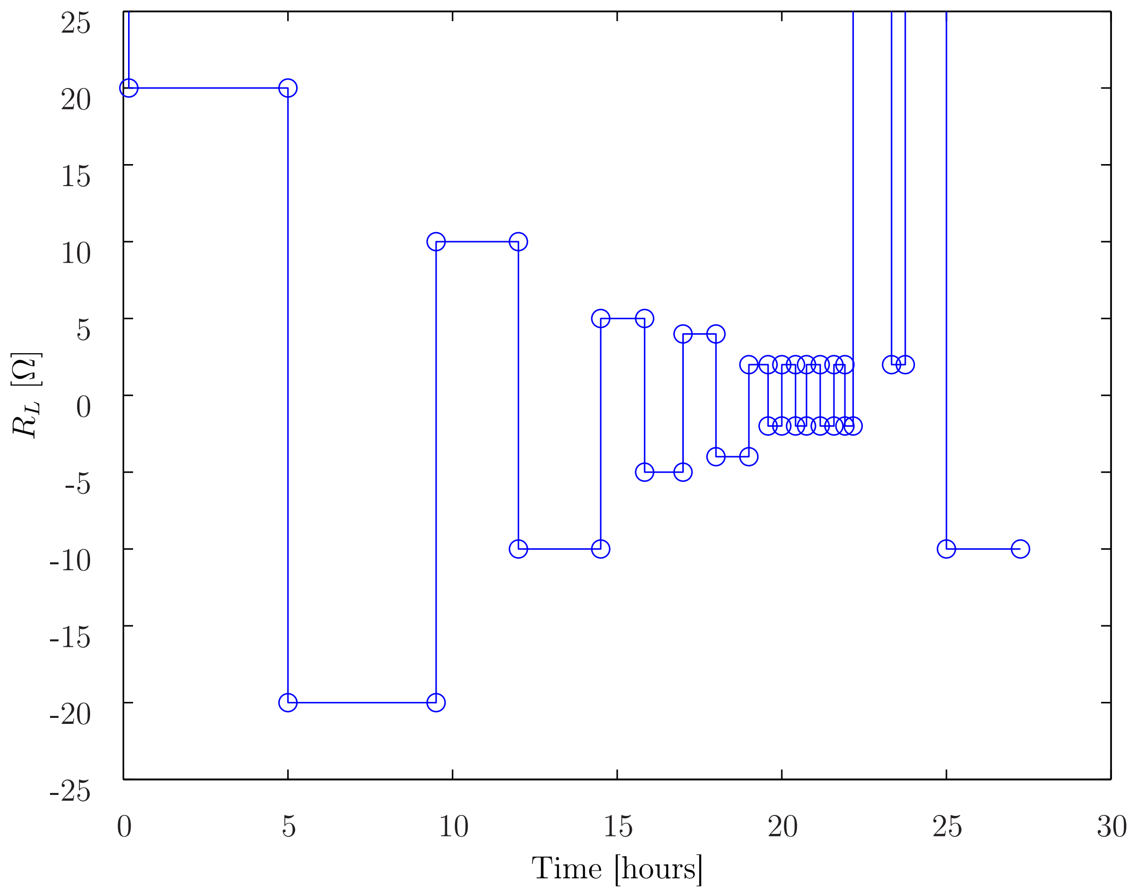

Figure 2.4: Input load R_L on the battery.

-03. Due to the large size of the data, the input and output were performed using files. The time measurements were of the CPU time used by the filters, disregarding the time taken to read and write the data. The use of CPU time results in less variability between runs. Additionally, to increase the accuracy of the discretized integration in the prediction steps, the sampling period is divided by a positive integer M and the integration is performed in M steps. Thus, the prediction and update portions of the filters can be calculated at different rates.

The formulas used to implement the filters were already described in Chapter 1. The only filter with tunable parameters was the UKF. Parameter values of $\alpha = 0.05$, $\beta = 2$, and $\kappa = 0$ were used, where the values were chosen to be the typical values

