IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 5, 2020, accepted July 29, 2020, date of publication August 24, 2020, date of current version September 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017774

A Sparse Learning Machine for Real-Time SOC Estimation of Li-ion Batteries

LI ZHANG[1], KANG LI[2], (Senior Member, IEEE), DAJUN DU[1], YUANJUN GUO[3], MINRUI FEI[1], AND ZHILE YANG[3,4], (Member, IEEE)

[1]Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China
[2]School of Electronic and Electrical Engineering, University of Leeds, Leeds LS2 9JT, U.K.
[3]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[4]CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen 518055, China

Corresponding author: Zhile Yang (zl.yang@siat.ac.cn)

This work was supported in part by NSFC under Grant 61673256, in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R030243/1, in part by the Key Project of Science and Technology Commission of Shanghai Municipality under Grant 19500712300, in part by the 111 Project under Grant D18003, in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030310671, and in part by the Outstanding Young Researcher Innovation Fund of Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences under Grant 201822. The work of Li Zhang was supported in part by the financial support of Chinese Scholarship Council to take a one-year visiting research at the University of Leeds, from 2018 to 2019, and in part by the University of Leeds during her visiting period.

ABSTRACT The state of charge (SOC) estimation of Li-ion batteries has attracted substantial interests in recent years. Kalman Filter has been widely used in real-time battery SOC estimation, however, to build a suitable dynamic battery state-space model is a key challenge, and most existing methods still use the off-line modelling approach. This paper tackles the challenge by proposing a novel sparse learning machine for real-time SOC estimation. This is achieved first by developing a new learning machine based on the traditional least squares support vector machine (LS-SVM) to capture the process dynamics of Li-ion batteries in real-time. The least squares support vector machine is the least squares version of the conventional support vector machines (SVMs) which suffers from low model sparseness. The proposed learning machine reduces the dimension of the projected high dimensional feature space with no loss of input information, leading to improved model sparsity and accuracy. To accelerate computation, mapping functions in the high feature space are selected using a fast recursive method. To further improve the model accuracy, a weighted regularization scheme and the differential evolution (DE) method are used to optimize the parameters. Then, an unscented Kalman filter (UKF) is used for real-time SOC estimation based on the proposed sparse learning machine model. Experimental results on the Federal Urban Drive Schedule (FUDS) test data reveal that the performance of the proposed algorithm is significantly enhanced, where the maximum absolute error is only one sixth of that obtained by the conventional LS-SVMs and the mean square error of the SOC estimations reaches to 10^{-7}, while the proposed method is executed nearly 10 times faster than the conventional LS-SVMs.

INDEX TERMS Sparse learning machine, state-of-charge (SOC), least squares support vector machine (LS-SVM), differential evolution (DE), unscented Kalman filter (UKF).

I. INTRODUCTION

In recent years, the transportation electrification through mass roll-out of electric vehicles has been considered as an important measure to tackle the global challenges of climate change and environmental pollutions due to substantive consumption of fossil fuels in the transportation sector [1]. Li-ion batteries are widely used for energy storage because of their favourable advantages of high energy density, high power performance and long cycle life [2], [3]. However, the battery pack is often composed of hundreds or thousands battery cells. The battery management system (BMS) is essential to achieve a high operation safety level and high efficiency in battery charging and discharging [4]. The state

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Zhou.

of charge (SOC), indicating the remaining capacity, is an important battery state to estimate in BMS in order to maintain the battery operation safety and efficiency [5].

The battery SOC estimation has been intensively researched in the literature. The conventional direct measurement methods [1], including the ampere-hour method [6] and the open circuit voltage (OCV) based method [7], may easily lead to accumulated deviations due to the measurement errors. Besides, the OCV-based method takes long time (more than 1 hour) to measure the OCV which makes this method hardly applicable on-line [5]. To overcome this drawback, different model-based methods have been developed, such as the equivalent electric circuit models (EECMs) [1], [8], artificial neural networks [9] and least square support vector machine (LS-SVM) [2], etc. However, the majority of the existing model based approaches involve a complex offline model identification phase. When they are applied online, the resultant models are sensitive to measurement errors.

To reduce the impact of measurement errors on the estimation accuracy, the Kalman filter is widely adopted. The Kalman filter uses both the model outputs and the measurements to achieve optimal estimation of state variables by minimizing the mean square errors of the outputs. However, the classical Kalman filter is only suitable for linear systems. For the complex nonlinear processes of battery charging and discharging, the extended Kalman filter (EKF) [10]–[12], the unscented Kalman filter (UKF) [13], [14] and other Kalman filter variants [2], [15] have been successfully applied to the battery SOC estimation. The EKF method is capable of producing more accurate SOC estimation through linearizing the battery state-space equation using Taylor expansion series. The iterative process of the EKF heavily depends on the initial model and the initial state estimation. Furthermore, the Jacobian matrix may not be obtainable in some cases [15]. Unlike the EKF method, the UKF method based on the unscented transform is more robust, accurate and easier to implement. Similarly to the EKF method, the performance of the UKF method is still highly dependent on the initial model. In order to overcome the shortcomings of UKF, the adaptive unscented filter (AUKF) method [16] which adaptively adjusts the process noise covariance and the measurement noise covariance in the estimation process, has been successfully applied to the SOC estimation.

In summary, the existing model based methods for the battery SOC estimation use the offline models. Their accuracy will be compromised due to real-time noise and the initial state-space model, though the AUKF can reduce the impacts of measurement and process noises at the cost of increased computation cost. To overcome the drawback, online battery model identification has been researched in the last few years, such as the RTLS-based observer method [3] and the Gaussian process regression method [17]. This paper follows the similar technical route to build an online battery state-space model. In order to reduce the computation time incurred by existing approaches, a sparse learning machine (LM) trained only using a very small number of samples is developed for SOC estimation, in replacement of the conventional LS-SVM model. Although the traditional LS-SVM method can find the solutions quickly by solving a set of linear equations, yet poor sparseness and model accuracy are the shortcomings [18]–[20].

To enhance the sparsity of LS-SVM models, various methods have been proposed in the literature. They can be generally categorized into two groups: sparse samples methods and features selection methods [18], [20]. The sparse samples methods often involve reduction or selection of support vectors. The reduction methods prune the support vectors to approximate the original SVM model. For example, Suykens *et al.* proposed a class of pruning methods based on the spectrum of the support values [21], [22]. Other works [23]–[25] proposed to improve these pruning methods by iteratively solving a smaller set of linear equations. For the selection methods, the model sparsity is considered in advance by fixing the number of the prototype vectors which are iteratively selected [26]–[29]. These methods induce a low rank kernel matrix to approximate the primal space. For example, in [18], [28] the model sparsity is achieved by directly selecting support vectors (SVs) from the training data set based on a certain criterion. The main issue with these methods lies in the fact that a small improvement in the sparsity is often achieved at the cost of significantly increased computational effort, while information embedded into the removed data is lost. For the features selection method, the selection of kernel functions is a key stage. Jiao *et al.* [30] proposed a fast sparse approximation method LS-SVM (FSA-LSSVM) which iteratively build an approximated model by selecting the basic kernel function one by one. Zhou [19], [29] given an equivalent form of the cost function based on the kernel function to look for a low-rank approximation. However, Li and Zhang *et al.* [31] presented a sparse learning machine based on the LS-SVMs cost function, and the fast recursive algorithm (FRA) [32] is used to select the mapping function.

In this paper, a sparse learning machine namely S-DE-WLM is used to build the battery state-space model based our early work in [31], and the main technical contributions are summarized as follows:

- The UKF method which has proved to be more effective than a few other KF variants is applied to the SOC estimation of Li-ion batteries. Compared to the EKF method, the UKF method is capable of producing more accurate estimations for strongly nonlinear system with a large initial deviation. Besides, the UKF method without measurement noise adjustment reduce the overall computation time compared to the AUKF method. In order to increase the online estimation accuracy, a battery state-space model is built first using the proposed sparse learning machine. The new model for online SOC estimation is not reliant on the first principle laws governing the battery's electrochemical processes and can be adjusted for real-time battery operation.

- To strike the balance between the model accuracy and model complexity, a regularization term is added to the traditional sum of squared error cost function. Further, a weighting scheme is applied to adjust the contribution of each training sample based on its proximity to the next forecasting time interval. The closer to the next forecasting interval, the sample will be given a larger weight. Using the weighted regularization scheme, the accumulated measurement error can be reduced and the model accuracy for short-term prediction can therefore be enhanced in real-time applications. In detail, the online model is built using a narrow sliding window of data samples at each time interval, e.g. 10 samples at each time interval. The number of selected samples, i.g. 10 samples, is determined by trial-and-error through extensive experiments. It is a trade-off between the computational complexity of model training and resultant model accuracy. Among these data samples, not all have the same level of contributions in predicting the SOC at the next time interval. Intuitively, the closer the stronger the influence becomes.
- To improve the sparsity of the proposed model, a fast recursive method is applied to select the nonlinear feature based on the weighted LS-SVM formulation. Instead of using the kernel trick, the feature mapping functions is determined first. According to the principle of parsimony, the minimum feature space is then determined which uses the least square method to calculate the net contribution of a feature to the cost function. Based on this contribution, a projected feature is selected or removed by the learning machine.
- Finally, the differential evolution (DE) algorithm [33]–[35] is adopted to optimize the centers and widths of the RBF mapping function to improve the model accuracy.

