MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Online Data-Driven Battery Voltage Prediction

Pajovic, M.; Sahinoglu, Z.; Wang, Y.; Orlik, P.V.; Wada, T.

TR2017-101 July 2017

Abstract

We consider in this article battery state of power (SoP) estimation, in particular, we propose two algorithms for predicting voltage corresponding to a future current profile that is known to be demanded by the battery load. The proposed algorithms belong to the class of data-driven methods and are based on the Gaussian Process Regression (GPR) framework. In comparison to conventional model-based approaches, datadriven approaches circumvent the issue of observability of SoP from measurements, especially pronounced in batteries with flat open circuit voltage (OCV) characteristic. In addition, the GPR framework admits accurate modeling of a fairly complicated battery dynamics using training data. Finally, the considered setup enables a relatively easy access to training data whenever the necessity for retraining arises, such as due to battery aging. The proposed algorithms aim to handle diverse battery operating conditions involving smooth and abruptly changing voltage/current measurements with both relatively small and large training datasets. The algorithms are tested on two such datasets, and the measured prediction performance and computation time verify their viability for real-time industrial use. We conclude with a number of possible directions for future research.

IEEE International Conference on Industrial Informatics (INDIN)

Online Data-Driven Battery Voltage Prediction

Milutin Pajovic, Zafer Sahinoglu, Yebin Wang, Philip V. Orlik, and Toshihiro Wada

***Abstract*—We consider in this article battery state of power (SoP) estimation, in particular, we propose two algorithms for predicting voltage corresponding to a future current profile that is known to be demanded by the battery load. The proposed algorithms belong to the class of data-driven methods and are based on the Gaussian Process Regression (GPR) framework. In comparison to conventional model-based approaches, data-driven approaches circumvent the issue of observability of SoP from measurements, especially pronounced in batteries with flat open circuit voltage (OCV) characteristic. In addition, the GPR framework admits accurate modeling of a fairly complicated battery dynamics using training data. Finally, the considered setup enables a relatively easy access to training data whenever the necessity for retraining arises, such as due to battery aging. The proposed algorithms aim to handle diverse battery operating conditions involving smooth and abruptly changing voltage/current measurements with both relatively small and large training datasets. The algorithms are tested on two such datasets, and the measured prediction performance and computation time verify their viability for real-time industrial use. We conclude with a number of possible directions for future research.**

***Index Terms*—Battery management system, Gaussian process regression, Lithium-ion battery, State of power, Voltage prediction.**

I. Introduction

Rechargeable batteries are nowadays widely deployed. For example, they are used as a backup power supply in case of power outages in hospitals, data centers, communication base stations, government premises, to store energy in solar panels, as well as to supply energy to electric vehicles, implant medical devices, consumer electronic devices, etc. Battery management system (BMS) manages the operation of a battery and ensures its safety and proper functioning. The BMS employs sensors which measure a variety of physical quantities related to a battery, such as voltage, current, temperature, internal resistance, etc. These measurements are supplied to the algorithms hosted in the BMS to estimate different measures quantifying the state of the battery. The BMS further manages the battery based on the algorithms' outputs.

Traditionally, the state of health (SoH) and state of charge (SoC) have been used to quantify the state of a battery. In some literature, the battery SoH quantifies its maximum capacity, i.e., the amount of charge it can hold, with respect to the rated (nominal) capacity. The SoH of a newly fabricated battery without defects is 100%. However, as the battery ages, its capability to store energy diminishes and the SoH decreases. On the other hand, the SoC quantifies battery's capacity at a given time with respect to its maximum capacity. The SoC is between 100%, when the battery is fully charged, and 0%, when the battery is fully discharged. A number of algorithms for battery SoH and/or SoC estimation have been proposed in the literature. Most of them are based on a battery equivalent circuit model [1], [2], or electrochemical model [3], and embed some form of Kalman filtering. On the other hand, data-driven SoH/SoC estimation algorithms started to emerge with the developments of machine learning and data analytics methods [4], [5]. In comparison to conventional models, the models revealed by the data-driven methods can, in general, describe a wider class of battery models, thereby providing better estimation performance. However, due to battery aging, the battery model parameters become inaccurate after a certain time period, and hence require retraining. A major difficulty with retraining is the lack of an easy access to the true SoH and SoC values over a range of operating conditions. This is especially an issue in the data-driven SoH/SoC estimation methods.

In addition to the SoH and SoC, the battery state of power (SoP) is also used as a metric to assess its functionality. Essentially, the SoP indicates how much power a battery can provide at a given time. Compared to the SoH and SoC, the SoP estimation has been less explored in the literature, although the SoP is a metric that can actually be measured. Model-based approaches for the estimation of SoP [6] and peak power [7], [8] prevail. There, the SoP is estimated from the SoH and SoC (as well as resistances) and therefore the estimation accuracy depends on the quality of the SoH/SoC estimators. However, achieving accurate SoH/SoC estimation is difficult for batteries with "flat"-OCV characteristics, such as LiFePO4, SCiB, NiMH, because of poor observability of the SoH and SoC from the measurements. The direct estimation of the SoP from the measurements can have stronger observability, nevertheless there are no models which are physically understandable.

In this paper, we study the problem of SoP estimation. In particular, we consider a problem of predicting battery voltage corresponding to a future current profile, assuming it is known and available to the BMS. This is a realistic setup in a number of applications, for example, in the uninterruptible power supply (UPS) systems. Assuming the present time is t, we propose two algorithms to predict voltage corresponding to future time $t+T$, assuming the current demand at $t+T$ is known. The proposed algorithms are data-driven for the following reasons. First, in comparison to the model-based SoP estimation, data-driven SoP estimation does not require

M. Pajovic, Z. Sahinoglu, Y. Wang, and P. V. Orlik are with Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA. {pajovic, zafer, yebinwang, porlik}@merl.com

T. Wada is with the Advanced Technology R&D Center, Mitsubishi Electric Corporation, 8-1-1, Tsukaguchi-honmachi, Amagasaki City, 661-8661, Japan. Wada.Toshihiro@bx.MitsubishiElectric.co.jp

SoH and SoC estimation and thus circumvents the issue of observability. Second, given that the true value of the quantity being estimated is available immediately after the time period T, this setting enables a relatively easy access to the training data at any time of operation. Thus, the training data is easily available when the utilized model needs to be retrained because of the battery aging, unlike in the SoH/SoC estimation. Finally, the models learned from data-driven methods are able to more accurately fit the actual battery dynamics than the conventional models. As an aside note, the proposed data-driven algorithms can be relatively easily cast to solve an alternative problem of predicting the current corresponding to future voltage demand. Also note that once the voltage is predicted, the battery SoP is given as the product between the predicted voltage and the current demand. The voltage prediction has already been studied in the literature using conventional battery models [9], [10]. However, we were unable to find a data-driven battery voltage prediction algorithm in the literature.

