Towards fast embedded moving horizon state-of-charge estimation for lithium-ion batteries

Yiming Wan[a,*], Songtao Du[a], Jiayu Yan[a], Zhuo Wang[a]

[a]*School of Artificial Intelligence and Automation, Key Laboratory of Image Processing and Intelligent Control, Engineering Research Center of Autonomous Intelligent Unmanned Systems, Ministry of Education of China, Huazhong University of Science and Technology, Wuhan, 430074, Hubei, China*

Abstract

For state-of-charge (SOC) estimation with high precision, moving horizon estimation (MHE) has recently emerged as a competitive alternative to various Kalman filters or observers due to its significantly improved accuracy and robustness. However, MHE demands considerably higher computational complexity, preventing its deployment on a low-cost embedded microcontroller. To address this challenge, we propose a fast moving horizon SOC estimation algorithm to retain the benefits of MHE at a vastly reduced computational cost. In particular, we consider joint SOC and parameter estimation for an SOC-dependent equivalent circuit model subject to model mismatch. The proposed fast joint MHE (jMHE) algorithm performs a fixed number of Gauss-Newton (GN) iterations to approximate the fully converged solution. At each iteration, the GN Hessian matrix is factorized by exploiting its block tridiagonal structure to construct computationally efficient forward-backward recursions. To further speed up computations, another fast jMHE algorithm wtih an Event-Triggered Relinearization strategy (jMHE-ETR) is proposed to avoid refactorizing the GN Hessian matrix at each iteration. Using experimental datasets under different operating conditions, it is verified on both laptop and Raspberry Pi microcontroller that compared to the conventional optimal MHE solved with an off-the-shelf solver, the proposed fast jMHE and jMHE-ETR algorithms both have a slight loss of performance

*Corresponding author

Email addresses: ywan@hust.edu.cn (Yiming Wan), s_t_du@hust.edu.cn (Songtao Du), jiayuyan@hust.edu.cn (Jiayu Yan), wangzhuo@hust.edu.cn (Zhuo Wang)

(still significantly better than extended Kalman filtering) while reducing its computational cost by an order of magnitude.

Keywords: State-of-charge estimation, moving horizon estimation, real-time computation, lithium-ion battery.

---

1. Introduction

The use of lithium-ion batteries (LIBs) in energy storage and electric vehicles has grown rapidly in recent years. To ensure safety and reliability, an advanced battery management system (BMS) plays a significant role, which typically consists of battery state monitoring, charge/discharge control and balance control [1, 2]. Among them, the battery state-of-charge (SOC) is an important parameter used to protect LIBs from overcharge and overdischarge [3] as well as to mitigate cell imbalance [4]. However, SOC cannot be measured directly, and needs to be inferred from available measurements. Accurate SOC estimation is challenging, because battery measurement information is often limited including only current, voltage, and temperature, and the battery dynamics is strongly nonlinear and varies with operating conditions.

Motivated by the above practical needs and challenges, SOC estimation methods have been extensively investigated in literature, which are mainly classified into three categories: electrochemical model based approach, data-driven approach, and equivalent circuit model (ECM) based approach [3, 5]. Although the electrochemical model provides a detailed description of reaction and transport mechanisms, its real-time application is difficult, because a large number of unknown model parameters need to be identified from a limited number of measured variables, and a high-dimensional state space leads to heavy computation burden [6, 7, 8, 9, 10]. To avoid difficulty in obtaining an explicit battery model, the data-driven method directly constructs an estimator from training data, with measured variables (e.g., current, voltage, temperature) as inputs and SOC as output. The learned estimator can be in the form of deep neural network [11, 12, 13, 14, 15] or Gaussian process regression [16, 17]. Despite its great potential, the data-driven SOC estimation method still has several limitations in its state-of-the-art: firstly, it needs a huge volume of high-quality training data including unmeasurable SOCs; secondly, its estimation performance is guaranteed only in the operating conditions covered by the training data, and cannot be generalized to any unseen operating region.

Compared to electrochemical and data-driven models, the ECM strikes a balance between model complexity and accuracy, thus has been widely used in real-time SOC estimation. The ECM uses electrical circuit components to represent the dominating electrochemical processes within a battery cell. To describe nonlinear dynamics over a wide operating region, the ECM can be enhanced by allowing its parameters to vary with SOC, temperature, aging, and current load [18, 19, 20], or by cascade connection with a static nonlinear function to account for the effect of high current densities or/and low temperature [21]. Any type of ECM inevitably suffers from model mismatch, as a result of time-varying operation condition, aging, or manufacturing inconsistency. To cope with such model mismatch, joint or dual SOC and parameter estimation has been investigated in [22, 23, 24], which imposes an increased degree of nonlinearity due to interactions between states and parameters. To enhance robustness against unexpected outliers in voltage and current measurements, an outlier-resistant SOC estimation method was proposed in [25, 26] by using outlier diagnosis, extended Kalman filtering, and Student's-T filtering. To cope with non-Gaussian noises in SOC estimation, the maximum correntropy criterion with adaptive kernel width was adopted by [27] in an iterative extended Kalman filtering scheme. To address time-varying characteristic of process and measurement noises, variational Bayesian analysis was introduced into the unscented Kalman filter in [28] for robust SOC estimation. Although various nonlinear Kalman filters or observers have been reported in literature, how to avoid divergence and achieve fast convergence remains a challenge for SOC estimation, due to the unknown initial state, nonlinear dynamics, and model mismatch.

Compared to the aforementioned nonlinear Kalman filters or observers, moving horizon estimation (MHE) is a competitive alternative. At each time instant, MHE generally formulates a nonlinear constrained least squares problem using measurement data over a receding horizon. The above MHE problem is then solved to generate state estimates via an iterative numerical optimization algorithm, which is different from the structure of conventional filters or observers. Due to these above features, MHE naturally handles nonlinearity, enhances robustness to a poor initial guess, produces smoother state estimates with higher accuracy, and achieves a faster rate of convergence [29]. Motivated by these benefits, MHE has been recently leveraged for battery state monitoring in a few studies [30, 31, 32, 33, 34, 35, 36, 37]. In [30], an MHE approach for SOC estimation was proposed for an SOC-dependent

ECM whose open circuit voltage (OCV) and parameters all have polynomial dependence on SOC. Such an MHE approach was further extended to address model mismatch [31, 32] via joint SOC and parameter estimation. To address bias, noise corruption or unavailability of current sensors, joint estimation of SOC and load current was investigated in [33, 34] in the MHE framework. In [35], a multiscale MHE method applied to an electrochemical model was proposed for simultaneous SOC and sate-of-health (SOH) estimation. The same MHE formulation was also further leveraged in [36] for a multi-timescale co-estimation hierarchy for SOC, SOH, and state of power. To better cope with large discrepancy in the initial guess and battery cell inconsistencies, the MHE combined with auto-regressive long short-term memory network was proposed in [37].

The improved performance of MHE is generally achieved at the price of increased computational cost due to repeatedly solving an optimization problem over a receding time horizon. For SOC estimation, the associated nonlinear MHE problems in [30, 31, 32, 33, 34, 35, 36, 37] are all computed by using off-the-shelf optimization solvers. These general-purpose solvers are not suitable for real-time implementation on a low-cost and resource-limited embedded microcontroller of a BMS. Within a short sampling interval, a BMS needs to perform various computation tasks, thus may not have sufficient time or resource to allow solving each MHE problem to full convergence. To address the above issue of real-time computation, two main approaches have been reported in literature. The first one leverages a feedforward neural network (NN) to approximate the mapping from the available data of each MHE problem to its generated state estimate [38]. By doing so, the computationally expensive online optimization is replaced by the computationally cheaper NN. However, the above NN based approach cannot guarantee accurate estimation outside the range of operating conditions covered the training data, which is the inherent limitation on the generalization capability of any learning based method. The other approach for real-time implementation of nonlinear MHE is to develop fast algorithms by limiting the number of iterations [39, 40, 41] and exploiting the specific structure of the MHE problem [42, 43]. Such fast MHE algorithms have not been fully investigated for SOC estimation.

In this paper, we work on a fast MHE algorithm for joint SOC and parameter estimation using an SOC-dependent ECM subject to model mismatch. To retain the benefits of MHE at a vastly reduced computational cost, we propose a fast joint MHE (jMHE) algorithm which extends the generalized linear Kalman smoothing algorithm in [42] to the SOC-depedent nonlinear ECM. Specifically, we perform a fixed number of Gauss-Newton (GN) iterations to approximately solve each nonlinear MHE problem within a sampling interval. In each iteration, we solve the GN equation via computationally efficient forward-backward recursions by exploiting the block tridiagonal structure of the GN Hessian matrix. Moreover, we introduce an event-triggered relinearization strategy to avoid updating and refactorizing the GN Hessian matrix if the linearization point is subject to a sufficiently small change with respect to the previous iteration. The proposed fast jMHE algorithm is validated on both laptop and Raspberry Pi microcontroller using experimental data under different operating conditions, with comparison to the conventional joint extended Kalman filter (EKF) and the optimal jMHE.

The contribution of this paper lies in the following aspects:

i) All existing moving horizon SOC estimation literature such as [30, 31, 32, 33, 34, 35, 36, 37] solve the associated MHE problem by using a general-purpose optimization solver which is not customized for real-time computation on a low-cost microcontroller. To address this issue, we propose the fast jMHE algorithm to retain the benefits of MHE at a vastly reduced computational cost compared to using an off-the-shelf general-purpose solver.

ii) Different from the fast MHE literature [42, 43], our fast jMHE algorithm additionally introduces an event-triggered relinearization strategy which further improves computation efficiency at the expense of a slight loss of performance.

iii) We demonstrate the real-time feasibility of our proposed fast jMHE algorithm on a low-cost Raspberry Pi microcontroller. This bridges the gap to implementing moving horizon SOC estimation on an embedded BMS.

The rest of this paper is as follows. Section 2 describes the SOC-dependent ECM and the associated joint SOC and parameter estimation problem. The moving horizon SOC estimation problem formulation and its fast algorithm are elaborated in Section 3. In Section 4, the proposed fast jMHE algorithm is verified by comparison with the conventional joint EKF and optimal jMHE. Finally, some concluding remarks are given in Section 5.

2. SOC-dependent ECM and problem description

In this section, we first describe the SOC-dependent ECM of the LIB, and then present the joint SOC and parameter estimation problem.

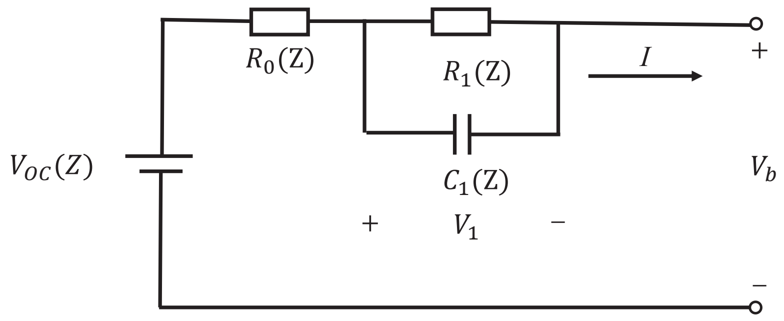


Figure 1: First-order SOC-dependent ECM for LIBs.