The remainder of this paper is organized as follows. Section II introduces the battery state-space model for SOC estimation. In Section III, a sparse learning machine based on the LS-SVM formulation is presented for online modelling. The SOC estimation approach is detailed in Section IV using UKF assisted with the proposed sparse learning machine. The experimental and results are presented in Section V. Finally, Section VI concludes this paper.

II. THE BATTERY MODEL

In order to estimate the SOC of Li-ion batteries in real-time using the UKF method, an adaptive online model in the state-space expression is built first. The flowchart of the battery SOC estimation is illustrated in Fig.1.

In Fig.1, a small set of data samples $\{[\boldsymbol{soc\_m}, \boldsymbol{i}]; \boldsymbol{v}\}$ at the time instant k are used to build the sparse learning machine model, where $\boldsymbol{soc\_m} = [soc\_m(k), \cdots, soc\_m(k-m)]$ is the past $m+1$ SOC values, $\boldsymbol{i} = [i(k), \cdots, i(k-m)]$ and $\boldsymbol{v} = [v(k), \cdots, v(k-m)]$ are the past m+1 terminal current and voltage measurements. The model is adaptively adjusted using the DE algorithm based on the estimation error

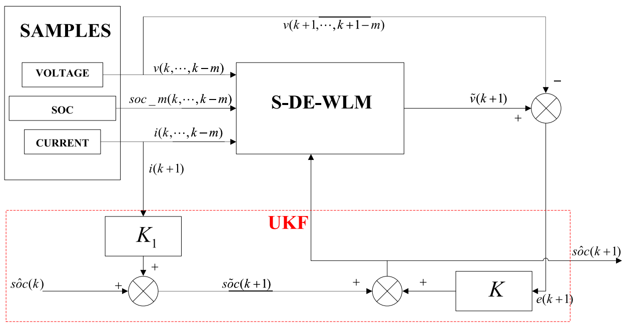

FIGURE 1. The structure for battery SOC estimation.

$\boldsymbol{e} = [e(k), \cdots, e(k-m)]$ in order to produce a more accurate prediction output $v(k+1)$ at the time instant $k+1$.

In addition, a state equation which describes the estimated states is dynamically updated over time. In this paper, the state equation is built based on the battery SOC dynamics. The complete model including a state equation for dynamic SOC forecasting and a measurement equation is given below.

A. STATE EQUATION

Battery SOC is defined as the ratio of the remaining charge respect to the fully capacity. Suppose the nominal capacity and the remaining capacity are denoted as Q_n and Q_t respectively. Then SOC_t is given by [2]

$$\begin{aligned} SOC_t &= Q_t/Q_n \\ &= (Q_0 - \int_0^t \eta \cdot i(\tau)\,d\tau)/Q_n \\ &= SOC_0 - \int_0^t \eta \cdot i(\tau)\,d\tau/Q_n \end{aligned} \tag{1}$$

where $i(\tau)$ is the charging/discharging current at time instant τ, and $\eta = 1/3600$.

Eqn.(1) can be expressed in the following recursive form

$$SOC_k = SOC_{k-1} - \eta \cdot \Delta t \cdot i_k/Q_n \tag{2}$$

where SOC_k and SOC_{k-1} denote the SOC at the time instant k and $k-1$ respectively. Δt denotes the time interval of the sampling period, and i_k represents the terminal current at time instant k.

B. MEASUREMENT EQUATION

In general, the battery terminal behaviour which is described as a highly nonlinear model can correlate the SOC and the terminal current to the terminal voltage. The sparse learning machine which is capable of capturing the nonlinear characteristics of a process is used to build the measurement equation that exhibits a nonlinear behaviour. Assuming the discharging current, the SOC and the terminal voltage at the time instant k are i_k, SOC_k, and V_k respectively, the measurement equation [31] can be expressed as follows

$$\begin{aligned} V_k &= h(SOC_k, i_k) + b \\ &= \sum_{i=1}^{m} w_{m,j} \exp\{-\frac{(u_k - uc_j)^T \Gamma_j^{-1} (u_k - uc_j)}{2}\} + b \end{aligned} \tag{3}$$

where $u_k = [i_k, SOC_k]$ consists of the current i_k, the state of charge SOC_k at the time instant k respectively. The Gaussian function $\exp\{-\frac{1}{2}(u_k - uc_j)^T \Gamma_j^{-1}(u_k - uc_j)\}$ is a feature map to produce the j^{th} feature, $w_{m,j}$ are the linear coefficients, and m is the final dimension of the feature space. The hyparameters uc_j and Γ_j are selected training samples center and the variance, and b is the constant term in the model.

III. A SPARSE LEARNING MACHINE

Evidently, an accurate model needs to be built for the measurement equation as shown in section II. In order to enhance the model accuracy and to reduce the computation complexity, a sparse learning machine is presented in the following section. To achieve the sparse solution while considering the unbalanced data samples, a weighting scheme is adopted to adjust the contributions of samples to the sum of the squared errors (SSE) cost function [36]. In the following, a weighted LS-SVM model is presented first in part A, then a fast algorithm to build a sparse learning machine is given in part B. Then part C provides the details of parameter optimization in the sparse learning machine. Finally, the whole sparse learning machine construction procedure is summarized in part D.

A. THE WEIGHTED LS-SVM MODEL

Consider a nonlinear multiple-input single-output system

$$y = f(\boldsymbol{x}) \tag{4}$$

where $\boldsymbol{x} \in \Re^m$ and $y \in \Re$.

Now suppose the above nonlinear system is approximated using a SVM model

$$\boldsymbol{y} = \boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{w} + \boldsymbol{e} \tag{5}$$

where $\mathbf{x} \in \Re^{N\times k}$ and $\boldsymbol{y} \in \Re^{N\times 1}$ denotes N pairs of samples $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$. $\boldsymbol{w} \in \Re^{(N+1)\times 1}$ presents the linear parameter vector. $\boldsymbol{e} \in \Re^{N\times 1}$ is the model residual vector. $\boldsymbol{\Phi} = [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \cdots, \Phi_N(\mathbf{x}), \boldsymbol{E}] \in \Re^{N\times(N+1)}$ including the constant terms ($\boldsymbol{E} = [1, \cdots, 1]^T$) are series of mapping functions projecting the input space to a high-dimensional feature space.

The weighted least squares algorithm [37] used in the model identification can be formulated as follows

$$\begin{aligned} &\min_{\boldsymbol{w}, e_i} \boldsymbol{J} = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{c} \cdot \sum_{i=1}^{N} \frac{1}{2\lambda_i} \cdot e_i^2 \\ &st\ e_i = y_i - \boldsymbol{\Phi}(\boldsymbol{x}_i) \cdot \boldsymbol{w} \end{aligned} \tag{6}$$

where $\|\boldsymbol{w}\|^2$ represents the regularization term for a trade-off between the magnitude of the coefficients and the estimation error, c is the regularization parameter. $\boldsymbol{\Phi}(\boldsymbol{x}_i) = [\Phi_1(\boldsymbol{x}_i), \Phi_2(\boldsymbol{x}_i), \cdots, \Phi_N(\boldsymbol{x}_i), 1]$ represents $N + 1$-dimensional features induced by a sample. $[\lambda_i],\ i = 1, \cdots, N$ is defined as a sequence of random numbers to weigh the contribution of each sample.

The following cost function can be obtained using the Lagrangian method

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{c} \cdot \sum_{i=1}^{N} \frac{1}{2\lambda_i} \cdot e_i^2 - \sum_{i=1}^{N} \alpha_i \{\boldsymbol{\Phi}(\mathbf{x_i}) \cdot \boldsymbol{w} + e_i - y_i\} \tag{7}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]^T$ is the Lagrange multiplier vector.

The Karush-Kuhn-Tucker (KKT) conditions for optimizing the above cost function are given below

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \boldsymbol{\Phi}(\boldsymbol{x}_i), \\ &\frac{\partial \mathcal{L}}{\partial e_i} = 0 \Rightarrow \alpha_i = \frac{e_i}{c \cdot \lambda_i}, \quad \forall i \in \{1, \cdots, N\} \\ &\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \Rightarrow \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_i) + e_i - y_i = 0, \quad \forall i \in \{1, \cdots, N\} \end{aligned} \tag{8}$$

Thus the optimality problem can be formulated as

$$\boldsymbol{y} = \mathbf{M}\boldsymbol{\alpha} \tag{9}$$

where $\mathbf{M} = \mathbf{K} + c \cdot \Lambda$ is a positive definite symmetric matrix and $\mathbf{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is the matrix of the kernel function. A typical kernel function is radial basis function (RBF) given by $\mathbf{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|/(2\sigma^2))$. $\Lambda = diag(\lambda_1, \cdots, \lambda_N)$ is the weight vector related to the error spectrum.

Substituting the solution $\hat{\boldsymbol{\alpha}}$ of linear equations (9) into (8), the resultant LS-SVM model for (5) is given as

$$\hat{\boldsymbol{y}} = \mathbf{K}\hat{\boldsymbol{\alpha}} \tag{10}$$

where $\hat{\boldsymbol{y}}$ is the estimation outputs.