The proposed voltage prediction algorithms are based on the Gaussian process regression (GPR) framework [11], which is a Bayesian non-parametric technique suitable for modeling highly complicated input-output relations with relatively small training datasets. The GPR models unknown quantity as a Gaussian distributed random variable whose predicted mean is a point estimate, while the predicted variance is used to quantify the uncertainty in the point estimate. This uncertainly quantification, which naturally follows from the GPR framework, is one of the GPR advantages over other methods. The GPR framework has been successfully applied for battery SoH [12], [13], and SoC estimation [14], [15].

The rest of the paper is organized as follows. Section II presents two proposed voltage prediction algorithms. Section III validates the performance of the algorithms on simulated and experimental datasets. Section IV concludes the paper and provides possible directions for further research.

II. Voltage Prediction Algorithms

A. Problem statement

The essence of the problem considered in this paper is to predict battery voltage corresponding to future current demand, assumed known and available. An equivalent problem is to predict current given the future voltage demand. We focus on the former problem, but note that the proposed algorithms can also handle the latter problem.

We assume the measurements of battery's physical quantities (such as voltage, current and temperature) up to the present time are available and utilized for voltage prediction. Formally, the measurements of the voltage, current and temperature of a battery at discrete time t are denoted, respectively, V_t, I_t and T_t. The measurements are taken with the sampling period ΔT. The future current demand at M subsequent time instants $t+1, \ldots, t+M$, following the time instant t, is $I_{t+1}, \ldots, I_{t+M}$. Our goal is to predict voltage corresponding to the future current demand, i.e., to estimate $V_{t+1}, \ldots, V_{t+M}$ using the available data.

B. One-step voltage prediction

A simpler problem of predicting voltage corresponding to time $t+1$ from the measurements at times $t, \ldots, t-L$ and the current demand corresponding to time $t+1$, I_{t+1}, is considered first. We refer to L as the memory size since it specifies the number of previous measurements used for voltage prediction.

The measurements and future current are collected in an input vector $\mathbf{x}_t$, of length $3L+1$, formatted as

$$\mathbf{x}_t = \begin{bmatrix} I_{t+1} & V_t & I_t & T_t & \ldots & V_{t-L} & I_{t-L} & T_{t-L} \end{bmatrix}^T. \tag{1}$$

The output y_t from the one-step voltage prediction algorithm is the voltage corresponding to time $t+1$,

$$y_t = V_{t+1}. \tag{2}$$

The proposed prediction algorithms are based on the Gaussian process regression (GPR) framework [11]. The crux of the method is in modeling the joint probability density function of N outputs $y_{t_1}, y_{t_2}, \ldots, y_{t_N}$ as the joint Gaussian probability density,

$$p(y_{t_1}, y_{t_2}, \ldots, y_{t_N}) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \tag{3}$$

where the zero mean is assumed without loss of generality. The covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$ is given by

$$[\mathbf{\Sigma}]_{ij} = \kappa(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}), \tag{4}$$

where $\kappa(\mathbf{x}_{t_i}, \mathbf{x}_{t_j})$ is the kernel function measuring similarity between inputs $\mathbf{x}_{t_i}$ and $\mathbf{x}_{t_j}$. Intuitively, similar input vectors should yield similar outputs, i.e., voltage values.

The selection of a kernel from a number of standard kernels available in the literature, is driven by the specifics of a particular problem. Conceptually, one ought to choose a kernel that captures the insights in how output depends on the input. In that sense, a problem specific kernel can be designed. Alternatively, one may choose a kernel from several different candidates which yields the best performance.

An example of a kernel function commonly used in the GPR framework is the squared exponential (SE) kernel, defined as

$$\kappa_{\text{SE}}(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}) = \sigma_v^2 \exp(-\mathbf{x}_{t_i}^T \mathbf{D} \mathbf{x}_{t_j}), \tag{5}$$

where $\mathbf{D}$ is the diagonal matrix whose diagonal entries indicate the relevance of each entry in the input vector. Those entries, along with the kernel strength σ_v^2 are referred to as the hyper-parameters, collectively denoted $\mathcal{H}$.

The hyper-parameters are estimated from the training data. The training data $\mathcal{D}$ consists of N input-output pairs $(\mathbf{x}_{t_i}, y_{t_i})$, where $i = 1, \ldots, N$. The negative log-likelihood (NLL) function of the training data follows directly from the joint Gaussian distribution (3)

$$\text{NLL}(\mathcal{D}, \mathcal{H}) = \frac{1}{2} \log\det \mathbf{\Sigma} + \frac{1}{2} \mathbf{y}^T \mathbf{\Sigma}^{-1} \mathbf{y} + \frac{N}{2} \log(2\pi), \tag{6}$$

where $\mathbf{y} = \begin{bmatrix} y_{t_1} & y_{t_2} & \ldots & y_{t_N} \end{bmatrix}^T$ and the entries in $\mathbf{\Sigma}$ are evaluated using (4) for hyper-parameters $\mathcal{H}$. The maximum likelihood (ML) point estimate of the hyper-parameters is

obtained as the argument which minimizes the NLL,

$$\hat{\mathcal{H}} = \arg\min_{\mathcal{H}} \text{NLL}(\mathcal{D}, \mathcal{H}). \tag{7}$$

The optimization problem (7) can be solved using a gradient based method. We note that the gradient of the NLL (6) is given in a closed form for most available kernel functions. However, the NLL function is, in general, non-convex and the optimization (7) may therefore converge to a local minimum. To alleviate this issue, several optimization routines initialized differently may be run in parallel, and once all of them converge, the one resulting in the smallest NLL value yields the point estimate $\hat{\mathcal{H}}$. A flow chart of the training stage of the one-step voltage prediction is shown in Algorithm 1.

