SIMULTANEOUS CELL STATE ESTIMATION VIA DENSE ADAPTIVE EXTENDED KALMAN FILTER

by

LUKE NUCULAJ

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

2024

Oakland University
Rochester, Michigan

Thesis Advisory Committee:

Jun Chen, Ph.D., Chair
Darrin Hanna, Ph.D.
Manohar Das, Ph.D.

To my father, my mother, my siblings Isabella, Alek,

Leonardo, and my beloved.

Without your constant love and support, this would have

never been possible. This is for you.

ACKNOWLEDGMENTS

The author wishes to express his appreciation for extensive mentorship provided by Jun Chen, Ph.D., under whom I have grown significantly as an engineer in all aspects, but also as an eager student of the sciences. Thanks are also due to committee members Darrin Hanna, Ph.D. and Manohar Das, Ph.D. for their contributions to this dissertation – their insights have proven highly valuable in its development. Likewise, acknowledgment should be extended to peers Ali Irshayyid, Zhaodong Zhou, and Christopher Rother.

Luke Nuculaj

ABSTRACT

SIMULTANEOUS CELL STATE ESTIMATION VIA DENSE ADAPTIVE EXTENDED KALMAN FILTER

by

LUKE NUCULAJ

Adviser: Jun Chen, Ph.D.

This work addresses the computational intractability apropos of extended Kalman filters (EKF) in the context of battery cell state estimation under limited voltage measurement. A novel, compact variation of the Kalman filter – namely, the “dense extended Kalman filter" (DEKF) – is proposed, which leverages unique information about each of the cell’s inherent physical properties and net currents at each time step to compress sparsely-populated covariance matrices and state vectors into a dense form, one which does not vary with the number of cells in the pack. The computation saving in terms of floating point operations (FLOPs) reduction is analytically compared and illustrated through simulation. More specifically, the DEKF offers significant resource savings while maintaining estimation accuracy, reducing the estimation algorithm’s time complexity from $\mathscr{O}(N^3)$ to $\mathscr{O}(N)$, where N is the number of cells in a serial-connected string. Furthermore, a special case where all serial-connected cells share the same discharge current, i.e., no balancing or leakage, is also studied and demonstrated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AEKF Adaptive Extended Kalman Filter

DAEKF Dense Adaptive Extended Kalman Filter

DEKF Dense Extended Kalman Filter

EKF Extended Kalman Filter

EV Electric Vehicle

FLOP Floating-Point Operation

IAEKF Intelligent Adaptive Extended Kalman Filter

MPC Model Predictive Control

OCV Open-Circuit Voltage

RFF Relative Fitness Factor

RMSE Root Mean Squared Error

SOC State of Charge

CHAPTER ONE

INTRODUCTION

With the growing popularity of electric vehicles (EVs) in recent years, their superiority to their gasoline-powered counterparts in areas of reduced carbon emissions and cost efficiency have rightfully catapulted EVs to the frontier of today's cutting-edge, infrastructural technology [1, 2]. At the heart of EVs lay hundreds of battery cells – typically of the lithium-ion (Li-Ion) variety [3–5]. Due to the inevitability of manufacturing variations, battery cells exhibit voltage and state-of-charge (SOC) imbalances with one another which – in turn – curtail both battery life and performance, ultimately reducing an electric vehicle's range [6–9]. To combat this, nondissipative cell balancing techniques are frequently employed in conjunction with advanced control techniques – model predictive control (MPC) being among the most commonly explored and appealing control techniques by virtue of its ability to account for system constraints [6, 10, 11]. However, the dual problem of a battery cell's harshly nonlinear dynamics and its internally complex, electrochemical processes renders direct measurement of SOC a challenging task, if not an impossible one [12–15]. This fact necessitates a reliable means of SOC estimation, as an ill-informed cell balancing controller is prone to overcharging/discharging multiple cells at a time, thereby exacerbating the degradation of the pack.

Despite their computational lightness, traditional Coulomb counting methods for SOC estimation suffer a great deal from accumulated, current integration error [16, 17]. On the other hand, the extended Kalman filter (EKF) [18–22], which unionizes a pre-conceived mathematical model of the cell's internal, nonlinear dynamics and terminal voltage measurements to accurately estimate cell SOC while mitigating the drift that plagues the Coulomb counting method, has been widely researched in literature [17, 23–27]. For example, reference [24] explores the adaptive extended Kalman filter (AEKF), which employs a covariance matching approach for quick yet reliable online estimation, yielding a maximum SOC estimation error of less than 2%. Reference [26] builds upon the AEKF, introducing an intelligent AEKF (IAEKF), which monitors the changes in the fixed-length error innovation sequence's (EIS) distribution, and updates the innovation covariance matrices accordingly. In comparison the AEKF, this method sees the decrease of the estimator's root mean squared error (RMSE) by 43.34%, while the computational overhead only increases by 4.59%.

While these improved estimators show promising results, as soon as the task becomes simultaneous (and with limited measurement [28]), multi-cell SOC estimation instead of single-cell, variants of the EKF suffer immensely from computational latency and hefty memory requirements [29–32]. For purposes of multi-cell SOC estimation, this fact renders traditional Kalman filtering over a large number of cells wholly impractical for embedded deployment, wherein computing power and memory resources are tightly constrained [29]. While there exists some utility in heuristic, data-driven methods of pack SOC estimation [33, 34], their "black-box" nature is not desirable in the context of deterministic estimation methods. While the research on deterministic pack SOC estimation is scarce at best, [35] exploits the local observability of a nominal battery model to enable interval estimation of pack SOC, scalable by virtue of the interval observer's number of states being independent of the number of cells. Although this method realizes cell heterogeneity in the form of different SOC initialization, electrical parameters, and unevenly distributed currents, the assignment of an identical OCV-SOC (open circuit voltage) relationship to each cell is restrictive, in spite of its favorable CPU time with respect to cell number.

This paper further explores the extended Kalman filter as a viable means of estimation, but rather than dealing explicitly with each cell's system dynamics in the form of sparse state vectors/matrices, a novel, dense extended Kalman filter (DEKF) is proposed. This method of cell SOC estimation re-imagines the entire pack as a single "average" cell and perform state estimation over the said average cell, whose dimensions are constant regardless of the number of cells in the string. Furthermore, a relative fitness factor (RFF) is uniquely defined for each cell based on how much its SOC changes with respect to the average cell – the RFF is largely a function of cell parameters. A cell's RFF is defined in such a way that when multiplied by the change in the average state, the change in said cell's state can be recovered. As will be shown later, this can be exploited to recover the changes in every single cell's state from the average state, hence reducing the state estimation problem for each single cell to that of the average cell whose size is constant. Similar to [35], the state vector's size is invariable with respect to the number of cells, but the DEKF's covariance matrices are also a fixed size. With this in mind, it was found by way of numerical simulation that for the 100-cell problem, an adaptive rendition of the DEKF saved over 16 million FLOPs of computation in comparison to the sparse extended Kalman filter while exhibiting nearly identical performance. The adaptive DEKF's scalability in terms of estimation performance is closely examined for various cell numbers, and is shown to grow more accurate with a larger cell number. This is because the measurement – and consequently, the measurement noise – are being scaled down by a greater amount for a large cell number N, which sees the adaptive DEKF relying on the measurements more to balance the model predictions. See Section 6.2 for more details.

The rest of the paper is organized as follows. Section II formulates the problem of cell SOC estimation in a serial-connected string, while Section III presents the traditional sparse EKF for cell SOC estimation. Section IV introduces the concept of relative fitness factors, forming the basis for the DEKF. Section V derives the DEKF and establishes its equivalence to sparse formulation, while section VI provides the main simulation results. Section VII discusses a special case with homogeneous cell current and Section VIII concludes the paper.

CHAPTER TWO

CELL STATE ESTIMATION PROBLEM

2.1 Battery Model

Consider a serial-connected battery with N cells, as shown in Fig. 2.1. The highly nonlinear chemical processes that occur within battery cells are difficult to precisely model. Instead, a first-order equivalent circuit model (ECM) is adopted as a suitable representation of a Li-Ion battery's system dynamics [27, 36–38]. See Fig. 2.2, where the open-circuit voltage (V_{oc}), open-circuit resistance (R_o), terminal voltage (y), total cell current (u), relaxation resistance (R_p) and relaxation capacitance (C_p) are all model parameters. The cell dynamics are specified by

$$\begin{aligned}
\dot{s}^i &= -\eta^i \frac{u^i}{3600C^i} \\
\dot{V}^i &= -\frac{V^i}{R_p^i C_p^i} + \frac{u^i}{C_p^i} \\
y^i &= V_{oc}^i - V^i - u^i R_o^i,
\end{aligned}$$

where the superscript i denotes the ith cell, s^i is the ith cell's state of charge (SOC), V^i is the ith cell's relaxation voltage, η^i is the Coulombic efficiency, C^i is the cell capacity with unit of Amp-hours (the latter two being constant with respect to cell states). Additionally, u_k^i is the ith cell's total current at time step k, and is the sum of an applied balancing current β_k^i and the battery pack current u_k. Finally, the convention that $u_k^i > 0$ signifies discharging and $u_k^i < 0$ charging is used in this work.

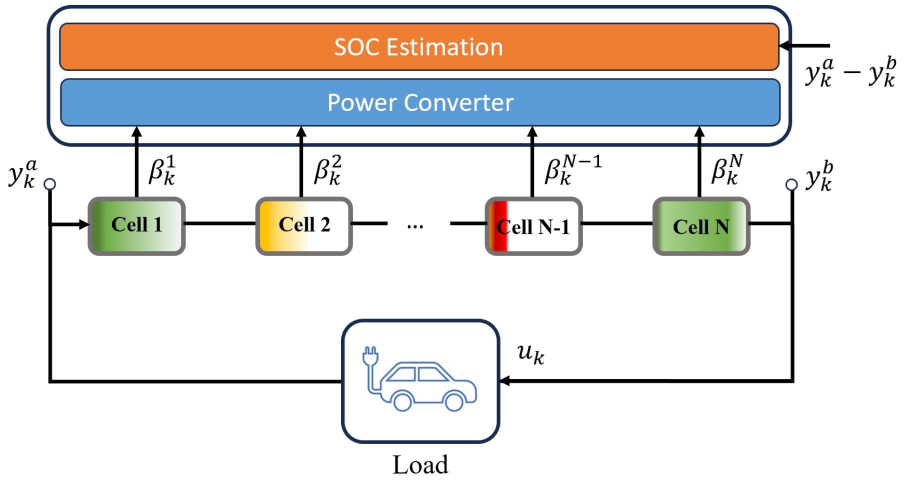

Figure 2.1: Structure of serial-connected battery cells with balancing currents and pack terminal voltage measurement $(y_k^a - y_k^b)$.

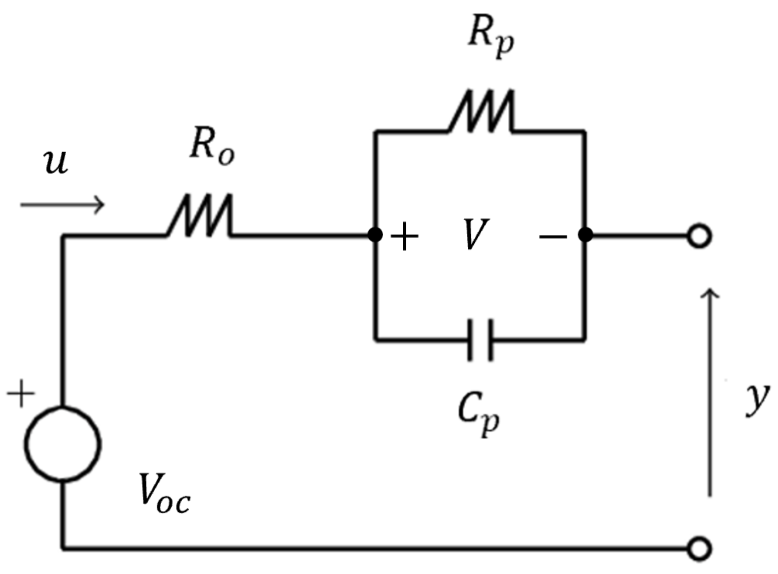

Figure 2.2: Equivalent circuit model of a battery cell.

Forward Euler method can be used to discretize the above cell dynamics with a sampling time T_s, as follows:

$$s^i_{k+1} = s^i_k - \eta^i \frac{T_s}{3600C^i} u^i_k \tag{2.1a}$$

$$V^i_{k+1} = \left(1 - \frac{T_s}{R^i_p C^i_p}\right) V^i_k + \frac{T_s}{C^i_p} u^i_k \tag{2.1b}$$

$$y^i_k = V^i_{oc,k} - V^i_k - u^i_k R^i_o. \tag{2.1c}$$

Denoting $x^i := \begin{bmatrix} s^i & V^i \end{bmatrix}^T$, the linear state equation for a single cell can be compactly represented as

$$x^i_{k+1} = A^i x^i_k + B^i u^i_k, \tag{2.2}$$

where

$$A^i = \begin{bmatrix} 1 & 0 \\ 0 & 1 - \frac{T_s}{R^i_p C^i_p} \end{bmatrix} \quad B^i = \begin{bmatrix} -\eta^i \frac{T_s}{3600C^i} \\ \frac{T_s}{C^i_p} \end{bmatrix}. \tag{2.3}$$

Note that it is typical of the ECM's electrical parameters to vary as a function of SOC [14, 24, 27], making (2.2) nonlinear. However, to simplify the notation, in this work, we consider R_o, R_p, and C_p constant values that do not vary as a function of SOC, making the state equation (2.2) linear. Moreover, the measurement function (2.1c) is still nonlinear, since the open circuit voltage V_{oc} is nonlinear as a function of SOC – typically stored in a lookup table [39].