B. A FAST ALGORITHM TO BUILD A SPARSE LEARNING MACHINE

Though the computation load in the training process is partly reduced by introducing the kernel function, to solve the KKT equations (9) is still computationally expensive. Thus a fast recursive method is presented to build a sparse learning machine applied to the LS-SVM formulation. Unlike the kernel method, the nonlinear features are constituted by the mapping functions [18], [31]. The basis mapping functions are first selected using the entire training samples for a high-dimensional feature space. Gaussian Basis function has a good approximation performance for unknown nonlinear systems, so here the feature space with m dimension is written as

$$\begin{aligned} \boldsymbol{\Phi}_m(\boldsymbol{x}_i) &= [\Phi_1(\boldsymbol{x}_i), \cdots, \Phi_m(\boldsymbol{x}_i)] \\ \Phi_j(\boldsymbol{x}_i) &= \exp\{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{s}_j)^T \Gamma_j^{-1}(\boldsymbol{x}_i - \boldsymbol{s}_j)\} \end{aligned} \tag{11}$$

where $j = 1, \cdots, m$ and m is the maximum dimension of the mapped high dimensional space $\mathcal{F}^m$. Γ_j^{-1} is the width (variance) matrix and $\boldsymbol{s}_j$ is selected training samples center matrix.

Including the constant terms, the $N+1$ candidate features in matrix notation ($\mathbf{\Phi}$ in Eqn. (5)) are given as

$$\mathbf{\Phi}_N(\mathbf{x}) = \begin{bmatrix} \Phi_1(\mathbf{x}_1) & \cdots & \Phi_N(\mathbf{x}_1) & 1 \\ \Phi_1(\mathbf{x}_2) & \cdots & \Phi_N(\mathbf{x}_2) & 1 \\ \cdots & \cdots & \cdots & \\ \Phi_1(\mathbf{x}_N) & \cdots & \Phi_N(\mathbf{x}_N) & 1 \end{bmatrix} \tag{12}$$

Noting that (6) is a constrained optimization problem with one equality which is solved by the direct substitution method, the cost function can thus be redefined as

$$\min_{\mathbf{w},\mu_i} \mathbf{J} = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{c}\cdot\sum_{i=1}^{N}\frac{1}{2\lambda_i}\cdot(y_i - \mathbf{\Phi}(\mathbf{x_i})\cdot\mathbf{w})^2 \tag{13}$$

and

$$\mathbf{J} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2c}\cdot(\mathbf{y}-\mathbf{\Phi}\cdot\mathbf{w})^T\Lambda^{-1}(\mathbf{y}-\mathbf{\Phi}\cdot\mathbf{w}) \tag{14}$$

where $\mathbf{\Phi} = [\Phi_1,\cdots,\Phi_N,1_{Nx1}]^T$, $\Phi_j = [\Phi_j(\mathbf{x}_1),\cdots,\Phi_j(\mathbf{x}_N)]$ and $\Lambda = diag(\lambda_1,\cdots,\lambda_N)$.

Then the partial derivative of the cost function with respect to the variable $\mathbf{w}$ is given by

$$\frac{\partial \mathbf{J}}{\partial \mathbf{w}} = \mathbf{w} + \frac{1}{c}\cdot\mathbf{\Phi}^T\mathbf{\Phi}\cdot\mathbf{w} - \frac{1}{c}\cdot\mathbf{\Phi}^T\Lambda^{-1}\mathbf{y} \tag{15}$$

Setting the gradient (15) to zero, the minimal $\mathbf{w}$ can be computed as

$$\hat{\mathbf{w}} = (c\cdot\mathbf{I} + \mathbf{\Phi}^T\Lambda^{-1}\mathbf{\Phi})^{-1}\Phi^T\Lambda^{-1}\mathbf{y} \tag{16}$$

According to the well-known matrix inversion lemma $[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{D}\mathbf{A}^{-1}\mathbf{B}+\mathbf{C}^{-1}]^{-1}\mathbf{D}\mathbf{A}^{-1}$, $\mathbf{w}$ can be reformulated as

$$\begin{aligned}\hat{\mathbf{w}} &= \frac{1}{c}\cdot(\mathbf{I} - \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda)^{-1}\Phi)\mathbf{\Phi}^T\Lambda^{-1}\mathbf{y} \\ &= \frac{1}{c}\cdot\mathbf{\Phi}^T(\mathbf{I} - (\mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda)^{-1}\mathbf{\Phi}\mathbf{\Phi}^T)\Lambda^{-1}\mathbf{y} \\ &= \frac{1}{c}\cdot\mathbf{\Phi}^T(\mathbf{I} + \frac{\Lambda^{-1}}{c}\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\Lambda^{-1}\mathbf{y} \\ &= \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda)^{-1}\mathbf{y}\end{aligned} \tag{17}$$

Therefore the linear model in the feature space is built by

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{x})\cdot\hat{\mathbf{w}} \tag{18}$$

Then the estimation error is given by

$$\begin{aligned}\mathbf{e} &= \mathbf{y} - \mathbf{\Phi}(\mathbf{x})\cdot\hat{\mathbf{w}} \\ &= c\cdot\Lambda(\mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda)^{-1}\mathbf{y}\end{aligned} \tag{19}$$

Substituting (17) and (19) into (14), the minimal cost function can be expressed as

$$\begin{aligned}\mathbf{J} &= (\mathbf{\Phi}^T(c\cdot\Lambda + \mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{y})^T(\mathbf{\Phi}^T(c\cdot\Lambda + \mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{y}) \\ &\quad + \frac{1}{c}\cdot(c\cdot\Lambda(c\cdot\Lambda + \mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{y})^T\Lambda^{-1}(c\cdot\Lambda(c\cdot\Lambda + \mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{y}) \\ &= \mathbf{y}^T(\mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda)^{-1}\mathbf{y}\end{aligned} \tag{20}$$

Referring to (17) and (20), define a recursive matrix $\mathbf{M}$

$$\mathbf{M} = \mathbf{\Phi}\mathbf{\Phi}^T + c\cdot\Lambda \tag{21}$$

According to the FRA method [32], the features are selected from the candidate feature matrix one by one based on their contributions to the cost function. Suppose k features were selected, the recursive matrix $\mathbf{M}$ is calculated by

$$\mathbf{M}_k = \mathbf{\Phi}_k\mathbf{\Phi}_k^T + c\cdot\Lambda \tag{22}$$

where $k = 1,\cdots,m$ and $\mathbf{M}_0 = c\cdot\Lambda$.

According to the definition in (22), the matrix $\mathbf{M}_k$ has the following properties:

$$\mathbf{M}_k^T = (\mathbf{\Phi}_k\mathbf{\Phi}_k^T + c\cdot\Lambda)^T = \mathbf{M}_k \tag{23}$$

$$\begin{aligned}\mathbf{M}_{k+1}^{-1} &= (\mathbf{\Phi}_{k+1}\mathbf{\Phi}_{k+1}^T + c\cdot\Lambda)^{-1} \\ &= (\mathbf{\Phi}_k\mathbf{\Phi}_k^T + \Phi_{k+1}\Phi_{k+1}^T + c\cdot\Lambda)^{-1} \\ &= (\mathbf{M}_k + \Phi_{k+1}\Phi_{k+1}^T)^{-1} \\ &= \mathbf{M}_k^{-1} - \frac{\mathbf{M}_k^{-1}\Phi_{k+1}\Phi_{k+1}^T\mathbf{M}_k^{-1}}{1+\Phi_{k+1}^T\mathbf{M}_k^{-1}\Phi_{k+1}} \\ &= \mathbf{M}_0^{-1} - \sum_{i=1}^{k}\frac{\mathbf{M}_{i-1}^{-1}\Phi_i\Phi_i^T\mathbf{M}_{i-1}^{-1}}{1+\Phi_i^T\mathbf{M}_{i-1}^{-1}\Phi_i}\end{aligned} \tag{24}$$

where $k = 0,\cdots,m-1$ and $\mathbf{M}_0^{-1} = \frac{1}{c}\cdot\Lambda^{-1} = \frac{1}{c}\cdot diag(\lambda_1^{-1},\cdots,\lambda_N^{-1})$ is a diagonal matrix.

The cost function and the $\mathbf{w}$ estimation with k features become

$$\hat{\mathbf{w}}_k = \Phi_k^T\mathbf{M}_k^{-1}\mathbf{y} \tag{25}$$

$$\mathbf{J}_k = \mathbf{y}^T\mathbf{M}_k^{-1}\mathbf{y} \tag{26}$$

Now, the net contribution of the $(k+1)^{th}$ feature to the cost function can be given

$$\begin{aligned}\Delta\mathbf{J}_{k+1} &= \mathbf{y}^T(\mathbf{M}_k^{-1} - \mathbf{M}_{k+1}^{-1})\mathbf{y} \\ &= \mathbf{y}^T\frac{\mathbf{M}_k^{-1}\Phi_{k+1}\Phi_{k+1}^T\mathbf{M}_k^{-1}}{1+\Phi_{k+1}^T\mathbf{M}_k^{-1}\Phi_{k+1}}\mathbf{y}\end{aligned} \tag{27}$$

where $k = 0,\cdots,m-1$.