Algorithm 1 Training stage of the one-step predictor

1: **Input:** Training data $\mathcal{D} = \{(\mathbf{x}_{t_1}, y_{t_1}), \ldots, (\mathbf{x}_{t_N}, y_{t_N})\}$
2: **Input:** Kernel function $\kappa(.,.)$ parameterized by $\mathcal{H}$
3: Run optimization: $\hat{\mathcal{H}} = \arg\min_{\mathcal{H}} \text{NLL}(\mathcal{D}, \mathcal{H})$
4: Evaluate inverse covariance matrix $\mathbf{\Sigma}^{-1}$ for $\hat{\mathcal{H}}$ using (4)
5: **Output:** Learned hyper-parameters $\hat{\mathcal{H}}$
6: **Output:** Inverse covariance matrix $\mathbf{\Sigma}^{-1}$

The estimated hyper-parameters $\hat{\mathcal{H}}$ and training data $\mathcal{D}$ are used to infer the output y_t corresponding to the input $\mathbf{x}_t$ in the testing/operation stage. To do so, we first recall that the joint distribution of the training data outputs $y_{t_1}, \ldots, y_{t_N}$, and the target output y_t is Gaussian,

$$p(y_{t_1}, y_{t_2}, \ldots, y_{t_N}, y_t) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}'). \tag{8}$$

The covariance matrix $\mathbf{\Sigma}' \in \mathbb{R}^{(N+1)\times(N+1)}$ is given by

$$\mathbf{\Sigma}' = \begin{bmatrix} \mathbf{\Sigma} & \boldsymbol{\kappa} \\ \boldsymbol{\kappa}^T & \kappa_0 \end{bmatrix}. \tag{9}$$

Above, $\mathbf{\Sigma}$ is the covariance matrix corresponding to the training data and obtained from (4), $\boldsymbol{\kappa} \in \mathbb{R}^{N\times 1}$ is the vector of kernel functions evaluated at each input $\mathbf{x}_{t_i}$ from the training data $\mathcal{D}$, and the test input $\mathbf{x}_t$,

$$[\boldsymbol{\kappa}]_i = \kappa(\mathbf{x}_{t_i}, \mathbf{x}_t), i = 1, \ldots, N, \tag{10}$$

while κ_0 is given by

$$\kappa_0 = \kappa(\mathbf{x}_t, \mathbf{x}_t). \tag{11}$$

Note that all kernels κ in the covariance matrix $\mathbf{\Sigma}'$ are evaluated using the estimated hyper-parameters $\hat{\mathcal{H}}$, obtained in the training stage.

The unknown output y_t is inferred from the joint distribution (8) by conditioning on the known training data outputs $y_{t_1}, \ldots, y_{t_N}$. Since the joint distribution is Gaussian, the conditional distribution is also Gaussian,

$$p(y_t | y_{t_1}, \ldots, y_{t_N}) \sim \mathcal{N}(\mu_t, \sigma_t^2), \tag{12}$$

where the mean μ_t and variance σ_t^2 are given by

$$\mu_t = \boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1} \mathbf{y} \tag{13}$$

$$\sigma_t^2 = \kappa_0 - \boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1} \boldsymbol{\kappa}. \tag{14}$$

The mean value of the inferred distribution is the point estimate of the predicted voltage,

$$\hat{V}_{t+1} = \mu_t, \tag{15}$$

while the variance represents prediction uncertainly and can be used, for example, to specify the corresponding 95% confidence interval,

$$\left[\mu_t - 1.96\sigma_t, \mu_t + 1.96\sigma_t\right]. \tag{16}$$

A flow chart of the one-step voltage prediction in the operational stage is shown in Algorithm 2.

Algorithm 2 Operational stage of the one-step predictor

1: **Input:** Training data $\mathcal{D} = \{(\mathbf{x}_{t_1}, y_{t_1}), \ldots, (\mathbf{x}_{t_N}, y_{t_N})\}$
2: **Input:** Kernel function $\kappa(.,.)$ and hyper-parameters $\hat{\mathcal{H}}$
3: **Input:** Inverse covariance matrix $\mathbf{\Sigma}^{-1}$
4: **Input:** Measurements $V_t, I_t, T_t, \ldots, V_{t-L}, I_{t-L}, T_{t-L}$
5: **Input:** Future current I_{t+1}
6: $\mathbf{y} = \begin{bmatrix} y_{t_1} & y_{t_2} & \cdots & y_{t_N} \end{bmatrix}^T$
7: $\mathbf{x}_t = \begin{bmatrix} I_{t+1} & V_t & I_t & T_t & \ldots & V_{t-L} & I_{t-L} & T_{t-L} \end{bmatrix}^T$
8: $[\boldsymbol{\kappa}]_i = \kappa(\mathbf{x}_{t_i}, \mathbf{x}_t), i = 1, \ldots, N$
9: $\kappa_0 = \kappa(\mathbf{x}_t, \mathbf{x}_t)$
10: $\mu_t = \boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1} \mathbf{y}$
11: $\sigma_t^2 = \kappa_0 - \boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1} \boldsymbol{\kappa}$
12: **Output:** Predicted mean μ_t and variance σ_t^2

As a final remark, the extension of the presented development to the case of a non-zero mean vector in (3) is relatively straightforward. In the stationary case, the non-zero mean is estimated as the sample mean of the training data outputs. The estimated mean is then subtracted from the training data outputs and, upon performing all processing steps of the training and testing stage, added to (13) to yield the final voltage estimate.

In the following, we present two algorithms for multi-step voltage prediction.

C. Parallel Multi-Step Voltage Prediction (P-MSVP)

The P-MSVP consists of M parallel branches, indexed by $m = 1, \ldots, M$, where each branch m performs a separate m-step voltage prediction. Thus, the basic routine of the P-MSVP is the m-step voltage prediction, which is a generalization of the one-step voltage prediction, described in Section II-B.