Consider the collection of state update equations across all N cells

$$X_{k+1} := \begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \\ \vdots \\ x^N_{k+1} \end{bmatrix} = \begin{bmatrix} A^1 x^1_k + B^1 u^1_k \\ A^2 x^2_k + B^2 u^2_k \\ \vdots \\ A^N x^N_k + B^N u^N_k \end{bmatrix},$$

and define the sparse state vector as $X_k = \begin{bmatrix} x_k^1 & x_k^2 & \dots & x_k^N \end{bmatrix}^T$, which is the state vector for the entire battery pack at time step k. This allows for the state dynamics across the entire pack to be represented as

$$X_{k+1} = AX_k + BU_k, \tag{2.4}$$

where

$$A = \begin{bmatrix} A^1 & 0 & \dots & 0 \\ 0 & A^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A^N \end{bmatrix} \tag{2.5}$$

$$B = \begin{bmatrix} B^1 & 0 & \dots & 0 \\ 0 & B^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B^N \end{bmatrix}, \tag{2.6}$$

are block-diagonal, and cell current matrix U_k is defined as $\begin{bmatrix} u_k^1 & u_k^2 & \dots & u_k^N \end{bmatrix}^T$. The measurement function, in the case of pack-level dynamics, is a scalar quantity representative of the pack's terminal voltage defined as

$$y_k = \sum_{i=1}^{N} \left(V_{oc,k}^i - V_k^i - u_k^i R_o^i \right) := h(X_k, U_k). \tag{2.7}$$

Example 1. *Consider a battery with $N = 5$ serial-connected cells, with cell parameters listed in Table 2.1 and initial states listed in Table 2.2. Moreover, $T_s = 10$ and all cell currents are equal to $4.6A$. Fig. 2.3 plots the SOC, relaxation voltage, and terminal voltage for each cell. As can be seen, due to the heterogeneous cell parameters (see Table 2.1), the cells' SOC significantly differ from each other, making cell level state estimation a challenging task, especially under limited sensor capability.*

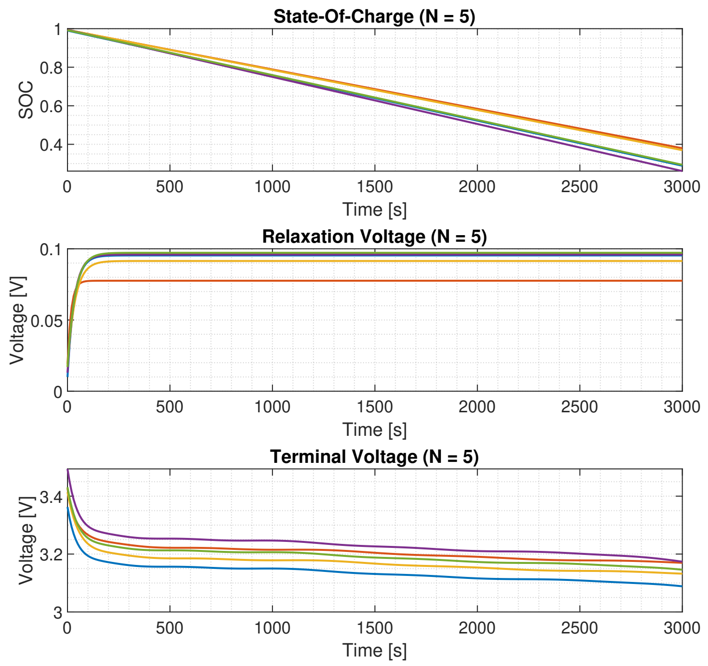

Figure 2.3: Plots of SOC, relaxation voltage, and terminal voltage for five cells.

Table 2.1: Nominal Circuit Parameters

Parameter	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
C	4.293	5.249	4.717	4.201	4.941
η	0.785	0.839	0.768	0.803	0.900
$R_p(\cdot 10^{-2})$	2.072	1.686	1.987	2.086	2.113
$C_p(\cdot 10^{3})$	1.874	1.373	2.148	1.870	2.004
$R_o(\cdot 10^{-2})$	1.718	1.565	1.292	1.344	1.184

Table 2.2: Initial States

State	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
s_0	0.990	0.993	0.994	0.994	0.992
V_0	0.010	0.019	0.017	0.013	0.017

2.2 Problem Formulation

Given the battery dynamic model (2.4), the primary objective of this paper is to find a computationally efficient state estimation algorithm to estimate X_k, with only one voltage sensor to measure the pack terminal voltage. Formally, the problem being addressed in this paper is described below.

Problem 1. *Given battery dynamic model (2.4) and output equation (2.7), find a computationally efficient algorithm for estimating X_k based on U_k and y_k.*

Remark 1. *Note that Problem 1 restricts the number of voltage sensors to only 1, i.e., only the pack terminal voltage is measured. This setting is similar to [35], which also assumes only the pack terminal voltage measurement is available. However, our work is different from [35] in that only interval estimation is performed in [35], while the state for all cells is estimated in our work. Note also that, the assumption that only terminal voltage is measured can be beneficial in reducing manufacturing cost while at the same time can be restrictive. In the future, we will also consider the scenario that multiple cell terminal voltages are also measured, and the corresponding optimal sensor configuration problem.*

CHAPTER THREE

SPARSE EXTENDED KALMAN FILTER

To solve Problem 1, the extended Kalman filter (EKF) can be applied, due to the nonlinearity of the battery model (particularly the output equation). This section describes a straightforward application of EKF – termed as *sparse EKF* for the remainder of this paper – for estimating over N serial-connected battery cells, which has a complexity of $O(N^3)$ since the sizes of the covariance matrices and vectors grow proportionally to N. We will later show a computationally efficient variant of the EKF to solve Problem 1 with a complexity of $O(N)$.

3.1 Sparse Prediction Model and Time Update

Because of the inherent deviations of real-life systems from mathematical models, (2.4) and (2.7) take on the form

$$X_{k+1} = AX_k + BU_k + W_k$$
$$y_k = h(X_k, U_k) + v_k,$$

where $W_k := \begin{bmatrix} w_k^1 & w_k^2 & \dots & w_k^N \end{bmatrix}^T$, w_k^i $(\sim \mathcal{N}(0, Q_k^i))$ is the process noise of the the ith cell, and v_k $(\sim \mathcal{N}(0, R_k))$ is the measurement noise, the latter two satisfying zero-mean Gaussian distributions with covariances Q_k^i and R_k. Retaining (2.5) and (2.6), the complete time-update for the sparse EKF is the following:

$$\hat{X}_{k+1}^- = A\hat{X}_k^+ + BU_k \tag{3.1a}$$
$$P_{k+1}^- = AP_k^+A^T + Q_k, \tag{3.1b}$$

where (3.1a) is the state update and (3.1b) is the covariance update. The initial sparse process covariance P_0^+ and sparse process noise covariance Q_k are block-diagonal matrices defined as

$$P_0^+ = \begin{bmatrix} P_0^{1,+} & 0 & \dots & 0 \\ 0 & P_0^{2,+} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_0^{N,+} \end{bmatrix} \tag{3.2}$$

$$Q_k = \begin{bmatrix} Q_k^1 & 0 & \dots & 0 \\ 0 & Q_k^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q_k^N \end{bmatrix}, \tag{3.3}$$

where $P_0^{i,+}$ and Q_k^i are the initial process covariance and process noise covariance at time step k, respectively, for the ith cell. Note that (3.1a) and (3.1b) rely on real-time computations over matrices (3.2) and (3.3), which grow quadratically with N. Finally, the predicted output, which is evaluated over the sparse EKF's prediction $\hat{X}_{k+1}^-$, can be computed as

$$\hat{y} = h\left(\hat{X}_{k+1}^-, U_k\right), \tag{3.4}$$

where h is defined in (2.7).

3.2 Sparse Kalman Gain and Measurement Update

Recall that the open circuit voltage V_{oc} is a nonlinear function of the SOC, making h in (2.7) nonlinear. Therefore, to compute the Kalman gain, one needs to first linearize h

as follows:

$$
\begin{aligned}
H_{k+1} &= \partial_X h(X,U)\Big|_{(\hat{X}^-_{k+1},U_k)} \\
&= \partial_X \left[\sum_{i=1}^{N} \Big(V^i_{oc}(X(i,1)) \right. \\
&\qquad\qquad \left. -R^i_o U(i) - X(i,2)\Big)\right]\Bigg|_{(\hat{X}^-_{k+1}.U_k)} .
\end{aligned}
\tag{3.5}
$$

Here, $\hat{X}^-_{k+1}(i)$ is the ith cell's predicted state vector at time step $k+1$, $U_k(i)$ is equivalent to u^i_k, and $X(i,1)$ and $X(i,2)$ are the SOC and V of the ith cell, respectively. Evaluating the Jacobian in (3.5), we have

$$
H_{k+1} = \left[\frac{\partial V^1_{oc}}{\partial X(i,1)} \quad -1 \quad \cdots \quad \frac{\partial V^N_{oc}}{\partial x^N(1)} \quad -1\right]\Bigg|_{(\hat{X}^-_{k+1},U_k)} , \tag{3.6}
$$

where $X(i,1)$ is the ith cell's SOC. The resulting sparse Kalman gain matrix and the measurement update can be calculated:

$$
K_{k+1} = \frac{P^-_{k+1}H^T_{k+1}}{H_{k+1}P^-_{k+1}H^T_{k+1} + R_k}. \tag{3.7a}
$$

$$
\hat{X}^+_{k+1} = \hat{X}^-_{k+1} + K_{k+1}(y_{k+1} - h(\hat{X}^-_{k+1}, U_k)) \tag{3.7b}
$$

$$
P^+_{k+1} = (I - K_{k+1}H_{k+1})P^-_{k+1}. \tag{3.7c}
$$

Recall from the previous section that R_k is the measurement noise covariance such that $v_k \sim \mathcal{N}(0, R_k)$.

Remark 2. *While the sparse EKF provides a passable framework for cell state estimation, a notable drawback is the increasing size of matrices used in calculations as N increases, making it computationally heavy. In fact, the complexity of such a naive EKF approach requires $O(N^3)$ complexity each time step. More particularly, the time update requires $O(N^2)$ complexity, even if the sparsity of A and B matrices are explicitly* *exploited, and the measurement update requires* $O(N^3)$ *complexity. The total complexity is* $16N^3 + 30N^2 + (4P + 18)N + 6M + 1$*, where M is the size of a moving window for adaptive parameter tuning, and P is related to the resolution of the OCV-SOC curve. See Sections 5.3 and 6.2 for more details.*

Such a high complexity makes the sparse EKF approach not suitable for real-time implementation, especially in embedded environments. To address these concerns, in the following sections, a “dense extended Kalman filter” (DEKF) is developed, whose complexity is linear with respect to N, making it suitable for real-time implementation. We start by introducing a key element for the proposed DEKF – namely, the relative fitness factors – in the next section.

CHAPTER FOUR

RELATIVE FITNESS FACTORS

By introducing the concept of “relative fitness factors" accompanied with complete examples, this section sets out to form the basis for the DEKF, a novel, multi-cell SOC estimation technique that addresses the unmanageable resource requirements of the sparse EKF.