In order to simplify the computation of the net contribution and the linear parameter, two new intermediate matrices $\mathbf{A}\in\Re^{m\times N} = \{a_{k+1,i}\}$ and $\mathbf{B}\in\Re^m = \{b_{k+1}\}$ are defined

$$\begin{aligned}a_{k+1,i} &= \Phi_{k+1}^T\mathbf{M}_k^{-1}\Phi_i \\ b_{k+1} &= \mathbf{y}^T\mathbf{M}_k^{-1}\Phi_{k+1}\end{aligned} \tag{28}$$

where $k = 0,\cdots,m-1$ and $i = 1,\cdots,N$. Note that $(a_{k+1,i} = \Phi_{k+1}^T\mathbf{M}_k^{-1}\Phi_i) \neq (a_{i,k+1} = \Phi_i^T\mathbf{M}_{i-1}^{-1}\Phi_{k+1})$ because of $\mathbf{M}_k \neq \mathbf{M}_{i-1}$.

$a_{k+1,i}$ can be iteratively calculated as follows:

$$\begin{aligned}a_{k+1,i} &= \Phi_{k+1}^T\mathbf{M}_k^{-1}\Phi_i \\ &= \Phi_{k+1}^T(\mathbf{M}_0^{-1} - \sum_{j=1}^{k}\frac{\mathbf{M}_{j-1}^{-1}\Phi_j\Phi_j^T\mathbf{M}_{j-1}^{-1}}{1+\Phi_j^T\mathbf{M}_{j-1}^{-1}\Phi_j})\Phi_i \\ &= \Phi_{k+1}^T\mathbf{M}_0^{-1}\Phi_i - \sum_{j=1}^{k}\frac{\Phi_{k+1}^T\mathbf{M}_{j-1}^{-1}\Phi_j\Phi_j^T\mathbf{M}_{j-1}^{-1}\Phi_i}{1+\Phi_j^T\mathbf{M}_{j-1}^{-1}\Phi_j} \\ &= \Phi_{k+1}^T\mathbf{M}_0^{-1}\Phi_i - \sum_{j=1}^{k}\frac{a_{j,k+1}a_{j,i}}{1+a_{j,j}}\end{aligned} \tag{29}$$

Similarly,

$$\begin{aligned}
a_{i,k+1} &= \Phi_i^T \boldsymbol{M}_{i-1}^{-1} \Phi_{k+1} \\
&= \Phi_i^T (\boldsymbol{M}_0^{-1} - \sum_{j=1}^{i-1} \frac{\boldsymbol{M}_{j-1}^{-1} \Phi_j \Phi_j^T \boldsymbol{M}_{j-1}^{-1}}{1 + \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_j}) \Phi_{k+1} \\
&= \Phi_i^T \boldsymbol{M}_0^{-1} \Phi_{k+1} - \sum_{j=1}^{i-1} \frac{\Phi_i^T \boldsymbol{M}_{j-1}^{-1} \Phi_j \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_{k+1}}{1 + \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_j} \\
&= \Phi_i^T \boldsymbol{M}_0^{-1} \Phi_{k+1} - \sum_{j=1}^{i-1} \frac{a_{j,k+1} a_{j,i}}{1 + a_{j,j}} \qquad (30)
\end{aligned}$$

where i and k reflect the position of the features in the matrix $\boldsymbol{\Phi}$ described by (12).

Particularly, the bottom triangular elements in $\boldsymbol{A}$ with $k > i$ can be computed as

$$\begin{aligned}
a_{k+1,i} &= \Phi_{k+1}^T \boldsymbol{M}_0^{-1} \Phi_i - \sum_{j=1}^{k} \frac{a_{j,k+1} a_{j,i}}{1 + a_{j,j}} \\
&= \Phi_i^T \boldsymbol{M}_0^{-1} \Phi_{k+1} - \sum_{j=1}^{i-1} \frac{a_{j,k+1} a_{j,i}}{1 + a_{j,j}} - \sum_{j=i}^{k} \frac{a_{j,k+1} a_{j,i}}{1 + a_{j,j}} \\
&= \frac{a_{i,k+1}}{1 + a_{i,i}} - \sum_{j=i+1}^{k} \frac{a_{j,k+1} a_{j,i}}{1 + a_{j,j}} \qquad (31)
\end{aligned}$$

While for $\boldsymbol{B}$, by using (24), b_{k+1} can be computed

$$\begin{aligned}
b_{k+1} &= \boldsymbol{y}^T \boldsymbol{M}_k^{-1} \Phi_{k+1} \\
&= \boldsymbol{y}^T (\boldsymbol{M}_0^{-1} - \sum_{j=1}^{k} \frac{\boldsymbol{M}_{j-1}^{-1} \Phi_j \Phi_j^T \boldsymbol{M}_{j-1}^{-1}}{1 + \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_j}) \Phi_{k+1} \\
&= \boldsymbol{y}^T \boldsymbol{M}_0^{-1} \Phi_{k+1} - \sum_{j=1}^{k} \frac{\boldsymbol{y}^T \boldsymbol{M}_{j-1}^{-1} \Phi_j \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_k}{1 + \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_j} \\
&= \boldsymbol{y}^T \boldsymbol{M}_0^{-1} \Phi_{k+1} - \sum_{j=1}^{k} \frac{a_{j,k+1} b_j}{1 + a_{j,j}} \qquad (32)
\end{aligned}$$

Then, substituting the definition of $a_{k+1,i}$ and b_{k+1} into (27), the net contribution of a new feature to the cost function can be calculated as

$$\begin{aligned}
\Delta \boldsymbol{J}_{k+1} &= \boldsymbol{y}^T \frac{\boldsymbol{M}_k^{-1} \Phi_{k+1} \Phi_{k+1}^T \boldsymbol{M}_k^{-1}}{1 + \Phi_{k+1}^T \boldsymbol{M}_k^{-1} \Phi_{k+1}} \boldsymbol{y} \\
&= \frac{(b_{k+1})^2}{1 + a_{k+1,k+1}} \qquad (33)
\end{aligned}$$

By computing the net contribution of each candidate feature to the cost function using equation (33), the best feature can be selected one by one.

After m features are selected, the estimated linear parameters $\boldsymbol{w}$ is given by

$$\begin{aligned}
\hat{\boldsymbol{w}} &= \boldsymbol{\Phi}_m^T \boldsymbol{M}_m^{-1} \boldsymbol{y} \\
&= \begin{bmatrix} \Phi_1^T \\ \Phi_2^T \\ \vdots \\ \Phi_m^T \end{bmatrix} \boldsymbol{M}_m^{-1} \boldsymbol{y} = \begin{bmatrix} \Phi_1^T \boldsymbol{M}_m^{-1} \boldsymbol{y} \\ \Phi_2^T \boldsymbol{M}_m^{-1} \boldsymbol{y} \\ \vdots \\ \Phi_m^T \boldsymbol{M}_m^{-1} \boldsymbol{y} \end{bmatrix} \qquad (34)
\end{aligned}$$

With (28), the estimate of each element $\hat{w}_{m,i}$, $i = 1, \cdots, m$ is computed as

$$\begin{aligned}
\hat{w}_{m,i} &= \Phi_i^T \boldsymbol{M}_m^{-1} \boldsymbol{y} \\
&= \Phi_i^T (\boldsymbol{M}_0^{-1} - \sum_{j=1}^{m} \frac{\boldsymbol{M}_{j-1}^{-1} \Phi_j \Phi_j^T \boldsymbol{M}_{j-1}^{-1}}{1 + \Phi_j^T \boldsymbol{M}_{j-1}^{-1} \Phi_j}) \boldsymbol{y} \\
&= \Phi_i^T \boldsymbol{M}_0^{-1} \boldsymbol{y} - \sum_{j=1}^{m} \frac{a_{j,i} b_j}{1 + a_{j,j}} \\
&= \boldsymbol{y}^T \boldsymbol{M}_0^{-1} \Phi_i - \sum_{j=1}^{i-1} \frac{a_{j,i} b_j}{1 + a_{j,j}} - \sum_{j=i}^{m} \frac{a_{j,i} b_j}{1 + a_{j,j}} \\
&= \frac{b_i}{1 + a_{i,i}} - \sum_{j=i+1}^{m} \frac{a_{j,i} b_j}{1 + a_{j,j}} \qquad (35)
\end{aligned}$$

C. MODEL PARAMETERS

In the aforementioned learning machine framework, a number of parameters including the regularization parameter, the weights for the training errors, as well as the parameters in the mapping function are introduced. Though the introduced regularization parameter has been discussed in [31] and [38], there is still no effective way to optimize the parameter. Here, the regularization parameter is dynamically adjusted in the experiments. On the other hand, a sparse solution is achieved by selecting the mapping function while the sample information is retained. Further, based on [39], a weighting scheme is introduced to the training errors, where the samples closer to the next prediction time are given larger weights. That is, for 10 samples used to train model, 10 random weight values are generated within the range of (0,10). These weight values are sorted in the ascending order, the smallest is ranked as the first, and the largest is ranked the last. Then, to predict the voltage at time instant $k+1$, the sample at time instant k is given the largest weight, while the sample at $k-9$ is given the smallest weight.