The input vector of the m-step voltage prediction routine at some discrete time t, $\mathbf{x}_t^{(m)}$, is formatted by concatenating the future current demand at $t+m$, and the measurements of the voltage, current and temperature at $t, t-m, \ldots, t-mL$,

$$\mathbf{x}_t^{(m)} = \begin{bmatrix} I_{t+m} & \mathbf{V}_t^{(m)} & \mathbf{I}_t^{(m)} & \mathbf{T}_t^{(m)} \end{bmatrix}^T, \tag{17}$$

where

$$\begin{aligned}\mathbf{V}_t^{(m)} &= \left[\begin{array}{cccc} V_t & V_{t-m} & \dots & V_{t-mL}\end{array}\right]^T \\ \mathbf{I}_t^{(m)} &= \left[\begin{array}{cccc} I_t & I_{t-m} & \dots & I_{t-mL}\end{array}\right]^T \\ \mathbf{T}_t &= \left[\begin{array}{cccc} T_t & T_{t-m} & \dots & T_{t-mL}\end{array}\right]^T\end{aligned}$$

The target output is the voltage corresponding to time $t+m$,

$$y_t^{(m)} = V_{t+m}. \tag{18}$$

The training stage of the m-step voltage prediction routine learns the hyper-parameters $\mathcal{H}^{(m)}$ corresponding to the utilized kernel function by minimizing the negative log-likelihood (NLL) of the training data outputs, where the training data $\mathcal{D}^{(m)}$ consists of $N^{(m)}$ input-output pairs, $(\mathbf{x}_{t_i}^{(m)}, y_{t_i}^{(m)})$, $i = 1, \dots, N^{(m)}$. In the testing/operational stage, the mean $\mu_t^{(m)}$ and variance $\sigma_t^{2(m)}$ of the inferred Gaussian distribution of the voltage corresponding to the future current I_{t+m} are evaluated following the procedure described in Section II-B. The flow charts of the training and operational stages are shown in Algorithms 3 and 4, respectively. Note that the `for` loops in the flow charts are executed in parallel.

Algorithm 3 P-MSVP: Training stage

Input: Training data $\mathcal{D} = \{(\mathbf{x}_{t_1}, y_{t_1}), \dots, (\mathbf{x}_{t_N}, y_{t_N})\}$
Input: Kernel function $\kappa(.,.)$ parameterized by $\mathcal{H}$
for $m = 1 : M$ **do**
  Downsample $\mathcal{D}$ by m to obtain $\mathcal{D}^{(m)}$
  Run optimization: $\hat{\mathcal{H}}^{(m)} = \arg\min_{\mathcal{H}} \text{NLL}(\mathcal{D}^{(m)}, \mathcal{H})$
  Evaluate corresponding $\mathbf{\Sigma}_m^{-1}$ using (4)
end for
Outputs: $\hat{\mathcal{H}}^{(m)}$, $m = 1, \dots, M$
Outputs: $\mathbf{\Sigma}_m^{-1}$, $m = 1, \dots, M$

Algorithm 4 P-MSVP: Operational stage

Input: Training data $\mathcal{D} = \{(\mathbf{x}_{t_1}, y_{t_1}), \dots, (\mathbf{x}_{t_N}, y_{t_N})\}$
Input: Kernel function $\kappa(.,.)$ parameterized by $\mathcal{H}$
Input: Measurements $V_t, I_t, T_t, \dots,$
Input: Future current demand $I_{t+1}, \dots, I_{t+M}$
for $m = 1 : M$ **do**
  Downsample $\mathcal{D}$ by m to obtain $\mathcal{D}^{(m)}$
  Format $\mathbf{x}_t^{(m)}$ as in (17)
  $[\boldsymbol{\kappa}]_i = \kappa(\mathbf{x}_t^{(m)}, \mathbf{x}_{t_i}), i = 1, \dots, |\mathcal{D}^{(m)}|$, $\mathbf{x}_{t_i} \in \mathcal{D}^{(m)}$
  $\kappa_0 = \kappa(\mathbf{x}_t^{(m)}, \mathbf{x}_t^{(m)})$
  $\mu_t^{(m)} = \boldsymbol{\kappa}^T \mathbf{\Sigma}_m^{-1} \mathbf{y}$
  $\sigma_t^{2,(m)} = \kappa_0 - \boldsymbol{\kappa}^T \mathbf{\Sigma}_m^{-1} \boldsymbol{\kappa}$
end for
Outputs: $\mu_t^{(m)}$, $\sigma_t^{2,(m)}$, $m = 1, \dots, M$

D. Recursive Multi-Step Voltage Prediction (R-MSVP)

The main idea behind the R-MSVP is to treat the predicted voltage corresponding to a certain time instant $t+m$ as the measured voltage when predicting the voltage corresponding to the following time instant $t+m+1$. More precisely, at some time instant t, the one-step voltage prediction yields $\mu_t^{(1)}$, evaluated using (13). To predict voltage corresponding to $t+2$, the point estimate of the predicted voltage corresponding to $t+1$, $\mu_t^{(1)}$, is treated as the measured voltage and used along with the measurements of the voltage, current and temperature at time instants $t, t-1, \dots, t-L$, as well as future current demands I_{t+1} and I_{t+2} in a one-step prediction algorithm to obtain $\mu_t^{(2)}$. The process continues in this spirit until all M voltage predictions are obtained. We note that one form of recursion in the SoC estimation using the GPR framework has been porposed in [16].

The basic routine of the R-MSVP is the one-step prediction algorithm, described in Section II-B. The training stage of the R-MSVP algorithm is the same as in Section II-B and consists of learning the hyper-parameters of the kernel function by minimizing the negative log-likelihood of the training data $\mathcal{D}$, consisting of N input-output pairs $(\mathbf{x}_{t_i}, y_{t_i}), i = 1, \dots, N$. The estimated hyper-parameters are used for voltage prediction over the whole prediction time horizon.

In the testing/operational stage, the R-MSVP at time t serially predicts voltages corresponding to $t+1, \dots, t+M$. The voltage corresponding to time instant $t+m$ is predicted using the one-step prediction algorithm where the input is formatted as

$$\mathbf{x}_t^{(m)} = \left[\begin{array}{cccc} I_{t+m} & \mathbf{V}_t^{(m)} & \mathbf{I}_t^{(m)} & \mathbf{T}_t\end{array}\right]^T, \tag{19}$$

with

$$\begin{aligned}\mathbf{V}_t^{(m)} &= \left[\begin{array}{cccccc} \mu_t^{(m-1)} & \dots & \mu_t^{(1)} & V_t & \dots & V_{t-L+m-1}\end{array}\right]^T \\ \mathbf{I}_t^{(m)} &= \left[\begin{array}{cccccc} I_{t+m-1} & \dots & I_{t+1} & I_t & \dots & I_{t-L+m-1}\end{array}\right]^T \\ \mathbf{T}_t &= \left[\begin{array}{cccc} T_t & T_{t-1} & \dots & T_{t-L}\end{array}\right]^T\end{aligned}$$

and where $\mu_t^{(p)}$ is the mean of the inferred Gaussian distribution of the predicted voltage corresponding to $t+p$. Note that the formatting in (19) holds when $L > m$, while the case when $L \leq m$ is handled in an analogous manner.