The SOC-dependent first-order ECM shown in Fig. 1 is widely adopted due to its simplicity and accuracy. The open-circuit voltage V_{oc} represents the electromotive force of the LIB. The ohmic internal resistance R_0 reflects the ohmic characteristics. The battery polarization resistance R_1 and polarization capacitor C_1 characterize the hysteresis characteristics during the polarization effect. The SOC Z indicates the ratio of the remaining capacity to the nominal capacity, ranging from 0 to 1. In this paper, it is assumed that the SOC estimation is conducted under a constant temperature, hence the above circuit parameters only depend on SOC.

According to the electric circuit analysis, the first-order ECM in Fig. 1 is expressed as

$$\frac{dZ}{dt} = -\frac{I}{C_n}, \tag{1a}$$

$$\frac{dV_1}{dt} = \frac{I}{C_1(Z)} - \frac{V_1}{R_1(Z)C_1(Z)}, \tag{1b}$$

$$V_b = V_{\mathrm{oc}}(Z) - R_0(Z)I - V_1, \tag{1c}$$

where C_n is the nominal capacity of the LIB, I represents the load current whose value is positive at discharge, V_1 denotes the voltage across the parallel RC circuit, and V_b is the terminal voltage. As in [31], we use polynomial functions to describe the SOC-dependent parameters $V_{\mathrm{oc}}(Z)$, $R_0(Z)$, $R_1(Z)$, and $C_1(Z)$ as

$$V_{\mathrm{oc}}(Z)=\sum_{j=0}^{n}\alpha_j Z^j,\ R_0(Z)=\beta_{10}+\sum_{j=1}^{m_1}\beta_{1,j}Z^j, \tag{2a}$$

$$R_1(Z)=\beta_{20}+\sum_{j=1}^{m_2}\beta_{2,j}Z^j,\ C_1(Z)=\beta_{30}+\sum_{j=1}^{m_3}\beta_{3,j}Z^j, \tag{2b}$$

where α_j, $\beta_{1,j}$, $\beta_{2,j}$ and $\beta_{3,j}$ are polynomial coefficients, n and m_i are the polynomial orders. Note that the nominal capacity C_n and the polynomial dependencies in (2) can be obtained offline via the capacity test, the OCV test and parameter identification methods [44, 18]. Since the battery aging is out of scope of this paper, the battery capacity C_n is assumed to be a known constant.

For the SOC estimation using sampled measurements of the current I and the terminal voltage V_b, let Δt denote the small sampling interval, Z_k, I_k, $V_{1,k}$, and $V_{b,k}$ represent the samples of Z, I, V_1, and V_b at time $k\Delta t$, respectively. Define $R_{0,k}=R_0(Z_k)$, $R_{1,k}=R_1(Z_k)$, $C_{1,k}=C_1(Z_k)$, and $\tau_{1,k}=R_{1,k}C_{1,k}$. With these notations, we transform the continuous-time ECM (1) into the following discrete-time model

$$Z_{k+1}=Z_k-\frac{I_k\Delta t}{C_n}+w_{1,k}, \tag{3a}$$

$$\begin{aligned}V_{1,k+1}&=V_{1,k}\exp\left(-\frac{\Delta t}{\tau_{1,k}}\right)+I_kR_{1,k}\left[1-\exp\left(-\frac{\Delta t}{\tau_{1,k}}\right)\right]\\&\quad+w_{2,k},\end{aligned} \tag{3b}$$

$$V_{b,k}=V_{\mathrm{oc}}(Z_k)-V_{1,k}-I_kR_{0,k}+v_k, \tag{3c}$$

where $w_{1,k}$ and $w_{2,k}$ are process noises due to discretization errors, and v_k denotes the measurement noise.

To account for the model mismatch due to time-varying operating conditions, it is necessary to update the ECM parameters in (3) to improve the SOC estimation performance. As reported in [31], only the zero-order polynomial coefficients β_{10}, β_{20}, and β_{30} defined in (2) are updated online as $\beta_{10,k}$, $\beta_{20,k}$, and $\beta_{30,k}$, respectively, to compensate for the model mismatch, which avoids the high computational cost required for updating all model parameters. This leads to a joint SOC and parameter estimation problem. To this end, the following augmented state-space model is introduced according to (3):

$$x_{k+1} = f(x_k, u_k) + w_k, \tag{4a}$$

$$y_k = h(x_k, u_k) + v_k, \tag{4b}$$

with the augmented state x_k, the system input u_k, the system output y_k, the nonlinear functions $F(x_k, u_k)$ and $h(x_k, u_k)$ defined as

$$x_k = \begin{bmatrix} Z_k & V_{1,k} & \beta_{10,k} & \beta_{20,k} & \beta_{30,k} \end{bmatrix}^\top, \tag{5a}$$

$$u_k = I_k, \quad y_k = V_{b,k}, \tag{5b}$$

$$f(x_k, u_k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \exp\left(-\frac{\Delta t}{\tau_{1,k}}\right) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} -\frac{\Delta t}{C_n} \\ R_{1,k}\left[1 - \exp\left(-\frac{\Delta t}{\tau_{1,k}}\right)\right] \\ 0 \\ 0 \\ 0 \end{bmatrix} u_k, \tag{5c}$$

$$h(x_k, u_k) = V_{\mathrm{oc}}(Z_k) - V_{1,k} - I_k R_{0,k}. \tag{5d}$$

The noise term w_k in (4) consists of $w_{1,k}$ and $w_{2,k}$ in (3) as well as the random-walk noises for the time-varying parameters $\beta_{10,k}$, $\beta_{20,k}$, and $\beta_{30,k}$.

Using the above augmented state-space model (4), the joint estimation problem aims at estimating SOC at each time k while updating parameter estimates of $\beta_{10,k}$, $\beta_{20,k}$, and $\beta_{30,k}$.

3. Fast moving horizon joint SOC and parameter estimation

In this section, we formulate an MHE problem for joint SOC and parameter estimation, and propose its fast algorithm. Over each time horizon, the proposed algorithm performs a fixed number of structure-exploiting GN iterations to approximate the fully converged solution of the optimal MHE. This significantly reduces the computation time per sample compared to the optimal MHE algorithm, which is desirable for real-time implementation within a short sampling interval.

3.1. MHE formulation for joint SOC and parameter estimation

The MHE approach copes with joint SOC and parameter estimation by exploiting the input and output data over the sliding time window of finite length N. At time k, the jMHE problem is formulated as the following nonlinear least-squares optimization problem over the time window $[l, k]$ according to the augmented state-space model (4), i.e.,

$$\begin{aligned}\min_{\{x_i\}_{i=l}^{k}} \ & \frac{1}{2}\left\|x_l - \widehat{x}_{l|k-1}\right\|^2_{P_l^{-1}} + \frac{1}{2}\sum_{i=l}^{k-1}\left\|x_{i+1} - f(x_i, u_i)\right\|^2_{Q^{-1}} \\ & + \frac{1}{2}\sum_{i=l}^{k}\left\|y_i - h(x_i, u_i)\right\|^2_{R^{-1}}, \end{aligned} \tag{6}$$

where $\|s\|^2_M = s^\top M s$ represents a weighted vector norm with M being a positive definite matrix. Let $\widehat{x}_{i|j}$ denote the estimate of x_i computed at time j, then $\widehat{x}_{l|k-1}$ in the first term of (6) is the a priori state estimate of x_l at time $k-1$. The weighting matrices P_l^{-1}, Q^{-1}, and R^{-1} in (6) will be discussed later. Note that there are two scenarios for the value of l in (6):

i) If $k < N$, set $l = 0$. This means that the time window $[0, k]$ of a growing length, i.e., all the information up to the current time k, is considered.
ii) If $k \geq N$, set $l = k - N + 1$. This corresponds to using a sliding time window $[k - N + 1, k]$ of length N.

In the statistical setting, the solution of the jMHE problem (6) can be interpreted as maximum likelihood estimation if the initial state, the process noise and the measurement noise are all zero-mean Gaussian distributed, with the positive definite matrices Q and R in (6) being the noise covariances [29]. Moreover, the first term in the cost function of (6), called the arrival cost, is derived from the negative logarithm of the conditional density of $\widehat{x}_{l|k-1}$ derived at time $k-1$, which captures the information about x_l before the current time k. The matrix P_0 is a tuning parameter used for $k < N$. In the case of $k \geq N$, P_l needs to be updated as k increases, and it can be iteratively computed via the classic extended Kalman filter, i.e.,

$$\begin{aligned} P_l = {} & Q + A_{l-1}P_{l-1}A_{l-1}^\top \\ & - A_{l-1}P_{l-1}C_{l-1}^\top\left(R + C_{l-1}P_{l-1}C_{l-1}^\top\right)^{-1}C_{l-1}P_{l-1}A_{l-1}^\top, \end{aligned} \tag{7}$$

with $A_{l-1} = \nabla_x F(\widehat{x}_{l-1|k-1}, u_{l-1})$ and $C_{l-1} = \nabla_x h(\widehat{x}_{l-1|k-1}, u_{l-1})$. In the deterministic setting, the matrices Q and R are tuning parameters, and the arrival cost term in (6) can be regarded as a quadratic approximation to the cost before the current time horizon. Interested readers are referred to [29, 40] for more details about MHE and the derivation of its arrival cost.

Most literature such as [30, 31, 32, 33, 34, 35, 36, 37] that applied various MHE formulations to SOC estimation fully solves each single MHE problem per sample to convergence by using off-the-shelf optimization solvers, which involves a considerably high computational burden. This is often not allowed by a BMS that needs to simultaneously perform different computation tasks on a low-cost microcontroller within a short sampling interval. As such, we proceed to develop a fast jMHE algorithm in the following subsection to significantly reduce the computational cost while slightly compromising its estimation accuracy compared to the fully converged MHE solution.

3.2. Fast joint MHE algorithm

The proposed fast algorithm performs a fixed number of structure-exploiting GN iterations to solve the jMHE problem (6) at each time instant k. As depicted in Fig. 2, the basic idea consists of two aspects: 1) a shifted warm start obtained from the previous solution; 2) GN iterations that leverage the block tridiagonal structure of the GN Hessian matrix to construct efficient forward-backward recursions.