4.1 Derivation of Individual RFF

The driving paradigm of the dense EKF is that estimates are made about average state vectors and dense covariance matrices that are invariable in size for all N – it is unique information about each cell’s system dynamics that provide insight to how every SOC in the pack changes with respect to the “average" cell over time. In order to determine this change, begin by averaging (2.1a) and (2.1b) over all N cells to obtain

$$\frac{1}{N}\sum_{i=1}^{N} s_{k+1}^{i} = \frac{1}{N}\sum_{i=1}^{N} s_{k}^{i} - \frac{1}{N}\sum_{i=1}^{N} \frac{\eta^{i} T_s}{3600C^{i}} u_k^{i} \tag{4.1a}$$

$$\frac{1}{N}\sum_{i=1}^{N} V_{k+1}^{i} = \frac{1}{N}\sum_{i=1}^{N} \left(1-\frac{T_s}{\tau_p^{i}}\right) V_k^{i} + \frac{1}{N}\sum_{i=1}^{N} \frac{T_s}{C_p^{i}} u_k^{i}, \tag{4.1b}$$

where time constant $\tau_p^i = R_p^i C_p^i$. Denote $s_{\mu,k} := \frac{1}{N}\sum_{i=1}^{N} s_k^i$ and $V_{\mu,k} := \frac{1}{N}\sum_{i=1}^{N} V_k^i$ as the average SOC at time step k and average V at time step k, respectively. From (4.1), the changes in $s_{\mu,k}$ and $V_{\mu,k}$ are

$$\Delta s_{\mu,k} = s_{\mu,k+1} - s_{\mu,k} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\eta^{i} T_s}{3600C^{i}} u_k^{i} \tag{4.2a}$$

$$\Delta V_{\mu,k} = V_{\mu,k+1} - V_{\mu,k} = \frac{1}{N}\sum_{i=1}^{N} \left(\frac{T_s}{C_p^{i}} u_k^{i} - \frac{T_s}{\tau_p^{i}} V_k^{i}\right). \tag{4.2b}$$

Without loss of generality, we will now step through how to quantify the degree to which s_k^i changes with respect to $s_{\mu,k}$. Retrieving the control input term from (2.1a) yields

$$\gamma_{s,k}^i = \frac{\Delta s_k^i}{\Delta s_{\mu,k}} = \frac{-\frac{\eta^i T_s}{3600C^i} u_k^i}{-\frac{1}{N}\sum_{i=1}^N \frac{\eta^i T_s}{3600C^i} u_k^i} = \frac{\frac{\eta^i}{C^i} u_k^i}{\frac{1}{N}\sum_{i=1}^N \frac{\eta^i}{C^i} u_k^i}, \tag{4.3}$$

where $\gamma_{s,k}^i$ is the ith cell's SOC "relative fitness factor" (RFF) at time step k. From this methodology, $\gamma_{V,k}^i$ naturally follows

$$\gamma_{V,k}^i = \frac{\frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i}}{\frac{1}{N}\sum_{i=1}^N \left(\frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i}\right)}. \tag{4.4}$$

At any given time step, each cell has a set of two RFFs – one for each of its states. A cell's RFFs signify the degree to which each of its states s_k^i and V_k^i evolve with respect to average states $s_{\mu,k}$ and $V_{\mu,k}$. In other words, we can recover the change in states as

$$s_{k+1}^i = s_k^i + \gamma_{s,k}^i \Delta s_{\mu,k} \tag{4.5a}$$

$$V_{k+1}^i = V_k^i + \gamma_{V,k}^i \Delta V_{\mu,k}. \tag{4.5b}$$

The following theorem establishes the utility of the RFFs in computing each cell's change in state by proving the equivalence of (4.5) to the existing state dynamic equations.

Theorem 1. *The SOC update equation (4.5a) is equivalent to (2.1a), and the relaxation voltage update equation (4.5b) is equivalent to (2.1b).*

Proof. First we prove equivalence for the SOC update equation. Substituting (4.2a) and (4.3) into (4.5a),

$$\begin{aligned} s_{k+1}^i &= s_k^i + \left(\frac{-\frac{\eta^i T_s}{3600C^i} u_k^i}{-\frac{1}{N}\sum_{i=1}^N \frac{\eta^i T_s}{3600C^i} u_k^i}\right)\left(-\frac{1}{N}\sum_{i=1}^N \frac{\eta^i T_s}{3600C^i} u_k^i\right) \\ &= s_k^i - \frac{\eta^i T_s}{3600C^i} u_k^i, \end{aligned}$$

Table 4.1: Cell Currents

u_k^1	u_k^2	u_k^3	u_k^4	u_k^5
1.6	3.6	4.6	5.6	7.6

which reproduces (2.1a). As for the relaxation voltage update equation, a similar substitution can be made by substituting (4.2b) and (4.4) into (4.5b), where we get

$$\begin{aligned} V_{k+1}^i &= V_k^i + \frac{\frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i}}{\frac{1}{N}\sum_{i=1}^N \frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i}} \frac{1}{N} \sum_{i=1}^N \frac{T_s}{C_p^i} u_k^i - \frac{T_s}{\tau_p^i} V_k^i \\ &= V_k^i + \frac{T_s}{C_p^i} u_k^i - \frac{T_s}{\tau_p^i} V_k^i \\ &= \left(1 - \frac{T_s}{\tau_p^i}\right) V_k^i + \frac{T_s}{C_p^i} u_k^i, \end{aligned}$$

which reproduces (2.1b). This completes the proof. □

Theorem 1 established that the RFF can be used to quantify how the ith state vector x_k^i changes with respect to the state vector $x_{\mu,k}$ of an "averaging" model, defined as $x_{\mu,k} := \begin{bmatrix} s_{\mu,k} & V_{\mu,k} \end{bmatrix}^T$. To express this mathematically, the Jacobian $\Gamma_k^i = \partial x_k^i / \partial x_{\mu,k}$ can be defined as

$$\Gamma_k^i = \frac{\partial x_k^i}{\partial x_{\mu,k}} = \begin{bmatrix} \frac{\partial x_k^i(1)}{\partial x_{\mu,k}(1)} & \frac{\partial x_k^i(1)}{\partial x_{\mu,k}(2)} \\ \frac{\partial x_k^i(2)}{\partial x_{\mu,k}(1)} & \frac{\partial x_k^i(2)}{\partial x_{\mu,k}(2)} \end{bmatrix} = \begin{bmatrix} \gamma_{s,k}^i & 0 \\ 0 & \gamma_{V_k}^i \end{bmatrix}. \tag{4.6}$$

Then, (4.5) can be compactly represented as

$$x_{k+1}^i = x_k^i + \Gamma_k^i \Delta x_{\mu,k}. \tag{4.7}$$

Example 2. *Consider the five-cell serial-connected battery as discussed in Example 1. Consider $T_s = 0.1$, and the initial cell currents as they appear in Table 4.1. The*

Table 4.2: Relative Fitness Factors

γ^1	γ^2	γ^3	γ^4	γ^5
0.37430	0.70661	0.91970	1.31445	1.69994

denominator term in (4.3) can be computed as

$$\frac{1}{5}\sum_{i=1}^{5}\frac{\eta^i}{C^i}u_k^i = \frac{1}{5}\left(\frac{(0.785)(1.6)}{4.293}+\frac{(0.839)(3.6)}{5.249}\right.$$
$$\left.+\frac{(0.768)(4.6)}{4.717}+\frac{(0.803)(5.6)}{4.201}+\frac{(0.900)(7.6)}{4.941}\right)$$
$$= 0.81433.$$

Then each cell's SOC RFF is computed as:

$$\gamma_{s,k}^1 = \frac{0.29257}{0.81433} = 0.35927$$
$$\gamma_{s,k}^2 = \frac{0.57542}{0.81433} = 0.70661$$
$$\gamma_{s,k}^3 = \frac{0.74895}{0.81433} = 0.91970$$
$$\gamma_{s,k}^4 = \frac{1.07041}{0.81433} = 1.31445$$
$$\gamma_{s,k}^5 = \frac{1.38433}{0.81433} = 1.69994.$$

For easy reference, the computed SOC RFF for each cell is listed in Table 4.2. Given u_k^i for each cell and T_s, the change in average SOC for this time step can be found to be $-2.26205\cdot 10^{-5}$ (negative since current is being drawn out of the battery). Using *the computed RFFs, the change in each cell's SOC can be calculated,*

$$\begin{aligned}
\Delta s_k^1 &= \gamma_k^1 \Delta s_{\mu,k} = -8.12692 \cdot 10^{-6} \\
\Delta s_k^2 &= \gamma_k^2 \Delta s_{\mu,k} = -1.59839 \cdot 10^{-5} \\
\Delta s_k^3 &= \gamma_k^3 \Delta s_{\mu,k} = -2.08041 \cdot 10^{-5} \\
\Delta s_k^4 &= \gamma_k^4 \Delta s_{\mu,k} = -2.97336 \cdot 10^{-5} \\
\Delta s_k^5 &= \gamma_k^5 \Delta s_{\mu,k} = -3.84537 \cdot 10^{-5}.
\end{aligned}$$

Now, to show that each cell's SOC change derived from the average indeed matches that which the individual cell dynamics produce, the same quantities are computed based on (2.1a), as follows.

$$\begin{aligned}
\Delta s_k^1 &= -\frac{(0.785)(0.1)(1.6)}{(3600)(4.293)} = -8.12692 \cdot 10^{-6} \\
\Delta s_k^2 &= -\frac{(0.839)(0.1)(3.6)}{(3600)(5.249)} = -1.59839 \cdot 10^{-5} \\
\Delta s_k^3 &= -\frac{(0.768)(0.1)(4.6)}{(3600)(4.717)} = -2.08041 \cdot 10^{-5} \\
\Delta s_k^4 &= -\frac{(0.803)(0.1)(5.6)}{(3600)(4.201)} = -2.97336 \cdot 10^{-5} \\
\Delta s_k^5 &= -\frac{(0.900)(0.1)(7.6)}{(3600)(4.941)} = -3.84537 \cdot 10^{-5}.
\end{aligned}$$

Note that the concept of RFF provides an alternate perspective of the multi-cell state estimation problem: the RFF method requires only the change in average state (SOC and relaxation voltage) be known in order to compute the change in each cell's state estimate. Later on, this technique will prove vital in the reduction of computational overhead for SOC estimation over large N. The concept of RFFs is further illustrated through the following numerical example.

4.2 Constructing the RFF Matrix

In the context of cell SOC estimation in a battery pack, the main concern is considering the states of all cells at once, not individually. For this reason, we recall the sparse state vector from Section 3.1 and compute the pack-level Jacobian

$$\Gamma_k = \frac{\partial X_k}{\partial x_{\mu,k}} = \begin{bmatrix} \Gamma_k^1 & \Gamma_k^2 & \dots & \Gamma_k^N \end{bmatrix}^T, \tag{4.8}$$

where Γ_k (termed as "RFF matrix") is of size $2N \times 2$. Extrapolating (4.7) to the entire pack,

$$X_{k+1} = X_k + \Gamma_k \Delta x_{\mu,k}$$

offers a useful method for computing changes to the entire pack's states as a function of changes in $x_{\mu,k}$, the size of which does not change with N. The next section leverages this core concept into an estimation algorithm – termed "dense extended Kalman filter" (DEKF) – by developing an initial framework and concurrently proving theoretical equivalence to the sparse method outlined in Section 3.1. The next two lemmas related to the left pseudoinverse of Γ will be utilized in Section 5.3.

Lemma 1. *The left pseudoinverse* $\Gamma^\dagger$ *exists and is given by*

$$\Gamma^\dagger = \begin{bmatrix} \frac{\gamma_s^1}{\Sigma_{i=1}^N \left(\gamma_s^i\right)^2} & 0 & \dots & \frac{\gamma_s^N}{\Sigma_{i=1}^N \left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\gamma_V^1}{\Sigma_{i=1}^N (\gamma_V^i)^2} & \dots & 0 & \frac{\gamma_V^N}{\Sigma_{i=1}^N (\gamma_V^i)^2} \end{bmatrix}. \tag{4.9}$$

Proof. Writing out the full form of Γ as defined in (4.8) gives

$$\Gamma = \begin{bmatrix} \gamma_s^1 & 0 & \gamma_s^2 & 0 & \dots & \gamma_s^N & 0 \\ 0 & \gamma_V^1 & 0 & \gamma_V^2 & \dots & 0 & \gamma_V^N \end{bmatrix}^T.$$

Observing the full form of Γ, for each non-zero element in a given column, its corresponding element in the other column is zero, and vice versa. For this reason, it is clear that the column vectors of Γ are linearly independent as there is no nontrivial linear combination of these vectors which equals the zero vector. Such linear independence of the columns of Γ guarantees the existence of a left pseudoinverse [40]. Next,

$$\begin{aligned}\Gamma^T\Gamma &= \begin{bmatrix} \gamma_s^1 & 0 & \dots & \gamma_s^N & 0 \\ 0 & \gamma_V^1 & \dots & 0 & \gamma_V^N \end{bmatrix} \begin{bmatrix} \gamma_s^1 & 0 \\ 0 & \gamma_V^1 \\ \vdots & \vdots \\ \gamma_s^N & 0 \\ 0 & \gamma_V^N \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_{i=1}^N \left(\gamma_s^i\right)^2 & 0 \\ 0 & \Sigma_{i=1}^N (\gamma_V^i)^2 \end{bmatrix},\end{aligned}$$

which is symmetric positive definite. Therefore,

$$\begin{aligned}\left(\Gamma^T\Gamma\right)^{-1} &= \begin{bmatrix} \Sigma_{i=1}^N \left(\gamma_s^i\right)^2 & 0 \\ 0 & \Sigma_{i=1}^N (\gamma_V^i)^2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \frac{1}{\Sigma_{i=1}^N \left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{1}{\Sigma_{i=1}^N (\gamma_V^i)^2} \end{bmatrix}.\end{aligned}$$

Next, multiplying $\left(\Gamma^T\Gamma\right)^{-1}$ by Γ^T from the right gives

$$\begin{aligned}
\Gamma^\dagger &= \left(\Gamma^T\Gamma\right)^{-1}\Gamma^T \\
&= \begin{bmatrix} \frac{1}{\Sigma_{i=1}^N\left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{1}{\Sigma_{i=1}^N(\gamma_V^i)^2} \end{bmatrix} \begin{bmatrix} \gamma_s^1 & 0 & \dots & \gamma_s^N & 0 \\ 0 & \gamma_V^1 & \dots & 0 & \gamma_V^N \end{bmatrix} \\
&= \begin{bmatrix} \frac{\gamma_s^1}{\Sigma_{i=1}^N\left(\gamma_s^i\right)^2} & 0 & \dots & \frac{\gamma_s^N}{\Sigma_{i=1}^N\left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\gamma_V^1}{\Sigma_{i=1}^N(\gamma_V^i)^2} & \dots & 0 & \frac{\gamma_V^N}{\Sigma_{i=1}^N(\gamma_V^i)^2} \end{bmatrix}.
\end{aligned}$$

This completes the proof. □

Remark 3. *To verify that* $\Gamma^\dagger\Gamma = I$*, we have*

$$\Gamma^\dagger\Gamma = \begin{bmatrix} \frac{\Sigma_{i=1}^N\left(\gamma_s^i\right)^2}{\Sigma_{i=1}^N\left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\Sigma_{i=1}^N(\gamma_V^i)^2}{\Sigma_{i=1}^N(\gamma_V^i)^2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

While Lemma 1 and Remark 3 prove that $\Gamma^\dagger\Gamma = I_{2\times 2}$, $\Gamma\Gamma^\dagger$ is generally not an identity matrix. However, the next lemma establishes that $\Gamma\Gamma^\dagger$ acts as an identity matrix when being multiplied to a change in the sparse state ΔX.