A flow chart of the operational stage of the R-MSVP is shown in Algorithm 5. Note that the flow chart of the training stage of the R-MSVP is the same as for the one-step prediction and is shown in Algorithm 2.

E. Discussion

1) Computational Complexity: The computational complexity of the one-step voltage prediction in the operational stage can be assessed from (13) and (14) noting that the mean μ_t is given as a linear combination of the training data outputs, with the coefficients given by the vector-matrix product $\boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1}$. Assuming the covariance matrix of the training data, $\mathbf{\Sigma}$, is inverted at the end of the training stage, the computational complexity of the one-step voltage predictor is $O(N^2)$ per prediction.

Since both P-MSVP and R-MSVP perform M one-step voltage predictions at any time instant t, the overall computational complexity in the operational stage of either algorithm

Algorithm 5 Operational stage of the R-MSVP

Input: Training data $\mathcal{D} = \{(\mathbf{x}_{t_1}, y_{t_1}), \ldots, (\mathbf{x}_{t_N}, y_{t_N})\}$
Input: Kernel function $\kappa(.,.)$ and hyper-parameters $\hat{\mathcal{H}}$
Input: Inverse covariance matrix $\mathbf{\Sigma}^{-1}$
Input: Measurements $V_t, I_t, T_t, \ldots, V_{t-L}, I_{t-L}, T_{t-L}$
Input: Future currents $I_{t+1}, \ldots, I_{t+M}$
$\mathbf{y} = \begin{bmatrix} y_{t_1} & y_{t_2} & \ldots & y_{t_N} \end{bmatrix}^T$
for $m = 1 : M$ **do**
  Format $\mathbf{x}_t^{(m)}$ as in (19)
  $[\boldsymbol{\kappa}]_i = \kappa(\mathbf{x}_t^{(m)}, \mathbf{x}_{t_i}), i = 1, \ldots, N$
  $\kappa_0 = \kappa(\mathbf{x}_t^{(m)}, \mathbf{x}_t^{(m)})$
  $\mu_t^{(m)} = \boldsymbol{\kappa}^T \mathbf{\Sigma}^{-1} \mathbf{y}$
end for
Output: Predicted voltages $\hat{V}_{t+m} = \mu_t^{(m)}$, $m = 1, \ldots, M$

is $O(MN^2)$. On the other hand, while R-MSVP relies on a single one-step predictor, the P-MSVP employs M separate one-step predictors. This implies that training the P-MSVP is M times more computationally demanding than training the R-MSVP.

While the P-MSVP and R-MSVP have the same computational complexity in the operational stage, the computational times needed to output all M voltage predictions are different because the former admits parallel implementation, while the later performs predictions serially. This implies that the P-MSVP is M times faster than the R-MSVP.

2) Online Retraining: Both proposed algorithms require training data to learn hyper-parameters of the utilized kernel function. However, the battery model changes due to aging and the hyper-parameters should be re-estimated after a certain time period. While this restriction poses significant challenges in the data-driven SoC and SoH estimation, this is not so of an issue in the battery voltage prediction. Namely, the current, voltage and temperature of a battery are continuously being measured and thus the training data is immediately available as soon as the voltage prediction error (also easily available) starts indicating that the need for retraining has arisen.

3) Possible Enhancements: The main disadvantage of the P-MSVP algorithm is that its each parallel branch effectively downsamples the incoming measurements, possibly introducing aliasing, which detrimentally impacts the performance. Therefore, this method may not be suitable for the scenarios where voltage and current exhibit abrupt changes. A possible approach to alleviate this issue is to incorporate anti-aliasing filtering in each parallel branch.

On the other hand, the R-MSVP can handle data with abrupt changes as long as the sampling period ΔT of the measurement system is appropriately selected. However, the R-MSVP relies on voltage predictions corresponding to $t+1, \ldots, t+m$ to predict voltage for $t+m+1$, implying that the estimation error may accumulate over a relatively long prediction time horizon. A possible way to overcome this problem is to feed back not only the predicted mean μ_t, but a certain number of samples (i.e., particles) taken from the predicted Gaussian distribution. This approach is currently under our active exploration.

III. Results and Discussion

The proposed algorithms are validated using simulated and experimental datasets. As a performance metric, we report the maximum relative prediction error (MRE), defined as

$$\text{MRE}(m) = \max_t \left| \frac{V_{t+m} - \hat{V}_{t+m}}{V_{t+m}} \right|. \tag{20}$$

Although the simulated and experimental datasets are quite different, we make no effort in optimizing the prediction performance with respect to kernel function, so that the same kernel is used in all tests for both algorithms. This kernel is given as the sum of two squared exponential kernels and a neural network kernel,

$$\kappa(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}) = \sum_{s=1}^{2} \sigma_s^2 \exp(-\mathbf{x}_{t_i}^T \mathbf{D}_s \mathbf{x}_{t_j}) \tag{21}$$
$$+\sigma_3^2 \arcsin \frac{\sigma_p^2 (1 + \mathbf{x}_{t_i}^T \mathbf{x}_{t_j})}{\sqrt{(1 + \sigma_p^2 + \sigma_p^2 \mathbf{x}_{t_i}^T \mathbf{x}_{t_i})(1 + \sigma_p^2 + \sigma_p^2 \mathbf{x}_{t_j}^T \mathbf{x}_{t_j})}},$$

where $\mathcal{H} = \{\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_p^2, \mathbf{D}_1, \mathbf{D}_2\}$ are the hyper-parameters. Being the sum of three elementary kernels, this kernel embodies a relatively rich space of nonlinear maps.

Since we are not aware of any data-driven voltage prediction algorithms in the literature, this part validates the performance of the proposed algorithms only.