At each time k, the jMHE problem (6) over the time interval $[l, k]$ is solved iteratively. For $k \geq N$,We determine the initial guess $\{\widehat{x}^{(0)}_{i|k}\}^{k}_{i=l}$ by a warm start strategy that shifts the jMHE solution $\{\widehat{x}_{i|k-1}\}^{k-1}_{i=l}$ at the previous time $k-1$, i.e.,

$$\begin{aligned} \widehat{x}^{(0)}_{i|k} &= \widehat{x}_{i|k-1},\ i \in [l, k-1], \\ \widehat{x}^{(0)}_{k|k} &= f(\widehat{x}_{k-1|k-1}, u_{k-1}), \end{aligned} \tag{8}$$

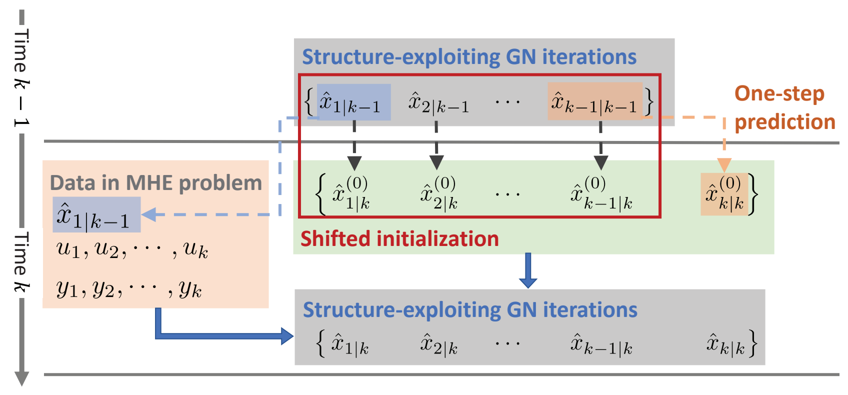
with $f(\cdot)$ defined in (4a).

At each GN iteration, we compute the solution $\{\widehat{x}^{(j)}_{i|k}\}^{k}_{i=l}$ from the previous solution $\{\widehat{x}^{(j-1)}_{i|k}\}^{k}_{i=l}$. Firstly, we equivalently rewrite (6) as

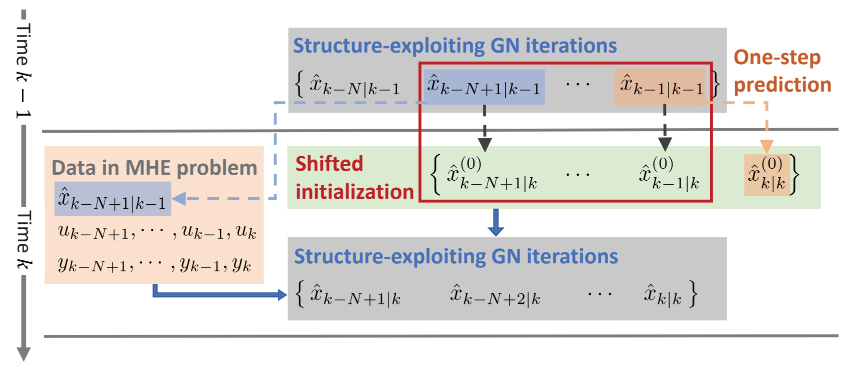
$$\min_{X_k} \frac{1}{2} \left\| F(X_k, \mathcal{I}_k) \right\|^2_{W^{-1}}, \tag{9}$$

where $X_k = \{x_i\}^{k}_{i=l}$ consists of the optimization variables,

$$\mathcal{I}_k = \{\widehat{x}_{l|k-1}, u_l, y_l, \cdots, u_{k-1}, y_{k-1}, y_k\}$$


Time $k-1$
Structure-exploiting GN iterations
$\{\hat{x}_{1|k-1}\quad \hat{x}_{2|k-1}\quad \cdots \quad \hat{x}_{k-1|k-1}\}$
One-step prediction
Data in MHE problem
$\hat{x}_{1|k-1}$
$u_1, u_2, \cdots, u_k$
$y_1, y_2, \cdots, y_k$
$\{\hat{x}^{(0)}_{1|k}\quad \hat{x}^{(0)}_{2|k}\quad \cdots \quad \hat{x}^{(0)}_{k-1|k}\quad \hat{x}^{(0)}_{k|k}\}$
Shifted initialization
Time k
Structure-exploiting GN iterations
$\{\hat{x}_{1|k}\quad \hat{x}_{2|k}\quad \cdots \quad \hat{x}_{k-1|k}\quad \hat{x}_{k|k}\}$

(a) $k < N$


Time $k-1$
Structure-exploiting GN iterations
$\{\hat{x}_{k-N|k-1}\quad \hat{x}_{k-N+1|k-1}\quad \cdots \quad \hat{x}_{k-1|k-1}\}$
One-step prediction
Data in MHE problem
$\hat{x}_{k-N+1|k-1}$
$u_{k-N+1}, \cdots, u_{k-1}, u_k$
$y_{k-N+1}, \cdots, y_{k-1}, y_k$
Shifted initialization
$\{\hat{x}^{(0)}_{k-N+1|k}\quad \cdots \quad \hat{x}^{(0)}_{k-1|k}\quad \hat{x}^{(0)}_{k|k}\}$
Time k
Structure-exploiting GN iterations
$\{\hat{x}_{k-N+1|k}\quad \hat{x}_{k-N+2|k}\quad \cdots \quad \hat{x}_{k|k}\}$

(b) $k \geq N$

Figure 2: Schematic diagram of the MHE algorithm.

represents the data for the MHE problem (6) at time k, W is a block diagonal matrix whose diagonal blocks are $P_l, R_l, Q_l, R_{l+1}, \cdots, Q_{k-1}, R_k$, i.e.,

$$W = \text{diag}\{P_l, R_l, Q_l, R_{l+1}, \cdots, Q_{k-1}, R_k\}, \tag{10}$$

and $F(X_k, \mathcal{I}_k)$ is defined as

$$F(X_k, \mathcal{I}_k) = \begin{bmatrix} x_l - \widehat{x}_{l|k-1} \\ y_l - h(x_l, u_l) \\ x_{l+1} - f(x_l, u_l) \\ y_{l+1} - h(x_{l+1}, u_{l+1}) \\ \vdots \\ x_k - f(x_{k-1}, u_{k-1}) \\ y_k - h(x_k, u_k) \end{bmatrix}. \tag{11}$$

Then, we derive the following least-squares (LS) subproblem from (9) by replacing $F(X_k, \mathcal{I}_k)$ with its first-order Taylor expansion with respect to $\widehat{X}_k^{(j-1)} = \{\widehat{x}_{i|k}^{(j-1)}\}_{i=l}^k$:

$$\min_{\Delta X_k} \frac{1}{2} \left\| F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) + J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) \Delta X_k \right\|_{W^{-1}}^2, \tag{12}$$

where $F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$, ΔX_k, and the Jacobian $J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) = \frac{\partial}{\partial X_k} F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$ are expressed as

$$F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) = \begin{bmatrix} r_{x,l}^\top & r_{y,l}^\top & r_{f,l}^\top & r_{y,l+1}^\top & \cdots & r_{f,k-1}^\top & r_{y,k}^\top \end{bmatrix}^\top, \tag{13a}$$

$$\Delta X_k = \begin{bmatrix} \Delta x_l^\top & \Delta x_{l+1}^\top & \cdots & \Delta x_k^\top \end{bmatrix}^\top, \tag{13b}$$

$$J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) = \begin{bmatrix} I_{n_x} & & & & \mathbf{0} \\ -C_l & & & & \\ -A_l & I_{n_x} & & & \\ & -C_{l+1} & & & \\ & -A_{l+1} & \ddots & I_{n_x} & \\ & & \ddots & -C_{k-1} & \\ & & & -A_{k-1} & I_{n_x} \\ \mathbf{0} & & & & -C_k \end{bmatrix}, \tag{13c}$$

with

$$\Delta x_i = x_i - \widehat{x}_{i|k}^{(j-1)}, \; r_{x,l} = \widehat{x}_{l|k}^{(j-1)} - \widehat{x}_{l|k-1}, \tag{14a}$$

$$r_{f,i} = \widehat{x}_{i+1|k}^{(j-1)} - f\left(\widehat{x}_{i|k}^{(j-1)}, u_i\right), \; r_{y,i} = y_i - h\left(\widehat{x}_{i|k}^{(j-1)}, u_i\right), \tag{14b}$$

$$A_i = \nabla_x f\left(\widehat{x}_{i|k}^{(j-1)}, u_i\right), \; C_i = \nabla_x h\left(\widehat{x}_{i|k}^{(j-1)}, u_i\right). \tag{14c}$$

Note that the superscript $(j-1)$ is omitted for Δx_i, $r_{x,l}$, $r_{f,i}$, $r_{y,i}$, A_i, and C_i in (14) for the sake of notation simplicity. The first-order optimality condition of the above LS subproblem (12) is then derived as the GN equation

$$J^\top(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) W^{-1} J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)\Delta X_k = -J^\top(\widehat{X}_k^{(j-1)}, \mathcal{I}_k) W^{-1} F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k). \tag{15}$$

Using (10) and (13c), the above GN equation (15) can be explicitly expressed as

$$\underbrace{\begin{bmatrix} \Phi_l & -\Gamma_l^\top & & \mathbf{0} \\ -\Gamma_l & \Phi_{l+1} & -\Gamma_{l+1}^\top & \\ & \ddots & \ddots & \ddots \\ \mathbf{0} & & -\Gamma_{k-1} & \Phi_k \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \Delta x_l \\ \Delta x_{l+1} \\ \vdots \\ \Delta x_k \end{bmatrix}}_{\Delta X_k} = \underbrace{\begin{bmatrix} r_{d,l} \\ r_{d,l+1} \\ \vdots \\ r_{d,k} \end{bmatrix}}_{r}, \tag{16}$$

with

$$\Phi_l = P_l^{-1} + A_l^\top Q^{-1} A_l + C_l^\top R^{-1} C_l, \tag{17a}$$
$$\Phi_i = Q^{-1} + A_i^\top Q^{-1} A_i + C_i^\top R^{-1} C_i,\ i \in [l+1, k-1], \tag{17b}$$
$$\Phi_k = Q^{-1} + C_k^\top R^{-1} C_k, \tag{17c}$$
$$\Gamma_i = Q^{-1} A_i,\ i \in [l, k-1], \tag{17d}$$
$$r_{d,l} = -P_l^{-1} r_{x,l} + A_l^\top Q^{-1} r_{f,l} + C_l^\top R^{-1} r_{y,l}, \tag{17e}$$
$$r_{d,i} = -Q^{-1} r_{f,i-1} + A_i^\top Q^{-1} r_{f,i} + C_i^\top R^{-1} r_{y,i},\ i \in [l+1, k-1], \tag{17f}$$
$$r_{d,k} = -Q^{-1} r_{f,k-1} + C_k^\top R^{-1} r_{y,k}. \tag{17g}$$

To efficiently solving the GN equation (16), the block tridiagonal structure of the GN Hessian matrix M is exploited. First, we perform an LU decomposition $M = LU$, with

$$\Sigma_l = \Phi_l,\ \Sigma_{i+1} = \Phi_{i+1} - \Gamma_i \Sigma_i^{-1} \Gamma_i^\top,\ i \in [l, k-1], \tag{18a}$$

$$L = \begin{bmatrix} \Sigma_l & & & \mathbf{0} \\ -\Gamma_l & \Sigma_{l+1} & & \\ & \ddots & \ddots & \\ \mathbf{0} & & -\Gamma_{k-1} & \Sigma_k \end{bmatrix}, \tag{18b}$$

$$U = \begin{bmatrix} I_{n_x} & -\Sigma_l^{-1}\Gamma_l^\top & & \mathbf{0} \\ & I_{n_x} & -\Sigma_{l+1}^{-1}\Gamma_{l+1}^\top & \\ & & \ddots & \\ \mathbf{0} & & & I_{n_x} \end{bmatrix}. \tag{18c}$$

Then, we solve the GN equation (16) via forward-backward recursions as detailed in Algorithm 2, i.e., the forward recursion computes $\Delta X'_k$ from $L\Delta X'_k = r$ and the backward recursion computes ΔX_k from $U\Delta X_k = \Delta X'_k$. With the obtained solution ΔX_k, the estimate is updated as $X_k^{(j)} = X_k^{(j-1)} + \Delta X_k$.