Furthermore, the centers in the mapping functions which are directly selected from the training samples may not be the optimal ones. Thus the centers and the widths of the mapping functions are optimized using a computationally effective method such as the differential evolution (DE) algorithm in this paper. Defining a nonlinear vector $\boldsymbol{X} = [x_1, x_2]^T$ to represent the optimized centers and widths of the mapping functions where $x_i \in [x_{i,min}, x_{i,max}]$, $i = 1, 2$, the procedures of the DE algorithm used to optimize the parameters are given as follows.

Step 1 Initialization. The evolutionary parameters such as the upper/lower bounds $[\boldsymbol{X}_{min}, \ \boldsymbol{X}_{max}]$, the maximum number of generations T, the mutation parameter $F \in [0, 2]$, the population size N and the crossover factors $CR \in [0, 1]$ are initialized firstly. Then real-valued vectors are generated as the initial population. Hence, the i^{th} vector is expressed as

$$\boldsymbol{X}_{i,0} = [x_{i,0}^1, x_{i,0}^2]^T \tag{36}$$

Step 2 Evaluation. The net contribution of each individual solution is computed using (33)

Step 3 Mutation. The i^{th} mutated vector $\boldsymbol{V}_{i,t}$ at the t^{th} ($t = 0, \cdots, T$) generation is produced by three randomly selected individuals at the t^{th} ($t = 0, \cdots, T$) generation. Suppose 3 integers $(r_1, r_2, r_3 \in 1, \cdots, N$ and $r_1 \neq r_2 \neq r_3 \neq i)$ are selected randomly, the mutated vector is given as

$$\boldsymbol{V}_{i,t} = \begin{cases} \boldsymbol{X}_{r_1,t} + F(\boldsymbol{X}_{r_2,t} - \boldsymbol{X}_{r_3,t}), & \boldsymbol{V}_{i,t} \in [\boldsymbol{X}_{min}, \boldsymbol{X}_{max}] \\ \boldsymbol{X}_{min} + F(\boldsymbol{X}_{max} - \boldsymbol{X}_{min}), & otherwise \end{cases} \tag{37}$$

Step 4 Crossover. A portion $u_{i,t}^j$ of parameters in the existing target vector $\boldsymbol{U}_{i,t}$ at the t^{th} ($t = 0, \cdots, T$) generation is replaced by

$$u_{i,t}^j = \begin{cases} v_{i,t}^j & rand_j \leq CR \ or \ j = randn_i \\ x_{i,t}^j & rand_j > CR \ and \ j \neq randn_i \end{cases} \quad j = 1, 2 \tag{38}$$

where $rand_j \in [0, 1]$ and $randn_i \in 1, 2$, $v_{i,t}^j$ is an element of $V_{i,t}$.

Step 5 Selection. Comparing the cost values of $X_{i,t}$ and $U_{i,t}$, the better one will be selected into the next generation

$$\boldsymbol{X}_{i,t+1} = \begin{cases} \boldsymbol{U}_{i,t} & net(\boldsymbol{U}_{i,t}) > net(\boldsymbol{X}_{i,t}) \\ \boldsymbol{X}_{i,t} & net(\boldsymbol{U}_{i,t}) \leq net(\boldsymbol{X}_{i,t}) \end{cases} \quad i = 1, \cdots, N \tag{39}$$

Step 6 Iteration. Steps 2-6 are repeated until the evolution has reached T generations.

D. THE SPARSE LEARNING MACHINE CONSTRUCTION PROCEDURE AND COMPUTATIONAL COMPLEXITY ANALYSIS

Now given all the above preminaries, the whole procedure to produce the sparse learning machine is summarized as follows.

Step 1. Initialization:

a) Select suitable values for the regulation parameters c and the weight matrix Λ of errors based on experiments;
b) Set the max dimension n of the high-dimension feature space with the initial dimension $k = 0$, the size (ss) of the population, the initial population $([s_j; \Gamma_j^{-1}], \ j = 1, \cdots, ss)$ for initial candidate mapping functions $(\Phi_0 = [\Phi_1, \cdots, \Phi_{ss}])$ using the training samples;
c) Set the parameters of the DE algorithm.

Step 2. Forward learning machine construction:

a) Set $k = k+1$, at the k^{th} step, $1 \leq k \leq n$ step, calculate $a_{i,j}$ $(i = 1, \cdots, k)$ and b_j $(j = 1, \cdots, M)$ using (29), (30) and (32). Then compute the net contributions of all candidate solutions to the cost function using (33) generated by the initial mapping functions.
b) Optimize the mapping function using the DE algorithm and update the net contributions.
c) Select the feature with the maximum net contribution as the k^{th} column in the feature matrix.
d) If the desired number of dimension (n) is reached, or MSE satisfies the accuracy, go to Step 3. Otherwise, go back to 2(a).

Step 3. calculate the linear parameter using (35).

It is clear that solving the weighted LS-SVM is equivalent to computing the inverse of a full $N \times N$ matrix in (9), with the complexity $O(N^3)$. Here using the above proposed forward selection assisted method to construct the sparse learning machine, the computational complexity is reduced to $O(n \cdot N^2)$, where n is the number of selected features.

IV. BATTERY SOC ESTIMATION USING UKF METHOD

From section II, the real-time battery SOC estimation procedure has two parts. Firstly, a sparse learning machine model is built using the proposed algorithm in section III to capture the nonlinear behaviours of batteries. Then substituting the measurement model into the batteries state-space model, the batteries SOC can be estimated using the UKF method. From the (2) and (3), the state-space model for UKF is formulated as

$$\begin{cases} SOC_k = SOC_{k-1} - \eta \cdot \Delta t \cdot i_k / Q_n + s_k \\ = f(SOC_{k-1}, i_k) + s_k \\ V_k = \sum_{i=1}^m w_{m,j} \exp\{-\frac{1}{2}(u_k - uc_j)^T \Gamma_j^{-1}(u_k - uc_j)\} + v_k \\ = h(SOC_k, i_k) + v_k \end{cases} \tag{40}$$

where s_k and v_k represent the process and the measurement noises respectively. $f(\cdot)$ is the state equation which indicates the relationship of the state variable (SOC) and control variable (i). $h(\cdot)$ is the measurement equation identified by the proposed model.

Then the unscented Kalman Filter (UKF) is applied which is shown in Fig.1. The predicted SOC $(\tilde{soc}(k+1))$ at the time instant $k + 1$ is calculated by the state equation with $K_1 = \eta \cdot \Delta t / Q_n$. Furthermore, $\tilde{soc}(k+1)$ is updated as $\hat{soc}(k+1)$ using the proportion function with gain K. The details of the UKF method is described as follows

Step 1 Initialization. Calculate the initial states and the covariance matrix as

$$\hat{SOC}_0 = E[SOC_0] \tag{41}$$

$$P_0 = E[(SOC_0 - \hat{SOC}_0)^T(SOC_0 - \hat{SOC}_0)] \tag{42}$$

Step 2 Unscented Transformation.

1) In order to simplify the functions $f(\cdot)$ and $h(\cdot)$, the state matrix is extended as

$$S\hat{O}C^a_{k-1} = [S\hat{O}C^T_{k-1}, 0, 0] \quad (43)$$

$$P^a_{k-1} = diag(P_{k-1}, Q, R) \quad (44)$$

where Q and R is the covariance of the process noise and the measurement noise respectively. $S\hat{O}C^a_0 = [S\hat{O}C^T_0, 0, 0]$, $P^a_0 = diag(P_0, Q, R)$.

2) Compute the sigma points by

$$\begin{aligned}
\chi_{k-1,0} &= S\hat{O}C^a_{k-1} \\
\chi_{k-1,i} &= S\hat{O}C^a_{k-1} + (\sqrt{(L+\lambda)P^a_{k-1}})_i, \quad i = 1, \cdots, L \\
\chi_{k-1,i} &= S\hat{O}C^a_{k-1} - (\sqrt{(L+\lambda)P^a_{k-1}})_i, \quad i = L+1, \cdots, 2L
\end{aligned} \quad (45)$$

where $S\hat{O}C^a_{k-1}$ and P^a_{k-1} are the states and the covariance matrix at the time instant $k-1$ respectively. $(\sqrt{(L+\lambda)P^a_{k-1}})_i$ stands for the i^{th} row of the square-rooting matrix. L is the number of dimensions after the state-space is extended as formulated in (43) and (44), and $L = 3$ in this paper. $\lambda = \alpha^2(L+t) - L$ in which $t = 0$ represents the distribution of the sigma samples, while $\alpha \in [10^{-2}, 1]$ impacts the non-linear characteristic of the measurements.

3) Compute the weights ($c^{(m)}$) for the mean and ($c^{(c)}$) for the covariance as follows

$$\begin{cases}
c^{(m)}_0 = \lambda/(L+\lambda) \\
c^{(c)}_0 = \lambda/(L+\lambda) + (1 - \alpha^2 + \beta) \\
c^{(m)}_i = c^{(c)}_i = \lambda/2(L+\lambda) \qquad i = 1, \cdots, 2L
\end{cases} \quad (46)$$

where $\beta = [0, 2]$ is determined by the distribution of the states.

