Tensor Network Based MIMO Volterra Model for Lithium-ion Batteries

Yangsheng Hu, Raymond A. de Callafon, Ning Tian, and Huazhen Fang

***Abstract*— Accurate battery modeling is fundamental for battery management system to function well and extract the full potential from a battery without violating constraints. In this paper, a Tensor Network (TN) based Volterra double-capacitor (VDC) model for lithium-ion batteries is developed to improve the prediction performance of the nonlinear double-capacitor (NDC) model. It is shown that the VDC model maintains the advantages of the NDC model to account for the rate capacity effect and the voltage recovery effect. In addition, the VDC model is capable of predicting both static and dynamic nonlinearities simultaneously in a more accurate way. To estimate the TN-cores in the VDC model, a Bond Core Sweeping Algorithm is proposed and shown to lead to a low-rank representation. A comparison based on experimental data demonstrates that the VDC model gives greater prediction accuracy than the NDC model and Thevenin model, showing significant promise to enhance future battery applications.**

***Index Terms*— Batteries, equivalent circuit model (ECM), system identification, tensor network (TN), Volterra model.**

I. Introduction

RECHARGEABLE batteries have become a major driver for applications ranging from portable consumer electronics to electric vehicles and microgrid applications. Among them, lithium-ion batteries (LiBs), having a high power/energy density, a long life span, high energy storage efficiency, and environment friendliness, are attracting more and more attention in both research and application fields [1]–[5]. There are roughly two major developed efforts in LiBs: one focuses on the battery cell design, materials selection for electrodes and electrolytes, pack structure optimization etc., to develop high-performance batteries; the other one implements advanced battery management systems (BMS) to maximize the utilization efficiency and safety for given types of batteries generally based on estimation of the State-of-Charge (SoC), State-of-Power (SoP), and State-of-Health (SoH).

Algorithms for BMS generally require mathematical models that describe battery physics and dynamics. The exisiting battery models mainly fall into two categories [6]: 1) electrochemical models, which leverage electrochemical principles to capture the dynamics of lithium-ion intercalation and diffusion as well as various associated processes [7]–[12], and 2) equivalent circuit models, which are a network of circuit components to describe a LiB's electrical behaviors [2], [13]–[18]. A brief comparison between the two models can be found in [18], where a nonlinear double-capacitor (NDC) model was proposed to bridge the gap between physical models and ECM. The NDC model is a modification of a linear double-capacitor model [19] while maintaining feature to account for rate capacity effects and energy recovery effects. The nonlinear mapping introduced in the NDC model is based on the observation of the nonlinear SoC-OCV (Open Circuit Voltage) curve. However, the SoC-OCV curve is a static feature for the LiBs fully at rest and does not reflect the full nonlinearity when the batteries are used dynamically. As shown in the validation experiment for the identification 2.0 approach in [18], the resulting SoC-OCV curve identified by the NDC model does not fully match the measured curve when the training process was trying to capture both the static and dynamic nonlinearities. Despite the less accurate static prediction, the dynamic prediction was still improved by using the NDC model. This observation provided the motivation for this paper to propose a new model to account for the two types of nonlinearities simultaneously.

Both electrochemical models and ECMs have physical explanations for the underlying processes happening inside of the battery during charging/discharging. However, due to the limited knowledge about the complicated dynamic processes involved in charging/discharging, it is impossible to model all the processes. Pure data-driven modeling approaches such as system identification or machine learning provide an alternative to first-principles modeling [20]–[24]. Instead of figuring out the detailed electrochemical and physical processes, data-driven modeling uncovers the underlying relationship between the historical input/output data from the perspectives of statistics and optimization. These techniques have been demonstrated effective in different battery modeling applications [25]–[31]. However, a large amount of data is generally required to guarantee the effectiveness of such models, which can easily suffer overfitting problems.

On the other hand, from the perspective of formulating a model, it will be very time-consuming to take into account a complicated mix of electrochemical reactions and side processes even if the expert knowledge is known. The prediction accuracy heavily relies on the accuracy of first-principles modeling and the estimation of the physical parameters is not trivial. However, machine learning methods make use of a wealth of data and can efficiently formulate an accurate model to learn the input/output relationships from practical

Yangsheng Hu and Raymond A. de Callafon are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92094-0411 USA (e-mail: yah071@ucsd.edu; callafon@ucsd.edu).

Ning Tian and Huazhen Fang are with the Department of Mechanical Engineering, University of Kansas, Lawrence, KS 66045 USA (e-mail: ning.tian@ku.edu; fang@ku.edu).

observations for effective prediction. From the perspective of computational complexity, a full-scale electrochemical model, e.g., the DFN model, is computationally expensive. However, a machine learning model can usually be formulated into a computationally efficient structure. This is very attractive for battery applications.

In this paper, a hybrid combination of an ECM and tensor network (TN) based supervised learning is used for battery modeling to enable physics-informed data-driven learning of the unknown nonlinear dynamics inherent to batteries. This hybrid structure harnesses the combined strengths of a physics-based model and a machine learning model. For the ECM part, the linear double-capacitor model is used to extract the voltage and SoC features within the battery. Note that the limitation is that the features generated by the model may not be accurate due to uncertainty of both modeling and measurement. These features are then fed into a supervised learning framework as shown in [32] to learn the nonlinear dynamics between the raw input features and output terminal voltage. However, compared to [32], this paper introduces a new modification that encodes the input features further into a feature map described by a Volterra model which can also be equivalently written into a TN representation for better computational efficiency. The reason for using TN representation is that it can mitigate the curse of dimensionality encounterd by a normal Volterra model with a large memory length and a high model degree [33]. Thus, a TN representation allows the Volterra model to capture more complex nonlinear dynamics from the numerical perspective. In addition, we propose the Bond Core Sweeping Algorithm, which uses the ε-truncated singular value decomposition (SVD) [34], to ignore less significant modes and seek the low-rank property intrinsic to data to avoid overfitting.

A. Statement of Contributions

This paper presents the following contributions.

First, a novel TN-based Volterra double-capacitor (VDC) model is developed. A linear double-capacitor model proposed in [19] with the currents as the input signals is used as an initial feature extractor. The raw features produced by the linear double-capacitor model are then fed into a supervised learning structure described by a TN-based Volterra model to predict the output terminal voltages. The proposed VDC model maintains some advantages of the linear double-capacitor model such as capturing the rate capacity effect and the voltage recovery effect. On the other hand, there are different sources of the dynamic nonlinearities, including but not limited to unmodeled dynamics varying with different SoC and C-rate. The linear double-capacitor model or the NDC model, as an approximation of the true dynamics inherent to a LiB, inevitably involves model mismatch. The proposed VDC model then has an advantage over the NDC model proposed in [18] in terms of accounting for both static and dynamic nonlinearities simultaneously in a more accurate way by incorporating a data-driven technique. In this paper, only discharging cases are considered. The VDC model will facilitate real-time voltage prediction with high accuracy and may benefit model-based algorithm design in BMS, including SoP prediction, charging control, etc. To our best knowledge, this study is the first to apply TN-based Volterra techniques in the application of battery modeling.

Second, a Bond Core Sweeping Algorithm is developed to estimate the TN-cores of the proposed VDC model. The proposed algorithm is developed by modifying the sweeping optimization algorithm in [32]. The first major difference is that a least squares technique is used to estimate each bond core instead of using a gradient descent algorithm. The reason is that the convergence of the gradient descent algorithm can be rather slow for a problem in which the gradient is rank deficient. This problem is encountered in our application. The second major difference is that an ε-truncated singular value decomposition is used when splitting the bond core into two individual TN-cores. The advantage is that the model complexity can be adjusted and simplified by emphasizing on the low-rank estimation. Seeking a low-rank estimation is important to avoid overfitting.

Third, experimental validation is performed to assess the performance of the proposed VDC model. The approach to generating the training data for a nonlinear dynamic battery modeling is presented. The effect of the current magnitude and SoC range swept by the training data on the model prediction is also investigated. A comparison of the VDC model with the NDC model is also given to show the efficacy of the proposed VDC model.

Note that this work is a significant extension of our previous work [35], in which the basic idea of using a TN-based Volterra model to do nonlinear mapping is illustrated. The parameter estimation for the TN representation of a more general multi-input multi-output (MIMO) Volterra model is developed in this paper. In addition, more experiments are performed to validate the proposed algorithm, and the effect of using partial data in a machine learning based method is also investigated.

B. Organization

The remaining part of this paper is organized as follows. Section II presents the TN representation of a MIMO Volterra model. The VDC model of battery dynamics and the corresponding parameter estimation are proposed in Section III. In addition, the approach to the acquisition of training data is investigated. In Section IV, experimental validation is performed to demonstrate the efficiency of the proposed model and algorithm. Section V summarizes this paper.

II. MIMO Volterra Model in the TN Representation

Denote the output signal as $\mathbf{y}(t) \in \mathbb{R}^m$ and the input signal $\mathbf{u}(t) \in \mathbb{R}^p$. The αth element of a vector $\mathbf{x}(t)$ is written as $x^{(\alpha)}(t)$. Assume that there is a nonlinear relationship between $\mathbf{u}(t)$ and $\mathbf{y}(t)$ that can be described by a MIMO Volterra model.

A normal description of a discrete-time p-input m-output Volterra system of degree d can be described as

$$\mathbf{y}(t) = \mathbf{h}_0 + \sum_{i=1}^{d} \sum_{k_1,\cdots,k_i=0}^{M-1} \sum_{\alpha_1,\cdots,\alpha_i=0}^{p-1} \mathbf{h}_{1:i}^{k,\alpha} \phi_{1:i}^{k,\alpha}(\mathbf{u}(t)) \quad (1)$$

where M is the memory length, $\phi_{1:i}^{k,\alpha}(\mathbf{u}(t)) \in \mathbb{R}$ and $\mathbf{h}_{1:i}^{k,\alpha} \in \mathbb{R}^m$ are the ith Volterra kernel function and corresponding kernel coefficient. Specifically, $\phi_{1:i}^{k,\alpha}$ and $\mathbf{h}_{1:i}^{k,\alpha}$ are the abbreviations of functions of $\{k_1, \alpha_1; k_2, \alpha_2; \cdots; k_i, \alpha_i\}$ such that

$$\phi_{1:i}^{k,\alpha}(\mathbf{u}(t)) = \prod_{j=1}^{i} u^{(\alpha_j+1)}(t - k_j) \quad (2)$$

$$\mathbf{h}_{1:i}^{k,\alpha} = \mathbf{h}_i(k_1, \alpha_1; k_2, \alpha_2; \cdots; k_i, \alpha_i) \quad (3)$$

The kernel function $\phi_{1:i}^{k,\alpha}(\cdot)$ is a multiplication of different degrees among the possibly different input components at possibly different time instants. All kernel functions serve as a feature map to map the input vector $\mathbf{u}(t)$ into a higher dimensional space, which helps create nonlinear mappings from the input $\mathbf{u}(t)$ to output $\mathbf{y}(t)$ by specifying the kernel coefficients. This allows one to capture high order nonlinearity and even coupled dynamics among different input channels and past states. Thus, this model structure has great potential for nonlinear system identification.

