Simultaneous Cell State Estimation via Dense Adaptive Extended Kalman Filter

Luke Nuculaj and Jun Chen, *Senior Member, IEEE*

***Abstract*—This work addresses the computational intractability apropos of extended Kalman filters (EKFs) in the context of battery cell state estimation under limited voltage measurement. A novel, compact variation of the Kalman filter, namely the "dense EKF" (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 whose size does not vary with the number of cells in the pack. The computational savings in terms of floating-point operations (FLOPs) reduction are 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 $\mathcal{O}(N^3)$ to $\mathcal{O}(N)$, 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, that is, no balancing or leakage, is also studied and demonstrated.**

***Index Terms*—Adaptive extended Kalman filter (EKF), battery cell state estimation, complexity, limited sensor measurement.**

Nomenclature

DAEKF	Dense adaptive extended Kalman filter.
DEKF	Dense extended Kalman filter.
ECM	Equivalent circuit model.
EKF	Extended Kalman filter.
EV	Electric vehicle.
FLOP	Floating-point operation.
OCV	Open-circuit voltage.
RFF	Relative fitness factor.
RMSE	Root mean square error.
SOC	State of charge.

I. Introduction

With the growing popularity of 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], [4], [5]. Due to the inevitability of manufacturing variations, battery cells exhibit voltage and SOC imbalances with one another which, in turn, curtail both battery life and performance, ultimately reducing an EV's range [6], [7]. To combat this, nondissipative cell-balancing techniques are frequently employed in conjunction with advanced control techniques—model predictive control is among the most commonly explored and appealing control techniques by virtue of its ability to account for system constraints [8], [9]. 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 [10], [11]. 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 [12], [13]. On the other hand, the EKF [14], [15], [16], which unionizes a preconceived 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], [18]. For example, [17] explores the adaptive EKF, which employs a covariance matching approach for quick, yet reliable online estimation, yielding a maximum SOC estimation error of less than 2%. Sun et al. [19] built upon the AEKF, introducing an intelligent AEKF, which monitors the changes in the fixed-length error innovation sequence's (EIS) distribution and updates the innovation covariance matrices accordingly. In comparison to the AEKF, this method sees the decrease of the estimator's root-mean-square 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 [20]), multicell SOC estimation instead of single-cell, variants of the EKF suffer immensely from computational latency and hefty memory requirements [21], [22]. For purposes of multicell 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. While there exists some utility in heuristic, data-driven methods of pack SOC estimation [23], [24], their "closed-box" nature is not desirable in the context of deterministic estimation meth-

Received 21 February 2025; accepted 14 May 2025. Date of publication 4 June 2025; date of current version 23 October 2025. This work was supported in part by the Oakland University School of Engineering and Computer Science (OU SECS) Faculty Start-Up Fund and in part by the National Science Foundation under Award 2237317. Recommended by Associate Editor S. Evangelou. *(Corresponding author: Jun Chen.)*

The authors are with the Department of Electrical and Computer Engineering, Oakland University, Rochester, MI 48309 USA (e-mail: lukenuculaj@oakland.edu; junchen@oakland.edu).

Digital Object Identifier 10.1109/TCST.2025.3571557

ods. While the research on deterministic pack SOC estimation is scarce at best, [25] 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 relationship to each cell is restrictive, despite its favorable CPU time with respect to cell number.

The work presented in the current manuscript further explores the EKF 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 EKF (DEKF) is proposed. This method of cell SOC estimation reimagines the entire pack as a single "average" cell and performs state estimation over the said average cell, whose dimensions are constant regardless of the number of cells in the string. Furthermore, an 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. As will be shown later, the RFF 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 [25], 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 EKF 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 accurately 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 VII for more details).

Note that though [25] considers a similar setting in that only one voltage sensor is used, only interval estimation is performed in [25]. The proposed DEKF in this article, on the other hand, estimates the state for each cell. The contribution of this work is summarized as follows.

1) We introduce a notion termed RFF that characterizes how each cell's state would change with respect to that of an average cell.
2) A new variant of EKF, termed DEKF, is proposed to simultaneously estimate the SOC of each cell with only one voltage sensor (measuring the terminal voltage), which can significantly reduce the computational complexity (from cubic order to linear order).
3) The estimation performance of DEKF is analytically studied. In particular, it is proved that the measurement update step will not incur any error, while a theoretical error bound of time update is provided.
4) Extensive simulations are conducted to analyze the performance of the proposed DEKF, together with sensitive analysis to illustrate the robustness of the proposed approach under cell degradation.

The rest of the article is organized as follows. Section II contains an explanation of all of the relevant notation and nomenclature used throughout the article. Section III formulates the problem of cell SOC estimation in a serial-connected string, while Section IV presents the traditional sparse EKF for cell SOC estimation. Section V introduces the concept of RFFs, which forms the basis for the DEKF. Section VI derives the DEKF and establishes its equivalence to sparse formulation, while Section VII provides the main simulation results. Section VIII discusses a special case with homogeneous cell current, and Section IX concludes the article with a summary of our findings and a discussion of future work. The appendix contains several lemmas used throughout the article.

II. Notation and Nomenclature

Let $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, and $\mathbb{N}$ denote the field of real numbers, the set of real column vectors of length n, the set of m-by-n real matrices, and the set of nonnegative integers, respectively. The transpose of a matrix $A \in \mathbb{R}^{m\times n}$ is denoted by $A^\top$. The Moore-Penrose pseudoinverse of A is denoted by $A^\dagger := (A^\top A)^{-1}A^\top$. A symmetric matrix $B \in \mathbb{R}^{n\times n}$ ($B = B^\top$) is said to be positive definite if and only if $x^\top Bx > 0$ for all $x \in \mathbb{R}^n\backslash\{0\}$ and is denoted by $B \succ 0$. Similarly, B is said to be positive semidefinite if and only if $x^\top Bx \geq 0$ for all $x \in R^n$, denoted by $\succeq 0$. For a vector $x \in \mathbb{R}^n$, the Euclidean norm $\|x\|_2 := (x(1)^2 + x(2)^2 + \ldots + x(n)^2)^{1/2}$ and the infinity norm $\|x\|_\infty := \max_i |x(i)|$. For a matrix $A \in \mathbb{R}^{m\times n}$, its induced infinity-norm $\|A\|_\infty := \max_{1\leq i\leq m} \sum_{j=1}^{n} |A(i, j)|$, following from the standard definition for induced p-norms $\|A\|_p = \sup_{x\neq 0}(\|Ax\|_p/\|x\|_p)$. Any variable name accompanied by superscript $(\cdot)^i$ indicates that is of or belongs to the ith cell in the battery pack. For $k \in \mathbb{N}$, any variable name accompanied by a subscript $(\cdot)_k$ indicates that it is the variable as it exists at the kth discrete time step and a subscript $(\cdot)_\mu$ indicates that it is a "dense" variable belonging to the DEKF (more details provided in Section VI). Combinations of subscripts and superscripts may occur, but this does not impact the aforementioned meaning of the individual terms that appear. Unless otherwise specified, vector $\hat{x}$ with the hat notation indicates that it is an estimate of x; superscripts $\hat{x}^-$ and $\hat{x}^+$ indicate prior and posterior estimates, respectively. Unless specified otherwise, elements of a vector x are indexed via $x(j)$, where $j = 1$ points to the first element in x. For convenience, a tabulated list of the acronyms used throughout this article can be found in Nomenclature.

III. Cell State Estimation Problem

A. Battery Model

Consider a serial-connected battery with N cells, as shown in Fig. 1. The highly nonlinear chemical processes that occur within battery cells are difficult to precisely model. Instead, a first-order ECM is adopted as a suitable representation of a Li-ion battery's system dynamics [26], [27] (see Fig. 2),

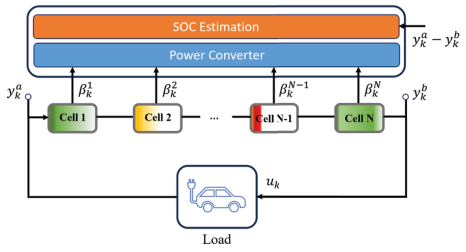

Fig. 1. Structure of serial-connected battery cells with balancing currents and pack terminal voltage measurement ($y_k^a - y_k^b$).

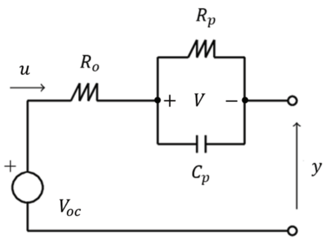

Fig. 2. ECM of a battery cell.

where the OCV (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

$$\dot{s}^i = -\eta^i \frac{u^i}{3600C^i} \tag{1a}$$

$$\dot{V}^i = -\frac{V^i}{R_p^i C_p^i} + \frac{u^i}{C_p^i} \tag{1b}$$

$$y^i = V_{\text{oc}}^i - V^i - u^i R_o^i \tag{1c}$$

where s^i is the ith cell's SOC, V^i is the relaxation voltage, η^i is the Coulombic efficiency, and C^i is the cell capacity with the 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.

Forward Euler method can be used to discretize (1) with a sampling time T_s as follows:

$$s_{k+1}^i = s_k^i - \eta^i \frac{T_s}{3600C^i} u_k^i \tag{2a}$$

$$V_{k+1}^i = \left(1 - \frac{T_s}{R_p^i C_p^i}\right) V_k^i + \frac{T_s}{C_p^i} u_k^i \tag{2b}$$

$$y_k^i = V_{\text{oc},k}^i - V_k^i - u_k^i R_o^i. \tag{2c}$$

Denoting $x^i := \begin{bmatrix} s^i & V^i \end{bmatrix}^\top$, (2a) and (2b) can be compactly represented as follows:

$$x_{k+1}^i = A^i x_k^i + B^i u_k^i \tag{3}$$

where

$$A^i = \begin{bmatrix} 1 & 0 \\ 0 & 1 - \dfrac{T_s}{R_p^i C_p^i} \end{bmatrix} \quad B^i = \begin{bmatrix} -\eta^i \dfrac{T_s}{3600C^i} \\ \dfrac{T_s}{C_p^i} \end{bmatrix}. \tag{4}$$

Note that the ECM's electrical parameters typically vary as a function of SOC [10], [17], [27], making (3) nonlinear. However, to simplify the notation in this work, we consider R_o, R_p, and C_p as constant values that do not vary with SOC, thus making the state (3) linear. A sensitivity analysis for these parameters and the accompanying discussion can be found in Section VII-B. Moreover, the measurement function (2c) remains nonlinear, since the OCV V_{oc} is nonlinear with respect to SOC. In this work, we follow a common approach in the literature by approximating the OCV-SOC behavior with a polynomial [28], [29] (see Section VII-A for details).