A. Test on Simulated Data

The simulated data was generated with a battery model based on equivalent circuit model, including thermal equation. In particular, the simulated model consists of two RC circuits ($R_{d1} = 0.72$ mΩ, $C_{d1} = 2678.49$ F, $R_{d2} = 0.08$ mΩ, $C_{d2} = 2678.49$ F) and one resistance ($R_{sct} = 2.04$ mΩ), which are connected in series [17]. The simulated voltage, current and temperature are shown in Fig. 1. The sampling period is $\Delta T = 1$ sec. The first 10,000 data points are used for training, while the remaining data is used for testing.

The simulated battery data is used to test the accuracy of the P-MSVP in predicting voltages 10 to 20 seconds (steps) ahead. The P-MSVP predictor consists of 11 parallel branches, such that each branch corresponds to a particular m, where $m = 10, 11, \ldots, 20$ steps. The m-th branch downsamples the data by factor m so that the sampling period is $m\Delta T = m$ seconds. A one-step predictor in each branch is trained, i.e., the corresponding hyper-parameters are learned. Note that due to the downsampling, the number of data points used for training the predictors is different across branches. Namely, the m-th branch uses $\lfloor 10,000/m \rfloor$ data points for training. The predictors in each branch, however, have the same memory length $L = 2$. This relatively short memory length is selected because the measurements exhibit relatively smooth variations.

Once the hyper-parameters of the predictors across branches are estimated, the remaining portion of the data is used for

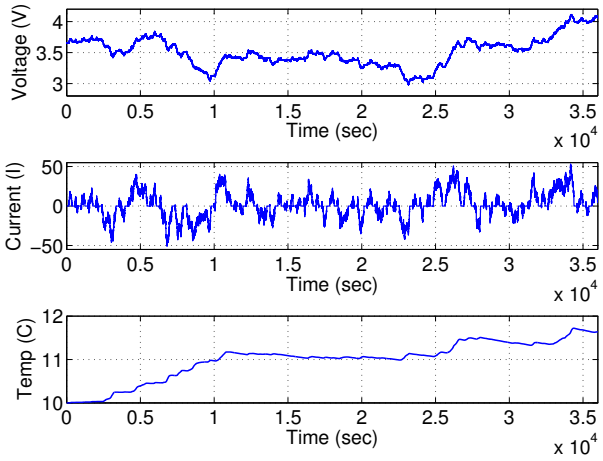

Fig. 1: Simulated battery data (top: voltage in V, middle: current in A, bottom: temperature in ^{o}C).

testing the predictors. As such, the predictor in the m-th branch at some time instant t takes the measurements of the voltage, current and temperature at t, $t-m$ and $t-2m$, as well as the future current at $t+m$ and performs one-step voltage prediction to yield the mean and variance of the inferred Gaussian distribution of the voltage corresponding to time instant $t+m$. The mean is the point estimate of the predicted voltage, while the variance is used to compute the 95% confidence interval.

The predicted voltage, 95% confidence interval and measured (i.e., simulated) voltage are shown in Fig. 2 for $m=20$ step ahead voltage prediction. The curves corresponding to the predicted and true voltage are almost indistinguishable. To give a better sense of the estimation error, the absolute and relative estimation errors corresponding to this case are shown in Fig. 3.

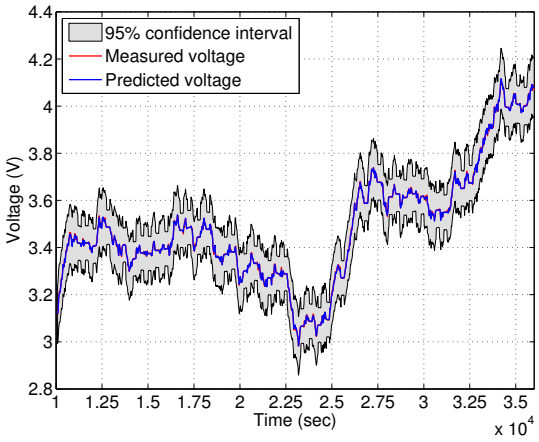

Fig. 2: P-MSVP on simulated data: 20 step voltage prediction.

The MRE performance corresponding to all branches is summarized in Fig. 4. As can be observed, the MRE's for

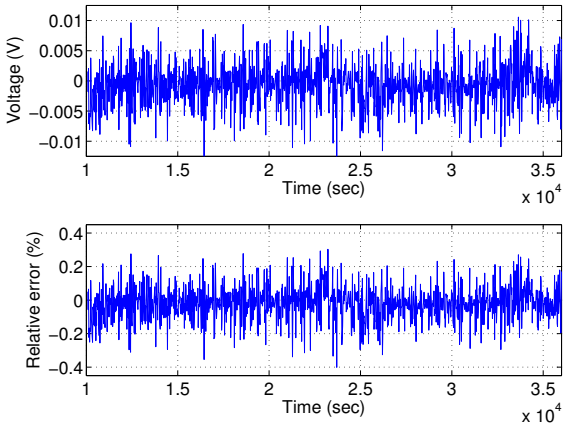

Fig. 3: Absolute (top panel) and relative (bottom panel) error in 20 step ahead voltage prediction on simulated data with the P-MSVP.

all considered steps m are similar and below 0.5%.

In general, we expect the R-MSVP to yield better performance than the P-MSVP because the R-MSVP does not involve downsampling, which eliminates a high frequency components in the time series. However, the MRE of 0.5% is a fairly good prediction performance and we thus do not test the R-MSVP on the simulated data. Instead, the R-MSVP is tested in the following part on a more challenging dataset.

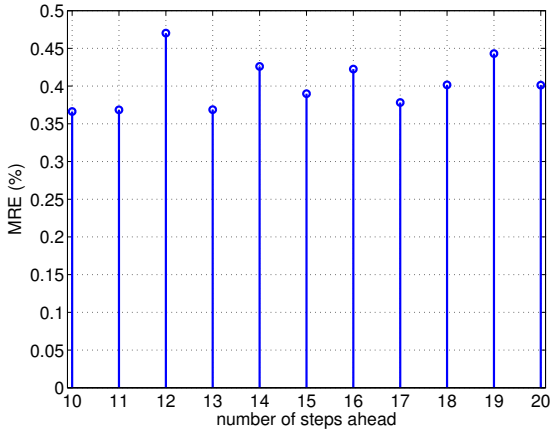

Fig. 4: MRE for P-MSVP with simulated data.