However, it should be noted that, for the generated high dimensional space, the number of all the kernel coefficients is $1 + \sum_{i=1}^{d} (pM)^i$ and suffers from the curse of dimensionality as the degree d goes up. Avoiding the choice of a large d prevents an overburdened storage requirement for a computer, but limits the model's capability of capturing complex and even coupled dynamics. On the other hand, even if the storage demand is not a problem, recording every detailed kernel coefficient may be unnecessary since not every kernel will be dominantly active in practice, which motivates us to seek a low-rank representation of (1).

TN representation can be used to address this issue. In the following discussion, tensors refer to multidimensional arrays. A d-way tensor is denoted using a boldface calligraphic letter as $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. Each entry of $\boldsymbol{\mathcal{T}}$ is denoted as $\boldsymbol{\mathcal{T}}^{(i_1 i_2 \cdots i_d)}$ via d integer indices $(i_1 i_2 \cdots i_d)$.

Definition 1: (The k-Mode Product [34]) For a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ and a matrix $\mathbf{U} \in \mathbb{R}^{p_k \times n_k}$, the k-mode product $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{T}} \times_k \mathbf{U}$ is defined such that

$$\boldsymbol{\mathcal{X}}^{(i_1 \cdots i_{k-1} j i_{k+1} \cdots i_d)} = \sum_{i_k=1}^{n_k} \mathbf{U}^{(j i_k)} \boldsymbol{\mathcal{T}}^{(i_1 \cdots i_{k-1} i_k i_{k+1} \cdots i_d)}$$

and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times p_k \times n_{k+1} \times \cdots \times n_d}$.

Definition 2: (Tensor Train (TT) Decomposition [34]) The d-way tensor $\boldsymbol{\mathcal{T}}$ can be represented by a linear TN such that

$$\boldsymbol{\mathcal{T}}^{(i_1 i_2 \cdots i_d)} = \sum_{\alpha_0,\cdots,\alpha_d} \boldsymbol{\mathcal{T}}_1^{(\alpha_0 i_1 \alpha_1)} \boldsymbol{\mathcal{T}}_2^{(\alpha_1 i_2 \alpha_2)} \cdots \boldsymbol{\mathcal{T}}_d^{(\alpha_{d-1} i_d \alpha_d)}$$

where $\boldsymbol{\mathcal{T}}_1, \cdots, \boldsymbol{\mathcal{T}}_d$ are called TT-cores. Each $\boldsymbol{\mathcal{T}}_k$ is a 3-way tensor of dimensions $r_{k-1} \times n_k \times r_k$, where r_{k-1}, r_k are called the TT-ranks and $r_0 = r_d = 1$.

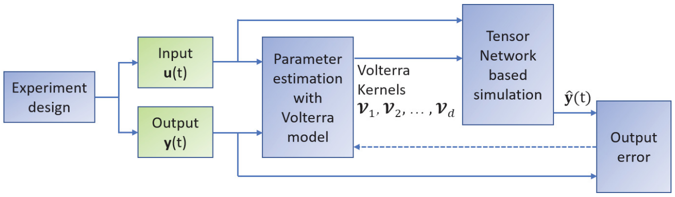

Experiment design
Input u(t)
Output y(t)
Parameter estimation with Volterra model
Volterra Kernels $\boldsymbol{\mathcal{V}}_1, \boldsymbol{\mathcal{V}}_2, \ldots, \boldsymbol{\mathcal{V}}_d$
Tensor Network based simulation
$\hat{\mathbf{y}}(t)$
Output error

Fig. 1: The illustration of identifying a TN based Volterra model.

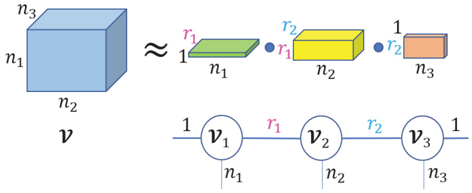

n_3
n_1
n_2
$\boldsymbol{\mathcal{V}}$
≈
r_1
1
n_1
r_2
r_1
n_2
1
r_2
n_3
1
$\boldsymbol{\mathcal{V}}_1$
r_1
$\boldsymbol{\mathcal{V}}_2$
r_2
$\boldsymbol{\mathcal{V}}_3$
1
n_1
n_2
n_3

Fig. 2: A linear Tensor Network.

Following the discussion and formulation in [33], one can incorporate all the kernel coefficients into a $(d+1)$-way Volterra tensor $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{(pM+1) \times \cdots \times (pM+1) \times m}$ such that the Volterra system output can be simulated as

$$\begin{aligned} \mathbf{y}^{\mathrm{T}}(t) &= \boldsymbol{\mathcal{V}} \times_1 \mathbf{u}_t^{\mathrm{T}} \times_2 \mathbf{u}_t^{\mathrm{T}} \cdots \times_d \mathbf{u}_t^{\mathrm{T}} \\ &= (\boldsymbol{\mathcal{V}}_1 \times_2 \mathbf{u}_t^{\mathrm{T}})(\boldsymbol{\mathcal{V}}_2 \times_2 \mathbf{u}_t^{\mathrm{T}}) \cdots (\boldsymbol{\mathcal{V}}_d \times_2 \mathbf{u}_t^{\mathrm{T}}) \end{aligned} \quad (4)$$

where

$$\mathbf{u}_t = \left[1, \mathbf{u}^{\mathrm{T}}(t), \mathbf{u}^{\mathrm{T}}(t-1), \cdots, \mathbf{u}^{\mathrm{T}}(t-M+1)\right]^{\mathrm{T}} \in \mathbb{R}^{pM+1}$$

and $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ are the TN-cores of the Volterra tensor $\boldsymbol{\mathcal{V}}$. The last core $\boldsymbol{\mathcal{V}}_d \in \mathbb{R}^{r_{d-1} \times (pM+1) \times m}$ and the remaining $\boldsymbol{\mathcal{V}}_i \in \mathbb{R}^{r_{i-1} \times (pM+1) \times r_i}$ with $r_0 = 1$. In fact, the TN-representation used here is a generalization of the TT-decomposition with $r_d = m$. Once the TN-cores are obtained, one can simulate the system using the second row of (4) without referring back to its dual representation $\boldsymbol{\mathcal{V}}$, which suffers from the curse of dimensionality.

The number of stored elements in the TN representation $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ is at a magnitude of $O((d-1)r^2(pM+1) + mr(pM+1))$, which will greatly reduce the storage requirement if the maximal TN-rank $r = \max\{r_1, \cdots, r_d\}$ is sufficiently small. Fortunately, a low value for the maximal TN-rank r is very common in practical applications, as is the case with battery modeling. Thus, (4) computed with the TN representation $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ serves as a low-rank representation of (1). The corresponding computational complexity of using (4) is approximately $O(d(pM+1)r^2)$.

The process of identifying a TN based Volterra model from input/output experimental data is illustrated in Fig. 1. An illustration of decomposing a simple 3-way tensor of dimension $n_1 \times n_2 \times n_3$ into a combination of three low-rank TN-cores is shown in Fig. 2.

Remark 1: In this paper, the second line in (4) will be adopted to simulate a TN-based Volterra model prediction $\hat{\mathbf{y}}(t)$ given estimated TN-cores $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ and input features $\mathbf{u}(t)$. Also, the formulation of our proposed algorithm in the following section allows the readers to rely on only the 2-mode tensor product throughout the paper in addition to basic linear algebra. For easy reference, $(\boldsymbol{\mathcal{V}}_i \times_2 \mathbf{u}_t^{\mathrm{T}}) \in \mathbb{R}^{r_{i-1} \times 1 \times r_i}$

can be squeezed into a matrix $\mathbf{M}_i \in \mathbb{R}^{r_{i-1} \times r_i}$ by ignoring the second dimension and each component is computed as

$$\mathbf{M}_i\left(\alpha_{i-1}, \alpha_i\right) = \sum_{k=1}^{pM+1} \mathbf{u}_t^{(k)} \boldsymbol{\mathcal{V}}_i^{(\alpha_{i-1} k \alpha_i)}$$

where $1 \leq \alpha_{i-1} \leq r_{i-1}$ and $1 \leq \alpha_i \leq r_i$. A more numerically efficient way is using matrix product during implementation as shown in Algorithm 1 following a MATLAB fashion.

Algorithm 1: 2-Mode Tensor Product

```
Input: TN-core $\boldsymbol{\mathcal{V}}_i$, input vector $\mathbf{u}_t$ in (4)
Output: The matrix representation $\mathbf{M}_i$ of $(\boldsymbol{\mathcal{V}}_i \times_2 \mathbf{u}_t^{\mathrm{T}})$
$\boldsymbol{\mathcal{V}}_i \leftarrow \mathrm{permute}(\boldsymbol{\mathcal{V}}_i, [2,1,3])$
$\mathbf{V}_i \leftarrow \mathrm{reshape}(\boldsymbol{\mathcal{V}}_i, [pM+1, r_{i-1} r_i])$
$\mathbf{M}_i \leftarrow \mathrm{reshape}(\mathbf{u}_t^{\mathrm{T}} \mathbf{V}_i, [r_{i-1}, r_i])$
```

III. Battery Modeling and Parameter Estimation

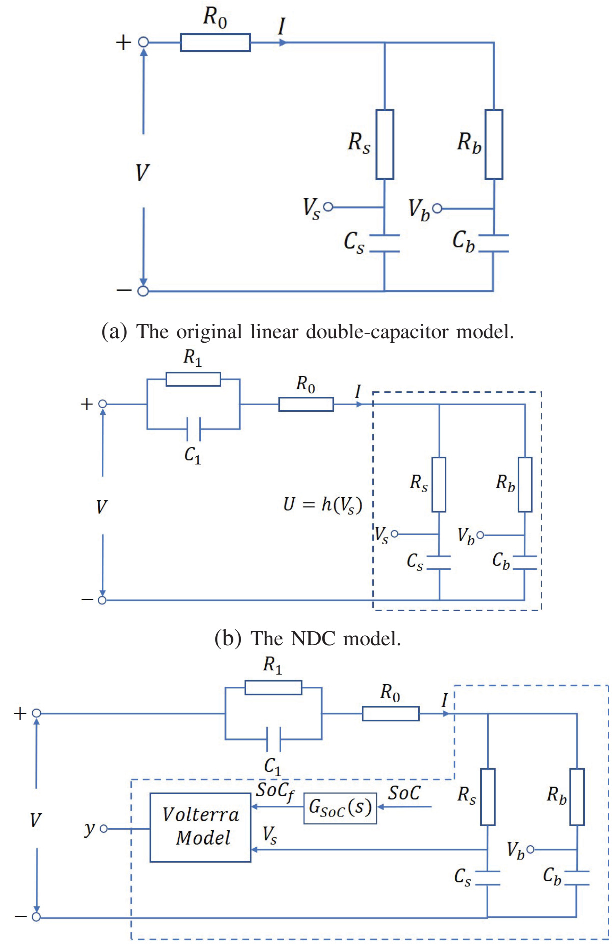
A. Linear and Nonlinear Double-Capacitor Model

The original linear double-capacitor model for a battery was proposed in [19] and consists of two serial Resistor-Capacitor circuits in parallel (i.e., R_b-C_b and R_s-C_s) and a resistor R_0, as shown in Fig. 3a. The resistor R_0 represents the collected instantaneous ohmic resistance. The R_s-C_s circuit corresponds to the electrode surface region exposed to the electrolyte, while the R_b-C_b to the bulk inner part of the electrode. The double-capacitor structure can capture both the rate capacity effect and the voltage recovery effect since it can model the charge migration between the near-surface and bulk inner domains of an electrode from the perspective of a single-particle model (SPM) [36]–[38].