Leveraging (4), the collection of state update equations across all N cells

$$X_{k+1} := \begin{bmatrix} x_{k+1}^1 \\ x_{k+1}^2 \\ \vdots \\ x_{k+1}^N \end{bmatrix} = \begin{bmatrix} A^1 x_k^1 + B^1 u_k^1 \\ A^2 x_k^2 + B^2 u_k^2 \\ \vdots \\ A^N x_k^N + B^N u_k^N \end{bmatrix} \tag{5}$$

and define the sparse state vector as $X_k = \begin{bmatrix} x_k^1 & x_k^2 & \dots & x_k^N \end{bmatrix}^\top$, which is the state vector for the entire battery pack at time step k. Rewriting (5) gives

$$X_{k+1} = AX_k + BU_k \tag{6}$$

where

$$A = \begin{bmatrix} A^1 & 0 & \dots & 0 \\ 0 & A^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A^N \end{bmatrix} \tag{7}$$

$$B = \begin{bmatrix} B^1 & 0 & \dots & 0 \\ 0 & B^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B^N \end{bmatrix} \tag{8}$$

are block-diagonal matrices consisting of (4) for each cell, and cell current matrix U_k is defined as $\begin{bmatrix} u_k^1 \; u_k^2 \dots u_k^N \end{bmatrix}^\top$. The measurement function, in the case of pack-level dynamics, is a scalar quantity representative of the pack terminal voltage defined as follows:

$$y_k = \sum_{i=1}^{N} \left(V_{\text{oc},k}^i - V_k^i - u_k^i R_o^i\right) := h(X_k, U_k). \tag{9}$$

Example 1: Consider a battery with $N = 5$ serial-connected cells, with cell parameters listed in Table I and initial states listed in Table II. Moreover, $T_s = 10$, and all cell currents are equal to 4.6 A. Fig. 3 plots the SOC, relaxation voltage, and terminal voltage for each cell. As can be seen, due to the heterogeneous cell parameters (see Table I), the cells' SOC significantly differ from each other, making cell-level state estimation a challenging task, especially under limited sensor capability.

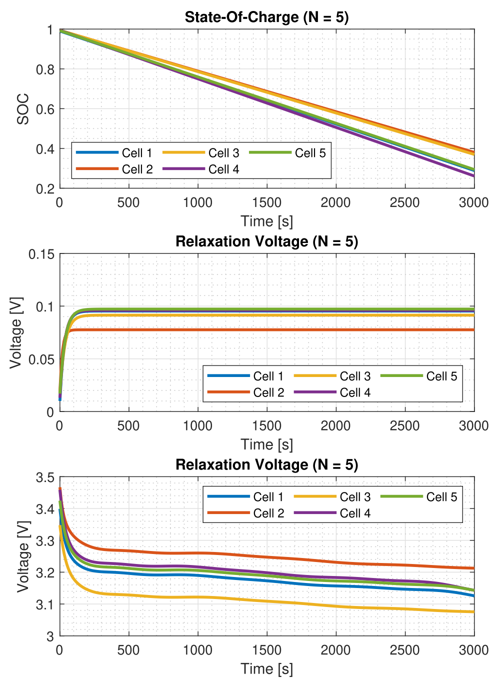

Fig. 3. Plots of SOC, relaxation voltage, and terminal voltage for five cells.

TABLE I
NOMINAL CIRCUIT PARAMETERS

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

TABLE II
INITIAL STATES

State	Unit	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
s_0	[-]	0.990	0.993	0.994	0.994	0.992
V_0	[V]	0.010	0.019	0.017	0.013	0.017

B. Problem Formulation

Given the battery dynamic model (6), the primary objective of this article 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 article is described below.

Problem 1: Given battery dynamic model (6) and output (9), 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, that is, only the pack terminal voltage is measured. This setting is similar to [25], which also assumes only the pack terminal voltage measurement is available. However, our work is different from [25] in that only interval estimation is performed in [25], 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 in which multiple cell terminal voltages are also measured and the corresponding optimal sensor configuration problem.

IV. SPARSE EKF

To solve Problem 1, the 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 article—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 $O(N)$.

A. Sparse Prediction Model and Time Update

Because of the inherent deviations of real-life systems from mathematical models, (6) and (9) take on the form

$$\begin{aligned} X_{k+1} &= AX_k + BU_k + W_k \\ y_k &= h(X_k, U_k) + v_k \end{aligned}$$

where $W_k := \begin{bmatrix} w_k^1 & w_k^2 & \dots & w_k^N \end{bmatrix}^\top$, w_k^i $(\sim \mathcal{N}(0, Q_k^i))$ is the process noise of 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 (7) and (8), the complete time-update for the sparse EKF is the following:

$$\hat{X}_{k+1}^- = A\hat{X}_k^+ + BU_k \tag{10a}$$

$$P_{k+1}^- = AP_k^+A^\top + Q_k \tag{10b}$$

where (10a) is the state update and (10b) is the covariance update. The initial sparse process covariance P_0^+ and sparse process noise covariance Q_k are block-diagonal matrices defined as follows:

$$P_0^+ = \begin{bmatrix} P_0^{1,+} & 0 & \dots & 0 \\ 0 & P_0^{2,+} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_0^{N,+} \end{bmatrix} \tag{11}$$

$$Q_k = \begin{bmatrix} Q_k^1 & 0 & \dots & 0 \\ 0 & Q_k^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q_k^N \end{bmatrix} \tag{12}$$

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 (10a) and (10b) rely on real-time computations over matrices (11) and (12), 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 follows:

$$\hat{y} = h\left(\hat{X}_{k+1}^-, U_k\right) \tag{13}$$

where h is defined in (9).

B. Sparse Kalman Gain and Measurement Update

Recall that the OCV V_{oc} is a nonlinear function of the SOC, making h in (9) nonlinear. Therefore, to compute the Kalman gain, we must obtain Jacobian H by linearizing (13)

$$\begin{aligned} H_{k+1} = \partial_X h(X,U)\Big|_{(\hat{X}^-_{k+1},U_k)} &= \partial_X \Bigg[\sum_{i=1}^{N}\Big(V^i_{\text{oc}}(X(i,1)) \\ &\quad - R^i_o U(i) - X(i,2)\Big)\Bigg]\Bigg|_{(\hat{X}^-_{k+1},U_k)}. \end{aligned} \tag{14}$$

Here, $U_k(i)$ 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 (14), we have

$$H_{k+1} = \left[\,\frac{\partial V^1_{\text{oc}}}{\partial X(i,1)}\;-1\;\cdots\;\frac{\partial V^N_{\text{oc}}}{\partial x^N(1)}\;-1\,\right]\Bigg|_{(\hat{X}^-_{k+1},U_k)}. \tag{15}$$

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

$$K_{k+1} = \frac{P^-_{k+1}H^\top_{k+1}}{H_{k+1}P^-_{k+1}H^\top_{k+1} + R_k} \tag{16a}$$

$$\hat{X}^+_{k+1} = \hat{X}^-_{k+1} + K_{k+1}\left(y_{k+1} - h\left(\hat{X}^-_{k+1},U_k\right)\right) \tag{16b}$$

$$P^+_{k+1} = \left(I - K_{k+1}H_{k+1}\right)P^-_{k+1}. \tag{16c}$$

Recall from Section IV-A that R_k is the measurement noise covariance such that $v_k \sim \mathcal{N}(0,R_k)$. Note that for our case, we are limited to a single-voltage sensor for measurement. Hence, $R_k \in \mathbb{R}$, so the fraction notation for Kalman gain equations akin to (16a) is used throughout this article.

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 $16N^3 + 32N^2 + (4P+14)N + 6M$, 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 VI and VII for more details).

Such a high complexity makes the sparse EKF approach unsuitable for real-time implementation, especially in embedded environments. To address these concerns, Sections V and VI, a 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 RFFs—in Section V.

V. Relative Fitness Factors

A. Derivation of Individual RFF

The driving paradigm of the DEKF 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 into how every SOC in the pack changes with respect to the "average" cell over time. To determine this change, begin by averaging (2a) and (2b) over all N cells to obtain

$$\frac{1}{N}\sum_{i=1}^{N} s^i_{k+1} = \frac{1}{N}\sum_{i=1}^{N} s^i_k - \frac{1}{N}\sum_{i=1}^{N}\frac{\eta^i T_s}{3600C^i}u^i_k \tag{17a}$$

$$\frac{1}{N}\sum_{i=1}^{N} V^i_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{T_s}{\tau^i_p}\right)V^i_k + \frac{1}{N}\sum_{i=1}^{N}\frac{T_s}{C^i_p}u^i_k \tag{17b}$$

where time constant $\tau^i_p = R^i_p C^i_p$. Denote $s_{\mu,k} := (1/N)\sum_{i=1}^{N} s^i_k$ and $V_{\mu,k} := (1/N)\sum_{i=1}^{N} V^i_k$ as the average SOC at time step k and average V at time step k, respectively. From (17), the changes in $s_{\mu,k}$ and $V_{\mu,k}$ are

$$\Delta s_{\mu,k} = s_{\mu,k+1} - s_{\mu,k} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\eta^i T_s}{3600C^i}u^i_k \tag{18a}$$

$$\Delta V_{\mu,k} = V_{\mu,k+1} - V_{\mu,k} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{T_s}{C^i_p}u^i_k - \frac{T_s}{\tau^i_p}V^i_k\right). \tag{18b}$$

Without loss of generality, we will now step through how to quantify the degree to which s^i_k changes with respect to $s_{\mu,k}$. Retrieving the control input term from (2a) yields

$$\gamma^i_{s,k} = \frac{\Delta s^i_k}{\Delta s_{\mu,k}} = \frac{-\frac{\eta^i T_s}{3600C^i}u^i_k}{-\frac{1}{N}\sum_{i=1}^{N}\frac{\eta^i T_s}{3600C^i}u^i_k} = \frac{\frac{\eta^i}{C^i}u^i_k}{\frac{1}{N}\sum_{i=1}^{N}\frac{\eta^i}{C^i}u^i_k} \tag{19}$$

where $\gamma^i_{s,k}$ is the ith cell's SOC RFF at time step k. From this methodology, $\gamma^i_{V,k}$ naturally follows:

$$\gamma^i_{V,k} = \frac{\frac{u^i_k}{C^i_p} - \frac{V^i_k}{\tau^i_p}}{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{u^i_k}{C^i_p} - \frac{V^i_k}{\tau^i_p}\right)}. \tag{20}$$

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^i_k and V^i_k evolve with respect to average states $s_{\mu,k}$ and $V_{\mu,k}$. In other words, we can recover the change in states as

$$s^i_{k+1} = s^i_k + \gamma^i_{s,k}\Delta s_{\mu,k} \tag{21a}$$

$$V^i_{k+1} = V^i_k + \gamma^i_{V,k}\Delta V_{\mu,k}. \tag{21b}$$

The following theorem establishes the utility of the RFFs in computing each cell's change in state by proving the equivalence of (21) to the existing state dynamic (2a) and (2b). The proof of Theorem 1 is provided in the Appendix.

Theorem 1: The SOC update (21a) is equivalent to (2a), and the relaxation voltage update (21b) is equivalent to (2b).