Lemma 2. *Denote* $\Delta X_k := X_{k+1} - X_k \in \mathbb{R}^{2N\times 1}$, $\Delta\tilde{X}_k := \Gamma\Gamma^\dagger\Delta X \in \mathbb{R}^{2N\times 1}$*, then we have* $\Delta\tilde{X}_k = \Delta X_k$.

Proof. Retaining the definition of $\Gamma^{\dagger}$ from Lemma 1, the following is computed

$$\Gamma\Gamma^{\dagger} = \begin{bmatrix} \frac{(\gamma_s^1)^2}{\Sigma(\gamma_s^i)^2} & 0 & \dots & \frac{(\gamma_s^1)(\gamma_s^N)}{\Sigma(\gamma_s^i)^2} & 0 \\ 0 & \frac{(\gamma_V^1)^2}{\Sigma(\gamma_V^i)^2} & \dots & 0 & \frac{(\gamma_V^1)(\gamma_V^N)}{\Sigma(\gamma_V^i)^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{(\gamma_s^N)(\gamma_s^1)}{\Sigma(\gamma_s^i)^2} & 0 & \dots & \frac{(\gamma_s^N)^2}{\Sigma(\gamma_s^i)^2} & 0 \\ 0 & \frac{(\gamma_V^N)(\gamma_V^1)}{\Sigma(\gamma_V^i)^2} & \dots & 0 & \frac{(\gamma_V^N)^2}{\Sigma(\gamma_V^i)^2} \end{bmatrix}. \tag{4.10}$$

Note that $\Gamma\Gamma^{\dagger} \in \mathbb{R}^{2N \times 2N}$. For brevity's sake, we prove the equivalence of the first element of $\Delta\tilde{X}_k$. Performing the computation with the result in (4.10) gives the following

$$\begin{aligned} \Delta\tilde{X}_k(1) &= \sum_{j=1}^{N} \frac{(\gamma_s^1)(\gamma_s^j)}{\sum_{i=1}^{N}(\gamma_s^i)^2} \Delta s_k^i \\ &= \sum_{j=1}^{N} \frac{(\gamma_s^1)(\gamma_s^j)}{\sum_{i=1}^{N}(\gamma_s^i)^2} \gamma_s^j \Delta s_{\mu,k} \\ &= \gamma_s^1 \frac{\sum_{j=1}^{N}(\gamma_s^j)^2}{\sum_{i=1}^{N}(\gamma_s^i)^2} \Delta s_{\mu,k} \\ &= \gamma_s^1 \Delta s_{\mu,k} = \Delta s_k^i = \Delta X(1). \end{aligned}$$

Following the same argument, it can be shown that $\Delta\tilde{X}_k = \Gamma\Gamma^{\dagger}\Delta X_k$. □

Lemma 2 reveals a useful property of $\Gamma\Gamma^{\dagger}$: when being left multiplied to a change in the sparse state ΔX, that same change in the sparse state is the result of the calculation.

CHAPTER FIVE

DENSE EXTENDED KALMAN FILTER

This section presents the proposed dense extended Kalman filter (DEKF). The essential idea is to use EKF to estimate the states of an "averaging model", and then distribute the state estimation to each cell using the RFF matrix. In pursuit of a filtering algorithm that estimates over the dense state, we first define the dense model as follows

$$x_{\mu,k+1} = A_\mu x_{\mu,k} + B_\mu U_k + w_{\mu,k} \tag{5.1a}$$

$$y_{\mu,k} = h_\mu(X_k, U_k) + v_{\mu,k}, \tag{5.1b}$$

where $x_{\mu,k} := \begin{bmatrix} s_{\mu,k} & V_{\mu,k} \end{bmatrix}^T$, $A_\mu = \Gamma_k^\dagger A \Gamma_k \in \mathbb{R}^{2\times 2}$, $B_\mu = \Gamma_k^\dagger B \in \mathbb{R}^{2\times N}$, $h_\mu(\hat{X}_{k+1}^-, U_k) = \frac{1}{N} h(\hat{X}_{k+1}^-, U_k)$, $w_{\mu,k} \sim \mathcal{N}(0, \Gamma_k^\dagger Q_k \Gamma_k)$ and $v_{\mu,k} \sim \mathcal{N}(0, \frac{1}{N^2} R_k)$.

5.1 Dense Time Update

Given the dense model (5.1), its time update equations are given by

$$\hat{x}_{\mu,k+1}^- = A_\mu \hat{x}_{\mu,k}^+ + B_\mu U_k \tag{5.2a}$$

$$P_{\mu,k+1}^- = A_\mu P_{\mu,k}^+ A_\mu^T + Q_{\mu,k}. \tag{5.2b}$$

The estimate over the dense state vector is then "distributed" to each cell using the RFF matrix, as follows:

$$\hat{X}_{k+1}^- = \hat{X}_k^+ + \Gamma_k (\hat{x}_{\mu,k+1}^- - \hat{x}_{\mu,k}^+) \tag{5.3a}$$

$$P_{k+1}^- = \Gamma_k P_{\mu,k+1}^- \Gamma_k^T. \tag{5.3b}$$

The following theorem derives the above dense time update, guaranteeing its near-equivalence to the sparse time update.

Theorem 2. *Under the assumption that $T_s << \tau_p^i$, the time update on $\hat{X}_{k+1}^-$ as computed by (5.2a) and (5.3a) is almost equivalent to the sparse time update as computed by (3.1a), and P_{k+1}^- as computed by (5.2b) and (5.3b) is equivalent to the sparse time update as computed by (3.1b).*

Proof. Let's begin with (3.1a). First, apply the definition (4.8) to solve the matrix-differential equation for sparse vector X_k in terms of $x_{\mu,k}$:

$$\Gamma_k = \frac{\partial X_k}{\partial x_{\mu,k}}$$

$$X_k = \Gamma_k x_{\mu,k} - \Gamma_k x_{\mu,0} + X_0.$$

$$X_k = \Gamma_k x_{\mu,k} - \Gamma_k \Lambda X_0 + X_0$$

$$X_k = \Gamma_k x_{\mu,k} + (I - \Gamma_k \Lambda) X_0, \tag{5.4}$$

where $\Lambda = \frac{1}{N}\begin{bmatrix} I & I & \dots & I \end{bmatrix}$ and is of size $2 \times 2N$, X_0 is an initial condition for the sparse state vector, and $x_{\mu,0}$ is an initial condition for dense state vector, which is derived directly from X_0. Substituting (5.4) into (3.1a),

$$\Gamma_k \hat{x}_{\mu,k+1}^- + (I - \Gamma_k \Lambda) X_0 = A(\Gamma_k \hat{x}_{\mu,k}^+ + (I - \Gamma_k \Lambda) X_0) + BU_k$$

$$\Gamma_k \hat{x}_{\mu,k+1}^- = A \Gamma_k \hat{x}_{\mu,k}^+ + (A(I - \Gamma_k \Lambda) - (I - \Gamma_k \Lambda)) X_0 + BU_k.$$

Multiplying the left pseudoinverse $\Gamma_k^\dagger$ returns

$$\begin{aligned} \hat{x}_{\mu,k+1}^- &= \Gamma_k^\dagger A \Gamma_k \hat{x}_{\mu,k}^+ + \Gamma_k^\dagger (A(I - \Gamma_k \Lambda) - (I - \Gamma_k \Lambda)) X_0 \\ &\qquad + \Gamma_k^\dagger BU_k \\ &= A_\mu \hat{x}_{\mu,k}^+ + \Gamma_k^\dagger (A - I)(I - \Gamma_k \Lambda) X_0 + B_\mu U_k, \end{aligned} \tag{5.5}$$

effectively isolating $\hat{x}_{\mu,k+1}^-$.

As a result of the discretized system dynamics (2.3), for a sufficiently small $\frac{T_s}{\tau_p^i}$, $A \to I$. For this reason, (5.5) can be approximated as,

$$\hat{x}_{\mu,k+1}^{-} \approx A_\mu \hat{x}_{\mu,k}^{+} + B_\mu U_k.$$

To prove the equivalence of the time update on the covariance matrix, we first have

$$P_{\mu,k+1}^{-} = \mathbb{E}\left[(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})^T\right] \tag{5.6}$$

over the prediction. Multiplying Γ_k to (5.6), we have

$$\begin{aligned}
\Gamma_k P_{\mu,k+1}^{-} \Gamma_k^T = & \\
&= \Gamma_k \mathbb{E}\left[(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})^T\right] \Gamma_k^T \\
&= \mathbb{E}\left[\Gamma_k (x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})^T \Gamma_k^T\right] \\
&= \mathbb{E}\left[(X_{k+1} - \hat{X}_{k+1}^{-})(X_{k+1} - \hat{X}_{k+1}^{-})^T\right] \\
&= P_{k+1}^{-},
\end{aligned} \tag{5.7}$$

which works out to be the sparse predicted process covariance P_{k+1}^{-} as calculated in (3.1b). Substituting (3.1b) into the right hand side of (5.7), we have

$$\Gamma_k P_{\mu,k+1}^{-} \Gamma_k^T = A \Gamma_k P_{\mu,k}^{+} \Gamma_k^T A^T + Q_k. \tag{5.8}$$

Multiplying $\Gamma_k^\dagger$, we have

$$\Gamma_k^\dagger \left[\Gamma_k P_{\mu,k+1}^{-} \Gamma_k^T\right] (\Gamma_k^\dagger)^T = \Gamma_k^\dagger \left[A \Gamma_k P_{\mu,k}^{+} \Gamma_k^T A^T + Q_k\right] (\Gamma_k^\dagger)^T$$

Utilizing the fact that $\Gamma_k^\dagger \Gamma_k = I$ (Lemma 1), we have

$$\begin{aligned}
P_{\mu,k+1}^{-} &= \left(\Gamma_k^\dagger A \Gamma_k\right) P_{\mu,k}^{+} \left(\Gamma_k^\dagger A \Gamma_k\right)^T + \Gamma_k^\dagger Q_k (\Gamma_k^\dagger)^T \\
&= A_\mu P_{\mu,k}^{+} A_\mu^T + Q_{\mu,k}.
\end{aligned} \tag{5.9}$$

This completes the proof. □

Remark 4. *Though the proposed dense time update (5.2)-(5.3) incurs certain error, as captured by Theorem 2, it significantly reduces the FLOPs requirement. In fact, the regular sparse time update requires a theoretical FLOP count of* $12N^2 + 6N$*, whereas the proposed dense time update requires only* $36N + 14$*, due largely in part to many of the associated dense matrices having constant size with respect to cell number. See Section 6.2 for more details.*

5.2 Dense Measurement Update

Continuing with the dense model (5.1), its measurement update equations are given by

$$K_{\mu,k+1} = \frac{P^-_{\mu,k+1} H^T_{\mu,k+1}}{H_{\mu,k+1} P^-_{\mu,k+1} H^T_{\mu,k+1} + R_{\mu,k}}, \tag{5.10a}$$

$$\hat{x}^+_{\mu,k+1} = \hat{x}^-_{\mu,k+1} + K_{\mu,k+1}\left(y_{\mu,k+1} - h_\mu(\hat{X}^-_{k+1}, U_k)\right) \tag{5.10b}$$

$$\hat{X}^+_{k+1} = \hat{X}^-_{k+1} + \Gamma_k(\hat{x}^+_{\mu,k+1} - \hat{x}^-_{\mu,k+1}) \tag{5.10c}$$

$$P^+_{\mu,k+1} = (I - K_{\mu,k+1} H_{\mu,k+1}) P^-_{\mu,k+1}, \tag{5.10d}$$

where $R_{\mu,k} = \frac{R_k}{N^2}$, $y_{\mu,k+1} = \frac{1}{N} y_{k+1}$, and $H_{\mu,k+1} \in \mathbb{R}^{1\times 2}$ as given by

$$H_{\mu,k+1} = \left.\frac{\partial(\frac{1}{N} h(X,U))}{\partial x_\mu}\right|_{(\hat{X}^-_{k+1}, U_k)}. \tag{5.11}$$

The two next lemmas connect the dense Kalman gain $K_{\mu,k+1}$ and $H_{\mu,k+1}$ to the sparse Kalman gain K_{k+1} and H_{k+1}.