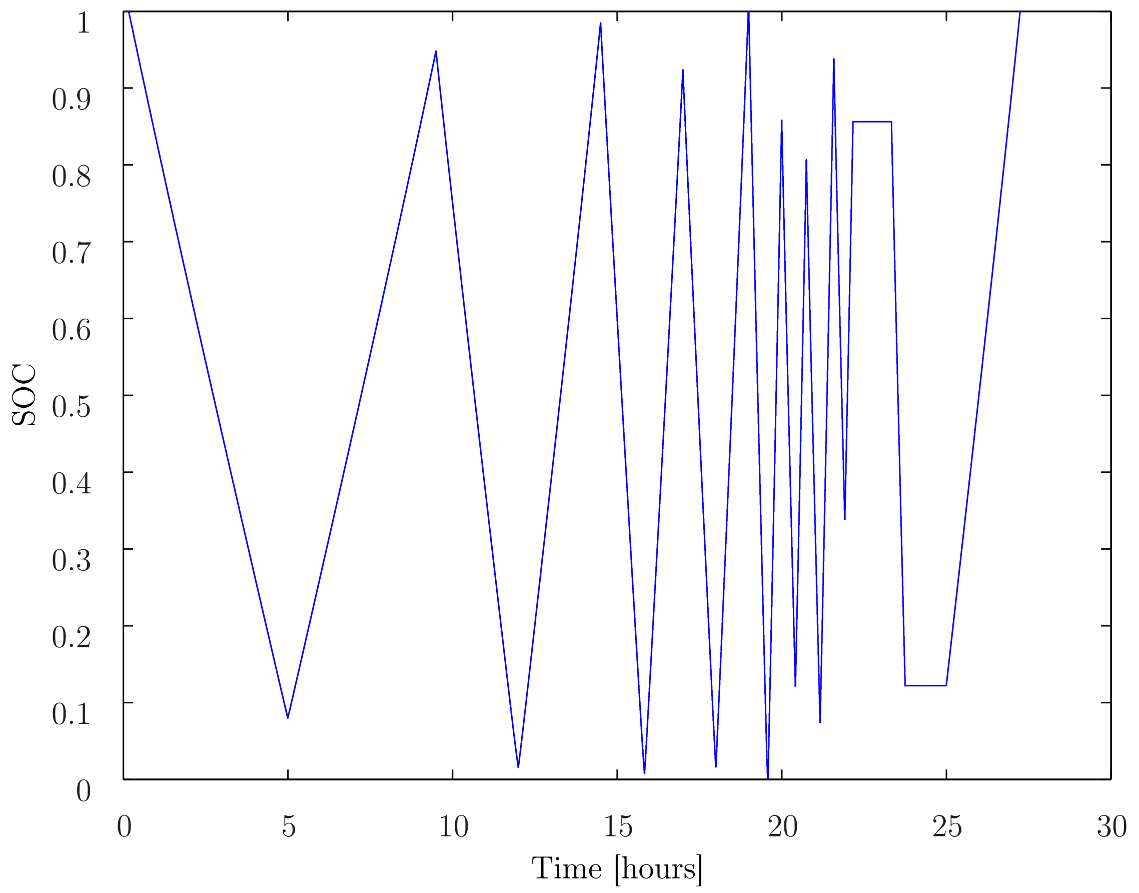

Figure 2.5: True SOC for one run due to input load.

for a Gaussian distribution. It was noticed that α could not be too small for this problem; otherwise, the covariance matrix quickly loses positive definiteness. The choice of α was a compromise between the typical choice of a small value and the stability gained from a larger value. Additionally, as α approaches 1, the UKF becomes very similar to the third-order CKF used by this thesis, so the chosen value is on the small side to better differentiate the two filters. The filtering was performed assuming an initial state of $x_0 = [1, 0, 0]^\top$ and an initial covariance matrix of $P_0 = 10^{-6} I_3$, where the initial state was chosen to match the actual state and the initial covariance was experimentally tuned for fast convergence.

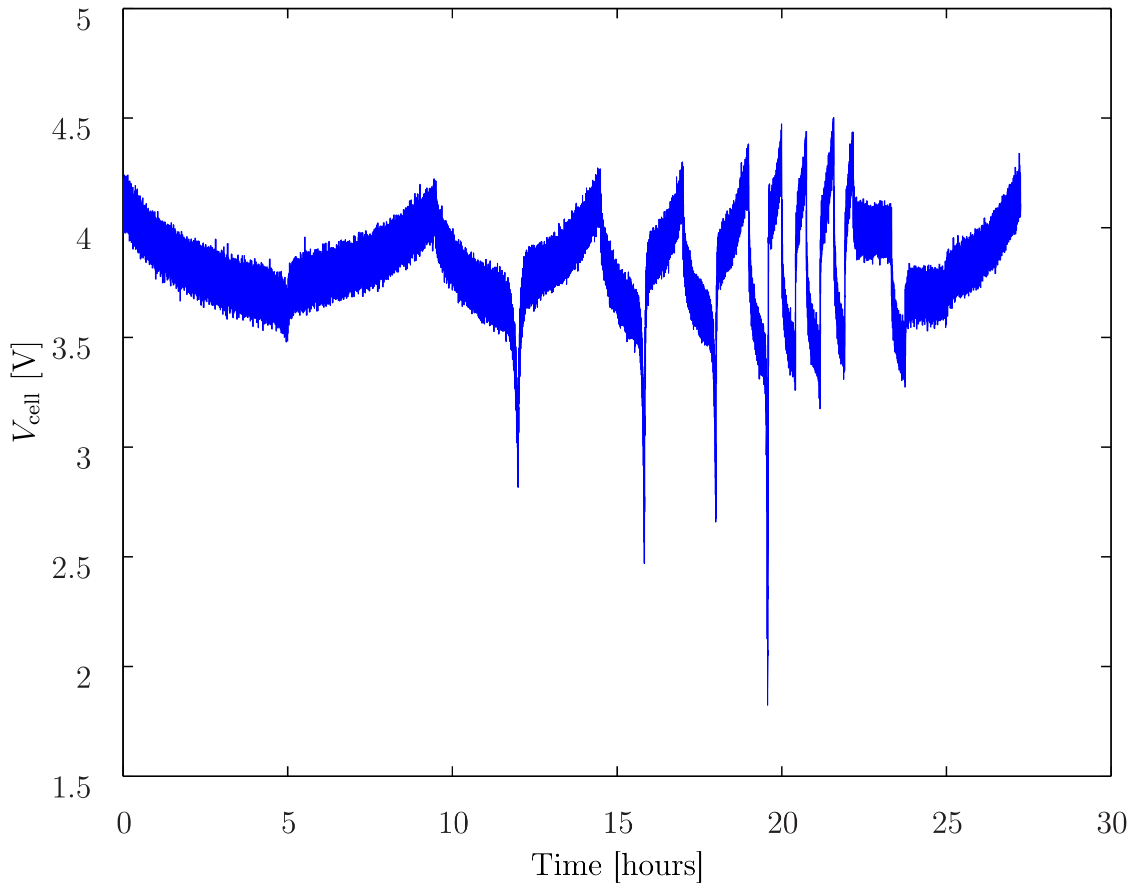

Figure 2.6: Noisy measurement V_{cell} for one run.

Chapter 3

Filtering Results

The study was concerned with the accuracy and speed of the nonlinear filters on estimation of the SOC x_1. The accuracy was measured using the mean RMSE (MRMSE). The RMSE is defined as

$$\mathrm{RMSE}(kT_s) = \sqrt{\frac{1}{N_{\text{trials}}} \sum_{j=1}^{N_{\text{trials}}} \left(\hat{x}_1^j(kT_s) - x_1^j(kT_s) \right)^2}, \tag{3.1}$$

where T_s is the sample period and the superscript j indicates the jth Monte Carlo trial. This study used $N_{\text{trials}} = 100$ Monte Carlo trials. Then, the MRMSE is the mean of the RMSE over time, giving

$$\mathrm{MRMSE} = \frac{1}{N} \sum_{k=1}^{N} \mathrm{RMSE}(kT_s), \tag{3.2}$$

where N is the total number of times at which the filtering was performed. An additional measure of accuracy was the number of trials in which the filter estimate diverged. A divergence is considered an absolute error in teh estimated SOC greater than 0.1 V or any failure in the filtering process, such as due to a non-invertible matrix or a non-positive definite covariance matrix. In addition, after a numerical failure for a filter in a trial, no attempt was made to keep filtering the system, and the remainder of the SOC values are assumed to be the worst case of zero. The speed was measured using the CPU time, which is the sum of the times used by the filter program on each of the CPU cores. The use of the CPU time results in less variability between runs compared to the clock time. In addition, the time taken to read and write the data is not counted.