Theorem 1 established that the RFF can be used to quantify how the ith state vector x^i_k changes with respect to the state vector $x_{\mu,k}$ of an "averaging" model, defined as $x_{\mu,k} := \left[s_{\mu,k}\ V_{\mu,k}\right]^\top$. To express this mathematically, the Jacobian $\Gamma^i_k = \partial x^i_k/\partial x_{\mu,k}$ can be defined as follows:

$$\Gamma^i_k = \frac{\partial x^i_k}{\partial x_{\mu,k}} = \begin{bmatrix} \frac{\partial x^i_k(1)}{\partial \mu_{\mu,k}(1)} & \frac{\partial x^i_k(1)}{\partial x_{\mu,k}(2)} \\ \frac{\partial x^i_k(2)}{\partial x_{\mu,k}(1)} & \frac{\partial x^i_k(2)}{\partial x_{\mu,k}(2)} \end{bmatrix} = \begin{bmatrix} \gamma^i_{s,k} & 0 \\ 0 & \gamma^i_{V,k} \end{bmatrix}. \tag{22}$$

TABLE III
INITIAL CONDITIONS

Metric	Unit	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
V_k^i	[V]	0.80	0.85	0.90	0.95	1.00
u_k^i	[A]	1.6	3.6	4.6	5.6	7.6

TABLE IV
RELATIVE FITNESS FACTORS

$\gamma_{V,k}^1$	$\gamma_{V,k}^2$	$\gamma_{V,k}^3$	$\gamma_{V,k}^4$	$\gamma_{V,k}^5$
0.8664	1.4958	0.8311	0.9370	0.8696

Then, (21) can be compactly represented as follows:

$$x_{k+1}^i = x_k^i + \Gamma_k^i \Delta x_{\mu,k}. \tag{23}$$

Example 2: Consider the five-cell serial-connected battery as discussed in Example 1. Consider $T_s = 0.1$, the initial cell currents and relaxation voltages as they appear in Table III, and cell parameters as they appear in Table I. The denominator term in (20) can be computed as follows:

$$\begin{aligned}
&\frac{1}{5}\sum_{i=1}^{5}\left(\frac{u_k^i}{C_p^i} - \frac{V_k^i}{\tau_p^i}\right)\\
&= \frac{1}{5}\left(\frac{1.6}{1.874\cdot 10^3} - \frac{0.8}{(1.874\cdot 10^3)(2.072\cdot 10^{-2})}\right.\\
&\quad + \frac{3.6}{1.373\cdot 10^3} - \frac{0.85}{(1.373\cdot 10^3)(1.686\cdot 10^{-2})}\\
&\quad + \frac{4.6}{2.148\cdot 10^3} - \frac{0.9}{(2.148\cdot 10^3)(1.987\cdot 10^{-2})}\\
&\quad + \frac{5.6}{1.870\cdot 10^3} - \frac{0.95}{(1.870\cdot 10^3)(2.086\cdot 10^{-2})}\\
&\quad \left. + \frac{7.6}{2.004\cdot 10^3} - \frac{1}{(2.004\cdot 10^3)(2.113\cdot 10^{-2})}\right)\\
&= -2.279\cdot 10^{-2}.
\end{aligned}$$

Then, each cell's voltage RFF is computed as follows:

$$\begin{aligned}
\gamma_{V,k}^1 &= \frac{-1.975\cdot 10^{-2}}{-2.279\cdot 10^{-2}} = 0.8664\\
\gamma_{V,k}^2 &= \frac{-3.410\cdot 10^{-2}}{-2.279\cdot 10^{-2}} = 1.4958\\
\gamma_{V,k}^3 &= \frac{-1.895\cdot 10^{-2}}{-2.279\cdot 10^{-2}} = 0.8311\\
\gamma_{V,k}^4 &= \frac{-2.136\cdot 10^{-2}}{-2.279\cdot 10^{-2}} = 0.9370\\
\gamma_{V,k}^5 &= \frac{-1.982\cdot 10^{-2}}{-2.279\cdot 10^{-2}} = 0.8696.
\end{aligned}$$

For easy reference, the computed voltage RFF for each cell is listed in Table IV. Given u_k^i for each cell and T_s, the change in average relaxation voltage for this time step can be found to be $-2.279\cdot 10^{-3}$. Using the computed RFFs listed in Table IV, the change in each cell's relaxation voltage can be calculated

$$\begin{aligned}
\Delta V_k^1 &= \gamma_{V,k}^1 \Delta V_{\mu,k} = -1.975\cdot 10^{-3}\\
\Delta V_k^2 &= \gamma_{V,k}^2 \Delta V_{\mu,k} = -3.410\cdot 10^{-3}\\
\Delta V_k^3 &= \gamma_{V,k}^3 \Delta V_{\mu,k} = -1.895\cdot 10^{-3}\\
\Delta V_k^4 &= \gamma_{V,k}^4 \Delta V_{\mu,k} = -2.136\cdot 10^{-3}\\
\Delta V_k^5 &= \gamma_{V,k}^5 \Delta V_{\mu,k} = -1.982\cdot 10^{-3}.
\end{aligned}$$

Now, to show that each cell's change in relaxation voltage derived from the average indeed matches that which the individual cell dynamics produce, the same quantities are computed based on (2b), as follows:

$$\begin{aligned}
\Delta V_k^1 &= \frac{0.1}{1.874\cdot 10^3}\cdot 1.6 - \frac{0.1}{(1.874\cdot 10^3)(2.072\cdot 10^{-2})}\cdot 0.8\\
&= -1.975\cdot 10^{-3}\\
\Delta V_k^2 &= \frac{0.1}{1.373\cdot 10^3}\cdot 3.6 - \frac{0.1}{(1.373\cdot 10^3)(1.686\cdot 10^{-2})}\cdot 0.85\\
&= -3.410\cdot 10^{-3}\\
\Delta V_k^3 &= \frac{0.1}{2.148\cdot 10^3}\cdot 4.6 - \frac{0.1}{(2.148\cdot 10^3)(1.987\cdot 10^{-2})}\cdot 0.9\\
&= -1.895\cdot 10^{-3}\\
\Delta V_k^4 &= \frac{0.1}{1.870\cdot 10^3}\cdot 5.6 - \frac{0.1}{(1.870\cdot 10^3)(2.086\cdot 10^{-2})}\cdot 0.95\\
&= -2.136\cdot 10^{-3}\\
\Delta V_k^5 &= \frac{0.1}{2.004\cdot 10^3}\cdot 7.6 - \frac{0.1}{(2.004\cdot 10^3)(2.113\cdot 10^{-2})}\cdot 1\\
&= -1.982\cdot 10^{-3}.
\end{aligned}$$

To relate the RFF to the Kalman filter algorithm, we recognize (2a) and (2b) as being the time update equations whose generalization over N cells is expressed as (10a). Writing (2a) and (2b) as the time update for an individual cell gives

$$\begin{aligned}
\hat{s}_{k+1}^{i,-} &= \hat{s}_k^{i,+} - \eta^i \frac{T_s}{3600C^i} u_k^i\\
\hat{V}_{k+1}^{i,-} &= \left(1 - \frac{T_s}{R_p^i C_p^i}\right)\hat{V}_k^{i,+} + \frac{T_s}{C_p^i} u_k^i.
\end{aligned}$$

Consequently, (19) and (20) are defined as follows:

$$\hat{\gamma}_{s,k}^i := \frac{\frac{\eta^i}{C^i} u_k^i}{\frac{1}{N}\sum_{i=1}^N \frac{\eta^i}{C^i} u_k^i} \tag{24a}$$

$$\hat{\gamma}_{V,k}^i := \frac{\frac{u_k^i}{C_p^i} - \frac{\hat{V}_k^{i,+}}{\tau_p^i}}{\frac{1}{N}\sum_{i=1}^N \left(\frac{u_k^i}{C_p^i} - \frac{\hat{V}_k^{i,+}}{\tau_p^i}\right)} \tag{24b}$$

which will be useful later in Section VI and proofs thereof. Note that (19) and (24a) are identical, as they are both independent of SOC.

B. 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 IV and compute the pack-level Jacobian

$$\Gamma_k = \frac{\partial X_k}{\partial x_{\mu,k}} = \begin{bmatrix}\Gamma_k^1 & \Gamma_k^2 & \dots & \Gamma_k^N\end{bmatrix}^\top \tag{25}$$

where Γ_k (termed as "RFF matrix") is of size $2N \times 2$. Extrapolating (23) to the entire pack

$$X_{k+1} = X_k + \Gamma_k \Delta x_{\mu,k} \tag{26}$$

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 the proposed DEKF by developing an initial framework and concurrently proving theoretical equivalence to the sparse method outlined in Section IV. The next two lemmas concern the left pseudoinverse of Γ, which will be utilized in Section VI and whose proofs are provided in the Appendix.

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

$$\Gamma^\dagger = \begin{bmatrix} \frac{\gamma_s^1}{\sum_{i=1}^N \left(\gamma_s^i\right)^2} & 0 & \dots & \frac{\gamma_s^N}{\sum_{i=1}^N \left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\gamma_V^1}{\sum_{i=1}^N (\gamma_V^i)^2} & \dots & 0 & \frac{\gamma_V^N}{\sum_{i=1}^N (\gamma_V^i)^2} \end{bmatrix}. \tag{27}$$

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

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

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

While Lemma 1 and Remark 3 prove that $\Gamma^\dagger\Gamma = I_{2\times 2}$, $\Gamma\Gamma^\dagger$ is generally not an identity matrix. Lemma 2 addresses this concern by proving 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.

VI. Dense EKF

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

$$x_{\mu,k+1} = A_\mu x_{\mu,k} + B_\mu U_k + w_{\mu,k} \tag{28a}$$
$$y_{\mu,k} = h_\mu(X_k, U_k) + v_{\mu,k} \tag{28b}$$

where $x_{\mu,k} := \left[s_{\mu,k}\ V_{\mu,k}\right]^\top$, $A_\mu = \hat{\Gamma}_k^\dagger A \hat{\Gamma}_k \in \mathbb{R}^{2\times 2}$, $B_\mu = \hat{\Gamma}_k^\dagger B \in \mathbb{R}^{2\times N}$, $h_\mu(\hat{X}_{k+1}^-, U_k) = (1/N)h(\hat{X}_{k+1}^-, U_k)$, $w_{\mu,k} \sim \mathcal{N}(0, \hat{\Gamma}_k^\dagger Q_k \hat{\Gamma}_k)$, and $v_{\mu,k} \sim \mathcal{N}(0, (1/N^2)R_k)$. For reasons discussed in Section V-A and illustrated by (24), we have

$$\hat{\Gamma}_k = \left.\frac{\partial X}{\partial x_\mu}\right|_{X=\hat{X}_k^+}. \tag{29}$$

We claim that, for the problem at hand (see Problem 1), a dense model (28) over x_μ exists, and it is enough to show that an alternative Kalman filtering method (the DEKF) capable of estimating over this model can be derived from the existing Kalman filter (10) and (16). Sections VI-A and VI-B present the derivations and, where necessary, establish upper bounds on the resulting errors.

A. Dense Time Update

Given the dense model (28), its time update equations are given by

$$\hat{x}_{\mu,k+1}^- = A_\mu \hat{x}_{\mu,k}^+ + B_\mu U_k \tag{30a}$$
$$P_{\mu,k+1}^- = A_\mu P_{\mu,k}^+ A_\mu^\top + Q_{\mu,k}. \tag{30b}$$

The estimate over the dense state vector is then "distributed" to each cell using (26)

$$\hat{X}_{k+1}^- = \hat{X}_k^+ + \hat{\Gamma}_k \left(\hat{x}_{\mu,k+1}^- - \hat{x}_{\mu,k}^+\right) \tag{31a}$$
$$P_{k+1}^- = \hat{\Gamma}_k P_{\mu,k+1}^- \hat{\Gamma}_k^\top. \tag{31b}$$

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

Theorem 2: Under the assumption that $T_s \ll \tau_p^i$, the time update on $\hat{X}_{k+1}^-$ as computed by (30a) and (31a) is almost equivalent to the sparse time update as computed by (10a) with an error bound listed in (38), and P_{k+1}^- as computed by (30b) and (31b) is equivalent to the sparse time update as computed by (10b).