Step 3 Prediction. The predicted state and the corresponding state covariance are updated as

$$\chi_{k|k-1,i} = f(\chi_{k-1,i}, i_k) \quad (47)$$

$$S\tilde{O}C_{k|k-1} = \sum_{i=0}^{2L} c^{(m)}_i \chi_{k|k-1,i} \quad (48)$$

$$P_{k|k-1} = \sum_{i=0}^{2L} c^{(c)}_i (\chi_{k|k-1,i} - S\hat{O}C_{k|k-1})(\chi_{k|k-1,i} - S\hat{O}C_{k|k-1})^T \quad (49)$$

while the prior estimations of the measurements are obtained as

$$Z_{k|k-1,i} = h(\chi_{k|k-1,i}, i_k) \quad (50)$$

$$\tilde{V}_{k|k-1} = \sum_{i=0}^{2L} c^{(m)}_i Z_{k|k-1,i} \quad (51)$$

Step 4 Correction (Estimation). Based on the prediction of the voltage measurement, the state is modified as follows

$$P_{z,k} = \sum_{i=0}^{2L} (Z_{k|k-1,i} - \tilde{V}_{k|k-1})(Z_{k|k-1,i} - \tilde{V}_{k|k-1})^T \quad (52)$$

$$P_{xz,k} = \sum_{i=0}^{2L} c^{(c)}_i (\chi_{k|k-1,i} - S\tilde{O}C_{k|k-1})(Z_{k|k-1,i} - \tilde{V}_{k|k-1})^T \quad (53)$$

$$K = P_{xz,k} P^{-1}_{z,k} \quad (54)$$

$$S\hat{O}C_k = S\hat{O}C_{k|k-1} + K_k(V_k - \hat{V}_{k|k-1}) \quad (55)$$

V. EXPERIMENTAL RESULTS

To build the battery model using the proposed algorithm, a series of experiments were conducted on a 5-Ah LiFePo4 battery using a multi-function Arbin BT2000 battery test device and the test data were recorded by a host computer. The battery was placed in a temperature chamber with constant temperature of 25°C. The voltage and current were recorded every 1s until the measurements falls within 0.02% of the full scale range (FSR) at low power applications and 0.05% of the FSR at high power applications. The Federal Urban Drive Schedule (FUDS) was tested on the batteries as shown in [1].

Firstly, a normalization process is applied to the acquired data because the magnitudes for the current, SOC and voltage are quite different. Here, the max-min normalization method defined below is adopted

$$\boldsymbol{U} = \boldsymbol{U}_{min} + (\boldsymbol{U}_{max} - \boldsymbol{U}_{min}) \circ [(\boldsymbol{uu} - \boldsymbol{uu}_{min}) \oslash (\boldsymbol{uu}_{max} - \boldsymbol{uu}_{min})] \quad (56)$$

where $\boldsymbol{U}$ is the normalized matrix ranges from $\boldsymbol{U}_{min}$ to $\boldsymbol{U}_{max}$, and $\boldsymbol{uu}$ is the actual matrix. Accordingly, the $\boldsymbol{uu}_{min}$ and $\boldsymbol{uu}_{max}$ consist of the minimum value and the maximum value of each variable respectively. The operators $\circ$ and $\oslash$ denote the Hadamard product operator and the inverse operator of Hadamard product respectively.

Secondly, a series of tests were conducted to choose suitable regularization parameter and the weights. As indicated earlier, the weights for the errors are randomly generated and the difference between two consecutive weights is set around 0.5 and the maximum weights is set to 10. The maximal feature dimension is 4. Then for the DE algorithm the mutation parameter is set as $F_1 = 0.08$ for optimizing the centres and $F_2 = 0.8$ for the widths. The crossover factors $CR = 0.2$ and $CR = 0.6$ are applied for the centers and the widths respectively. The population size $n = 8$ and the maximum generation is set to $l = 20$. Considering that the battery model varies with time, the measurement equation is retrained when every 10 new training samples are collected. The training and validation performances using the different weights are given in Table 1.

In Table 1, the RMSE (root-mean-square error) is chosen as the model performance indicator. The test results are illustrated in Fig. 2. It is clear that the model is more accurate using the ascending weights which the absolute errors are almost within ±0.02. Especially for non-smooth ramp data points, the prediction results are more accurate than the cases when the same weights or the descending weights are applied to the errors.

TABLE 1. Performance of S-DE-WLM under different parameter distributions.

Weights	Descending	Same	Ascending
RMSE (Validation)	2.5%	2.19%	1.35%
MAX (Validation)	15.79%	11.72%	9.28%
Regularization	4.35	9.4	3.2

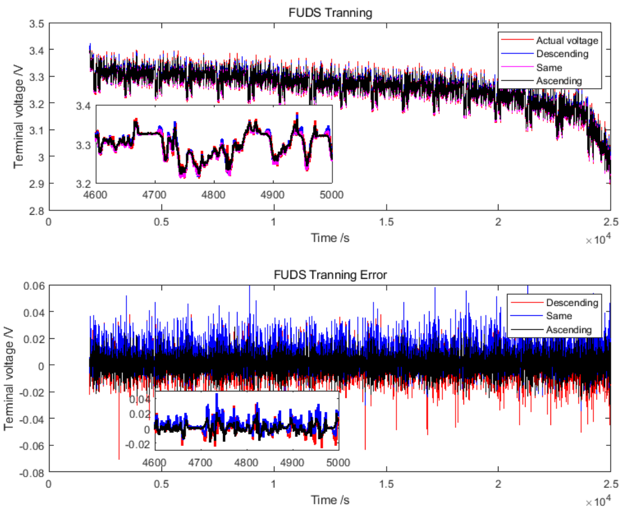

FIGURE 2. Training results of the FUDS process using different weights.

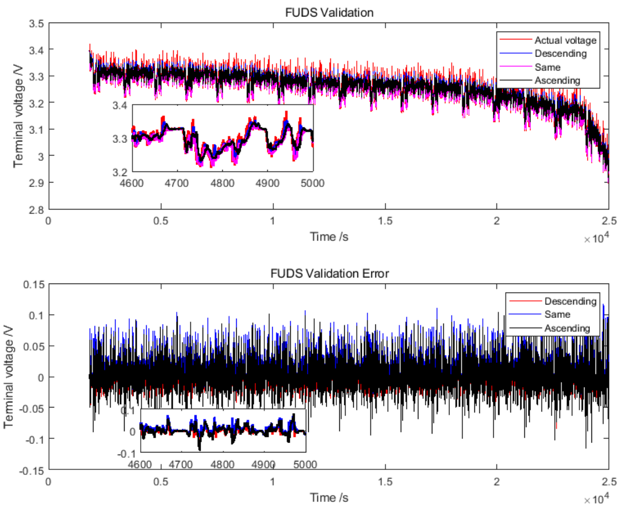

FIGURE 3. Validating results of the FUDS process using different weights.

Then, the proposed model is compared with the conventional LS-SVM model and the SVM model. The LS-SVM method is detailed in [40] and the RBF kernel function is tuned by both the k-fold cross validation (LSSVM in the following figures and tables) and the DE method (DE-LSSVM in the following figures and tables). The SVM model is also built for the comparison purpose where the parameters are optimized by both the k-fold cross validation (SVM in the following figures and tables) and the DE method (DE-SVM in the following figures and tables) [41], [42]. In order to verify the proposed method, two numbers of mapping functions

TABLE 2. Computation time of different algorithms.

	MAX time of consume(s)	Mean time of consume(s)
LSSVM	0.5823	0.5435
DE-LSSVM	0.4283	0.3991
SVM	0.5712	0.5491
DE-SVM	0.5431	0.5278
S-DE-WLM-N10	0.06802	0.06471
S-DE-WLM-N4	0.0221	0.0181

TABLE 3. Performance of different algorithms.

	RMSE (Validation)	MAX (Validation)
LSSVM	1.26%	56.97%
DE-LSSVM	1.63%	10.85%
SVM	2.2%	12.6%
DE-SVM	2.2%	11.72%
S-DE-WLM-N10	2.03%	9.6%
S-DE-WLM-N4	1.35%	9.28%

including 10 and 4 are tested in this paper which are named as 'S-DE-WLM-N10' and 'S-DE-WLM-N4' respectively. The training and validation computation time and the performance of the three methods are listed in Table 2 and Table 3.

The maximum computation time and the mean computation time shown in Table 2 are calculated on the whole cycle simulations using 23170 samples. The maximum time for all algorithms are less than the sampling time (1 second). However, the average computation time using the proposed new modelling approach is significantly less than the other two methods. This is due to the introduction of the fast recursive algorithm for constructing a sparse features space thus reducing the algorithm complexity. The root mean square error (RMSE) and the maximum absolute error (MAX) in Table 3 reveals that the proposed method can achieve better accuracy than the other method with much less time.

The training results and the validation results are detail illustrated in Fig 4 and Fig 5 respectively.