To satisfy the real-time requirement, we perform a fixed number of GN iterations without requiring a fully converged solution to the original nonlinear MHE problem (9). The details are summarized in Algorithms 1 and 2.

Algorithm 1 Fast jMHE algorithm at each time k

```
Input: the input and output sequence $\{u_i, y_i\}_{i=l}^{k}$ at time $k$, the state estimates $\{\widehat{x}_{i|k-1}\}_{i=l}^{k-1}$ at time $k-1$, and the fixed number of GN iterations $m$. Note that $l = 0$ if $k < N$, and $l = k - N + 1$ if $k > N$.
1: Use $P_0$ in the arrival cost if $k < N$; update $P_l$ according to (7) if $k \geq N$;
2: Determine the initial guess $\left\{\widehat{x}_{i|k}^{(0)}\right\}_{i=l}^{k}$ according to (8);
3: for $j = 1 \to m$ do                                ▷Structure-exploiting GN iterations
4:     Update $r_{x,l}$, $\{r_{f,i}, A_i\}_{i=l}^{k-1}$ and $\{r_{y,i}, C_i\}_{i=l}^{k}$ in (14) using $\{\widehat{x}_{i|k}^{(j-1)}, u_i\}_{i=l}^{k}$;
5:     Update $\{\Phi_i\}_{i=l}^{k}$, $\{\Gamma_i\}_{i=l}^{k-1}$, and $\{r_{d,i}\}_{i=l}^{k}$ in (17);
6:     Compute the solution $\{\Delta x_i\}_{i=l}^{k}$ to (16) using Algorithm 2;
7:     Update $\widehat{x}_{i|k}^{(j)} \leftarrow \widehat{x}_{i|k}^{(j-1)} + \Delta x_i$ for $i \in [l, k]$;
8: end for
Output: $\widehat{x}_{i|k} = \widehat{x}_{i|k}^{(m)}$ for $i \in [l, k]$.
```

3.3. Event-triggered relinearization

To further reduce the computational cost, we introduce an event-triggered relinearization (ETR) strategy, to construct the fast jMHE-ETR algorithm described in Algorithms 3 and 4. For the conventional GN method, relinearization is conducted at each iteration to derive the LS subproblem (12). However, we propose the ETR strategy to perform relinearization only if necessary, such that the computational cost is further reduced with a slight loss of estimation performance.

The proposed ETR strategy is depicted in Fig. 3, and explained as follows. To this end, we introduce the linear system

$$J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}} \Delta X_k = -J_{\text{lin}}^{\top} W^{-1} F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k). \tag{19}$$

Algorithm 2 Forward-backward recursion to solve (16) in the fast jMHE algorithm

```
Input: $\{\Phi_i, r_{d,i}\}_{i=l}^{k}$, $\{\Gamma_i\}_{i=l}^{k-1}$
1: Initialization: $\Sigma_l \leftarrow \Phi_l$, $\Delta x'_l \leftarrow \Sigma_l^{-1} r_{d,l}$
2: for $i = l+1 \rightarrow k$ do                                  ▷ Forward recursion
3:     $\Sigma_i \leftarrow \Phi_i - \Gamma_{i-1}\Sigma_{i-1}^{-1}\Gamma_{i-1}^{\top}$
4:     $\Delta x'_i \leftarrow \Sigma_i^{-1}(r_{d,i} + \Gamma_{i-1}\Delta x'_{i-1})$
5: end for
6: $\Delta x_k \leftarrow \Delta x'_k$
7: for $i = k-1 \rightarrow l$ do                                  ▷ Backward recursion
8:     $\Delta x_i \leftarrow \Delta x'_i + \Sigma_i^{-1}\Gamma_i^{\top}\Delta x_{i+1}$
9: end for
Output: $\{\Delta x_i\}_{i=l}^{k}$
```

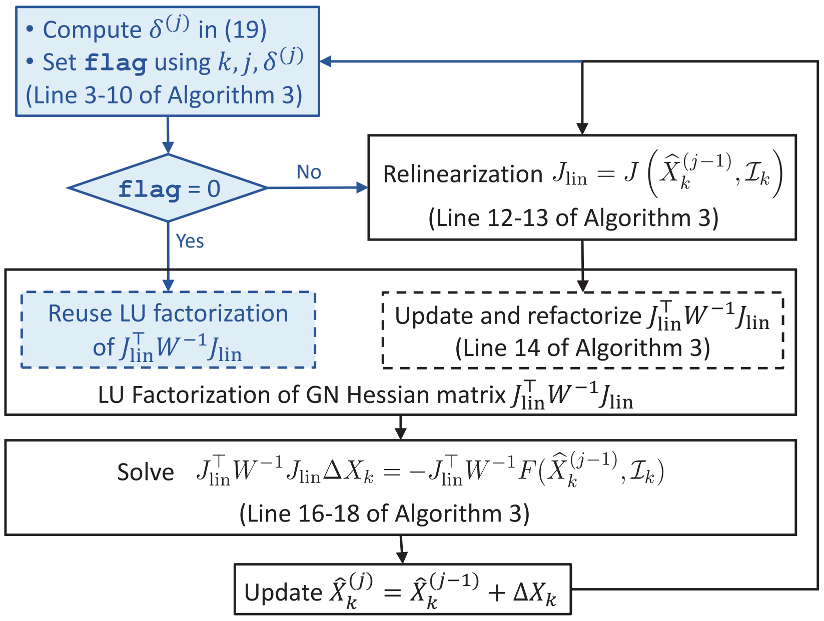

• Compute $\delta^{(j)}$ in (19)
• Set flag using $k, j, \delta^{(j)}$
(Line 3-10 of Algorithm 3)
flag = 0
No
Yes
Relinearization $J_{\text{lin}} = J\left(\widehat{X}_k^{(j-1)}, \mathcal{I}_k\right)$
(Line 12-13 of Algorithm 3)
Reuse LU factorization of $J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}}$
Update and refactorize $J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}}$
(Line 14 of Algorithm 3)
LU Factorization of GN Hessian matrix $J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}}$
Solve $J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}} \Delta X_k = -J_{\text{lin}}^{\top} W^{-1} F(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$
(Line 16-18 of Algorithm 3)
Update $\widehat{X}_k^{(j)} = \widehat{X}_k^{(j-1)} + \Delta X_k$

Figure 3: A flow chart of GN iterations with the ETR strategy.

With (19), the proposed fast jMHE algorithm in the previous subsection can be regarded as first performing relinearization to update $J_{\text{lin}} = J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$, and then solving (19) at each GN iteration. In contrast, the fast jMHE-ETR algorithm does not implement relinearization if J_{lin} used in the previous iteration is still a good approximation to the new Jacobian matrix $J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$. To achieve this goal without explicitly computing $J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$, we evaluate the difference between the solution $\left\{\widehat{x}_{i|k}^{(j-1)}, u_i\right\}_{i=l}^{k}$ from the $(j-1)$th iteration and the linearization point $\left\{x_i^{\text{lin}}, u_i^{\text{lin}}\right\}_{i=0}^{N-1}$ used to obtain J_{lin}, i.e.,

$$\delta^{(j)} = \max_{i\in[l,k]} \frac{\left\| \begin{bmatrix} \widehat{x}_{i|k}^{(j-1)} \\ u_i \end{bmatrix} - \begin{bmatrix} x_{i-l}^{\text{lin}} \\ u_{i-l}^{\text{lin}} \end{bmatrix} \right\|_2}{\left\| \begin{bmatrix} x_{i-l}^{\text{lin}} \\ u_{i-l}^{\text{lin}} \end{bmatrix} \right\|_2}. \tag{20}$$

With a predefined small positive scalar ε, $\delta^{(j)} \leq \varepsilon$ implies that the new Jacobian $J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$ can be well approximated by the previous Jacobian J_{lin} obtained at $\{x_i^{\text{lin}}, u_i^{\text{lin}}\}_{i=0}^{N-1}$. In this case, we do not need relinearization to update the Jacobian J_{lin} as $J(\widehat{X}_k^{(j-1)}, \mathcal{I}_k)$, and still get a valid descent direction ΔX_k by solving (19).

With the above ETR strategy (see line 3-10 in Algorithm 3), we save substantial computational cost in the following two aspects:

i) The GN Hessian matrix $J_{\text{lin}}^{\top} W^{-1} J_{\text{lin}}$ in (19) remains unchanged, and its LU factorization in (18) does not need to be recalculated. As such, the linearized system matrices $\{A_i\}_{i=l}^{k-1}$ and $\{C_i\}_{i=l}^{k}$ in (14c) as well as Φ_i, Γ_i in (17a)-(17d) and Σ_i in (18) are not updated, and their values obtained in the previous iteration are directly used.

ii) When solving the GN equation (16), the matrices Σ_i, Σ_i^{-1}, $\Sigma_i^{-1}\Gamma_i^{\top}$, and $\Sigma_i^{-1}\Gamma_{i-1}$ do not need to be updated as in Algorithm 2. Instead, their previous values are saved and updated in line 14 of Algorithm 3, and then directly used in Algorithm 4 if relinearization is not performed.