The accuracy and speed were compared for sampling periods of $T_s = 30$, 150, and 300 seconds. Additionally, integration steps of $M = 1, 2, 4, \ldots, 256$ were used for each sampling period. The divergences of the filters for the different sampling periods is shown as a function of the number of integration steps in Tables 3.1, 3.3 and 3.5. The filtering times are shown in Tables 3.2, 3.4 and 3.6. The MRMSE for the filters are shown in Figures 3.1, 3.3 and 3.5. Detailed views of the MRMSE for large M are shown in Figures 3.2, 3.4 and 3.6.

Table 3.1: Number of divergences in 100 Monte Carlo runs for $T_s = 30$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	90	56	0	0	0	0	0	0	0
UKF	100	100	0	0	0	0	0	2	15
CKF	100	100	0	0	0	0	0	0	0
SLF	100	100	0	0	0	0	0	0	0

Table 3.2: Filtering time for 100 Monte Carlo runs for $T_s = 30$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	5.655	5.568	5.753	11.23	22.43	44.27	87.85	174.0	356.0
UKF	6.116	14.49	39.79	79.01	157.7	314.8	626.8	1237	2297
CKF	5.545	12.76	34.80	68.81	137.4	274.0	546.1	1091	2173
SLF	5.057	7.585	19.03	36.91	74.06	147.0	293.7	583.2	1153

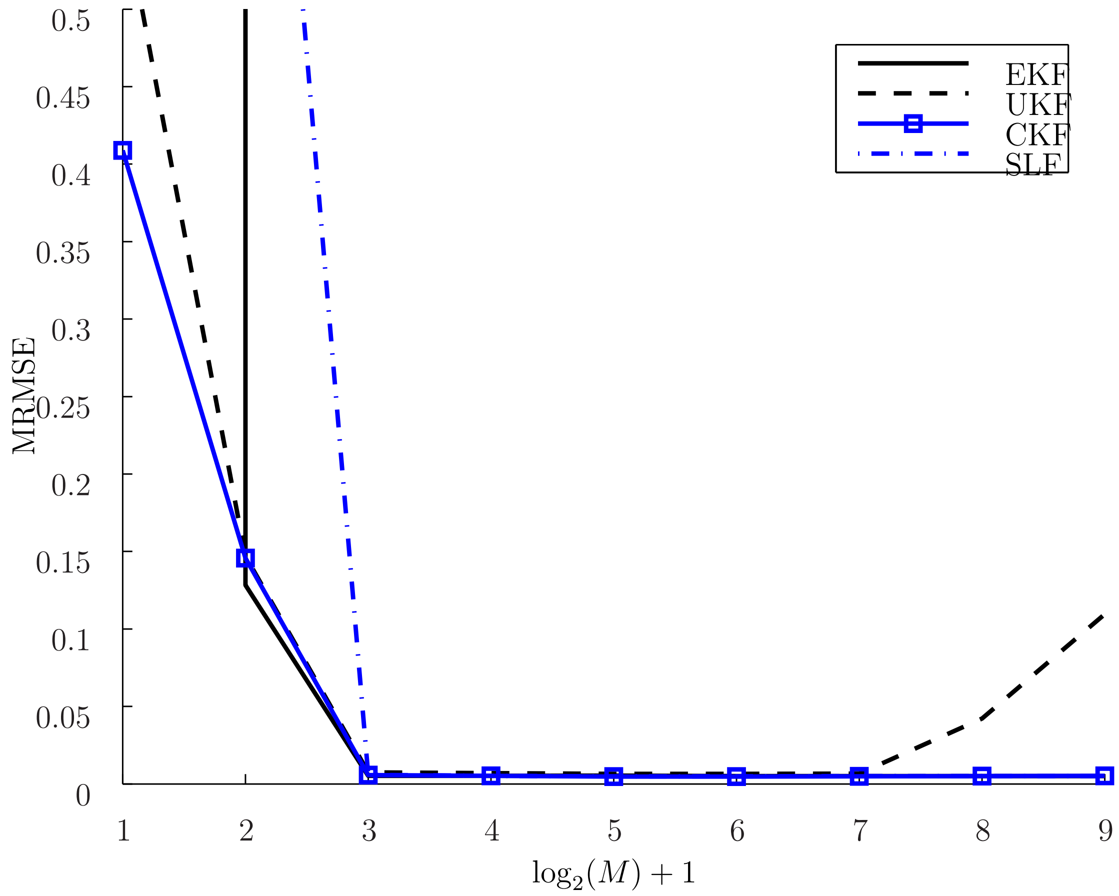

Figure 3.1: SOC error for $T_s = 30$ s as a function of number of integration steps M.