However, the linear double-capacitor model is unable to describe a defining characteristic of batteries, e.g., the nonlinear SoC-OCV curve. Thus, the linear double-capacitor model is working well only around a certain SoC range so that the linear approximation is reasonable. In order to resolve this issue, the NDC model was proposed in [18], where a nonlinear mapping of V_s was introduced to approximate the SoC-OCV curve. A parallel RC circuit R_1-C_1 was also introduced there to account for the voltage transients related to the charge transfer on the electrode/electrolyte interface and the ion mass diffusion in the battery [39]. The NDC model is shown in Fig. 3b. Two scenarios can be considered for the parameter estimation of the NDC model: constant-current charging/discharging; variable-current charging/discharging. Experiment validation illustrated that the NDC model can have higher accuracy than other different ECMs under comparison. Also, the basic NDC model without the R_1-C_1 part is almost comparable to the full model, especially if the value of $R_1 C_1$ is small.

B. Hybrid Double-Capacitor Model Development

Despite the advantages of the NDC model, the validation results in [18] indicate a space for further improvements: the validation data show current-dependence of the parameters, which is not considered by the model; the NDC model is estimated independently for constant and variable current scenarios, and the model estimated for one scenario offers less predictive accuracy for the other; the SoC-OCV curve identification for the scenario of variable current matches the true one with relatively limited accuracy since the identifiability of this defining characteristic becomes less with many parameters including the nonlinear mapping to be estimated therein. As a final remark, the SoC-OCV curve is a static feature for batteries when they are fully at rest. On the other hand, if the nonlinear mapping is not optimized during training the model and simply fixed as the SoC-OCV curve even for the scenario of variable current, then the static prediction error at rest will be reduced. However, the nonlinear transient dynamics may not be well approximated since the nonlinear mapping is not optimized from the discharging data. Thus, the simple nonlinear mapping introduced in the NDC model may not fully account for the nonlinear transient dynamics. In addition, when the battery stops discharging, the true terminal voltage transient takes a long time to reach the full rest state whereas the predicted one by the NDC model arrives at the steady state almost instantaneously. This phenomenon implies



(a) The original linear double-capacitor model.

(b) The NDC model.

(c) The VDC model.

Fig. 3: Illustration of different double-capacitor models. The components within the rectangle are treated as a whole. In (b), $h(\cdot)$ is a nonlinear function. In (c), $G_{\mathrm{SoC}}(s)$ is a linear filter and y is the voltage prediction of the part within the rectangle.

that additional dynamics can be modeled within the battery after the discharging is stopped and before it achieves the fully steady state.

Despite the limitation of the nonlinear mapping within the NDC model, it still shows the potential to propose a new structure to capture the static and dynamic nonlinearities in a more accurate way. To improve the above mentioned points of the NDC model, a hybrid double-capacitor model is proposed in this paper as shown in Fig. 3c. A linear double-capacitor is used to extract meaningful features of the batteries and a multiple input single output (MISO) Volterra system is followed to synthesize all important features to predict the voltage. The proposed model is referred to as the VDC model in the following discussion.

The state-space equation of the linear part in Fig. 3b or Fig. 3c, which corresponds to a modified version of the linear double-capacitor model appended with one R-C component, is given by

$$\begin{bmatrix} \dot{V}_b(t) \\ \dot{V}_s(t) \\ \dot{V}_1(t) \end{bmatrix} = A \begin{bmatrix} V_b(t) \\ V_s(t) \\ V_1(t) \end{bmatrix} + BI(t) \tag{5}$$

similar to [18], where

$$A = \begin{bmatrix} \frac{-1}{C_b(R_b+R_s)} & \frac{1}{C_b(R_b+R_s)} & 0 \\ \frac{1}{C_s(R_b+R_s)} & \frac{-1}{C_s(R_b+R_s)} & 0 \\ 0 & 0 & \frac{-1}{R_1C_1} \end{bmatrix}, B = \begin{bmatrix} \frac{R_s}{C_b(R_b+R_s)} \\ \frac{R_b}{C_s(R_b+R_s)} \\ \frac{1}{C_1} \end{bmatrix}$$

Note that $I > 0$ for charging, $I < 0$ for discharging, $V_b = V_s = 1\,\mathrm{V}$ for $\mathrm{SoC} = 1$, $V_b = V_s = 0\,\mathrm{V}$ for $\mathrm{SoC} = 0$, and SoC is computed as

$$\mathrm{SoC} = \frac{C_bV_b + C_sV_s}{C_b + C_s} \tag{6}$$

Additional dynamics, which is not captured by the NDC model, can be attributed to the slow charge diffusion process inside the battery. The battery becomes fully at rest when the the process reaches its equilibrium. The additional and slow dynamics is not obvious by using the cycling data with a high switching frequency. This phenomenon can be modeled as SoC dynamics formulated as a first-order transfer function model

$$G_{\mathrm{SoC}}(s) = \frac{1}{\alpha s + 1} \tag{7}$$

shown in Fig. 3c. The time constant of the slow charge diffusion process is accounted for by the parameter α. The static gain of the first-order model is normalized to 1, since the SoC as a whole within the battery remains the same whereas it takes some time for all the charges to diffuse toward a steady state. From the electrochemical perspective, it describes the dynamics of the lithium concentration at the surface of the particle around the average lithium concentration in the solid. When the battery stops charging/discharging, R_s-C_s and R_b-C_b are in the same loop and V_s and V_b will reach the same equilibrium simultaneously. Thus, V_b provides no more dynamic information than V_s. The introduction of the filter (7) is necessary since it adds features for slow dynamics to the input of the Volterra model besides V_s. The time constant α requires to be adjusted during the parameter estimation. Denote the output of $G_{\mathrm{SoC}}(s)$ in (7) as $\mathrm{SoC}_f(t)$ at the time instant t.

The battery impedance is typically a function of SoC [40]. However, one can break it down into a constant part R_0 due to the materials of current collectors and the inside impedance which is related to the SoC. The inside impedance can be attributed to the SoC status.

The V_s is the reflection of the surface charge and related to the major part of the terminal voltage. Different current profile will affect the dynamic changing of V_s and SoC_f. Thus, the current dependence of the model parameters mentioned in the experimental validation in [18] can be described by the nonlinear interaction between V_s and SoC_f.

Finally, the voltage V_s and SoC dynamics SoC_f are selected as the input features for the nonlinear mapping described by the Volterra system. The transient voltage V_1 due to charge transfer and the constant resistance R_0 are separated from the nonlinear mapping. The input vector $\mathbf{u}(t)$ of the Volterra system is given as

$$\mathbf{u}(t) = [\mathrm{SoC}_f(t), V_s(t)]^{\mathrm{T}} \tag{8}$$

where $V_s(t)$ and $\mathrm{SoC}(t)$ are obtained by simulating the linear double-capacitor model in (5). One should be careful about the initial state when simulating $\mathrm{SoC}_f(t)$ using the filter $G_{\mathrm{SoC}}(s)$. For example, for a fully charged battery, the initial $\mathrm{SoC}(0) = 1$ and it should be set as the initial state of $G_{\mathrm{SoC}}(s)$ in $\mathrm{SoC}_f(t) = G_{\mathrm{SoC}}(s)\mathrm{SoC}(t)$. The other approach is to filter the signal $\mathrm{SoC}(t) - 1$ instead for a fully charged battery if the initial state of $G_{\mathrm{SoC}}(s)$ is set as zero, i.e., $\mathrm{SoC}_f(t) = G_{\mathrm{SoC}}(s)(\mathrm{SoC}(t) - 1)$.

Note that the VDC model consists of a linear double-capacitor model and a Volterra system. The linear double-capacitor model can be determined separately by parameter estimation, where R_0 along with other parameters within the linear model will be determined. The linear part is the same with the one in the NDC model and can be obtained from the NDC model identification results. Then, given the linear double-capacitor model and the input current signals, V_s and SoC can be computed, although their accuracy can be affected by uncertainty of both modeling and measurement. Finally, the Volterra system serves as a nonlinear correction to improve the voltage prediction.

The terminal voltage $V(t)$ consists of $V_1(t)$, $I(t)R_0$, and the output $\mathbf{y}(t)$ of the nonlinear mapping described by the Volterra system, i.e.,

$$\mathbf{y}(t) = V(t) - I(t)R_0 - V_1(t) \tag{9}$$

The Volterra model is capable of grasping the nonlinearity and complex coupling among the input features. TN-based representation makes it possible to capture nonlinearity of high degree while seeking the low rank to simplify the parameterization and prevent overfitting. Moreover, a Volterra system is always bounded-input bounded-output stable, allowing a reliable learning of parameters in battery applications.

Remark 2: It is worth coming up with a physics-informed data-driven model. First, the parameters can be learned efficiently from the practical data and the numerical complexity of simulation is guaranteed if the model structure is appropriately constructed. Second, adding physics-informed components makes the simulation more insightful and robust. Despite the potential of an electrochemical model for high-fidelity simulation, its physical parameters may not be identifiable from experiment data and the numerical complexity increases a lot if many processes are included. Also, it has no way to handle the uncertainty due to unmodeled mechanisms, which can be learned directly from practical data by data-driven techniques.

C. The Bond Core Sweeping Algorithm

Given the input features $\mathbf{u}(t)$ and the output prediction $\mathbf{y}(t)$, the next is to estimate the TN-cores $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ in (4). Although a MISO Volterra system identification is required for our application, the estimation algorithm will be presented in terms of a MIMO one for general purpose. The idea of sweeping optimization proposed in [32] is used in this paper. However, the inherent ill-condition feature of the Volterra structure in this paper will result in a very slow convergence rate for the gradient method, which is used during sweeping in [32]. Thus, the local optimization during sweeping will be modified to compute a least-squares solution to handle the ill-condition problem.

Note that the dual representation $\boldsymbol{\mathcal{V}}$, which suffers from the curse of dimensionality, will not be referred to during the whole estimation process. Instead, the TN-cores are optimized directly, and the TN-ranks grow and shrink adaptively during training to concentrate resources on the most useful correlations within the data for learning.