3.4. Comparisons and discussions

Similar MHE formulations were adopted for SOC estimation or joint SOC and parameter estimation in [30, 31, 32, 33, 34, 35, 36, 37]. However, in these

Algorithm 3 Fast jMHE-ETR algorithm at each time k

```
Input: the input and output sequence $\{u_i, y_i\}_{i=l}^{k}$ at time $k$, the state estimates $\{\widehat{x}_{i|k-1}\}_{i=l}^{k-1}$ at time $k-1$, and the fixed number of GN iterations $m$. The arrival-cost weighting matrix $P_l$ is fixed to $P_0$. Note that $l = 0$ if $k < N$, and $l = k - N + 1$ if $k \geq N$. The scalar parameter $\varepsilon$ is set to a small value for ETR.
 1: Determine the initial guess $\left\{\widehat{x}_{i|k}^{(0)}\right\}_{i=l}^{k}$ according to (8);
 2: for $j = 1 \to m$ do                          ▷Structure-exploiting GN iterations
 3:     Compute $\delta^{(j)}$ in (20) if $(k < N, j > 1)$ or $k \geq N$;
 4:     if $k < N$ and $j = 1$ then
 5:         flag ← 1;
 6:     else if $\delta^{(j)} > \varepsilon$ then
 7:         flag ← 1;
 8:     else
 9:         flag ← 0.
10:     end if
11:     if flag = 1 then                          ▷Event-triggered relinearization
12:         Update $\widehat{x}_{i-l}^{\text{lin}} \leftarrow \widehat{x}_{i|k}^{(j-1)}, u_{i-l}^{\text{lin}} \leftarrow u_i$ for $i \in [l, k]$;
13:         Update $\{A_i\}_{i=l}^{k-1}$ and $\{C_i\}_{i=l}^{k}$ in (14) using $\{\widehat{x}_{i|k}^{(j-1)}, u_i\}_{i=l}^{k}$;
14:         Update $\{\Phi_i\}_{i=l}^{k}$, $\{\Gamma_i\}_{i=l}^{k-1}$ in (17), and $\{\Sigma_i\}_{i=l}^{k}$ in (18a), then compute $S_0, \cdots, S_{N-1}$, $W_0, \cdots, W_{N-1}$, $E_1, \cdots, E_{N-1}$ as follows
            $$\begin{cases} S_{i-l} = \Sigma_i^{-1},\ W_{i-l} = \Sigma_i^{-1}\Gamma_i^{\top} & i = l, \cdots, k \\ E_{i-l} = \Sigma_i^{-1}\Gamma_{i-1}, & i = l+1, \cdots, k \end{cases}$$
15:     end if
16:     Update $r_{x,l}$, $\{r_{f,i}\}_{i=l}^{k-1}$ and $\{r_{y,i}\}_{i=l}^{k}$ in (14) using $\{\widehat{x}_{i|k}^{(j-1)}, u_i\}_{i=l}^{k}$;
17:     Update $\{r_{d,i}\}_{i=l}^{k}$ in (17) ;
18:     Perform Algorithm 4 to compute $\{\Delta x_i\}_{i=l}^{k}$;
19:     Update $\widehat{x}_{i|k}^{(j)} \leftarrow \widehat{x}_{i|k}^{(j-1)} + \Delta x_i$ for $i \in [l, k]$;
20: end for
Output: $\widehat{x}_{i|k} = \widehat{x}_{i|k}^{(m)}$ for $i \in [l, k]$.
```

Algorithm 4 Forward-backward recursion to solve (16) in the fast jMHE-ETR algorithm

```
Input: ${S_{i-l}, W_{i-l}, r_{d,i}\}_{i=l}^{k}$, $\{E_{i-l}\}_{i=l+1}^{k}$
1: $\Delta x'_l \leftarrow S_0 r_{d,l}$
2: for $i = l+1 \rightarrow k$ do                          ▷Forward recursion
3:     $\Delta x'_i \leftarrow S_{i-l} r_{d,i} + E_{i-l} \Delta x'_{i-1}$
4: end for
5: $\Delta x_k \leftarrow \Delta x'_k$
6: for $i = k-1 \rightarrow l$ do                          ▷Backward recursion
7:     $\Delta x_i \leftarrow \Delta x'_i + W_{i-l} \Delta x_{i+1}$
8: end for
Output: $\{\Delta x_i\}_{i=l}^{k}$
```

latest literature, the MHE or jMHE problems for SOC estimation are all fully solved to convergence by using a general-purpose optimization solver, without explicitly considering the limited time allowed for real-time implementation. For each iteration of a general-purpose solver, the computation complexity of solving the MHE problem is dominated by the cost of solving the GN equation, and is $O(N^3(n_x + n_w)^3)$ if both the state and process noise sequences are estimated, where N is the horizon length, n_x and n_w represent the dimensions of x_i and w_i, respectively [45]. If only the state sequence is computed without estimating the process noises, the above computation complexity becomes $O(N^3 n_x^3)$. In both of the above two cases, the computation complexity is cubic with respect to the horizon length N, which is undesirable in terms of computation efficiency.

Unlike the general-purpose solvers, the proposed fast jMHE and jMHE-ETR algorithms exploit the block tridiagonal structure of the GN Hessian matrix. By doing so, the computation complexity of solving the GN equation in Algorithm 2 is reduced to $O(N n_x^3)$ that is linear with respect to the horizon length N.

In addition, our proposed fast jMHE and jMHE-ETR algorithms consume less memory than using the general-purpose solver. The proposed fast algorithms do not need to store all entries of the Jacobian matrix in (13c), the GN Hessian matrix in (16), and the factored LU matrices L and U in (18), but just require the sparse non-zero blocks in these matrices to be stored. For example, the Jacobian matrix in (13c) is stored as $\{A_i\}_{i=l}^{k-1}$ and $\{C_i\}_{i=l}^{k}$, which uses $(N-1)n_x^2 + N n_x n_y$ memory entries, with $N = k - l + 1$. In contrast, the general-purpose solver treats the above matrices as dense, thus requires all of their entries to be stored, e.g., $N^2 n_x(n_x + n_y)$ memory entries are consumed for the Jacobian matrix in (13c) by the general-purpose solver.

Both our previous work in [46] and our proposed fast algorithms in this paper rely on the structure-exploiting GN iterations. Compared to [46], the proposed fast algorithms in this paper has been improved in the following aspects:

i) The proposed fast jMHE algorithm in this paper further reduces the computation complexity per GN iteration from $O(N(n_x + n_w)^3)$ to $O(Nn_x^3)$ while retaining exactly the same estimation accuracy. Specifically, the work in [46] formulated the MHE problem as

$$\begin{aligned} \min_{\{x_i\}_{i=l}^{k}, \{w_i\}_{i=l}^{k-1}} \quad & \frac{1}{2}\left\|x_l - \widehat{x}_{l|k-1}\right\|^2_{P_l^{-1}} + \frac{1}{2}\sum_{i=l}^{k-1}\|w_i\|^2_{Q^{-1}} \\ & + \frac{1}{2}\sum_{i=l}^{k}\|y_i - h(x_i, u_i)\|^2_{R^{-1}} \\ \text{s.t.} \quad & x_{i+1} = f(x_i, u_i) + w_i,\ i \in [l, k-1], \end{aligned} \tag{21}$$

to simultaneously estimates the state sequence $\{x_i\}_{i=l}^{k}$ and process noise sequence $\{w_i\}_{i=l}^{k-1}$, hence its computation complexity per iteration is $O(N(n_x + n_w)^3)$. In comparison, the newly formulated problem (6) in this paper is equivalent to (21), because the former can be derived from the latter by replacing the process noise w_i with $x_{i+1} - F(x_i, u_i)$ in the cost function. As such, with the same number of GN iterations, solving (6) produces exactly the same SOC estimates as solving (21). But the computational cost is reduced to $O(Nn_x^3)$ due to eliminating the need to estimate the process noises.

ii) The proposed fast jMHE algorithm in this paper implements a fixed number of GN iterations as long as it is allowed in real-time. An increased number of GN iterations generally results in improved estimation accuracy at the price of higher computational cost, as will be illustrated in Section 4. However, the work in [46] performed only one Gauss-Newton iteration over each time horizon.

iii) We introduce the ETR strategy in Section 3.3 to further reduce the computational burden at the cost of a slight performance loss, which has been reported in neither [46] nor other fast MHE literature.

iv) As will be seen in Section 4.5, we evaluate the performance and computational cost of the proposed fast MHE algorithm on a low-cost Raspberry Pi microcontroller, which has not been done in [46] or other fast MHE literature.

4. Estimation results using experimental data

To verify the effectiveness of our proposed fast jMHE and jMHE-ETR algorithms, we adopt the experimental datasets of 18650 LiNiMnCoO2/Graphite lithium-ion cells released by Center for Advanced Life Cycle Engineering (CALCE) at University of Maryland for case study in this section [47]. Firstly, we describe the datasets, and give the SOC-dependent ECM identification result in Section 4.1; in Section 4.2, we introduce the estimation algorithms to be implemented, and compare their estimation performance and computational costs given a set of tuning parameters; in Sections 4.3 and 4.4, we explain the effect of tuning the iteration number and the ETR threshold, respectively; and finally, in Section 4.5 we deploy the aforementioned algorithms on a Raspberry Pi microcontroller to further illustrate the capability of our proposed fast jMHE and jMHE-ETR algorithms to satisfy real-time computation requirement.

The source code of this paper is available at `https://data.mendeley.com/datasets/3zv48n4jn5/3` [48].

4.1. Dataset description and ECM identification

As described in [47], the test cells were placed inside a thermal chamber whose temperature is controlled at 25°C, and the data were collected by an Arbin BT2000 battery test system. An incremental-current OCV test was performed to construct the OCV-SOC curve, and datasets were generated under four operating conditions including the Federal Urban Driving Schedule (FUDS), the US06 Highway Driving Schedule, the Dynamic Stress Test (DST) and the Beijing Dynamic Stress Test (BJDST), which are available at `https://calce.umd.edu/battery-data`. Under each one of the above operating conditions, the sampling rate is 0.1Hz during the charging process; in the discharging process, the sampling rate is 0.1Hz when the SOC is above 0.8, and it increases to 1Hz when the SOC is below 0.8. The current profiles are shown in Fig. 4.

Before performing online SOC estimation, the SOC-dependent ECM described in Section 2 is identified using the OCV test data and the FUDS

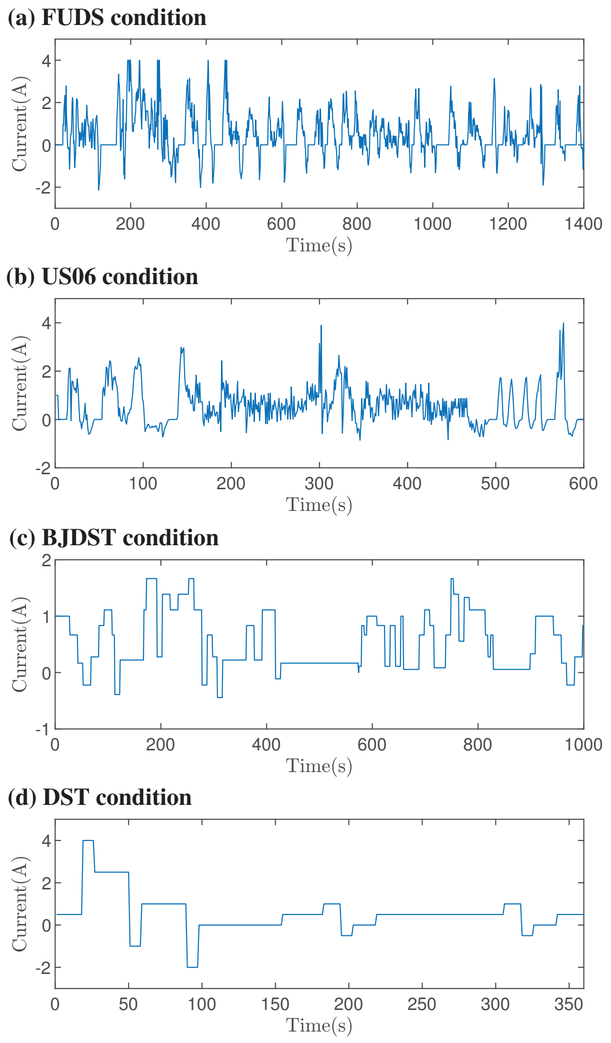


Figure 4: One cycle of current profiles under four operating conditions.

dataset. Considering the two sampling rates (0.1Hz and 1Hz, as mentioned in the previous paragraph) when collecting the datasets, the data with the sampling rate 1Hz, which cover the SOC range $0 \sim 0.8$ in the discharging process, are selected for the ECM identification. The offline model identification consists of two steps: the OCV-SOC relation in the form of a polynomial function is first constructed by using data from the OCV test, then the SOC-dependent ECM parameters $R_0(Z)$, $R_1(Z)$, and $C_1(Z)$ are identified from the FUDS dataset using the multi-ARX algorithm described in our previous work [18]. The identified result is listed in Table 1. The accuracy of the identified ECM is validated by the root mean square error of the predicted terminal voltage, i.e., 0.0017V, 0.0022V, 0.0017V, and 0.0023V using the FUDS, US06, BJDST, and DST datasets, respectively.