Proof: We begin with (10a). First, solve the matrix-differential (29) for sparse vector X_k in terms of $x_{\mu,k}$:

$$\hat{\Gamma}_k = \left.\frac{\partial X_k}{\partial x_{\mu,k}}\right|_{X=\hat{X}_k^+}$$
$$\begin{bmatrix}\hat{\Gamma}_k^1 & \hat{\Gamma}_k^2 & \dots & \hat{\Gamma}_k^N\end{bmatrix}^\top = \left.\begin{bmatrix} \frac{\partial x_k^1}{\partial x_{\mu,k}} & \frac{\partial x_k^2}{\partial x_{\mu,k}} & \dots & \frac{\partial x_k^N}{\partial x_{\mu,k}} \end{bmatrix}^\top\right|_{X=\hat{X}_k^+}$$
$$\hat{\gamma}_s^i = \left.\frac{\partial x_k^i(1)}{\partial x_{\mu,k}(1)}\right|_{\hat{x}_k^{i,+}(1)}, \quad \hat{\gamma}_V^i = \left.\frac{\partial x_k^i(2)}{\partial x_{\mu,k}(2)}\right|_{\hat{x}_k^{i,+}(2)}, i \in [1, \dots, N]. \tag{32}$$

For $i \in [1, \dots, N]$, we can solve the differential equations from (32) via separation of variables for the SOC $x_k^i(1)$

$$\int_{\hat{x}_k^{i,+}(1)}^{x^i(1)} \partial x_k^i(1) = \int_{\hat{x}_{\mu,k}^+(1)}^{x_\mu(1)} \hat{\gamma}_{s,k}^i \partial x_{\mu,k}(1)$$
$$x_k^i(1)\Big|_{\hat{x}_k^{i,+}(1)}^{x_k^i(1)} = \hat{\gamma}_{s,k}^i x_{\mu,k}(1)\Big|_{\hat{x}_{\mu,k}^+(1)}^{x_{\mu,k}(1)}$$
$$x_k^i(1) - \hat{x}_k^{i,+}(1) = \hat{\gamma}_{s,k}^i \left(x_{\mu,k}(1) - \hat{x}_{\mu,k}^+(1)\right)$$
$$x_k^i(1) = \hat{\gamma}_{s,k}^i x_{\mu,k}(1) - \hat{\gamma}_{s,k}^i \hat{x}_{\mu,k}^+(1) + \hat{x}_k^{i,+}(1). \tag{33}$$

We must be cautious with the integration for $x_k^i(2)$ since $\hat{\gamma}_V^i$ is not a constant, but dependent on our integration variable. The steps are the following:

$$\int \partial x_{\mu,k}(2) = \int \frac{\partial x_k^i(2)}{\hat{\gamma}_V^i} \tag{34a}$$

$$= \int \frac{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{u_k^j}{C_p^j} - \frac{x_k^{j,+}(2)}{\tau_p^j}\right)}{\frac{u_k^i}{C_p^i} - \frac{x_k^{i,+}(2)}{\tau_p^i}} \partial x_k^i(2) \tag{34b}$$

$$= \int \frac{1}{N} + \frac{\delta}{\frac{u_k^i}{C_p^i} - \frac{x_k^{i,+}(2)}{\tau_p^i}} \partial x_k^i(2) \tag{34c}$$

$$= \frac{1}{N} x_k^i(2) - \delta\tau_p^i \ln\left|\frac{u_k^i}{C_p^i} - \frac{x_k^i(2)}{\tau_p^i}\right| + C \tag{34d}$$

where $\delta := (1/N)\sum_{j=1,\neq i}^{n}((u_k^j/C_p^j) - (x_k^{j,+}(2)/\tau_p^j))$ and C is the constant of integration. We consider the first-order approximation of the result from (34) for small deviations around nominal points $\hat{x}_k^{i,+}(2)$ and $\hat{x}_{\mu,k}^{+}(2)$

$$x_{\mu,k}(2) \approx \hat{x}_{\mu,k}^{+}(2) + \left(\frac{1}{N} + \frac{\delta}{\frac{u_k^i}{C_p^i} - \frac{\hat{x}^{i,+}(2)}{\tau_p^i}}\right) \cdot \left(x_k^i(2) - \hat{x}_k^{i,+}(2)\right) \tag{35a}$$

$$x_{\mu,k}(2) \approx \hat{x}_{\mu,k}^{+}(2) + \frac{1}{\hat{\gamma}_V^i} \cdot \left(x_k^i(2) - \hat{x}_k^{i,+}(2)\right) \tag{35b}$$

$$x_k^i(2) \approx \hat{\gamma}_V^i x_{\mu,k}(2) - \hat{\gamma}_V^i \hat{x}_{\mu,k}^{+}(2) + \hat{x}_k^{i,+}(2) \tag{35c}$$

where the error in (35c) is $\mathcal{O}(\Delta^2)$, $\Delta := x_k^i(2) - \hat{x}_k^{i,+}(2)$. Assembling the vector equation from solutions (33) and (35) for all $i \in [1, \ldots, N]$ and simplifying

$$\begin{aligned} X_k &\approx \hat{\Gamma}_k x_{\mu,k} - \hat{\Gamma}_k \hat{x}_{\mu,k}^{+} + \hat{X}_k^{+} \\ X_k &\approx \hat{\Gamma}_k x_{\mu,k} - \hat{\Gamma}_k \Lambda \hat{X}_k^{+} + \hat{X}_k^{+} \\ X_k &\approx \hat{\Gamma}_k x_{\mu,k} + \left(I - \hat{\Gamma}_k \Lambda\right) \hat{X}_k^{+} \end{aligned} \tag{36}$$

where $\Lambda = (1/N)\left[I\ I\ \ldots\ I\right]$ and is of size $2 \times 2N$. For the time update, we have $\hat{X}_{k+1}^{-} = \hat{\Gamma}_k \hat{x}_{\mu,k+1}^{-} + (I - \hat{\Gamma}_k \Lambda)\hat{X}_k^{+}$ and $\hat{X}_k^{+} = \hat{\Gamma}_k \hat{x}_{\mu,k+1}^{-} + (I - \hat{\Gamma}_k \Lambda)\hat{X}_k^{+}$. Substituting these instances of (36) into (10a)

$$\begin{aligned} \hat{\Gamma}_k \hat{x}_{\mu,k+1}^{-} + \left(I - \hat{\Gamma}_k \Lambda\right)\hat{X}_k^{+} &= A\left(\hat{\Gamma}_k \hat{x}_{\mu,k}^{+} + \left(I - \hat{\Gamma}_k \Lambda\right)\hat{X}_k^{+}\right) \\ &\quad + BU_k \\ \hat{\Gamma}_k \hat{x}_{\mu,k+1}^{-} &= A\hat{\Gamma}_k \hat{x}_{\mu,k}^{+} + (A - I)\left(I - \hat{\Gamma}_k \Lambda\right)\hat{X}_k^{+} \\ &\quad + BU_k. \end{aligned}$$

Multiplying the left pseudoinverse $\hat{\Gamma}_k^{\dagger}$ gives

$$\begin{aligned} \hat{x}_{\mu,k+1}^{-} &= \hat{\Gamma}_k^{\dagger} A \hat{\Gamma}_k \hat{x}_{\mu,k}^{+} + \hat{\Gamma}_k^{\dagger}(A - I)\left(I - \hat{\Gamma}_k \Lambda\right)\hat{X}_k^{+} + \hat{\Gamma}_k^{\dagger} BU_k \\ &= A_\mu \hat{x}_{\mu,k}^{+} + \hat{\Gamma}_k^{\dagger}(A - I)\left(I - \hat{\Gamma}_k \Lambda\right)\hat{X}_k^{+} + B_\mu U_k \end{aligned} \tag{37}$$

effectively isolating $\hat{x}_{\mu,k+1}^{-}$. Equation (37) can be approximated as follows:

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

with error $\hat{\Gamma}_k^{\dagger}(A-I)(I-\hat{\Gamma}_k\Lambda)\hat{X}_k^{+}$. We can derive an upper bound on this error induced by the infinity norm

$$\|\hat{\Gamma}_k^{\dagger}(A - I)(I - \hat{\Gamma}_k \Lambda)\hat{X}_k^{+}\|_\infty \tag{38a}$$

$$\leq \|\hat{\Gamma}_k^{\dagger}\|_\infty \cdot \|A - I\|_\infty \cdot \|I - \hat{\Gamma}_k \Lambda\|_\infty \cdot \|\hat{X}_k^{+}\|_\infty \tag{38b}$$

$$= \|\hat{\Gamma}_k^{\dagger}\|_\infty \cdot \left(\max_{i\in[1,\ldots,N]}\left|\frac{T_s}{\tau_p^i}\right|\right) \cdot \|I - \hat{\Gamma}_k \Lambda\|_\infty \cdot \|\hat{X}_k^{+}\|_\infty \tag{38c}$$

$$= \left(\max\left\{\frac{\sum_{i=1}^{N}\hat{\gamma}_s^i}{\sum_{i=1}^{N}(\hat{\gamma}_s^i)^2}, \frac{\sum_{i=1}^{N}\hat{\gamma}_V^i}{\sum_{i=1}^{N}(\hat{\gamma}_V^i)^2}\right\}\right) \cdot \left(\max_{i\in[1,\ldots,N]}\left|\frac{T_s}{\tau_p^i}\right|\right) \cdot \left(\max_{i\in[1,\ldots,2N]}\left|1 - \frac{\hat{\gamma}^i}{N}\right| + \left|\hat{\gamma}^i - \frac{\hat{\gamma}^i}{N}\right|\right) \cdot \|\hat{X}_k^{+}\|_\infty \tag{38d}$$

where the upper bound vanishes as the sampling period decreases (i.e., $T_s \to 0$).

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

$$P_{\mu,k+1}^{-} = \mathbb{E}\left[(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-})^\top\right] \tag{39}$$

over the prediction. Multiplying $\hat{\Gamma}_k$ to (39), we have

$$\begin{aligned} &\hat{\Gamma}_k P_{\mu,k+1}^{-} \hat{\Gamma}_k^\top \\ &= \hat{\Gamma}_k \mathbb{E}\left[\left(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-}\right)\left(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-}\right)^\top\right]\hat{\Gamma}_k^\top \\ &= \mathbb{E}\left[\hat{\Gamma}_k\left(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-}\right)\left(x_{\mu,k+1} - \hat{x}_{\mu,k+1}^{-}\right)^\top \hat{\Gamma}_k^\top\right] \\ &= \mathbb{E}\left[\left(X_{k+1} - \hat{X}_{k+1}^{-}\right)\left(X_{k+1} - \hat{X}_{k+1}^{-}\right)^\top\right] \\ &= P_{k+1}^{-} \end{aligned} \tag{40}$$

which works out to be the sparse predicted process covariance P_{k+1}^{-} as calculated in (10b). Substituting (10b) into the right-hand side of (40), we have

$$\hat{\Gamma}_k P_{\mu,k+1}^{-} \hat{\Gamma}_k^\top = A\hat{\Gamma}_k P_{\mu,k}^{+} \hat{\Gamma}_k^\top A^\top + Q_k.$$

Multiplying $\hat{\Gamma}_k^{\dagger}$, we have

$$\hat{\Gamma}_k^{\dagger}\left[\hat{\Gamma}_k P_{\mu,k+1}^{-} \hat{\Gamma}_k^\top\right]\left(\hat{\Gamma}_k^{\dagger}\right)^\top = \hat{\Gamma}_k^{\dagger}\left[A\hat{\Gamma}_k P_{\mu,k}^{+} \hat{\Gamma}_k^\top A^\top + Q_k\right]\left(\hat{\Gamma}_k^{\dagger}\right)^\top.$$

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

$$\begin{aligned} P_{\mu,k+1}^{-} &= \left(\hat{\Gamma}_k^{\dagger} A \hat{\Gamma}_k\right) P_{\mu,k}^{+} \left(\hat{\Gamma}_k^{\dagger} A \hat{\Gamma}_k\right)^\top + \hat{\Gamma}_k^{\dagger} Q_k \left(\hat{\Gamma}_k^{\dagger}\right)^\top \\ &= A_\mu P_{\mu,k}^{+} A_\mu^\top + Q_{\mu,k}. \end{aligned}$$

This completes the proof. ■

Remark 4: Though the proposed dense time update (30) and (31) 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 $3N^2 + 6N$, whereas the proposed dense time update requires only $53N + 26$, due largely in part to many of the associated dense matrices having constant size with respect to cell number (see Section VII for more details).