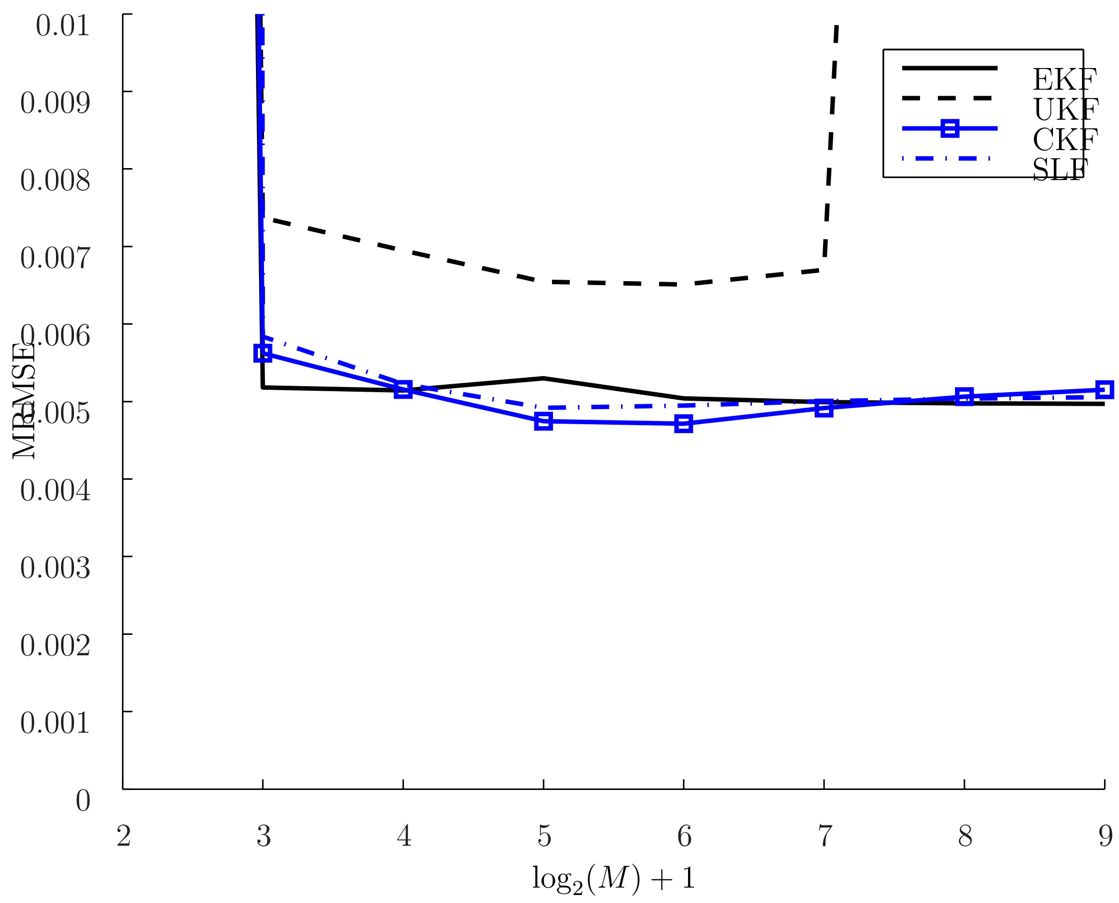

Figure 3.2: Detailed view of SOC error for $T_s = 30$ s as a function of number of integration steps M.

Table 3.3: Number of divergences in 100 Monte Carlo runs for $T_s = 150$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	100	100	98	77	0	0	0	0	0
UKF	100	100	100	100	0	0	0	0	0
CKF	100	100	100	100	0	0	0	0	0
SLF	100	100	100	0	0	0	0	0	0

Table 3.4: Filtering time for 100 Monte Carlo runs for $T_s = 150$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	0.8610	2.494	7.534	8.623	4.272	8.454	17.22	34.41	68.39
UKF	0.8559	1.744	3.616	11.45	31.48	62.96	125.4	251.8	502.8
CKF	0.0310	1.621	3.204	10.11	28.04	55.88	112.4	221.2	437.9
SLF	1.027	2.000	3.684	7.443	14.65	29.52	58.87	118.4	235.8

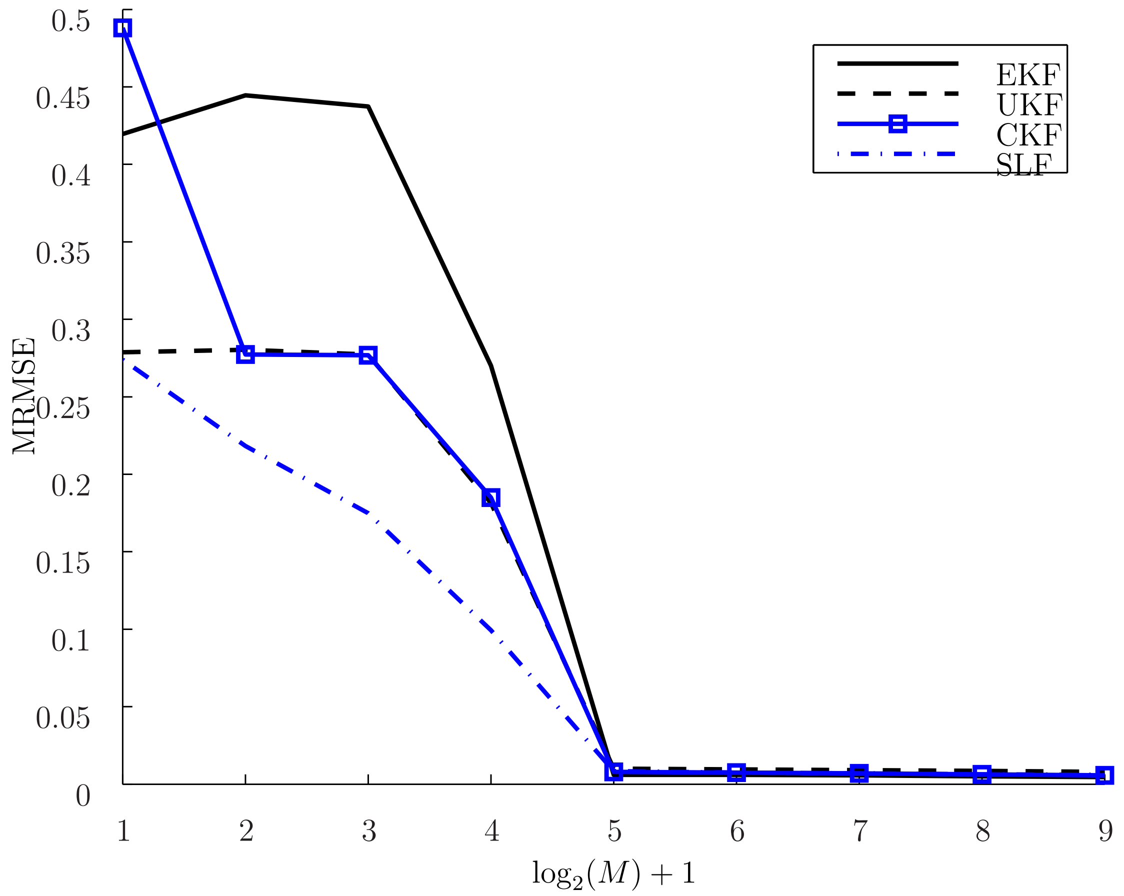

Figure 3.3: SOC error for $T_s = 150$ s as a function of number of integration steps M.