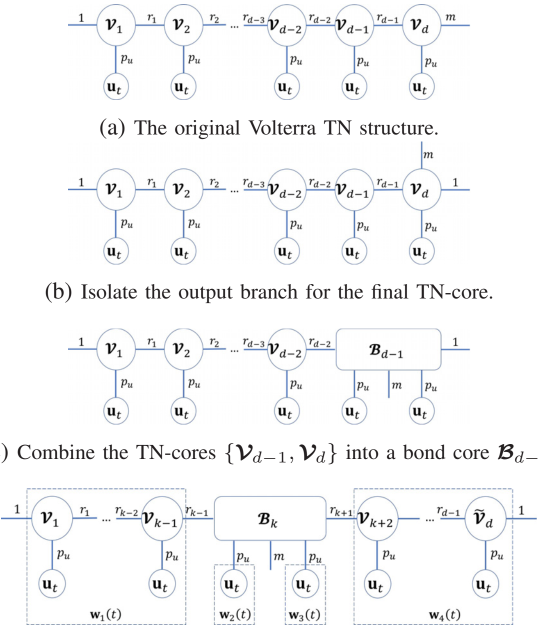
The cost function to be minimized is

$$J = \frac{1}{2}\sum_{t=1}^{N}\sum_{l=1}^{m}\left(f^{(l)}\left(\mathbf{u}\left(t\right)\right) - y^{(l)}\left(t\right)\right)^2 \qquad (10)$$

where $f^{(l)}(\mathbf{u}(t)) := \hat{y}^{(l)}(t)$ is the lth prediction output $\hat{\mathbf{y}}(t)$ of the proposed Volterra model in (4). The outline of the bond core sweeping algorithm is illustrated in Fig. 4. In order to adaptively estimate the internal TN-ranks, two neighboring TN-cores are merged into a bond core during each local optimization whereas the other TN-cores are fixed. The local optimization will sweep left and right to iteratively minimize the cost function. The output branch will be moving along with the bond core so that the parameters can be adjusted regarding different outputs during each local optimization.

The original representation of TN-cores are shown in Fig. 4a. The sweeping algorithm starts from the rightmost. The index order for each TN-core follows a counterclockwise direction. All the TN-cores are randomly initialized as left orthogonal to facilitate the numerical stability of the sweeping algorithm which starts from the right most [41]. A TN-core $\boldsymbol{\mathcal{V}}_k \in \mathbb{R}^{r_{k-1}\times p_u \times r_k}$ is left orthogonal if it can be reshaped into an $r_{k-1}p_u \times r_k$ matrix $\mathbf{V}_k$ for which $\mathbf{V}_k^{\mathrm{T}}\mathbf{V}_k = \mathbf{I}_{r_k}$.

Thus, the rightmost TN-core is firstly modified by adding one more dimension and isolating the output branch as shown in Fig. 4b. Then, the rightmost two TN-cores $\{\boldsymbol{\mathcal{V}}_{d-1}, \boldsymbol{\mathcal{V}}_d\}$ are merged to formulate a bond core $\boldsymbol{\mathcal{B}}_{d-1}$ as shown in Fig. 4c. The bond core parameters are estimated by minimizing the cost function (10) while fixing the other TN-cores. The estimated bond core should be split up back into two updated TN-cores. Similar steps are then implemented for the next two TN-cores $\{\boldsymbol{\mathcal{V}}_{d-2}, \boldsymbol{\mathcal{V}}_{d-1}\}$. When the bond core sweeping reaches the leftmost, the sweeping continues by changing to a right direction. The sweeping procedures go left and right for several iterations of local optimizations until the estimation error is below a given threshold. In order to maintain the original TN structure shown in Fig. 4a, the sweeping algorithm should also stop at the rightmost finally. Thus, the termination criterion for Algorithm 3 is: the cost in (10) for current estimate is smaller than a given threshold while the bond core to be updated at this iteration arrives at the right most, i.e., $k = d - 1$.


1
$\boldsymbol{\mathcal{V}}_1$
r_1
$\boldsymbol{\mathcal{V}}_2$
r_2
r_{d-3}
$\boldsymbol{\mathcal{V}}_{d-2}$
r_{d-2}
$\boldsymbol{\mathcal{V}}_{d-1}$
r_{d-1}
$\boldsymbol{\mathcal{V}}_d$
m
p_u
$\mathbf{u}_t$
$\boldsymbol{\mathcal{B}}_{d-1}$
r_{k-2}
$\boldsymbol{\mathcal{V}}_{k-1}$
r_{k-1}
$\boldsymbol{\mathcal{B}}_k$
r_{k+1}
$\boldsymbol{\mathcal{V}}_{k+2}$
$\widetilde{\boldsymbol{\mathcal{V}}}_d$
$\mathbf{w}_1(t)$
$\mathbf{w}_2(t)$
$\mathbf{w}_3(t)$
$\mathbf{w}_4(t)$

(a) The original Volterra TN structure.

(b) Isolate the output branch for the final TN-core.

(c) Combine the TN-cores $\{\boldsymbol{\mathcal{V}}_{d-1}, \boldsymbol{\mathcal{V}}_d\}$ into a bond core $\boldsymbol{\mathcal{B}}_{d-1}$.

(d) Four fixed parts $\{\mathbf{w}_1(t), \mathbf{w}_2(t), \mathbf{w}_3(t), \mathbf{w}_4(t)\}$ during the local optimization to estimate an intermediate bond core $\boldsymbol{\mathcal{B}}_k$. Note that the output branch is shifting along with $\boldsymbol{\mathcal{B}}_k$.

Fig. 4: Illustration of estimating the bond cores.

For simplicity of notations, denote $p_u = pM + 1$. In terms of an intermediate iteration shown in Fig. 4d where the TN-cores $\{\boldsymbol{\mathcal{V}}_k, \boldsymbol{\mathcal{V}}_{k+1}\}$ form the bond tensor $\boldsymbol{\mathcal{B}}_k \in \mathbb{R}^{r_{k-1}\times p_u \times m \times p_u \times r_{k+1}}$, the output prediction $\hat{\mathbf{y}}(t)$ can be computed as

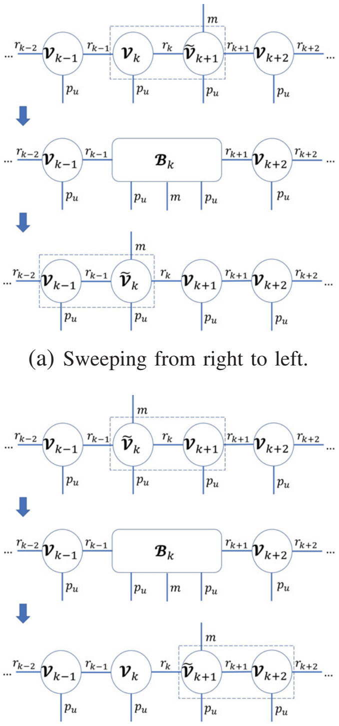
$$\hat{\mathbf{y}}^{\mathrm{T}}\left(t\right) = \boldsymbol{\mathcal{B}}_k \times_1 \mathbf{w}_1(t) \times_2 \mathbf{w}_2^{\mathrm{T}}(t) \times_4 \mathbf{w}_3^{\mathrm{T}}(t) \times_5 \mathbf{w}_4^{\mathrm{T}}(t) \qquad (11)$$

where

$$\begin{aligned} \mathbf{w}_1\left(t\right) &= \left(\boldsymbol{\mathcal{V}}_1 \times_2 \mathbf{u}_t^{\mathrm{T}}\right) \cdots \left(\boldsymbol{\mathcal{V}}_{k-1} \times_2 \mathbf{u}_t^{\mathrm{T}}\right) \\ \mathbf{w}_2\left(t\right) &= \mathbf{w}_3\left(t\right) = \mathbf{u}_t \\ \mathbf{w}_4\left(t\right) &= \left(\boldsymbol{\mathcal{V}}_{k+2} \times_2 \mathbf{u}_t^{\mathrm{T}}\right) \cdots \left(\widetilde{\boldsymbol{\mathcal{V}}}_d \times_2 \mathbf{u}_t^{\mathrm{T}}\right) \end{aligned} \qquad (12)$$

and $\widetilde{\boldsymbol{\mathcal{V}}}_d \in \mathbb{R}^{r_{d-1}\times p_u \times 1}$ is a modified final TN-core with the output branch on $\boldsymbol{\mathcal{V}}_d \in \mathbb{R}^{r_{d-1}\times p_u \times m}$ shifted to the bond core $\boldsymbol{\mathcal{B}}_k$.

The local optimization is to fix the other TN-cores $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_{k-1}, \boldsymbol{\mathcal{V}}_{k+2}, \cdots, \boldsymbol{\mathcal{V}}_d\}$ and estimate the bond core $\boldsymbol{\mathcal{B}}_k$ from (11). Due to the ill-condition feature of the Volterra



(a) Sweeping from right to left.

(b) Sweeping from left to right.

Fig. 5: Illustration of shifting the output branch during bond core sweeping. The rectangular box drawn with dashed lines is showing the pair of TN-cores to be merged into a bond core. A tilde sign will be added over the notation of the TN-core whose dimension is different from its original one.

model, the gradient method as in [32] is not used here. Instead, one can solve the local optimization via least squares. Similar ideas were also used in [33]. Denote $\mathcal{B}_k^{(l)} \in \mathbb{R}^{r_{k-1}\times p_u \times 1 \times p_u \times r_{k+1}}$ as the lth sub-tensor of $\mathcal{B}_k$ corresponding to the lth output, where $l = 1, 2, \cdots, m$. Also, denote the pseudo-inverse operator as $\mathrm{pinv}(\cdot)$. Then, $\mathcal{B}_k$ can be estimated as

$$\mathcal{B}_k\left(:, :, l, :, :\right) = \mathcal{B}_k^{(l)} \tag{13}$$

where

$$\begin{aligned}
&\mathcal{B}_k^{(l)} = \mathrm{reshape}(\mathrm{vec}(\mathcal{B}_k^{(l)}), [r_{k-1}, p_u, 1, p_u, r_{k+1}]) \\
&\mathrm{vec}(\mathcal{B}_k^{(l)}) = \mathrm{pinv}\left(\mathbf{W}\right)\mathbf{Y}_l \\
&\mathbf{W} = \begin{bmatrix} \mathbf{w}_4^{\mathrm{T}}(1) \otimes \mathbf{w}_3^{\mathrm{T}}(1) \otimes \mathbf{w}_2^{\mathrm{T}}(1) \otimes \mathbf{w}_1(1) \\ \mathbf{w}_4^{\mathrm{T}}(2) \otimes \mathbf{w}_3^{\mathrm{T}}(2) \otimes \mathbf{w}_2^{\mathrm{T}}(2) \otimes \mathbf{w}_1(2) \\ \vdots \\ \mathbf{w}_4^{\mathrm{T}}(N) \otimes \mathbf{w}_3^{\mathrm{T}}(N) \otimes \mathbf{w}_2^{\mathrm{T}}(N) \otimes \mathbf{w}_1(N) \end{bmatrix} \\
&\mathbf{Y}_l = \begin{bmatrix} y^{(l)}(1) \\ y^{(l)}(2) \\ \vdots \\ y^{(l)}(N) \end{bmatrix}
\end{aligned} \tag{14}$$

and N is the number of available training data points. Note that $\mathbf{W}$ only needs to be computed for once during each iteration.

Once the bond core $\mathcal{B}_k$ has been updated, one can split it up into two new TN-cores and move the output branch to the next TN-core. The SVD is used to collapse the bond core while the TN-rank r_k is updated by observing the number of dominant singular values. In order to determine the dominant singular values, an ε-truncated SVD [34], presented in Algorithm 2 following a MATLAB fashion, is performed with the sum of squares of the truncated singular values not being greater than a percentage of ε^2 of the total one. The next step is then to update the bond core $\mathcal{B}_{k-1}$ if sweeping left or $\mathcal{B}_{k+1}$ if sweeping right. The implementation of splitting up the bond core and shifting the output branch for both left and right sweeping is illustrated in Fig. 5, and the complete algorithm in a MATLAB fashion is summarized in Algorithm 3. Since the algorithm is set to stop sweeping at the rightmost TN-core, $\mathcal{V}_d$ should be permuted back to the original three-way tensor with TN-rank $r_d = m$ in the final step.