B. Dense Measurement Update

Continuing with the dense model (28), its measurement update equations are given by

$$K_{\mu,k+1} = \frac{P_{\mu,k+1}^{-} H_{\mu,k+1}^\top}{H_{\mu,k+1} P_{\mu,k+1}^{-} H_{\mu,k+1}^\top + R_{\mu,k}} \tag{41a}$$

$$\hat{x}_{\mu,k+1}^{+} = \hat{x}_{\mu,k+1}^{-} + K_{\mu,k+1}\left(y_{\mu,k+1} - h_\mu\left(\hat{X}_{k+1}^{-}, U_k\right)\right) \tag{41b}$$

$$\hat{X}_{k+1}^{+} = \hat{X}_{k+1}^{-} + \hat{\Gamma}_k\left(\hat{x}_{\mu,k+1}^{+} - \hat{x}_{\mu,k+1}^{-}\right) \tag{41c}$$

$$P_{\mu,k+1}^{+} = \left(I - K_{\mu,k+1} H_{\mu,k+1}\right) P_{\mu,k+1}^{-} \tag{41d}$$

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

$$H_{\mu,k+1} = \left.\frac{\partial\left(\frac{1}{N}h(X,U)\right)}{\partial x_\mu}\right|_{\left(\hat{X}^-_{k+1},U_k\right)}. \tag{42}$$

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 (42) and the sparse Jacobian H_{k+1} as computed by (15) satisfy

$$H_{\mu,k+1} = \frac{1}{N}H_{k+1}\hat{\Gamma}_k. \tag{43}$$

Proof: For brevity, we opt for short-hand notation $h := h(X,U)$ for derivations contained within this section. The scalar $(1/N)$ can be moved to the front of the expression in (42) and the Jacobian can be evaluated

$$\begin{aligned}
H_{\mu,k+1} = \frac{1}{N}\frac{\partial h}{\partial x_\mu} &= \left[\ \tfrac{1}{N}\textstyle\sum_{i=1}^N \frac{\partial V^i_\infty}{\partial x_\mu(1)}\ -\tfrac{1}{N}\frac{\partial \sum_{i=1}^N V^i}{\partial x_\mu(2)}\ \right] \\
&= \left[\frac{1}{N}\sum_{i=1}^N \frac{\partial V^i_{\text{co}}}{\partial x_\mu(1)} \quad -\frac{1}{N}N\frac{\partial x_\mu(2)}{\partial x_\mu(2)}\right] \\
&= \left[\frac{1}{N}\sum_{i=1}^N \frac{\partial V^i_{\text{oc}}}{\partial x_\mu(1)} \quad -\frac{1}{N}N\right] \\
&= \left[\frac{1}{N}\left(\frac{\partial V^1_{\text{oc}}}{\partial x_\mu(1)} + \frac{\partial V^2_{\text{oc}}}{\partial x_\mu(1)} + \cdots + \frac{\partial V^N_{\text{oc}}}{\partial x_\mu(1)}\right) \quad -1\right].
\end{aligned}$$

Recall from (22) that $\frac{\partial x^i(1)}{\partial x_\mu(1)} = \hat{\gamma}^i_s$, therefore

$$\begin{aligned}
H_{\mu,k+1} &= \left[\frac{1}{N}\left(\hat{\gamma}^1_s\frac{\partial V^1_{\text{oc}}}{\partial x_1(1)} + \cdots + \hat{\gamma}^N_s\frac{\partial V^N_{\text{oc}}}{\partial x_N(1)}\right) \quad -1\right] \\
&= \left[\sum_{i=1}^N \frac{1}{N}\hat{\gamma}^i_s\frac{\partial V^i_{\text{oc}}}{\partial x_i(1)} \quad -1\right] \\
&= \left[\ \tfrac{1}{N}\textstyle\sum_{i=1}^N \hat{\gamma}^i_s\frac{\partial V^i_{\text{co}}}{\partial x_i(1)} \quad -1\right].
\end{aligned}$$

Now, multiplying (15) by $\hat{\Gamma}_k$ gives

$$\begin{aligned}
H_{k+1}\hat{\Gamma}_k &= \left[\ \tfrac{\partial V^1_{\text{oc}}}{\partial x_1(1)}\ -1\ \tfrac{\partial V^2_{\text{oc}}}{\partial x_2(1)}\ -1\ \cdots\ \tfrac{\partial V^N_{\text{oc}}}{\partial x_N(1)}\ -1\ \right]\hat{\Gamma}_k \\
&= \left[\sum_{i=1}^N \hat{\gamma}^i_{s,k}\frac{\partial V^i_{\text{oc}}}{\partial x_i(1)} - \sum_{i=1}^N \hat{\gamma}^i_{V,k}\right].
\end{aligned}$$

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

$$\begin{aligned}
\frac{1}{N}H_{k+1}\hat{\Gamma}_k &= \frac{1}{N}\left[\sum_{i=1}^N \hat{\gamma}^i_{s,k}\frac{\partial V^i_{oc}}{\partial x_i(1)} \quad -N\right] \\
&= \left[\ \tfrac{1}{N}\textstyle\sum_{i=1}^N \hat{\gamma}^i_{s,k}\frac{\partial V^i_{\text{oc}}}{\partial x_i(1)}\ -1\right] = H_{\mu,k+1}.
\end{aligned}$$

This completes the proof. ■

Lemma 4: The dense Kalman gain $K_{\mu,k+1}$ as computed by (41a) and the sparse Kalman gain K_{k+1} computed by (16a) satisfy

$$K_{k+1} = \frac{1}{N}\hat{\Gamma}_k K_{\mu,k+1}. \tag{44}$$

Proof: According to (41a) and (43) and the fact that $R_{\mu,k} = (R_k/N^2)$, we have

$$\begin{aligned}
\frac{1}{N}\hat{\Gamma}_k K_{\mu,k+1} &= \frac{1}{N}\hat{\Gamma}_k\frac{P^-_{\mu,k+1}H^\top_{\mu,k+1}}{H_{\mu,k+1}P^-_{\mu,k+1}H^\top_{\mu,k+1} + \frac{1}{N^2}R_k} \\
&= \frac{1}{N}\frac{\hat{\Gamma}_k P^-_{\mu,k+1}\left(\frac{1}{N}H_{k+1}\hat{\Gamma}_k\right)^\top}{\left(\frac{1}{N}H_{k+1}\hat{\Gamma}_k\right)P^-_{\mu,k+1}\left(\frac{1}{N}H_{k+1}\hat{\Gamma}_k\right)^\top + \frac{1}{N^2}R_k} \\
&= \frac{1}{N}N\frac{\hat{\Gamma}_k P^-_{\mu,k+1}\left(H_{k+1}\hat{\Gamma}_k\right)^\top}{\left(H_{k+1}\hat{\Gamma}_k\right)P^-_{\mu,k+1}\left(H_{k+1}\hat{\Gamma}_k\right)^\top + R_k} \\
&= \frac{\hat{\Gamma}_k P^-_{\mu,k+1}\hat{\Gamma}^\top_k H^\top_{k+1}}{H_{k+1}\hat{\Gamma}_k P^-_{\mu,k+1}\hat{\Gamma}^\top_k H^\top_{k+1} + R_k}.
\end{aligned}$$

Utilizing (40), we have

$$\frac{1}{N}\hat{\Gamma}_k K_{\mu,k+1} = \frac{P^-_{k+1}H^\top_{k+1}}{H_{k+1}P^-_{k+1}H^\top_{k+1} + R_k} = K_{k+1}.$$

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 (41) to the sparse measurement update outlined in (16).

Theorem 3: The measurement update on $\hat{X}^+_{k+1}$ and P^+_{k+1} as computed by (41) is equivalent to the sparse measurement update as computed by (16).

Proof: From (41c), we have

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

which establishes the equivalence of the state correction of (41) with that of (16).

As for the covariance equation, begin with (41d) and substitute (40) to get

$$\begin{aligned}
P^+_{\mu,k+1} &= \left(I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1}\right)P^-_{\mu,k+1} \\
\hat{\Gamma}_k P^+_{\mu,k+1}\hat{\Gamma}^\top_k &= \hat{\Gamma}_k\left(I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1}\right)P^-_{\mu,k+1}\hat{\Gamma}^\top_k \\
P^+_{k+1} &= \hat{\Gamma}_k\left(I_{2\times 2} - K_{\mu,k+1}H_{\mu,k+1}\right)P^-_{\mu,k+1}\hat{\Gamma}^\top_k.
\end{aligned}$$

Substituting (44) and (43) and simplifying further, we get

$$\begin{aligned}
P^+_{k+1} &= \hat{\Gamma}_k\left(I_{2\times 2} - N\hat{\Gamma}^\dagger_k K_{k+1}\frac{1}{N}H_{k+1}\hat{\Gamma}_k\right)P^-_{\mu,k+1}\hat{\Gamma}^\top_k \\
&= \hat{\Gamma}_k\left(\hat{\Gamma}^\dagger_k I_{2N\times 2N}m\hat{\Gamma}_k - \hat{\Gamma}^\dagger_k K_{k+1}H_{k+1}\hat{\Gamma}_k\right)P^-_{\mu,k+1}\hat{\Gamma}^\top_k \\
&= \hat{\Gamma}_k\hat{\Gamma}^\dagger_k\left(I_{2N\times 2N} - K_{k+1}H_{k+1}\right)\hat{\Gamma}_k P^-_{\mu,k+1}\hat{\Gamma}^\top_k \\
&= \hat{\Gamma}_k\hat{\Gamma}^\dagger_k\left(I_{2N\times 2N} - K_{k+1}H_{k+1}\right)P^-_{k+1} \\
&= \hat{\Gamma}_k\hat{\Gamma}^\dagger_k P^+_{k+1} = P^+_{k+1}.
\end{aligned}$$

Note that Lemma 2 is utilized to arrive at 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 $(16N^3 + 32N^2 + (4P + 14)N + 6M)$ to linear $[(6P + 76)N + 6M + 110]$ 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.

C. Adaptive DEKF

Recall that $Q_{\mu,k} = \hat{\Gamma}_k^{\dagger} Q_k \hat{\Gamma}_k$. 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 (19), (20), and (22)]. While in the context of the regular DEKF this would be the case, there are existing methods of adaptive Kalman filtering [17], [30], [31] in which the dense noise covariance matrices are not computed as a function of $\hat{\Gamma}$, but instead as approximate solutions to the optimization problem $\Theta^* = \arg\min_{Q_k,R_k}[J(\Theta|Y_M)]$ s.t. $Q_k \geq 0, R_k > 0$, where $J(\Theta|Y_M) := \sum_{i=k-M+1}^{k}[\ln|\Sigma_i| + \nu_i^{\top}\Sigma_i^{-1}\nu_i]$, $\Theta := \begin{bmatrix} Q_k & R_k \end{bmatrix}$, the prefit 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 $Y_M := \begin{bmatrix} y_{k-M+1} & y_{k-M} & \dots & y_{k-1} & y_k \end{bmatrix}$. A full derivation of the solution can be located in [30, Appendix C], where suitable approximations of the optimal noise covariance matrices that maintain positive definiteness are found to be

$$Q_k^* = K_k \left[\frac{1}{M} \sum_{i=i_0}^{k} \nu_i \nu_i^{\top} \right] K_k^{\top} \tag{45}$$

$$R_k^* = \frac{1}{M} \sum_{i=i_0}^{k} \left[\epsilon_i \epsilon_i^{\top} + H_i P_i^+ H_i^{\top} \right] \tag{46}$$

where $i_0 = k - M + 1$ and postfit residual $\epsilon_k := y_k - \hat{y}_k^+$. This adaptive formulation of the covariance matrices is worked into the DEKF, which is referred to as the DAEKF and is the solution that generates the simulation results shown later in Section VII.