B. Test on Experimental Data

The experimental data was collected from a LiMn2O4/hardcarbon battery with a nominal capacity of 4.93 Ah in the Advanced Technology R&D Center, Mitsubishi Electric Corporation. Five consecutive cycles of charging and discharging with constant current were performed using a rechargeable battery test equipment produced by Fujitsu Telecom Networks. The obtained measurements of the

voltage, current and temperature are shown in Fig. 5. The sampling period of the measurements is $\Delta T = 1$ sec.

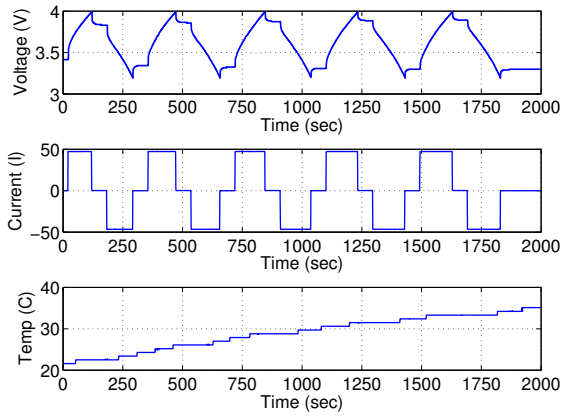

Fig. 5: Experimental data (top: voltage in V, middle: current in A, bottom: temperature in ^{o}C).

The training data consist of measurements corresponding to the first two charge-discharge cycles, which totals 721 data points. The R-MSVP algorithm is tested for voltage prediction $m = 1, 2, \ldots, 20$ sec (steps) ahead. The same memory size $L = 27$ (which optimizes the one-step predictor performance) is used for all steps m.

The predicted and measured (true) voltage corresponding to $m = 20$ are shown in Fig. 6. The corresponding absolute and relative estimation errors are shown in Fig. 7. As can be seen, the largest errors appear at the time instants at which the voltage exhibits abrupt changes. This emphasizes that the experimental data is a rather challenging one, despite the constant charge-discharge current and pseudo-periodic pattern of the measured current and voltage.

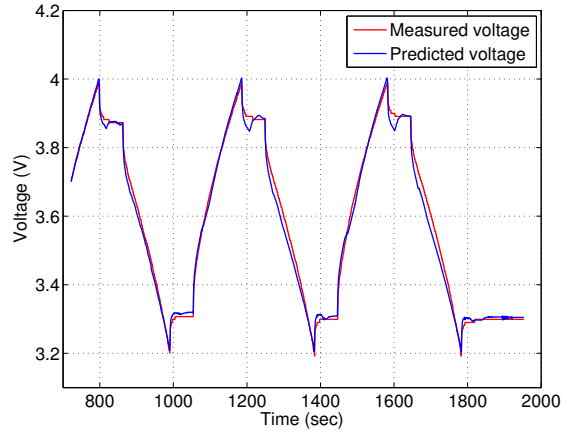

Fig. 6: R-MSVP on experimental data: 20 step ahead voltage prediction.

The MRE's achieved for different time horizon lengths m are summarized in Fig. 8. As can be seen, the MRE is below

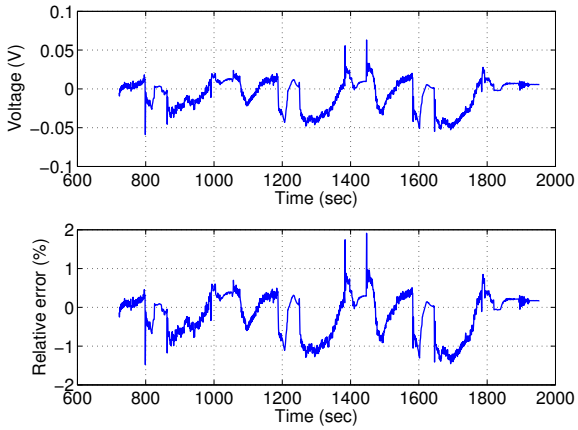

Fig. 7: Absolute (top panel) and relative (bottom panel) error in 20 step ahead voltage prediction on experimental data with the R-MSVP.

1.9% for all considered steps m. As with $m = 20$, the largest errors in all cases occur at the instants of abrupt changes in the current and voltage.

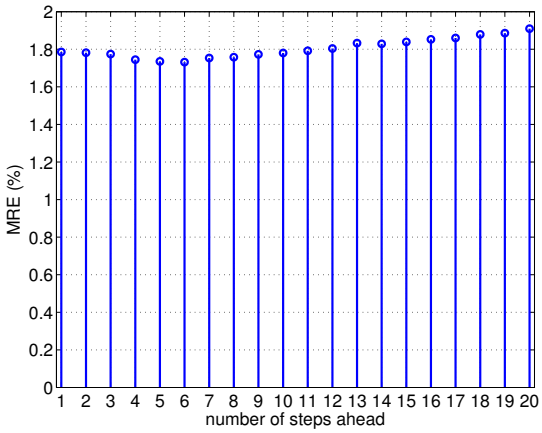

Fig. 8: MRE for R-MSVP with experimental data.

The computation time needed to make 20 voltage predictions in series (as the R-MSVP does) at each time instant has also been measured using the `tic` and `toc` routines in the `Matlab` implementation. The measured average time to deliver all 20 predictions is 0.25 sec. We recall that the R-MSVP is M times slower than the P-MSVP, implying that both algorithms are viable solutions with respect to computation time.

The P-MSVP has been also tested using the experimental data. It is confirmed that it underperforms the R-MSVP. More importantly, we note that the P-MSVP is less suitable in this case because the experimental data exhibits abrupt changes in voltage and current so that the downsampling causes aliasing. In addition, the effective training data size used for training one-step predictors in each parallel branch becomes fairly

small as m increases. For example, when $m = 10$, only 72 data points are available to train the corresponding one step predictor.