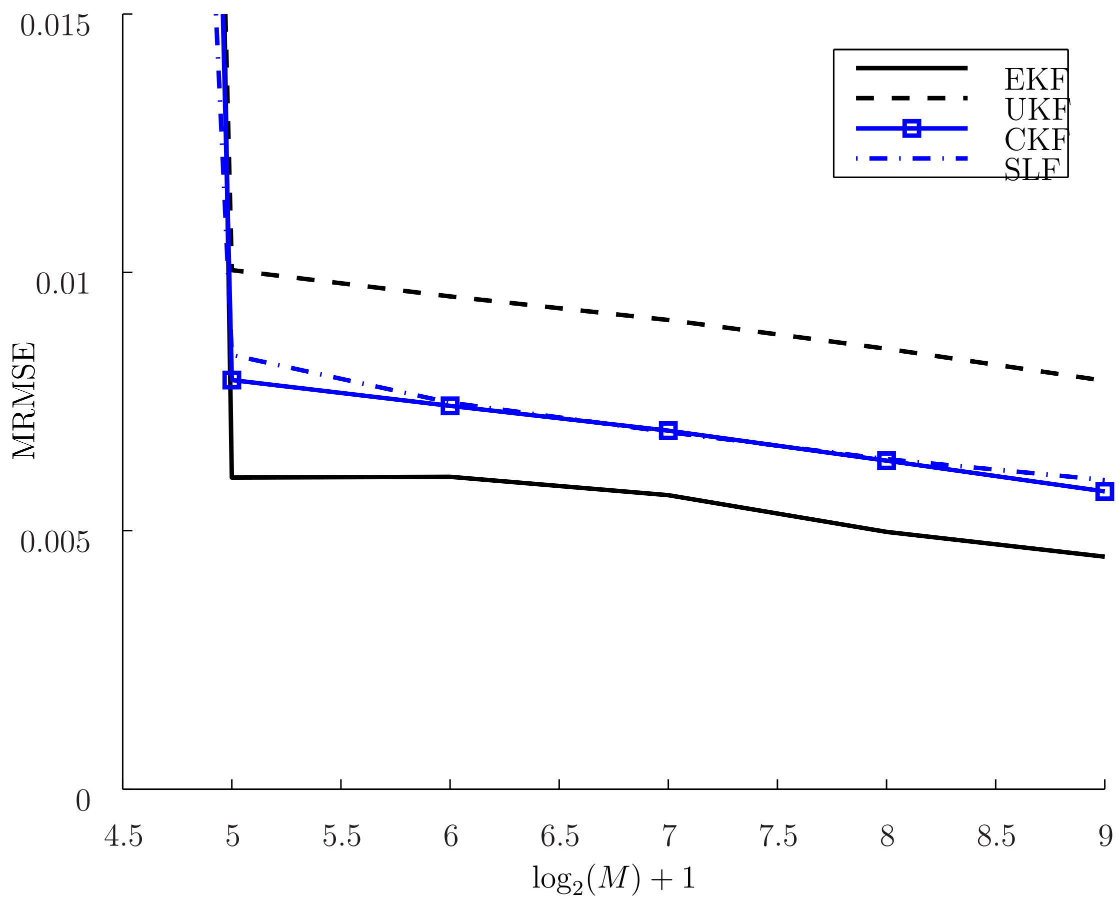

Figure 3.4: Detailed view of SOC error for $T_s = 150$ s as a function of number of integration steps M.

Table 3.5: Number of divergences in 100 Monte Carlo runs for $T_s = 300$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	100	100	100	100	68	0	0	0	0
UKF	100	100	100	100	100	0	0	0	0
CKF	100	100	100	100	0	0	0	0	0
SLF	100	100	100	100	0	0	0	0	0

Table 3.6: Filtering time for 100 Monte Carlo runs for $T_s = 300$ s as a function of number of integration steps M

Filter/M	1	2	4	8	16	32	64	128	256
EKF	0.2470	0.8910	3.572	7.913	8.174	4.451	8.755	17.62	34.38
UKF	0.0480	0.0480	1.543	3.477	9.372	31.64	63.17	125.9	252.0
CKF	0.0320	0.0640	1.535	3.061	10.28	28.27	55.80	109.7	217.9
SLF	0.4580	1.055	1.933	3.638	7.393	14.86	29.44	58.90	117.6