Table 1: Identified SOC-dependent ECM parameters.

ECM parameters	**SOC-dependent polynomials**
$V_{\text{oc}}(Z)$	$6.22Z^5 - 19.91Z^4 + 23.98Z^3 - 12.66Z^2$ $+3.29Z + 3.24$
$R_0(Z)$	$-0.638Z^5 + 1.63Z^4 - 1.60Z^3 + 0.774Z^2$ $-0.187Z + 0.089$
$R_1(Z)$	$7.74Z^5 - 16.88Z^4 + 13.00Z^3 - 4.17Z^2 + 0.507Z$ $+0.0027$
$C_1(Z)$	$124589.17Z^5 - 203175.87Z^4 + 92495.58Z^3$ $-644.92Z^2 - 6863.34Z + 1877.26$

4.2. Comparisons of estimation performance and computational costs

Considering the inexactness of the identified model in Table 1, we adopt the joint SOC and parameter estimation scheme using the augmented state-space model (4). The following joint SOC and parameter estimation algorithms are implemented for comparison:

- Joint EKF (jEKF);

- Optimal jMHE: the jMHE problem (6) over each time horizon is fully solved to obtain a converged solution, using the `MATLAB` `lsqnonlin` routine whose termination tolerance parameters `OptimalityTolerance` and `StepTolerance` are both set to 10^{-16};

- Fast jMHE, i.e., our proposed method described in Algorithms 1 and 2;

- Fast jMHE with ETR (jMHE-ETR), i.e., our proposed method described in Algorithms 3 and 4.

Results of the above SOC estimation algorithms under the US06, BJDST and DST conditions are compared in the following three aspects:

- Convergence speed: the amount of time it takes for the estimation error to converge from its initial nonzero value;

- Root mean square error (RMSE) of SOC:

$$\text{RMSE of SOC} = \sqrt{\frac{\sum_{i=1}^{N_d} \left|\widehat{Z}_i - Z_i\right|^2}{N_d}} \tag{22}$$

  where $\widehat{Z}_i$ represents the SOC estimate at time instant i, and N_d is the number of samples used in performance evaluation;

- Computational cost: the average and worst-case computation times spent in computing estimates over each time horizon. Our computational results in Sections 4.2 and 4.4 are obtained by running `MATLAB R2021b` on a laptop computer (4-core, Intel Core i7-8550U CPU @ 1.80GHz, 8GB memory) , and those in Section 4.5 are obtained by running `OCTAVE` on a Raspberry Pi 4 Model B microcontroller.

To evaluate estimation performance of aforementioned algorithms, we add to the terminal voltage measurement a zero-mean white noise with standard deviation 0.001V. All the considered estimation algorithms have the tuning parameters P_0, Q, and R whose values are determined via extensive grid search in this paper. The tuning parameters for jEKF are set as

$$\begin{aligned} P_0 &= \text{diag}(10^{-2}, 10^{-3}, 10^{-6}, 10^{-6}, 10^{-6}), \\ Q &= \text{diag}(10^{-6}, 10^{-2}, 10^{-6}, 10^{-6}, 10^{-6}), R = 10^{-6}, \end{aligned}$$

while the optimal jMHE, fast jMHE, and fast jMHE-ETR set their tuning parameters to the same values, i.e.,

$$P_0 = \mathrm{diag}(10^{-2}, 10^{-4}, 10^{-6}, 10^{-6}, 10^{-6}),$$
$$Q = \mathrm{diag}(10^{-9}, 10^{-1}, 10^{-6}, 10^{-6}, 10^{-6}), R = 10^{-6},$$

all with the horizon length $N = 3$. The fast jMHE and jMHE-ETR algorithms are implemented with different number of GN iterations $m = 2, 3, 4$ over each time horizon, and the triggering threshold ε for the fast jMHE-ETR is set to $\varepsilon = 0.01$.

With different initial state guesses, results of the above algorithms under the US06, BJDST, and DST conditions are depicted in Fig. 5. It can be easily seen that the jEKF gives the slowest convergence speed, while the optimal jMHE, fast jMHE, and fast jMHE-ETR algorithms achieve fast convergence. To compare the estimation errors, the RMSEs of SOC given by the above algorithms with the initial guess $\widehat{Z}_0 = 0.4$ under different conditions are listed in Table 2. The jEKF gives the largest RMSE of SOC, while the optimal jMHE achieves the best estimation performance overall. The RMSEs of our proposed fast jMHE and jMHE-ETR algorithms generally decrease as the number of iterations m increases. By setting the triggering threshold $\varepsilon = 0.01$, the fast jMHE-ETR achieves almost the same RMSEs as the fast jMHE.

The computational costs of the above algorithms are evaluated under the BJDST condition, as listed in Table 3. To obtained a fully converged solution, the optimal jMHE takes 5.80ms and 4 iterations in average per sample, and is the most computationally expensive due to using the general-purpose solver `lsqnonlin` in `MATLAB`. In contrast, the proposed fast jMHE significantly reduces the averaged computation time to 0.42ms per sample by performing the structure-exploiting GN iterations, while the fast jMHE-ETR further speeds up to take only 0.31ms per sample due to additionally adopting the ETR strategy. As expected, the joint EKF is the fastest one in terms of computation speed, but it gives the slowest convergence and the largest RMSE as illustrated in Fig. 5 and Table 2.

According to the above comparisons, our proposed fast jMHE and jMHE-ETR achieve significantly higher estimation accuracy than the conventional jEKF, and are also computationally much cheaper than the optimal jMHE. Therefore, our proposed fast jMHE and jMHE-ETR algorithms strike a good balance between estimation accuracy and computational cost.

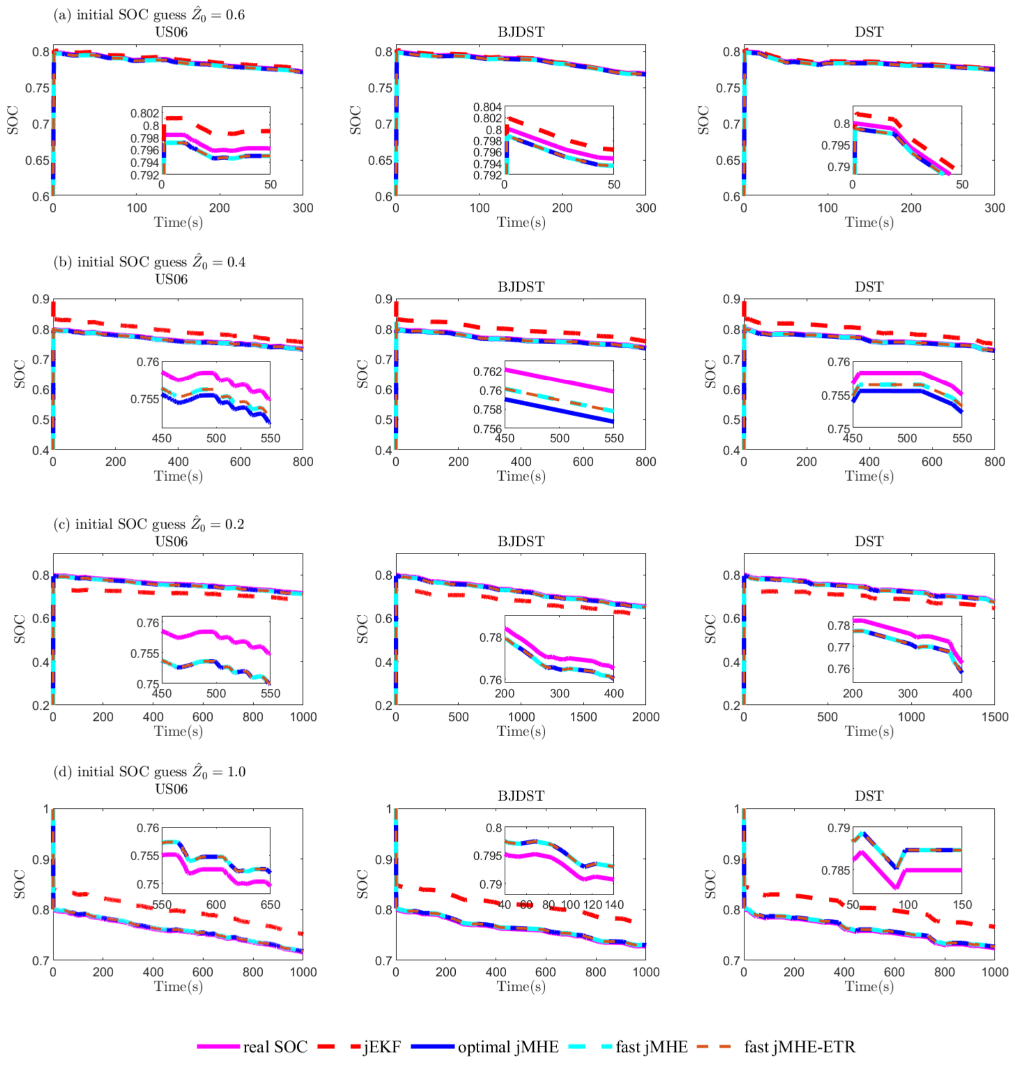


Figure 5: Results of implemented estimation algorithms with different initial SOC guesses under the US06, BJDST, and DST conditions.

Table 2: RMSEs of SOC given by implemented algorithms (with $\widehat{Z}_0 = 0.4$) under US06, BJDST, and DST conditions, tested on a laptop computer. The horizon lengths of the optimal jMHE, fast jMHE, and fast jMHE-ETR are all set to $N = 3$.