According to Fig 4, it is evident that the outputs of the resultant model produced by the proposed algorithm is similar as the conventional LS-SVM method. The modelling errors produced the proposed method are almost within $[-0.005, 0.005]$ which is similar to the LS-SVM method but has the similar performance as the DE-LSSVM, SVM, DE-SVM. While for predicting ramped non-smooth data points, the SVM method is the worst with the absolute error reaching 0.06. However, the validation results illustrated in Fig 5 show that the models built by the proposed algorithm, the DE-LSSVM method, the SVM model and the DE-SVM method all have better generalization performance than the LS-SVM model. The validation results obtained by the proposed method is more accurate where all the absolute errors are almost contained within a smaller range $[-0.02, 0.02]$ and only a few reached to 0.05, while the absolute errors of the LS-SVM model are sometime even greater than 0.5.

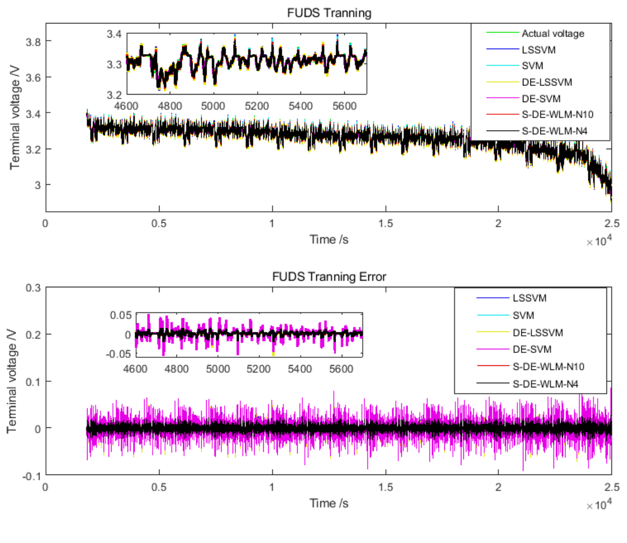

FIGURE 4. Training results of the FUDS process using different methods.

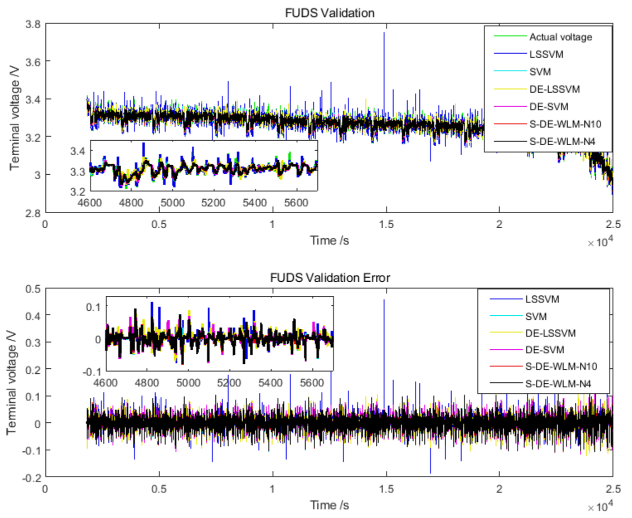

FIGURE 5. Validating results of the FUDS process using different methods.

The reason is that the model sparseness is lost using the LS-SVM method, leading to poor generalization performance. Though the DE method is adopted to improve this problem the computation time is still high. Accordingly, the S-DE-WLM improves the generalization capability while reduces the computational complexity.

Once the measurement equation developed by the proposed method is combined with the state equation, the UKF algorithm is then applied to estimate SOC. In the initialization process, t is set to zero given in section IV, while the measurement noise covariance, α and β are set as $1e{-}5, 5e{-}1$, 1 and 0 respectively. The estimated SOC which is illustrated in Fig 6 tracks well with its reference SOC which are calculated by the ampere-hour method ignoring the measurement noise in the lab environment based on the known initial SOC. Considering that the initial SOC in the actual working condition is unknown, the initial SOC is set to 0.7. It is clearly shown that the SOC estimation can quickly follow the actual SOC

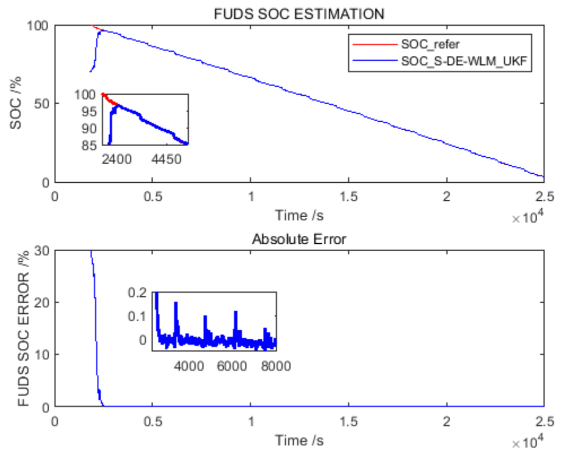

FIGURE 6. The estimated SOC of the FUDS process using the UKF.

values. Once the SOC estimation approaches to the reference SOC, the RMSE can reach $2.4882e{-}04$. The error of the SOC estimation is between -0.05% and $+0.2\%$ as shown in Fig 6, outperforming the results reported in the literature [2] which is between -0.79% and 0.94%. While in literature [43], the error is within $\pm 15\%$ using the Ampere hour counting method when it is applied in a practical environment. While the EKF achieves 5% error. Thus, the SOC estimation used the proposed method in this paper is more accurate.

VI. CONCLUSION

A novel sparse learning machine based on the LS-SVM formulation has been proposed for real-time SOC estimation in this paper. A mapping function instead of the kernel function is first applied to produce the nonlinear features space. In order to construct sparse learning machine, a fast selection method is used to select the mapping function, while the nonlinear parameters in the mapping functions are optimized by the DE algorithm. The proposed method is used to develop the SOC estimation model based on the FUDS test data sets. The experimental results confirm that the proposed method can build a more accurate battery model with excellent generalization performance in comparison with two other methods, namely SVM and LS-SVM models. Note that a small data set is used to train the model, and the proposed method is suitable for online implementations.

The obtained battery model is then used as the measurement equation, a highly accurate SOC estimation is achieved by using the UKF algorithm. The estimated SOC values are consistent with the referenced SOC with a 10^{-7} mean square error on the FUDS test data. It is demonstrated that the UKF method is more accurate than other KF variants on SOC estimations.

REFERENCES

[1] C. Zhang, K. Li, L. Pei, and C. Zhu, "An integrated approach for real-time model-based state-of-charge estimation of lithium-ion batteries," *J. Power Sources*, vol. 283, pp. 24–36, Jun. 2015.

[2] J. Meng, G. Luo, and F. Gao, "Lithium polymer battery state-of-charge estimation based on adaptive unscented Kalman filter and support vector machine," *IEEE Trans. Power Electron.*, vol. 31, no. 3, pp. 2226–2238, Mar. 2016.

[3] Z. Wei, C. Zou, F. Leng, B. H. Soong, and K.-J. Tseng, "Online model identification and state-of-charge estimate for lithium-ion battery with a recursive total least squares-based observer," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1336–1346, Feb. 2018.

[4] C. Zhang, K. Li, and J. Deng, "Real-time estimation of battery internal temperature based on a simplified thermoelectric model," *J. Power Sources*, vol. 302, pp. 146–154, Jan. 2016.

[5] M. U. Ali, A. Zafar, S. H. Nengroo, S. Hussain, M. J. Alvi, and H.-J. Kim, "Towards a smarter battery management system for electric vehicle applications: A critical review of lithium-ion battery state of charge estimation," *Energies*, vol. 12, no. 3, p. 446, Jan. 2019.

[6] Y.-M. Jeong, Y.-K. Cho, J.-H. Ahn, S.-H. Ryu, and B.-K. Lee, "Enhanced Coulomb counting method with adaptive SOC reset time for estimating OCV," in *Proc. IEEE Energy Convers. Congr. Expo. (ECCE)*, Sep. 2014, pp. 1313–1318.

[7] M. Petzl and M. A. Danzer, "Advancements in OCV measurement and analysis for lithium-ion batteries," *IEEE Trans. Energy Convers.*, vol. 28, no. 3, pp. 675–681, Sep. 2013.

[8] X. Tang, F. Gao, C. Zou, K. Yao, W. Hu, and T. Wik, "Load-responsive model switching estimation for state of charge of lithium-ion batteries," *Appl. Energy*, vol. 238, pp. 423–434, Mar. 2019.

[9] L. Zhang, K. Li, D. Du, M. Fei, and X. Li, "State-of-charge estimation of lithium batteries using compact RBF networks and AUKF," in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration. ICSEE, LSMS* (Communications in Computer and Information Science), vol. 763. Singapore: Springer, 2017.

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

[11] C. Huang, Z. Wang, Z. Zhao, L. Wang, C. S. Lai, and D. Wang, "Robustness evaluation of extended and unscented Kalman filter for battery state of charge estimation," *IEEE Access*, vol. 6, pp. 27617–27628, 2018.

[12] J. Xie, J. Ma, and K. Bai, "State-of-charge estimators considering temperature effect, hysteresis potential, and thermal evolution for $LiFePO_4$ batteries," *Int. J. Energy Res.*, vol. 42, no. 8, pp. 2710–2727, Jun. 2018.