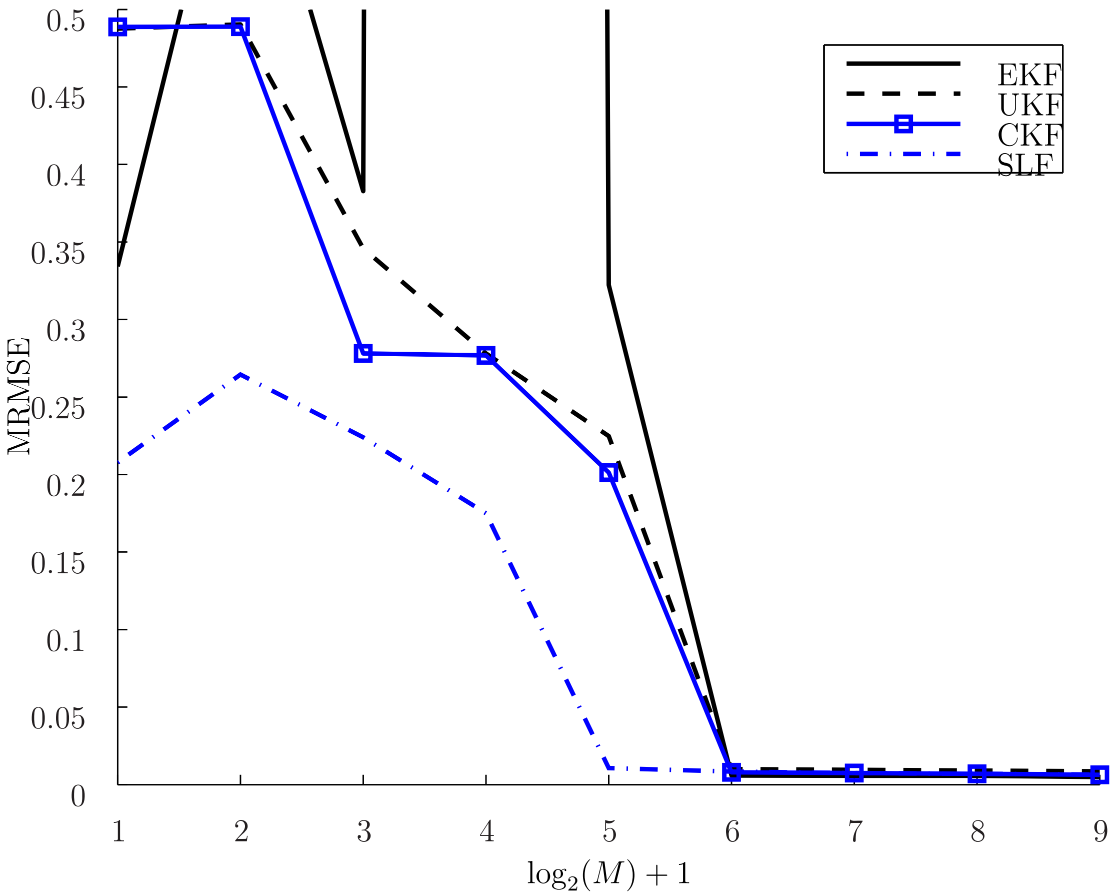

Figure 3.5: SOC error for $T_s = 300$ s as a function of number of integration steps M.

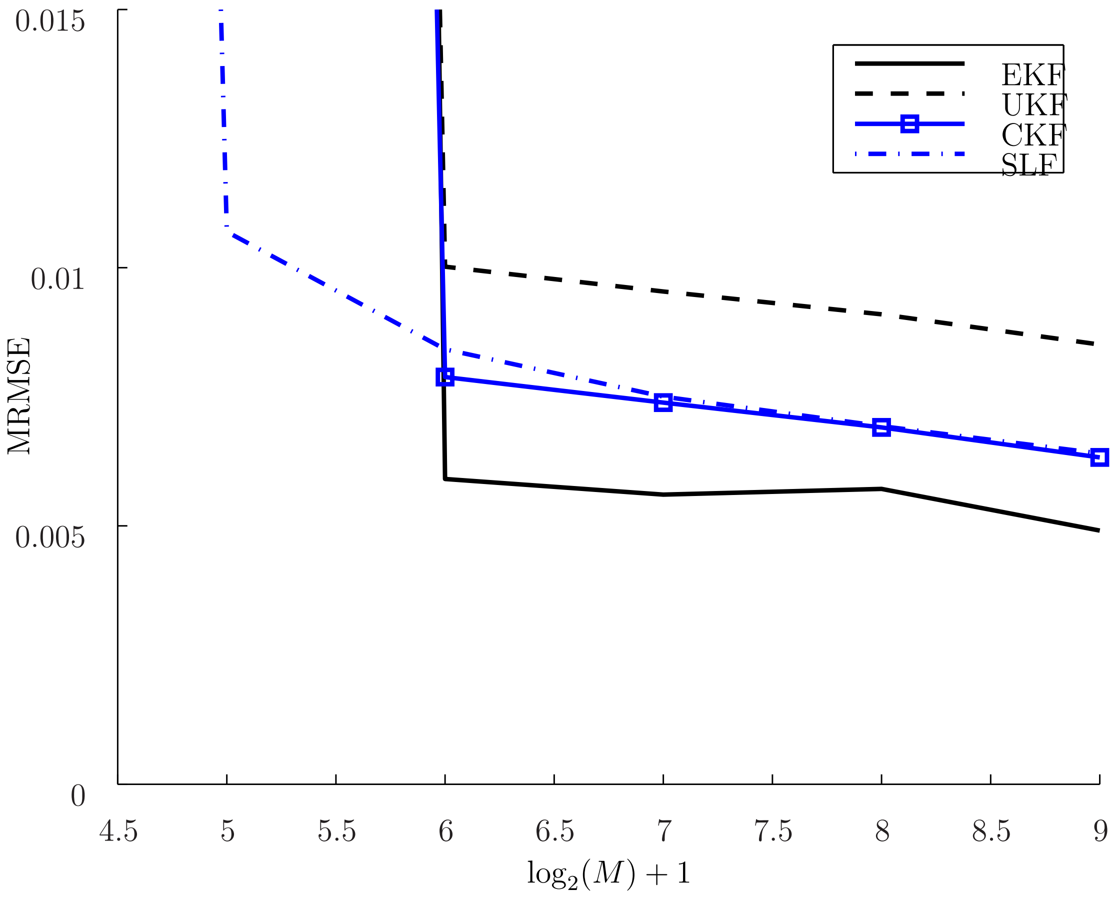

Figure 3.6: Detailed view of SOC error for $T_s = 300$ s as a function of number of integration steps M.