Algorithm 2: ε**-Truncated SVD**

```
Input: Matrix $\mathbf{B}$, expected amount $\varepsilon$ in percentage to
       be truncated
Output: Estimated rank $r$; truncated $\{\mathbf{L}_1, \mathbf{S}_1, \mathbf{Z}_1\}$ via
        SVD
$[\mathbf{L}, \mathbf{S}, \mathbf{Z}] \leftarrow$ SVD($\mathbf{B}$,'econ')
$r \leftarrow$ numerical rank determined by removing at most $\varepsilon$
  portion of insignificant singular values in $\mathbf{S}$ such that
  their sum of squares is not greater than $\varepsilon^2\|\mathbf{S}\|_F^2$
$\mathbf{L}_1 \leftarrow \mathbf{L}(:, 1:r)$
$\mathbf{S}_1 \leftarrow \mathbf{S}(1:r, 1:r)$
$\mathbf{Z}_1 \leftarrow \mathbf{Z}(:, 1:r)$
```

D. Acquisition of Training Data

We have proposed a VDC battery model to capture the complex and coupled dynamics and the Bond Core Sweeping Algorithm to estimate its parameters. In addition, the experiments to acquire the training data should be carefully designed to excite the major dynamics within the battery. An accurate prediction of the static SoC-OCV characteristic is also desired for the proposed model since the batteries are usually in a storage state for most of its life time in practice. Thus, we would like to train a model with both good dynamic and static predictions.

In order to achieve an accurate dynamic prediction, the model should be fed with data containing rich modes. This requirement is also known as persistent excitation in the field of system identification [20], [42]. A white excitation input signal sweeping the whole working range is desired but not practical in many applications. Instead, a relatively small data set with several different variable-current profiles are sufficient for the proposed VDC model, and this kind of training data is denoted as Dynamic Training Data (DTD).

The proposed VDC model is a hybrid model with both physically meaningful modeling and complexity-adjustable nonlinear mapping. One advantage of the Volterra structure is that the nonlinear mapping is always stable, and thus there is no need to worry about a divergent prediction during validation or in real-time battery monitoring. The adaptive adjustment

Algorithm 3: Bond Core Sweeping Algorithm

```
Input: $N$ input/output sampled data $\{\mathbf{u}(t), \mathbf{y}(t)\}$ in
       (8) and (9), memory length $M$, degree $d$,
       expected accuracy $\varepsilon$ in percentage
Output: TN-cores $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ in (4) minimizing
        (10)
Initialization: Construct $\mathbf{u}_t$ in (4); $r_0 \leftarrow 1, r_d \leftarrow m$;
  initialize left orthogonal TN-cores $\{\boldsymbol{\mathcal{V}}_1, \cdots, \boldsymbol{\mathcal{V}}_d\}$ of
  ranks 1, i.e., $r_i \leftarrow 1, i = 1, 2, \cdots, d-1$; the starting
  index $k \leftarrow d-1$; sweeping direction R2L ← 'left'
$\boldsymbol{\mathcal{V}}_d \leftarrow \mathrm{reshape}(\boldsymbol{\mathcal{V}}_d, [r_{d-1}, p_u, 1, m])$
$r_d \leftarrow 1$
while the termination criterion is not satisfied do
    Compute $\{\mathbf{w}_1(t), \mathbf{w}_2(t), \mathbf{w}_3(t), \mathbf{w}_4(t)\}$ in (12)
    for $l = 1, \cdots, m$ do
        $\boldsymbol{\mathcal{B}}_k(:,:,l,:,:) \leftarrow \boldsymbol{\mathcal{B}}_k^{(l)}$ as in (13)
    if R2L is 'left' then
        $\mathbf{B}_k \leftarrow \mathrm{reshape}(\boldsymbol{\mathcal{B}}_k, [r_{k-1} p_u m, p_u r_{k+1}])$
        $\{r, \mathbf{L}_1, \mathbf{S}_1, \mathbf{Z}_1\} \leftarrow \varepsilon$-truncated SVD on $\mathbf{B}_k$
        $r_k \leftarrow r$
        Split $\boldsymbol{\mathcal{B}}_k$ while keeping left orthogonal:
        $\boldsymbol{\mathcal{V}}_k \leftarrow \mathrm{reshape}(\mathbf{L}_1 \mathbf{S}_1, [r_{k-1}, p_u, m, r_k])$
        $\boldsymbol{\mathcal{V}}_{k+1} \leftarrow \mathrm{reshape}(\mathbf{Z}_1^{\mathrm{T}}, [r_k, p_u, r_{k+1}])$
        if $k > 1$ then
            $k \leftarrow k-1$
        else
            R2L←'right'
    else
        $\mathbf{B}_k \leftarrow \mathrm{reshape}(\boldsymbol{\mathcal{B}}_k, [r_{k-1} p_u, m p_u r_{k+1}])$
        $\{r, \mathbf{L}_1, \mathbf{S}_1, \mathbf{Z}_1\} \leftarrow \varepsilon$-truncated SVD on $\mathbf{B}_k$
        $r_k \leftarrow r$
        Split $\boldsymbol{\mathcal{B}}_k$ while keeping right orthogonal:
        $\boldsymbol{\mathcal{V}}_k \leftarrow \mathrm{reshape}(\mathbf{L}_1, [r_{k-1}, p_u, r_k])$
        $\boldsymbol{\mathcal{V}}_{k+1} \leftarrow \mathrm{reshape}(\mathbf{S}_1 \mathbf{Z}_1^{\mathrm{T}}, [r_k, m, p_u, r_{k+1}])$
        if $k < d-1$ then
            $k \leftarrow k+1$
        else
            R2L←'left'
            if The termination criterion is satisfied then
                break
$\boldsymbol{\mathcal{V}}_d \leftarrow \mathrm{permute}(\boldsymbol{\mathcal{V}}_d, [1, 3, 2, 4])$
$r_d \leftarrow m$
```

of the TN-ranks of the proposed model during training also reduces the overfitting problem since the low-rank feature within the data is explored. Thus, it is unnecessary to use a very large data set as in training pure black-box models such as deep neural networks.

In order to capture the static feature, the SoC-OCV information, which is obtained from specific tests, should also be fed into the VDC model. Thus, a characterization experiment should be performed to obtain the SoC-OCV curve, denoted as OCV $= h(\mathrm{SoC})$. Let $\{SoC_i, OCV_i\}, i = 1, \cdots, N_d$ be the N_d sampled points of the SoC-OCV curve. Then, the corresponding training data for the static feature can be constructed such that, for each pair $\{SoC_i, OCV_i\}$, the input/output data $\{\mathbf{u}(t), \mathbf{y}(t)\}$ in (8) and (9) is formulated as

$$
\begin{aligned}
V_s(t) &= \mathrm{SoC}_f(t) = SoC_i \\
\mathbf{y}(t) &= h(SoC_i)
\end{aligned}
\tag{15}
$$

for a time interval with certain duration, i.e., $t \in [t_i^L, t_i^R]$. The duration depends on how much emphasis we would like to put on the static feature during training. Since there are N_d pairs of $\{SoC_i, OCV_i\}$, there will be N_d such individual subintervals for (15). Denote this part of training data as Static Training Data (STD).

The reason why (15) reflects the static relation between the SoC and OCV is that the proposed VDC model is capable of describing the state of the battery when it is fully at rest. The battery will be at equilibrium when it is fully at rest after stopping charging/discharging. Thus, the current $I(t)$ and the transient voltage $V_1(t)$ will become zero and the other inner states $V_s(t)$, SoC(t), and SoC$_f(t)$ will become the same. This inherent property makes the VDC model more powerful in simultaneously capturing both dynamic and static features of the battery.

The benefit of using a VDC model is the capability to use the strong representation power to capture complex nonlinear dynamics all in one, which include not only SoC-OCV information but also dynamic nonlinearities and other possible effects eluding first-principles characterization. It directly makes use of data, which relies on the practical observation and is more efficient. It should be noted that the SoC-OCV information used here was obtained from a specific battery characterization experiment [18].

In sum, two types of training data are required: DTD and STD. The first part focuses on the dynamics and the second part focuses on the static feature. The STD is constructed based on the SoC-OCV curve and its duration can be selected based on how much weight to put on the static feature.

IV. Experimental Validation

This section presents the experimental validation of the proposed TN-based VDC model and the comparison with the other relevant methods. All the experiments in this section were conducted on a PEC SBT4050 battery tester. It can support charging/discharging with arbitrary current-, voltage-, or power- based loads up to 40 V and 50 A. A dedicated server prepares and configures a test offline and collects sampled experimental data online via the associated software, LifeTest. Charging/discharging tests were performed on a Panasonic NCR18650B lithium-ion battery cell, which was also used in [18], to generate experimental data. Note that the battery cell was set to operate between 3.2 V (fully discharged) and 4.2 V (fully charged). In this paper, only the discharging case is considered.

Two perspectives of validation are performed in the rest of this section. The first one shows the arrangement of training and validation data, the parameter estimation of TN-cores via the Bond Core Sweeping Algorithm, and the prediction accuracy of the estimated VDC model. The second one shows the requirements about training data to ensure the success of the identification algorithm, serving as a note for practitioners.

A. Training and Validation Results

The proposed hybrid VDC model consists of a linear double-capacitor model and a Volterra model. The parameters of the linear double-capacitor can be directly identified or using the linear part extracted from the identified NDC model in [18]. Note that the linear double-capacitor model here serves as a feature extractor to produce useful features that are fed into the Volterra system. Thus, slight distortion of using a different identified linear model is not a problem and will be accounted for during training of the Volterra model. As presented in Section III-D, two types of training data, i.e., DTD and STD, are required.

In order to persistently excite the VDC model, the dynamic modes in the training data should be sufficiently rich. Thus, variable-current discharging data should be included. Since the VDC model describes the nonlinear dynamics among the current $I(t)$, the voltage $V_s(t)$, and SoC dynamics $\mathrm{SoC}_f(t)$, the training data should sweep across the preset working range of current and SoC. Otherwise, the prediction of the nonlinear model outside the uncovered working range will show unexpected behaviors due to unmodeled dynamics. This phenomenon will be illustrated in Section IV-B.

In order to maintain the static SoC-OCV feature of the lithium-ion battery, additional training data described in (15) should also be included. This portion of training data will guarantee that the voltage prediction of the VDC model arrives at the corresponding value according to the SoC-OCV curve when the battery is fully at rest.

In order to better describe the low-frequency dynamics, at least one constant-current discharging data set should be included in the training data. The reason is that the variable-current discharging data focuses on the high frequency dynamics due to fast switching and thus contains little information about the modes lying in the low-frequency band.

Finally, the training data should include three parts: DTD1 (variable-current profile for high frequency dynamics), STD (SoC-OCV characteristics for static features), and DTD2 (constant-current profile for low frequency dynamics).

In our experiment, DTD1 consists of two variable-current profiles (0∼3 A and 0∼6 A). Note that the second one sweeps mainly across about 2∼6A. Thus, the two profiles allow the nonlinear dynamic modeling for working range 0∼6 A. STD describing the static features is formulated as in (15) with $\{SoC_i, OCV_i\}, i = 1, \cdots, 21$ being the 21 equidistantly data points sampled every 5% SoC from the SoC-OCV curve and the duration $t_i^R - t_i^L = 500$ sec. DTD2 consists of one constant-current profile (3.5 A).