Lemma 3. *The dense Jacobian* $H_{\mu,k+1}$ *as computed by (5.11) and the sparse Jacobian* H_{k+1} *as computed by (3.6) satisfy*

$$H_{\mu,k+1} = \frac{1}{N} H_{k+1} \Gamma_k. \tag{5.12}$$

Proof. For brevity, we opt for short-hand notation $h := h(X,U)$ for derivations contained within this section. The scalar $\frac{1}{N}$ can be moved to the front of the expression in (5.11) and the Jacobian can be evaluated:

$$\begin{aligned}
H_{\mu,k+1} =& \frac{1}{N}\frac{\partial h}{\partial x_\mu} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\frac{\partial V_{oc}^i}{\partial x_\mu(1)} & -\frac{1}{N}\frac{\partial \sum_{i=1}^{N} V^i}{\partial x_\mu(2)} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\frac{\partial V_{oc}^i}{\partial x_\mu(1)} & -\frac{1}{N}N\frac{\partial x_\mu(2)}{\partial x_\mu(2)} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\frac{\partial V_{oc}^i}{\partial x_\mu(1)} & -\frac{1}{N}N \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{N}(\frac{\partial V_{oc}^1}{\partial x_\mu(1)} + \frac{\partial V_{oc}^2}{\partial x_\mu(1)} + \cdots + \frac{\partial V_{oc}^N}{\partial x_\mu(1)}) & -1 \end{bmatrix}.
\end{aligned}$$

Recall from (4.6) that $\frac{\partial x^i(1)}{\partial x_\mu(1)} = \gamma_s^i$, therefore

$$\begin{aligned}
H_{\mu,k+1} &= \begin{bmatrix} \frac{1}{N}(\gamma_s^1\frac{\partial V_{oc}^1}{\partial x_1(1)} + \cdots + \gamma_s^N\frac{\partial V_{oc}^N}{\partial x_N(1)}) & -1 \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{N}\frac{1}{N}\gamma_s^i\frac{\partial V_{oc}^i}{\partial x_i(1)} & -1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\gamma_s^i\frac{\partial V_{oc}^i}{\partial x_i(1)} & -1 \end{bmatrix}.
\end{aligned}$$

Now, multiplying (3.6) by Γ_k gives

$$\begin{aligned}
H_{k+1}\Gamma_k &= \begin{bmatrix} \frac{\partial V_{oc}^1}{\partial x_1(1)} & -1 & \frac{\partial V_{oc}^2}{\partial x_2(1)} & -1 & \cdots & \frac{\partial V_{oc}^N}{\partial x_N(1)} & -1 \end{bmatrix}\Gamma_k \\
&= \begin{bmatrix} \sum_{i=1}^{N}\gamma_{s,k}^i\frac{\partial V_{oc}^i}{\partial x_i(1)} & -\sum_{i=1}^{N}\gamma_{V,k}^i \end{bmatrix}.
\end{aligned}$$

Without loss of generality, $\sum_{i=1}^{N}\gamma_V^i = N$. Hence,

$$\begin{aligned}
\frac{1}{N}H_{k+1}\Gamma_k =& \frac{1}{N}\begin{bmatrix} \sum_{i=1}^{N}\gamma_{s,k}^i\frac{\partial V_{oc}^i}{\partial x_i(1)} & -N \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}\gamma_{s,k}^i\frac{\partial V_{oc}^i}{\partial x_i(1)} & -1 \end{bmatrix} = H_{\mu,k+1}.
\end{aligned}$$

This completes the proof. □

Lemma 4. *The dense Kalman gain $K_{\mu,k+1}$ as computed by (5.10a) and the sparse Kalman gain K_{k+1} computed by (3.7a) satisfy*

$$K_{k+1} = \frac{1}{N}\Gamma_k K_{\mu,k+1} \tag{5.13}$$

Proof. According to (5.10a), (5.12) and the fact that $R_{\mu,k} = \frac{R_k}{N^2}$, we have

$$\begin{aligned}
\frac{1}{N}\Gamma_k K_{\mu,k+1} =& \frac{1}{N}\Gamma_k \frac{P^-_{\mu,k+1}H^T_{\mu,k+1}}{H_{\mu,k+1}P^-_{\mu,k+1}H^T_{\mu,k+1} + \frac{1}{N^2}R_k} \\
=& \frac{1}{N}\frac{\Gamma_k P^-_{\mu,k+1}(\frac{1}{N}H_{k+1}\Gamma_k)^T}{(\frac{1}{N}H_{k+1}\Gamma_k)P^-_{\mu,k+1}(\frac{1}{N}H_{k+1}\Gamma_k)^T + \frac{1}{N^2}R_k} \\
=& \frac{1}{N}N\frac{\Gamma_k P^-_{\mu,k+1}(H_{k+1}\Gamma_k)^T}{(H_{k+1}\Gamma_k)P^-_{\mu,k+1}(H_{k+1}\Gamma_k)^T + R_k} \\
=& \frac{\Gamma_k P^-_{\mu,k+1}\Gamma^T_k H^T_{k+1}}{H_{k+1}\Gamma_k P^-_{\mu,k+1}\Gamma^T_k H^T_{k+1} + R_k}
\end{aligned}$$

Utilizing (5.7), we have

$$\frac{1}{N}\Gamma_k K_{\mu,k+1} = \frac{P^-_{k+1}}{H_{k+1}P^-_{k+1}H^T_{k+1} + R_k} = K_{k+1}$$

This completes the proof. □

Now we are ready to present the main result of this section by introducing the following theorem that guarantees the equivalence of (5.10) to the sparse measurement update outlined in (3.7).

Theorem 3. *The measurement update on $\hat{X}^+_{k+1}$ and P^+_{k+1} as computed by (5.10) is equivalent to the sparse measurement update as computed by (3.7).*

Proof. From (5.10c), we have

$$\begin{aligned}
\hat{X}^{+}_{k+1} =& \hat{X}^{-}_{k+1} + \Gamma_k(\hat{x}^{+}_{\mu,k+1} - \hat{x}^{-}_{\mu,k+1}) \\
=& \hat{X}^{-}_{k+1} + \Gamma_k K_{\mu,k+1}\left(y_{\mu,k+1} - h_\mu(\hat{X}^{-}_{k+1}, U_k)\right) \\
=& \hat{X}^{-}_{k+1} + N K_{k+1}\left(y_{\mu,k+1} - h_\mu(\hat{X}^{-}_{k+1}, U_k)\right) \\
=& \hat{X}^{-}_{k+1} + K_{k+1}\left(N y_{\mu,k+1} - N h_\mu(\hat{X}^{-}_{k+1}, U_k)\right) \\
=& \hat{X}^{-}_{k+1} + K_{k+1}\left(y_{k+1} - h(\hat{X}^{-}_{k+1}, U_k)\right)
\end{aligned}$$

which establishes the equivalence of the state correction of (5.10) with that of (3.7).

As for the covariance equation, begin with (5.10d) and substitute (5.7) to get

$$\begin{aligned}
P^{+}_{\mu,k+1} &= (I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1})P^{-}_{\mu,k+1} \\
\Gamma_k P^{+}_{\mu,k+1}\Gamma_k^T &= \Gamma_k(I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1})P^{-}_{\mu,k+1}\Gamma_k^T \\
P^{+}_{k+1} &= \Gamma_k(I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1})P^{-}_{\mu,k+1}\Gamma_k^T.
\end{aligned}$$

Substituting (5.13) and (5.12) and simplifying further, we get

$$\begin{aligned}
P^{+}_{k+1} &= \Gamma_k(I_{2\times 2} - N\Gamma_k^{\dagger}K_{k+1}\frac{1}{N}H_{k+1}\Gamma_k)P^{-}_{\mu,k+1}\Gamma_k^T \\
&= \Gamma_k(\Gamma_k^{\dagger}I_{2N\times 2N}\Gamma_k - \Gamma_k^{\dagger}K_{k+1}H_{k+1}\Gamma_k)P^{-}_{\mu,k+1}\Gamma_k^T \\
&= \Gamma_k\Gamma_k^{\dagger}(I_{2N\times 2N} - K_{k+1}H_{k+1})\Gamma_k P^{-}_{\mu,k+1}\Gamma_k^T \\
&= \Gamma_k\Gamma_k^{\dagger}(I_{2N\times 2N} - K_{k+1}H_{k+1})P^{-}_{k+1} \\
&= \Gamma_k\Gamma_k^{\dagger}P^{+}_{k+1} = P^{+}_{k+1}.
\end{aligned}$$

Note that Lemma 2 is utilized to arrive the last equality. □

Remark 5. *In addition to the proposed dense measurement update's equivalence to the sparse measurement update, as captured by Theorem 3, it significantly reduces the FLOPs requirement. In fact, the proposed dense algorithm reduces the FLOP count from cubic* ($16N^3 + 4N^2 + (2P+4)N + 1$) *to linear* ($(2P+12)N + 25$) *complexity. This major* *reduction in computation time is owed mostly to the Kalman gain, measurement Jacobian, and process covariance matrices all having constant size in the dense measurement update. This fact confers a constant FLOP count on the dense state and covariance updates.*

5.3 Adaptive DEKF

Recall that $Q_{\mu,k} = \Gamma_k^{\dagger} Q_k \Gamma_k$. Therefore, $Q_{\mu,k}$ must also be computed at each time step since Γ_k changes with each time step (recall its dependence on the cell currents as outlined in (4.3), (4.4), and (4.6)). While in the context of the regular DEKF this would be the case, there are existing methods of adaptive Kalman filtering [24,41,42] in which the dense noise covariance matrices are not computed as a function of Γ, but instead as approximate solutions to the following optimization problem

$$\begin{aligned} \Theta^* = & \arg\min_{Q_k, R_k} \left[J(\Theta | Y_M) \right] \\ & \text{s.t.} \quad Q_k \succeq 0, R_k \succ 0, \end{aligned} \tag{5.14}$$

where the objective function is defined as

$$J(\Theta | Y_M) = \sum_{i=k-M+1}^{k} \left[\ln |\Sigma_i| + \nu_i^T \Sigma_i^{-1} \nu_i \right],$$

$\Theta := \begin{bmatrix} Q_k & R_k \end{bmatrix}$, the pre-fit residual $\nu_k := y_k - \hat{y}_k^-$ assumes a Gaussian distribution of $\mathcal{N}(0, \Sigma_k)$, M is an adjustable parameter describing the size of the window of past measurements, and $Y_M := \begin{bmatrix} y_{k-M+1} & y_{k-M} & \cdots & y_{k-1} & y_k \end{bmatrix}$. A full derivation of the solution to (5.14) can be located in Appendix C of [41], where suitable approximations of the optimal noise covariance matrices that maintain positive definiteness are found to be

$$Q_k^* = K_k \left[\frac{1}{M} \sum_{i=i_0}^{k} v_i v_i^T \right] K_k^T \tag{5.15}$$

$$R_k^* = \frac{1}{M} \sum_{i=i_0}^{k} \left[\varepsilon_i \varepsilon_i^T + H_i P_i^+ H_i^T \right], \tag{5.16}$$

where $i_0 = k - M + 1$ and post-fit residual $\varepsilon_k := y_k - \hat{y}_k^+$. This adaptive formulation of the covariance matrices is worked into the DEKF, which is referred to as the dense adaptive extended Kalman filter (DAEKF) and is the solution which generates the simulation results shown later in Section 6.2.

Remark 6. *The additional number of FLOPs incurred by the dense without the adaptive extension is linear (*$8N-3$*) and occurs in the time update when computing* $\Gamma_k^\dagger Q_k \Gamma_k$ *(recall that* Γ_k *is a function of balancing currents). The adaptive step removes the dependence of this computation on N, and instead renders its FLOPs count to a function of window size M (*$6M+21$*).*

5.4 Complete DAEKF Algorithm

The proposed DAEKF algorithm can be divided into two parts: the time update and the measurement update. The former is summarized in Algorithm 1, and the latter in Algorithm 2.

First we describe the time update portion of the DAEKF algorithm. In particular, Lines 3 and 4 compute Γ_k and $\Gamma_k^\dagger$ given U_k, which are then used to calculate $A_{\mu,k}$ and $B_{\mu,k}$ as shown in Line 5. Lines 6 through 9 check if this particular iteration of the DAEKF is the first one, in which case the program initializes $P_{\mu,0}^+$, $Q_{\mu,0}$, and $R_{\mu,0}$ in terms of sparse covariances, Γ_0, and $\Gamma_0^\dagger$. Line 11 uses the results computed in Line 5 and the corrected dense state $\hat{x}_{\mu,k}^+$ from the previous time step to generate the dense prediction $\hat{x}_{\mu,k+1}^-$. Line 12 uses the result obtained in Line 11 to predict the sparse state $\hat{X}_{k+1}^-$, which is a required computation since there is not yet a known way to develop a compact V_{oc} curve which is strictly a function of x_μ (see a more detailed explanation in Section 6.2). Line 13 computes the predicted dense process covariance $P^-_{\mu,k+1}$. The results obtained from Lines 11 through 13 are outputs of the procedure, which are passed onto the measurement update function.