Chapter 4

Discussion

In terms of speed, for a given number of integration steps, the EKF is about 3 to 4 times faster than the SLF and 7 times faster than the UKF and CKF. Additionally, the time about doubles for each doubling in the number of integration steps. Note that the majority of the time is spend in performing the numerical integration, so the complexity of the discretization method directly affects the speed. The EKF has the simplest discretization so it is naturally the fastest. The SLF uses the same method as the EKF but evaluates it multiple times per integration step, so it is somewhat slower. The UKF and CKF use a complex IT-1.5 discretization method that uses the Jacobians and Hessians of the system, so it is the slowest. Evaluation of the discretization at multiple sigma points further slows down these filters. Note that they are very fast for low numbers of integration steps, which can be explained by the filters encountering numerical problems with the Cholesky factorization and skipping to the next trial.

In terms of the number of divergences, the EKF performs the best for $T_s = 30$ seconds with some trials with good accuracy for $M = 1$ and 2, followed by the CKF and SLF. The UKF performs the worst, with numerical problems for large M. The other three filters have good accuracy for $M \geq 4$. For $T_s = 150$ seconds, the SLF has the lowest number of divergences, with good accuracy for $M \geq 8$, while the others need $M \geq 16$. The EKF is the second best with a few good trials at $M = 4$ and 8. The UKF and CKF tie for last place. For $T_s = 300$ seconds, the CKF and SLF are the best with good accuracy for $M \geq 16$. The other two need $M \geq 32$, but EKF is slighly better than the UKF with less divergences at $M = 16$. Overall, the SLF resulted in the best performance in terms of the number of divergences as a function of the number of integration steps. It achieved zero divergences at a equal or smaller number of integration steps for each of the sampling periods. The EKF and CKF are close behind in performance. The EKF was able to get fewer than 100 divergences for $T_s = 30$ and 150, where the CKF had 100 divergences. However, at $T_s = 300$, the CKF achieves zero divergences at a smaller M value. The worst performing filter was the UKF which had the highest number of divergences. In addition, it had some divergences for $T_s = 30$ seconds for $M = 128$ and 256, probably caused by numerical problems related to the unscented transform.

In terms of the MRMSE, for $T_s = 30$ seconds, the EKF, CKf, and SLF are about equal in performance for $M \geq 4$, at which the divergences are zero. The MRMSE is flat in that range for those three filters. The best accuracy is achieved by the CKF at $M = 32$. The UKF has poor performance at this sample rate and diverges for $M \geq 128$. For $T_s = 150$ seconds, the SLF achieves the best performance for $M \leq 8$, and the EKF has the worst performance. The CKF and UKF are about equal over the range, with the UKF being significantly better at $M = 1$. Larger values of M shown increases in accuracy for all the filters, with the EKF achieving the best accuracy for $M \geq 16$. The CKF and SLF are about equal over the same range, while the UKF has the worst accuracy. For $T_s = 300$ seconds, the SLF achieves the best accuracy for $M \leq 16$, and the EKF has very poor performance. The CKF and

UKF are about equal, with the CKF having a small edge in accuracy. For higher values of $M \geq 32$, the EKF achieves the best accuracy. The CKF and SLF are about equal over the range, and the UKF has the worst accuracy.

It can be seen that the trend of accuracy as a function of the number of integration steps is similar for sample period of 150 and 300 seconds, where the doubling in the number of integration steps at which no divergences are encountered from $T_s = 150$ to 300 seconds is explained by the doubling of the sample period. The SLF has the best performance for a small number of integration steps, while the EKF has the best performance for greater than or equal to 16 and 32 integration steps for the sample periods of 150 and 300, respectively. For a sample period of 30 seconds, the EKF, CKF, and SLF perform equally well for greater than or equal to 4 integration steps, while the UKF encounters numerical problems that cause it to diverge for greater than 64 integration steps.

Overall, the EKF has the best accuracy and the best speed, for sufficiently large numbers of integration steps. The speed is expected since it is the least complex of the studied filters. In addition, its poor performance at small numbers of integration steps is due to the nonlinearity of the system. However, its greater accuracy for large numbers of integration steps is surprising since the battery system is strongly nonlinear, particularly for SOC near zero. This effect could be caused by a non-Gaussian posterior distribution of the noises, which would negatively impact the Gaussian assumption used by the other filters. The LMMSE Kalman filter does not make such an assumption to minimize the error covariance. In fact, Kalman showed that the linear estimate with non-Gaussian noise can only be improved by nonlinear estimation when at least third-order probability distribution functions are considered [29]. As this was not the case with the chosen filters, it makes sense that the EKF performs the best when a sufficiently large number of steps are used to minimize the error of the numerical integration.

Bibliography

[1] Z. Chen, "Bayesian filtering: from Kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[2] C. F. Chiasserini and R. R. Rao, "A model for battery pulsed discharge with recovery effect," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, vol. 2, 1999, pp. 636–639.

[3] M. R. Jongerden and B. R. H. M. Haverkort, "Battery modeling," Centre for Telematics and Information Technology University of Twente, Enschede, Tech. Rep. TR-CTIT-08-01, 2008.

[4] K. L. Man, K. Wan, T. O. Ting, C. Chen, T. Krilavičius, J. Chang, and S. H. Poon, "Towards a hybrid approach to soc estimation for a smart battery management system (BMS) and battery supported cyber-physical systems (CPS)," in *2012 2nd Baltic Congress on Future Internet Communications (BCFIC)*, Apr. 2012, pp. 113–116.

[5] C. C. O'Gorman, D. Ingersoll, R. G. Jungst, and T. L. Paez, "Artificial neural network simulation of battery performance," in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, vol. 5, Jan. 1998, pp. 115–121.

[6] G. Capizzi, F. Bonanno, and G. M. Tina, "Recurrent neural network-based modeling and simulation of lead-acid batteries charge-discharge," *IEEE Transactions on Energy Conversion*, vol. 26, no. 2, pp. 435–443, Jun. 2011.

[7] J. Wang, Q. Chen, and B. Cao, "Support vector machine based battery model for electric vehicles," *Energy Conversion and Management*, vol. 47, no. 7–8, pp. 858–864, 2006.

[8] W. X. Shen, C. C. Chan, E. W. C. Lo, and K. T. Chau, "Adaptive neuro-fuzzy modeling of battery residual capacity for electric vehicles," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 3, pp. 677–684, Jun. 2002.