Finally, we emphasize few remarks. First, both algorithms in both datasets exhibit a fairly stable performance over the tested prediction time horizon of $M = 20$ seconds. The expectation is that the prediction performance will start to deteriorate as the time horizon M increases. The study of how rapidly this happens and at what M takes an effect is a possible future research. Second, the temperature measurements have been used in both tests. The impact of temperature on prediction performance needs to be closely studied as using only the measurements of the voltage and current might be a viable option. Alternatively, the proposed methods may be cast to predict battery temperature corresponding to future current and/or voltage demand. Finally, an interesting question for further investigation is how to select the memory size L. To give an idea, it ought to depend on the time constant of the battery dynamics (and, in turn, on the battery chemistry), and the expected voltage/current patterns (i.e., sampling period, presence or absence of the (pseudo)-periodicity).

IV. Conclusions and Future Research

The problem considered in this paper in concerned with estimating SoP of a battery. More specifically, two algorithms for predicting voltage corresponding to a future current demand have been proposed. Both methods are based on the main idea behind the Gaussian Process Regression (GPR) framework. The P-MSVP consists of multiple parallel one-step predictors acting upon appropriately downsampled measurements. The R-MSVP comprises of a one-step predictor which performs multiple one step ahead voltage predictions recursively. While the P-MSVP is better suited for larger training datasets with relatively smoothly varying voltage and/or current, the R-MSVP is suggested for smaller training datasets with abrupt changes in voltage and/or current. The considered voltage prediction setting is amenable to a relatively easy model retraining, the necessity of which may arise as a result of battery aging. Both methods can be recast to handle an alternative problem of predicting current corresponding to future voltage demand.

The proposed algorithms have been tested over future time horizon of 20 sec using simulated and experimentally measured battery data. The maximum relative prediction error (MRE) on the simulated data is below 0.5%. The experimental data with constant current charge-discharge is more challenging in the sense that voltage and current exhibit abrupt changes, while the training data size is relatively small. Still, MRE of below 1.9% is achieved over the whole 20 sec future time horizon. While both algorithms have similar computational complexities, the P-MSVP is faster since all voltage predictions over the considered time horizon are done in parallel. The measured average computation time of all 20 voltage predictions at each testing time instant with the slower R-MSVP is 0.25 sec. Along with the performance, this further confirms the viability of the proposed algorithms.

Possible directions for future research include incorporating anti-aliasing filters in the parallel branches of the P-MSVP algorithm with the goal to better handle abrupt current/voltage changes. Also, the R-MSVP may benefit from feeding back not only the predicted mean but also a certain number of samples (particles) from the predicted Gaussian distribution. Other items for future research include optimization with respect to the memory size L, validation on experimental data with dynamic charge-discharge profile, study of the impact of temperature on the prediction performance and comparison to other (if any) data-driven voltage prediction algorithms.

References

[1] A. E. Mejdoubi, A. Oukaour, H. Chaoui, H. Gualous, J. Sabor, and Y. Slamani, "State-of-charge and state-of-health Lithium-ion batteries' diagnosis according to surface temperature variation," *IEEE Trans. on Ind. Electron.*, vol. 63, no. 4, pp. 2391–2402, Apr. 2016.

[2] C. Zhang, L. Y. Wang, X. Li, W. Chen, G. G. Yin, and J. Jiang, "Robust and adaptive estimation of state of charge for Lithium-ion batteries," *IEEE Trans. on Ind. Electron.*, vol. 62, no. 8, pp. 4948–4957, Aug. 2015.

[3] S. Tang, Y. Wang, Z. Sahinoglu, T. Wada, S. Hara, and M. Krstic, "State-of-charge estimation of lithium-ion batteries via a coupled thermal-electrochemical model," *American Control Conference (ACC)*, pp. 5871–5877, Jul. 2015.

[4] J. N. Hu, J. J. Ju, H. B. Lin, X. P. Li, C. L. Jiang, X. H. Qiu, and W. S. Li, "State-of-charge estimation for battery management system using optimized support vector machine for regression," *Journal of Power Sources*, vol. 269, pp. 682–693, Dec. 2014.

[5] H. T. Lin, T. J. Liang, and S. M. Chen, "Estimation of battery state of health using probabilistic neural network," *IEEE Trans. on Ind. Inf.*, vol. 9, no. 2, pp. 679–685, May 2013.

[6] G. L. Plett, "High-performance battery-pack power estimation using a dynamic cell model," *IEEE Trans. on Veh. Tech.*, vol. 53, no. 5, pp. 1586–1593, Sept 2004.

[7] P. Malysz, J. Ye, R. Gu, H. Yang, and A. Emadi, "Battery state-of-power peak current calculation and verification using an asymmetric parameter equivalent circuit model," *IEEE Trans. on Vech. Tech.*, vol. 65, no. 6, pp. 4512–4522, June 2016.

[8] X. Hu, R. Xiong, and B. Egardt, "Model-based dynamic power assessment of lithium-ion batteries considering different operating conditions," *IEEE Trans. on Ind. Inf.*, vol. 10, no. 3, pp. 1948–1959, Aug 2014.

[9] D. N. Rakhmatov, "Battery voltage prediction for portable systems," *2005 IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 4098–4101, May 2005.

[10] Rakhmatov and Daler, "Battery voltage modeling for portable systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 2, pp. 29:1–29:36, Apr. 2009.

[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[12] D. Liu, J. Pang, J. Zhou, Y. Peng, and M. Pecht, "Prognostics for state of health estimation of lithium-ion batteries based on combination gaussian process functional regression," *Microelectronics Reliability*, vol. 53, pp. 832–839, April 2013.

[13] F. Li and J. Xu, "A new prognostics method for state of health estimation of lithium-ion batteries based on a mixture of gaussian process models and particle filter," *Microelectronics Reliability*, vol. 55, no. 7, pp. 1035–1045, June 2015.

[14] G. Ozcan, M. Pajovic, Z. Sahinoglu, Y. Wang, P. V. Orlik, and T. Wada, "Online state of charge estimation for lithium-ion batteries using gaussian process regression," in *Proc. of 42nd Annual Conf. of the IEEE Industrial Electronics Society (IECON)*, Oct. 2016, pp. 998–1003.

[15] ——, "Online battery state-of-charge estimation based on sparse gaussian process regression," in *Proc. of Power and Energy Society General Meeting (PESGM)*, July 2016.

[16] ——, "Battery state of charge estimation based on regular/recursive Gaussian process regression," *IEEE Trans. on Ind. Electron.*, submitted.

[17] G. L. Plett, *Battery Management Systems, Volume I: Battery Modeling*. Artech House, 2015.