Remark 6: The additional number of FLOPs incurred by the dense with the adaptive extension is linear $[(2P + 6)N + 6M + 21]$ and occurs in the time update when computing $\hat{\Gamma}_k^{\dagger} Q_k \hat{\Gamma}_k$ (recall that $\hat{\Gamma}_k$ is a function of balancing currents). The adaptive step is a function of window size M [more terms to sum together for (45) and (46)] and P [must re-evaluate polynomials for (48a)].

D. 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 2 and 3 compute $\hat{\Gamma}_k$ and $\hat{\Gamma}_k^{\dagger}$ given X_k^+ and U_k, which are then used to calculate $A_{\mu,k}$ and $B_{\mu,k}$ as shown in Line 4. Lines 5–9 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, $\hat{\Gamma}_0$, and $\hat{\Gamma}_0^{\dagger}$. Line 11 uses the results computed in Line 4 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

Algorithm 1 DAEKF Algorithm: Time Update

```
1: procedure TIME$\left(\hat{x}_{\mu,k}^+, P_{\mu,k}^+, U_k, \hat{X}_k^+, k\right)$
2;     $\hat{\Gamma}_k \leftarrow$ computing (29), (24a), (24b), (22), (25);
3:     $\hat{\Gamma}_k^{\dagger} \leftarrow$ computing (27);
4:     $A_{\mu,k} \leftarrow \hat{\Gamma}_k^{\dagger} A \hat{\Gamma}_k$; $B_{\mu,k} \leftarrow \hat{\Gamma}_k^{\dagger} B$;
5:     if $k = 0$ then
6:         $P_{\mu,0}^+ \leftarrow \hat{\Gamma}^{\dagger} P_0^+ \left(\hat{\Gamma}^{\dagger}\right)^{\top}$
7:         $Q_{\mu} \leftarrow \hat{\Gamma}^{\dagger} Q \left(\hat{\Gamma}^{\dagger}\right)^{\top}$;
8:         $R_{\mu} \leftarrow \frac{1}{N^2} R$;
9:     end if
10:
11:    $\hat{x}_{\mu,k+1}^- \leftarrow A_{\mu,k}\hat{x}_{\mu,k}^+ + B_{\mu,k}U_k$;
12:    $\hat{X}_{k+1}^- \leftarrow \hat{X}_k^+ + \hat{\Gamma}_k\left(\hat{x}_{\mu,k+1}^- - \hat{x}_{\mu,k}^+\right)$;
13:    $P_{\mu,k+1}^- \leftarrow A_{\mu,k} P_{\mu,k}^+ A_{\mu,k}^{\top} + Q_{\mu}$;
14:
15: return $\hat{x}_{\mu,k+1}^-, P_{\mu,k+1}^-, \hat{X}_{k+1}^-$
16: end procedure
```

Algorithm 2 DAEKF Algorithm: Measurement Update

```
1: procedure MEAS$\left(\hat{x}_{\mu,k+1}^-, P_{\mu,k+1}^-, \hat{X}_{k+1}^-, y_{\mu,k+1}, k\right)$
2:     $H_{\mu} \leftarrow$ evaluates (48a) at $\hat{X}_{k+1}^-$;
3:     $K_{\mu,k+1} \leftarrow$ computing (41a);
4:
5:     $\hat{x}_{\mu,k+1}^+ \leftarrow \hat{x}_{\mu,k+1}^- + K_{\mu,k+1}\left(y_{\mu,k+1} - (48\text{ b})\right)$;
6:     $\hat{X}_{k+1}^+ \leftarrow \hat{X}_{k+1}^- + \hat{\Gamma}_k\left(\hat{x}_{\mu,k+1}^+ - \hat{x}_{\mu,k+1}^-\right)$;
7:     $P_{\mu,k+1}^+ \leftarrow \left(I - K_{\mu,k+1}H_{\mu}\right)P_{\mu,k+1}^-$;
8:
9:     $h_{\mu}\left(\hat{X}_{k+1}^+\right) \leftarrow$ computing (48c);
10:    $\epsilon_{\mu,k+1} \leftarrow y_{\mu,k+1} - h_{\mu}\left(\hat{X}_{k+1}^+\right)$;
11:    if $k \geq M - 1$ then
12:        $Q_{\mu} \leftarrow$ computing (45);
13:        $R_{\mu} \leftarrow$ computing (46);
14:    end if
15:    $k \leftarrow k + 1$
16:
17: return $\hat{x}_{\mu,k+1}^+, P_{\mu,k+1}^+, \hat{X}_{k+1}^+$
18: end procedure
```

function of x_{μ} (see a more detailed explanation in Section VII). Line 12 computes the predicted dense process covariance $P_{\mu,k+1}^-$. The results obtained from Lines 11–13 are outputs of the procedure, which are passed onto the measurement update function.

Starting 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 VII) 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 $\hat{\Gamma}_k$ and the differences in the result from Line 5 and the predicted dense state. Line 7 simply computes 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 (48b) 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 postfit residual $\epsilon_{\mu,k+1}$. Lines 11–14 consist of a conditional statement that 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 result of (45) and (46). Afterward, the updated dense state $\hat{x}_{\mu,k+1}$, updated dense process covariance $P^{-}_{\mu,k+1}$, and update sparse state $\hat{X}^{+}_{k+1}$ are fed back into the time update function for the following iteration.

VII. Simulation Results

A. Implementation Details and Methodology

This section elaborates on environment-specific details for the DEKF implementation, the target CPU, as well as considerations regarding the direct computation of FLOPs for each step.

Recall that $H_{\mu,k+1}$ is computed at each time step using (43) 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

$$\min_{p^i} \|S^i p^i - V^i_{\mathrm{oc}}\|_2^2, \quad 1 \le i \le N \tag{47}$$

each evaluated over high-resolution, OCV-SOC data from its respective cell. In the case of (47), p^i is a $(P+1)\times 1$ vector of polynomial coefficients, S_i is an $L\times(P+1)$ design matrix computed from the SOCs of L samples from the ith cell's OCV-SOC curve, and V^i_{oc} is the corresponding output vector of size $L\times 1$. At each time step, the least-squares fit $F^i(s)$ and its derivative $\dot{F}^i(s) := (dF^i/ds^i)$ is evaluated to complete the following steps of the DAEKF algorithm:

$$H^{-}_{\mu,k+1} = \left.\frac{\partial h_\mu}{\partial x_\mu}\right|_{x_\mu=\hat{x}^{-}_{\mu,k+1}} \approx \left[\frac{1}{N}\cdot\sum_{i=1}^{N}\hat{\gamma}^{i}_{s,k}\dot{F}^i(x_i(1)) \quad -1\right]. \tag{48a}$$

$$h_\mu(\hat{X}^{-}_{k+1}) \approx \sum_{i=1}^{N}\left(F^i(x_i(1)) - R^i_o u^i_k - x_i(2)\right) \tag{48b}$$

$$h_\mu(\hat{X}^{+}_{k+1}) \approx h_\mu(\hat{X}^{-}_{k+1}) + H^{-}_{\mu,k+1}\left(\hat{x}^{+}_{\mu,k+1} - \hat{x}^{-}_{\mu,k+1}\right) \tag{48c}$$

where (48a) and (48b) are derived from $\dot{F}^i(s)$ and $F^i(s)$, respectively, to compute the first-order Taylor approximation of the measurement functions in (48c), which is used to compute the postfit residual $\epsilon_{\mu,k+1} = y_{\mu,k+1} - h_\mu(\hat{X}^{+}_{k+1})$ to update R_μ as shown in (46). For this specific implementation, an 8° polynomial is used to approximate the OCV-SOC behavior of each cell. A polynomial with such degree is common in literature [28], [29].

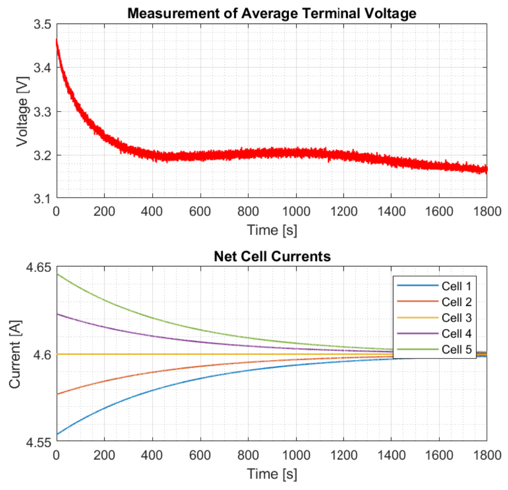

Fig. 4. Pack characteristics over five cells.

In the context of computing theoretical resource consumption, the considerations being made are the following: 1) structural redundancies of the matrices involved in the 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, Horner's rule is the method employed for evaluating polynomials of degree P, which requires $2P$ FLOPs to execute, a matrix's pseudoinverse is computed in accordance with its definition; and 3) 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 Gaussian distributions $\eta^i \sim \mathcal{N}(0.9, 0.1), C^i \sim \mathcal{N}(5, 0.5)$ as a means of inducing heterogeneity unto the pack. For clarity, other circuit parameters $R_p = 2.2166\cdot 10^{-2}$, $C_p = 1.9975\cdot 10^{3}$, and $R_o = 1.3435\cdot 10^{-3}$ are kept constant for all cells. Note that all battery parameters are derived from [32], [33], and [34], 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. The sampling period used in all experiments is $T_s = 0.1$ s. 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.

B. 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. 4). As for the initial states, the EKF initializes $\hat{x}^{i,+}_{k=0}(1) = 1, \hat{x}^{i,+}_{k=0}(2) = 0$ for $i \in [1,\ldots,N]$, and the DEKF initializes with the average values of the EKF's initial values: $\hat{x}^{+}_{\mu,k=0}(1) = 1, \hat{x}^{+}_{\mu,k=0}(2) = 0$. Furthermore, the initial sparse process covariance, the standard deviations for each cell's process noise w^i_k and the pack's measurement noise v_k are initialized as a block diagonal matrix of 10^{-6}, 10^{-4}, and

TABLE V
FLOP COMPARISON FOR DAEKF AND SPARSE ADAPTIVE EKF

Main Step	Sub-Step	Dense FLOPs	Sparse FLOPs
Time Update	$\hat{\Gamma}_k$	$9N$	-
	$\hat{\Gamma}_k^\dagger$	$6N$	-
	$A_{\mu,k}, B_{\mu,k}$	$26N-4$	-
	$\hat{x}_{\mu,k+1}^-$	$4N+4$	-
	$\hat{X}_{k+1}^-$	$8N+2$	$6N$
	$P_{\mu,k+1}^-$	24	-
	P_{k+1}^-	–	$3N^2$
	Total	$53N+26$	$3N^2+6N$
Kalman Gain	H_{k+1}	$(2P-2)N$	$(2P-2)N$
	$H_{\mu,k+1}$	$8N$	–
	K_{k+1}	–	$17N^2$
	$K_{\mu,k+1}$	35	–
	Total	$(2P+6)N+35$	$17N^2+(2P-2)N$
Measurement Update	$\hat{x}_{\mu,k+1}^+$	$(2P+3)N+6$	–
	$\hat{X}_{k+1}^+$	$8N+2$	$4N+1$
	$P_{\mu,k+1}^+$	20	–
	P_{k+1}^+	–	$16N^3+4N^2$
	Total	$(2P+11)N+28$	$16N^3+4N^2+4N+1$
Adaptive Steps	**Total**	$(2P+6)N+6M+21$	$8N^2+(2P+6)N+6M-1$
	Overall	$(6P+76)N+6M+110$	$16N^3+32N^2+(4P+14)N+6M$

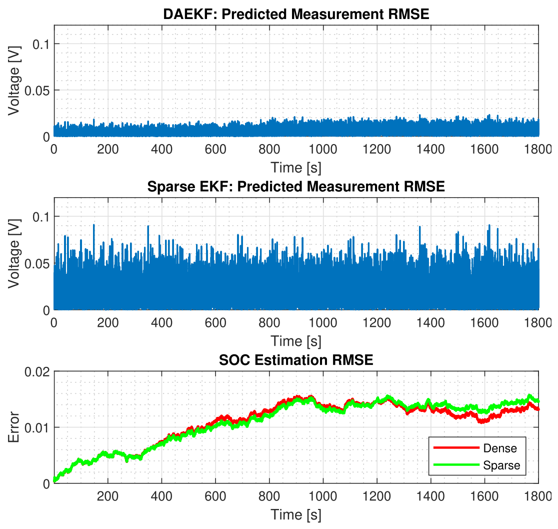

Fig. 5. Comparison of SOC estimation error and predicted measurement error between the sparse EKF and the DAEKF over five cells.