[13] R. Xiong, H. He, F. Sun, and K. Zhao, "Evaluation on state of charge estimation of batteries with adaptive extended Kalman filter by experiment approach," *IEEE Trans. Veh. Technol.*, vol. 62, no. 1, pp. 108–117, Jan. 2013.

[14] T. Wang, S. Chen, H. Ren, and Y. Zhao, "Model-based unscented Kalman filter observer design for lithium-ion battery state of charge estimation," *Int. J. Energy Res.*, vol. 42, no. 4, pp. 1603–1614, Mar. 2018.

[15] H. Aung, K. Soon Low, and S. Ting Goh, "State-of-Charge estimation of lithium-ion battery using square root spherical unscented Kalman filter (SQRT-UKFST) in nanosatellite," *IEEE Trans. Power Electron.*, vol. 30, no. 9, pp. 4774–4783, Sep. 2015.

[16] M. Partovibakhsh and G. Liu, "An adaptive unscented Kalman filtering approach for online estimation of model parameters and state-of-charge of lithium-ion batteries for autonomous mobile robots," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 357–363, Jan. 2015.

[17] G. O. Sahinoglu, M. Pajovic, Z. Sahinoglu, Y. Wang, P. V. Orlik, and T. Wada, "Battery state-of-charge estimation based on regular/recurrent Gaussian process regression," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4311–4321, May 2018.

[18] R. Mall and J. A. K. Suykens, "Very sparse LSSVM reductions for large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1086–1097, May 2015.

[19] L. Chen and S. Zhou, "Sparse algorithm for robust LSSVM in primal space," *Neurocomputing*, vol. 275, pp. 2880–2891, Jan. 2018.

[20] Y.-H. Shao, C.-N. Li, M.-Z. Liu, Z. Wang, and N.-Y. Deng, "Sparse L-norm least squares support vector machine with feature selection," *Pattern Recognit.*, vol. 78, pp. 167–181, Jun. 2018.

[21] J. A. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Geneva, Switzerland, vol. 2, Jun. 2000, pp. 757–760.

[22] D. Geebelen, J. A. K. Suykens, and J. Vandewalle, "Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 682–688, Apr. 2012.

[23] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

[24] X. Zeng and X. Chen, "SMO-based pruning methods for sparse least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1541–1546, Nov. 2005.

[25] Y. Li, C. Lin, and W. Zhang, "Improved sparse least-squares support vector machine classifiers," *Neurocomputing*, vol. 69, nos. 13–15, pp. 1655–1658, Aug. 2006.

[26] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Statist. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.

[27] P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van Hamme, and J. A. K. Suykens, "Sparse conjugate directions pursuit with application to fixed-size kernel models," *Mach. Learn.*, vol. 85, nos. 1–2, pp. 109–148, Oct. 2011.

[28] R. Mall and J. A. Suykens, "Sparse reductions for fixed-size least squares support vector machines on large scale data," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining. Adv. Knowl. Discovery Data Mining (PAKDD)*, in Lecture Notes in Computer Science, vol. 7818. Berlin, Germany: Springer, 2013.

[29] S. Zhou, "Sparse LSSVM in primal using Cholesky factorization for large-scale problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 783–795, Apr. 2016.

[30] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 685–697, May 2007.

[31] J. Zhang, K. Li, G. W. Irwin, and W. Zhao, "A regression approach to LS-SVM and sparse realization based on fast subset selection," in *Proc. 10th World Congr. Intell. Control Automat. (WCICA)*, 2012, pp. 612–617.

[32] K. Li, J.-X. Peng, and G. W. Irwin, "A fast nonlinear model identification method," *IEEE Trans. Autom. Control*, vol. 50, no. 8, pp. 1211–1216, Aug. 2005.

[33] S.-L. Cheng and C. Hwang, "Optimal approximation of linear systems by a differential evolution algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 31, no. 6, pp. 698–707, Nov. 2001.

[34] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 842–844.

[35] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[36] G. Fumera and F. Roli, "Cost-sensitive learning in support vector machines," in *VIII Convegno Associazione Italiana per L'Intelligenza Artificiale*. North America: CiteSeerX, 2002.

[37] M. Lapin, M. Hein, and B. Schiele, "Learning using privileged information: SVM+ and weighted SVM," *Neural Netw.*, vol. 53, pp. 95–108, May 2014.

[38] T. S. Rögnvaldsson, "A simple trick for estimating the weight decay parameter," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700. Berlin, Germany: Springer, 2012.

[39] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," 2013, *arXiv:1302.6613*. [Online]. Available: http://arxiv.org/abs/1302.6613

[40] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[41] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 3:1–27:27, 2011.

[42] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," *J. Mach. Learn. Res.*, vol. 6, pp. 1889–1918, Dec. 2005.

[43] J. C. Á. Antón, P. J. G. Nieto, F. J. de Cos Juez, F. S. Lasheras, C. B. Viejo, and N. R. Gutiérrez, "Battery state-of-charge estimator using the MARS technique," *IEEE Trans. Power Electron.*, vol. 28, no. 8, pp. 3798–3805, Aug. 2013.

LI ZHANG received the B.Sc. and M.Sc. degrees from the Anhui University of Technology, Anhui, China, in 2009 and 2013, respectively. She is currently pursuing the Ph.D. degree in control theory and control engineering with Shanghai University.

From 2018 to 2019, she was a Visiting Ph.D. Student with University of Leeds, Leeds, U.K. Her current research interests include nonlinear system modeling and identification, machine learning algorithm, and power systems.

KANG LI (Senior Member, IEEE) received the B.Sc. degree in industrial automation from Xiangtan University, Hunan, China, in 1989, the M.Sc. degree in control theory and applications from the Harbin Institute of Technology, Harbin, China, in 1992, the Ph.D. degree in control theory and applications from Shanghai Jiaotong University, Shanghai, China, in 1995, and the D.Sc. degree in engineering from Queen's University Belfast, U.K., in 2015.

From 1995 to 2002, he worked at Shanghai Jiaotong University, the Delft University of Technology, and Queen's University Belfast, Belfast, U.K., as a Research Fellow. From 2002 to 2018, he was a Lecturer, in 2002, a Senior Lecturer, in 2007, a Reader, in 2009, and a Chair Professor, in 2011, with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast. He currently holds the Chair of smart energy systems at the University of Leeds, U.K. He is also a Visiting Professor with Queen's University Belfast, Shanghai Jiaotong University, Tianjin University, and Shanghai University. He has authored/coauthored over 150 journal publications and edited/co-edited 15 conference proceedings. His research interests include nonlinear system modeling, identification, and control, and artificial intelligence, with substantial applications to energy and power systems, smart grid, electric vehicles, railway systems, and energy management in energy intensive manufacturing processes.

Dr. Li is the Chair of the IEEE UKRI Control and Communication Ireland chapter. He was the Secretary of the IEEE U.K. & Ireland Section.

DAJUN DU received the B.Sc. degree in electrical engineering and automation and the M.Sc. degree in control theory and control engineering from Zhengzhou University, Zhengzhou, China, in 2002 and 2005, respectively, and the Ph.D. degree in control theory and control engineering from Shanghai University, Shanghai, China, in 2010.

From 2008 to 2009, he was a Visiting Ph.D. Student with Queen's University Belfast, Belfast, U.K., where he was a Research Fellow, from 2011 to 2012. He is currently a Professor with Shanghai University. His current research interests include neural networks, system modeling and identification, and networked control systems.

YUANJUN GUO received the B.Sc. degree in information engineering and the M.Sc. degree in optoelectronic engineering from Chongqing University, Chongqing, China, in 2008 and 2011, respectively, and the Ph.D. degree from the School of Electrical, Electronics and Computer Science, Queen's University Belfast (QUB), U.K., in 2015.

She is currently an Assistant Professor with the Shenzhen Institute of Science and Technology, Chinese Academy of Sciences, Shenzhen China. Her research interests include power big data analysis, artificial intelligence, fault diagnosis, and other applications in energy and power systems.

MINRUI FEI received the B.S. and M.S. degrees in industrial automation from the Shanghai University of Technology, Shanghai, China, in 1984 and 1992, respectively, and the Ph.D. degree in control theory and control engineering from Shanghai University, Shanghai, in 1997.

Since 1998, he has been a Full Professor with Shanghai University. His current research interests include networked control systems, intelligent control, complex system modeling, hybrid network systems, and field control systems.

Dr. Fei is the Chairman of Embedded Instrument and System SubSociety, the Standing Director of China Instrument and Control Society, the Chairman of Life System Modeling and Simulation Sub-Society, and the Vice-Chairman of Intelligent Control and Intelligent Management Sub-Society.

ZHILE YANG (Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. degree in control engineering from Shanghai University (SHU), in 2010 and 2013, respectively, and the Ph.D. degree from the School of Electrical, Electronics and Computer Science, Queen's University Belfast (QUB), U.K.

He worked as a Research Assistant with QUB. He is currently an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. He is the author or coauthor of more than 100 articles in peer-reviewed international journals and conferences, and an Active Reviewer for over 40 international journals. His research interests include artificial intelligence methods and their applications on smart grid and advanced manufacturing. He is the Founding Chair of IEEE QUB student branch and an Active Member of IEEE PES, CIS, and SMC societies.