Starting off the measurement update portion of the DAEKF algorithm, Line 2 evaluates a least-squares fit of each cell's differentiated OCV (detailed explanation in Section 6.2) as a function of sparse prediction $\hat{X}^-_{k+1}$ to compute H_μ. Line 3 utilizes this result, along with $P^-_{\mu,k+1}$ and R_μ to compute the dense Kalman gain $K_{\mu,k+1}$. Line 5 corrects the prediction, computing the difference between the measurement $y_{\mu,k+1}$ and the predicted measurement, and using that to calculate $\hat{x}^+_{\mu,k+1}$. Line 6 computes the updated sparse vector $\hat{X}^+_{k+1}$ as a function of Γ_k and the differences in the result from Line 5 and the predicted dense state. Line 7 simply computes the the updated dense process covariance $P^+_{\mu,k+1}$ as a function of the results from Lines 2 and 3, and the procedure input $P^-_{\mu,k+1}$. Line 9 uses the results of Line 2 and (70b) to approximate $h_\mu(\hat{X}^+_{k+1})$, which the predicted measurement function evaluated over $\hat{X}^+_{k+1}$. This result is applied in Line 10, which computes the dense post-fit residual $\varepsilon_{\mu,k+1}$. Lines 11 through 14 consist on a conditional statement which checks at least $M-1$ iterations have elapsed. If such is the case, Lines 12 and 13 will overwrite Q_μ and R_μ with the sub-optimal solutions to (5.14). Line 15 simply increments the time step variable k to prepare for the next iteration of the DAEKF. Line 16 returns the updated dense state $\hat{x}_{\mu,k+1}$, updated dense process covariance $P^-_{\mu,k+1}$, and update sparse state $\hat{X}^+_{k+1}$, which are all fed back into the time update function for the following iteration.

Algorithm 1 DAEKF Algorithm: Time Update

```
input  : $\hat{x}^+_{\mu,k}$, $P^+_{\mu,k}$, $U_k$, $\hat{X}^+_k$, $k$
output: $\hat{x}^-_{\mu,k+1}$, $P^-_{\mu,k+1}$, $\hat{X}^-_{k+1}$
$\Gamma_k \leftarrow$ computing (4.8);
$\Gamma^\dagger_k \leftarrow$ computing (4.9);
$A_{\mu,k} \leftarrow \Gamma^\dagger_k A \Gamma_k$; $B_{\mu,k} \leftarrow \Gamma^\dagger_k B$;
if $k = 0$ then
    $P^+_{\mu,0} \leftarrow \Gamma^\dagger P^+_0 (\Gamma^\dagger)^T$;
    $Q_\mu \leftarrow \Gamma^\dagger Q (\Gamma^\dagger)^T$;
    $R_\mu \leftarrow \frac{1}{N^2} R$;
end
$\hat{x}^-_{\mu,k+1} \leftarrow A_{\mu,k}\hat{x}^+_{\mu,k} + B_{\mu,k}U_k$;
$\hat{X}^-_{k+1} \leftarrow \hat{X}^+_k + \Gamma_k(\hat{x}^-_{\mu,k+1} - \hat{x}^+_{\mu,k})$;
$P^-_{\mu,k+1} \leftarrow A_{\mu,k}P^+_{\mu,k}A^T_{\mu,k} + Q_\mu$;
```

Algorithm 2 DAEKF Algorithm: Measurement Update

```
input  : $\hat{x}^-_{\mu,k+1}$, $P^-_{\mu,k+1}$, $\hat{X}^-_{k+1}$, $y_{\mu,k+1}$, $k$
output: $\hat{x}^+_{\mu,k+1}$, $P^+_{\mu,k+1}$, $\hat{X}^+_{k+1}$
$H_\mu \leftarrow$ evaluates (6.3) at $\hat{X}^-_{k+1}$;
$K_{\mu,k+1} \leftarrow$ computing (5.10a);

$\hat{x}^+_{\mu,k+1} \leftarrow \hat{x}^-_{\mu,k+1} + K_{\mu,k+1}(y_{\mu,k+1} -$ (6.4));
$\hat{X}^+_{k+1} \leftarrow \hat{X}^-_{k+1} + \Gamma_k(\hat{x}^+_{\mu,k+1} - \hat{x}^-_{\mu,k+1})$;
$P^+_{\mu,k+1} \leftarrow (I - K_{\mu,k+1}H_\mu)P^-_{\mu,k+1}$;

$h_\mu(\hat{X}^+_{k+1}) \leftarrow$ computing (6.5);
$\varepsilon_{\mu,k+1} \leftarrow y_{\mu,k+1} - h_\mu(\hat{X}^+_{k+1})$;
if $k \geq M-1$ then
    $Q_\mu \leftarrow$ computing (5.15);
    $R_\mu \leftarrow$ computing (5.16);
end
$k \leftarrow k+1$
```

CHAPTER SIX

SIMULATION RESULTS

6.1 Implementation Details and Methodology

This subsection elaborates on the details specific to the DEKF implementation which yields the simulation results shown later on, the selected hardware for simulation, as well as considerations regarding the direct computation of FLoating-point OPerations (FLOPs) for each step.

Recall that $H_{\mu,k+1}$ is computed at each time step using (5.12) that relies on the computation of sparse H_{k+1}, which requires each cell's OCV-SOC curve be stored in memory and its derivative evaluated over $\hat{X}_{k+1}^{-}$ at each time step. As mentioned earlier, the OCV-SOC curve for any particular cell is typically stored in a lookup table – the size of which is directly related to the desired resolution. For a small number of cells, this may be a viable method, but large memory requirements – and thus poor scalability – become problematic as the number of cells grows. Instead, this work devotes offline computation time to calculating N least-squares polynomials of degree P,

$$\min_{p^i} \|S^i p^i - V_{oc}^i\|_2^2, \quad 1 \leq i \leq N, \tag{6.1}$$

each evaluated over high-resolution, OCV-SOC data from its respective cell. In the case of (6.1), p^i is a $(P+1)\times 1$ vector of polynomial coefficients, S_i is a $L\times(P+1)$ design matrix computed from the SOCs of L samples from the ith cell's OCV-SOC curve, and V_{oc}^i is the corresponding output vector of size $L\times 1$. The vector p^{i*}, and therefore the polynomial, which solves (6.1) is defined as

$$p^{i*} = S^{i\dagger} V_{oc}^{i}$$
$$F^{i}(s^{i}) = \sum_{j=0}^{P} p^{i*}(j)(s^{i})^{j}, \tag{6.2}$$

where s^j is the ith cell's SOC raised to the j^{th} power. At each time step, the least-squares fit $F^i(s)$ ands its derivative $\dot{F}^i(s) := \frac{dF^i}{ds^i}$ is evaluated to complete the following steps of the DAEKF algorithm

$$H^{-}_{\mu,k+1} = \left.\frac{\partial h_\mu}{\partial x_\mu}\right|_{x_\mu = \hat{x}^{-}_{\mu,k+1}} \approx \begin{bmatrix} \frac{1}{N} \cdot \sum_{i=1}^{N} \gamma^{i}_{s,k} \dot{F}^{i}(x_i(1)) & -1 \end{bmatrix} \tag{6.3}$$

$$h_\mu(\hat{X}^{-}_{k+1}) \approx \sum_{i=1}^{N} \Big(F^{i}(x_i(1)) - R^{i}_{o} u^{i}_{k} - x_i(2) \Big) \tag{6.4}$$

$$h_\mu(\hat{X}^{+}_{k+1}) \approx h_\mu(\hat{X}^{-}_{k+1}) + H^{-}_{\mu,k+1}(\hat{x}^{+}_{\mu,k+1} - \hat{x}^{-}_{\mu,k+1}), \tag{6.5}$$

where (6.3) and (6.4) are derived from $\dot{F}^i(s)$ and $F^i(s)$, respectively, to compute the first-order Taylor approximation of the measurement functions in (6.5), which is used to compute the post-fit residual

$$\varepsilon_{\mu,k+1} = y_{\mu,k+1} - h_\mu(\hat{X}^{+}_{k+1})$$

to update R_μ as shown in (5.16). For this specific implementation, a 21^{st}-degree polynomial is used to approximate the OCV-SOC behavior of each cell, meaning a 20^{th}-degree polynomial approximates its derivative. Surely, polynomials of this size are excessive in approximating open-circuit voltage characteristics, and in most cases can require fewer terms.

In the context of computing theoretical resource consumption, the considerations being made are (1) the structural redundancies of the matrices involved in calculation (i.e. full multiplications need not be performed for matrices A and B as they are block-diagonal), (2) matrix multiplication of two arbitrary matrices $A_{m\times n}$ and $B_{n\times p}$ requires $m \times p \times (2n-1)$ FLOPs, (3) Horner's Rule is the method employed for evaluating polynomials of degree P, which requires $2P$ FLOPs to execute, (4) a matrix's pseudoinverse is computed in accordance with its definition, and (5) double-precision floating-point format is used to represent and store numerical data in memory.

For each cell involved in the simulation, the following electrical parameters are selected from a Gaussian distribution

$$\eta^i \sim \mathscr{N}(0.9, 0.1), \quad C^i \sim \mathscr{N}(5, 0.5)$$

as a means of inducing heterogeneity unto the pack. For clarity, other circuit parameters $R_p = 1.1343 \cdot 10^{-2}$, $C_p = 2.8212 \cdot 10^3$, and $R_o = 8.7826 \cdot 10^{-3}$ are kept constant for all cells. Note that all battery parameters are derived from [43–45], which utilize a battery model simulation in which each circuit parameter's value is determined from experiments and stored as lookup tables of SOC and temperature. The constant values used are a result of averaging each parameter's value over the full range of SOCs at a temperature of $15°C$. Lastly, the following simulation results were obtained by running the DAEKF algorithm in MATLAB on an Intel i7-9750H CPU with six cores, 8 GB of RAM and 12 MB of cache running at 2.60 GHz.

6.2 Results and Discussion

The cell current trajectories are selected to be an exponential family of functions that are symmetric about a nominal battery pack current $u_k := 4.6$ A. (See Fig. 6.1 for an example.) Furthermore, the initial sparse process covariance, the standard deviations for each cell's process noise w_k^i and the pack's measurement noise v_k are initialized as a block diagonal matrix of 10^{-4}, 10^{-5}, and 10^{-2} respectively, and as a result of (5.7) and the equations for $Q_{\mu,k}$ and $R_{\mu,k}$ given in Section 5.3, the initial dense covariance matrices are

$$P_{\mu,0} = \begin{bmatrix} 9.884 \cdot 10^{-5} & 0 \\ 0 & 1.000 \cdot 10^{-4} \end{bmatrix}$$
$$Q_{\mu,0} = \begin{bmatrix} 1.977 \cdot 10^{-11} & 0 \\ 0 & 2.000 \cdot 10^{-11} \end{bmatrix}$$
$$R_{\mu,0} = \begin{bmatrix} 4 \cdot 10^{-6} \end{bmatrix},$$

$N = 5$, and $M = 20$. The measured terminal voltage and the cell currents are shown in Fig. 6.1. Over a period of 1500 seconds, Fig. 6.2 illustrates the results of the simulation, which show that the DAEKF exhibits good performance in estimating SOC over all five cells with the RMSE being on the order of 10^{-4} at best, and the maximum RMSE on predicted average terminal voltage $h_\mu(\hat{x}^+_{k+1})$ being within 20 mV ($\approx 0.6\%$ of its nominal value). Shifting to the sparse EKF's results, the estimation error is surprisingly slightly greater than that of the DAEKF, residing mostly within the neighborhood of $2 \cdot 10^{-3}$ to $3 \cdot 10^{-3}$. This is an artifact of the large predicted measurement error incurred by the sparse relative to the dense, which is mostly contained within 100 mV. The reason for the larger estimation error in sparse EKF is likely due to the need to invert a larger matrix when calculating Kalman gain, resulting in higher numerical error.