RMSE of SOC	**US06**	**BJDST**	**DST**
jEKF	0.0145	0.0091	0.0090
optimal jMHE	0.0018	0.0024	0.0019
fast jMHE ($m = 2$)	0.0037	0.0022	0.0040
fast jMHE ($m = 3$)	0.0017	0.0014	0.0022
fast jMHE ($m = 4$)	0.0018	0.0023	0.0019
fast jMHE-ETR ($m = 2, \varepsilon = 10^{-2}$)	0.0037	0.0022	0.0044
fast jMHE-ETR ($m = 3, \varepsilon = 10^{-2}$)	0.0017	0.0014	0.0022
fast jMHE-ETR ($m = 4, \varepsilon = 10^{-2}$)	0.0018	0.0023	0.0019

4.3. Tuning the number of iterations

In general, increasing the number of GN iterations results in a reduced RMSE and an increased computational cost, as demonstrated by Fig. 6. It should be noted that the RMSEs of SOC given by the fast jMHE and jMHE-ETR increase from 0.0014 to 0.0023 for the BJDST dataset when the number of GN iterations increases from 3 to 4. This observation implies that more iterations do not necessarily result in a smaller RMSE of SOC. The reason can be explained as follows. As in Fig. 6(b), increasing the number of iterations indeed leads to a decreased value of the objective function in (6) for the fast jMHE and jMHE-ETR. Although this improves the overall RMSE of the system state vector x_k, the RMSE of SOC does not necessarily decrease since SOC is just one element of x_k.

4.4. Tuning triggering threshold of fast jMHE-ETR

As shown in Tables 2 and 3, the proposed fast jMHE-ETR algorithm with its triggering threshold $\varepsilon = 10^{-2}$ achieves almost the same RMSEs of SOC as the fast jMHE while significantly reducing the computational cost. It is then of particular interest to investigate the rule-of-thumb for tuning the triggering threshold ε, i.e., how the RMSE and computational cost vary when increasing or decreasing the triggering threshold ε.

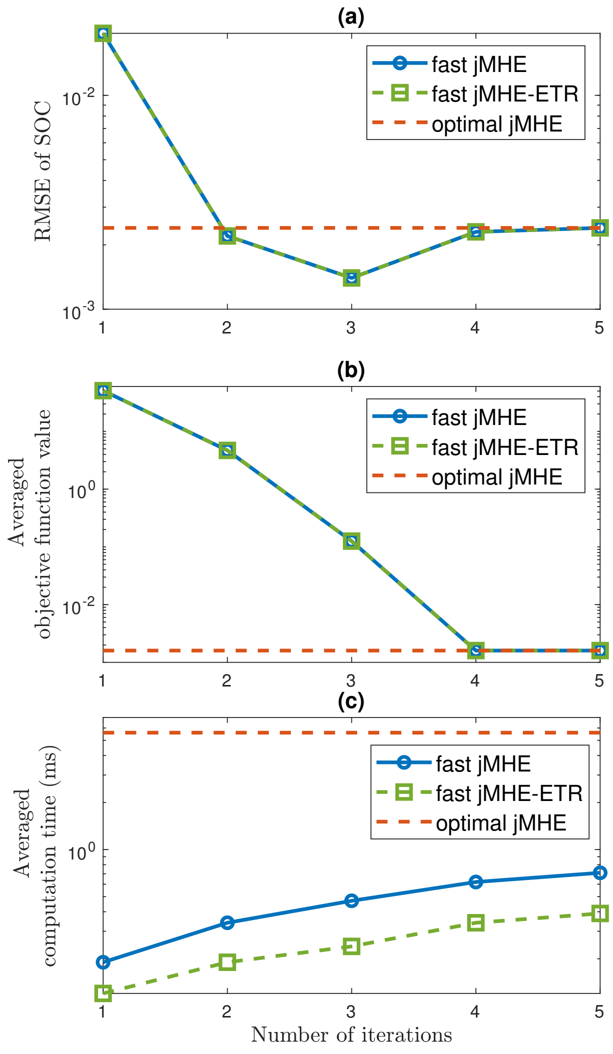


Figure 6: RMSE of SOC, averaged objective function value, and averaged computation time of the proposed fast jMHE and jMHE-ETR with different number of GN iterations under the BJDST condition, tested on a laptop computer. The initial SOC guess is set to $\widehat{Z}_0 = 0.4$, and the horizon length is $N = 3$.

Table 3: Computational costs of implemented algorithms (with $\widehat{Z}_0 = 0.4$) under the BJDST condition, tested on a laptop computer. The horizon lengths of the optimal jMHE, fast jMHE, and fast jMHE-ETR are all set to $N = 3$.

Computation time (ms)	**jEKF**	**optimal jMHE**	**fast jMHE** $m = 3$	**fast jMHE-ETR** $m = 3, \varepsilon = 10^{-2}$
Average	0.0133	5.60	0.42	0.31
Worst-case	0.62	74.10	2.80	1.90

By increasing the triggering threshold ε, fewer number of relinearizations are performed during GN iterations, as illustrated in Fig. 7. It can be seen that relinearizations are less frequently triggered at the 2nd and 3nd iterations than at the 1st iteration, and the triggered relinearizations become rather sparse along with time when the threshold ε increases to 0.01 and 0.1. Consequently, the averaged computation time monotonically decreases with ε, while the obtained RMSE slightly increases or remains almost the same, as demonstrated in Fig. 8.

4.5. Performance evaluation on a Raspberry Pi microcontroller

The results in Sections 4.2–4.4 are obtained by implementation with `MATLAB` on a laptop computer. To further evaluate the performance of all aforementioned algorithms on a low-cost microcontroller, we use `OCTAVE` on the Raspberry Pi 4 Model B, and present the results in Tables 4 and 5. As can be seen from Tables 2 and 4, the RMSEs of SOC on the Raspberry Pi microcontroller are at the same level as that produced by the laptop computer, and the differences between them are mainly due to different numerical routines of `OCTAVE` and `MATLAB`. As expected, the Raspberry Pi microcontroller is significantly slower than the laptop computer in terms of computation speed, according to Tables 3 and 5. On the Raspberry Pi microcontroller, compared to the optimal MHE, the proposed fast jMHE still reduces the computation time by an order of magnitude, and the proposed fast jMHE-ETR further reduces the computation time per sample to less than 10ms.

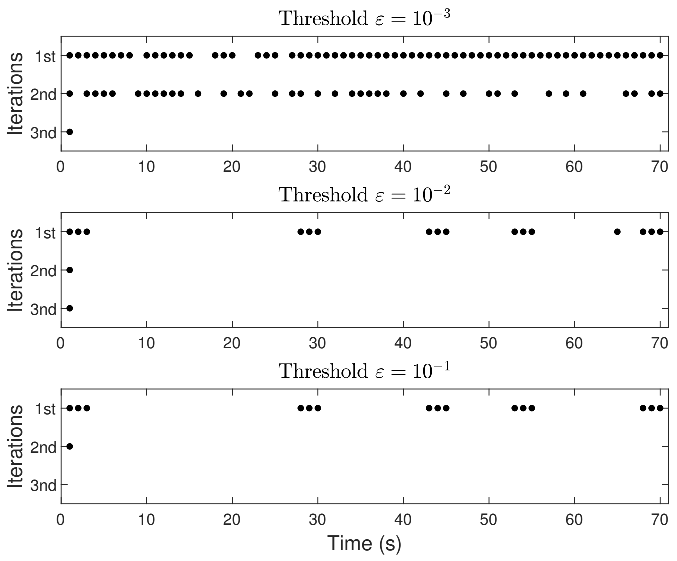


Figure 7: Relinearizations during the first 70s under the BJDST condition when implementing fast jMHE-ETR ($N = 3, m = 3$) with different values of ε, tested on a laptop computer. The initial SOC is set to $\widehat{Z}_0 = 0.2$, and 3 GN iterations per sample are implemented. A triggered relinearization is indicated by a solid black dot.

5. Conclusion

In this paper, we propose fast moving horizon SOC estimation using an SOC-dependent ECM subject to model mismatch, which aims at bringing the benefits of MHE to an embedded BMS. Instead of fully solving the MHE problem to convergence over each time horizon, the proposed fast jMHE algorithm performs a fixed number of structure-exploiting GN iterations and introduces the event-triggered strategy to account for limited time allowed within each sampling interval. As demonstrated by results evaluated on both laptop and Raspberry Pi microcontroller, the proposed fast jMHE algorithm achieves faster convergence speed and higher estimation accuracy than the joint EKF, and reduces the computation time per sample by more than an order of magnitude compared to the optimal jMHE implementation. Our future work will focus on extensions to cope with different temperatures and aging conditions, and real-world validations on an embedded BMS platform.

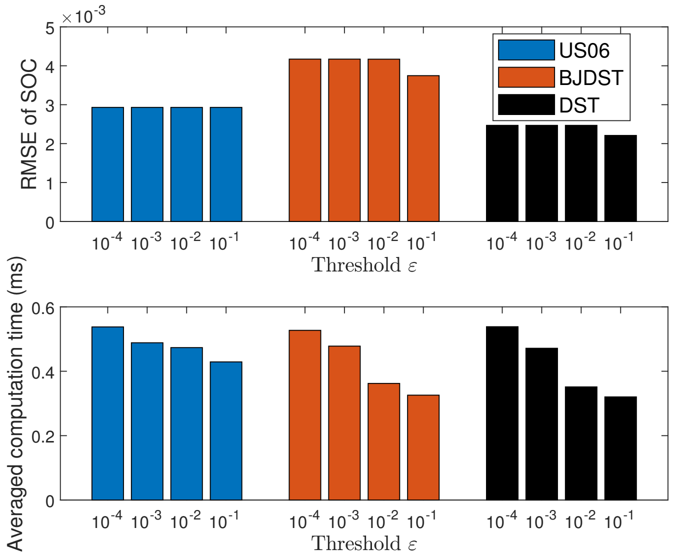


Figure 8: RMSE of SOC and averaged computation time of fast jMHE-ETR ($N = 3, m = 3$) with different values of ε, tested on a laptop computer. The initial SOC is set to $\widehat{Z}_0 = 0.2$.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 62373161); Hubei Provincial Natural Science Foundation for Innovation Groups (Grant No. 2021CFA026).

References

[1] X. Lin, Y. Kim, S. Mohan, J. B. Siegel, A. G. Stefanopoulou, Modeling and estimation for advanced battery management, Annual Review of Control, Robotics, and Autonomous Systems 2 (2019) 393–426.

[2] Z. Wei, J. Zhao, H. He, G. Ding, H. Cui, L. Liu, Future smart battery and management: advanced sensing from external to embedded multi-dimensional measurement, Journal of Power Sources 489 (2021) 229462.

[3] B. Yang, J. Wang, P. Cao, T. Zhu, H. Shu, J. Chen, J. Zhang, Z. J, Classification, summarization and perspectives on state-of-charge estimation

Table 4: RMSEs of SOC given by implemented algorithms (with $\widehat{Z}_0 = 0.4$) under US06, BJDST, and DST conditions, tested on Raspberry Pi 4 Model B. The horizon lengths of the optimal jMHE, fast jMHE, and fast jMHE-ETR are all set to $N = 3$.

RMSE of SOC	**US06**	**BJDST**	**DST**
jEKF	0.0135	0.0088	0.0081
optimal jMHE	0.0028	0.0040	0.0024
fast jMHE ($m = 3$)	0.0023	0.0030	0.0020
fast jMHE-ETR ($m = 3, \varepsilon = 10^{-2}$)	0.0023	0.0030	0.0020

Table 5: Computational costs of implemented algorithms (with $\widehat{Z}_0 = 0.4$) under the BJDST condition, tested on Raspberry Pi 4 Model B. The horizon lengths of the optimal jMHE, fast jMHE, and fast jMHE-ETR are all set to $N = 3$.