10^{-2}, respectively, and as a result of (40) and the equations for $Q_{\mu,k}$ and $R_{\mu,k}$ given in Section VI, the initial dense covariance matrices are $P_{\mu,0} = [1.918 \cdot 10^{-7}, 0; 0, 1.999 \cdot 10^{-7}]$, $Q_{\mu,0} = [9.198 \cdot 10^{-9}, 0; 0, 9.999 \cdot 10^{-9}]$, $R_{\mu,0} = 4 \cdot 10^{-6}$, $N = 5$, $M = 15$, and $P = 8$. The measured terminal voltage and the cell currents are shown in Fig. 4. For 1800 s, Fig. 5 illustrates the results of the simulation, which show that the DAEKF exhibits good performance in estimating SOC over all five cells with an average RMSE of $1.08 \cdot 10^{-2}$, 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). As shown in the third subplot, the DAEKF, and sparse EKF estimation error follow closely to one another, exhibiting a similar performance when estimating the SOC over five cells. However, there is a minor separation toward the end of the 1800-s run time. This separation is to be expected as expressed in (38).

In the context of FLOP count, a comprehensive comparison between the dense and sparse adaptive EKFs is found in Table V, 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 (16c). Thus, the multiplication of two arbitrary, square matrices of size $2N \times 2N$ yields an FLOP count proportional to N^3. While 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 can be borrowed to get an idea of the FLOPs count for both estimators. Doing so yields an FLOP count of 16 324 690 for the sparse and a mere 12 600 for the dense, reinforcing the perceived intractability of the sparse formulation of the adaptive EKF.

A simulation with $N = 100$, that is, with 100 cells, is also performed. The initial covariance matrices are the following, $P_{\mu,0} = [9.781 \cdot 10^{-9}, 0; 0, 9.999 \cdot 10^{-14}]$, $Q_{\mu,0} = [9.567 \cdot 10^{-9}, 0; 0, 9.999 \cdot 10^{-9}]$, $R_{\mu,0} = 2.500 \cdot 10^{-7}$, yielding the results shown in Fig. 6, where, in comparison to Fig. 5, 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 the DEKF's terminal voltage is around 25 mV, demonstrating a greater noise attenuation as a result of more cells compared to the sparse adaptive EKF. In addition to this, the covariance matrices are intentionally initialized such that the model

TABLE VI
SENSITIVITY OF DEKF'S AVERAGE RMSE TO ERROR IN ECM PARAMETER VALUES

-	-15%	-10%	-5%	$\pm 0\%$	$+5\%$	$+10\%$	$+15\%$
C_p	$8.1 \cdot 10^{-3}$	$8.0 \cdot 10^{-3}$	$7.9 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$7.9 \cdot 10^{-3}$	$7.9 \cdot 10^{-3}$
R_p	$1.3 \cdot 10^{-1}$	$9.7 \cdot 10^{-3}$	$8.1 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$9.8 \cdot 10^{-2}$	diverges	diverges
R_o	$9.3 \cdot 10^{-3}$	$8.4 \cdot 10^{-3}$	$7.9 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$8.5 \cdot 10^{-3}$	diverges	diverges

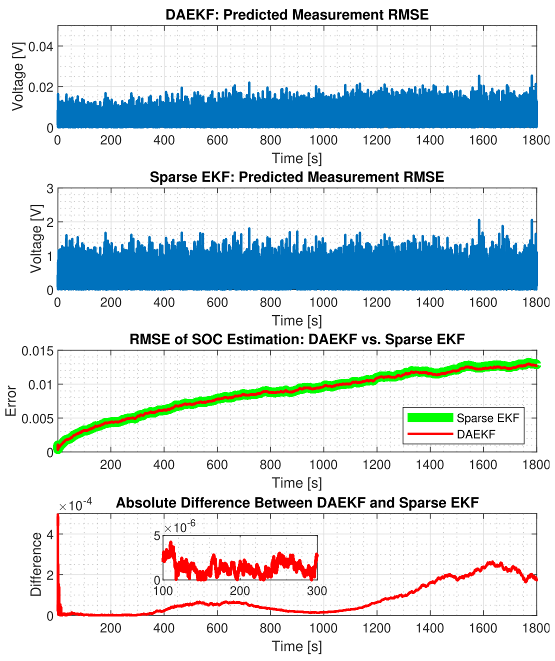

Fig. 6. DAEKF and sparse EKF performance comparison over 100 cells.

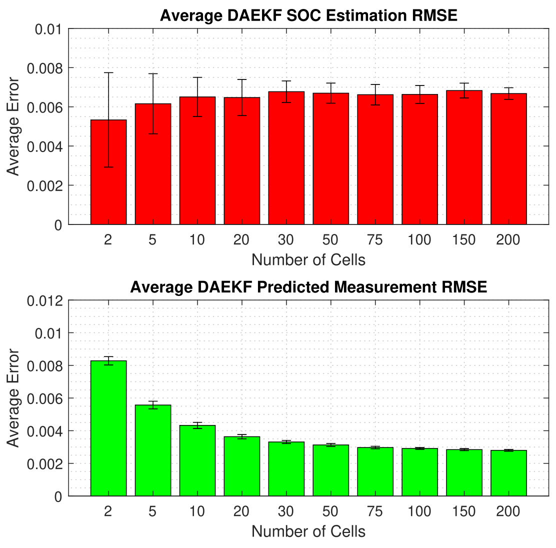

Fig. 7. Evolution of average RMSE in estimation and measurement prediction across 20 trials over various cell numbers.

predictions are favored greatly over the measurements. As a result, the 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 exist on the order of $10^{-4} - 10^{-6}$.

Keeping the initial covariance matrices, process noise, and measurement noise the same as in Fig. 6, the next set of results considers the sensitivity and scalability of the DAEKF in terms of average error. To begin with, the differences between the estimated values for the ECM parameters and their actual values are difficult, if not impossible, to overcome completely [35]. For this reason, it is worthwhile to investigate the SOC estimation errors that arise from the DAEKF when these disparities exist. In the case of $N = 5$, Table VI displays the average SOC RMSE of the DAEKF as ECM parameters C_p, R_p, and R_o deviate by $\pm 5\%$, $\pm 10\%$, and $\pm 15\%$ from their values as listed in Section VII-A. C_p is the least sensitive of the parameters as the largest error incurred by the DAEKF for C_p is $8.1 \cdot 10^{-3}$, a 3.85% increase from the baseline error. As for R_p and R_o, the error is stable within $\pm 5\%$, though the $9.8 \cdot 10^{-2}$ error is too large to be useful. However, once R_p and R_o deviate beyond $\pm 5\%$, the error grows to be unstable, even diverging for $\geq +10\%$. The relative insensitivity of C_p can be explained by recalling the definition of $\gamma^i_{V,k}$ (20) and observing $\tau^i_p (= R^i_p C^i_p)$ in the denominator. The parent function $f(x) = (1/x)$ has the derivative $f'(x) = -(1/x^2)$ with respect to x. Thus, for changes in $x \gg 1$ (i.e., C_p) that occur, the derivative function confirms that the added error is "small," while for $x \ll 1$ (i.e., R_p), the added error is "large."

Fig. 7 plots the estimation error for the 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. 7, the average SOC RMSE increases as more cells are introduced, but levels off at $\approx 7 \cdot 10^{-3}$ as N continues to grow. Meanwhile, the average predicted measurement RMSE decreases with more cells, starting at $8 \cdot 10^{-3}$ and leveling off at about $3 \cdot 10^{-3}$. For both cases, the standard deviation decreases with more cells, indicating that the DEKF performs more consistently for larger N. 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 that rely less on the model are less prone to problems of accumulated error, ergo smaller average error and smaller.

VIII. SPECIAL CASE WITH HOMOGENEOUS CELL CURRENT

Up to this point, the calculation of $\hat{\Gamma}$ has involved dynamic balancing currents, requiring its computation to occur online. However, the case of no balancing currents, that is, $u^1_k = u^2_k = \ldots = u^N_k$, offers the conversion of the calculation to an exclusively offline format. Without loss of generality, recall (19), where the SOC fitness factor is computed to be

$$\hat{\gamma}_{s,k}^{i} = \frac{\frac{\eta^i}{C^i} u_k^i}{\frac{1}{N}\sum_{i=1}^{N} \frac{\eta^i}{C^i} u_k^i} = \frac{\frac{\eta^i}{C^i}}{\frac{1}{N}\sum_{i=1}^{N} \frac{\eta^i}{C^i}} \tag{49}$$

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 (20), a similar simplification to the one in (49) can be made if (V_k^i/τ_p^i) is sufficiently small

$$\hat{\gamma}_{V,k}^{i} \approx \frac{\frac{u_k^i}{C_p^i}}{\frac{1}{N}\sum_{i=1}^{N} \frac{u_k^i}{C_p^i}}. \tag{50}$$

However, computing $\hat{\gamma}_{V,k}^{i}$ 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 (2b) over V_{k+1}^i is given by

$$\left| -\frac{T_s}{\tau_p^i}\hat{V}_k^{i,+} + \frac{T_s}{C_p^i}u_k^i \left(1 - \frac{\sum_{i=1}^{N}\left(\frac{u_k^i}{C_p^i} - \frac{\hat{V}_k^{i,+}}{\tau_p^i}\right)}{\sum_{i=1}^{N}\frac{u_k^i}{C_p^i}}\right)\right|.$$

Proof: To assess the error involved in using the offline approximation of $\hat{\gamma}_{V,k}^{i}$, we consider the difference between the discrete-time voltage update equation (2b) and the voltage update as recovered from multiplying $\hat{\gamma}_{V,k}^{i}$ to the change in the average (21b)

$$f_1 := \left(1 - \frac{T_s}{\tau_p^i}\right)\hat{V}_k^{i,+} + \frac{T_s}{C_p^i}u_k^i \tag{51a}$$

$$f_2 := \hat{V}_k^{i,+} + \hat{\gamma}_{V,k}^{i}\Delta V_{\mu,k}. \tag{51b}$$

For brevity, we denote (51a) and (51b) as f_1 and f_2, respectively. Substituting the approximation (50) and the actual value for $\Delta V_{\mu,k}$ as defined in (18b) and (51b) is expressed as follows:

$$\begin{aligned} f_2 &= \hat{V}_k^{i,+} + \frac{\frac{u_k^i}{C_p^i}}{\frac{1}{N}\sum_{i=1}^{N}\frac{u_k^i}{C_p^i}} \cdot \frac{1}{N}\sum_{i=1}^{N}\left(\frac{T_s}{C_p^i}u_k^i - \frac{T_s}{\tau_p^i}\hat{V}_k^{i,+}\right) \\ &= \hat{V}_k^{i,+} + \frac{\frac{T_s}{C_p^i}u_k^i}{\sum_{i=1}^{N}\frac{u_k^i}{C_p^i}} \cdot \sum_{i=1}^{N}\left(\frac{u_k^i}{C_p^i} - \frac{\hat{V}_k^{i,+}}{\tau_p^i}\right). \end{aligned} \tag{52}$$

To compute the error, (52) is subtracted from (51a), and the absolute value of the result is obtained

$$\begin{aligned} e &= |f_1 - f_2| \\ &= \left| -\frac{T_s}{\tau_p^i}\hat{V}_k^{i,+} + \frac{T_s}{C_p^i}u_k^i \left(1 - \frac{\sum_{i=1}^{N}\left(\frac{u_k^i}{C_p^i} - \frac{\hat{V}_k^{i,+}}{\tau_p^i}\right)}{\sum_{i=1}^{N}\frac{u_k^i}{C_p^i}}\right)\right|. \end{aligned}$$

This completes the proof. ■

Referring to Fig. 8, the conversion of computations involving $\hat{\Gamma}$ to an offline format visibly reduces the DAEKF's FLOPs required for the time update by ≈ 77%. In particular, the offline computations remove $41N - 4$ FLOPs from the online computation of the time update, since $\hat{\Gamma}_k$, $\hat{\Gamma}_k^{\dagger}$, $A_{\mu,k}$, and $B_{\mu,k}$ no longer change between time steps.