[9] M. R. Jongerden and B. R. H. M. Haverkort, "Which battery model to use?" *IET Software*, vol. 3, no. 6, pp. 445–457, Dec. 2009.

[10] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell,” *Journal of the Electrochemical Society*, vol. 140, no. 6, pp. 1526–1533, 1993.

[11] T. F. Fuller, M. Doyle, and J. Newman, “Simulation and optimization of the dual lithium ion insertion cell,” *Journal of the Electrochemical Society*, vol. 141, no. 1, pp. 1–10, 1994.

[12] ——, “Relaxation phenomena in lithium-ion insertion cells,” *Journal of the Electrochemical Society*, vol. 141, no. 4, pp. 982–990, 1994.

[13] J. Newman, *Fortran programs for the simulation of electrochemical systems*, `http://www.cchem.berkeley.edu/jsngrp/fortran.html`, 1998.

[14] J. C. Bezdek, “On the relationship between neural networks, pattern recognition and intelligence,” *International Journal of Approximate Reasoning*, vol. 6, no. 2, pp. 85–107, 1992.

[15] ——, “What is computational intelligence?” In *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks, and C. J. Robinson, Eds., IEEE Press, 1994, pp. 1–12.

[16] D. Doerffel and S. A. Sharkh, “A critical review of using the peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries,” *Journal of Power Sources*, vol. 155, no. 2, pp. 395–400, 2006.

[17] J. F. Manwell and J. G. McGowan, “Lead acid battery storage model for hybrid energy system,” *Solar Energy*, vol. 50, no. 5, pp. 399–405, 1993.

[18] D. N. Rakhmatov and S. B. K. Vrudhula, “An analytical high-level battery model for use in energy management of portable electronic systems,” in *Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design*, ser. ICCAD ’01, San Jose, California: IEEE Press, 2001, pp. 488–493.

[19] C. F. Chiasserini and R. R. Rao, “Pulsed battery discharge in communication devices,” in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom ’99, 1999, pp. 88–95.

[20] ——, “Improving battery performance by using traffic shaping techniques,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1385–1394, 2001.

[21] ——, “Energy efficient battery management,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1235–1245, 2001.

[22] V. Rao, G. Singhal, A. Kumar, and N. Navet, “Battery model for embedded systems,” in *Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, ser. VLSID ’05, Washington, DC, USA: IEEE Computer Society, 2005, pp. 105–110.

[23] L. A. Geddes, "Historical evolution of circuit models for the electrode-electrolyte interface," *Annals of Biomedical Engineering*, vol. 25, no. 1, pp. 1–14, 1997.

[24] J. E. B. Randles, "Kinetics of rapid electrode reactions," *Discussions of the Faraday Society*, vol. 1, pp. 11–19, 1947.

[25] G. T. A. Kovacs, "Introduction to the theory, design, and modeling of thin-film microelectrodes for neural interfaces," in *Enabling Technologies for Cultured Neural Networks*, D. A. Stenger and T. M. McKenna, Eds., New York: Academic Press, 1994, ch. 7.

[26] S. C. Hageman, "Simple PSpice models let you simulate common battery types," *Electronic Design News*, vol. 38, pp. 117–129, 1993.

[27] ——, "Using PSpice to simulate the discharge behavior of common batteries," in *MicroSim Application Notes*, ver. 8.0, MicroSim Corporation, 1997, pp. 260–286.

[28] M. Chen and G. A. Rincón-Mora, "Accurate electrical battery model capable of predicting runtime and i-v performance," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 504–511, Jun. 2006.

[29] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME—Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[30] X.-R. Li and V. P. Jilkov, "A survey of maneuvering target tracking: approximation techniques for nonlinear filtering," *Proceedings of the 2004 SPIE Conference on Signal and Data Processing of Small Targets*, vol. 5428, pp. 537–550, 2004.

[31] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[32] A. Gelb, Ed., *Applied Optimal Estimation*. The MIT Press, 1974.

[33] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 1997, pp. 182–193.

[34] I. Arasaratnam, S. Haykin, and T. R. Hurd, "Cubature Kalman filtering for continuous-discrete systems: theory and simulations," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 4977–4993, 2010.

[35] T. Mazzoni, "Computational aspects of continuous-discrete extended Kalman-filtering," *Computational Statistics*, vol. 23, no. 4, pp. 519–539, 2007.

[36] S. Särkkä, "On unscented Kalman filtering for state estimation of continuous-time nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1631–1641, 2007.

[37] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000.

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

[39] S. Särkkä and A. Solin, "On continuous-discrete cubature Kalman filtering," in *Proceedings of SYSID*, 2012, pp. 1221–1226.

[40] B. Jia, M. Xin, and Y. Cheng, "High-degree cubature Kalman filter," *Automatica*, vol. 49, no. 2, pp. 510–518, 2013.

[41] A. H. Stroud, *Approximate Calculation of Multiple Integrals*. Edgewood Cliffs, NJ: Prentice-Hall, Inc., 1971.

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

[43] D. Linden, "Factors affecting battery performance," in *Handbook of Batteries*, D. Linden and T. B. Reddy, Eds., 3rd ed., McGraw-Hill, 2001, ch. 3.

[44] C. K. Boggs, A. D. Doak, and F. L. Walls, "Measurement of voltage noise in chemical batteries," in *Proceedings of the 49th IEEE International Frequency Control Symposium*, May 1995, pp. 367–373.

[45] F. G. Stremler, *Introduction to Communication Systems*, 2nd ed. Reading, MA: Addison-Wesley, 1982.

[46] J. D. Lambert, *Numerical Methods for Ordinary Differential Equations*. New York: John Wiley, 1991.

[47] L. Brugnano, F. Mazzia, and D. Trigiante, "Fifty years of stiffness," in *Recent Advances in Computational and Applied Mathematics*, T. E. Simos, Ed., Springer Netherlands, pp. 1–21.

[48] A. Jossen, "Fundamentals of battery dynamics," *Journal of Power Sources*, vol. 154, no. 2, pp. 530–538, 2006.