In the context of FLOP count, a comprehensive comparison between the dense and sparse adaptive extended Kalman filters is found in Table 6.1, where M is the size of the measurement window from the adaptive step, N is the number of cells, and P is the degree of the polynomials used to approximate OCV-SOC characteristics. The most laborious step for the sparse AEKF is the measurement update, where there are no structural patterns that can be exploited in the computation of (3.7c). Thus, the multiplication of two arbitrary, square matrices of size $2N \times 2N$ yields a FLOP count proportional to N^3. While

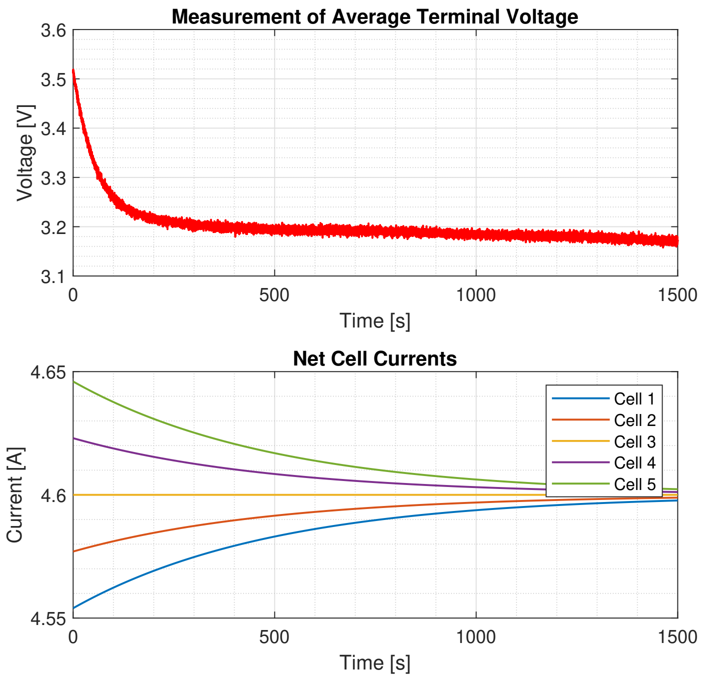

Figure 6.1: Pack characteristics over five cells.

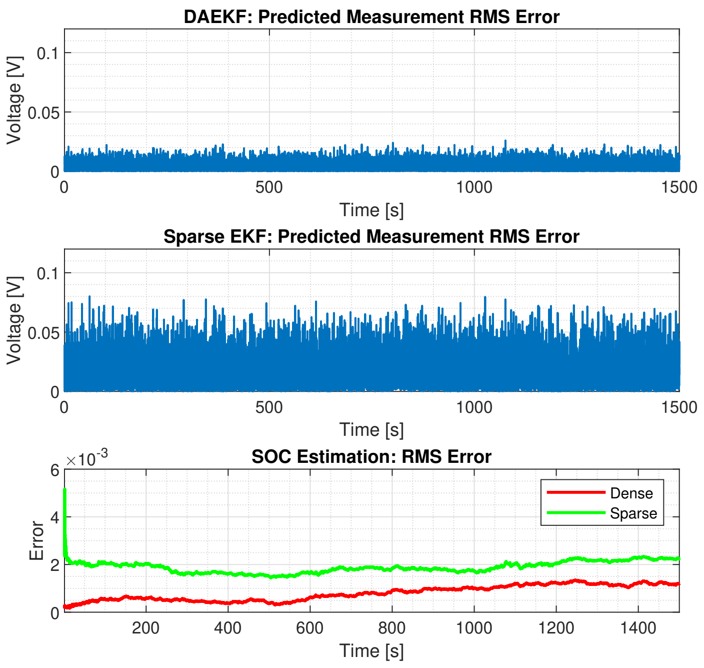

Figure 6.2: Comparison of SOC estimation error and predicted measurement error between the sparse EKF and the DAEKF over five cells.

Table 6.1: FLOP Comparison for Dense Adaptive EKF and Sparse Adaptive EKF

-	Dense	Sparse
Time Update	$36N+14$	$12N^2+6N$
Kalman Gain	$2PN+10$	$8N^2+2PN+1$
Measurement	$(2P+12)N+25$	$16N^3+4N^2+(2P+4)N+1$
Adaptive Step	$6M+21$	$8N^2+8N+6M-1$
Total	$(48+4P)N+6M+70$	$16N^3+30N^2+(4P+18)N+6M+1$

the measurement update is also the most laborious step for the DAEKF, it only grows linearly with N as well as P. The values of M, N, and P used to obtain the results for the 100-cell problem in Fig. 6.3 can be borrowed to get an idea of the FLOPs count for both estimators. Doing so yields a FLOPs count of 16,310,321 for the sparse, and a mere 13,390 for the dense, reinforcing the perceived intractability of the sparse formulation of the AEKF.

A simulation with $N = 100$, i.e., with 100 cells, is also performed. The initial covariance matrices are the following

$$P_{\mu,0} = \begin{bmatrix} 9.819\cdot 10^{-14} & 0 \\ 0 & 9.999\cdot 10^{-14} \end{bmatrix}$$

$$Q_{\mu,0} = \begin{bmatrix} 9.641\cdot 10^{-15} & 0 \\ 0 & 9.999\cdot 10^{-15} \end{bmatrix}$$

$$R_{\mu,0} = \begin{bmatrix} 1\cdot 10^{-8} \end{bmatrix},$$

yielding the results shown in Fig. 6.3, where, in comparison to Fig. 6.2, the effect of a larger cell number on the measurement noise can be observed in the RMSE curve of the predicted terminal voltage. Specifically, the maximum prediction error for terminal voltage is around 5 mV, demonstrating greater noise attenuation as a result of more cells.

In addition to this, the covariance matrices are intentionally initialized such that the model predictions are favored greatly over the measurements. As a result, cumulative error is exhibited, but the DAEKF and sparse EKF take on a nearly identical error trajectory. This observation is reinforced by the plot of the absolute differences between both filters' RMSE curves, which resides firmly in the vicinity of $6 \cdot 10^{-8}$.

Keeping covariance matrices, process noise, and measurement noise the same from Fig. 6.2. Fig. 6.4 plots the estimation error for different number of cells, where 20 simulation trials are run for each cell number. The averages and standard deviations (the error bars) for both the error of the predicted terminal voltage and the estimated SOC are shown here. Observing Fig. 6.4, the averages and standard deviations of the predicted measurement error as well as the average SOC estimation error are decreasing as a function of cell number. Because the inclusion of more cells scales down the measurement by a larger number, measurement noise is more heavily attenuated, which results in the DAEKF weighing measurements more favorably in its estimates. As a result, estimates which rely less on the model are less prone to problems of accumulated error, ergo smaller average error and smaller. It is also worth noting that, the estimation abilities of the DAEKF are consistent across various cell numbers, as demonstrated by Fig. 6.4.

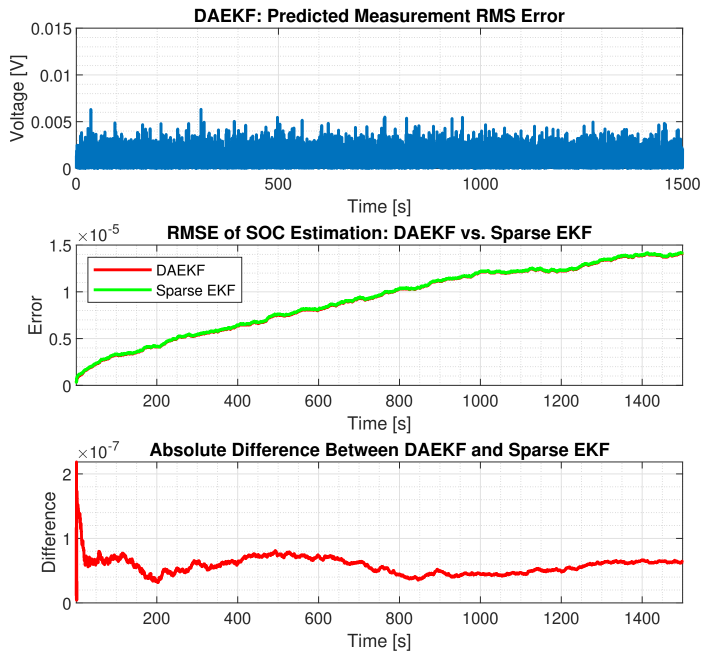

Figure 6.3: DAEKF and sparse EKF performance comparison over 100 cells.

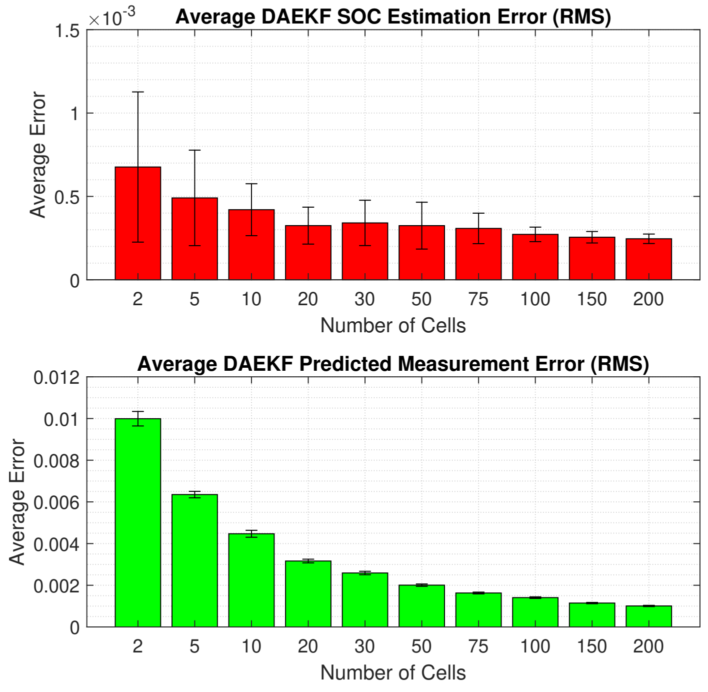

Figure 6.4: Evolution of average RMS errors in estimation and measurement prediction across 20 trials over various cell numbers.

CHAPTER SEVEN

SPECIAL CASE WITH HOMOGENEOUS CELL CURRENT

Up to this point, the calculation of Γ has involved dynamic balancing currents, requiring its computation to occur online. However, the case of no balancing currents, i.e., $u_k^1 = u_k^2 = \cdots = u_k^N$, offers the conversion of the calculation to an exclusively offline format. Without loss of generality, recall (4.3), where the state-of-charge fitness factor is computed to be

$$\gamma_{s,k}^i = \frac{\frac{\eta^i}{C^i} u_k^i}{\frac{1}{N}\sum_{i=1}^N \frac{\eta^i}{C^i} u_k^i} = \frac{\frac{\eta^i}{C^i}}{\frac{1}{N}\sum_{i=1}^N \frac{\eta^i}{C^i}}, \tag{7.1}$$

which is now constant with respect to time. Because cell capacities and efficiencies are quantities known *a priori*, the computation of each cell's SOC RFF in the case of no balancing currents can be relegated to an offline computation. However, the same does not hold for the RFF for relaxation voltage. Recalling how the voltage RFF is defined in (4.4), a similar simplification to the one in (7.1) can be made if $\frac{V_k^i}{\tau_p^i}$ is sufficiently small

$$\gamma_{V,k}^i \approx \frac{\frac{u_k^i}{C_p^i}}{\frac{1}{N}\sum_{i=1}^N \frac{u_k^i}{C_p^i}}. \tag{7.2}$$

However, computing $\gamma_{V,k}^i$ in this way incurs a certain amount of error, which is derived in the following theorem.

Theorem 4. *The error incurred in the computation of (2.1b) over V_{k+1}^i is given by*

$$\left| -\frac{T_s}{\tau_p^i} V_k^i + \frac{T_s}{C_p^i} u_k^i \left(1 - \frac{\sum_{i=1}^N \left(\frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i} \right)}{\sum_{i=1}^N \frac{u_k^i}{C_p^i}} \right) \right|. \tag{7.3}$$

Proof. To assess the error involved in using the offline approximation of $\gamma^i_{V,k}$, we consider the difference between the discrete-time voltage update equation (2.1b) and the voltage update as recovered from multiplying $\gamma^i_{V,k}$ to the change in the average (4.5b)

$$\begin{aligned} f_1 :=& \left(1 - \frac{T_s}{\tau^i_p}\right) V^i_k + \frac{T_s}{C^i_p} u^i_k \\ f_2 :=& V^i_k + \gamma^i_{V,k} \Delta V_{\mu,k}. \end{aligned}$$

For brevity, we denote these as f_1 and f_2, respectively. Substituting the approximation (7.2) and the actual value for $\Delta V_{\mu,k}$ as defined in (4.2b), f_2 is expressed as

$$\begin{aligned} f_2 &= V^i_k + \frac{\frac{u^i_k}{C^i_p}}{\frac{1}{N}\sum_{i=1}^N \frac{u^i_k}{C^i_p}} \cdot \frac{1}{N} \sum_{i=1}^N \left(\frac{T_s}{C^i_p} u^i_k - \frac{T_s}{\tau^i_p} V^i_k \right) \\ &= V^i_k + \frac{\frac{T_s}{C^i_p} u^i_k}{\sum_{i=1}^N \frac{u^i_k}{C^i_p}} \cdot \sum_{i=1}^N \left(\frac{u^i_k}{C^i_p} - \frac{V^i_k}{\tau^i_p} \right). \end{aligned}$$

To compute the error, f_2 is subtracted from f_1, and the absolute value of the result is obtained:

$$\begin{aligned} e &= |f_1 - f_2| \\ &= \left| -\frac{T_s}{\tau^i_p} V^i_k + \frac{T_s}{C^i_p} u^i_k \left(1 - \frac{\sum_{i=1}^N \left(\frac{u^i_k}{C^i_p} - \frac{V^i_k}{\tau^i_p} \right)}{\sum_{i=1}^N \frac{u^i_k}{C^i_p}} \right) \right|. \end{aligned}$$

This completes the proof. □

Referring to Fig. 7.1 and its tabulated values in Table 7.1, the conversion of computations involving Γ to an offline format visibly reduces the DAEKF's FLOPs required for the time update by 66%. To be specific, the offline computations remove $24N - 2$ FLOPs from the online computation of the time update. Thus, in the case of no

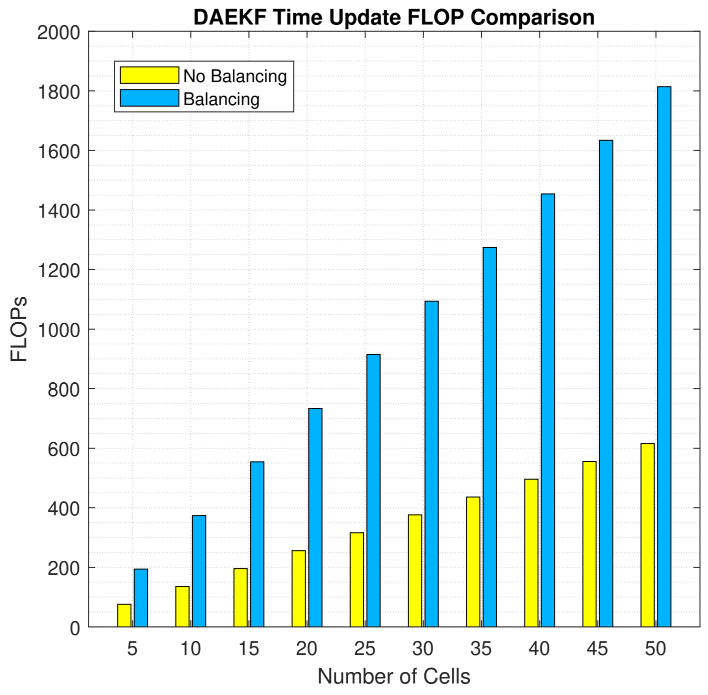

Figure 7.1: Comparison of DAEKF FLOP count between the cases of balancing currents and no balancing currents.