For comparison, a Thevenin model [43] and an NDC model were identified along with the proposed VDC model. The adopted Thevenin model consists of one nonlinear OCV source, one internal resistance R_0^{Tvn} and one parallel R_1^{Tvn}-C_1^{Tvn} circuit same as in the NDC model for comparison. The estimated parameters are $R_0^{Tvn} = 0.0759\ \Omega$, $R_1^{Tvn} = 0.0390\ \Omega$, $C_1^{Tvn} = 2,476$ F. The identified NDC model will provide parameters for the double-capacitor part in the VDC model. The resulting physical parameters were estimated by the 2.0 identification approach in [18] with the nonlinear function fixed as the SoC-OCV curve and are given as follows: $C_b = 10,905\,\mathrm{F}$, $C_s = 1.462\,\mathrm{F}$, $R_b = 0.0728\,\Omega$, $R_s = 0\,\Omega$, $R_1 = 0.0413\,\Omega$, $C_1 = 990\,\mathrm{F}$, and $R_0 = 0.0113\,\Omega$. These parameters determine the feature extractor described by the linear double-capacitor model in (5). Then, the TN-based Volterra model with input/output $\{\mathbf{u}(t), \mathbf{y}(t)\}$ defined in (8) and (9) was estimated using the Bond Core Sweeping Algorithm proposed in Section III-C with memory length $M = 3$, degree $d = 5$, filter time constant $\alpha = 1/0.003$, and $\varepsilon = 0.4$. Note that the truncation factor ε is adjusted to seek the low-rank estimation as much as possible while maintaining the prediction accuracy. This procedure will prevent the resulting model from being overfitted. In practice, one should increase ε if the available dynamic modes are getting richer with increasing model complexity. The estimated TN-ranks are illustrated in Table I. The training algorithm takes about 23 seconds on an Intel Core i5-10210U CPU with 16 GB RAM. The training results are shown in Fig. 6, Fig. 7, Fig. 8, and Fig. 9. The RMSEs were computed for the dynamic voltage prediction while the battery is discharging.

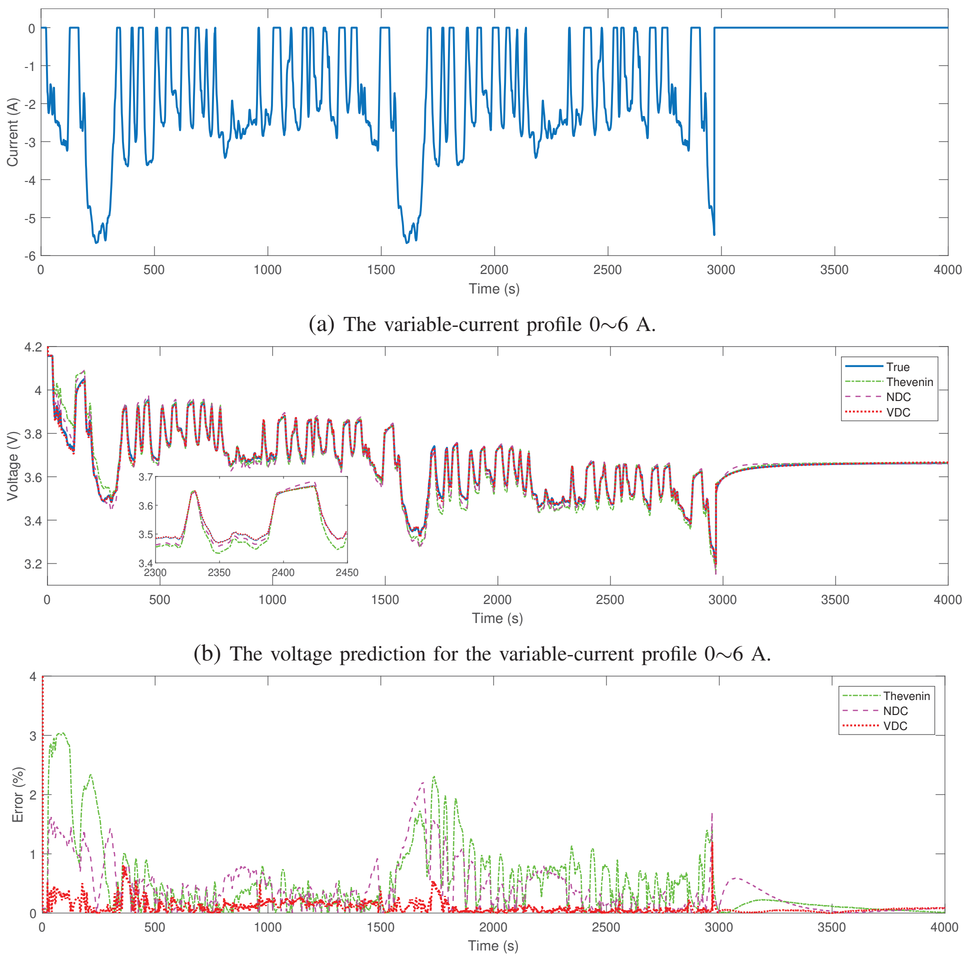
TABLE I: Estimated TN-ranks for $\{\mathcal{V}_1, \cdots, \mathcal{V}_5\}$.

TN-rank	$\widehat{r}_1$	$\widehat{r}_2$	$\widehat{r}_3$	$\widehat{r}_4$
value	4	7	6	7

For variable-current profiles, the VDC model produces the best voltage prediction both during cycling and resting. The NDC model uses a static nonlinear mapping estimated to balance the trade-off between the nonlinear dynamics and static SoC-OCV relationship. Despite the improvement in the dynamic prediction, the trade-off is obvious: the static nonlinear mapping cannot account for both dynamic and static nonlinearities simultaneously. Thus, the NDC model shows better prediction accuracy than the Thevenin model during discharging, whereas it is still less accurate than the VDC model due to the aforementioned tradeoff. Fig. 8 shows the prediction of SoC-OCV relationship using VDC model. It illustrates that the VDC model can be used to produce a more accurate dynamic voltage prediction without deteriorating the static prediction of the SoC-OCV feature. For the constant-current profile (3.5 A), one can observe that the Thevenin model is doing its best to minimize the error during discharging but is limited due to the lack of model fidelity. The VDC model is the best and can accurately capture the diffusion dynamics during the idling period.

The validation data consists of four constant-current profiles (1 A, 1.5 A, 2 A, 2.5 A) and one variable-current profile (1.5∼2.5 A). The prediction accuracy for the validation data is shown in Fig. 10 and Fig. 11.

For the constant-current profile in the validation data, the VDC model shows the best voltage prediction for all four cases in the following aspects as shown in Fig. 10: the dynamic nonlinearities are being accurately predicted during discharging; the transient diffusion dynamics are being accurately tracked during the idling period; the errors of static voltage predictions when at rest are small. The Thevenin model struggles to predict the true dynamics. For the variable-current profile in the validation data, the VDC model performs the best during



(a) The variable-current profile 0∼6 A.

(b) The voltage prediction for the variable-current profile 0∼6 A.

(c) Fitting error in percentage.

Fig. 6: Illustration of the voltage prediction by the proposed model for the variable-current profile 0∼6 A in the training data set DTD1. The RMSEs for different models are: 33.2mV (Thevenin); 20.8mV (NDC); 5.81mV (VDC).

both the discharging period and idling period. Similar to the training result, the NDC model is better in dynamic prediction than the Thevenin model.

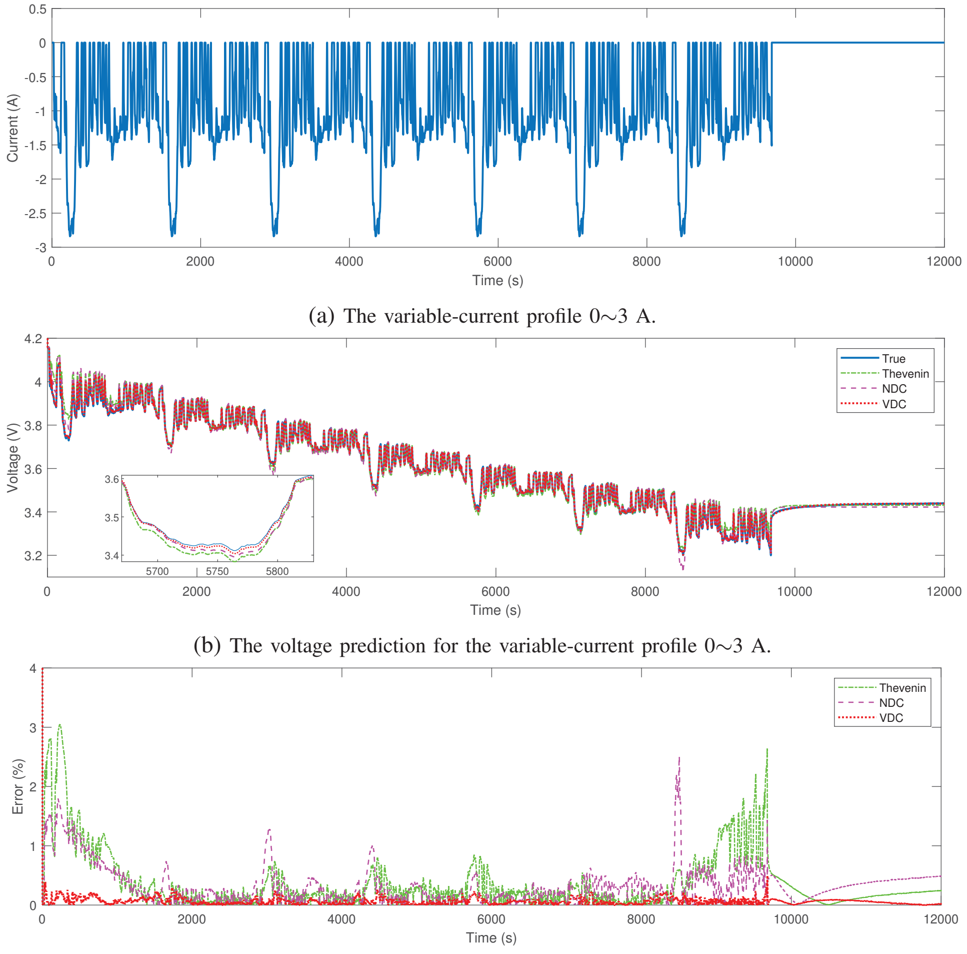
Remark 3: All models are determined and fixed for the whole SoC range. A Thevenin model with parameters identified for different SoCs may improve the accuracy of the model depending on how well the following assumption holds: The dynamics around a specific SoC and C-rate level can be well approximated by a linear model. In addition, one needs to perform additional HPPC test to produce the data to identify such an SoC-dependent model. The proposed VDC model captures these nonlinear dynamic behavior directly from the cycling data instead of performing linearization around an equilibrium.