Computation time (ms)	**jEKF**	**optimal jMHE**	**fast jMHE**	**fast jMHE-ETR**
			$m = 3$	$m = 3, \varepsilon = 10^{-2}$
Average	1.30	166.71	14.20	9.70
Worst-case	5.51	198.75	25.32	21.20

of lithium-ion batteries used in electric vehicles: A critical comprehensive survey, Journa of Energy Storage 39 (2021) 102572.

[4] J. Chen, Z. Zhou, Z. Zhou, X. Wang, B. Liaw, Impact of battery cell imbalance on electric vehicle range, Green Energy and Intelligent Transportation 1 (3) (2022) 100025.

[5] J. Meng, M. Ricco, G. Luo, M. Swierczynski, D. I. Stroe, A. I. Stroe, R. Teodorescu, An overview and comparison of online implementable SOC estimation methods for lithium-ion battery, IEEE Transactions on Industry Applications 54 (2) (2017) 1583–1591.

[6] Y. Gao, G. L. Plett, G. Fan, X. Zhang, Enhanced state-of-charge estimation of LiFePO4 batteries using an augmented physics-based model, Journal of Power Sources 544 (2022) 231889.

[7] Y. Li, B. Xiong, D. M. Vilathgamuwa, Z. Wei, C. Xie, C. Zou, Constrained ensemble Kalman filter for distributed electrochemical state estimation of lithium-ion batteries, IEEE Transactions on Industrial Informatics 17 (1) (2021) 240–250.

[8] W. Li, Y. Fan, F. Ringbeck, D. Jöst, X. Han, M. Ouyang, D. U. Sauer, Electrochemical model-based state estimation for lithium-ion batteries with adaptive unscented Kalman filter, Journal of Power Sources 476 (2020) 228534.

[9] G. Fan, Systematic parameter identification of a control-oriented electrochemical battery model and its application for state of charge estimation at various operating conditions, Journal of Power Sources 470 (2020) 228153.

[10] L. Wu, H. Pang, Y. Geng, X. Liu, J. Liu, K. Liu, Low-complexity state of charge and anode potential prediction for lithium-ion batteries using a simplified electrochemical model-based observer under variable load condition, International Journal of Energy Research 46 (9) (2022) 11834–11848.

[11] X. Gu, K. See, Y. Liu, B. Arshad, L. Zhao, Y. Wang, A time-series Wasserstein GAN method for state-of-charge estimation of lithium-ion batteries, Journal of Power Sources 581 (2023) 233472.

[12] Y. Liu, Y. He, H. Bian, W. Guo, X. Zhang, A review of lithium-ion battery state of charge estimation based on deep learning: Directions for improvement and future trends, Journal of Energy Storage 52 (2022) 104664.

[13] M. Ragone, V. Yurkiv, A. Ramasubramanian, B. Kashir, F. Mashayek, Data driven estimation of electric vehicle battery state-of-charge informed by automotive simulations and multi-physics modeling, Journal of Power Sources 483 (2021) 229108.

[14] R. Zou, Y. Duan, Y. Wang, J. Pang, F. Liu, S. R. Sheikh, A novel convolutional informer network for deterministic and probabilistic state-of-charge estimation of lithium-ion batteries, Journal of Energy Storage 57 (2023) 106298.

[15] Q. Yu, Y. Liu, S. Long, X. Jin, J. Li, W. Shen, A branch current estimation and correction method for a parallel connected battery system based on dual BP neural networks, Green Energy and Intelligent Transportation 1 (2) (2022) 100029.

[16] Z. Deng, X. Hu, X. Lin, Y. Che, L. Xu, W. Guo, Data-driven state of charge estimation for lithium-ion battery packs based on Gaussian process regression, Energy 205 (2020) 118000.

[17] K.-J. Lee, W.-H. Lee, K.-K. K. Kim, Battery state-of-charge estimation using data-driven Gaussian process Kalman filters, Journal of Energy Storage 72 (2023) 108392.

[18] K. Fan, Y. Wan, B. Jiang, State-of-charge dependent equivalent circuit model identification for batteries using sparse Gaussian process regression, Journal of Process Control 112 (2022) 1–11.

[19] X. Hua, C. Zhang, G. Offer, Finding a better fit for lithium ion batteries: a simple, novel, load dependent, modified equivalent circuit model and parameterization method, Journal of Power Sources 484 (2021) 229117.

[20] P. Xu, X. Hu, B. Liu, T. Ouyang, N. Chen, Hierarchical estimation model of state-of-charge and state-of-health for power batteries considering current rate, IEEE Transactions on Industrial Informatics 18 (9) (2022) 6150–6159.

[21] F. Naseri, E. Schaltz, D. I. Stroe, A. Gismero, E. Farjah, An enhanced equivalent circuit model with real-time parameter identification for battery state-of-charge estimation, IEEE Transactions on Industrial Electronics 69 (4) (2021) 3743–3751.

[22] H. Beelen, H. J. Bergveld, M. Donkers, Joint estimation of battery parameters and state of charge using an extended Kalman filter: a single-parameter tuning approach, IEEE Transactions on Control Systems Technology 29 (3) (2020) 1087–1101.

[23] M. Hossain, M. Haque, M. T. Arif, Kalman filtering techniques for the online model parameters and state of charge estimation of the li-ion batteries: A comparative analysis, Journal of Energy Storage 51 (2022) 104174.

[24] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, D. U. Sauer, Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation, Journal of Energy Storage 30 (2020) 101557.

[25] H. Chen, E. Tian, L. Wang, S. Liu, A joint online strategy of measurement outliers diagnosis and state of charge estimation for lithium-ion batteries, IEEE Transactions on Industrial Informatics 19 (5) (2023) 6387–6397.

[26] Z. Yun, W. Qin, W. Shi, State of charge estimation of lithium-ion batteries with non-negligible outlier observations based on Student's-T filter, Journal of Energy Storage 55 (2022) 105825.

[27] Z. Liu, Z. Zhao, Y. Qiu, B. Jing, C. Yang, State of charge estimation for li-ion batteries based on iterative Kalman filter with adaptive maximum correntropy criterion, Journa of Power Sources 580 (2023) 233282.

[28] Z. Yun, W. Qin, W. Shi, State of charge estimation of lithium-ion battery under time-varying noise based on variational Bayesian estimation methods, Journal of Energy Storage 52 (2022) 104916.

[29] J. B. Rawlings, B. R. Bakshi, Particle filtering and moving horizon estimation, Computers & Chemical Engineering 30 (2006) 1529–1541.

[30] J. Shen, Y. He, Z. Ma, H. Luo, Z. Zhang, Online state of charge estimation of lithium-ion batteries: a moving horizon estimation approach, Chemical Engineering Science 154 (2016) 42–53.

[31] J. Shen, J. Shen, Y. He, Z. Ma, Accurate state of charge estimation with model mismatch for Li-ion batteries: a joint moving horizon estimation approach, IEEE Transactions on Power Electronics 34 (5) (2018) 4329–4342.

[32] H. Ren, H. Zhang, Z. Gao, Y. Zhao, A robust approach to state of charge assessment based on moving horizon optimal estimation considering battery system uncertainty and aging condition, Journal of Cleaner Production 270 (2020) 122508.

[33] J. Shen, Q. Wang, G. Zhao, Z. Ma, Y. He, A joint moving horizon strategy for state-of-charge estimation of lithium-ion batteries under combined measurement uncertainty, Journal of Energy Storage 44 (2021) 103316.

[34] Z. Wei, J. Hu, Y. Li, H. He, W. Li, D. U. Sauer, Hierarchical soft measurement of load current and state of charge for future smart lithium-ion batteries, Applied Energy 307 (2022) 118246.

[35] X. Hu, D. Cao, B. Egardt, Condition monitoring in advanced battery management systems: moving horizon estimation using a reduced electrochemical model, IEEE/ASME Transactions on Mechatronics 23 (1) (2017) 167–178.

[36] X. Hu, H. Jiang, F. Feng, B. Liu, An enhanced multi-state estimation hierarchy for advanced lithium-ion battery management, Applied Energy 257 (2020) 114019.

[37] Y. Chen, C. Li, S. Chen, H. Ren, Z. Gao, A combined robust approach based on auto-regressive long short-term memory network and moving horizon estimation for state-of-charge estimation of lithium-ion batteries, International Journal of Energy Research 45 (9) (2021) 12838–12853.

[38] E. D. R. Lopes, M. M. Soudre, C. H. Llanos, H. V. H. Ayala, Nonlinear receding-horizon filter approximation with neural networks for fast state of charge estimation of lithium-ion batteries, Journal of Energy Storage 68 (2023) 107677.

[39] A. Alessandri, M. Gaggero, Fast moving horizon state estimation for discrete-time systems using single and multi iteration descent methods, IEEE Transactions on Automatic Control 62 (9) (2017) 4499–4511.

[40] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, H. G. Bock, A real-time algorithm for moving horizon state and parameter estimation, Computers & Chemical Engineering 35 (1) (2011) 71–83.

[41] A. Wynn, M. Vukov, M. Diehl, Convergence guarantees for moving horizon estimation based on the real-time iteration scheme, IEEE Transactions on Automatic Control 59 (8) (2014) 2215–2221.

[42] A. Aravkin, J. V. Burke, L. Ljung, A. Lozano, G. Pillonetto, Generalized Kalman smoothing: Modeling and algorithms, Automatica 86 (2017) 63–86.

[43] K. Baumgärtner, J. Frey, R. Hashemi, M. Diehl, Zero-order moving horizon estimation for large-scale nonlinear processes, Computers & Chemical Engineering 154 (2021) 107433.

[44] M. Lenz, D. Jöst, F. Thiel, S. Pischinger, D. U. Sauer, Identification of load dependent cell voltage model parameters from sparse input data using the mixed integer distributed ant colony optimization solver, Journal of Power Sources 437 (2019) 226880.

[45] N. Haverbeke, Efficient numerical methods for moving horizon estimation, Diss., Katholieke Universiteit Leuven, Heverlee, Belgium (2011).

[46] J. Yan, S. Li, Y. Wan, Lithium-ion battery state-of-charge estimation using a real-time moving horizon estimation algorithm, in: Proceedings of 2021 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes, Chengdu, China, 2021.

[47] F. Zheng, Y. Xing, J. Jiang, B. Sun, J. Kim, M. Pecht, Influence of different open circuit voltage tests on state of charge online estimation for lithium-ion batteries, Applied Energy 183 (2016) 513–525.

[48] S. Du, Y. Wan, Code from: Towards fast embedded moving horizon state-of-charge estimation for lithium-ion batteries, Mendeley Data, V3 (2023). `doi:10.17632/3zv48n4jn5.3`.
URL `https://data.mendeley.com/datasets/3zv48n4jn5/3`