Table 7.1: DAEKF FLOPs for No Balancing and Balancing Time Updates

Cell Number	Not Balanced	Balanced
5	76	194
10	136	374
15	196	554
20	256	734
25	316	914
30	376	1094
35	436	1274
40	496	1454
45	556	1634
50	616	1814

balancing currents, the growth rate of the dense time update's FLOP count grows one-third as quickly as the case of balancing currents.

CHAPTER EIGHT

CONCLUSION

In this paper, the dense formulation of the extended Kalman filter, termed as DEKF, was introduced to address the computational overhead and intractable resource demands that hinder the sparse extended Kalman filter. The DEKF's framework was developed from the theoretical standpoint, its equivalence to the sparse formulation demonstrated, the adaptive step appended to the general algorithm (DAEKF), and its overall performance assessed from the perspective of resource consumption as well as the ability to estimate multiple cells' state simultaneously. Comparing FLOP count, the sparse method's FLOP count exhibited poor scalability insofar as its proportionality to the number of cells N cubed, whereas the proposed dense method proved its superiority with a FLOP count growing linearly with N. To this end, a slight optimization of the DAEKF was introduced in the scenario of no balancing currents, where the RFF matrix Γ and adjacent computations thereto can be performed offline. As for estimating performance, the DAEKF maintained good SOC estimation for not only the selected five and hundred-cell cases but over a plethora of cell numbers, where the average error in the predicted measurement as well as its standard deviation gradually decreased for larger cell numbers. Future work directions include: (1) validate the DAEKF's estimation ability through hardware experiment, (2) event-triggered methods which employ streamlined methods for slowly changing balancing currents, and (3) assessing the feasibility of a “dense" OCV-SOC curve approximation.

REFERENCES

[1] P. Ahmadi, “Environmental impacts and behavioral drivers of deep decarbonization for transportation through electric vehicles,” *Journal of Cleaner Production*, 2019.

[2] H. Hao, X. Cheng, Z. Liu, , and F. Zhao, “Electric vehicles for greenhouse gas reduction in china: A cost-effectiveness analysis,” *Transportation Research Part D*, 2017.

[3] X. Chen, W. Shen, T. Vo, Z. Cao, and A. Kapoor, “An overview of lithium-ion batteries for electric vehicles,” *IEEE*, pp. 230–235, 2012.

[4] H. Askari, A. Khajepour, M. B. Khamesee, and Z. L. Wang, “Embedded self-powered sensing systems for smart vehicles and intelligent transportation,” *Nano Energy*, vol. 66, p. 104103, 2019.

[5] M. Dendaluce Jahnke, F. Cosco, R. Novickis, J. Pérez Rastelli, and V. Gomez-Garay, “Efficient neural network implementations on parallel embedded platforms applied to real-time torque-vectoring optimization using predictions for multi-motor electric vehicles,” *Electronics*, vol. 8, no. 2, 2019.

[6] J. Chen, A. Behal, and C. Li, “Active battery cell balancing by real time model predictive control for extending electric vehicle driving range,” *IEEE Transactions on Automation Science and Engineering*. Accepted June 2023.

[7] M. Einhorn, W. Roessler, and J. Fleig, “Improved performance of serially connected li-ion batteries with active cell balancing in electric vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 6, pp. 2448–2457, 2011.

[8] J. Huang, D. Shi, and T. Chen, “Event-triggered state estimation with an energy harvesting sensor,” *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4768–4775, 2017.

[9] J. Chiasson and B. Vairamohan, “Estimating the state of charge of a battery,” *IEEE Transactions on Control Systems Technology*, vol. 13, pp. 465–470, April 2005.

[10] A. Pozzi, M. Zambelli, A. Ferrara, and D. M. Raimondo, “Balancing-aware charging strategy for series-connected lithium-ion cells: A nonlinear model predictive control approach,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1862–1877, 2020.

[11] F. S. Hoekstra, L. W. Ribelles, H. J. Bergveld, and M. Donkers, “Real-time range maximisation of electric vehicles through active cell balancing using model-predictive control,” in *2020 American Control Conference*, (Denver, CO), pp. 2219–2224, July 1–3, 2020.

[12] H. He, R. Xiong, and J. Fan, "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach," *Energies*, vol. 4, no. 4, pp. 582–598, 2011.

[13] L. Zhang, H. Peng, Z. Ning, Z. Mu, and C. Sun, "Comparative research on rc equivalent circuit models for lithium-ion batteries of electric vehicles," *Applied Sciences*, vol. 7, no. 10, 2017.

[14] X. Lai, Y. Zheng, and T. Sun, "A comparative study of different equivalent circuit models for estimating state-of-charge of lithium-ion batteries," *Electrochimica Acta*, vol. 259, pp. 566–577, 2018.

[15] S. J. Moura, F. B. Argomedo, R. Klein, A. Mirtabatabaei, and M. Krstic, "Battery state estimation for a single particle model with electrolyte dynamics," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 453–468, 2016.

[16] K. Movassagh, S. A. Raihan, and B. Balasingam, "Performance analysis of coulomb counting approach for state of charge estimation," in *2019 IEEE Electrical Power and Energy Conference (EPEC)*, pp. 1–6, 2019.

[17] G. Fathoni, S. A. Widayat, P. A. Topan, A. Jalil, A. I. Cahyadi, and O. Wahyunggoro, "Comparison of state-of-charge (soc) estimation performance based on three popular methods: Coulomb counting, open circuit voltage, and kalman filter," in *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, pp. 70–74, 2017.

[18] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[19] G. Y. Kulikov and M. V. Kulikova, "Accurate numerical implementation of the continuous-discrete extended kalman filter," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 273–279, 2013.

[20] H. R. Hashemipour, S. Roy, and A. J. Laub, "Decentralized structures for parallel kalman filtering," *IEEE Transactions on automatic control*, vol. 33, no. 1, pp. 88–94, 1988.

[21] A. Tsiamis and G. J. Pappas, "Online learning of the kalman filter with logarithmic regret," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2774–2789, 2022.

[22] S. Liu, Z. Wang, Y. Chen, and G. Wei, "Protocol-based unscented kalman filtering in the presence of stochastic uncertainties," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1303–1309, 2019.

[23] G. L. Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 1. background,” *Journal of Power Sources*, vol. 134, no. 2, pp. 252–261, 2004.

[24] R. Xiong, H. He, F. Sun, and K. Zhao, “Evaluation on state of charge estimation of batteries with adaptive extended kalman filter by experiment approach,” *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 108–117, 2013.

[25] G. L. Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 3. state and parameter estimation,” *Journal of Power Sources*, vol. 134, no. 2, pp. 277–292, 2004.

[26] D. Sun, X. Yu, C. Wang, C. Zhang, R. Huang, Q. Zhou, T. Amietszajew, and R. Bhagat, “State of charge estimation for lithium-ion battery based on an intelligent adaptive extended kalman filter with improved noise estimator,” *Energy*, vol. 214, p. 119025, 2021.

[27] H. He, R. Xiong, X. Zhang, F. Sun, and J. Fan, “State-of-charge estimation of the lithium-ion battery using an adaptive extended kalman filter based on an improved thevenin model,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 4, pp. 1461–1469, 2011.

[28] O. C. Imer and T. Basar, “Optimal estimation with limited measurements,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1029–1034, IEEE, 2005.

[29] A. Valade, P. Acco, P. Grabolosa, and J.-Y. Fourniols, “A study about kalman filters applied to embedded sensors,” *Sensors*, vol. 17, no. 12, 2017.

[30] P. Closas, J. Vilà-Valls, and C. Fernández-Prades, “Computational complexity reduction techniques for quadrature kalman filters,” in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 485–488, 2015.

[31] D. Simon, “Kalman filtering,” *Embedded systems programming*, vol. 14, no. 6, pp. 72–79, 2001.

[32] M. Raitoharju and R. Piché, “On computational complexity reduction methods for kalman filter extensions,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 10, pp. 2–19, 2019.

[33] Z. Deng, X. Hu, X. Lin, Y. Che, L. Xu, and W. Guo, “Data-driven state of charge estimation for lithium-ion battery packs based on gaussian process regression,” *Energy*, vol. 205, p. 118000, 2020.

[34] L. Song, K. Zhang, T. Liang, X. Han, and Y. Zhang, "Intelligent state of health estimation for lithium-ion battery pack based on big data analysis," *Journal of Energy Storage*, vol. 32, p. 101836, 2020.

[35] D. Zhang, L. D. Couto, P. S. Gill, S. Benjamin, W. Zeng, and S. J. Moura, "Thermal-enhanced adaptive interval estimation in battery packs with heterogeneous cells," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 3, pp. 1102–1115, 2022.

[36] Z. Pei, X. Zhao, H. Yuan, Z. Peng, and L. Wu, "An equivalent circuit model for lithium battery of electric vehicle considering self-healing characteristic," *Journal of Control Science and Engineering*, vol. 2018, 2018.

[37] S. S. Madani, E. Schaltz, and S. Knudsen Kær, "An electrical equivalent circuit model of a lithium titanate oxide battery," *Batteries*, vol. 5, no. 1, p. 31, 2019.

[38] J. Wehbe and N. Karami, "Battery equivalent circuits and brief summary of components value determination of lithium ion: A review," in *2015 Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, (Beirut, Lebanon), pp. 45–49, 2015.

[39] H. He, X. Zhang, R. Xiong, Y. Xu, and H. Guo, "Online model-based estimation of state-of-charge and open-circuit voltage of lithium-ion batteries in electric vehicles," *Energy*, vol. 39, no. 1, pp. 310–318, 2012. Sustainable Energy and Environmental Protection 2010.

[40] A. Charnes, M. J. Kirby, and R. A. C. M. VA, "Properties of a generalized inverse with applications to linear programming theory," *McLean, Va., Research Analysis Corporation Technical Paper Number RAC-TP-171, August*, 1965.

[41] C. T. Fraser and S. Ulrich, "Adaptive extended kalman filtering strategies for spacecraft formation relative navigation," *Acta Astronautica*, vol. 178, pp. 700–721, 2021.

[42] J. N. Yang, S. Lin, H. Huang, and L. Zhou, "An adaptive extended kalman filter for structural damage identification," *Structural Control and Health Monitoring: The Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures*, vol. 13, no. 4, pp. 849–867, 2006.

[43] X. Lin, H. E. Perez, J. B. Siegel, A. G. Stefanopoulou, Y. Li, R. D. Anderson, Y. Ding, and M. P. Castanier, "Online parameterization of lumped thermal dynamics in cylindrical lithium ion batteries for core temperature estimation and health monitoring," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1745–1755, 2012.

[44] X. Lin, H. Fu, H. E. Perez, J. B. Siege, A. G. Stefanopoulou, Y. Ding, and M. P. Castanier, "Parameterization and observability analysis of scalable battery clusters for onboard thermal management," *Oil & Gas Science and Technology–Revue d'IFP Energies nouvelles*, vol. 68, no. 1, pp. 165–178, 2013.

[45] H. E. Perez, J. B. Siegel, X. Lin, A. G. Stefanopoulou, Y. Ding, and M. P. Castanier, "Parameterization and validation of an integrated electro-thermal cylindrical lfp battery model," in *Dynamic Systems and Control Conference*, vol. 45318, pp. 41–50, American Society of Mechanical Engineers, 2012.