B. The Effect of Training Data Range

Section IV-A has demonstrated the efficacy of the proposed method. Since nonlinear dynamics is modeled, it should be mentioned that the VDC model, due to its data-driven nature, will only capture the nonlinearity within the observed range of input current magnitudes and SoC values. Thus, it is necessary to select the training data sweeping across the normal working range. The nonlinearity is directly related to the input current magnitude and the battery dynamics keeps changing along with the SoC. For this reason, the current profiles should cover those common magnitudes in practice, and the training data should spread from full charge to almost empty.

In order to demonstrate the effect of training data range, the investigation of parameter estimation was performed on the two variable-current profiles in DTD1. The 0∼3 A variable-current profile spreads from SoC $= 100\%$ to SoC $= 14\%$ (almost empty). The 0∼6 A variable-current profile spreads from SoC $= 100\%$ to SoC $= 47\%$. The STD and the experimental data corresponding to 0∼6 A were used to train the TN Volterra model. The other one (0∼3 A) was used for validation. The results are shown in Fig. 12.

For the stage when the SoC is above 47%, the voltage prediction is good. However, when the SoC drops below 47%, the voltage prediction gets worse for the VDC model. The reason is that no training data sweeps below SoC $=$ 47%. Thus, the VDC model is not persistently excited in the SoC range below 47%. In other words, the poor prediction accuracy below 47% is due to the lack of informative data for that operating range. This observation also indicates a common



(a) The variable-current profile 0∼3 A.

(b) The voltage prediction for the variable-current profile 0∼3 A.

(c) Fitting error in percentage.

Fig. 7: Illustration of the voltage prediction of the proposed model for the variable-current profile 0∼3 A in the training data set DTD1. The RMSEs for different models are: 24.6mV (Thevenin); 17.2mV (NDC); 3.15mV (VDC).

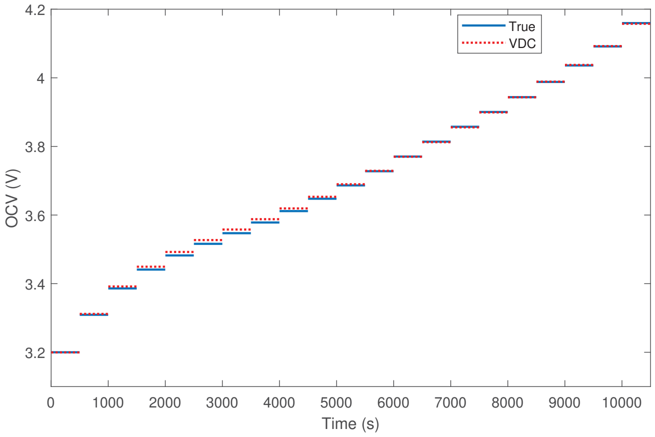


Fig. 8: The model prediction of SoC-OCV relationship for the training data set STD.

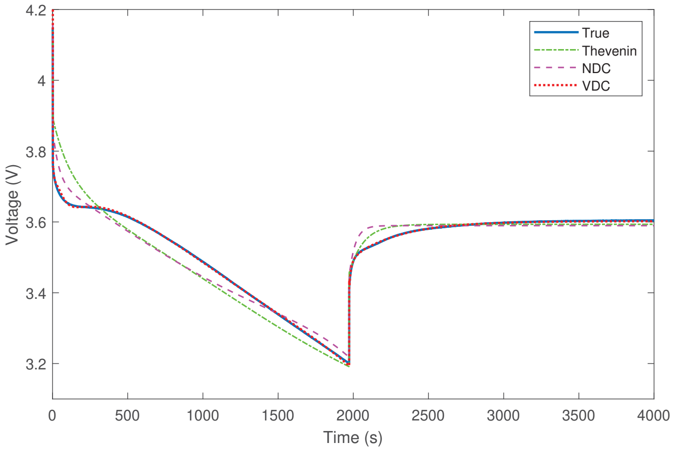


Fig. 9: The voltage prediction for the constant-current profile (3.5 A) in the training data set DTD2.

and important need when performing nonlinear modeling: the training data should cover the normal working range of the batteries.

V. Conclusion

This paper proposes a novel TN-based VDC model. The VDC model consists of two parts: a linear double-capacitor model and a TN-based Volterra model to capture nonlinear

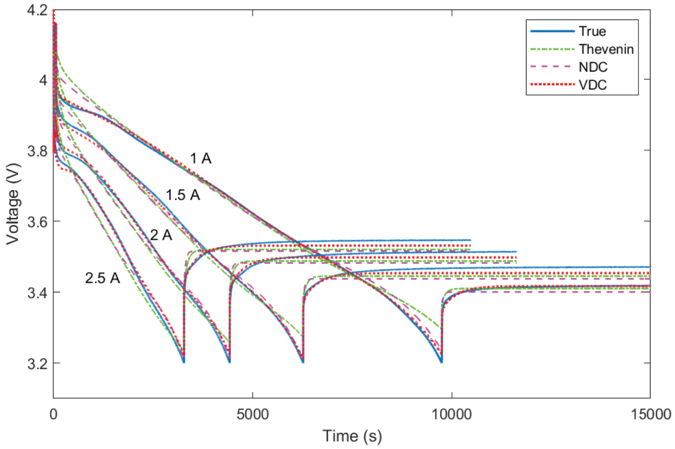


Fig. 10: The voltage prediction for the constant-current profiles (1 A, 1.5A, 2 A, 2.5A) in the validation data set.

dynamics. The proposed VDC model has the advantage to model both static and dynamic nonlinearities simultaneously in a more accurate way. It is by nature an SoC-dependent model, which will capture the changes in the battery dynamics as the SoC varies. Parameters of the VDC model are estimated via the Bond Core Sweeping Algorithm. It allows the Volterra model to seek a low-rank representation during training which also reduces the overfitting problem. The experimental results show that the VDC model produces a much more accurate voltage prediction than the Thevenin model and NDC model. The prediction error of the proposed model can be less than 0.5% as shown in the experimental validation. It is illustrated that the TN-based VDC model can serve as a powerful tool in modeling unknown and complex nonlinearities within the batteries using a data-driven model. Future work can be investigating using electrochemical models to produce useful features to feed into a Volterra model. This is attractive since the electrochemical part will provide insights into the true battery states and the Volterra part will help do nonlinear correction to account for unmodeled dynamics and approximation errors from the electrochemical part with a data-driven technique.

References

[1] J. B. Goodenough and K.-S. Park, “The li-ion rechargeable battery: a perspective,” *Journal of the American Chemical Society*, vol. 135, no. 4, pp. 1167–1176, 2013.

[2] G. L. Plett, *Battery management systems, Volume II: Equivalent-circuit methods*. Artech House, 2015.

[3] Y. Liang, C.-Z. Zhao, H. Yuan, Y. Chen, W. Zhang, J.-Q. Huang, D. Yu, Y. Liu, M.-M. Titirici, Y.-L. Chueh *et al.*, “A review of rechargeable batteries for portable electronic devices,” *InfoMat*, vol. 1, no. 1, pp. 6–32, 2019.

[4] B. Scrosati, J. Garche, and W. Tillmetz, *Advances in battery technologies for electric vehicles*. Woodhead Publishing, 2015.

[5] D. Tenfen, E. C. Finardi, B. Delinchant, and F. Wurtz, “Lithium-ion battery modelling for the energy management problem of microgrids,” *IET Generation, Transmission & Distribution*, vol. 10, no. 3, pp. 576–584, 2016.

[6] G. L. Plett, *Battery management systems, Volume I: Battery modeling*. Artech House, 2015.

[7] S. Santhanagopalan, Q. Guo, P. Ramadass, and R. E. White, “Review of models for predicting the cycling performance of lithium ion batteries,” *Journal of power sources*, vol. 156, no. 2, pp. 620–628, 2006.

[8] D. Di Domenico, A. Stefanopoulou, and G. Fiengo, “Lithium-ion battery state of charge and critical surface charge estimation using an electrochemical model-based extended kalman filter,” *Journal of dynamic systems, measurement, and control*, vol. 132, no. 6, 2010.

[9] J. C. Forman, S. J. Moura, J. L. Stein, and H. K. Fathy, “Genetic identification and fisher identifiability analysis of the doyle–fuller–newman model from experimental cycling of a lifepo4 cell,” *Journal of Power Sources*, vol. 210, pp. 263–275, 2012.

[10] G. K. Prasad and C. D. Rahn, “Model based identification of aging parameters in lithium ion batteries,” *Journal of power sources*, vol. 232, pp. 79–85, 2013.

[11] J. Li, K. Adewuyi, N. Lotfi, R. G. Landers, and J. Park, “A single particle model with chemical/mechanical degradation physics for lithium ion battery state of health (soh) estimation,” *Applied energy*, vol. 212, pp. 1178–1190, 2018.

[12] C. Zou, C. Manzie, and D. Nešić, “Model predictive control for lithium-ion battery optimal charging,” *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 947–957, 2018.

[13] X. Hu, S. Li, and H. Peng, “A comparative study of equivalent circuit models for li-ion batteries,” *Journal of Power Sources*, vol. 198, pp. 359–367, 2012.

[14] X. Lin, H. E. Perez, S. Mohan, J. B. Siegel, A. G. Stefanopoulou, Y. Ding, and M. P. Castanier, “A lumped-parameter electro-thermal model for cylindrical batteries,” *Journal of Power Sources*, vol. 257, pp. 1–11, 2014.

[15] A. Farmann and D. U. Sauer, “A comprehensive review of on-board state-of-available-power prediction techniques for lithium-ion batteries in electric vehicles,” *Journal of Power Sources*, vol. 329, pp. 123–137, 2016.

[16] B. Xia, X. Zhao, R. De Callafon, H. Garnier, T. Nguyen, and C. Mi, “Accurate lithium-ion battery parameter estimation with continuous-time system identification methods,” *Applied energy*, vol. 179, pp. 426–436, 2016.

[17] C. Zou, X. Hu, S. Dey, L. Zhang, and X. Tang, “Nonlinear fractional-order estimator with guaranteed robustness and stability for lithium-ion batteries,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5951–5961, 2017.

[18] N. Tian, H. Fang, J. Chen, and Y. Wang, “Nonlinear double-capacitor model for rechargeable batteries: Modeling, identification, and validation,” *IEEE Transactions on Control Systems Technology*, vol. 29, no. 1, pp. 370–384, 2020.

[19] V. H. Johnson, A. A. Pesaran, and T. Sack, “Temperature-dependent battery models for high-power lithium-ion batteries,” National Renewable Energy Lab., Golden, CO (US), Tech. Rep., 2001.

[20] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[21] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*. Cambridge university press, 2007.

[22] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[23] Y. Hu and R. A. de Callafon, “Optimal weighting for covariance based realization algorithm,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 5274–5279.

[24] Y. Hu, Y. Jiang, and R. A. de Callafon, “Variance reduction in covariance based realization algorithm with application to closed-loop data,” *Automatica*, vol. 113, p. 108683, 2020.