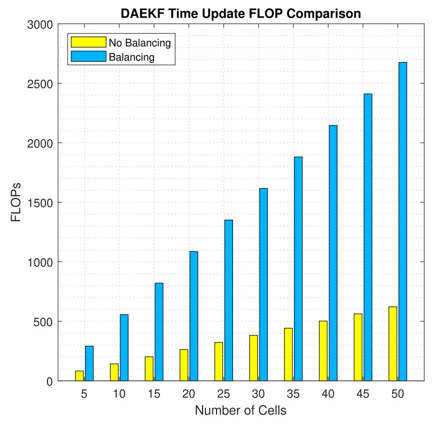

Fig. 8. Comparison of DAEKF FLOP count between the cases of balancing currents and no balancing currents.

IX. Conclusion

In this article, the dense formulation of the EKF was introduced to address the computational overhead and intractable resource demands that hinder the sparse EKF. 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 an 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 $\hat{\Gamma}$ and adjacent computations 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 experiments which include—but are not limited to—degraded conditions of one or more cells and different cell chemistries; 2) event-triggered methods that employ streamlined methods for slowly changing balancing currents; and 3) assessing the feasibility of a "dense" OCV-SOC curve approximation.

Appendix

Proof for Theorem 1: First, we prove equivalence for the SOC update equation. Substituting (18a) and (19) into (21a)

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

which reproduces (2a). As for the relaxation voltage update equation, a similar substitution can be made by substituting (18b) and (20) into (21b), 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 (2b). This completes the proof. ∎

Proof for Lemma 1: Writing out the full form of Γ as defined in (25) gives

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

Observing the full form of Γ, for each nonzero 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 [36]. Next

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

which is symmetric positive definite. Therefore,

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

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

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

This completes the proof. ∎

Proof for Lemma 2: Retaining the definition of $\Gamma^\dagger$ from Lemma 1, the following is computed:

$$\Gamma\Gamma^\dagger = \begin{bmatrix} \frac{\left(\gamma_s^1\right)^2}{\sum\left(\gamma_s^i\right)^2} & 0 & \dots & \frac{\left(\gamma_s^1\right)\left(\gamma_s^N\right)}{\sum\left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\left(\gamma_V^1\right)^2}{\sum\left(\gamma_V^i\right)^2} & \dots & 0 & \frac{\left(\gamma_V^1\right)\left(\gamma_V^N\right)}{\sum\left(\gamma_V^i\right)^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\left(\gamma_s^N\right)\left(\gamma_s^1\right)}{\sum\left(\gamma_s^i\right)^2} & 0 & \dots & \frac{\left(\gamma_s^N\right)^2}{\sum\left(\gamma_s^i\right)^2} & 0 \\ 0 & \frac{\left(\gamma_V^N\right)\left(\gamma_V^1\right)}{\sum\left(\gamma_V^i\right)^2} & \dots & 0 & \frac{\left(\gamma_V^N\right)^2}{\sum\left(\gamma_V^i\right)^2} \end{bmatrix}. \tag{53}$$

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 (53) gives the following:

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

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

References

[1] P. Ahmadi, "Environmental impacts and behavioral drivers of deep decarbonization for transportation through electric vehicles," *J. Cleaner Prod.*, vol. 225, pp. 1209–1219, Jul. 2019.

[2] H. Hao, X. Cheng, Z. Liu, and F. Zhao, "Electric vehicles for greenhouse gas reduction in China: A cost-effectiveness analysis," *Transp. Res. D, Transp. Environ.*, vol. 56, pp. 68–84, Oct. 2017.

[3] X. Chen, W. Shen, T. T. Vo, Z. Cao, and A. Kapoor, "An overview of lithium-ion batteries for electric vehicles," in *Proc. 10th Int. Power Energy Conf. (IPEC)*, Dec. 2012, pp. 230–235.

[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, Dec. 2019, Art. no. 104103.

[5] M. D. Jahnke, F. Cosco, R. Novickis, J. P. 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, p. 250, Feb. 2019.

[6] J. Huang, D. Shi, and T. Chen, "Event-triggered state estimation with an energy harvesting sensor," *IEEE Trans. Autom. Control*, vol. 62, no. 9, pp. 4768–4775, Sep. 2017.

[7] J. Chiasson and B. Vairamohan, "Estimating the state of charge of a battery," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 3, pp. 465–470, May 2005.

[8] J. Chen, A. Behal, Z. Li, and C. Li, "Active battery cell balancing by real-time model predictive control for extending electric vehicle driving range," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 4003–4015, Jul. 2024.

[9] 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 Trans. Control Syst. Technol.*, vol. 28, no. 5, pp. 1862–1877, Sep. 2020.

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

[11] 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 Trans. Control Syst. Technol.*, vol. 25, no. 2, pp. 453–468, Mar. 2017.

[12] K. Movassagh, S. A. Raihan, and B. Balasingam, “Performance analysis of Coulomb counting approach for state of charge estimation,” in *Proc. IEEE Electr. Power Energy Conf. (EPEC)*, Oct. 2019, pp. 1–6.

[13] 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 *Proc. 2nd Int. Conf. Autom., Cognit. Sci., Opt., Micro Electro Mech. Syst., Inf. Technol. (ICACOMIT)*, Oct. 2017, pp. 70–74.

[14] G. Y. Kulikov and M. V. Kulikova, “Accurate numerical implementation of the continuous-discrete extended Kalman filter,” *IEEE Trans. Autom. Control*, vol. 59, no. 1, pp. 273–279, Jan. 2014.

[15] A. Tsiamis and G. J. Pappas, “Online learning of the Kalman filter with logarithmic regret,” *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 2774–2789, May 2023.

[16] S. Liu, Z. Wang, Y. Chen, and G. Wei, “Protocol-based unscented Kalman filtering in the presence of stochastic uncertainties,” *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1303–1309, Mar. 2020.

[17] 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 Trans. Veh. Technol.*, vol. 62, no. 1, pp. 108–117, Jan. 2013.

[18] G. L. Plett, “Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 3. State and parameter estimation,” *J. Power Sources*, vol. 134, pp. 277–292, Aug. 2004.

[19] D. Sun et al., “State of charge estimation for lithium-ion battery based on an intelligent adaptive extended Kalman filter with improved noise estimator,” *Energy*, vol. 214, Jan. 2021, Art. no. 119025.

[20] O. C. Imer and T. Basar, “Optimal estimation with limited measurements,” in *Proc. 44th IEEE Conf. Decis. Control*, Dec. 2005, pp. 1029–1034.

[21] A. Valade, P. Acco, P. Grabolosa, and J.-Y. Fourniols, “A study about Kalman filters applied to embedded sensors,” *Sensors*, vol. 17, no. 12, p. 2810, Dec. 2017.

[22] P. Closas, J. Vilà-Valls, and C. Fernández-Prades, “Computational complexity reduction techniques for quadrature Kalman filters,” in *Proc. IEEE 6th Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process. (CAMSAP)*, Dec. 2015, pp. 485–488.

[23] 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, Aug. 2020, Art. no. 118000.

[24] 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,” *J. Energy Storage*, vol. 32, Dec. 2020, Art. no. 101836.

[25] 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 Trans. Control Syst. Technol.*, vol. 30, no. 3, pp. 1102–1115, May 2022.

[26] 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,” *J. Control Sci. Eng.*, vol. 2018, pp. 1–11, Jun. 2018.

[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 Trans. Veh. Technol.*, vol. 60, no. 4, pp. 1461–1469, May 2011.

[28] Z. Chen, Y. Fu, and C. C. Mi, “State of charge estimation of lithium-ion batteries in electric drive vehicles using extended Kalman filtering,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 3, pp. 1020–1030, Mar. 2013.

[29] Q. Yu, R. Xiong, C. Lin, W. Shen, and J. Deng, “Lithium-ion battery parameters and state-of-charge joint estimation based on H-infinity and unscented Kalman filters,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 8693–8701, Oct. 2017.

[30] C. T. Fraser and S. Ulrich, “Adaptive extended Kalman filtering strategies for spacecraft formation relative navigation,” *Acta Astronautica*, vol. 178, pp. 700–721, Jan. 2021.

[31] J. N. Yang, S. Lin, H. Huang, and L. Zhou, “An adaptive extended Kalman filter for structural damage identification,” *Struct. Control Health Monitor.*, vol. 13, no. 4, pp. 849–867, 2006.

[32] X. Lin et al., “Online parameterization of lumped thermal dynamics in cylindrical lithium ion batteries for core temperature estimation and health monitoring,” *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 5, pp. 1745–1755, Sep. 2013.

[33] X. Lin et al., “Parameterization and observability analysis of scalable battery clusters for onboard thermal management,” *Oil Gas Sci. Technol.-Revue d’IFP Energies nouvelles*, vol. 68, no. 1, pp. 165–178, Jan. 2013.

[34] 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 *Proc. Dyn. Syst. Control Conf.*, vol. 45318, 2012, pp. 41–50.

[35] R. O. Nemes, S. M. Ciornei, M. Ruba, and C. Martis, “Parameters identification using experimental measurements for equivalent circuit lithium-ion cell models,” in *Proc. 11th Int. Symp. Adv. Topics Electr. Eng. (ATEE)*, Mar. 2019, pp. 1–6.

[36] A. Charnes, M. J. Kirby, and R. A. C. M. Va, “*Properties of a generalized inverse with applications to linear programming theory*,” McLean, VA, USA, RAC-TP-171 Aug. 1965.

Luke Nuculaj received the B.S. and M.S. degrees in electrical and computer engineering from Oakland University, Rochester, MI, USA, in 2022 and 2024, respectively, where he is currently pursuing the Ph.D. degree.

His research interests include optimal control and state estimation as applied to automotive and battery systems.

Jun Chen (Senior Member, IEEE) received the bachelor’s degree in automation from Zhejiang University, Hangzhou, China, in 2009, and the Ph.D. degree in electrical engineering from Iowa State University, Ames, IA, USA, in 2014.

He is currently an Associate Professor with the ECE Department, Oakland University, Rochester, MI, USA. His research interests include artificial intelligence and optimal control, with applications in intelligent vehicles and energy systems.

Dr. Chen is currently a member of SAE. He is a recipient of the NSF CAREER Award, the Best Paper Award from IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the Faculty Recognition Award for Research from Oakland University.