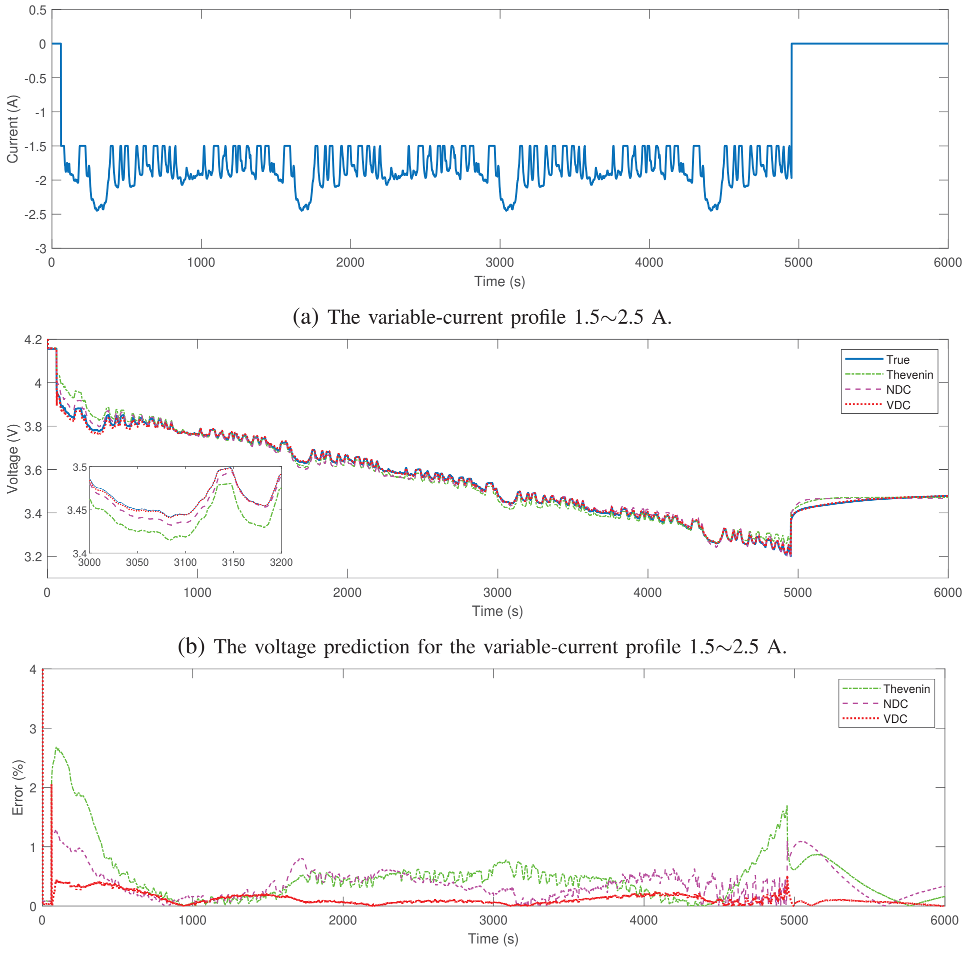
[25] D. Yang, X. Zhang, R. Pan, Y. Wang, and Z. Chen, “A novel gaussian process regression model for state-of-health estimation of lithium-ion battery using charging curve,” *Journal of Power Sources*, vol. 384, pp. 387–395, 2018.

[26] Y. Hu, S. A. R. Konakalla, and R. A. de Callafon, “Covariance based estimation for reduced order models of microgrid power flow dynamics,” *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 903–908, 2018.

[27] R. R. Ardeshiri, B. Balagopal, A. Alsabbagh, C. Ma, and M.-Y. Chow, “Machine learning approaches in battery management systems: State of the art: Remaining useful life and fault detection,” in *2020 2nd IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*, vol. 1. IEEE, 2020, pp. 61–66.

[28] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, “State-of-charge estimation of li-ion batteries using deep neural networks: A machine learning approach,” *Journal of Power Sources*, vol. 400, pp. 242–255, 2018.

[29] Y. L. Murphey, J. Park, Z. Chen, M. L. Kuang, M. A. Masrur, and A. M. Phillips, “Intelligent hybrid vehicle power control—part i: Machine learning of optimal vehicle power,” *IEEE Transactions on Vehicular Technology*, vol. 61, no. 8, pp. 3519–3530, 2012.



(a) The variable-current profile 1.5∼2.5 A.

(b) The voltage prediction for the variable-current profile 1.5∼2.5 A.

(c) Fitting error in percentage.

Fig. 11: Illustration of the voltage prediction of the proposed model for the variable-current profile 1.5∼2.5 A in the validation data set. The RMSEs for different models are: 25.9mV (Thevenin); 16.2mV (NDC); 6.15mV (VDC).

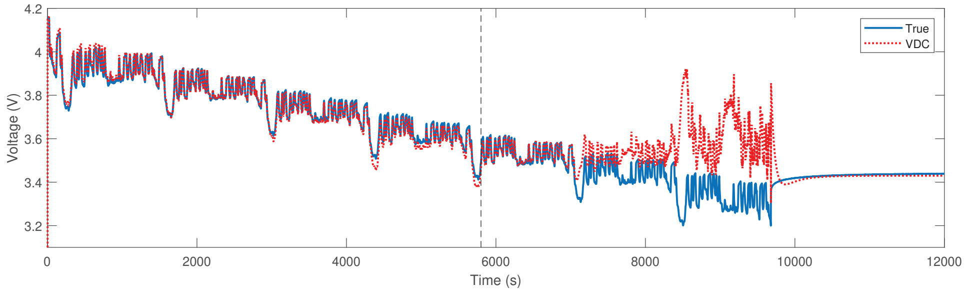


Fig. 12: Illustration of the effect of training data range on the model fitting. The training data for the VDC model includes STD and a variable-current profile 0∼6 A with the SoC spreading from 100% to 47%. The validation data is a variable-current profile 0∼3 A with the SoC spreading from 100% to 14%. The figure shows the voltage prediction on the validation data and the black vertical dashed line indicates the time instant where $\mathrm{SoC} = 47\%$.

[30] X. Hu, S. E. Li, and Y. Yang, "Advanced machine learning approach for lithium-ion battery state estimation in electric vehicles," *IEEE Transactions on Transportation electrification*, vol. 2, no. 2, pp. 140–149, 2015.

[31] M. A. Hannan, M. H. Lipu, A. Hussain, P. J. Ker, T. Mahlia, M. Mansor, A. Ayob, M. H. Saad, and Z. Dong, "toward enhanced state of charge estimation of lithium-ion batteries using optimized machine learning techniques," *Scientific reports*, vol. 10, no. 1, pp. 1–15, 2020.

[32] E. Stoudenmire and D. J. Schwab, "Supervised learning with tensor networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4799–4807.

[33] K. Batselier, Z. Chen, and N. Wong, "Tensor network alternating linear scheme for mimo volterra system identification," *Automatica*, vol. 84, pp. 26–35, 2017.

[34] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[35] Y. Hu, R. A. de Callafon, N. Tian, and H. Fang, "Modeling of lithium-ion batteries via tensor-network-based volterra model," in *2021 Modeling, Estimation and Control Conference (MECC)*, vol. 54, no. 20. Elsevier, 2021, pp. 509–515.

[36] M. Guo, G. Sikha, and R. E. White, “Single-particle model for a lithium-ion cell: Thermal behavior,” *Journal of The Electrochemical Society*, vol. 158, no. 2, p. A122, 2010.

[37] H. Fang, Y. Wang, and J. Chen, “Health-aware and user-involved battery charging management for electric vehicles: Linear quadratic strategies,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 911–923, 2016.

[38] M. B. Pinson and M. Z. Bazant, “Theory of sei formation in rechargeable batteries: capacity fade, accelerated aging and lifetime prediction,” *Journal of the Electrochemical Society*, vol. 160, no. 2, p. A243, 2012.

[39] D. Andre, M. Meiler, K. Steiner, C. Wimmer, T. Soczka-Guth, and D. Sauer, “Characterization of high-power lithium-ion batteries by electrochemical impedance spectroscopy. i. experimental investigation,” *Journal of Power Sources*, vol. 196, no. 12, pp. 5334–5341, 2011.

[40] M. Dubarry, N. Vuillaume, and B. Y. Liaw, “From single cell model to battery pack simulation for li-ion batteries,” *Journal of Power Sources*, vol. 186, no. 2, pp. 500–507, 2009.

[41] S. Holtz, T. Rohwedder, and R. Schneider, “The alternating linear scheme for tensor optimization in the tensor train format,” *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A683–A713, 2012.

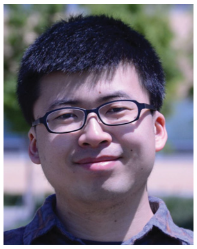
[42] Y. Hu, L. Tan, and R. A. de Callafon, “Persistent excitation condition for mimo volterra system identification with gaussian distributed input signals,” in *2019 IEEE 58th Annual Conference on Decision and Control (CDC)*. IEEE, 2019.

[43] N. Tian, Y. Wang, J. Chen, and H. Fang, “One-shot parameter identification of the thevenin’s model for batteries: Methods and validation,” *Journal of Energy Storage*, vol. 29, p. 101282, 2020.

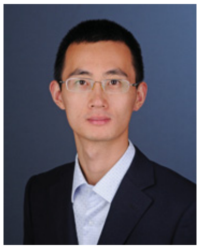
Yangsheng Hu received his B.Eng. degree in Automation from the Xidian University, Xi’an, China, in 2013, the M.Sc. degree in Control Science and Engineering from the University of Science and Technology of China, Hefei, China, in 2016, and the Ph.D. degree in Mechanical Engineering from the University of California, San Diego, CA, USA, in 2021. His research interests include control theory, system identification, subspace methods, and practical controller design for electromechanical and power systems.

Raymond A. de Callafon received his M.Sc. (1992) and his Ph.D. (1998) in Mechanical Engineering from the Delft University of Technology and is currently a Professor with the Dynamic Systems and Control group in the Department of Mechanical and Aerospace Engineering at the University of California, San Diego. He is involved in teaching, research, consulting and judicial expertise in the area of signal processing, parameter/state estimation, servo/adaptive control and embedded software development.

The work of Prof. de Callafon has been applied to structural damage detection problems, adaptive feedback tuning in active noise/vibration control, wildfire data assimilation and dynamic modeling and control of electric power systems. Prof. de Callafon has received the 2010 Technical Achievement Award from the Information Storage Industry Consortium (INSIC), a 2015 High Performance Computing (HPC) Editors Choice Award for building a cyberinfrastructure to better monitor, predict, and mitigate future wildfires and a 2016 North America Synchrophasor Initiative (NASPI) MVP award.



Ning Tian received the B.Eng. and M.Sc. degrees in thermal engineering from Northwestern Polytechnic University, Xi’an, China, in 2012 and 2015, and the Ph.D. degree in mechanical engineering from the University of Kansas, Lawrence, KS, USA, in 2020. His research interests include control theory and its application to advanced battery management.



Huazhen Fang (Member, IEEE) received the B.Eng. degree in computer science and technology from Northwestern Polytechnic University, Xi’an, China, in 2006, the M.Sc. degree from the University of Saskatchewan, Saskatoon, SK, Canada, in 2009, and the Ph.D. degree from the University of California, San Diego, CA, USA, in 2014, both in mechanical engineering. He is currently an Associate Professor of mechanical engineering with the University of Kansas, Lawrence, KS, USA. His research interests include control and estimation theory with application to energy management and cooperative robotics. He was a recipient of the 2019 National Science Foundation CAREER Award. He currently serves as an Associate Editor for IEEE Transactions on Industrial Electronics, IEEE Control Systems Letters, and Information Sciences, among others.